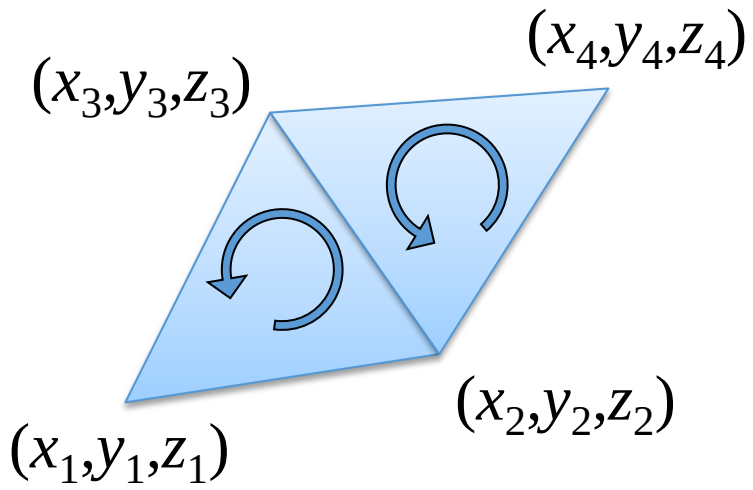
The Half Edge Mesh Representation

CS418 Interactive Computer Graphics

John C. Hart

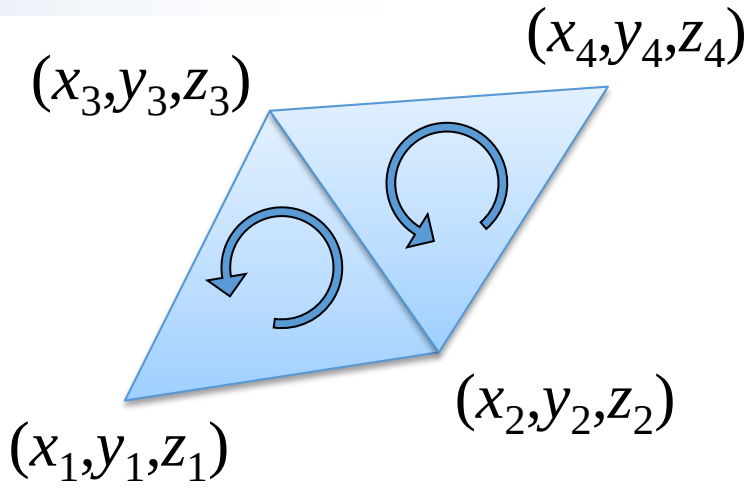
Good Meshes

- **Manifold**: 1. Every edge connects exactly two faces
  2. Vertex neighborhood is “disk-like”
- **Orientable**: Consistent normals
- **Watertight**: Orientable + Manifold
- **Boundary**: Some edges bound only one face
- **Ordering**: Vertices in CCW order when viewed from normal



Indexed Face Set

- Popular file format
  - VRML, Wavefront ".obj", etc.
- Ordered list of vertices
  - Prefaced by "v" (Wavefront)
  - Spatial coordinates x,y,z
  - Index given by order
- List of polygons
  - Prefaced by "f" (Wavefront)
  - Ordered list of vertex indices
  - Length = # of sides
  - Orientation given by order



v x_1 y_1 z_1
v x_2 y_2 z_2
v x_3 y_3 z_3
v x_4 y_4 z_4

f 1 2 3
f 2 4 3

Catmull-Clark Subdivision

- **Face vertex**

  average of face's vertices

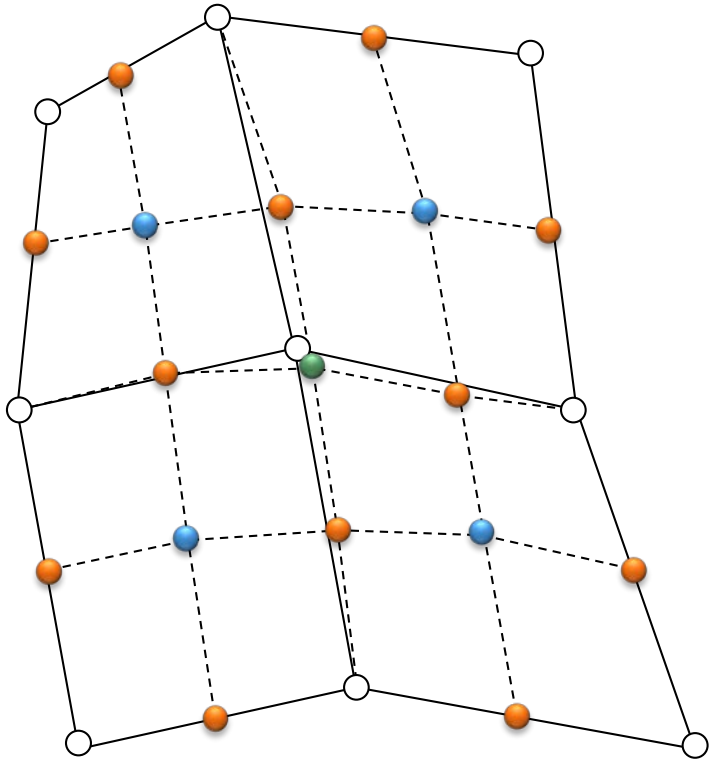
- **Edge vertex**

  average of edge's two vertices
  & adjacent face's two vertices

- **New vertex position**

  (1/valence) x sum of…

  - Ave. neighboring face points
  - 2 x ave. of edge points
  - (valence – 3) x original

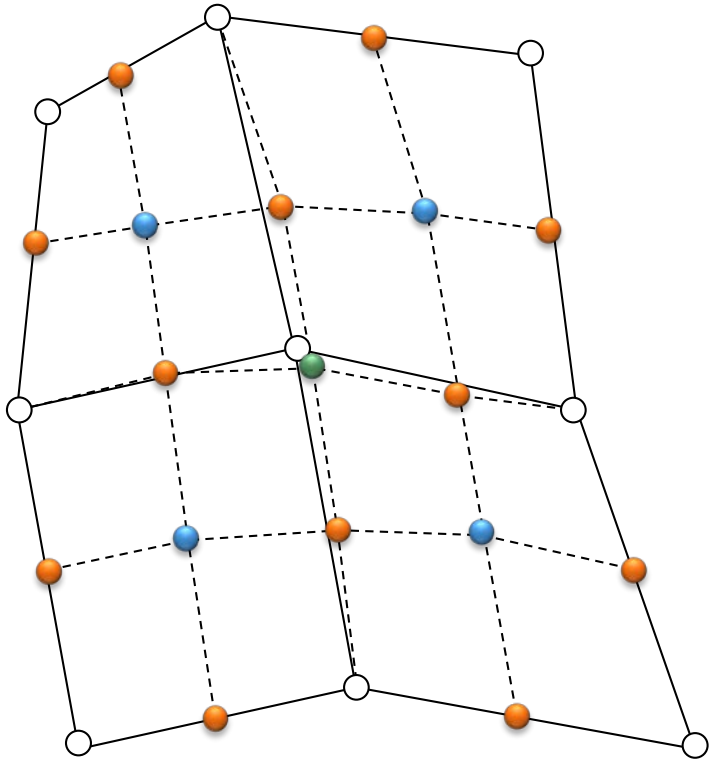


Implementation

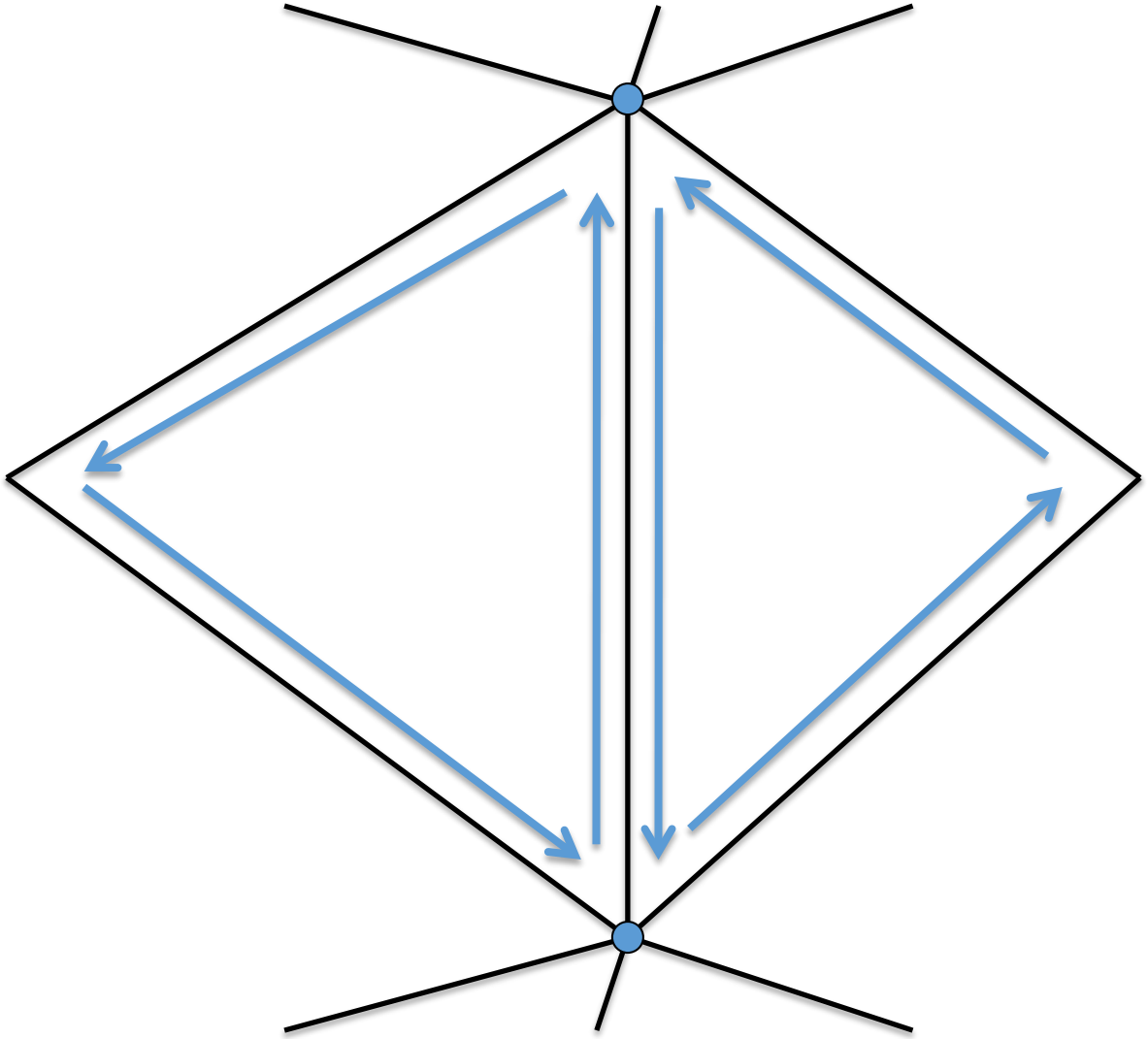
- **Face vertex**
  - For each face add vertex at centroid
- **Edge vertex**
  - How do we find each edge?
- **New vertex position**
  - For a given vertex how do we find neighboring faces and edges?

