Sample Final Solutions

Question Consider generalizing Cohen-Sutherland clipping algorithm to a clipping window that is not an upright rectangle. Specifically, consider the scenarios where you have a clipping window which is (a) a convex polygon of n sides and (b) a concave polygon of n sides. Is that possible to generalize the outcode mechanism for trivial-accept and trivial-reject cases in the standard Cohen-Sutherland algorithm? If so, provide a concise description of how it works, if not, give an counter example.

Question 1:

Check out here if you are not familiar with the Cohen-Sutherland clipping algorithm.

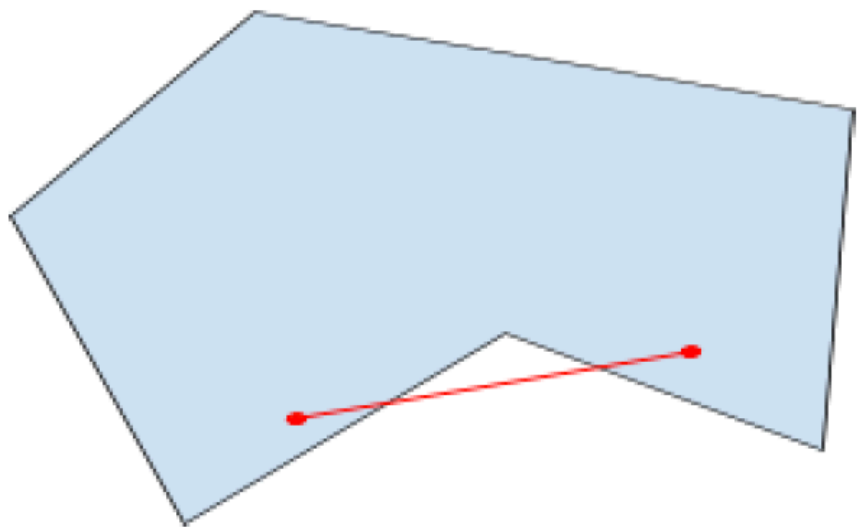
(a) Yes. Similar to the rectangle, extend each side of the convex polygon and define an inside half-space and an outside half-space for each side. Define an n-bit code, one bit per side. For each side, if the created region is located in the outside half-space, set the corresponding bit as 1, otherwise 0. Therefore, we have

Bitwise OR of endpoints == 0: trivial accept;

Bitwise AND of endpoints != 0: trivial reject.

(But in practice please use Cyrus–Beck algorithm for other convex polygons)

(b) No. For concave polygon, even if both endpoints of the line are inside of the clipping area, it doesn't mean the line is totally inside (see the following counter example).



Question We want to implement yet another 2D operation called tapering. Tapering is an operation that changes the width of an object perpendicular to an axis (the taper axis) passing through the centroid of the object. The change in the object's width is proportional to the distance away from the centroid along the taper axis, and the object becomes wider going in the positive sense of the axis and becomes narrower going in the negative sense of the axis. Given a taper axis direction of (a, b), an object centroid $(\bar{x}, \bar{y})$, and a taper factor t (t pixels change in width for every pixel movement away from the centroid along the taper axis), find the matrix in homogeneous coordinates that, when multiplied with a point with coordinate $[x, y, 1]^T$, generates the tapered version of that point. (**NOTE:** If the matrix is composed of several component transforms, you do not need to carry out the matrix multiplication to compose them. Also, you can leave trigonometric functions such as `sin` and `cos` as they are.)

Question 2:

One possible solution :

(1) Translate the object centroid to the origin;

(2) Rotate the object and make the taper axis have the same direction with positive x-axis (or y-axis);

(3) Translate each point along y-axis in a way that different points in different quadrants are treated differently. For example, supposed the the coordinate of the point (after (1) and (2)) is (x1, y1), then translate it along y-axis with the offset x1 * t * sign(y1).

(4) Rotate it back, making the taper axis the original direction (a, b).

(5) Translate the centroid back to the original position.

