CS480/CS680 Problem Set 2

Model Solution

1.

a) An answer:

```
// Algorithm to determine whether polygon vertices are given
// in CW or CCW order for a 2D simple and non-degenerate polygon

Input:
v[0], ..., v[N-1]       // N polygon vertices
Point p                 // point inside polygon

Output: True or False         // True if CCW, False if CW

// Find right and top sides of minimum bounding rectangle (MBR)
maxX = max(v[0].x, v[1].x, … , v[N-1].x)
maxY = max(v[0].y, v[1].y, … , v[N-1].y)

valid = False // tracks validity of point outside polygon

// Randomly produces a point and checks for general position
while(!valid) {
   // Generate random point outside the MBR and the polygon
   qx = rand(maxX, 2*maxX)
   qy = rand(maxY, 2*maxY)
   Point q = Point(qx,qy)
   ray = q - p          // vector from p to q

   valid = True

   for (j=0; j< N; j++) {
      if(j==N-1)
         k=0
      else
         k=j+1

      // Check if ray runs jth vertex
      if(segsIntersect(p,q,v[j],v[j]))
         valid = False

      // Check if ray is parallel to jth edge
      ej = v[j+1] - v[j]      // vector describing jth edge
      if(ray.x * ej.y - ej.x * ray.y == 0)
         valid = False
   }
}

sum = 0     // Stores current winding number

// Calculate winding number
for (j=0; j<N; j++){
```

```
   if(j==N-1)
      k=0
   else
      k=j+1

   ej = v[j+1] - v[j]   // vector describing jth edge

   if(segsIntersect(p,q, v[j], v[j+1])) {
      if (ray.x * ej.y - ray.y * ej.x > 0)
            sum++
      else
            sum--
   }
}

// If winding number is 1, return True, otherwise it's -1 and False
return (sum > 0)
```

b) An answer:

One may consider the midpoint of any edge and then generate two points on either side of the edge, offset by distance d from the line. If running the above algorithm returns a 1 or -1 with either point, this point is inside the polygon, and we will know the boundary points are given CCW or CW, respectively.

If a 0 is returned for both, then both points are outside the polygon and we will consider points with an offset of d/2. Continue to run the algorithm with iteratively halved offsets until a 1 or -1 is produced. This will have to happen eventually, as our polygon is non-degenerate.

2.

(a) The corresponding 4 x 4 homogeneous transform matrix is:

$$M = \begin{bmatrix} 1 & 0 & -\sqrt{3} & a \\ 0 & -1 & 0 & b+2 \\ \frac{\sqrt{3}}{2} & 0 & \frac{1}{2} & c \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(b) One possible decomposition of transformation matrix M is as follows:

$$M = TSR$$

The matrices are given by:

$$R = \begin{bmatrix} \frac{1}{2} & 0 & -\frac{\sqrt{3}}{2} & 0 \\ 0 & 1 & 0 & 0 \\ \frac{\sqrt{3}}{2} & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos(\frac{\pi}{3}) & 0 & -\sin(\frac{\pi}{3}) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(\frac{\pi}{3}) & 0 & \cos(\frac{\pi}{3}) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$S = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b+2 \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The transformation can be interpreted as a rotation R of $\frac{\pi}{3}$ clockwise about the y-axis (remember right-hand rule), followed by a scaling factor of (2,-1,1), followed by a translation by the vector (*a,b+2,c*).

3.
y-direction shear, example amount: shy = -0.5
reference line, xref = 1

matrix formulation: $\begin{bmatrix} 1 & 0 & 0 \\ shy & 1 & -shy * xref \\ 0 & 0 & 1 \end{bmatrix}$

matrix values: $\begin{bmatrix} 1 & 0 & 0 \\ -0.5 & 1 & 0.5 \\ 0 & 0 & 1 \end{bmatrix}$

Alternately, one can derive this as the composition $T_{out}(1,0)Sh_y(-0.5)T_{in}(-1,0)$.

4.

(a) $\cos\left(\frac{\theta}{2}\right) = -\frac{1}{\sqrt{2}}$, so $\theta = \frac{3\pi}{2}$; and $sin\left(\frac{3\pi}{4}\right)u = \left(\frac{1}{\sqrt{6}}, \frac{1}{\sqrt{6}}, \frac{1}{\sqrt{6}}\right) \rightarrow u = \left(\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}\right)$

(b) $\cos\left(\frac{\theta}{2}\right) = 0$, so $\theta = \pi$; and $sin\left(\frac{\pi}{2}\right)u = u = (0, -1, 0)$

(c) No, they are rotations about different axes.

