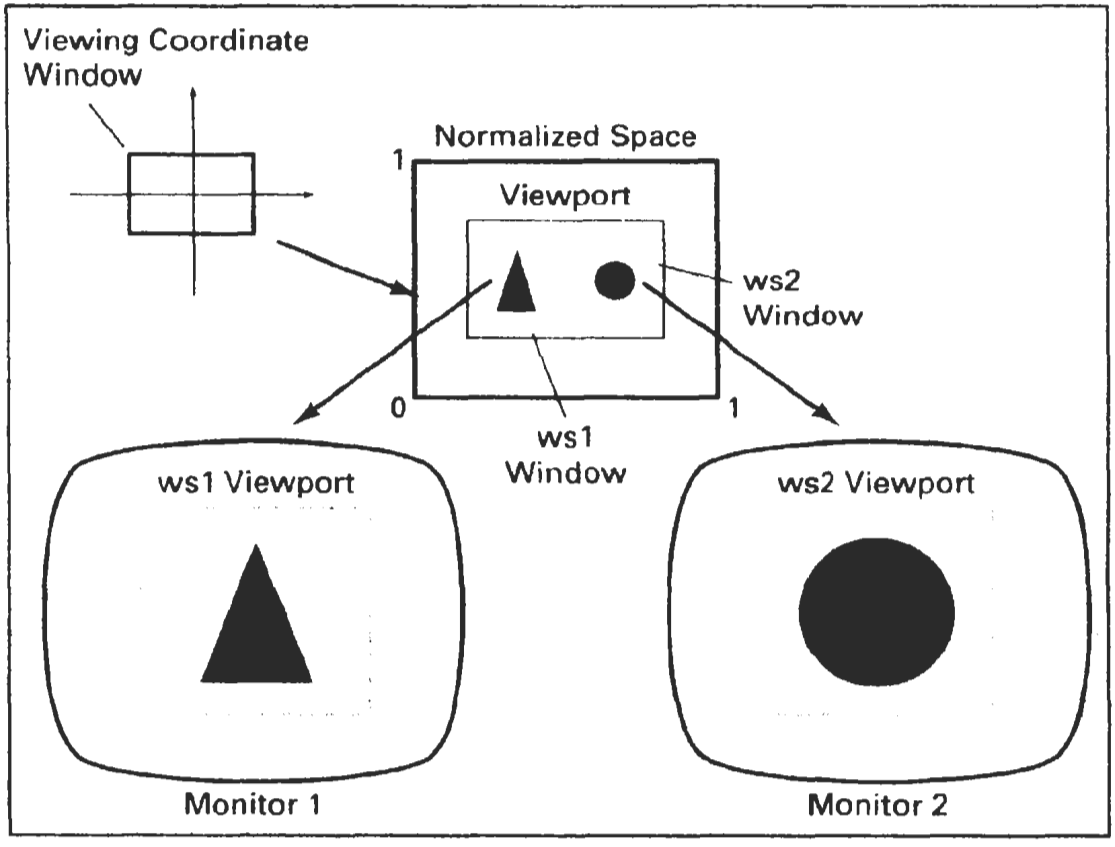
CHAPTER

6 Two-Dimensional Viewing



We now consider the formal mechanism for displaying views of a picture on an output device. Typically, a graphics package allows a user to specify which part of a defined picture is to be displayed and where that part is to be placed on the display device. Any convenient Cartesian coordinate system, referred to as the world-coordinate reference frame, can be used to define the picture. For a two-dimensional picture, a view is selected by specifying a subarea of the total picture area. A user can select a single area for display, or several areas could be selected for simultaneous display or for an animated panning sequence across a scene. The picture parts within the selected areas are then mapped onto specified areas of the device coordinates. When multiple view areas are selected, these areas can be placed in separate display locations, or some areas could be inserted into other, larger display areas. Transformations from world to device coordinates involve translation, rotation, and scaling operations, as well as procedures for deleting those parts of the picture that are outside the limits of a selected display area.

6-1 THE VIEWING PIPELINE

A world-coordinate area selected for display is called a **window**. An area on a display device to which a window is mapped is called a **viewport.** The window defines *what* is to be viewed; the viewport defines *where* it is to be displayed. Often, windows and viewports are rectangles in standard position, with the rectangle edges parallel to the coordinate axes. Other window or viewport geometries, such as general polygon shapes and circles, are used in some applications, but these shapes take longer to process. In general, the mapping of a part of a world-coordinate scene to device coordinates is referred to as a **viewing transformation**. Sometimes the two-dimensional viewing transformation is simply referred to as the *window-to-viewport transformation* or the *windowing transformation.* But, in general, viewing involves more than just the transformation from the window to the viewport. Figure 6-1 illustrates the mapping of a picture section that falls within a rectangular window onto a designated rectangular viewport.

