Week 1 study sheet: Graphics hardware and programming

Problem 1 (17 points) The American standard video system calls for a total of 525 horizontal lines. However only 483 of these 525 lines are actually visible because a time equivalent to 42 lines is required to reposition the electron beam from the bottom of the screen back to the top. For the duration of this "vertical retrace" period, the electron beam is made invisible or "blanked."

a) (3 points) The American standard video system also calls for a "viewing aspect ratio" of 4:3; that is, the visible viewing area is three-quarters as high as it is wide. How many pixels are on each scan line, assuming square pixels?

Problem 1 (continued)

b) (3 points) To avoid flicker, the whole screen needs to be refreshed at 60Hz. How much time does this allow for each scan line?

c) (3 points) When the electron beam reaches the right edge of the screen after each scan line is drawn, it is made invisible and rapidly returned to the left edge. This "horizontal retrace" period generally requires 17% of the time allowed for one scan line. How much time does this leave for accessing and displaying each pixel?

Problem 1 (continued)

In order to allow more time for transmitting and displaying each pixel, the American broadcast television standard uses an "interlaced" type of refresh, in which each video frame is broken into two "fields," each containing one-half of the picture. The two fields are "interlaced" in the sense that each field contains every other scan line: all odd-numbered scan lines are displayed in the first field, and all even-numbered scan lines are displayed in the second.

The purpose of an interlaced scan is to place some new information in all areas of the screen at a high enough rate to avoid flicker, while allowing the hardware more time for accessing and displaying each pixel.

d) (2 points) If the video controller displays each field in 1/60th of a second, what is the overall frame rate for displaying the entire screen?

e) (2 points) Assuming no additional vertical retrace time is required for painting the two fields, how much time is available for drawing each pixel?

Problem 1 (continued)

f) (4 points) An interlaced refresh works well as long as adjacent scan lines display similar information. In which parts, if any, of the following images would you expect to see flicker on an interlaced display:

- A single horizontal white line on a black background?

- A single vertical white line on a black background?

- A checkerboard of black and white, where each black or white square is 8×8 pixels?

- A checkerboard of black and white, where each black or white square is a single pixel?

Problem 2 (23 points) Color maps provide an inexpensive means for specifying a limited number of colors from a very large palette. The "VersaGraphics" color-mapped system has 16 color table entries, with each color table entry specifying 4 bits per color channel.

a) (2 points) How many colors can be displayed simultaneously on the VersaGraphics?

b) (2 points) How large is the palette of colors?

Problem 2 (continued)

c) (4 points) "VersaText" is a text editor that runs on the VersaGraphics that provides antialiased fonts, with black letters on a white background. How is VersaText likely to initialize the color table?

d) (4 points) "VersaSaver" is a screen saver application that runs on the same system. It instantly dims the screen to 40% of its normal brightness whenever there has been no user input for 2 minutes or more. How does VersaSaver work?

Problem 2 (continued)

The VersaGraphics hardware also includes a 4-bit "write mask" M that allows an application to write only selected bits of framebuffer memory. When the write mask is used, the call *DrawPixel*(x, y, K) updates the framebuffer memory $FB[x, y]$ as follows:

$$FB[x,y] \quad \leftarrow \quad (FB[x,y] \text{ AND } \overline{M}) \text{ OR } (K \text{ AND } M)$$

where $\overline{M}$ is the bitwise complement of M, and AND and OR are bitwise Boolean operations.

For example, when M is 1111 (binary), the *DrawPixel* routine behaves in the normal fashion—that is, *DrawPixel*(x, y, K) puts the color table index K into pixel (x, y). However, when M is 0001 (binary), only the low-order bit of pixel (x, y) is affected.

e) (5 points) The write mask allows us to think of the framebuffer as containing four separate bilevel images or "planes." What color table is appropriate for displaying in black and white the bilevel image drawn with write mask 0001?

Problem 2 (continued)

f) (6 points) The "VersaDraw" application is a drawing program that uses pop-up menus. These pop-up menus are especially nice in that they don't completely obliterate what's under them. Instead, when a pop-up menu appears, the image on the screen appears to dim, and the menu is superimposed as white text on top of the image. How does VersaDraw manage this feat?

Problem 3 (10 points) Suppose that you would like to display some scalar function $f(x, y)$ over a two-dimensional domain $x_{\min} \leq x \leq x_{\max}$ and $y_{\min} \leq y \leq y_{\max}$. (The function f might be the Mandelbrot set, for example, with (x, y) representing a single complex number.) Suppose, further, that the limits of the domain $x_{\min}, x_{\max}, y_{\min}, y_{\max}$ are specified by a user, using interactive "zooming in" and "zooming out" tools:

- To zoom in, the user left-clicks at some point (x_i, y_i). The system centers (x_i, y_i) in the window and scales the image up by a factor of two (in each dimension).
- To zoom out, the user right-clicks at some point (x_o, y_o). The system centers (x_o, y_o) in the window and scales the image down by a factor of two (in each dimension).

Write expressions for the new domain limits $x'_{\min}$, $x'_{\max}$, $y'_{\min}$, $y'_{\max}$ in terms of the old domain limits $x_{\min}$, $x_{\max}$, $y_{\min}$, $y_{\max}$ and the selected points (x_i, y_i) and (x_o, y_o), for both of these cases.