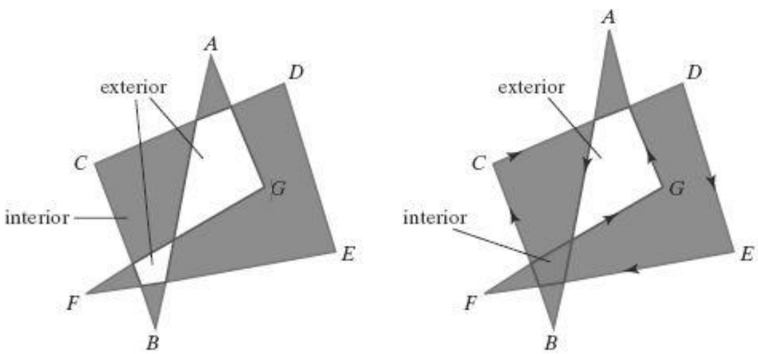
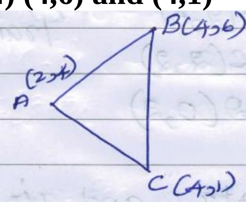
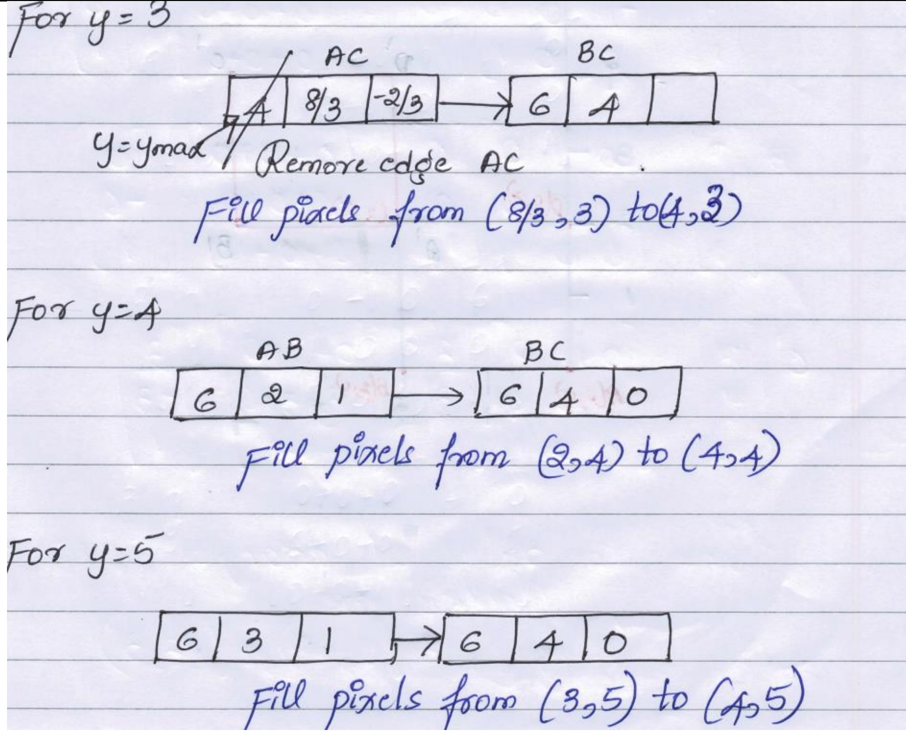
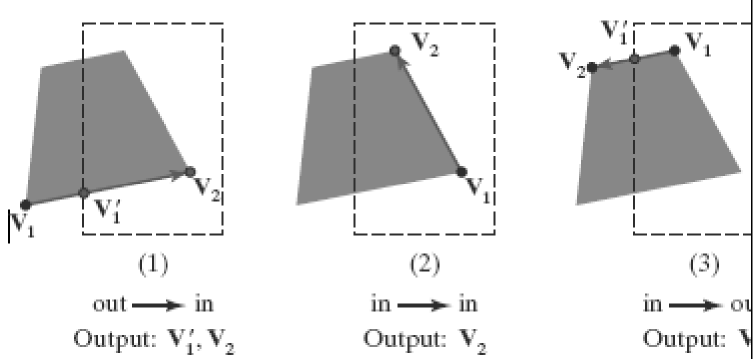
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

