5 Poly Objects

Polygons require a more complex type than the other primitive objects because they are variable sized structures. The polygon and polyline structures are similar. However, a polyline structure cannot be filled with a color since it does not necessarily form a closed shape. As you develop your system, you may want to add more fields into your polygon than the required fields listed below.

Polygon fields (C)

- int zBuffer; — whether to use the z-buffer; should default to true (1)
- int numVertex — Number of vertices
- Point *vertex — vertex information
- Vector *normal — surface normal information

Polyline fields (C)

- int zBuffer; — whether to use the z-buffer; should default to true (1).
- int numVertex — Number of vertices
- Point *vertex — vertex information

5.1 Poly Functions

Polygon

- Polygon *Polygon_create() – returns an allocated Polygon pointer initialized so that numVertex is 0 and vertex is NULL.
- Polygon *Polygon_init(int numV, Point *vlist) – returns an allocated Polygon pointer with the vertex list initialized to the points in vlist.
- void Polygon_delete(Polygon *p) – frees the internal data and the Polygon pointer.
- void Polygon_setNULL(Polygon *p) – initializes the existing Polygon to an empty Polygon.
- void Polygon_set(Polygon *p, int numV, Point *vlist) – initializes the vertex list to the points in vlist.
- void Polygon_setAll(Polygon *p, int numV, Point *vlist, Vector *nlist) – initializes the vertices and normals to the given values.
- void Polygon_zBuffer(Polygon *p, int flag) – sets the z-buffer flag to the given value.
- void Polygon_copy(Polygon *to, Polygon *from) – Allocates space and copies the vertex and normal data from one polygon to the other.
- void Polygon_free(Polygon *p) – frees the internal data for a Polygon.
- void Polygon_drawFrame(Polygon *p, Image *src, Pixel c) – draw the outline of the polygon using color c.
• **void Polygon drawFill(Polygon \*p, Image \*src, Pixel c)** – draw the filled polygon using color c. At each pixel the algorithm checks the z-buffer and draws the pixel only if the z-value of the polygon is in front of the existing z-value in the z-buffer. Remember to interpolate \( \frac{1}{z} \), rather than \( z \) when using perspective projection.

• **void Polygon shadeFill(Polygon \*p, Image \*src, Color \*clist)** – draw the filled polygon by interpolating the colors provided for each vertex in the array clist. At each pixel the function should check the z-buffer and draw the pixel only if the z-value is in front of the existing z-value. Interpolate the color as a homogeneous vector \([r, g, b, 1]\) when using perspective projection.

**Polyline**

• **Polyline \*Polyline\_create()** – returns an allocated Polyline pointer initialized so that numVertex is 0 and vertex is NULL.

• **Polyline \*Polyline\_init(int numV, Point \*vlist)** – returns an allocated Polyline pointer with the vertex list initialized to the points in vlist.

• **void Polyline\_delete(Polyline \*p)** – frees the data and the polyline pointer.

• **void Polyline\_setNULL(Polyline \*p)** – initializes the existing Polyline to an empty Polyline.

• **void Polyline\_set(Polyline \*p, int numV, Point \*vlist)** – initializes the vertex list to the points in vlist.

• **void Polyline\_zBuffer(Polyline \*p, int flag)** – sets the z-buffer flag to the given value.

• **void Polyline\_copy(Polyline \*to, Polyline \*from)** – Allocates space and copies the vertex data from one polygon to the others.

• **void Polyline\_free(Polyline \*p)** – frees the internal data for a polyline.

• **void Polyline\_drawFrame(Polyline \*p, Image \*src, Pixel c)** – draw the outline of the polyline using color c. If the zBuffer flag is set, should take into account the z-buffer values when drawing lines.

©2008 Bruce A. Maxwell