PRINCIPLES OF INTERACTIVE COMPUTER GRAPHICS

COMPUTER SCIENCE SERIES

Library of Congress Cataloging in Publication Data

Newman, William M 1939-
Principles of interactive computer graphics.

(McGraw-Hill computer science series) (McGraw-Hill series in artificial intelligence)
Bibliography : p.
Includes index.
1. Computer graphics. 2. Interactive computer systems. I. Sproull, Robert F., joint author.
II. Title.

T385.N48 1979 001.55 78-23825
ISBN 0-07-046338-7

PRINCIPLES OF INTERACTIVE COMPUTER GRAPHICS

5 6 7 8 9 0 **DODO** 8 3 2 1 0

The editors were Charles E. Stewart and Frances A. Neal; the production supervisor was Dominick Petrellese. R.R. Donnelley & Sons Company was printer and binder.

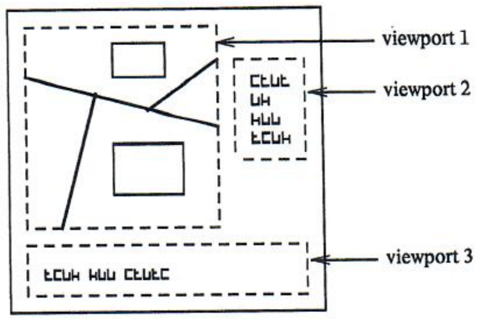

Figure 5-13 Use of viewports for map display (1), command menu (2), and user messages (3).

5-6 THE WINDOWING TRANSFORMATION

The windowing transformation is so named because it involves specifying a *window* in the world-coordinate space surrounding the information we wish displayed. This is by no means the only way such a transformation can be specified; for example, we can define the scale factor and translation to be applied to the picture, or in place of the translation we can define a world-coordinate point we wish transformed to a certain spot on the screen, say the screen center. Each of these methods may prove convenient in certain circumstances; the windowing transformation has the advantage of letting us specify directly the rectangle of interest in the world-coordinate picture definition.

In addition to the window, we can also define a *viewport*, a rectangle on the screen where we would like the window's contents displayed. In doing so, we exploit the ability of our clipping algorithm to clip to any right rectangular boundary. It is often useful to specify a viewport smaller than the screen, for we can then leave room for command menus, system messages, and so forth. Each such part of the picture may be displayed in a separate viewport, as shown in Figure 5-13.

Figure 5-14 illustrates the full windowing transformation from world to screen. We use the window to define *what* we want to display; we use the viewport to specify *where* on the screen to put it. Thus we can scan over a large picture by keeping the window size constant and varying its position; changing the window size alters the picture magnification. Normally we shall take care to keep the window and viewport similar in shape. In Figure 5-14 however, we see the distortion that can be achieved by making the shapes different.

The actual transformation we apply to each point is very simple. Let us suppose that the edges of the window are at $x = W_{xl}$, $x = W_{xr}$, $y = W_{yb}$, and $y = W_{yt}$ (see Figure 5-14), all measured in *world coordinates,* and that the corresponding edges of the viewport are at $x = V_{xl}$, $x = V_{xr}$, $y = V_{yb}$, and $y = V_{yt}$, all measured in *screen coordinates.* Then the point (x_w, y_w) in world coordinates transforms into the point (x_s, y_s) in screen coordinates, as follows:

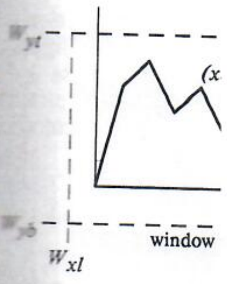

Figure 5-14 The windowin

$x_s = \frac{V}{W}$

$y_s = \frac{V}{W}$

These expressions
(x_w, y_w) within the wi
left-hand corner and i
the viewport. Adding
the position of (x_s, y_s)
Equations 5-2 redu

$x_s = a$

We therefore arrange t
and viewport are defin
involving only two mul
The complete win
by transforming the er
the line against the vi
picture quickly we nat
so in a couple of ways.
lines in a picture defini
with shared endpoints.
by comparing the worl
A more dramatic
gained by transforming
right rectangles, and sin

gure 5-13 Use of viewports for map
splay (1), command menu (2), and
er messages (3).

