5. Raster Graphics

5.1 Basics
5.2 Scan conversion
5.3 Text output
5.4 Raster and halftone images
5.5 Image processing
5.6 Compositing
5.7 Anti-aliasing

Course Structure

<table>
<thead>
<tr>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
</tr>
<tr>
<td>Human cognition and perception (aspects of human actions and perception)</td>
</tr>
<tr>
<td>Color perception and color models</td>
</tr>
<tr>
<td>Graphical output devices</td>
</tr>
<tr>
<td>Raster graphics</td>
</tr>
<tr>
<td>2D graphics</td>
</tr>
<tr>
<td>Interaction</td>
</tr>
<tr>
<td>GUI: foundation, basics</td>
</tr>
<tr>
<td>GUI: toolkits, development</td>
</tr>
<tr>
<td>Usability engineering</td>
</tr>
<tr>
<td>3D computer graphics: objects, transformations, visibility, illumination, shading</td>
</tr>
</tbody>
</table>
Overview

- **Basics**
  Structure and characteristics, coordinate rounding, BitBlt operations

- **Scan conversion of lines**
  Incremental approach, digital differential analyzer, Bresenham, line styles, line weights

- **Scan conversion of circles**
  First and second order DDA

- **Scan conversion of polygons**
  Active edge table algorithm

---

Overview

- **Text output**
  Font attributes, metrics

- **Halftone images**
  Picture types, color conversion, halftoning (dithering, Floyd-Steinberg error diffusion)

- **Image manipulation**
  Color tables and linear filters

- **Compositing**
  Transparency, RGBA, Porter/Duff operators

- **Anti-aliasing**
  Aliasing, pre/post filtering, super sampling
5. Raster Graphics

5.1 Basics

5.2 Scan conversion
5.3 Text output
5.4 Raster and halftone images
5.5 Image processing
5.6 Compositing
5.7 Anti-aliasing

Raster graphics pipeline

- video controller
  - D/A, Sync
- frame buffer
  - RGB
- raster subsystem
  - lines, polygons, text, pictures
- 2D raster graphics
- 3D graphics
  - geometry system
5.1 Basics

Characteristics of raster graphics

+ Image regeneration independent of the scene complexity
+ Filled surfaces with color variation per pixel (shading)
  - “Aliasing” effects
    (staircase effects, “jaggies”, Moiré effects, blinking)
  - Separation of graphical objects from associated pixels
    – Difficult selection of objects (picking)
    – Selective update

Frame buffer model

• Linear memory array
• Addressing as 2D array
  – Origin (start address)
    • Top left (X11, Java AWT)
    • Bottom left (OpenGL)
5.1 Basics

Mapping of 2D addresses to 1D memory
- Has to take the number of bits per pixel into account

\[
\text{setpixel (int } \times, \text{ int } y, \text{ value)} \\
\text{value} = \text{readpixel (int } \times, \text{ int } y)
\]

- Where is the pixel relative to the scan line? Equivalently to a shift of up to \(\frac{1}{2}\) or 1 pixel

```
OpenGL
Java
X11 / Foley
```

5.1 Basics

Difference between boundary (edge) and area filling

```
Drawpolygon ((2,2),(6,6))
Fillpolygon ((2,2),(6,6))
```
5.1 Basics

Floating-point coordinates
- Require consistent rounding to avoid holes and overlaps
  - E.g. segments \((x_a, x_b)\), and \((x_b, x_c)\)
  - First pixel \(x_a\) rounded up (unchanged if \(\text{frac}(x_a) = 0\))
  - Last pixel \(x_b\) rounded down (-1 if \(\text{frac}(x_b) = 0\))

Graphics state and raster operations
- Color for current drawing operations not as parameter, but state variable
  - Similarly for other attributes such as line style, filling style, etc.
  - Saves bandwidth
Graphics state and raster operations