v x_1 y_1 z_1
v x_2 y_2 z_2
v x_3 y_3 z_3
v x_4 y_4 z_4
…

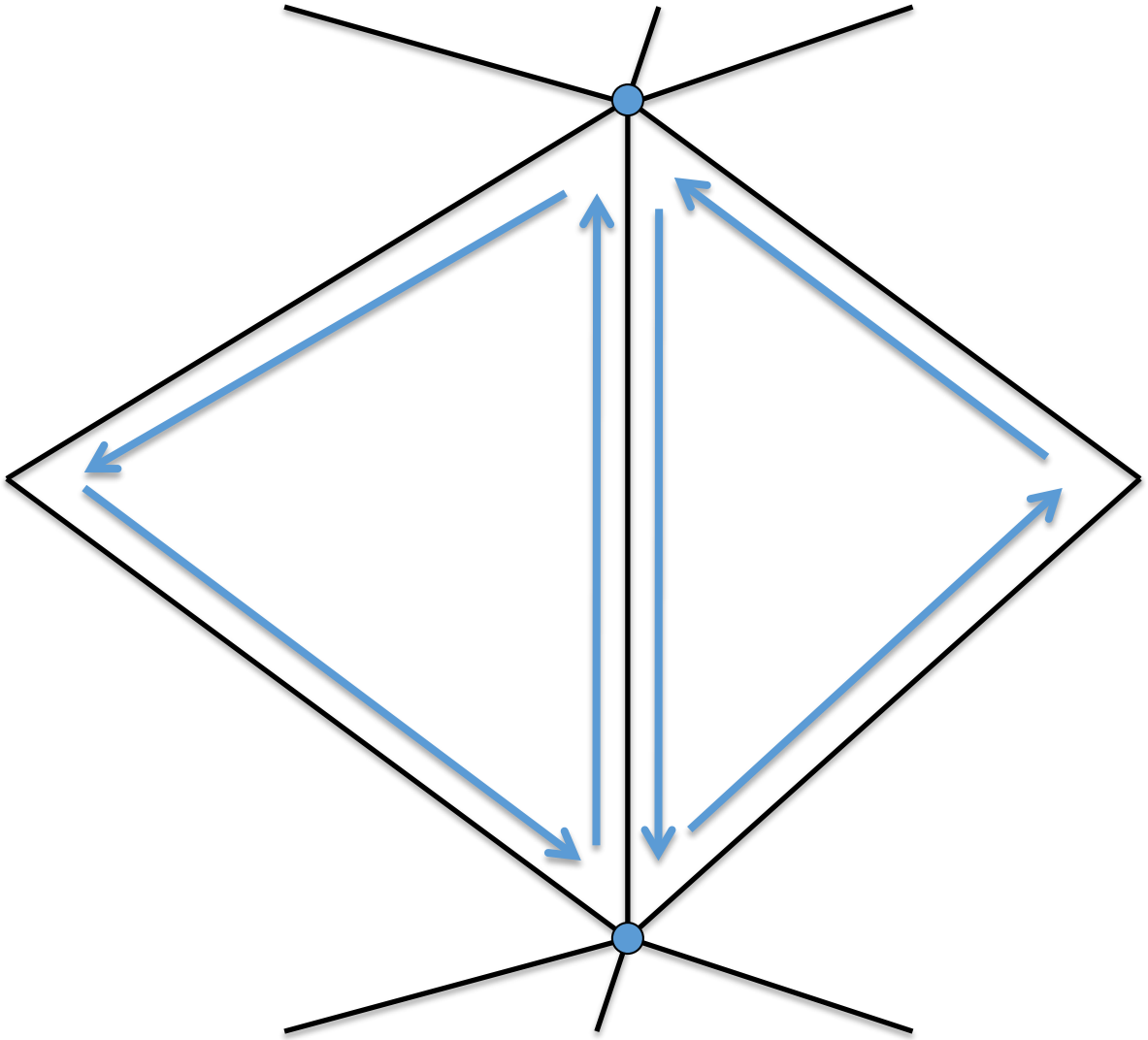
f 1 2 3 4
...



Half Edge

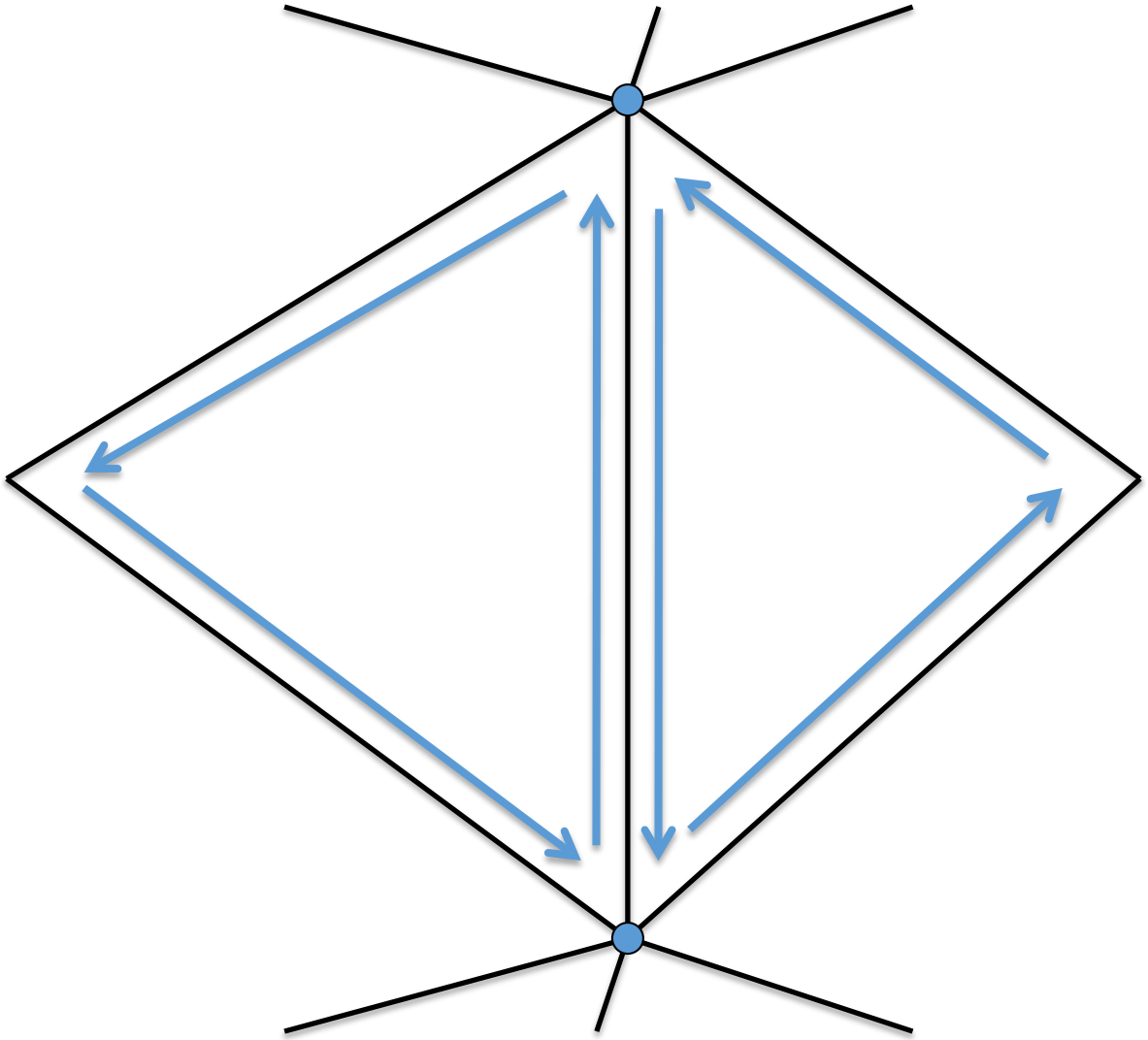


Half Edge



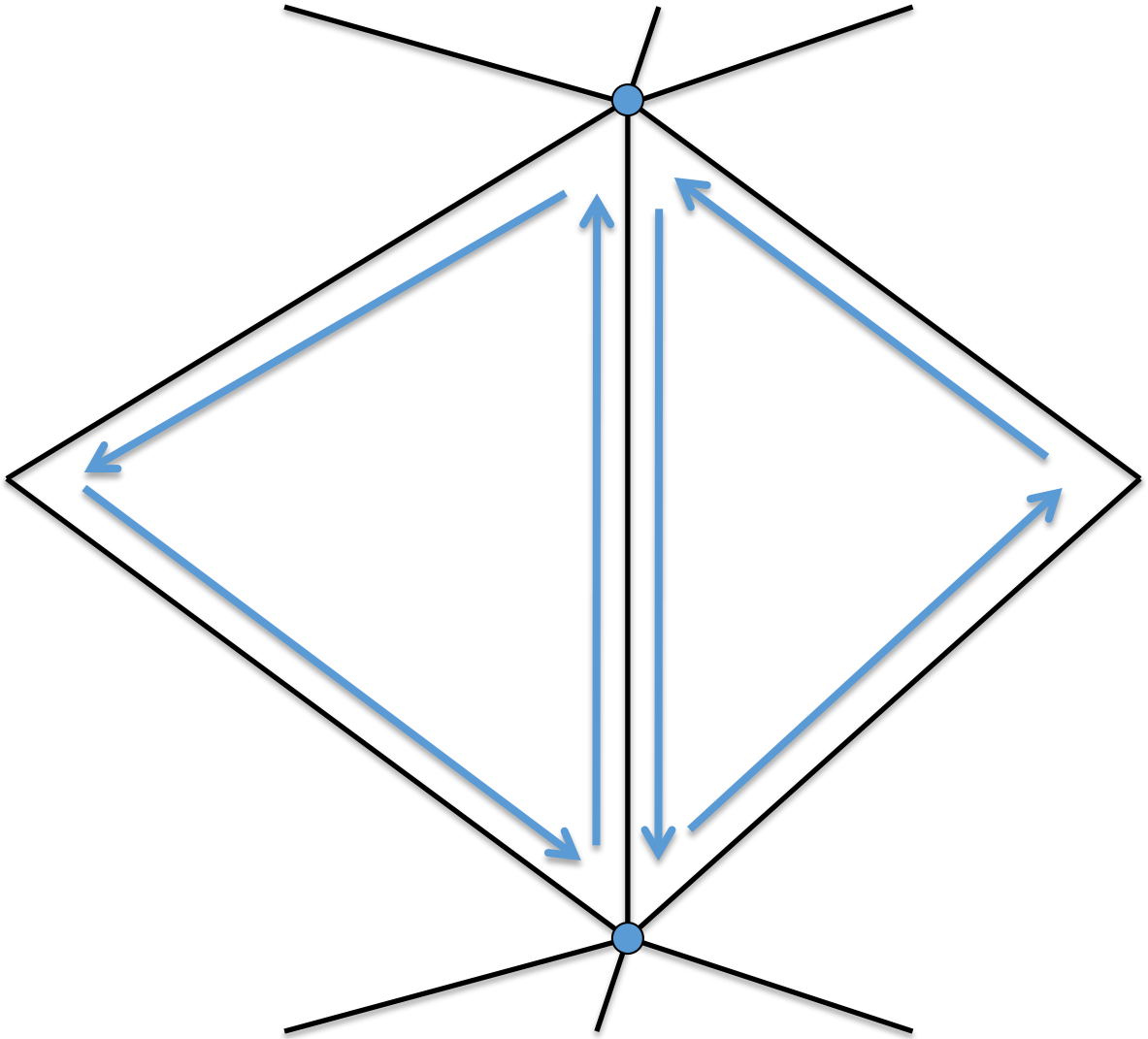
```
class HalfEdge {
  HalfEdge *opp;
  Vertex   *end;
  Face     *left;
  HalfEdge *next;
};

HalfEdge e;
```