In computer graphics terminology, the term *window* originally referred to an area of a picture that is selected for viewing, as defined at the beginning of this section. Unfortunately, the same term is now used in window-manager systems to refer to any rectangular screen area that can be moved about, resized, and made active or inactive. In this chapter, we will only use the term window to

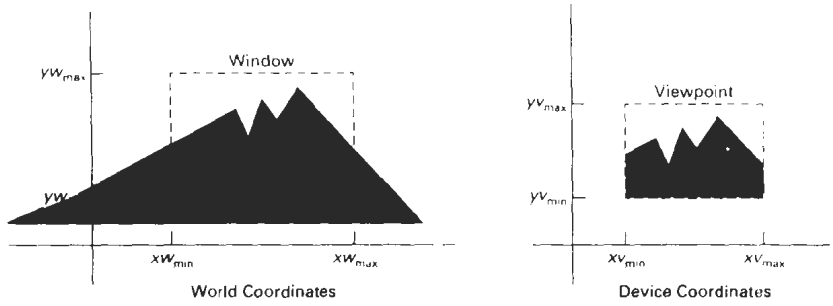


Figure 6-1
A viewing transformation using standard rectangles for the window and viewport.

refer to an area of a world-coordinate scene that has been selected for display. When we consider graphical user interfaces in Chapter 8, we will discuss screen windows and window-manager systems.

Some graphics packages that provide window and viewport operations allow only standard rectangles, but a more general approach is to allow the rectangular window to have any orientation. In this case, we carry out the viewing transformation in several steps, as indicated in Fig. 6-2. First, we construct the scene in world coordinates using the output primitives and attributes discussed in Chapters 3 and 4. Next, to obtain a particular orientation for the window, we can set up a two-dimensional **viewing-coordinate system** in the world-coordinate plane, and define a window in the viewing-coordinate system. The viewing-coordinate reference frame is used to provide a method for setting up arbitrary orientations for rectangular windows. Once the viewing reference frame is established, we can transform descriptions in world coordinates to viewing coordinates. We then define a viewport in normalized coordinates (in the range from 0 to 1) and map the viewing-coordinate description of the scene to normalized coordinates. At the final step, all parts of the picture that lie outside the viewport are clipped, and the contents of the viewport are transferred to device coordinates. Figure 6-3 illustrates a rotated viewing-coordinate reference frame and the mapping to normalized coordinates.

By changing the position of the viewport, we can view objects at different positions on the display area of an output device. Also, by varying the size of viewports, we can change the size and proportions of displayed objects. We achieve zooming effects by successively mapping different-sized windows on a

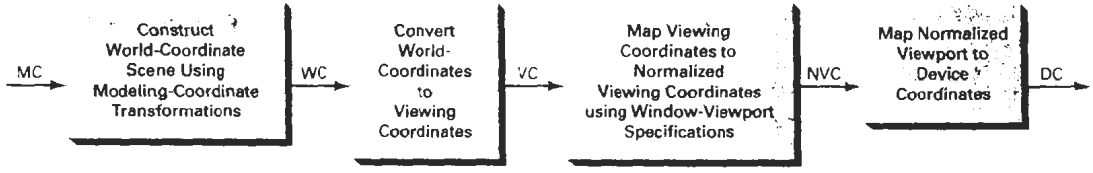


Figure 6-2
The two-dimensional viewing-transformation pipeline.

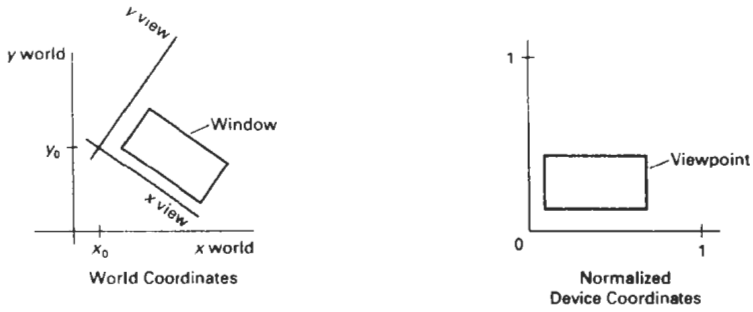


Figure 6-3
Setting up a rotated world window in viewing coordinates and the corresponding normalized-coordinate viewport.

fixed-size viewport. As the windows are made smaller, we zoom in on some part of a scene to view details that are not shown with larger windows. Similarly, more overview is obtained by zooming out from a section of a scene with successively larger windows. Panning effects are produced by moving a fixed-size window across the various objects in a scene.

