

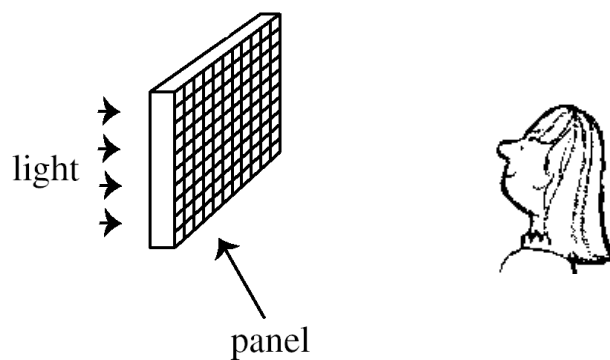
G-I Exercises

1. Scanning Process with Colour LUTs

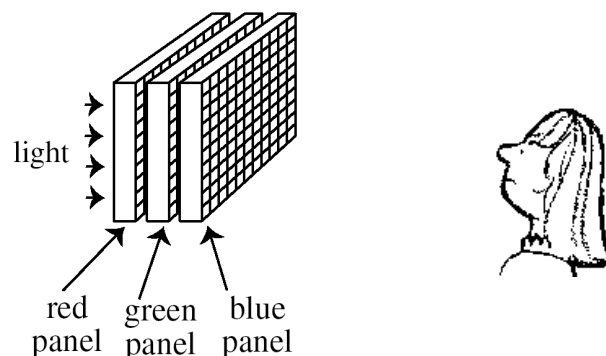
- Draw a detailed block diagram for the frame buffer scanning process where the frame buffer stores colour indices reference to a colour look-up table (LUT) for a CRT display.
- What are the limitations of this approach? Why would one want to use a colour LUT anyway? Reinforce your answer with an example for a typical graphics display with and without LUT.

2. An Ink Display Device

An inventor has developed a new monochrome display device: a transparent panel with a rectangular array of square pixels (see figure). The panel is tinted with a special ink which allows each pixel to range from totally transparent to transmitting only the colour of the ink. Each pixel has an 8-bit value. For example, with red ink, a pixel value of 0 would be totally transparent, 255 totally red (only red light is transmitted), and 100 a pale red.



- For a panel with 1600×1200 pixels, how long does a pixel have to hold its state before it can be refreshed if the frame buffer memory requires 8ns to read a single byte? What is the maximal refresh rate for this setup?
- With these settings can the display operate at a resolution of 1280×1024 and a refresh rate of 125Hz?
- The inventor found that he can make the special ink *any* colour he likes, but that each panel can be tinted with only one colour. He proposes to use three inks in three panels to make a 24-bit display: a red-tinted panel, a green-tinted panel and a blue-tinted panel will be stacked up to make a full-colour display (see figure). A value of (0, 0, 0) will be white (transparent), (255, 0, 0) red and (255, 255, 255) black.



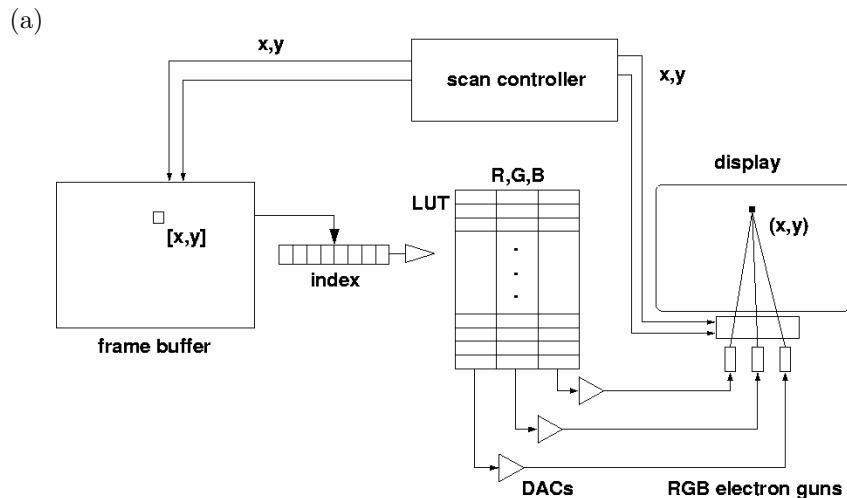
Explain why this display does not work correctly. How can the three-panel design be modified such that it will work?