IAT-2 Solution
CCA – 18CS62-Computer Graphics
6th Sem CSE - 2021

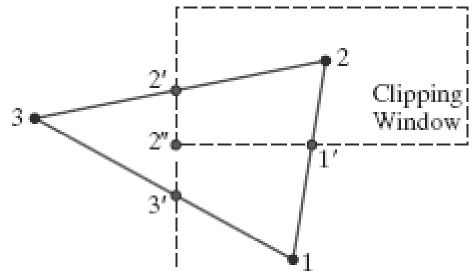
QN	Question	Sub-division of marks	Total
1	**Explain algorithms with example used to identify the interior area of a polygon.** • **Inside Outside** • **Non-Zero Unwinding Number** ***Inside-Outside Tests*** • Also called the *odd-parity rule* or the *even-odd rule.* • Draw a line from any position P to a distant point outside the coordinate extents of the closed polyline • Then we count the number of line-segment crossings along this line. • If the number of segments crossed by this line is odd, then P is considered to be an *interior* point Otherwise, P is an *exterior* point • We can use this procedure, for example,to fill the interior region between two concentric circles or two concentric polygons with a specified color. ***Nonzero Winding-Number rule*** • This counts the number of times that the boundary of an object “winds” around a particular point in the counterclockwise direction termed as winding number, • Initialize the winding number to 0 and again imagining a line drawn from any position P to a distant point beyond the coordinate extents of the object. • The line we choose must not pass through any endpoint coordinates.	 • 5 • 5	10

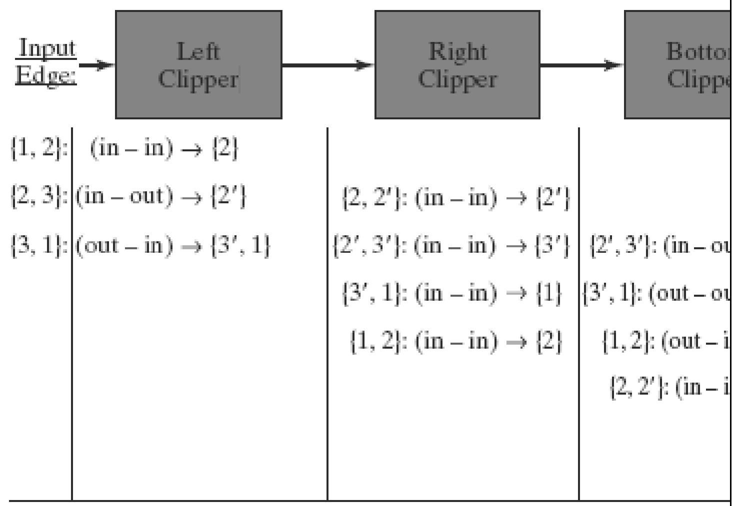
	• As we move along the line from position P to the distant point, we count the number of object nine segments that cross the reference line in each direction • We add 1 to the winding number every time we intersect a segment that crosses the line in the direction from right to left, and we subtract 1 very time we intersect a segment that crosses from left to right  Odd-Even Rule Nonzero Winding-Number Rule The nonzero winding-number rule tends to classify as interior some areas that the odd even rule deems to be exterior. Variations of the nonzero winding-number rule can be used to define interior regions in other ways define a point to be interior if its winding number is positive or if it is negative; or we could use any other rule to generate a variety of fill shapes		
2	**Apply and Illustrate the scanline fill algorithm to fill the polygon with end points (2,4) (4,6) and (4,1)**  For y=1 ET. AC [4 | (4) | -2/3] → BC [6 | (4) | 0] fill pixels from (4,1) to (4,1) For y=2 add slope to (4) 4−2/3 = 10/3 AC [4 | (10/3) | -2/3] → BC [6 | (4) | 0] fill pixels from (10/3, 2) to (4,2)	• 10	10

