

Angel and Shreiner: Interactive Computer Graphics, Sixth Edition

Chapter 2 Odd Solutions

2.9 We can solve this problem separately in the x and y directions. The transformation is linear, that is $x_s = ax + b$, $y_s = cy + d$. We must maintain proportions, so that x_s is in the same relative position in the viewport as x is in the window, hence

$$\frac{x - x_{min}}{x_{max} - x_{min}} = \frac{x_s - u}{w},$$
$$x_s = u + w \frac{x - x_{min}}{x_{max} - x_{min}}.$$

Likewise

$$y_s = v + h \frac{y - y_{min}}{y_{max} - y_{min}}.$$

2.11 Most practical tests work on a line by line basis. Usually we use scanlines, each of which corresponds to a row of pixels in the frame buffer. If we compute the intersections of the edges of the polygon with a line passing through it, these intersections can be ordered. The first intersection begins a set of points inside the polygon. The second intersection leaves the polygon, the third reenters and so on.

2.13 There are two fundamental approaches: vertex lists and edge lists. With vertex lists we store the vertex locations in an array. The mesh is represented as a list of interior polygons (those polygons with no other polygons inside them). Each interior polygon is represented as an array of pointers into the vertex array. To draw the mesh, we traverse the list of interior polygons, drawing each polygon.

One disadvantage of the vertex list is that if we wish to draw the edges in the mesh, by rendering each polygon shared edges are drawn twice. We can avoid this problem by forming an edge list or edge array, each element is a pair of pointers to vertices in the vertex array. Thus, we can draw each edge once by simply traversing the edge list. However, the simple edge list has no information on polygons and thus if we want to render the mesh in some other way such as by filling interior polygons we must add something to this data structure that gives information as to which edges form each polygon.

A flexible mesh representation would consist of an edge list, a vertex list and a polygon list with pointers so we could know which edges belong to which polygons and which polygons share a given vertex.

2.15 The Maxwell triangle corresponds to the triangle that connects the red, green, and blue vertices in the color cube.

2.19 Consider the lines defined by the sides of the polygon. We can assign a direction for each of these lines by traversing the vertices in a counter-clockwise order. One very simple test is obtained by noting that any point inside the object is on the left of each of these lines. Thus, if we substitute the point into the equation for each of the lines ($ax+by+c$), we should always get the same sign.

2.21 Each of the four tetrahedrons that are created has $1/8$ of the volume of the original tetrahedron. Hence, by keeping only these four and removing the middle volume, the resulting volume is half the original volume. Each of the triangles that we create when we subdivide has $1/4$ the area of the original face. Thus each subtetrahedron has the same area as one of the original faces and in total the surface area of the four subtetrahedrons is the same as the surface area of the original tetrahedron, even though the central section has been removed. If we repeat the calculation for the length of all edges, we find the length of all edges kept after a subdivision is greater than the length of the edges of the original tetrahedron.

2.25 If there is any motion in the display, such as with an animation, then the viewer will notice the blur. However, for static displays, long persistence phosphors give a much smoother display.