Viewports are typically defined within the unit square (normalized coordinates). This provides a means for separating the viewing and other transformations from specific output-device requirements, so that the graphics package is largely device-independent. Once the scene has been transferred to normalized coordinates, the unit square is simply mapped to the display area for the particular output device in use at that time. Different output devices can be used by providing the appropriate device drivers.

When all coordinate transformations are completed, viewport clipping can be performed in normalized coordinates or in device coordinates. This allows us to reduce computations by concatenating the various transformation matrices. Clipping procedures are of fundamental importance in computer graphics. They are used not only in viewing transformations, but also in window-manager systems, in painting and drawing packages to eliminate parts of a picture inside or outside of a designated screen area, and in many other applications.

6-2
VIEWING COORDINATE REFERENCE FRAME

This coordinate system provides the reference frame for specifying the world-coordinate window. We set up the viewing coordinate system using the procedures discussed in Section 5-5. First, a viewing-coordinate origin is selected at some world position: $\mathbf{P}_0 = (x_0, y_0)$. Then we need to establish the orientation, or rotation, of this reference frame. One way to do this is to specify a world vector $\mathbf{V}$ that defines the viewing y_v direction. Vector $\mathbf{V}$ is called the **view up vector**.

Given $\mathbf{V}$, we can calculate the components of unit vectors $\mathbf{v} = (v_x, v_y)$ and $\mathbf{u} = (u_x, u_y)$ for the viewing y_v and x_v axes, respectively. These unit vectors are used to form the first and second rows of the rotation matrix $\mathbf{R}$ that aligns the viewing $x_v y_v$ axes with the world $x_w y_w$ axes.

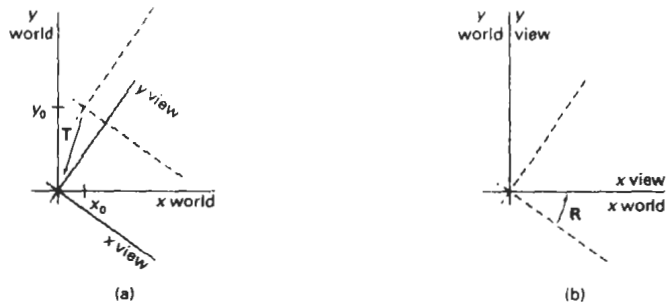


Figure 6-4
A viewing-coordinate frame is moved into coincidence with the world frame in two steps: (a) translate the viewing origin to the world origin, then (b) rotate to align the axes of the two systems.

We obtain the matrix for converting world-coordinate positions to viewing coordinates as a two-step composite transformation: First, we translate the viewing origin to the world origin, then we rotate to align the two coordinate reference frames. The composite two-dimensional transformation to convert world coordinates to viewing coordinates is

$$\mathbf{M}_{WC,VC} = \mathbf{R} \cdot \mathbf{T} \tag{6-1}$$

where $\mathbf{T}$ is the translation matrix that takes the viewing origin point $\mathbf{P}_0$ to the world origin, and $\mathbf{R}$ is the rotation matrix that aligns the axes of the two reference frames. Figure 6-4 illustrates the steps in this coordinate transformation.

6-3 WINDOW-TO-VIEWPORT COORDINATE TRANSFORMATION

Once object descriptions have been transferred to the viewing reference frame, we choose the window extents in viewing coordinates and select the viewport limits in normalized coordinates (Fig. 6-3). Object descriptions are then transferred to normalized device coordinates. We do this using a transformation that maintains the same relative placement of objects in normalized space as they had in viewing coordinates. If a coordinate position is at the center of the viewing window, for instance, it will be displayed at the center of the viewport.

