Polygonal Marching Cubes Organization Diagram
Polygonal Marching Cubes Data Structures

int FoundEdgeIntersection[12]

- One entry for each of the 12 edges.
- A FALSE (0) means S* did not intersect this edge
- A TRUE (!0) means S* did intersect this edge

Node EdgeIntersection[12]

- If an intersection did occur on edge #i, Node[i] will contain the interpolated x, y, z, nx, ny, and nz.

int FoundEdgeConnection[12][12]

- A TRUE in entry [i][j] or [j][i] means that Marching Squares has decided there needs to be a line drawn from Cube Edge #i to Cube Edge #j
- Both entry [i][j] and [j][i] are filled so that it won’t matter which order you search in later
Strategy in **ProcessCube()**: 

1. Use **ProcessCubeEdge()** 12 times to find which cube edges have S* intersections.

2. Return if no intersections found anywhere.

3. Call **ProcessCubeQuad()** 6 times to decide which cube edges will need to be connected. This is just a slightly modified version of doing a Marching Square. This leaves us with the **FoundEdgeConnection[][]** array filled.

4. Call **DrawCubeTriangles()** to create triangles from the connected edges.

Strategy in **DrawCubeTriangles()**: 

1. Look through the **FoundEdgeConnection[][]** array for a Cube Edge #a and a Cube Edge #b that have a connection between them.

2. If can’t find one, then you are done with this cube.

3. Now look through the **FoundEdgeConnection[][]** array for a Cube Edge #c that is connected to Cube Edge #b. If you can’t find one, something is wrong.

4. Draw a triangle using the **EdgeIntersection[]** nodes from Cube Edges #a, #b, and #c. Be sure to use **glNormal3f()** in addition to **glVertex3f()**.

5. Turn to FALSE the **FoundEdgeConnection[][]** entries from Cube Edge #a to Cube Edge #b.

6. Turn to FALSE the **FoundEdgeConnection[][]** entries from Cube Edge #b to Cube Edge #c.

7. **Toggle** the **FoundEdgeConnection[][]** entries from Cube Edge #c to Cube Edge #a. If this connection was there before, we don’t need it anymore. If it was not there before, then we just invented it and we will need it again.