3. Raster Algorithms

3.1 Raster Displays

- CRT
- LCD

(see Chapter 1 of the Notes)
3.2 Monitor Intensities & Gamma

- Compute intensity values by an illumination model
- Computed intensity values then are converted to allowable intensity levels for the particular graphics system in use
How should intensity levels be spaced?

- logarithmically, not linearly.

- Eye is sensitive to ratios of intensity levels rather than to their absolute values.

- ratio of successive intensities should be constant to get equal perceived brightness.
For each component of the RGB model:

- $I_0$: lowest attainable intensity (0.005 - 0.025)
- $n + 1$: number of intensity levels with equal perceived brightness

We should have:

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

with

$$r = \left(\frac{1}{I_0}\right)^{1/n}$$

and

$$I_n = rI_{n-1} = r^2I_{n-2} = \ldots = r^nI_0$$

from about 1/200 to 1/40 of the maximum intensity of 1.0.
Example:

\[ I_0 = \frac{1}{8}, \; n = 3, \; r = 2, \]
\[ I = \frac{1}{8}, \; \frac{1}{4}, \; \frac{1}{2}, \; 1 \]

\[ I_0 = \frac{1}{50}, \; n = 255, \; r = 1.0154595, \]
\[ I = 0.0200, \; 0.0203, \; 0.0206, \; 0.0209, \]
\[ 0.0213, \; 0.0216, \ldots, \; 0.9848, \; 1.0000 \]
How to display a desired intensity $I$?

1. Determine the nearest $I_j$
   
   $$j = \text{ROUND}(\log_r (I / I_0))$$

2. Calculate $I_j = r^j I_0$

3. Determine pixel value $V_j = \text{ROUND}((I_j / K)^{1/\gamma})$
   
   (Gamma correction of intensity) Why?

4. $V_j$ is placed in the appropriate pixel
Here is “Why?”

- Relationship between intensity of light output (displayed intensity $I$) and the number of electrons in the beam $N$:
  \[ I \propto N^\gamma \quad \text{with} \quad 2.2 \leq \gamma \leq 2.5 \]

- Relationship between the number of electrons in the beam and intensity value specified for the pixel:
  \[ N \propto V \]
Here is “Why?”

- Hence, the intensity value $I$ and the pixel value $V$ satisfy the following equation:

$$I = KV^\gamma$$

where $K$ is the maximum intensity
Here is “Why?”

- Hence, to display a particular intensity value $I$ for the pixel, the correct pixel value to produce this intensity is:

$$V = (I / K)^{1/\gamma}$$

or, simply

$$V = (I)^{1/\gamma}$$
How to find the value of $\gamma$?

Note that $I / K = V^\gamma$ or displayed intensity (between 0 and 1) $= V^\gamma$

If we can find a $V$ so that the displayed intensity $= 0.5$ then

$$\gamma = \ln 0.5 / \ln V$$
How to find such a $V$?

Display a checkerboard pattern next to a square of grey pixels with input $V$.

Adjust the value of $V$ until both sides match in average brightness.
In summary

To show a desired intensity $I$ at a particular pixel, the pixel value $V$ one should send to that pixel is:

$$V = I^{1/\gamma}$$

The so-called Gamma correction

References:
1. [www.graphics.cornell.edu/~westin/gamma/gamma.htm](http://www.graphics.cornell.edu/~westin/gamma/gamma.htm)
3.3 RGB Color

- color (light) is displayed by three primary lights: red, green, blue, in an **additive manner**

Red + green = yellow

![Diagram showing RGB color model]

Starts with darkness
3.3 RGB Color

- on the other hand, paints and crayons are generated using *subtractive color mixing*, with *primaries*: red, yellow, blue

Starts with light
3.3 RGB Color

- Individual contributions of each primary are added together to yield the result
- Most popular for CRT monitors

Colors are specified as:
Color = (r, g, b)
0 ≤ r, g, b ≤ 1
3.4 The Alpha Channel

- How to partially overwrite the contents of a pixel, such as in *compositing*
- Compositing is the process of combining separate image layers into a single picture
- A multi-bit gray-scale mask (called *alpha*) is maintained as a fourth *alpha channel in addition* to (r, g, b) color channels
3.4 The Alpha Channel

- *alpha channel is used to hold an opacity factor* (or, the fraction of the pixel covered by an opaque surface) for each pixel.

To composite a foreground color $c_f$ over background color $c_b$, and the fraction of the pixel covered by foreground is $\alpha$, use the formula

$$c = \alpha c_f + (1 - \alpha)c_b$$
3.5 Scan Converting Polygons
3.5 Scan Converting Polygons

Basic idea:
1. Compute \textit{x coordinates of intersection points} of current scan line with all edges
2. Sort intersection points by increasing \textit{x values}
3. Group intersection points by pairs
4. Fill in the pixels between each pair of intersection points on the current scan line
3.5 Scan Converting Polygons

Basic idea:
1. Compute *x coordinates of intersection points* of current scan line with all edges
2. Sort intersection points by increasing *x values*
3. Group intersection points by pairs
4. Fill in the pixels between each pair of intersection points on the current scan line

Costly operations; Would like to avoid these operations. How?
3.5 Scan Converting Polygons

Need to create a **bucket-sorted edge table (ET)** first:
- To determine which edges intersect current scan line
- To **efficiently compute intersection points** of these edges with the current scan line
- Vertical edges need to be shortened by 1 in y direction
3.5 Scan Converting Polygons
3.5 Scan Converting Polygons

- Scan line $y$: 
  - $(x_{\text{min}}, y)$
  - $(x_{\text{max}}, y_{\text{top}})$

- Polygons:
  - $e_1$, $e_2$, $e_3$, $e_4$, $e_5$, $e_6$

- Lines:
  - $y = 12$
  - $y = 7$
  - $y = 5$
  - $y = 1$

- Equations:
  - $y_{\text{top}}$
  - $x_{\text{min}}$
  - $\frac{1}{m}$

- Slope: $m$
3.5 Scan Converting Polygons

Also need to maintain an active-edge table (AET)

Purpose: keep track of the edges the current scan line intersects

How: when we move to a new scan line (bottom to top), new edges intersecting the new scan line are added into the AET, edges in AET which are no longer active (not intersected by the new scan line) are deleted
3.5 Scan Converting Polygons

Example:
In the previous example, when \( y = 2 \),

\[
\text{AET} \quad \rightarrow \quad e_1 \rightarrow e_6
\]

when \( y = 4 \),

\[
\text{AET} \quad \rightarrow \quad e_1 \rightarrow e_6
\]

when \( y = 8 \),

\[
\text{AET} \quad \rightarrow \quad e_2 \rightarrow e_5
\]
3.5 Scan Converting Polygons

When \( y = y_1 \)

AET \[ \rightarrow \]

When \( y = y_2 \)

AET \[ \rightarrow \]
3.5 Scan Converting Polygons

Algorithms:
- Set \( y \) to the \( y \)-coordinate of the first nonempty bucket
- Set AET to empty
- **Repeat until** the AET and ET are both empty
  - **Merge** edges from ET bucket \( y \) *with* edges in AET, maintaining AET sort order on \( x \)
  - **Fill in** pixels on scan line \( y \) *bounded by* pairs of \( x \)-coordinates from edges in AET
  - **Remove** from AET those edges for which \( y = \text{ytop} \)
  - For each edge remaining in AET, replace \( x \) with \( x + \frac{1}{m} \)
  - **Increment** \( y \) by 1, *to the coordinate of* the next scan line