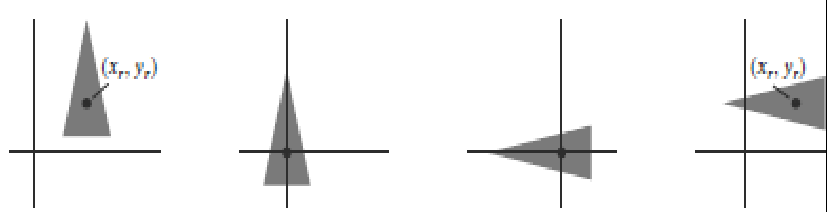
<table>
<tr><td></td><td>
</td><td></td><td></td></tr>
<tr><td>3</td><td>What is Clipping? Explain with an example sutherland-hodgeman polygon clipping algorithm
• Definition
• Clipping

There are four possible cases that need to be considered when processing a polygon edge against one of the clipping boundaries.
• One possibility is that the first edge endpoint is outside the clipping boundary and the second endpoint is inside.
• Or, both endpoints could be inside this clipping boundary.
• Another possibility is that the first endpoint is inside the clipping boundary and the second endpoint is outside.
• And, finally, both endpoints could be outside the clipping boundary
• To facilitate the passing of vertices from one clipping stage to the next, the output from each clipper can be formulated as shown in Figure below.

</td><td>

• 2
• 8</td><td>10</td></tr>
</table>

<table>
<tr><td></td><td>Example:



</td><td></td><td></td></tr>
<tr><td>4</td><td>Obtain matrix representation for rotation of a object about an arbitrary axis in 3D.

When a graphics package provides only a rotate function with respect to the coordinate origin, we can generate a two-dimensional rotation about any other pivot point (xr , yr) by performing the following sequence of translate-rotate-translate operations:

(a) Original Position of Object and Pivot Point
(b) Translation of Object so that Pivot Point (x_r, y_r) is at Origin
(c) Rotation about Origin
(d) Translation of Object so that the Pivot Point is Returned to Position (x_r, y_r)

Translate the object so that the pivot-point position is moved to the coordinate origin.</td><td>• 10</td><td>10</td></tr>
</table>

<table>
<tr>
<td></td>
<td>

Rotate the object about the coordinate origin.

Translate the object so that the pivot point is returned to its original position.

$$\begin{bmatrix} 1 & 0 & x_r \\ 0 & 1 & y_r \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -x_r \\ 0 & 1 & -y_r \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos\theta & -\sin\theta & x_r(1-\cos\theta) + y_r \sin\theta \\ \sin\theta & \cos\theta & y_r(1-\cos\theta) - x_r \sin\theta \\ 0 & 0 & 1 \end{bmatrix}$$

This transformation sequence is illustrated in Figure 9. The composite transformation matrix for this sequence is obtained with the concatenation which can be expressed in the form

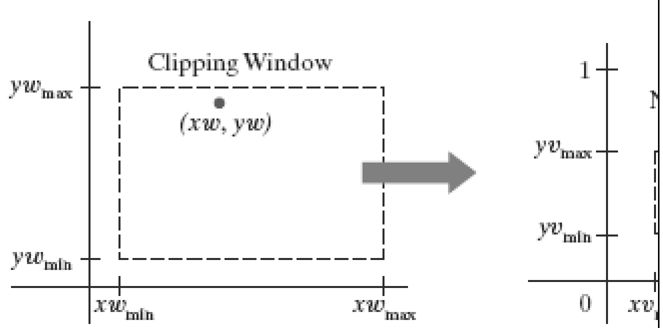
$$\mathbf{T}(x_r, y_r) \cdot \mathbf{R}(\theta) \cdot \mathbf{T}(-x_r, -y_r) = \mathbf{R}(x_r, y_r, \theta)$$

In general, a rotate function in a graphics library could be structured to accept parameters for pivot-point coordinates, as well as the rotation angle, and to generate automatically the rotation matrix of equation.

</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>

Design a transformation matrix for window to viewport transformation.