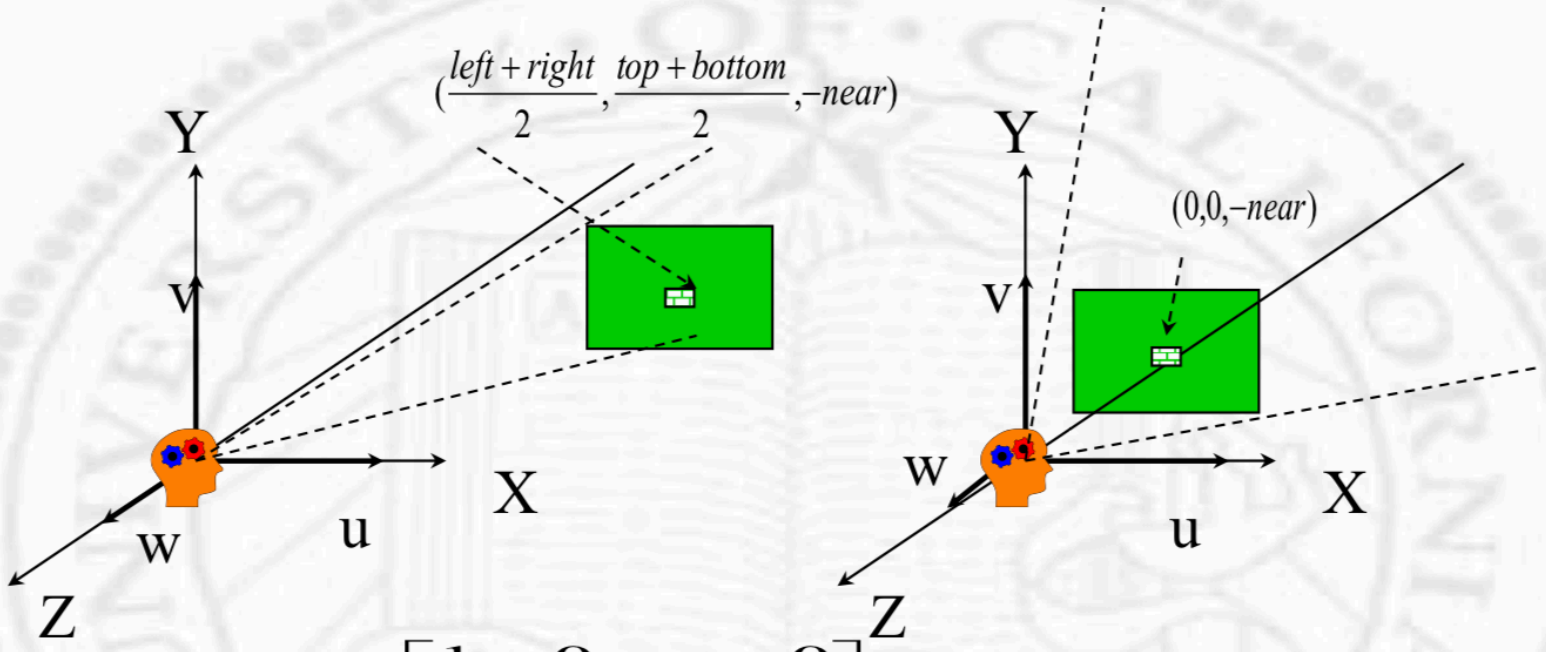
I believe you can easily fill in the concrete matrix operations. :)

Question Suppose in `OpenGL` the user chooses a perspective transformation and uses `glFrustum` to specify the view volume. The parameters to `glFrustum` are

left	5	view volume left edge
right	10	view volume right edge
top	-10	view volume top edge
bottom	-20	view volume bottom edge
near	4	view volume near clipping plane distance
far	10	view volume far clipping plane distance

Suppose that the viewer is at the origin and looks down along the $-z$ axis. It should be apparent that the view volume specified above is skewed to the right and downward from the $-z$ axis. Suppose we would like to center the view volume (and all the scene objects) around the $-z$ axis without changing the depth (z) of the scene objects. (Or this is the *shear* operation used in `PHIGS`.) Derive a transformation matrix in homogeneous coordinate which, when multiplied with an object point, will bring the point in the original view volume into a view volume which is centered around the $-z$ axis.

Shear



$$SH = \begin{bmatrix} 1 & 0 & a & 0 \\ 0 & 1 & b & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$a = \frac{left + right}{2 \, near}, b = \frac{top + bottom}{2 \, near}$$