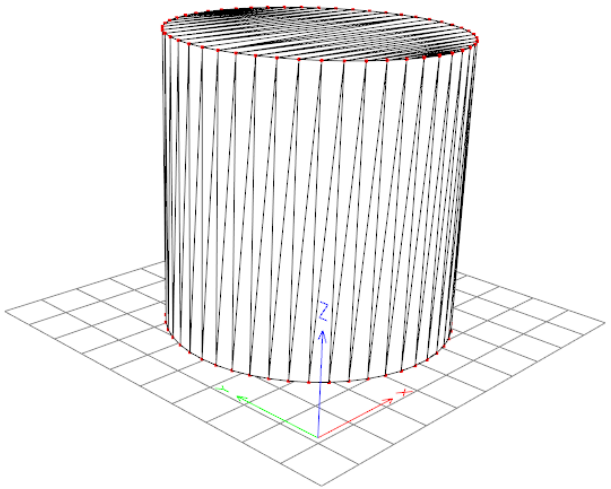
Meshes

CS418 Computer Graphics

John C. Hart

Simple Meshes



- Cylinder

  $(x,y,z) = (\cos\theta, \sin\theta, z)$

- Cone

  $(x,y,z) = (|z|\cos\theta, |z|\sin\theta, z)$

- Sphere

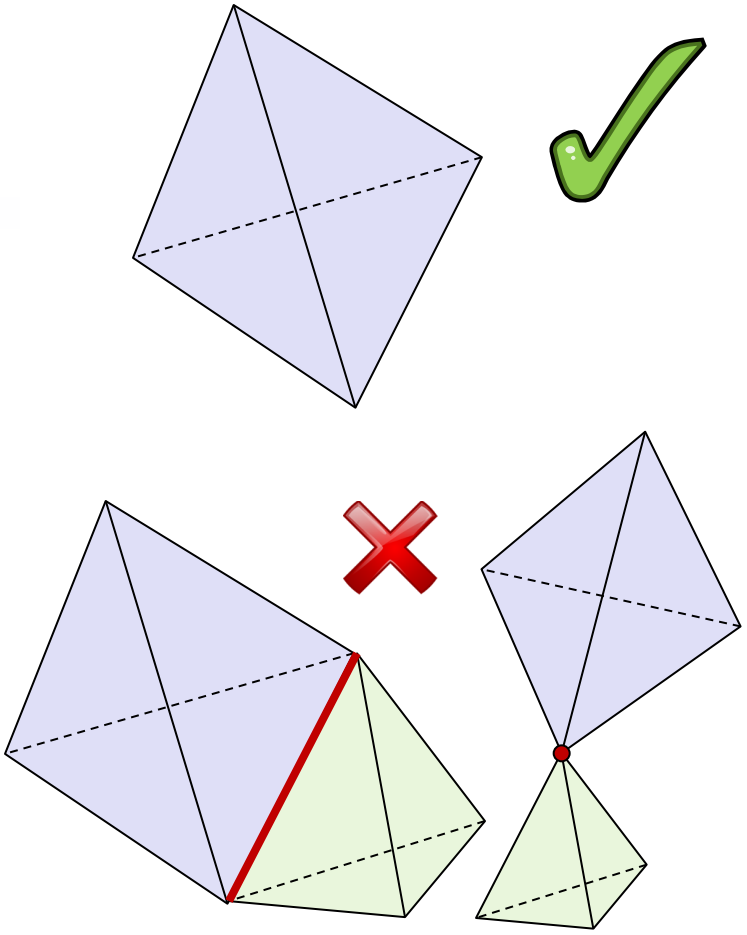
  $(x,y,z) = (\cos\phi\cos\theta, \cos\phi\sin\theta, \sin\phi)$

