Quad-Edge Data Structure and findComponent function

implementation:
- one data structure per component
  - has pointers to the four adjacent diamonds
  - has pointers to 4 vfs
reference to this structure also includes 3 booleans:
  - dual, edge orientation, surface orientation

equivalently we have:
- rotation (mod 4)
- surface orientation (boolean)

alternatively:
- just one vf pointer (head)
  - links: nfirst, next

```plaintext
e, v, f
```

to allow nonorientable surfaces:
reference to this structure also includes a boolean (surface orientation)

also possible:
Flag:

```plaintext
vcecf
```

implementation issues:
- vertex, face should have pointers to some incident halfedges
should have an *arbitrary* \( r \in \mathbb{R} \)

Boolean field *seen* in vertex, halfedge, face

Find component (halfedge \( h \)). (assuming "seen" flags cleared)
- breadth-first search using \( \{ \) queue \( \} \) of halfedges
- depth-first search using \( \{ \) stack \( \} \) of halfedges

\[
\begin{align*}
\text{int } & V = 0, E = 0, F = 0, B = 0 ; \\
x. & \text{push}(h) \\
\text{while } (x. \text{not empty}) & \\
\{ & h = x. \text{pop}() ; \\
& \text{if } (h. \text{seen}) \\
& \{ & h. \text{seen} = \text{true} ; \\
& & h. \text{opp. seen} = \text{true} ; \\
& & E += 1 ; \\
& & \text{if } (! h. \text{left. seen}) \\
& & \{ & h. \text{left. seen} = \text{true} ; \\
& & & h. \text{left. perforated} ? B += 1 ; F += 1 ; \\
& & \} \\
& & \text{if } (! h. \text{opp. left. seen}) \\
& & \{ & \ldots \\
& & \} \\
& & \text{if } (! h. \text{head. seen}) \\
& & \{ & h. \text{head. seen} = \text{true} ; \\
& & & V += 1 ; \\
& & \} \\
& & \text{if } (! h. \text{tail. seen}) \\
& & \{ & \ldots \\
& & \} \\
& & x. \text{push}(h. \text{vnext}) ; \\
& & x. \text{push}(h. \text{fnext}) ; \\
& & \} \\
\end{align*}
\]
\[ \theta = \frac{v - w}{v} \]