- Shifting and/or copying of rectangular areas in the frame buffer (e.g. windows, icons, scrolling text)
- Address calculation, possibly in hardware
  - Block transfer (with DMA) BitBlt

\[
\text{copypixel} \left( x_{\text{min}}, y_{\text{min}}, x_{\text{len}}, y_{\text{len}}, x, y, \text{Rasterop} \right)
\]

\[
\begin{array}{ccccccc}
0 & 1 & 0 & 0 & 0 & 1 & 1
\end{array}
\begin{array}{ccccccc}
0 & 0 & 0 & 1 & 1 & 1 & 0
\end{array}
\begin{array}{c}
\text{replace}
\end{array}
\begin{array}{c}
\text{or}
\end{array}
\begin{array}{c}
\text{and}
\end{array}
\begin{array}{c}
\text{xor}
\end{array}
\]

repetition restores old values \( \Rightarrow \) cursor, highlighting

Thomas Schlegel
Graphical-Interactive Systems WS 09/10

---

Graphics state:
Example X11

Thomas Schlegel
Graphical-Interactive Systems WS 09/10
5. Raster Graphics

5.1 Basics

5.2 Scan conversion

5.3 Text output

5.4 Raster and halftone images

5.5 Image processing

5.6 Compositing

5.7 Anti-aliasing

---

5.2 Scan Conversion

Scan conversion of lines

- Simple for vertical, horizontal, and diagonal lines
- Notice: Different pixels per length → diagonal dimmer
- In general: Incremental approach:

\[ y = mx + b \quad \text{for } |m| \leq 1 \]

at every step: \( x = x + 1 \)

“sometimes”: \( y = y + 1 \)
5.2 Scan Conversion

Digital differential analyzer (DDA)

\[ x_{i+1} = x_i + 1 \]
\[ y_{i+1} = m(x_{i+1}) + b \]
\[ = m(x_i + 1) + b \]
\[ = mx_i + b + m \]
\[ = y_i + m \]

```c
int xa, xe, ya, ye;
int x, y, dx, dy;
float error, m;
dx = xe - xa;
dy = ye - ya;
m = float(dy)/float(dx);
error = 0.0;
for (x = xa; x < xe; x++) {
    setpixel(x, round(y));
y = y + m;
}
```

Consider rounding errors

expensive FP operations

Scan conversion of lines: Code transformation

```c
error = -0.5;
for (x = xa; x <= xe; x++) {
    setpixel(x, y, value);
    if (error >= 0.5) {
        y = y + 1;
        error = error - 1;
    }
}
```

```c
error = -0.5 * dx;
for (x = xa + 1; x <= xe; x++) {
    setpixel(x, y, value);
    error = error + dy;
    if (error >= 0) {
        y = y + 1;
        error = error - dx;
    }
}
```
5.2 Scan Conversion

Scan conversion of lines:
Code transformation

error = dy - 0.5 * dx;
setpixel(xa, y, value);
for (x = xa + 1; x < xe; x++) {
    if (error < 0) {
        error = error + dy;
    } else {
        y = y + 1;
        error = error + dy - dx;
    }
    setpixel(x, y, value);
}

error = 2 * dy - dx;
setpixel(xa, y, value);
for (x = xa + 1; x < xe; x++) {
    if (error < 0) {
        error += 2 * dy;
    } else {
        y = y + 1;
        error += 2 * (dy - dx);
    }
    setpixel(x, y, value);
}

- Consider rounding errors:
  - By scaling the error variable: integer algorithm possible

Bresenham (1963)

int x = xa, y = ya, dx, dy, error;
error = 2 * dy - dx;
setpixel(x, y);
for (x = xa + 1; x < xe; x++) {
    if (error < 0) {
        error += 2 * dy;
    } else {
        error += 2 * (dy - dx);
        y++;
    }
    setpixel(x, y);
}
5.2 Scan Conversion

Scan conversion of circles (DDA principle)

\[ x = r \cos \varphi \]
\[ y = r \sin \varphi \]
\[
\frac{dx}{d\varphi} = -r \sin \varphi = -y \\
\frac{dy}{d\varphi} = r \cos \varphi = x
\]

\[ dx = -y \varphi \\
\frac{dy}{d\varphi} = x \varphi \]

Better: second order

\[
\frac{d^2x}{d\varphi^2} = -x \\
\frac{d^2y}{d\varphi^2} = -y
\]

\[
\Delta^2 x_i = \Delta x_i - \Delta x_{i-1} = x_{i+1} - 2x_i + x_{i-1}
\]

\[
x_{i+1} = x_i \Delta \varphi^2 + 2x_i - x_{i-1} = x_i \left(2 - \Delta \varphi^2 \right) - x_{i-1}
\]

\[
y_{i+1} = y_i \left(2 - \Delta \varphi^2 \right) - y_{i-1}
\]

