5 Polygons

Polygons require a more complex type than the other primitive objects because they are variable sized structures. If you are using C++, polygon structures are a good place to begin using the standard template library [STL]. The polygon and polyline structures are similar. However, a polyline structure cannot be filled since it does not necessarily form a closed shape. You may want to put more fields into your polygon (and you will definitely need to later on), but for now these are the minimum required fields.

**Polygon fields (option 1)**

- 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

**Polygon fields (option 2)**

- int zBuffer; – whether to use the z-buffer; should default to true (1)
- std::vector<Point> – STL vector holding an array of vertices.
- std::vector<Vector> – STL vector holding an array of surface normals.

**Polyline fields (option 2)**

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

**Polyline fields (option 2)**

- int zBuffer; – whether to use the z-buffer; should default to true (1)
- std::vector<Point> – STL vector holding an array of vertices.
5.1 Method Specification

Polygon

- Polygon() – initialize numVertex to 0 and vertex to NULL.
- Polygon(std::vector<Point> &vlist) – initialize the vertex list to the points in vlist.
- void set(std::vector<Point> &vlist) – initialize the vertex list to the points in vlist.
- void set(std::vector<Point> &vlist, std::vector<Vector> &nlist) – initializes the vertices and normals to the given values.
- void zBuffer(int flag) – sets the z-buffer flag to the given value.
- void drawFill(Image *src, Pixel p) – 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 1/2, rather than z when using perspective projection.
- void shadeFill(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 \( \begin{bmatrix} r \\ g \\ b \\ 1 \end{bmatrix} \) when using perspective projection.

Polyline

- Polyline() – initialize numVertex to 0 and vertex to NULL.
- Polyline(std::vector<Point> &vlist) – initialize the vertex list to the points in vlist.
- void set(std::vector<Point> &vlist) – initialize the vertex list to the points in vlist.
- void zBuffer(int flag) – sets the z-buffer flag to the given value.
- void draw(Image *src, Pixel p) – draw the lines defined by the vertex list using color p. If the zBuffer flag is set, should take into account the z-buffer values when drawing lines.