Computer Graphics

Lecture 7

Output primitives:

A picture can be described in several ways. Assuming we have a raster display, a picture is completely specified by the set of intensities for the pixel positions in the display. At the other extreme, we can describe a picture as a set of complex objects, such as trees and terrain or furniture and walls, positioned at specified coordinate locations within the scene. Shapes and colours of the objects can be described internally with pixel arrays or with sets of basic geometric structures, such as straight line segments and polygon colour areas. The scene is then displayed either by loading the pixel arrays into the frame buffer or by scan converting the basic geometric-structure specifications into pixel patterns. Typically, graphics programming packages provide functions to describe a scene in terms of these basic geometric structures, referred to as output primitives, and to group sets of output primitives into more complex structures. Each output primitive is specified with input coordinate data and other information about the way that object is to be displayed. Points and straight line segments are the simplest geometric components of pictures. Additional output primitives that can be used to construct a picture include circles and other conic sections, quadric surfaces, spline curves and surfaces, polygon colour areas, and character strings. We begin our discussion of picture-generation procedures by examining device-level algorithms for displaying two-dimensional output primitives, with particular emphasis on scan-conversion methods for raster graphics systems.

POINTS AND LINES: Point plotting is accomplished by converting a single coordinate position furnished by an application program into appropriate operations for the output device in use. With a CRT monitor, for example, the electron beam is turned on to illuminate the screen phosphor at the selected location. How the electron beam is positioned depends on the display technology. A random-scan (vector) system stores point-plotting instructions in the display list, and coordinate values in these instructions are converted to deflection voltages that position the electron beam at the screen locations to be plotted during each refresh cycle. For a black-andwhite raster system, on the other hand, a point is plotted by setting the bit value corresponding to a specified screen position within the frame buffer to 1. Then, as the electron beam sweeps across each horizontal scan line, it emits a burst of electrons (plots a point) whenever a value of 1 is encountered in the frame buffer. With an RGB system, the frame buffer is loaded with the colour codes for the intensities that are to be displayed at the screen pixel positions.

Line drawing is accomplished by calculating intermediate positions along the line path between two specified endpoint positions. An output device is then directed to fill in these positions between the endpoints. For analog devices, such as a vector pen plotter or a random-scan display, a straight line can be drawn smoothly from one endpoint to the other. Linearly varying horizontal and vertical deflection voltages are generated that are proportional to the required changes in the x and y directions to produce the smooth line.

Digital devices display a straight line segment by plotting discrete points between the two endpoints. Discrete coordinate positions along the line path are calculated from the equation of the line. For a raster video display, the line colour (intensity) is then loaded into the frame buffer at the corresponding pixel coordinates. Reading from the frame buffer, the video controller then "plots" the screen pixels. Screen locations are referenced with integer values, so plotted positions may only approximate actual Line positions between two specified endpoints. A computed line position of (10.48,20.51), for example, would be converted to pixel position (10,21). This rounding of coordinate values to integers causes lines to be displayed with a stairstep appearance ("the jaggies"), as represented in Fig 2-14. The characteristic stairstep shape of raster lines is particularly noticeable on systems with low resolution, and we can improve their appearance somewhat by displaying them on high-resolution systems. More effective techniques for smoothing raster lines are based on adjusting pixel intensities along the line paths.

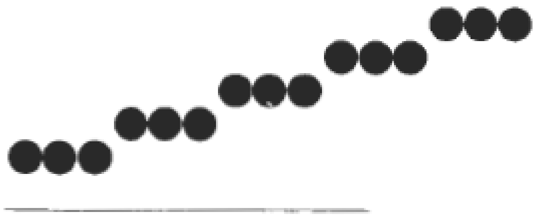
Fig 2-14(Stairstep effect produced when a line is generated as a series of pixel positions)