5.2 Scan Conversion

Line styles

- E.g. OpenGL LineStipple (factor, pattern):
  16 bit pattern

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Style</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00FF</td>
<td>------</td>
</tr>
<tr>
<td>0xAAAAA</td>
<td>------</td>
</tr>
</tbody>
</table>

factor 2

\[
\text{if (pattern \ [(i/factor) \mod 16]) setpixel (x, y)}
\]
5.2 Scan Conversion

Line thickness

- Pixel replication for
  - Columns for $|m| > 1$
  - Rows else
- Begin / end is too thin around 45° angle

5.2 Scan Conversion

Brushes

- Simulate thick pen
  - Brush
  - Ideally by a circle
  - Typical approximation by a square

- Fill boundary polygon
  - Different caps and joints
5.2 Scan Conversion

Brush vs. pixel replication

Line connections

Join Round        Join Miter        Join Bevel

Endpoint of both lines

Enlargement of joint
5.2 Scan Conversion

Line end points

- Notice: Filling the scan lines between Bresenham lines does not always result in internal points
  → Special filling algorithm
  - For arbitrary non-convex polygons several spans must be filled

for each scanline
  parity = 0
  for each x_i in sorted point of intersection list
    parity += 1
    if (odd (parity))
      fill from x_i to x_{i+1}
5.2 Scan Conversion

Fill conventions

Between Bresenham lines

Polygon fill only interior points

Slivers (triangles with acute angles) may result from scan conversion
5.2 Scan Conversion

Edge table algorithm

- Avoid overlaps between neighboring polygons (take integer positions into account)
  - Round as before: right edge not covered
  - Do not count $y_1$ integer vertices: top margin not covered
  - Do not count vertices from horizontal lines

- Use edge coherence
  - Calculate intersections of an edge with successive scan lines incrementally
  - Consider only edges that intersect the current scan line (active edges)
  - Preprocessing provides edge table (ET)

Edge table algorithm

- Assume all edges run from bottom to top:
  $$E_i : ((x_{0,i}, y_{0,i}), (x_{1,i}, y_{1,i})), y_{0,i} < y_{1,i}$$
- Ignore horizontal edges (where $y_{0,i} = y_{1,i}$)
- Create the edge table (ET) as a series of buckets, one bucket for each scanline covered by the polygon
  $$B_j : (y_j) \quad \text{min}(y_{0,i}) \leq j < \text{max}(y_{1,i})$$
**Edge table**

- In each bucket, store a list of edges that start at the corresponding scanline.
- For each edge, store
  - its ending scanline $y_{1,i}$ (i.e. $y_{\text{max}}$)
  - its horizontal starting position $x_{0,i}$ (i.e. $x$ value of $y_{\text{min}}$)
  - its inverse slope $\frac{1}{m_i} = \frac{x_{1,i} - x_{0,i}}{y_{1,i} - y_{0,i}}$

****

**Active Edge Table (AET)**

- The active edge table is a list of currently active edges $A_k: (y_{1,k}, x_k, 1/m_k)$
- For each edge, store
  - its ending scanline $y_{1,k}$
  - its current horizontal position $x_k$
  - its inverse slope $\frac{1}{m_k}$
Active Edge Table (AET) algorithm

\[
\text{AET} = () \\
y = 0 \\
\text{while } \text{AET} \neq () \text{ or } \text{ET} \neq \{\} \\
\quad \text{remove entries with } y_{1,k} = y \text{ from AET} \\
\quad \text{move ET entries from bucket } B_y \text{ to AET} \\
\quad \text{sort AET entries by their } x_k \text{ value} \\
\quad \text{fill every other span } (x_{0-1}, x_{2-3}, \ldots) \\
\quad y = y + 1 \\
\quad \text{for each active edge (in AET)} \\
\quad \quad x_k = x_k + 1/m_k
\]

- Special algorithms for triangles, rectangles, circles/arcs

5.2 Scan Conversion

AET: example

Scan line 9: \( y_{max} x \frac{1}{m} \) \( FA \) and \( EF \) are removed – pixel \( (9/2) \) is not drawn!