Figure 6-5 illustrates the window-to-viewport mapping. A point at position (xw, yw) in the window is mapped into position (xv, yv) in the associated viewport. To maintain the same relative placement in the viewport as in the window, we require that

$$\frac{xv - xv_{min}}{xv_{max} - xv_{min}} = \frac{xw - xw_{min}}{xw_{max} - xw_{min}}$$
$$\frac{yv - yv_{min}}{yv_{max} - yv_{min}} = \frac{yw - yw_{min}}{yw_{max} - yw_{min}} \tag{6-2}$$

Figure 6-5
A point at position (xw, yw) in a designated window is mapped to viewport coordinates (xv, yv) so that relative positions in the two areas are the same.

Solving these expressions for the viewport position (xv, yv), we have

$$\begin{aligned} xv &= xv_{min} + (xw - xw_{min})sx \\ yv &= yv_{min} + (yw - yw_{min})sy \end{aligned} \tag{6-3}$$

where the scaling factors are

$$\begin{aligned} sx &= \frac{xv_{max} - xv_{min}}{xw_{max} - xw_{min}} \\ sy &= \frac{yv_{max} - yv_{min}}{yw_{max} - yw_{min}} \end{aligned} \tag{6-4}$$

Equations 6-3 can also be derived with a set of transformtions that converts the window area into the viewport area. This conversion is performed with the following sequence of transformations:

1. Perform a scaling transformation using a fixed-point position of (xw_{min}, yw_{min}) that scales the window area to the size of the viewport.
2. Translate the scaled window area to the position of the viewport.

Relative proportions of objects are maintained if the scaling factors are the same ($sx = sy$). Otherwise, world objects will be stretched or contracted in either the x or y direction when displayed on the output device.

Character strings can be handled in two ways when they are mapped to a viewport. The simplest mapping maintains a constant character size, even though the viewport area may be enlarged or reduced relative to the window. This method would be employed when text is formed with standard character fonts that cannot be changed. In systems that allow for changes in character size, string definitions can be windowed the same as other primitives. For characters formed with line segments, the mapping to the viewport can be carried out as a sequence of line transformations.

From normalized coordinates, object descriptions are mapped to the various display devices. Any number of output devices can be open in a particular application, and another window-to-viewport transformation can be performed for each open output device. This mapping, called the **workstation transforma-**

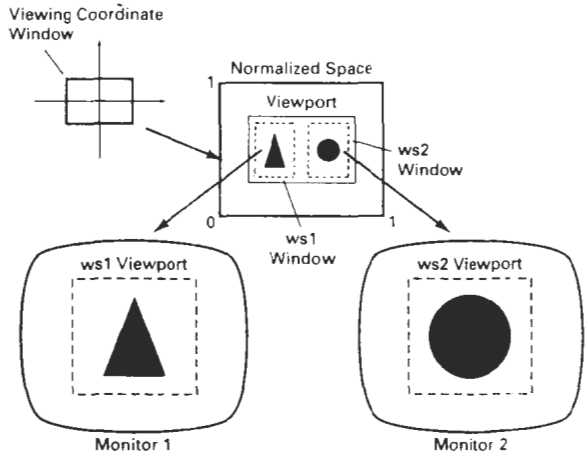


Figure 6-6
Mapping selected parts of a scene in normalized coordinates to different video monitors with workstation transformations.

tion, is accomplished by selecting a window area in normalized space and a viewport area in the coordinates of the display device. With the workstation transformation, we gain some additional control over the positioning of parts of a scene on individual output devices. As illustrated in Fig. 6-6, we can use workstation transformations to partition a view so that different parts of normalized space can be displayed on different output devices.

6-4
TWO-DIMENSIONAL VIEWING FUNCTIONS

We define a viewing reference system in a PHIGS application program with the following function:

```
evaluateViewOrientationMatrix (x0, y0, xV, yV,
                               error, viewMatrix)
```

where parameters `x0` and `y0` are the coordinates of the viewing origin, and parameters `xV` and `yV` are the world-coordinate positions for the view up vector. An integer error code is generated if the input parameters are in error; otherwise, the `viewMatrix` for the world-to-viewing transformation is calculated. Any number of viewing transformation matrices can be defined in an application.

To set up the elements of a window-to-viewport mapping matrix, we invoke the function

```
evaluateViewMappingMatrix (xwmin, xwmax, ywmin, ywmax,
    xvmin, xvmax, yvmin, yvmax, error, viewMappingMatrix)
```

Here, the window limits in viewing coordinates are chosen with parameters `xwmin`, `xwmax`, `ywmin`, and `ywmax`; and the viewport limits are set with the nor-