- Torus

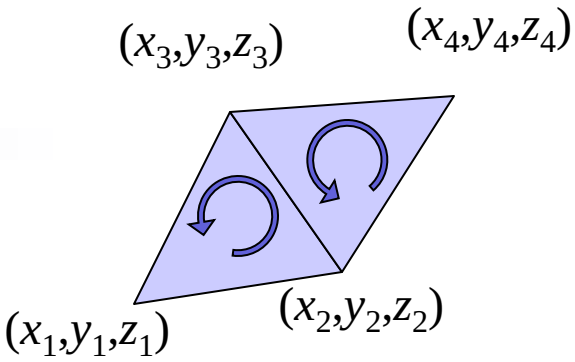
  $(x,y,z) = ((R + \cos\phi)\cos\theta, (R + \cos\phi)\sin\theta, \sin\phi)$

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

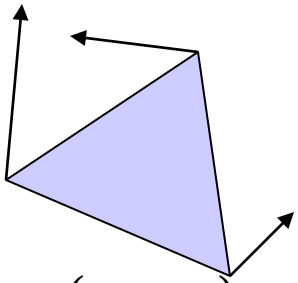
f 1 2 3
f 2 4 3

Other Attributes

- Vertex normals
  - Prefixed w/ “vn” (Wavefront)
  - Contains x,y,z of normal
  - Not necessarily unit length
  - Not necessarily in vertex order
  - Indexed as with vertices
- Texture coordinates
  - Prefixed with “vt” (Wavefront)
  - Not necessarily in vertex order
  - Contains u,v surface parameters
- Faces
  - Uses “/” to separate indices
  - Vertex “/” normal “/” texture
  - Normal and texture optional
  - Can eliminate normal with “//”

(x_2,y_2,z_2)
(a_2,b_2,c_2)
(u_2,v_2)

(x_0,y_0,z_0)
(a_0,b_0,c_0)
(u_0,v_0)

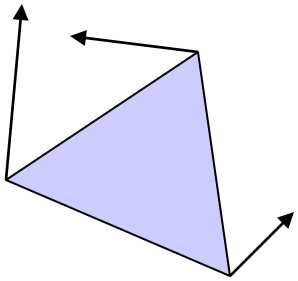


(x_1,y_1,z_1)
(a_1,b_1,c_1)
(u_1,v_1)

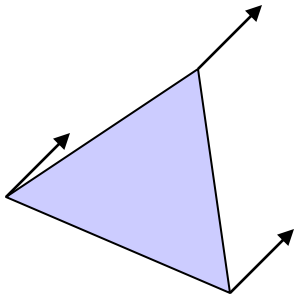
v x_0 y_0 z_0
v x_1 y_1 z_1
v x_2 y_2 z_2