For the raster-graphics device-level algorithms, object positions are specified directly in integer device coordinates. For the time being, we will assume that pixel positions are referenced according to scan-line number and column number (pixel position across a scan line). This addressing scheme is illustrated in Fig. 2-15. Scan lines are numbered consecutively from 0, starting at the bottom of the screen; and pixel columns are numbered from 0, left to right across each scan line.
To load a specified colour into the frame buffer at a position corresponding to column x along scan line y, we will assume we have available a low-level procedure of the form

setPixel(x,y)

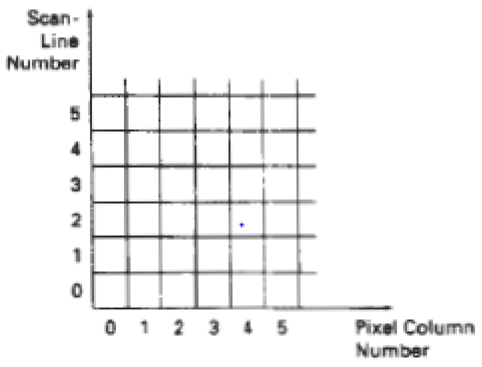

Fig 2-15(Pixe1 positions referenced by scan-line number and column number)

We sometimes will also want to be able to retrieve the current frame-buffer intensity setting for a specified location. We accomplish this with the low-level function

getPixel(x,y)

LINE-DRAWING ALGORITHMS:
The Cartesian slope-intercept equation for a straight line is

$$y = m.\, x + b \qquad (2.1)$$

With m representing the slope of the line and b as they intercept. Given that the two endpoints of a line segment are specified at positions (x1, y1) and (x2, y2) as shown in Fig. 2-16, we can determine values for the slope m and y intercept b with the following calculations

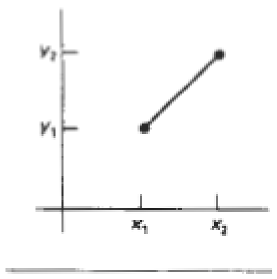

Fig 2-16(Line path between endpoint positions (x1, y1) and (x2, y2))

$$m = (y2-y1) / (x2-x1) \qquad (2.2)$$

$$b = y1 - m.\ x1 \qquad (2.3)$$

Algorithms for displaying straight lines are based on the line equation 2.1 and the calculations given in Eqs. 2.2 and 2.3
For any given x interval Δx along a line, we can compute the corresponding y interval Δy from Eq 3.2 as

$$\Delta y = m\ \Delta x \qquad (2.4)$$

Similarly, we can obtain the x interval Δx corresponding to a specified Δy as

$$\Delta x = \Delta y / m \qquad (2.5)$$

These equations form the basis for determining deflection voltages in analog devices. For lines with slope magnitudes I m I < 1, Δx can be set proportional to a small horizontal deflection voltage and the corresponding vertical deflection is then set proportional to Δy as calculated from Eq. 2.4. For lines whose slopes have magnitudes I m I > 1, Δy can be set proportional to a small vertical deflection voltage with the corresponding horizontal deflection voltage set proportional to Δx, calculated from Eq. 2.5. For lines with m = 1, Δx = Δy and the horizontal and vertical deflections voltages are equal. In each case, a smooth line with slope m is generated between the specified endpoints.

On raster systems, lines are plotted with pixels, and step sizes in the horizontal and vertical directions are constrained by pixel separations. That is, we must "sample" a line at discrete positions and determine the nearest pixel to the line at each sampled position. This scan conversion process for straight lines is illustrated in Fig. 2-17, for a near horizontal line with discrete sample positions along the x axis.

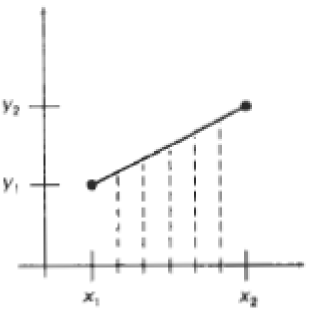

Fig 2-17(Straight line segment with five sampling positions along the x-axis between *x*, and *x2)*