Half Edge



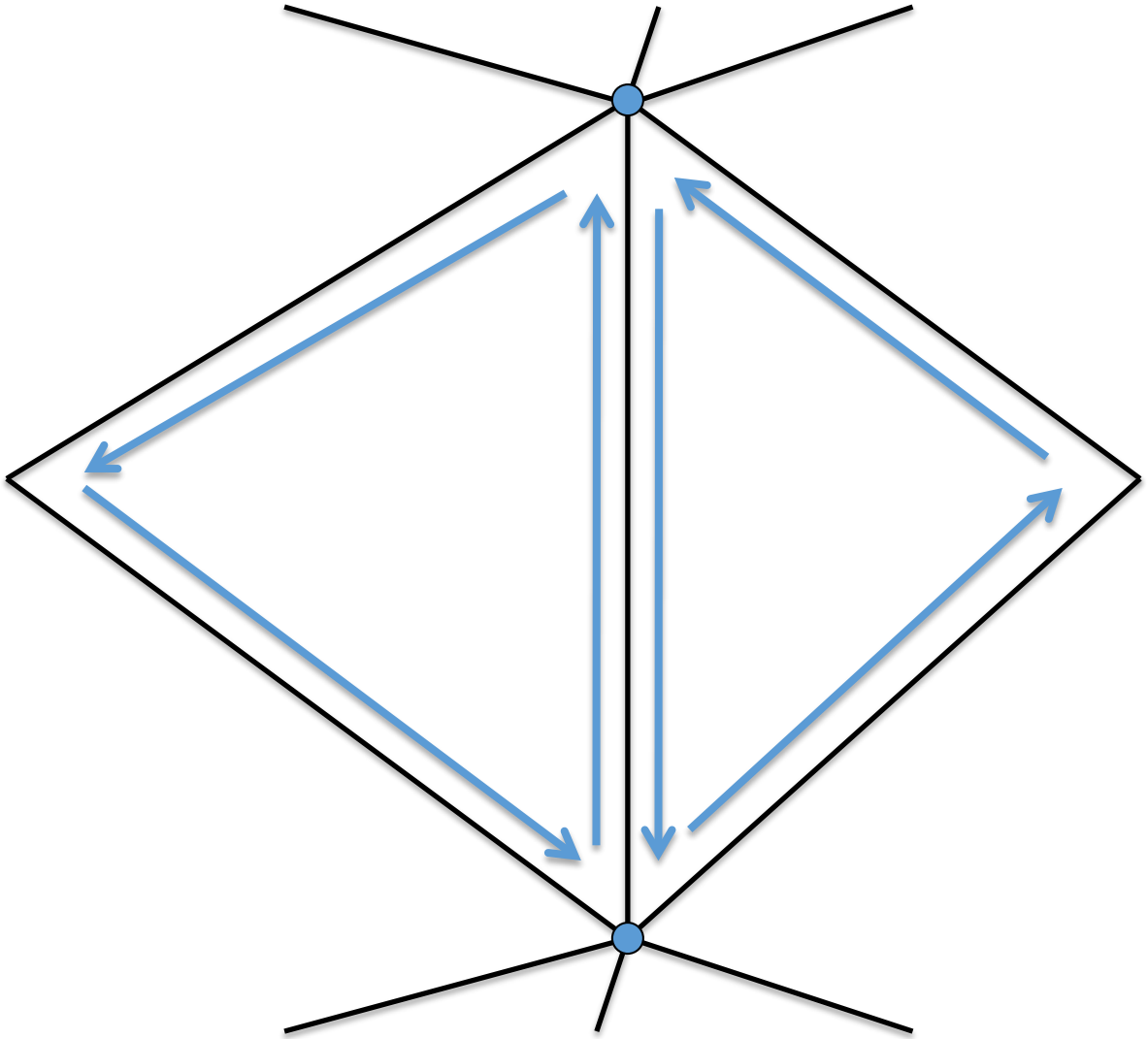
```
class HalfEdge {
  HalfEdge *opp;
  Vertex   *end;
  Face     *left;
  HalfEdge *next;
};

HalfEdge e;
```

Half Edge



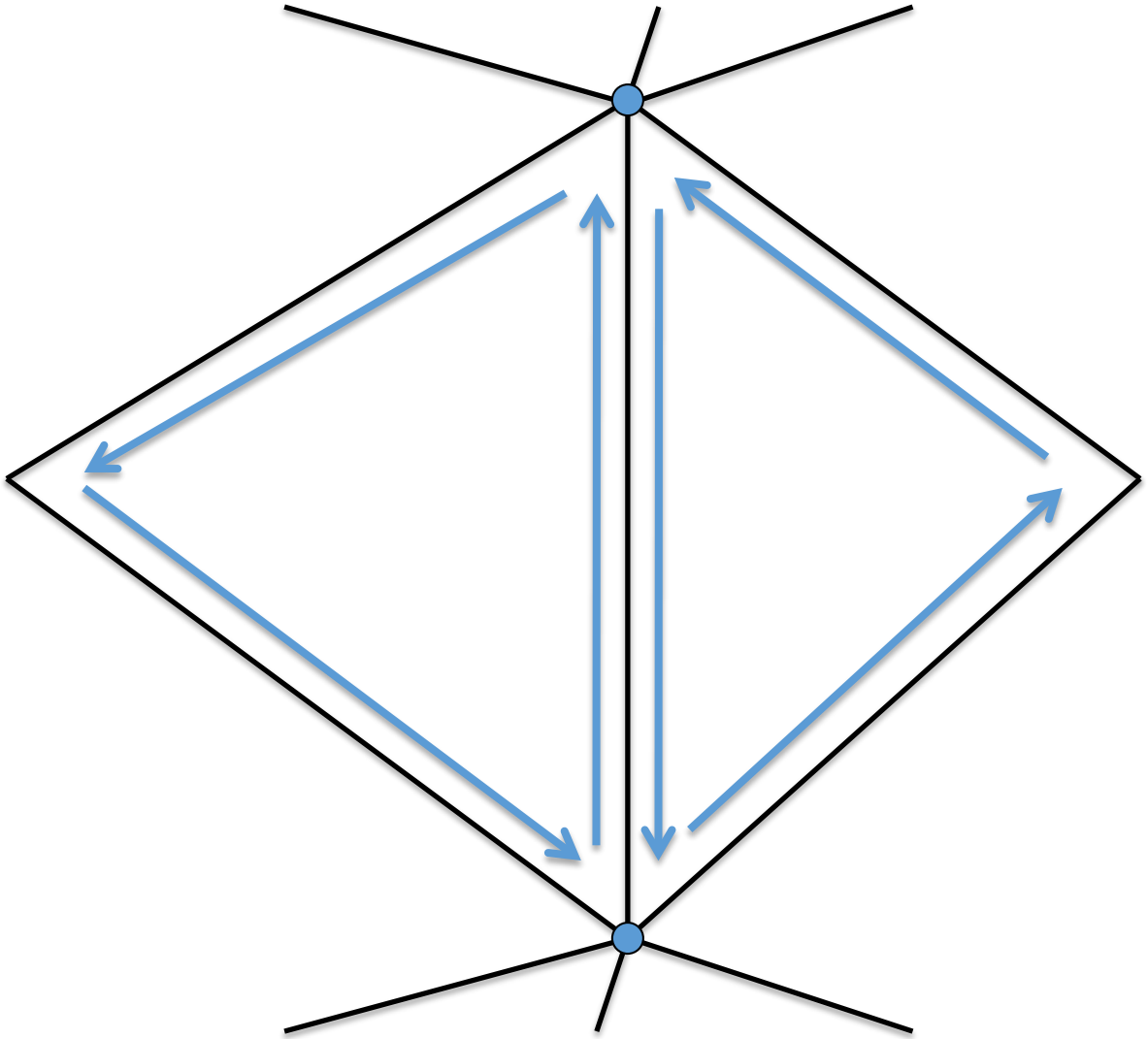
```
class HalfEdge {
  HalfEdge *opp;
  Vertex   *end;
  Face     *left;
  HalfEdge *next;
};

HalfEdge e;
```

Half Edge

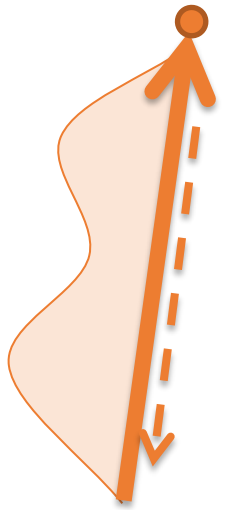


```
class HalfEdge {
  HalfEdge *opp;
  Vertex   *end;
  Face     *left;
  HalfEdge *next;
};

HalfEdge e;
```

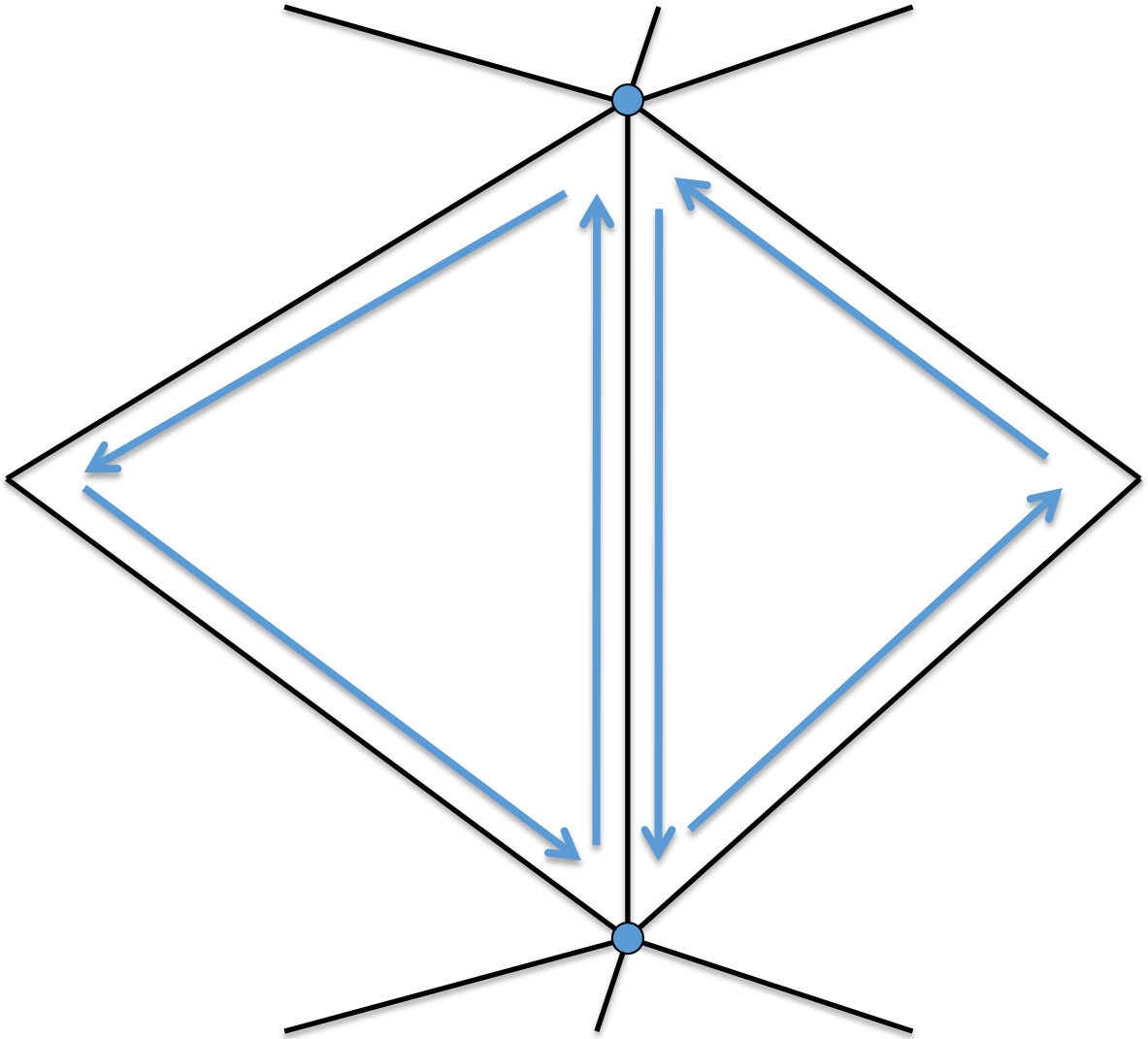
Half Edge



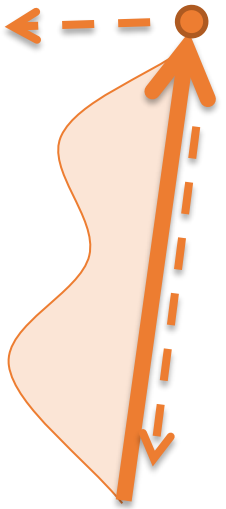
```
class HalfEdge {
  HalfEdge *opp;
  Vertex   *end;
  Face     *left;
  HalfEdge *next;
};

HalfEdge e;
```