Scan line 10: \( x_k \) temporarily rounded here
5.2 Scan Conversion

Filling non-convex polygons

- **Even-odd**
  count intersections

- **Winding rule**
  count intersections in same direction

Outline of polygon to fill  EvenOddRule  WindingRule

Pattern with foreground and background

- **LineSolid**
  `---` `---` `---` `---` `---` `---` `---` `---`

- **LineOnOffDash**
  `●` `●` `●` `●` `●` `●` `●` `●` `●` `●` `●`

- **LineDoubleDash**
  `●` `●` `●` `●` `●` `●` `●` `●` `●` `●` `●`

- **FillSolid**
  `●` `●` `●` `●` `●` `●` `●` `●` `●` `●` `●`

- **FillTiled**
  `●` `●` `●` `●` `●` `●` `●` `●` `●` `●` `●`

- **FillStippled**
  `●` `●` `●` `●` `●` `●` `●` `●` `●` `●` `●`

- **FillOpaqueStippled**
  `●` `●` `●` `●` `●` `●` `●` `●` `●` `●` `●`
5. Raster Graphics

5.1 Basics
5.2 Scan conversion
5.3 Text output
5.4 Raster and halftone images
5.5 Image processing
5.6 Compositing
5.7 Anti-aliasing

5.3 Text Output

Overview
– Rasterize each letter into a small bitmap
– Copy these into the frame buffer

Font: collection of different text attributes
– Family: Courier, Helvetica, Times Roman
– Point size: 10 pt, 24 pt
– Weight: plain, bold, italic
– Slant, set width, spacing
– Manufacturer (foundry) Adobe, Bitstream, ...
– Character encoding schemes:
  • ISO8859-1 8-bit Latin 1 character set with ä, ö, ü, ß
  • Unicode 16-bit character set with Kanji

for printers 1 point ≈ \frac{1}{72} inch:
10 pt \times \frac{1}{72} \text{ in} \times 600 \text{ dpi} \approx 83 \text{ dots}
5.3 Text Output

**fonts:**
- Roman
- Bookman Old Style
- italic
  - skript

**character height:**
- That is text with different character height

**character direction:**
- 0 degree
- 45 degree
- 90 degree
- 180 degree

**character pitch:**
- normal pitch
- larger pitch

**character alignment:**
- left
- center
- right

**character zooming factor:**
- zoom factor = 1
- factor < 1
- factor > 1

---

Font metric

- ascent (XFontStruct)
- descent (XFontStruct)
- origin (x,y) pixel (typical)
- lbearing
- rbearing
- width (XCharStruct)
- ascent (XCharStruct)
- descent (XCharStruct)
- baseline
5.3 Text Output

Font metric
- For each font: height, ascent, descent
- For each character: width, ascent, descent
- Routines for the computation of the text extent
- Fonts are often stored in graphics board
  - LoadFont -> FontCache

proportional: aeiou
monospace: aeiou

5.3 Text Output

Scalable fonts
- Do not store font in many point sizes
  - generate from outline representation
    (often by B-splines)
  - Costly rasterization –
    often done by the font server
### 5.3 Text Output

#### Font management

![Diagram of font management system](image)

#### Font styles

<table>
<thead>
<tr>
<th>Foundry</th>
<th>Weight</th>
<th>Set Width</th>
<th>Points (in tenths of a point)</th>
<th>Average Width (in tenths of a pixel)</th>
<th>Vertical Resolution in dpi</th>
<th>Horizontal Resolution in dpi</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-adobe-courier-bold-o-normal--10-100-75-75-m-60-iso8859-1</code></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>-adobe-courier-medium-r-normal--0-[0 0 0 0]-200-200-p-0-iso8859-1</code></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>-adobe-courier-medium-i-normal--0-[12 0 0 8]-200-200-p-0-iso8859-1</code></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Three variations on the same scalable font: `pointsize=8`, `setsize=6, 8, and 12`
5.3 Text Output

Font matrices

Rotation

\[
\begin{bmatrix}
\cos \theta & \sin \theta \\
-\sin \theta & \cos \theta
\end{bmatrix}
\]