3. Graphics Pipeline Speedup

A simple way to reduce the number of calculations in a graphics pipeline when displaying polygonal meshes is to use a mesh primitive, such as a triangle strip. A triangle strip is a sequence of three or more vertices, in which every consecutive set of three vertices $[(v_1, v_2, v_3), (v_2, v_3, v_4), (v_3, v_4, v_5), \text{etc.}]$ defines a triangle. Assume you intend to process n three-dimensional triangles by a graphics pipeline. What is the maximum speedup you could obtain in the front-end subsystem (transformations on the geometry) by converting triangles into triangle strips? Compute the number of multiplications and additions required to perform a transformation on a set of separate triangles and a triangle strip to support your answer.

G-I Exercise Solutions

1. Scanning Process with Colour LUTs

**Limitation:**

- a frame buffer with n bits per pixel can at most refer to 2^n colours in an LUT
- if we have $w \geq n$ bits to represent a colour in the LUT, we can display at most 2^w of 2^n colours at the same time

(b) Reason for usage:

- LUTs require less memory / cheaper hardware
- example: 1280×1024 pixels, 24 bit colours (8 bits per RGB colour) and 8 bit indices:
 - without LUT: $1280 \times 1024 \times 24/8 = 3932160$ bytes
 - with LUT: $1280 \times 1024 \times 8/8 + 2^8 \times 24/8 = 1311488$ bytes
 - with LUT only $\approx \frac{1}{3}$ of memory (can be approximated by directly comparing bits needed for colour / index representation)
- LUTs can be used for simple animations by changing colours in the LUT without changing the image itself: areas of the same colour that do not change between two frames are coloured with one index and each area which changes between frames is coloured with a separate indices. Then depending on which indices refer to the same colour in the LUT a different frame is displayed.

2. An Ink Display Device

- (a)
- Time to display one frame: $1600 \times 1200 \times 8\text{ns} = 15360000\text{ns} \approx 15\text{ms}$
Each pixel has to hold its state approximately for at least 15ms
 - This gives a maximal refresh rate of approximately $1/0.0153 \approx 65\text{Hz}$
- (b)
- The maximal refresh rate of the display at the resolution is $1/(1280 \times 1024 \times 8\text{ns}) = 95\text{Hz} (\approx 1600 \times 1200 / (1280 \times 1024) \times 65\text{Hz})$. Hence, a 125Hz refresh rate cannot be achieved.

Alternative:

- There are $1280 \times 1024 = 1310720$ pixels on the display
 - At 125Hz we have $1280 \times 1024 \times 125 = 163840000$ pixels per second
 - So we have $1/(1280 \times 1024 \times 125) \approx 6.1\text{ns}$ per pixel
 - The video memory operates at 8ns and hence cannot support this refresh rate.
- (c)
- The panel does not work correctly because the inks have the wrong colours. The device works by absorbing light (subtractive colour model), not by emitting light (additive colour model). The red panel will absorb blue and green light, the green

panel absorbs red and blue light and the blue panel absorbs red and green light. So only two of the panels need to have a value of 255 to display black (e.g. yellow will be black) which is wrong.

- To make this work, the ink colours have to be changed to cyan, magenta and yellow (subtractive colour model). (Not required, but note that this still does not display *all possible* colours a human can see as whole chunks of the spectrum are absorbed rather than a continuously varying absorption across the spectrum. But this is true for other “full-colour” displays as well).

3. Graphics Pipeline Speedup

- the front-end consists primarily of transformations on homogeneous coordinates, i.e. 4×4 matrices on 4 dimensional vectors
- one vertex requires 16 multiplications and 12 additions per transformation
- n triangles with three points require $3n16 = 48n$ multiplications and $3n12 = 36n$ additions per transformation
- a triangle strip with m vertices defines $m - 2$ triangles, i.e. the minimum number of vertices for n triangles is $n + 2$ in a single triangle strip
- so, the maximum speedup for n triangles can be achieved with a triangle strip defined by $n + 2$ vertices
- this requires $(n + 2)16 = 16n + 32$ multiplications and $(n + 2)12 = 12n + 24$ additions per transformation
- hence, the maximum speedup is about 3