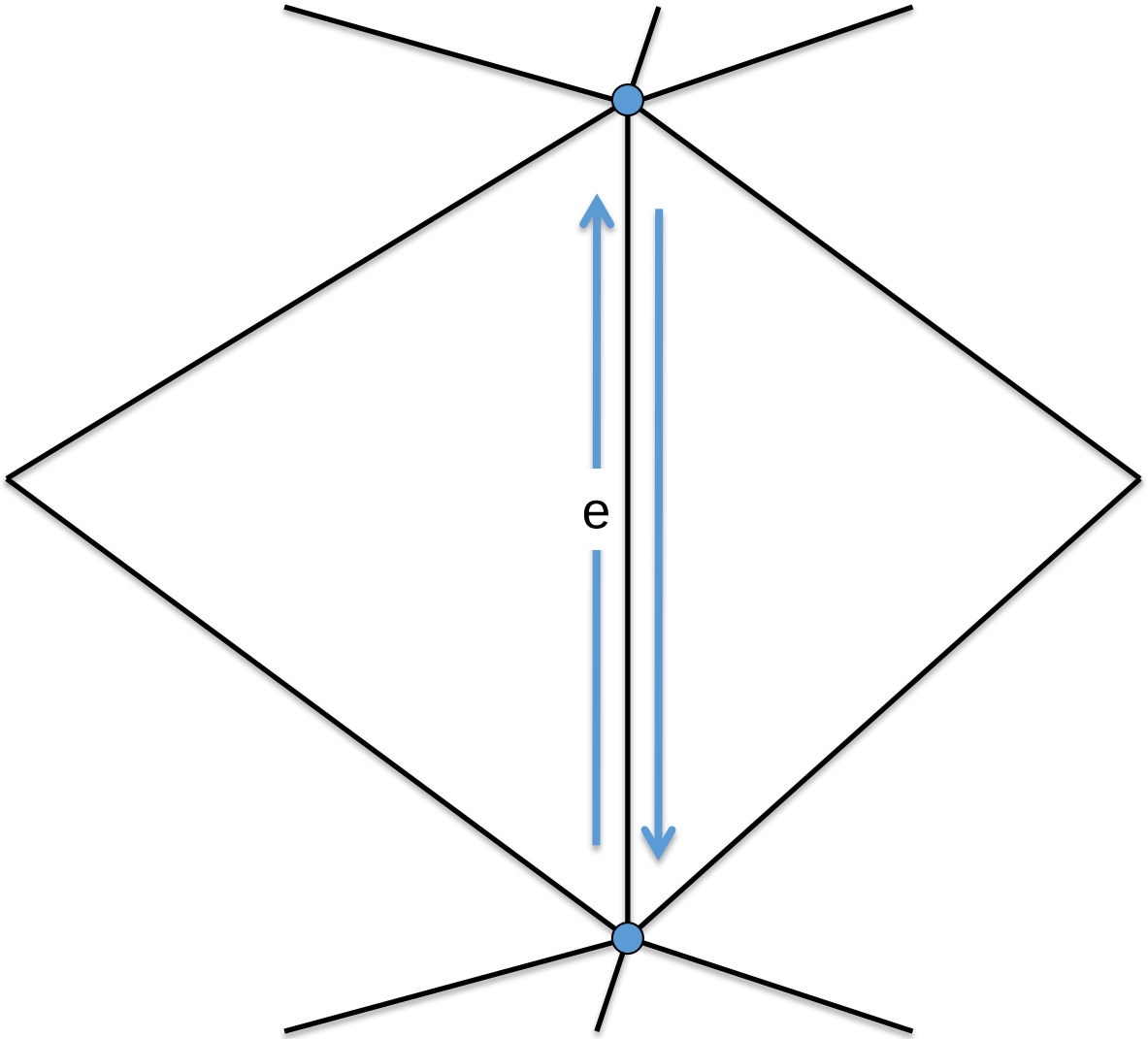
Half Edge



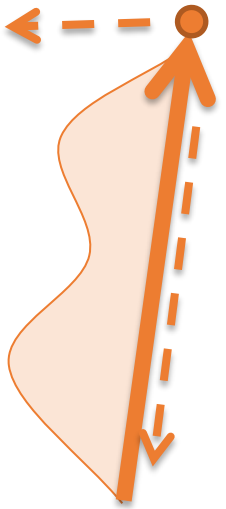
```
class HalfEdge {
  HalfEdge *opp;
  Vertex   *end;
  Face     *left;
  HalfEdge *next;
};

HalfEdge e;
```



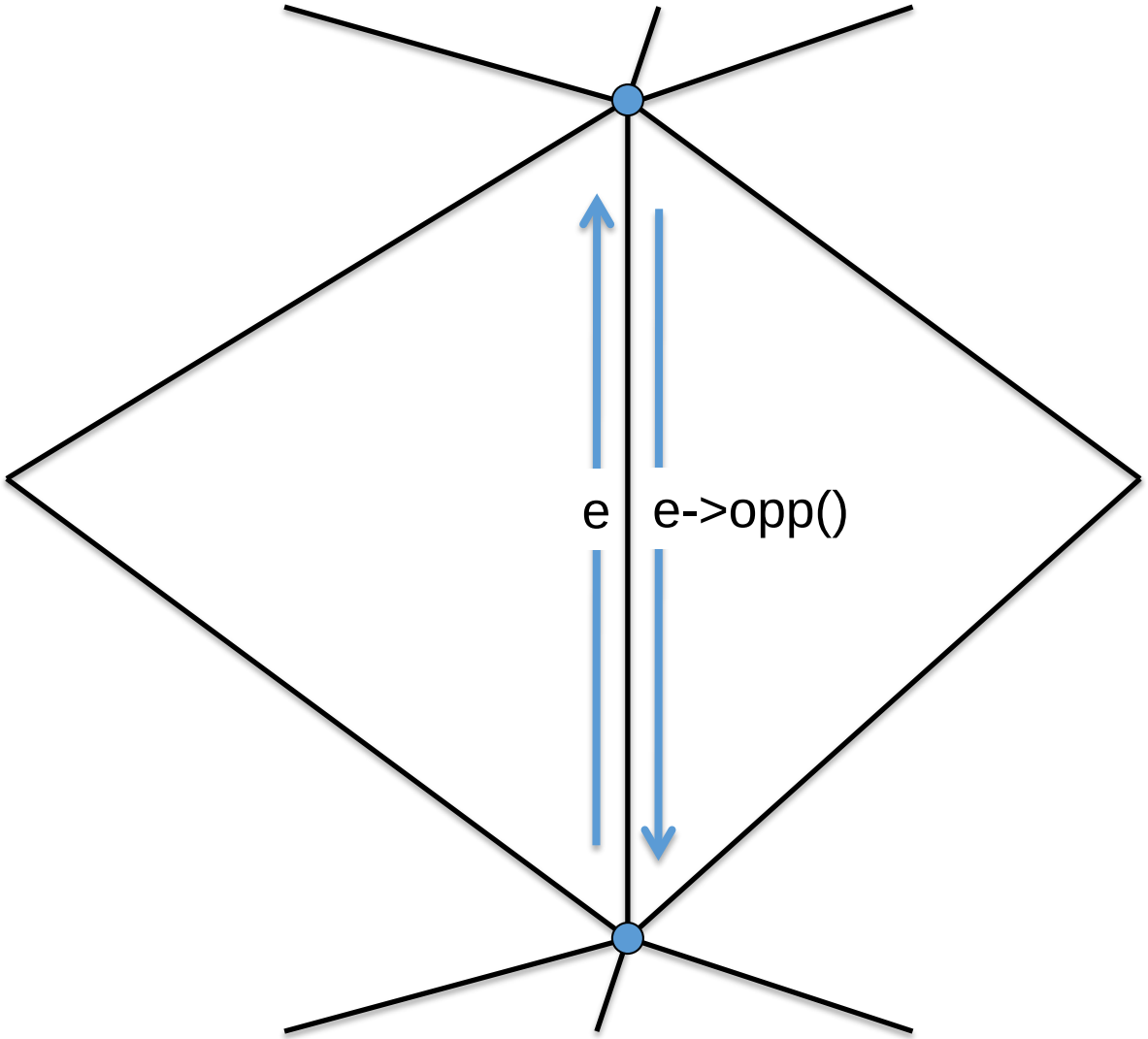
Half Edge



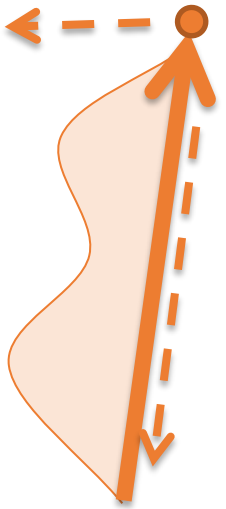

```
class HalfEdge {
  HalfEdge *opp;
  Vertex   *end;
  Face     *left;
  HalfEdge *next;
};

HalfEdge e;
```



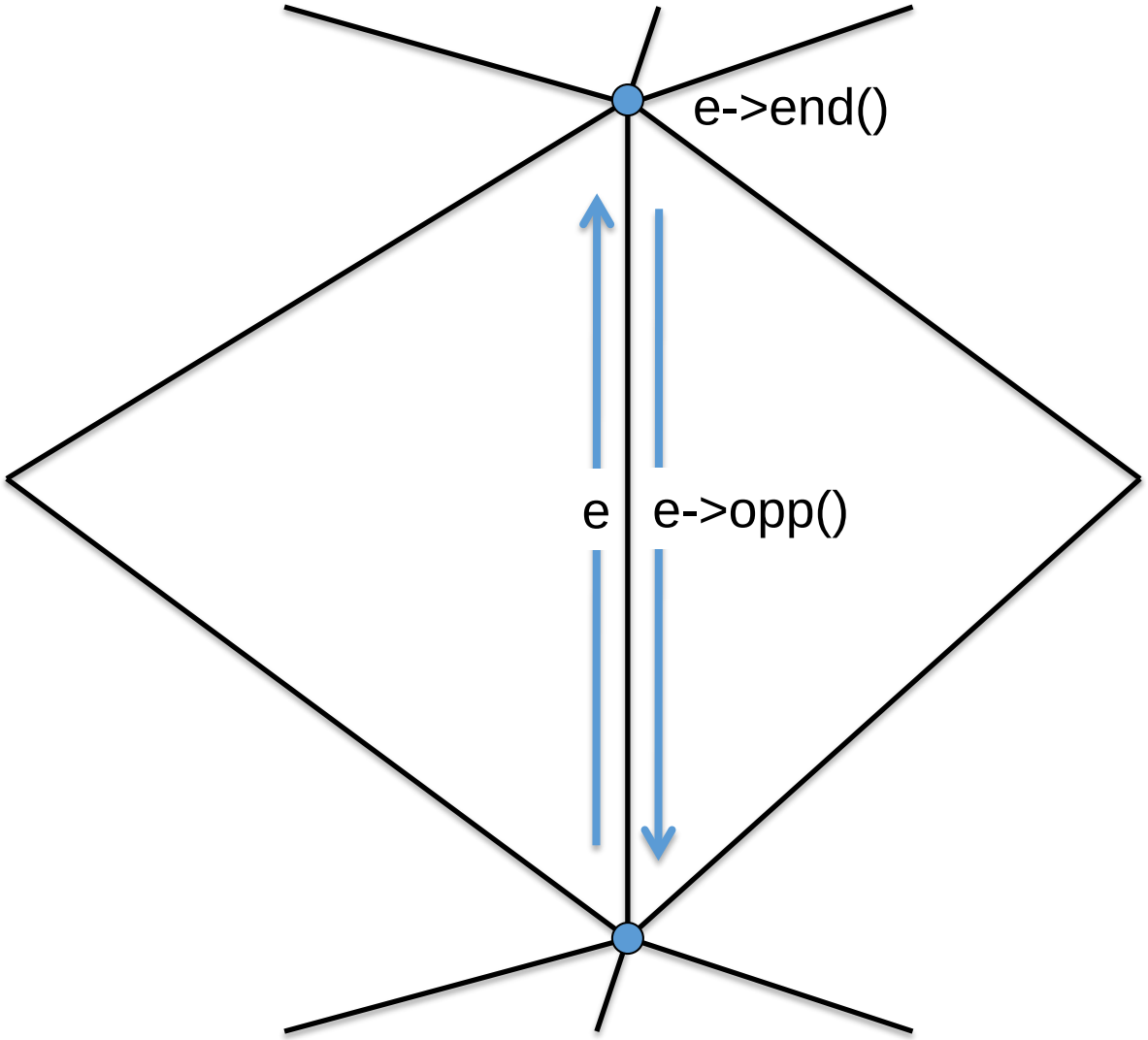
Half Edge




```
class HalfEdge {
  HalfEdge *opp;
  Vertex   *end;
  Face     *left;
  HalfEdge *next;
};

HalfEdge e;
```

