CS3621 Introduction to Computing with Geometry
Drill Exercises – Geometric Transformations and Simple Models

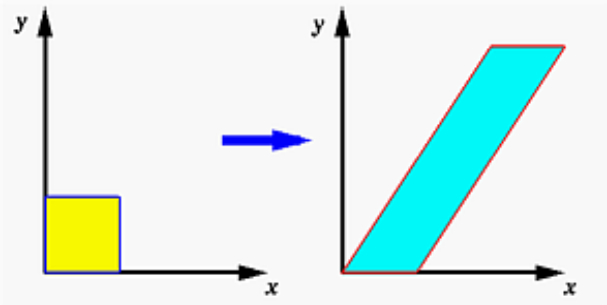
1. Suppose the point $(1, 1, 1)$ is rotated about the x-, then y-, and then z- axis 45°. What are the results? Write down the rotation matrices and the results.

2. Suppose the point $(1, 1, 1)$ is first rotated about the x-axis 45° and then about the z-axis 30°. What is the rotation matrix and the result?

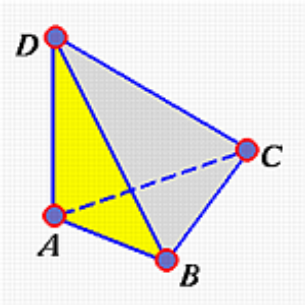
3. Suppose the point $(1, 1, 1)$ is first rotated about the y-axis 45° and then translated in the direction of $\langle 2, 2, 2\rangle$. What is the resulting transformation matrix and the result?

4. Suppose the point $(1, 1, 1)$ is first translated in the direction of $\langle 2, 2, 2\rangle$ and then rotated about the y-axis 45°. What is the resulting transformation matrix and the result? Is this result identical to the one you obtained in the previous problem?

5. An application needs to transform a unit square whose vertices are (0,0), (1,0), (1,1) and (0,1) to a parallelogram whose vertices are (0,0), (1,0), (3,3) and (2,3) as show below. Explain how you will achieve this. Write down each individual transformation and its transformation matrix, compute the composite transformation, and verify your result. Note that the answer is *not* unique.

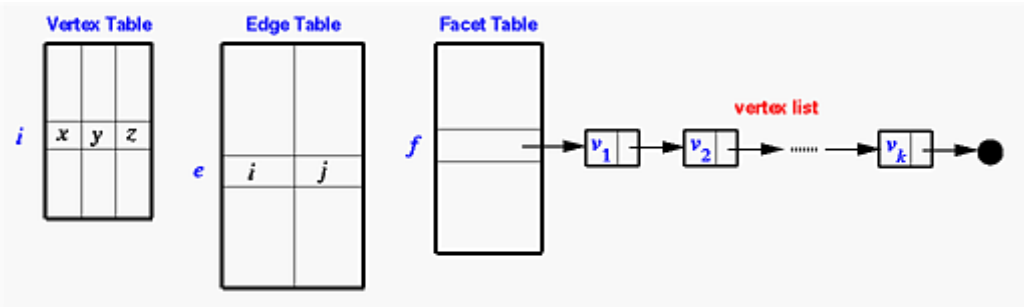


6. Suppose we have a tetrahedron with four vertices $A = (0, 0, 0)$, $B = (1, 0, 0)$, $C = (0, 1, 0)$ and $D = (0, 0, 1)$, and edges AB, BC, AC, DA, DB and DC (see the figure below). The four faces are ABC (bottom), ABD (left), BCD (right) and ACD (back). Design a wireframe representation for this tetrahedron solid.



7. We can extend the wireframe data structure to include face information as shown below. In addition to a vertex table and an edge table, we have a facet table whose f-th entry represents facet f and contains a pointer to a linked list, a *vertex list*. Each node of a vertex list contains a vertex of that facet and the list is built in either clockwise or counter-clockwise order. This representation or its variants are frequently used in graphics programming.

   Answer the following important questions:



(a) Suppose V, E and F are the number of vertices, edges, and facets. Can you come up with a *very* accurate estimation of memory usage? What is the worst case? You can assume one word for each integer, real or pointer.