ıuse it involves specifying a
ıg the information we wish
ıch a transformation can be
factor and translation to be
tion we can define a world-
ı spot on the screen, say the
rove convenient in certain
the advantage of letting us
e world-coordinate picture

ı *viewport*, a rectangle on the
s displayed. In doing so, we
lip to any right rectangular
smaller than the screen, for
tem messages, and so forth.
separate viewport, as shown

ınsformation from world to
vant to display; we use the
'hus we can scan over a large
arying its position; changing
ormally we shall take care to
In Figure 5-14 however, we
ıe shapes different.
ɔoint is very simple. Let us
$'_{xl}, x = W_{xr}, y = W_{yb}$, and
'd coordinates, and that the
$'_{xl}, x = V_{xr}, y = V_{yb}$, and
the point (x_w, y_w) in world
ɛen coordinates, as follows:

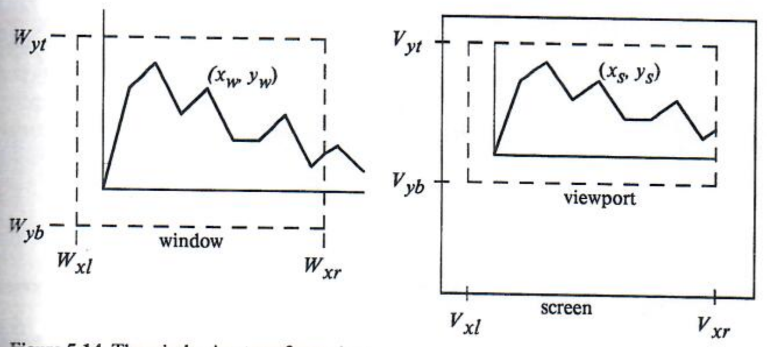

Figure 5-14 The windowing transformation.

$$x_s = \frac{V_{xr} - V_{xl}}{W_{xr} - W_{xl}}(x_w - W_{xl}) + V_{xl}$$

$$y_s = \frac{V_{yt} - V_{yb}}{W_{yt} - W_{yb}}(y_w - W_{yb}) + V_{yb} \qquad (5\text{-}2)$$

These expressions are derived by determining the position of the point (x_w, y_w) within the window as a fraction of full displacement from the bottom left-hand corner and interpreting this as a fraction of full displacement across the viewport. Adding the offset of the viewport's bottom left-hand corner gives the position of (x_s, y_s) on the screen.

Equations 5-2 reduce to the form

$$x_s = ax_w + b \qquad y_s = cy_w + d \qquad (5\text{-}3)$$

We therefore arrange to compute the values of *a, b, c,* and *d* when the window and viewport are defined, so that we can transform each point by a computation involving only two multiplications and two additions.

The complete windowing transformation can then be applied to a picture by transforming the endpoints of each line, using Equations 5-3, and clipping the line against the viewport boundary. In the interests of transforming the picture quickly we naturally wish to speed up the computation, and we can do so in a couple of ways. The first of these is based on the observation that the lines in a picture definition are often connected as a sequence of line segments with shared endpoints. We can avoid transforming each shared endpoint twice by comparing the world coordinates of each point and its predecessor.

A more dramatic improvement in the speed of transformation can be gained by transforming only visible lines. Since both window and viewport are right rectangles, and since both circumscribe the displayed information, we may use either of them as a clipping region. The more efficient implementation of the windowing transformation therefore clips *first*, using the window as clipping region, and then performs the transformation of Equations 5-3 on those lines which are at least partially visible. When a small portion of a very large picture is being viewed, the advantage of performing clipping before transformation is considerable.

EXERCISES

5-1 Program the Sutherland-Hodgman polygon clipping algorithm in the language of your choice, using recursion if possible.

5-2 The Sutherland-Hodgman algorithm can be used to clip lines against a nonrectangular boundary. What uses might this have? What modifications to the algorithm would be necessary? What restrictions would apply to the shape of the clipping region?

5-3 Some displays possess hardware for displaying circular arcs. Design an algorithm to clip circles and generate arcs suitable for passing to such a display.

5-4 Extend the clipping program given on page 66 to perform the complete viewing transformation of a line from world coordinates to screen coordinates.

5-5 Application programs often use floating-point numbers to define pictures, whereas the display uses integers. Should the conversion from floating-point to integer format be done before or after clipping? How should numbers be rounded? Does the decision whether to clip to the window or viewport depend on the relative speeds of integer and floating-point arithmetic?

5-6 Under what circumstances would midpoint clipping be preferable to the use of the program on page 66?

5-7 What additional logic is needed in the clipping algorithm to keep track of linked visible line segments?

5-8 The Cohen-Sutherland clipping algorithm is optimized in favor of clipping pictures much larger than the window. What features would you look for in an algorithm to clip pictures only slightly larger than the window? Devise such an algorithm.

5-9 Hand-simulate the polygon clipping alrithm to verify that it produces the same result as shown in Figure 5-9. Copy the figure of the arrow to graph paper and step through the flow chart of Figure 5-11.