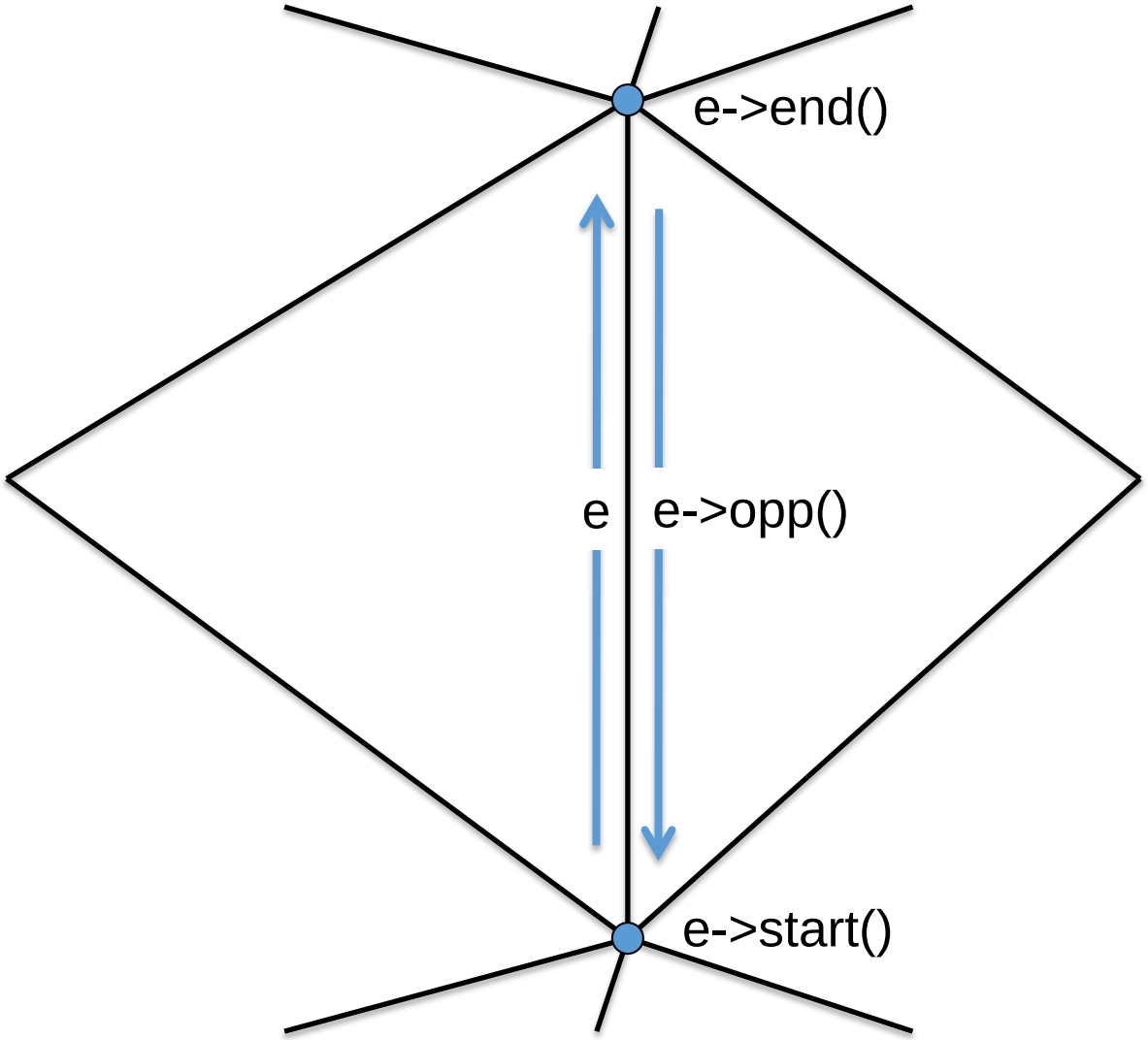


Half Edge



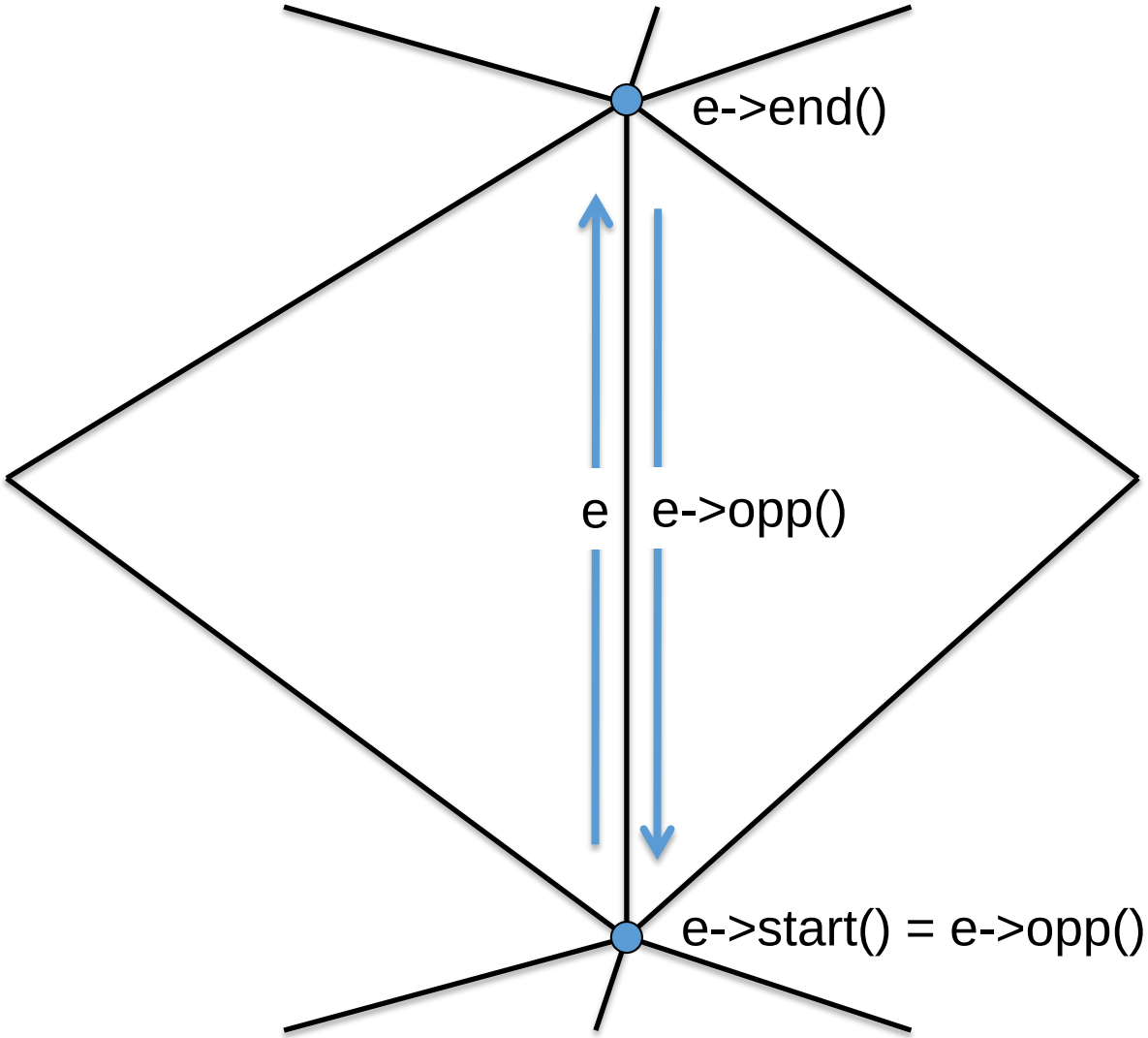

```
class HalfEdge {
  HalfEdge *opp;
  Vertex   *end;
  Face     *left;
  HalfEdge *next;
};

HalfEdge e;
```

Half Edge



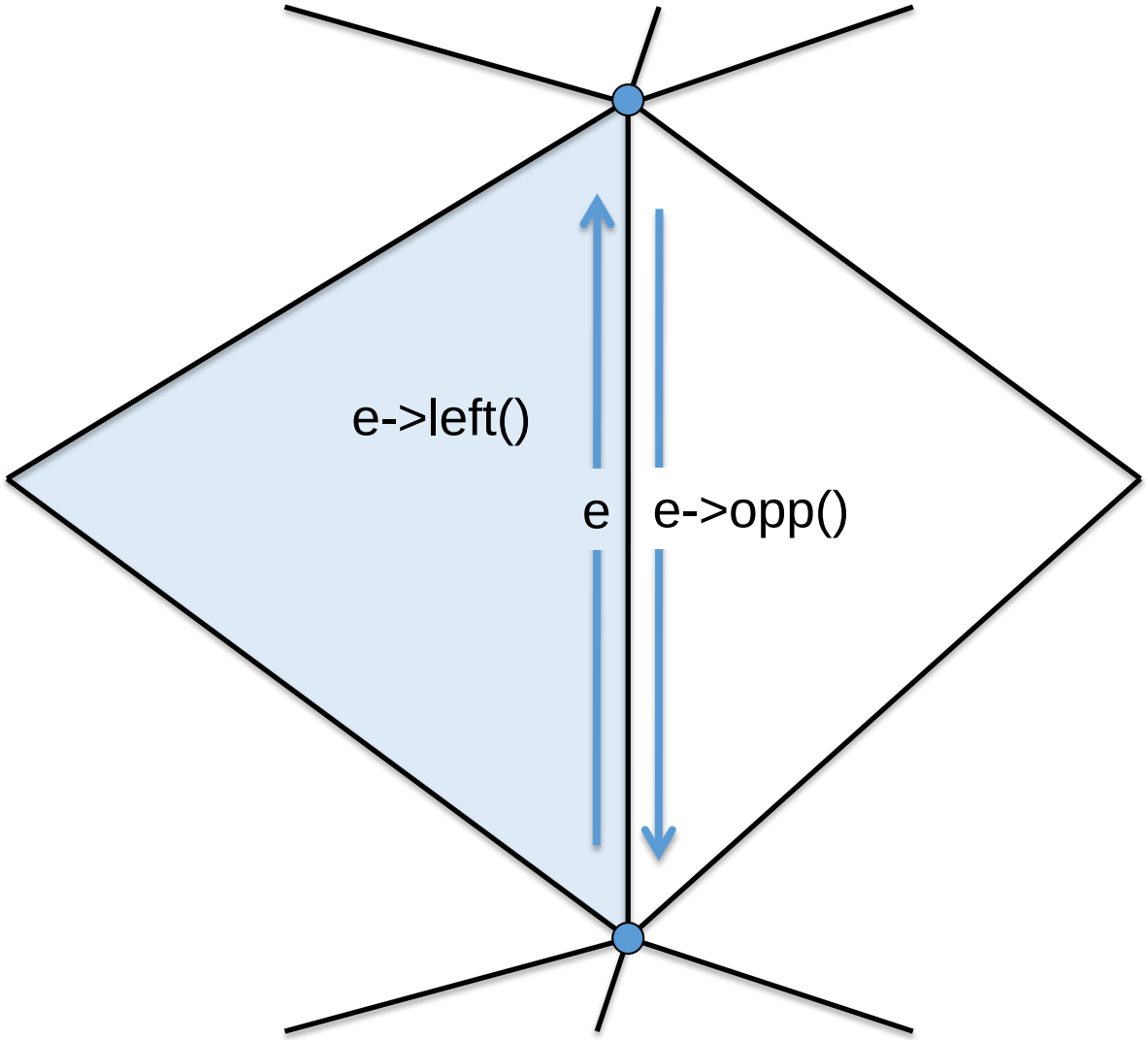

```
class HalfEdge {
  HalfEdge *opp;
  Vertex   *end;
  Face     *left;
  HalfEdge *next;
};

HalfEdge e;
```