vn a_0 b_0 c_0
vn a_1 b_1 c_1
vn a_2 b_2 c_2

vt u_0 v_0
vt u_1 v_1
vt u_2 v_2



f 0/0/0 1/1/1 2/2/2



f 0/0/0 1/0/1 2/0/2

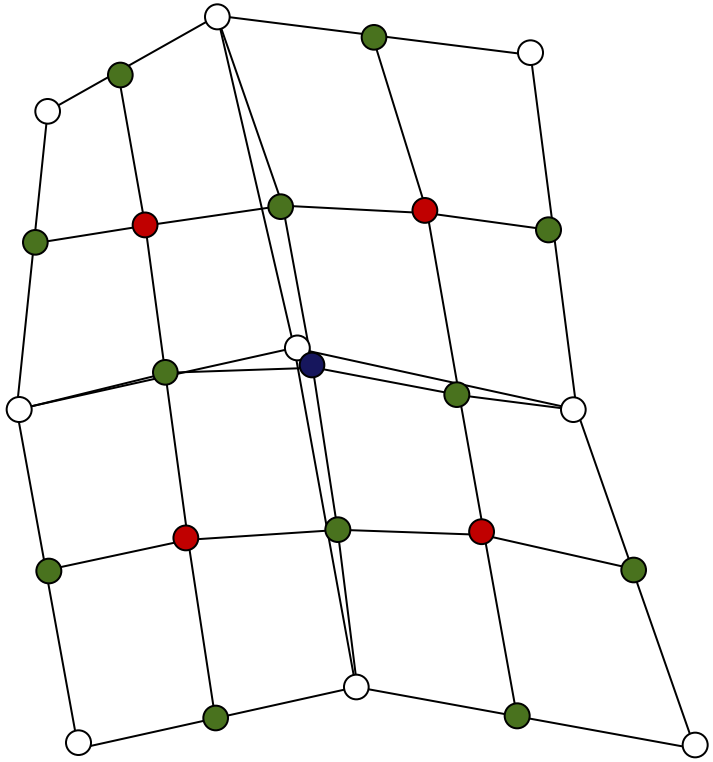
Catmull Clark Subdivision

- First subdivision generates quad mesh
- Generates B-spline patch when applied to 4x4 network of vertices
- Some vertices extraordinary (valence ≠ 4)

Rules:

- 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…
  - Average of neighboring face points
  - 2 x average of neighboring edge points
  - (valence – 3) x original vertex position

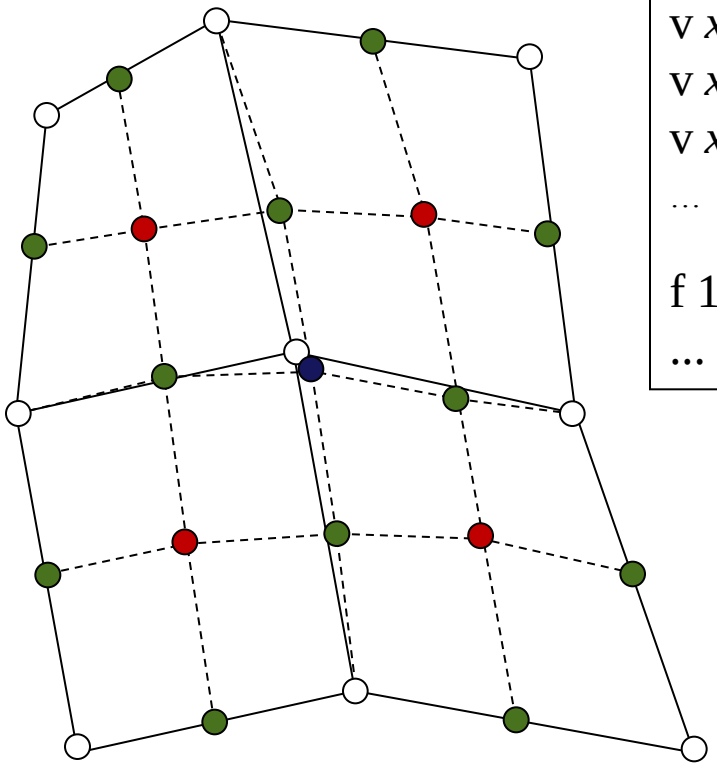
(boundary edge points set to edge midpoints, boundary vertices stay put)



Implementation

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…
Average of neighboring face points
2 x average of neighboring edge points
(valence – 3) x original vertex position