(b) It is also possible that the nodes of a vertex list store edges rather than vertices. Would this help save space?

(c) Suppose we know that on average each facet has k vertices (*e.g.*, $k = 3$ for triangles). Would this information help?

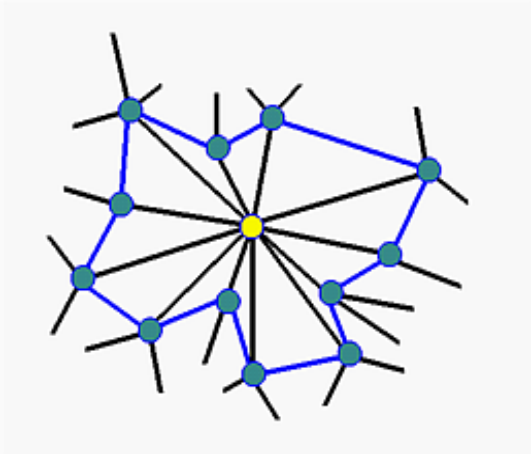
(d) Given a vertex v, develop an algorithm that lists all adjacent edges of v. An edge is adjacent to v if that edge has v as one of its vertices. How many comparisons are required (in worst case)?

(e) Given a vertex v, develop an algorithm that lists all adjacent facets of v. A facet is adjacent to v if that facet has v as its vertex. How many comparisons are required (in worst case)?

(f) Given an edge e, develop an algorithm that lists all adjacent facets of e. A facet is adjacent to e if that facet has e as its edge. How many comparisons are required (in worst case)?

(g) Given a facet f, develop an algorithm that lists all adjacent facets of f. A facet is adjacent to f if they have a common edge or a common vertex. How many comparisons are required (in worst case)?

(h) The concepts of *star* and *link* are very important in polyhedron modeling and editing. Suppose a polyhedron's facets are *all* triangles. Given a vertex, the yellow one in the figure below, the *star* of this vertex consists of *all* triangles that are adjacent to this vertex listed in clockwise or counter-clockwise order. The *link* of this vertex consists of all vertices, listed in either clockwise or counter-clockwise order, that are on the edges adjacent to but different from the given vertex. Note that once we have a properly ordered vertices list, we can connect them together to form a *link* as shown in blue in the figure below.



Develop an algorithm that accepts a vertex and reports the link (in either vertex or edge form) of this vertex. From this link information, you can easily generate the star.

8. Suppose we accept the concept of imaginary objects. Here is something interesting. The general equation of a circle with center (a, b) and radius r is

$$(x - a)^2 + (y - b)^2 = r^2$$

Expanding it gives

$$x^2 + y^2 - 2ax - 2by + (a^2 + b^2 - r^2) = 0$$

Thus, if a general second degree equation represents a circle, this equation does not have the xy term! Because a circle is a special case of an ellipse, a circle intersects the line at infinity[1] in *no* points. Well, if we accept imaginary quantities, a circle or an ellipse intersects the line at infinity in two *imaginary* points. Let us compute these two imaginary points.

Replacing x and y by x/w and y/w, respectively, we have

$$x^2 + y^2 - 2axw - 2byw + (a^2 + b^2 - r^2)w^2 = 0$$

Setting w to zero for computing the intersection points of this circle and the line at infinity, we have

$$x^2 + y^2 = 0$$

Obviously, this equation does not have any meaningful *real* solutions. Note that $(0, 0, 0)$ is *not* a legal point in homogeneous coordinates. But, if we accept imaginary numbers, the above equation has solutions $x = 1$ and $y = \pm i$, where $i = \sqrt{-1}$. Therefore, any circle intersects the line at infinity in two points at infinity $(1, i, 0)$ and $(1, -i, 0)$. Now, we not only use the concept at infinity, but also the concept of *imaginary* numbers! These two points have a special name: the *circular points*.

Given a general second degree equation that represents a conic, if we know this conic intersects the line at infinity at the circular points, can we conclude that this conic is a circle?

[1] The line at infinity consists of all points, in homogeneous coordinates, of form $(x, y, 0)$.