Half Edge



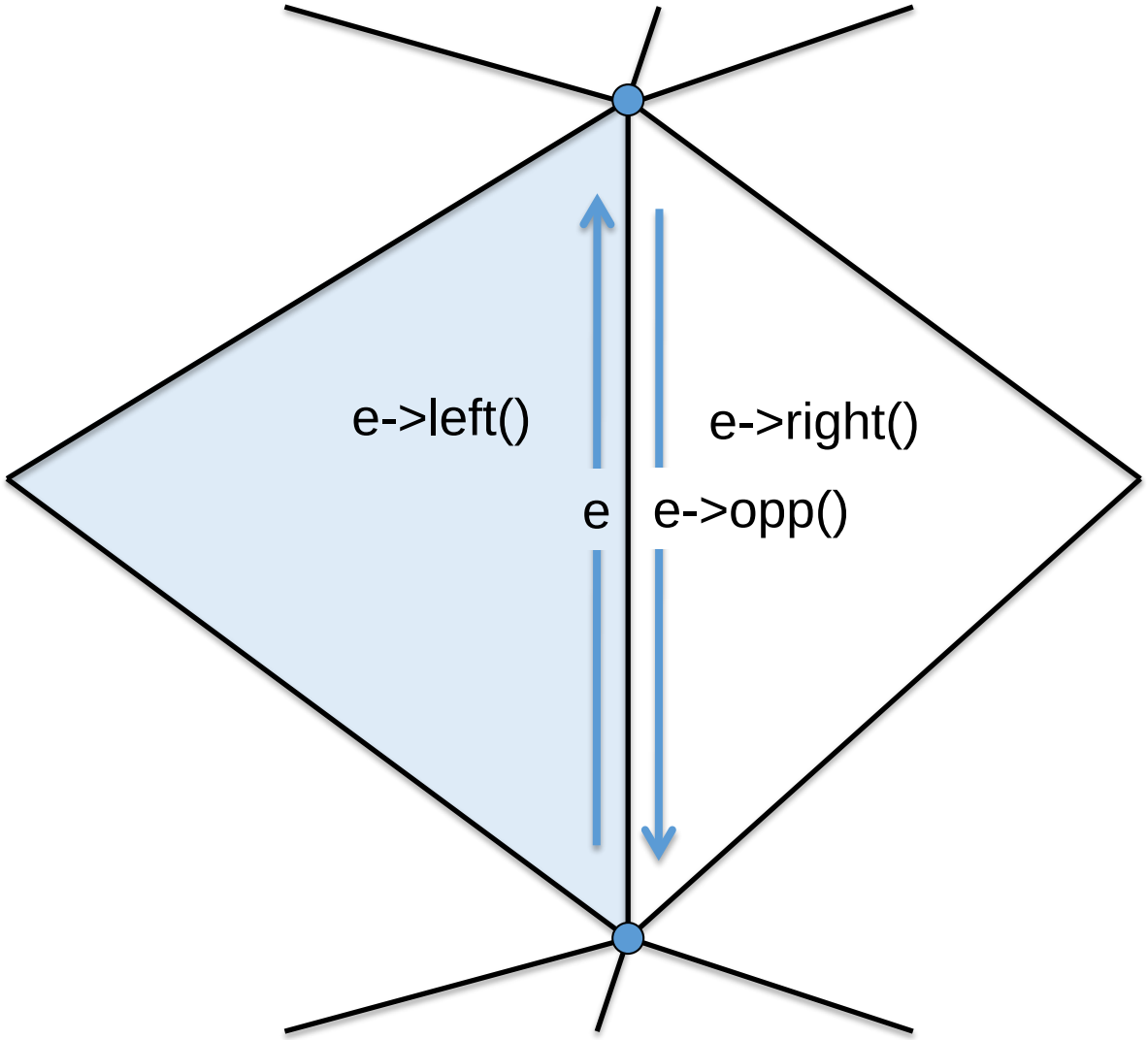

```
class HalfEdge {
  HalfEdge *opp;
  Vertex   *end;
  Face     *left;
  HalfEdge *next;
};

HalfEdge e;
```

Half Edge



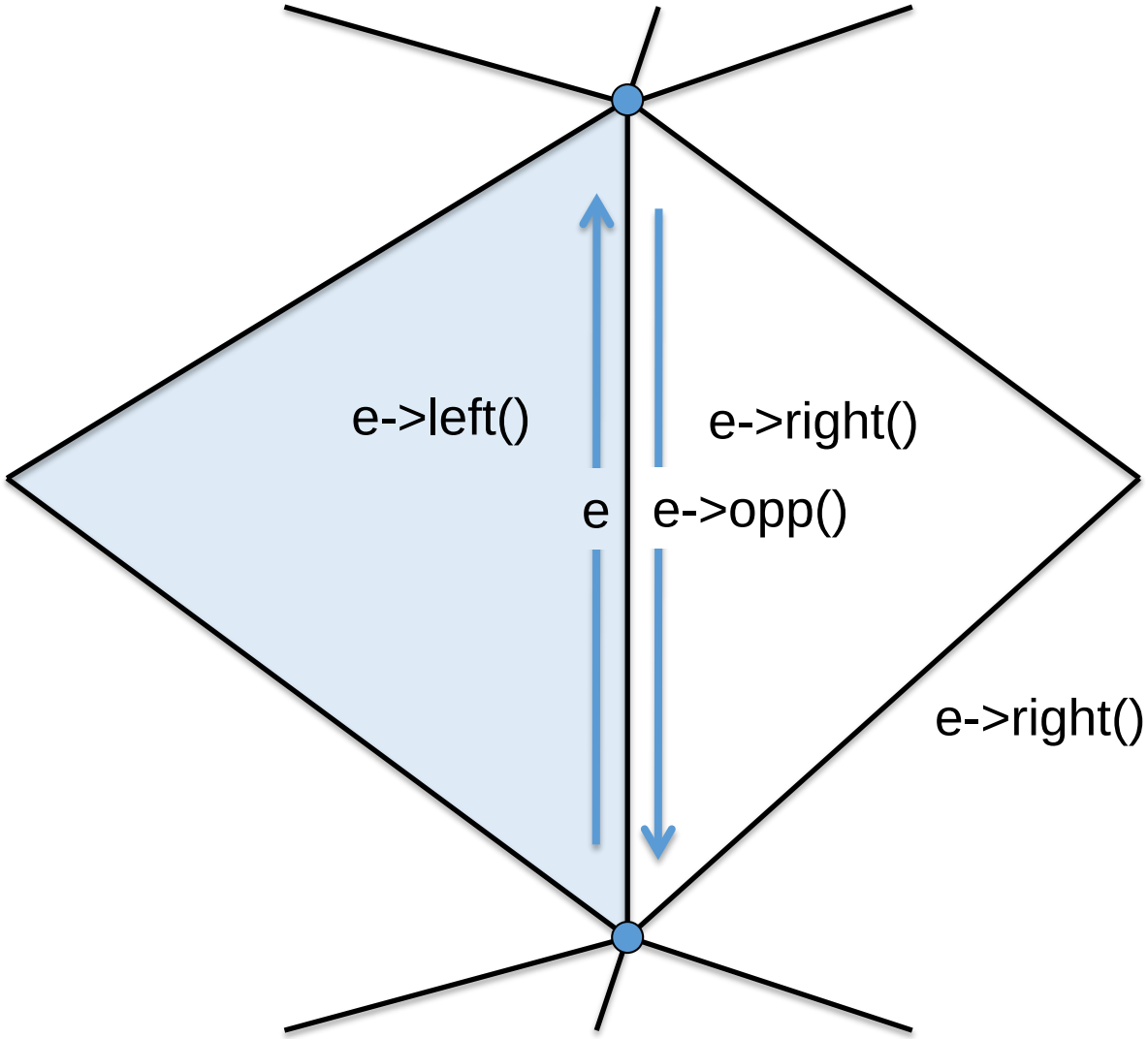

```
class HalfEdge {
  HalfEdge *opp;
  Vertex   *end;
  Face     *left;
  HalfEdge *next;
};

HalfEdge e;
```

Half Edge




```
class HalfEdge {
  HalfEdge *opp;
  Vertex   *end;
  Face     *left;
  HalfEdge *next;
};

HalfEdge e;
```

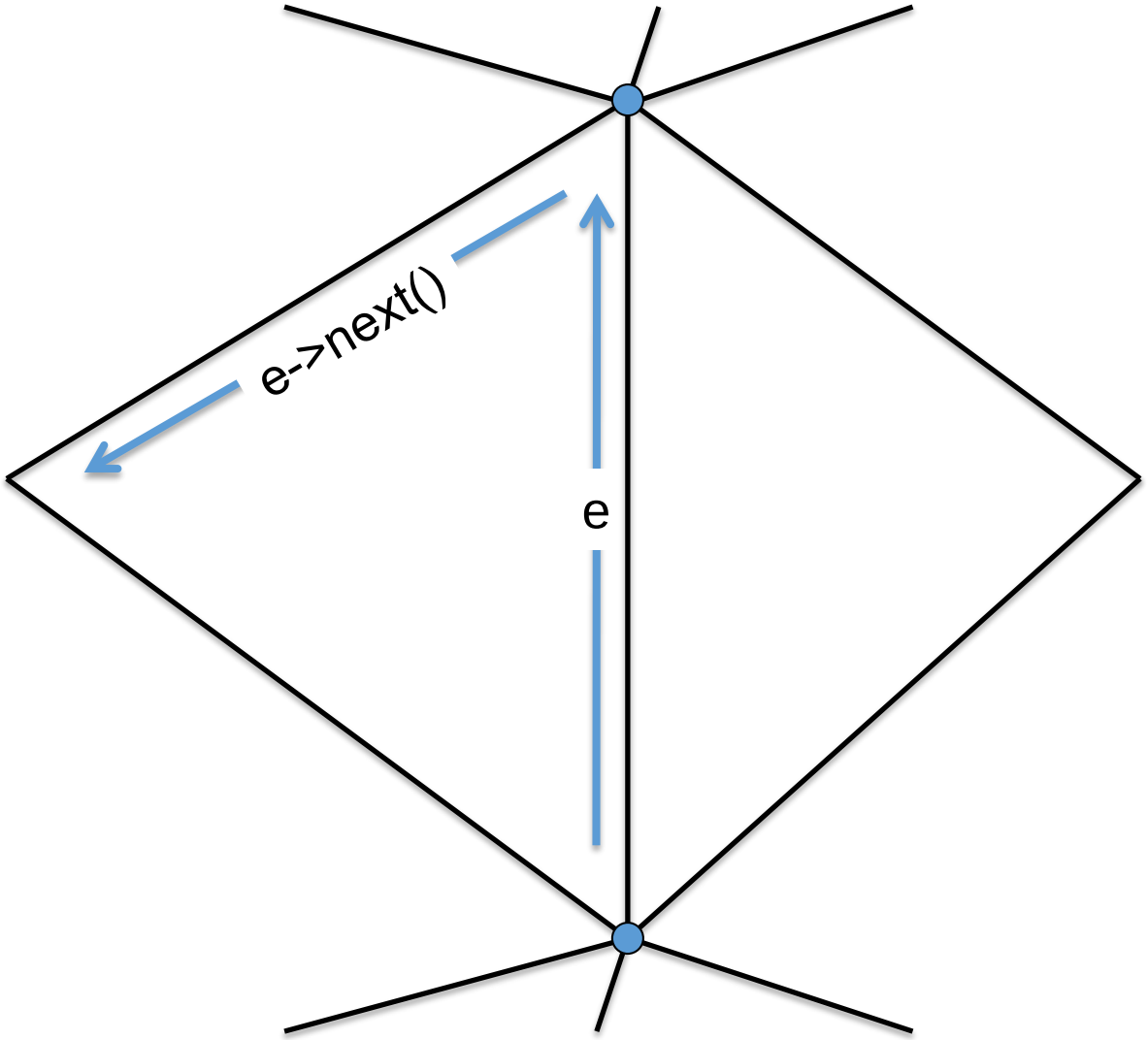
Half Edge




```
class HalfEdge {
  HalfEdge *opp;
  Vertex   *end;
  Face     *left;
  HalfEdge *next;
};

HalfEdge e;
```