- We first consider a viewport defined with normalized coordinate values between 0 and 1.
- Object descriptions are transferred to this normalized space using a transformation that maintains the same relative placement of a point in the viewport as it had in the clipping window Position (xw, yw) in the clipping window is mapped to position (xv, yv) in the associated viewport.



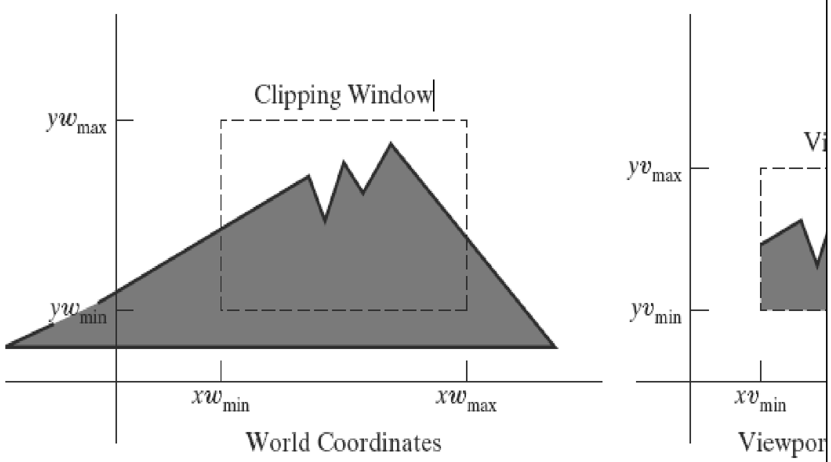
</td>
<td>

- 10

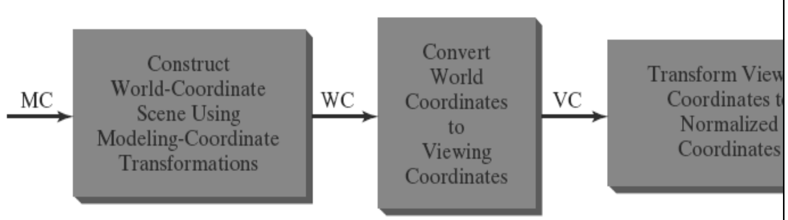
</td>
<td>10</td>
</tr>
</table>

	$\dfrac{xv - xv_{\min}}{xv_{\max} - xv_{\min}} = \dfrac{xw - xw_{\min}}{xw_{\max} - xw_{\min}}$ $\dfrac{yv - yv_{\min}}{yv_{\max} - yv_{\min}} = \dfrac{yw - yw_{\min}}{yw_{\max} - yw_{\min}}$ Solving these expressions for the viewport position (xv, yv), we have xv = sxxw + tx yv = syyw + ty Where, $s_x = \dfrac{xv_{\max} - xv_{\min}}{xw_{\max} - xw_{\min}}$ $s_y = \dfrac{yv_{\max} - yv_{\min}}{yw_{\max} - yw_{\min}}$ $t_x = \dfrac{xw_{\max}xv_{\min} - xw_{\min}xv_{\max}}{xw_{\max} - xw_{\min}}$ $t_y = \dfrac{yw_{\max}yv_{\min} - yw_{\min}yv_{\max}}{yw_{\max} - yw_{\min}}$ We could obtain the transformation from world coordinates to viewport coordinates with the following sequence: • Scale the clipping window to the size of the viewport using a fixed-point position of (xwmin, ywmin). • Translate (xwmin, ywmin) to (xvmin, yvmin). $\mathbf{S} = \begin{bmatrix} s_x & 0 & xw_{\min}(1 - s_x) \\ 0 & s_y & yw_{\min}(1 - s_y) \\ 0 & 0 & 1 \end{bmatrix}$ $\mathbf{T} = \begin{bmatrix} 1 & 0 & xv_{\min} - xw_{\min} \\ 0 & 1 & yv_{\min} - yw_{\min} \\ 0 & 0 & 1 \end{bmatrix}$ $\mathbf{M}_{\text{window, normviewp}} = \mathbf{T} \cdot \mathbf{S} = \begin{bmatrix} s_x & 0 & t_x \\ 0 & s_y & t_y \\ 0 & 0 & 1 \end{bmatrix}$		
6	**What is 2D Viewing? With the help of diagram explain 2D viewing pipeline architecture** • **Viewing** • **Pipeline Architecture** • A section of a two-dimensional scene that is selected for display is called a clipping Window.	• 2 • 8	10