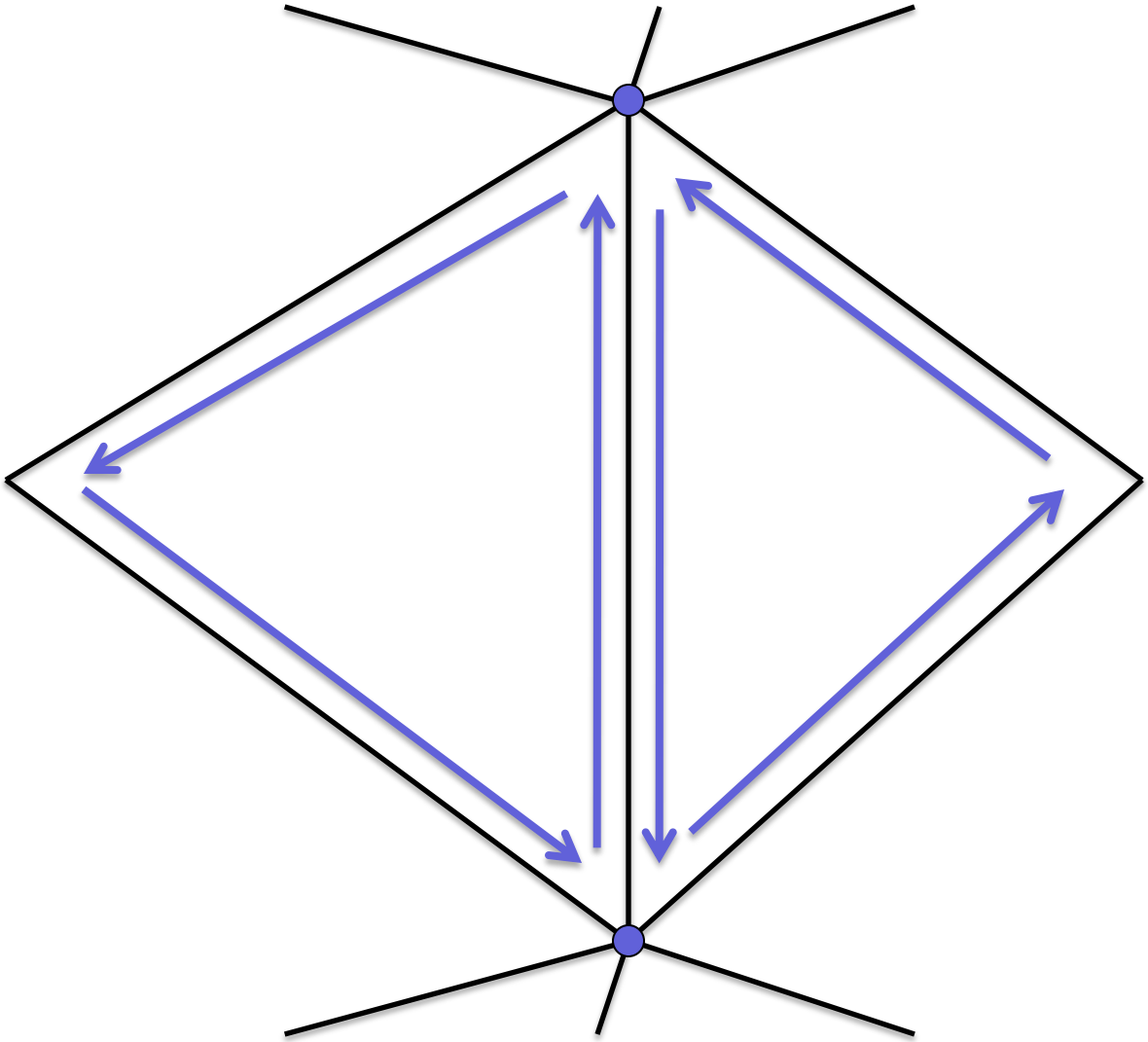
- Face vertex
  - For each face
    add vertex at its centroid
- Edge vertex
  - How do we find each edge?
- New vertex position
  - For a given vertex
    how do we find neighboring faces and edges?



