A.4.1 Surface Attributes

Between the index of the last vertex of each face and the semicolon which ends the face, there may be an attribute specification, which is the keyword `attribute` followed by an index into the psurf's attribute table, all delimited by square brackets. By default, the attribute index is 0, and its value carries over from one face to the next, so the attribute specification actually sets the "current" value, to be assigned to all subsequent faces until its value is changed again.

This is only part of the attribute information. The psurf itself does not define the material properties associated with each face, but it does specify which faces are made of the same material. When a psurf is associated with a segment, it will be instantiated with a set of surface attributes. The number of attributes will match the number of attribute indices specified in the psurf file, and the attributes will be associated with the faces according to each face's attribute index. The details of how attributes are associated with faces is described in Section A.3.7.

A.4.2 Face Smoothness

Along with the attribute specification, each face has a *smoothness* flag. This flag tells whether the face models a flat surface like a polyhedron, or whether it is being used to model a small piece of a curved surface. In computer graphics, it is sometimes convenient to model curved surfaces with a mesh of small polygons. The polygons themselves are an artifact of the model, not a geometric property.

When an object is drawn in wireframe, it's drawn with its edges. When it is shaded, its faces are filled in. If a face is flat, it has a constant surface normal across the entire face. If the face is smooth, the surface normal is computed at each of the vertices of the face in terms of the normals of the adjacent faces, and the normal at points in the interior of the face is interpolated from the values at the vertices. This is *phong shading*.

It is important *not* to use smooth shading with psurfs which are not polygon meshes. If a psurf consists of a few big faces and the angles between them are great, then the object will not look right if it is smooth shaded.

A.4.3 Node Coloring

It is also possible to assign rgb color values to individual nodes in a psurf file. The format is to place an rgb value after the node definition in the psurf file. A cube, with 7 white vertices and one red vertex, would look like the following:

```
    0.0     0.0     0.0 [ rgb( 1.0, 1.0, 1.0) ]
    0.0   100.0     0.0 [ rgb( 1.0, 1.0, 1.0) ]
  100.0   100.0     0.0 [ rgb( 1.0, 1.0, 1.0) ]
  100.0     0.0     0.0 [ rgb( 1.0, 1.0, 1.0) ]
    0.0     0.0   100.0 [ rgb( 1.0, 1.0, 1.0) ]
    0.0   100.0   100.0 [ rgb( 1.0, 1.0, 1.0) ]
  100.0   100.0   100.0 [ rgb( 1.0, 0.0, 0.0) ] {the red one}
  100.0     0.0   100.0 [ rgb( 1.0, 1.0, 1.0) ]
;;
1 2 3 4 [smooth] [attribute 0];
1 4 8 5 [smooth] [attribute 0];
3 7 8 4 [smooth] [attribute 0];
1 5 6 2 [smooth] [attribute 0];
5 8 7 6 [smooth] [attribute 0];
2 6 7 3 [smooth] [attribute 0];
;;
```

You can use the command set node rgb to edit the color of a node. Note that you should use make segment smooth to properly blend the node colors across a face, using Gouraud shading. Note that lighting will not effect segments that have node colors. *Jack* uses the node colors to display the results of a radiosity rendering, therefore the lights are disabled in the environment for those segments which have node colors.

A.4.4 Big Psurfs

A psurf file may also have an optional specification `[big]` on the first line of the file. This specification will inhibit *Jack* from generating display lists for the edges of the psurf. This is useful if the psurf is imported from other CAD system that can't break up geometry and must send it as one long list of nodes and faces. *Jack* usually creates edge lists for displaying psurfs in wireframe. This spec is only necessary if the psurf has more then 1000-2000 nodes.

A.4.5 Face Orientation

Psurf faces have a specific orientation, which means that a face really has just one side. When viewing an object in wireframe, this is of no consequence since the object is displayed with its edges. However, when an object is shaded its intensity is defined in terms of the angle between the face and the light sources. The orientation of the face depends upon the order of the vertices. The surface normal is defined to point in the direction given by the right hand rule. This means that a counter-clockwise traversal by the right hand rule yields an outward pointing normal. In other words, place your right hand on the polygon and sweep your fingers around the vertices in the order they occur in the file. Your thumb then points in the direction of the surface normal.

It is imperative that the faces of psurfs be oriented consistently. When objects are shaded in *Jack*, only the side of the surface normal will be shaded correctly. The other side will be effectively shadowed and will be illuminated with ambient light.

Psurfs usually represent collections of polyhedra, so the nodes, edges, and faces form a planar graph[2]. This places some important restrictions on how the faces may be defined:

- Each edge (pair of nodes) should be contained in at most two faces.
- No pair of nodes (an edge) should be contained in two faces in the same order.

This means that if you traverse each face of a psurf, you should never travel along the same edge more than twice, and you should never travel along the same edge twice in the same direction. If you have, the psurf does not define a valid polyhedron, and if you try to shade it, its surface normals are not well-defined.

There are exceptions to these rules, particularly in the case of open objects, i.e. sets of faces which are connected but which do not bound a closed region in space.

A.4.6 Binary Psurfs and BPS

For faster reading and display, you may pre-process psurf files and convert them into "binary psurfs" using the program **bps**. **bps** reads a text psurf file and produces another file with the same base name but with the suffix `.bps`. This is a binary psurf file, and it represents the same geometric information about the psurf in a form which is easier for *Jack* to read and more efficient for it to display.

[2] A planar graph is a graph which is capable of be drawn in a plane without crossing edges. It does *not* mean that all nodes of the graph lie in a plane in space