- Sometimes the clipping window is alluded to as the world window or the viewing window
- Graphics packages allow us also to control the placement within the display window using another “window” called the viewport.
- The clipping window selects what we want to see; the viewport indicates where it is to be viewed on the output device.
- By changing the position of a viewport, we can view objects at different positions on the display area of an output device
- Usually, clipping windows and viewports are rectangles in standard position, with the rectangle edges parallel to the coordinate axes.
- We first consider only rectangular viewports and clipping windows, as illustrated in Figure.



-
- Viewing Pipeline



-
- The mapping of a two-dimensional, world-coordinate scene description to device coordinates is called a two-dimensional viewing transformation.
- This transformation is simply referred to as the window-to-viewport transformation or the windowing transformation
- We can describe the steps for two-dimensional viewing as indicated in Figure Once a world-coordinate scene has been constructed, we could set up a separate two dimensional,

	viewing coordinate reference frame for specifying the clipping window. • To make the viewing process independent of the requirements of any output device, graphics systems convert object descriptions to normalized coordinates and apply the clipping routines. • Systems use normalized coordinates in the range from 0 to 1, and others use a normalized range from −1 to 1. • At the final step of the viewing transformation, the contents of the viewport are transferred to positions within the display window. • Clipping is usually performed in normalized coordinates. • This allows us to reduce computations by first concatenating the various transformation matrices.		
7	• **Briefly explain the three different classification of light-material interaction** • **What are different light sources.** Light can leave a surface through two fundamental processes; self-emission and reflection. Each point(x,y,z) on the surface can emit light that contains the direction of emission(θ,ϕ) and the intensity of energy emitted at each wavelength λ.Thus, a general light source can be represented by a six-variable illumination function I(x,y,z,θ,ϕ,λ). There are mainly five basic types of sources, • Color sources • Ambient light • Point sources • Spot light • Distant light sources **Color sources:** Not only light sources emit different amounts of light at different frequencies but their directional properties can vary with frequency. Human visual system is based on 3 color theory that tells us we perceive 3-tristimulus values rather than a full color distribution. For most applications we can thus model light sources as having 3 components-red, green and blue and can use each of the 3 color sources to obtain the corresponding color component that a human observer sees. Thus the three component intensity or illuminating function is	• 5 • 5	10

given by L=(Lr,Lg,Lb), where each component is individual intensities of red, green and blue respectively.

Ambient light: We can look at the derived effect of the sources to achieve a uniform light level in the room. This uniform lighting is called ambient light. It is possible to calculate ambient intensity at each point in the environment. Thus ambient illumination is characterized by intensity Ia that is identical at every point in the scene. Ambient source has red, green and blue components Lar,Lag and Lab. Ambient light depends on the color of the light sources in the environment. For example, a blue light bulb in a white room creates blue ambient light.

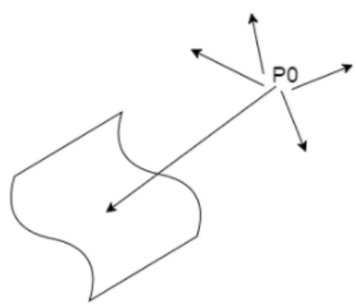
Point sources: An ideal point source emits light in all directions equally. The source located at a point p0p0 by a three-component color function is as follows,

L(p0)=(Lr(p0),Lg(p0),Lb(p0))

Here L(p0)L(p0) is used to refer to any component.

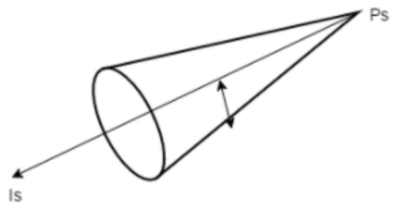
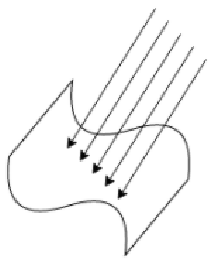
The intensity of illumination received from a point source located at p0p0 at a point p is proportional to the inverse square of the distance from the source.