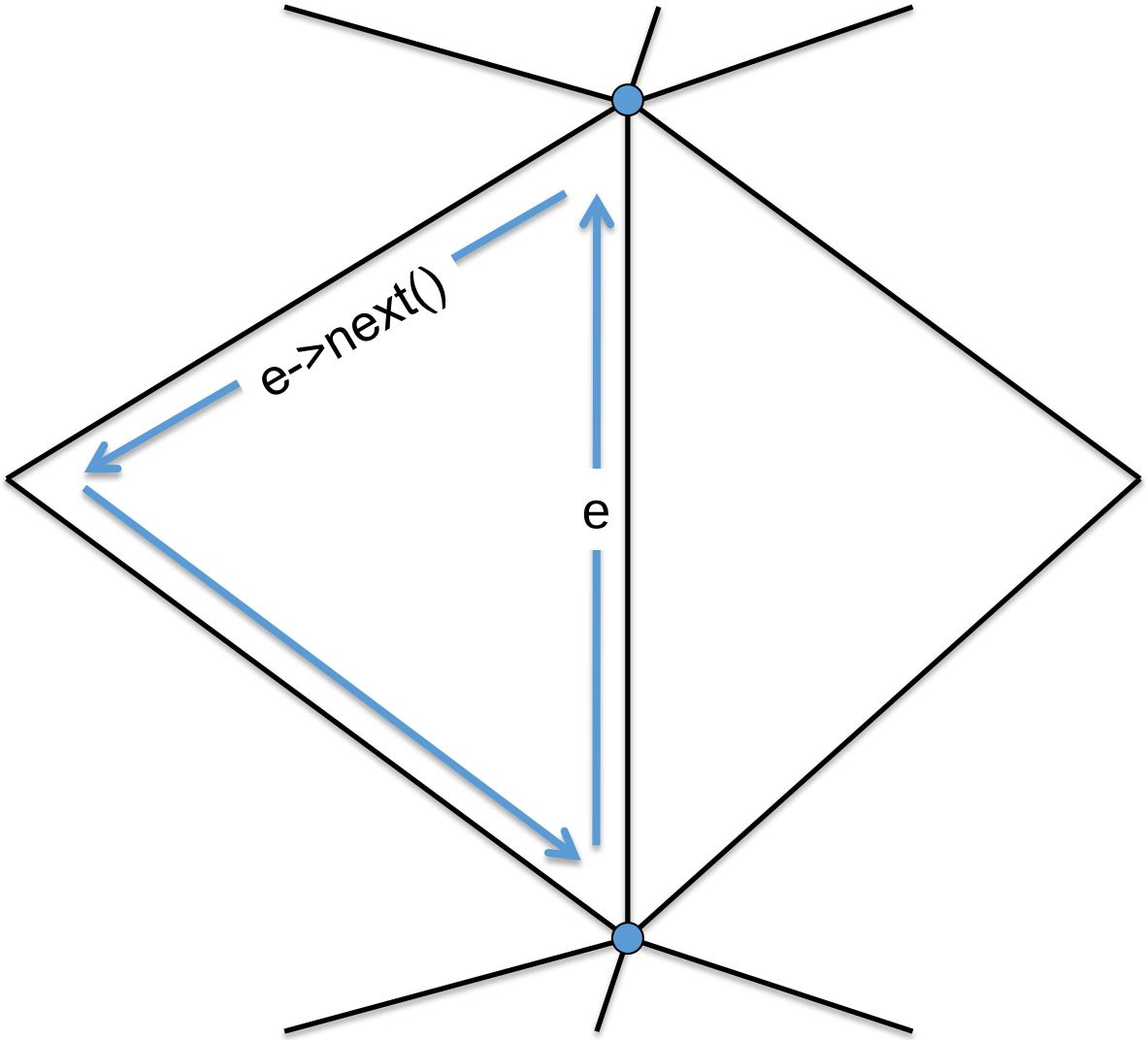
e->right() = e->opp()->left();

Half Edge



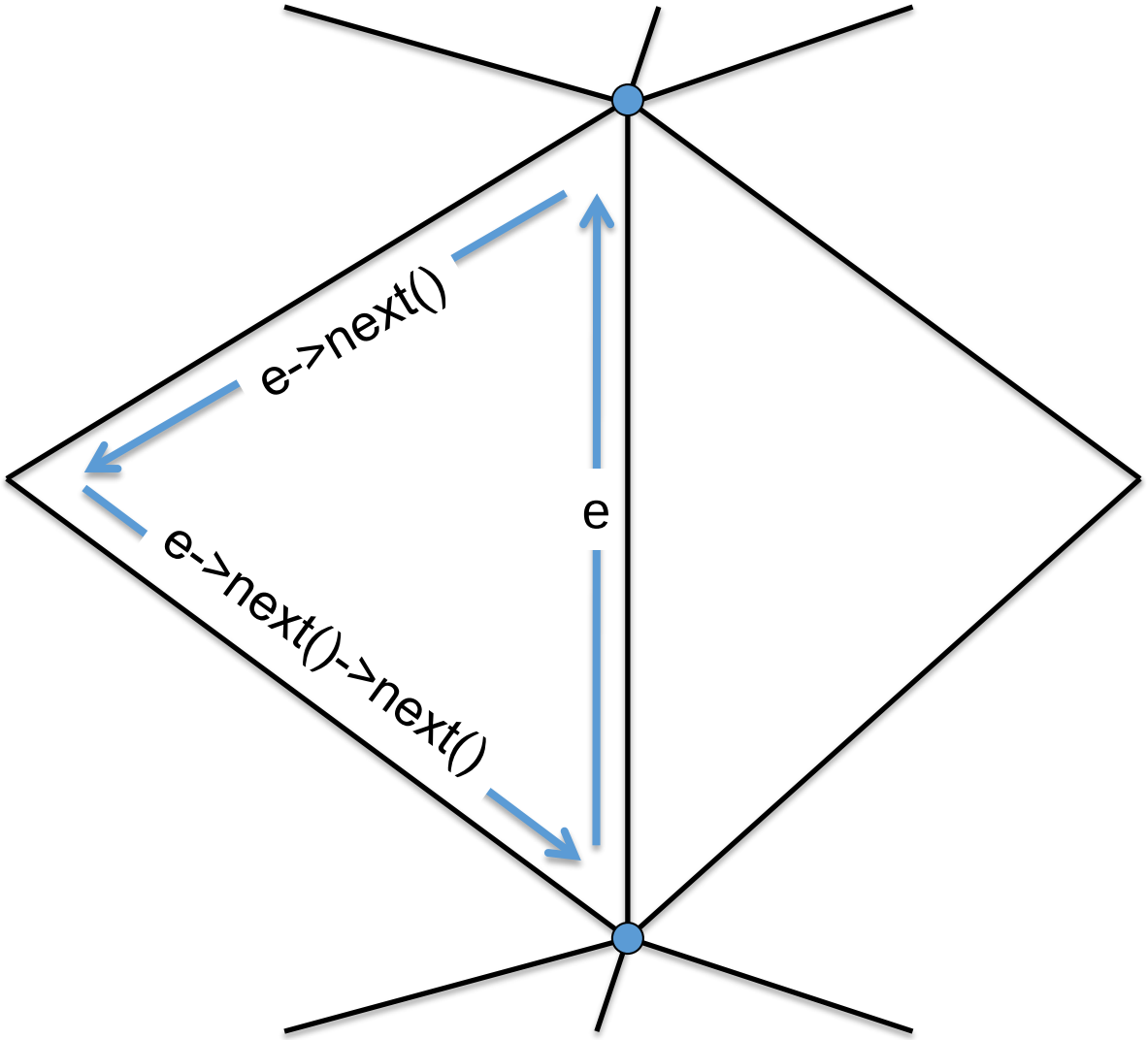

```
class HalfEdge {
  HalfEdge *opp;
  Vertex   *end;
  Face     *left;
  HalfEdge *next;
};

HalfEdge e;
```

Half Edge



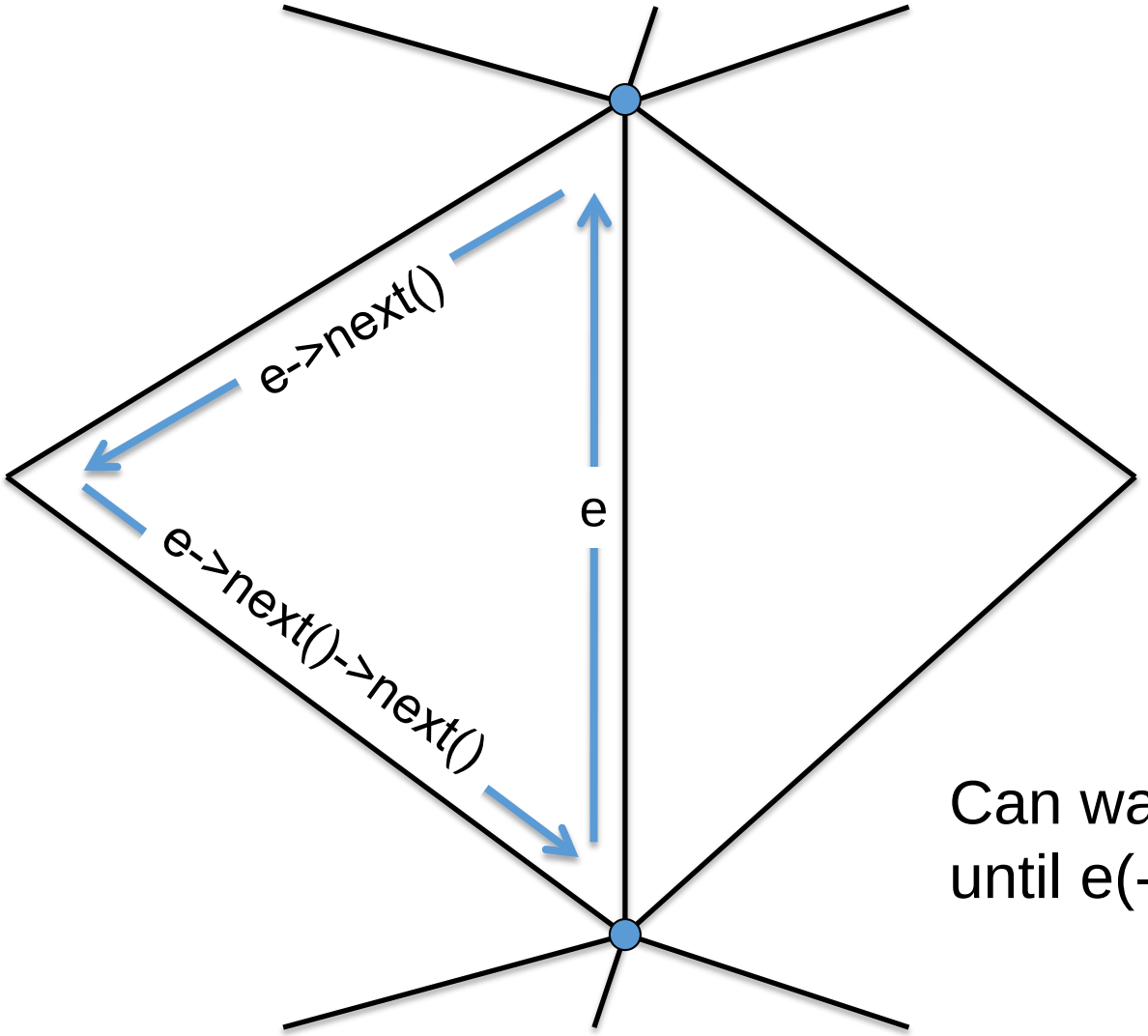

```
class HalfEdge {
  HalfEdge *opp;
  Vertex   *end;
  Face     *left;
  HalfEdge *next;
};

HalfEdge e;
```