Glyph Rotation

Horizontal Mirroring

\[
\begin{bmatrix}
-1 & 0 \\
0 & 1
\end{bmatrix}
\]

Mirror

Vertical Mirroring

\[
\begin{bmatrix}
1 & 0 \\
0 & -1
\end{bmatrix}
\]

5.3 Text Output

Font matrices

Artificial Obliquing

Anamorphic Scaling
5.3 Text Output

Font matrices

\[
\begin{bmatrix}
10 & 0 \\
0 & 10
\end{bmatrix} \times \begin{bmatrix}
1 & 0 \\
\tan^{-1}(\omega) & 1
\end{bmatrix}
\]

5. Raster Graphics

5.1 Basics
5.2 Scan conversion
5.3 Text output
5.4 Raster and halftone images
5.5 Image processing
5.6 Compositing
5.7 Anti-aliasing
5.4 Raster and Halftone Images

Discretization of a 2D distribution of color/brightness (in a photo)

- Parameters and properties:
  - Spatial resolution
  - Color and luminance resolutions

<table>
<thead>
<tr>
<th>Name</th>
<th>Colors</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bilevel, black/white</td>
<td></td>
<td>1 bit/pixel</td>
</tr>
<tr>
<td>Grayscale, intensity</td>
<td></td>
<td>8 bit/pixels (12 bit/pixels in medicine)</td>
</tr>
<tr>
<td>Color/lookup table</td>
<td></td>
<td>8 bit/pixels</td>
</tr>
<tr>
<td>True color</td>
<td></td>
<td>24 bit/pixels</td>
</tr>
</tbody>
</table>

- Consider:
  - Scanning
  - Reconstruction
  - Sampling theorem
  - Filtering
  - Convolution
5.4 Raster and Halftone Images

Conversion of color and luminance
- True color → gray scale (by luminance equation)
- True color → 8 bit color
  - Uniform quantization (R: 3, G: 3, B: 2 bits)
  - Generate LUT
  - Popularity approach
    - Occupy the LUT with the 256 most frequent colors
  - Median cut algorithm
    - Each LUT entry for approx. same amount of pixels

Conversion of color and luminance
- Grayscale → binary picture
  - Threshold value
  - Halftoning / dithering
    - Exchange spatial resolution for intensity resolution
  - The eye averages intensity variations with high spatial resolution
    - Differently sized points (newspaper)
    - Differently filled patterns, with equally sized points
5.4 Raster and Halftone Images

Dithering

- "Clustered-dot ordered dither" for printing
  - E.g. 2*2 dither mask

![Dither Masks](image)

- In general:
  - \( n \times n \) pixels \( \rightarrow n^2 + 1 \) gray levels
  - Described by dither matrix
    - Set pixel \((i, j)\) with \( D^n(i, j) < \) intensity

\[
D^2 = \begin{pmatrix} 1 & 3 \\ 2 & 0 \end{pmatrix}
\]

\[
D^3 = \begin{pmatrix} 1 & 0 & 3 \\ 6 & 8 & 4 \\ 5 & 2 & 7 \end{pmatrix}
\]
5.4 Raster and Halftone Images

Dithering

Variable size

2x2 Dither

8x8 ordered dither

Floyd-Steinberg

5.4 Raster and Halftone Images

Dithering

Original

Grayscale

Ordered Dither

Error Diffusion
5.4 Raster and Halftone Images

Dithering

- Which pixel patterns?
  - Avoid horizontal and / or vertical lines
  - Grow from the inside outward
  - No isolated points
- Recursive construction

\[
D^{(n)} = \begin{pmatrix}
4D_0^2 + D_0 \cdot U_a^2 & 4D_0^2 + D_1 \cdot U_a^2 \\
4D_0^2 + D_{10} \cdot U_a^2 & 4D_0^2 + D_{11} \cdot U_a^2 \\
\end{pmatrix} \\
\text{with } U^n = \begin{pmatrix}
1 & \ldots & 1 \\
\vdots & \ddots & \vdots \\
1 & \ldots & 1 \\
\end{pmatrix}_{n \times n} = \text{unity matrix}
\]

- If source and destination picture have same resolution
  - Cover picture with \( D^n \)
  - Error diffusion (Floyd-Steinberg)
    - Distribute intensity errors on neighboring pixels
5.4 Raster and Halftone Images