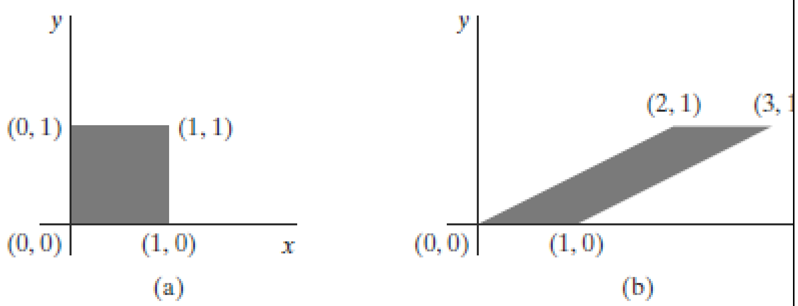
$$L(p, p_0) = \frac{1}{|p - p_0|^2} L(p_0)$$



Spot lights: Spot lights are characterized by narrow range of angles through which light is emitted. A simple spotlight can be constructed from a point source by emitting the angles at which light from the source can be seen. For this a cone is used, whose apex is at Ps, which points in the direction Is, and whose width is determined by angle θ. If θ=180, then the spot light becomes a point source. Always cosines are convenient functions for lighting calculations. Therefore the cosine of the angle θ can be calculated by the dot product of u and v, which are any unit-length vectors as follows,

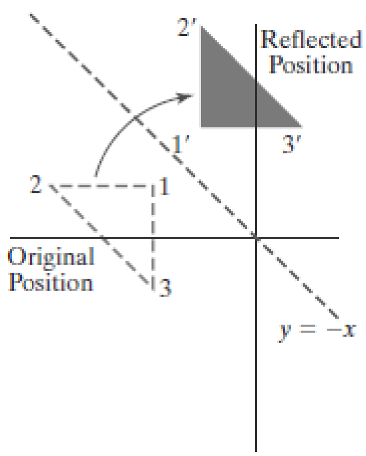
cosθ=u.v

	 **Distant light sources:** Most shading calculations require the direction from the point on the surface to the light source position. As we move across the surface the vectors keep changing.But if the light source is far from the surface, the vector does not change much as we move from point to point. The calculations for distant light sources are similar to calculations for parallel projections, they just replace the location of the light source with the direction of the light source. The figure shows a parallel light source. 		
8	• **Explain 3D Translation, Scaling and Rotation in a homogenous coordinate system** • **Write a note on inverse transformation and reflection** **Shear** A transformation that distorts the shape of an object such that the transformed shape appears as if the object were composed of internal layers that had been caused to slide over each other is called a **shear.** Two common shearing transformations are those that shift coordinate *x* values and those that shift *y* values. $$\begin{bmatrix} 1 & sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$ $$x' = x + sh_x \cdot y, \qquad y' = y$$	• 5 • 5	10

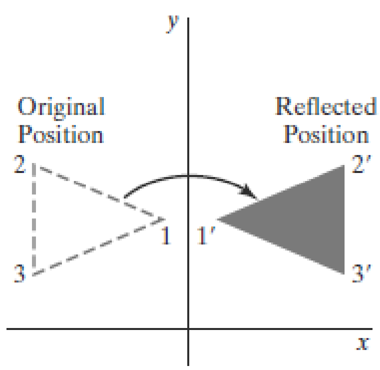


Reflection

A transformation that produces a mirror image of an object is called a reflection. For a two-dimensional reflection, this image is generated relative to an axis of reflection by rotating the object 180◦ about the reflection axis. We can choose an axis of reflection in the xy plane or perpendicular to the xy plane. When the reflection axis is a line in the xy plane, the rotation path about this axis is in a plane perpendicular to the xy plane. For reflection axes that are perpendicular to the xy plane, the rotation path is in the xy plane. Some examples of common reflections follow.



$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$