(d) Two rotations commute if and only if multiplication of their representative quaternions commute. Using scalar-vector notation $q_1=(s_1,\boldsymbol{v_1})$ and $q_2=(s_2,\boldsymbol{v_2})$, we have that $q_1q_2=q_2q_1$ if and only if $\boldsymbol{v_1}\times\boldsymbol{v_2}=\boldsymbol{v_2}\times\boldsymbol{v_1}$. This occurs only when these cross products are 0, which occurs only when the vectors are parallel or one of them is zero. Thus, two rotations commute only when they are about the same axis, or when one of them is the identity rotation (no rotation). If these are not satisfied, they do not commute.
(Technically, we also need to show that we can't have $q_1q_2 = -q_2q_1$, but this is simple to do and left as an exercise).

5.

Step 1: Build a **uvn** coordinate system

$$\mathbf{u} = \frac{u}{|u|}$$

$$\mathbf{v} = \frac{(u_y, -u_x, 0)}{|u|}$$

n = u x v

Step 2: Translate such that **C** coincides with the canonical origin

$$\mathbf{T_{in}} = \begin{bmatrix} 1 & 0 & 0 & -C_x \\ 0 & 1 & 0 & -C_y \\ 0 & 0 & 1 & -C_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Step 3: Transform **uvn** basis to **xyz** basis

$$\mathbf{R_{in}} = \begin{bmatrix} \frac{u_x}{|u|} & \frac{u_y}{|u|} & \frac{u_z}{|u|} & 0 \\ \frac{v_x}{|v|} & \frac{v_y}{|v|} & \frac{v_z}{|v|} & 0 \\ \frac{n_x}{|n|} & \frac{n_y}{|n|} & \frac{n_z}{|n|} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Note: If u, v, and n are unit vectors as derived in step 1, then we do not need to divide the elements of this matrix by the magnitude of the relevant vectors.

Step 4: Scale along the **u**-axis (now coinciding with the x-axis)

$$\mathbf{S} = \begin{bmatrix} S_u & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Step 5: Rotate **xyz** basis to **uvn** basis

$$\mathbf{R_{out}} = \begin{bmatrix} \frac{u_x}{|u|} & \frac{v_x}{|v|} & \frac{n_x}{|n|} & 0 \\ \frac{u_y}{|u|} & \frac{v_y}{|v|} & \frac{n_y}{|n|} & 0 \\ \frac{u_z}{|u|} & \frac{v_z}{|v|} & \frac{n_z}{|n|} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Note: If u, v, and n are unit vectors as derived in step 1, then we do not need to divide the elements of this matrix by the magnitude of the relevant vectors.

Step 6: Translate back such that C is back to its original location

$$\mathbf{T_{out}} = \begin{bmatrix} 1 & 0 & 0 & C_x \\ 0 & 1 & 0 & C_y \\ 0 & 0 & 1 & C_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The required 3D homogeneous transformation matrix is: $M = \mathbf{T_{out}}\ \mathbf{R_{out}}\ \mathbf{S}\ \mathbf{R_{in}}\ \mathbf{T_{in}}$

6.

- Select an origin for the plane coordinate system

  Select any point (x_0, y_0, z_0) s.t. $ax_0 + by_0 + cz_0 = d$ (i.e. any point on the plane). This point will be treated as the origin of the plane coordinate system. To map this point to the xyz canonical origin we need the following translation:

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & -x_0 \\ 0 & 1 & 0 & -y_0 \\ 0 & 0 & 1 & -z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Find the mapping between the canonical xyz and the uvn. WLOG, we assume that one of a,b non-zero.

  Getting an orthogonal basis for the plane:

  u = (a, b, c) the normal to the plane
  v = (-b, a, 0) a vector orthogonal to the plane normal
  n = v x u a vector orthogonal u and v

  Mapping the canonical xyz to uvn:

$$\mathbf{R_{out}} = \begin{bmatrix} \frac{\boldsymbol{n}}{|\boldsymbol{n}|} \mid \frac{\boldsymbol{v}}{|\boldsymbol{v}|} \mid \frac{\boldsymbol{u}}{|\boldsymbol{u}|} & 0 \\ 0 & 1 \end{bmatrix}$$

$$\mathbf{R_{in}} = \mathbf{R_{out}}^{-1} = \mathbf{R_{out}}^{T}$$

- Find the reflection transformation with respect to the canonical xy-plane

$$\mathbf{Q} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Finally, the required transformation matrix is:

$$\mathbf{M} = \mathbf{T}^{-1}\, \mathbf{R_{out}}\, \mathbf{Q}\, \mathbf{R_{in}}\, \mathbf{T}$$