Clustered-dot ordered dither with 45° alignment

- For a single region

- Applied to a larger image
5. Raster Graphics

5.1 Basics
5.2 Scan conversion
5.3 Text output
5.4 Raster and halftone images
5.5 Image processing
5.6 Compositing
5.7 Anti-aliasing

5.5 Image Processing

Color table transformations

E.g. contrast improvement with the help of a histogram

also nonlinear:

brighter
darker

windowing
5.5 Image Processing

Filtering

• Linear filter

\[ f(t) \ast h(t) = \int_{-\infty}^{+\infty} f(t') \cdot h(t-t') \, dt' \]

Discrete

\[ g_m = \sum_{i=-l}^{+l} f_{m-i} \cdot h_i \]

\[ \begin{array}{cccccc}
2 & 4 & 6 & 4 & 2 & 0 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 1 & \ast \\
\end{array} \]

\[ \begin{array}{cccc}
4 & 5 & 4 & 2 & 1 & 2 \\
\end{array} \]
5.5 Image Processing

Convolution

(a) $f(\tau)$

(b) $g(\tau)$

(c) $g(-\tau)$

(d) $g(x-\tau)$

(e) $f(\tau)g(x-\tau)$

(f) $f(\tau)g(x-\tau)$

(g) $f(x)^*g(x)$

5.5 Image Processing

Convolution

signal

sinc filter

applying the filter

filtered signal
5.5 Image Processing

Picture convolution in 2D

\[ g_{m,n} = \sum_{i=-L}^{+L} \sum_{j=-J}^{+J} f_{m-i,n-j} \cdot h_{ij} \]

- Low-pass filter
  - Smears out details

- High-pass filter

Filtering

Low Pass Filter

High Pass Filter
5.5 Image Processing

Morphological operators (non-linear filters)

- Erosion (min)
- Dilation (max)
- Open
- Close

5. Raster Graphics

5.1 Basics
5.2 Scan conversion
5.3 Text output
5.4 Raster and halftone images
5.5 Image processing
5.6 Compositing
5.7 Anti-aliasing
Combining pictures (α-blending, compositing)

- Example
  - Green glass (80% transparency) in front of blue background
  - Mixed color = 20% green + 80% blue
    \[
    = 0.2 \times (0, 1, 0) + 0.8 \times (0, 0, 1) = (0, 0.2, 0.8)
    \]
  - Opacity \( \alpha \), \( A = (1 - \text{transparency}) \)

- 4 components per pixel:
  - RGBA (e.g. 32 bits)
  - Default: \( A = 1 \), i.e., no transparency (opaque)

Combining pictures (α-blending, compositing)

- Other interpretation
  - 2 polygons \( A \) and \( B \) overlap at a pixel
  - Assumption: independent distribution of the two surfaces

\( \Rightarrow \) Compute mixed color according to the ratio of the covered surface

\[
\begin{align*}
\alpha_A \cdot \alpha_B \\
\alpha_A \cdot (1 - \alpha_B) \\
(1 - \alpha_A) \cdot (1 - \alpha_B) \\
\alpha_B \cdot (1 - \alpha_A)
\end{align*}
\]
### 5.6 Compositing