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
…

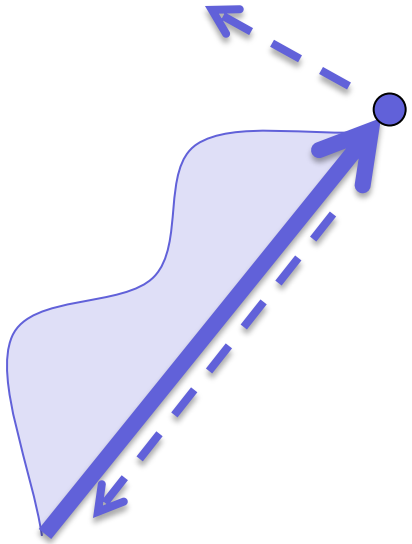
f 1 2 3 4
…

Half Edge

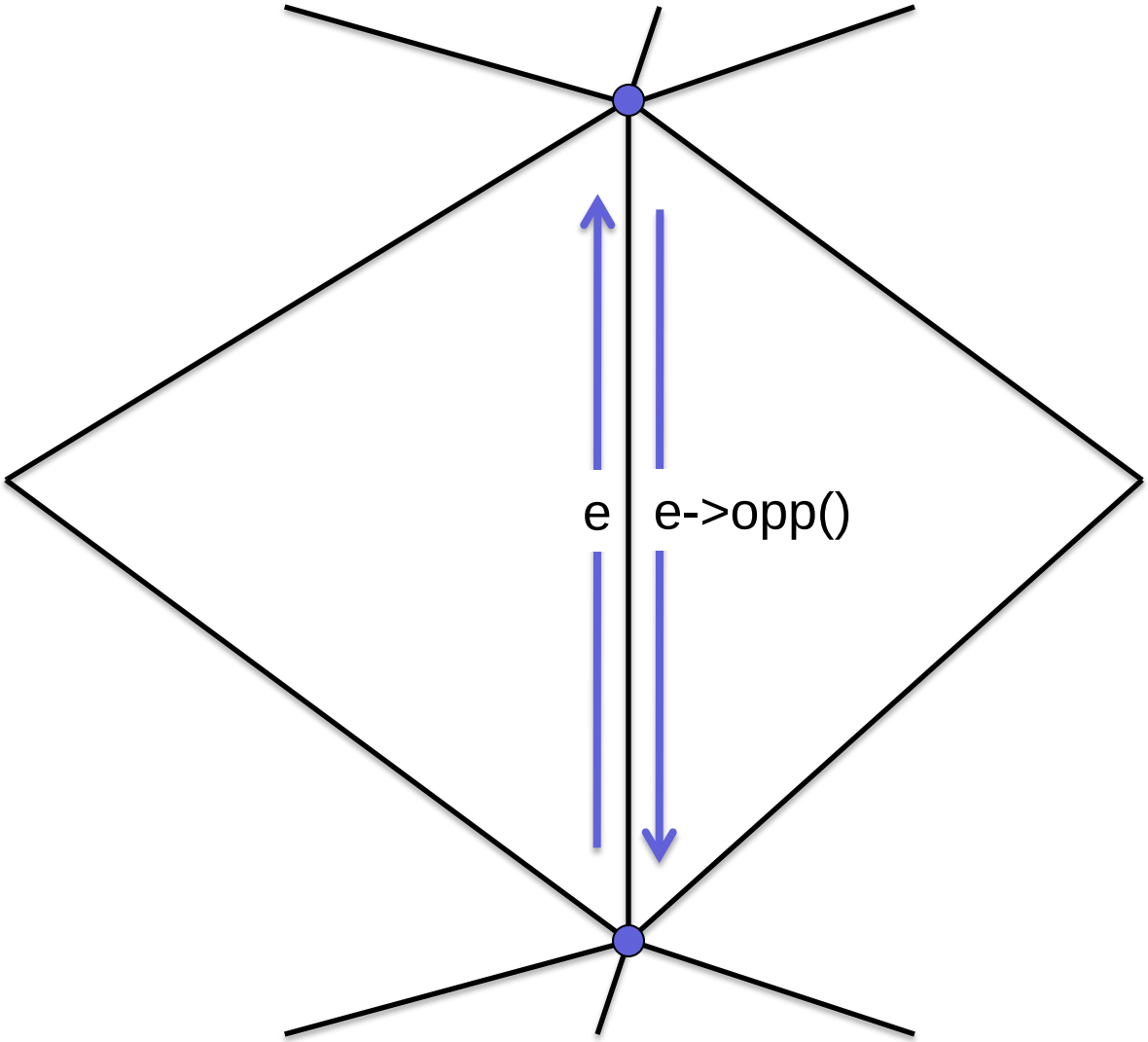


```
class HalfEdge {
  HalfEdge *opp;
  Vertex   *end;
  Face     *left;
  HalfEdge *next;
};

HalfEdge e;
```