Half Edge




```
class HalfEdge {
  HalfEdge *opp;
  Vertex   *end;
  Face     *left;
  HalfEdge *next;
};

HalfEdge e;
```

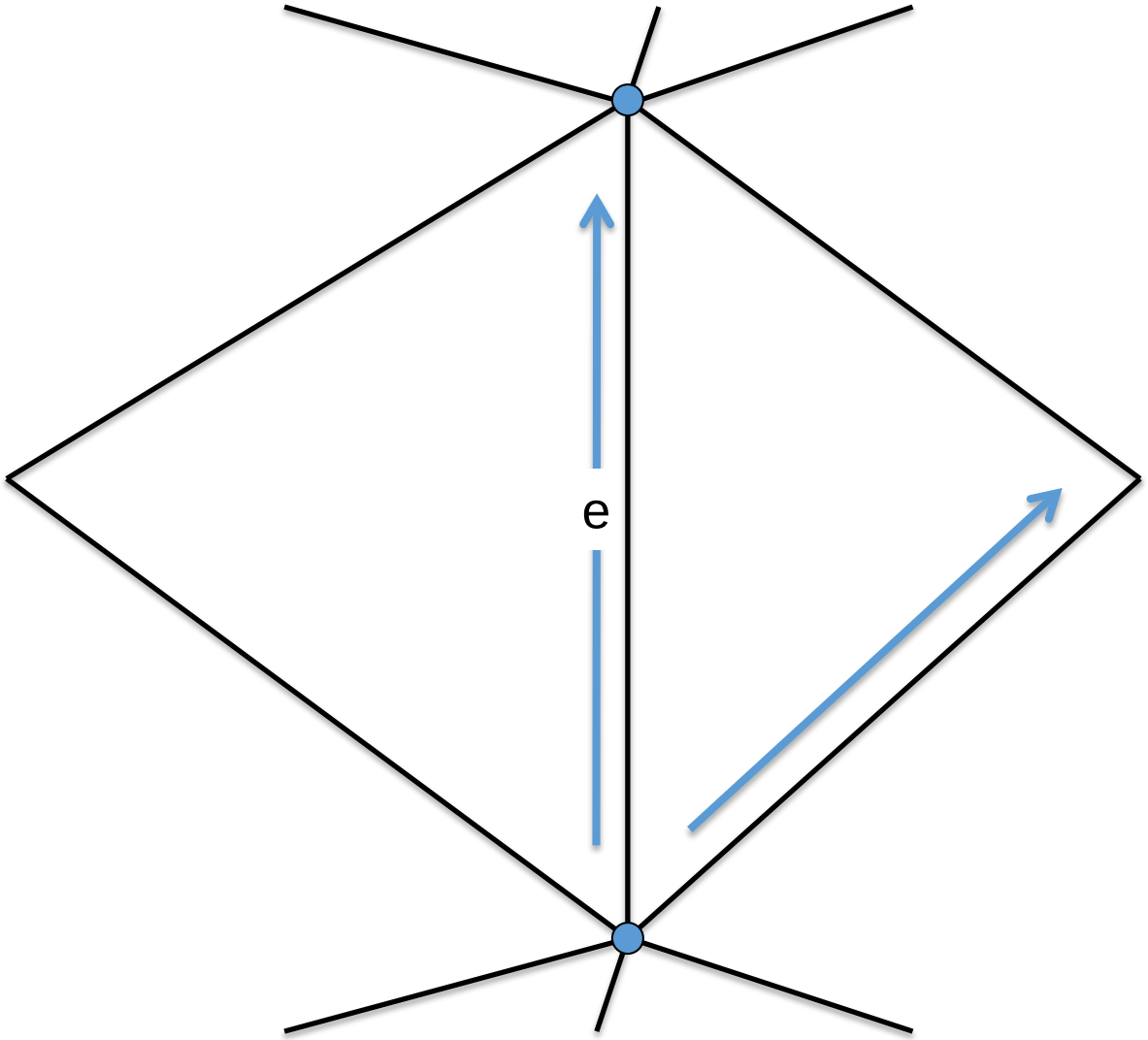
Half Edge




```
class HalfEdge {
  HalfEdge *opp;
  Vertex   *end;
  Face     *left;
  HalfEdge *next;
};

HalfEdge e;
```

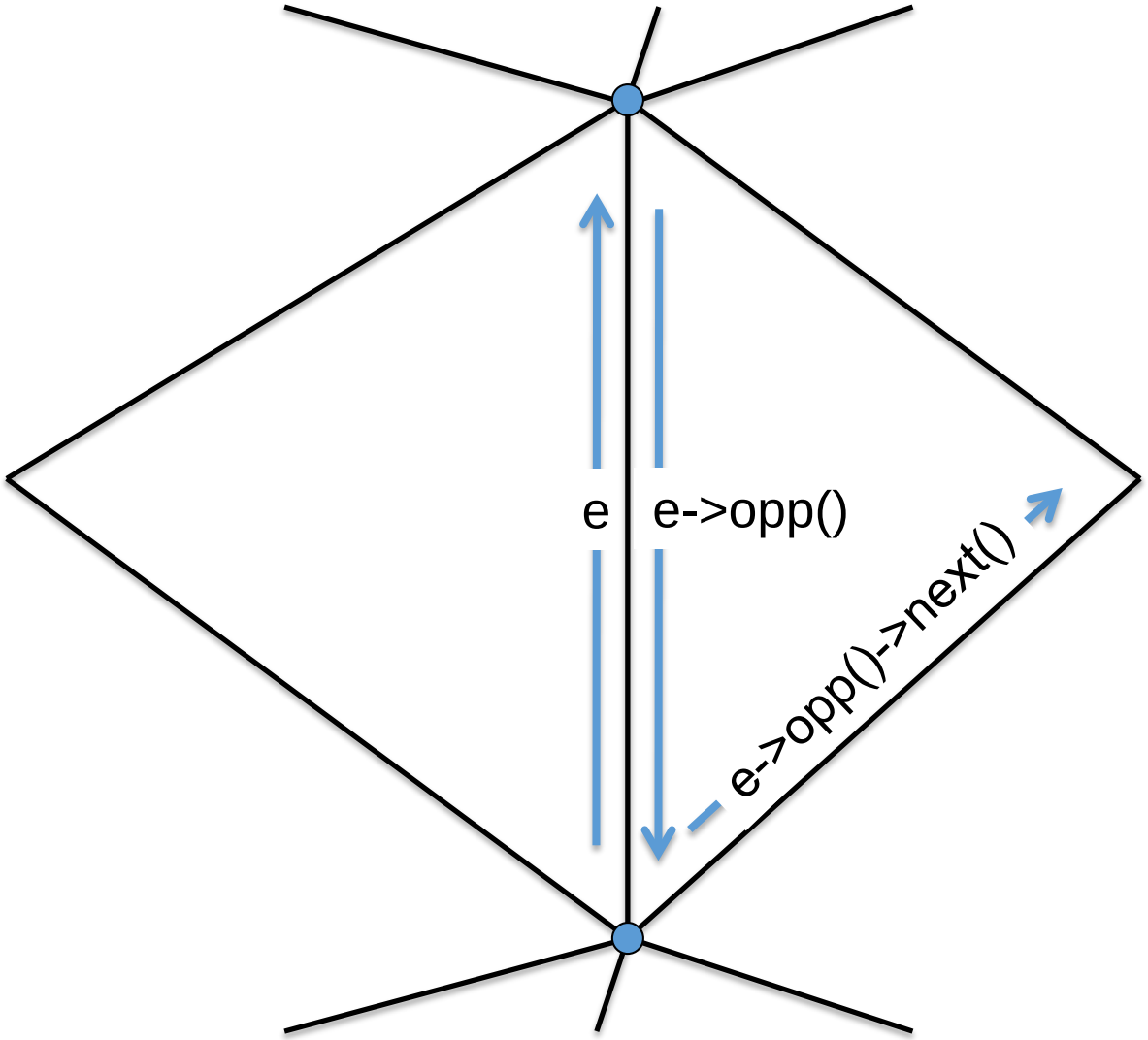
Can walk around left face
until $e(->next)^n = e$

Half Edge



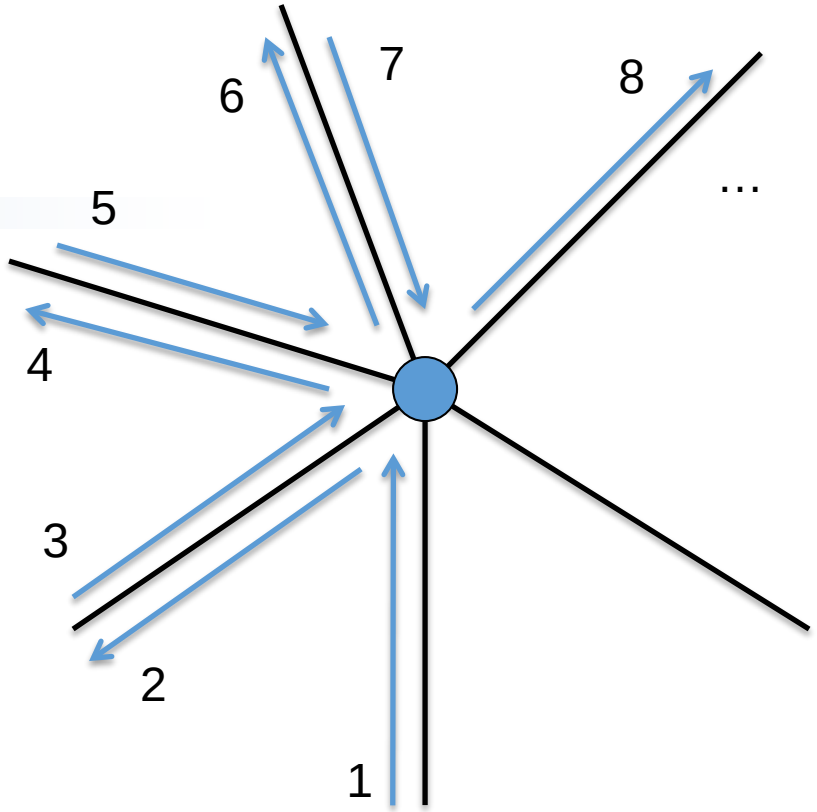

```
class HalfEdge {
  HalfEdge *opp;
  Vertex   *end;
  Face     *left;
  HalfEdge *next;
};

HalfEdge e;
```

Half Edge




```
class HalfEdge {
  HalfEdge *opp;
  Vertex   *end;
  Face     *left;
  HalfEdge *next;
};

HalfEdge e;
```

Vertex Star Query



1. e
2. e->next()
3. e->next()->opp()
4. e->next()->opp()->next()
5. e->next()->opp()->next()->opp()
6. e->next()->opp()->next()->opp()->next()
7. e->next()->opp()->next()->opp()->next()->opp()
8. e->next()->opp()->next()->opp()->next()->opp()->next()

… until $e(->next()->opp())^n == e$

Half-Edge Implementation

Vertices

```
v <x1> <y1> <z1>
v <x2> <y2> <z2>
v <x3> <y3> <z3>
…
```

Faces

```
f 1 2 3
f 3 2 4
f 3 4 5
…
```

Half-Edge Implementation

Vertices

```
v <x1> <y1> <z1> <he>
v <x2> <y2> <z2> <he>
v <x3> <y3> <z3> <he>
…
```

Faces

```
f 1 2 3 <he>
f 3 2 4 <he>
f 3 4 5 <he>
…
```

Half Edges

```
HalfEdge *opp;
Vertex *end;
Face *left;
HalfEdge *next;
```

```
HalfEdge *opp;
Vertex *end;
Face *left;
HalfEdge *next;
```

…