<table>
<thead>
<tr>
<th>operation</th>
<th>quadruple</th>
<th>diagram</th>
<th>$F_A$</th>
<th>$F_B$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>clear</strong></td>
<td>(0, 0, 0, 0)</td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$A$</td>
<td>(0, A, 0, A)</td>
<td></td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$B$</td>
<td>(0, 0, B, B)</td>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$A$ <em>over</em> $B$</td>
<td>(0, A, B, A)</td>
<td></td>
<td>1</td>
<td>$1 - \alpha_A$</td>
</tr>
<tr>
<td>$B$ <em>over</em> $A$</td>
<td>(0, A, B, B)</td>
<td></td>
<td>$1 - \alpha_B$</td>
<td>1</td>
</tr>
<tr>
<td>$A$ <em>in</em> $B$</td>
<td>(0, 0, 0, A)</td>
<td></td>
<td>$\alpha_B$</td>
<td>0</td>
</tr>
<tr>
<td>$B$ <em>in</em> $A$</td>
<td>(0, 0, 0, B)</td>
<td></td>
<td>0</td>
<td>$\alpha_A$</td>
</tr>
<tr>
<td>$A$ <em>held out by</em> $B$</td>
<td>(0, A, 0, 0)</td>
<td></td>
<td>$1 - \alpha_B$</td>
<td>0</td>
</tr>
<tr>
<td>$B$ <em>held out by</em> $A$</td>
<td>(0, 0, B, 0)</td>
<td></td>
<td>0</td>
<td>$1 - \alpha_A$</td>
</tr>
<tr>
<td>$A$ <em>atop</em> $B$</td>
<td>(0, 0, B, A)</td>
<td></td>
<td>$\alpha_B$</td>
<td>$1 - \alpha_A$</td>
</tr>
<tr>
<td>$B$ <em>atop</em> $A$</td>
<td>(0, A, 0, B)</td>
<td></td>
<td>$1 - \alpha_B$</td>
<td>$\alpha_A$</td>
</tr>
<tr>
<td>$A$ <em>xor</em> $B$</td>
<td>(0, A, B, 0)</td>
<td></td>
<td>$1 - \alpha_B$</td>
<td>$1 - \alpha_A$</td>
</tr>
</tbody>
</table>

Contribution to area $(0, A, B, AB)$

---

### 5.6 Compositing

12 compositing operators *(Porter, Duff 1984)*

- Pre-multiplying colors: $c_o = \alpha_o \ C_o$
- Resulting color: $c_o = F_A \alpha_A C_A + F_B \alpha_B C_B$
  $= F_A C_A + F_B C_B$

- Important: $A$ *over* $B = (\alpha_A R_A, \alpha_A G_A, \alpha_A B_A) + (1 - \alpha_A) (\alpha_B R_B, \alpha_B G_B, \alpha_B B_B)$
- Often no alpha-channels in the frame buffer (therefore pre-multiplied): transparent $C$ over framebuffer $C_{FB}$
  - $C_{FB} = \alpha C + (1-\alpha)C_{FB}$
- Other operators
  
  $$
  \begin{align*}
  \text{darken}(A, \varsigma) &= (\varsigma R, \varsigma G, \varsigma B, \alpha_A) \\
  \text{fade}(A, \delta) &= (\delta R, \delta G, \delta B, \delta \alpha_A) \\
  \text{blend}(A, B) &= \text{fade}(A, t) \text{ plus fade}(B, 1-t) \\
  \end{align*}
  $$
  
  $0 \leq \varsigma \leq 1$
  $0 \leq \delta \leq 1$
  $t = 1..0$
5.6 Compositing

**Screen-door transparency**
- Transparency without $\alpha$ blending

![Screen-door transparency diagram]

5.1 Basics
5.2 Scan conversion
5.3 Text output
5.4 Raster and halftone images
5.5 Image processing
5.6 Compositing

**5.7 Anti-aliasing**
5.7 Anti-Aliasing

- Discrete raster representation (local/temporally)
  - Staircase artifacts (jaggies)
  - Scanning problems (pixel flashing, car wheels)
- Anti-aliasing: smear intensity over several pixels (filtering) (needs gray values)
- Remove high frequencies before rasterization (low-pass filters): pre-filtering
- During the scan conversion
  e.g. Gupta-Sproull method for line rasterization
5.7 Anti-Aliasing

- Supersampling
  - Rasterize with higher resolution (e.g. $2^2$ subpixels)
    - Only few possibilities
      $\Rightarrow$ hardware LUT
  - Averaging the subpixels
  - Weighted averaging (weighted area sampling)
- Post filtering (subsequent smoothing)
  - E.g. Bartlett filter
    $\frac{1}{36} \begin{pmatrix} 1 & 2 & 2 & 1 \\ 2 & 4 & 4 & 2 \\ 2 & 4 & 4 & 2 \\ 1 & 2 & 2 & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 & 2 & 1 \\ 2 & 4 & 6 & 4 & 2 \\ 3 & 6 & 9 & 6 & 3 \\ 2 & 4 & 6 & 4 & 2 \\ 1 & 2 & 3 & 2 & 1 \end{pmatrix}$
References