Half Edge



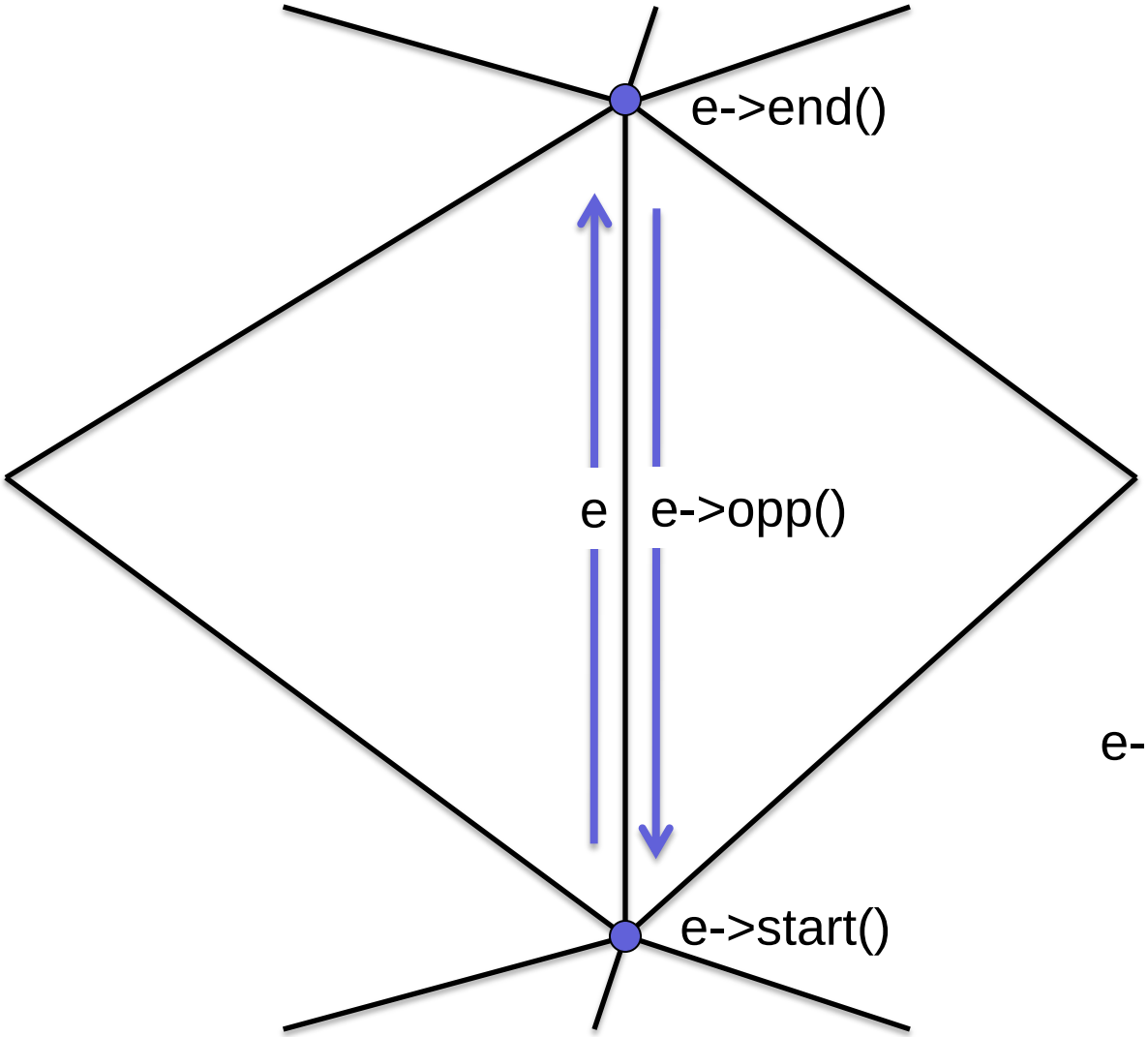

```
class HalfEdge {
  HalfEdge *opp;
  Vertex   *end;
  Face     *left;
  HalfEdge *next;
};

HalfEdge e;
```

Half Edge

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

HalfEdge e;
```

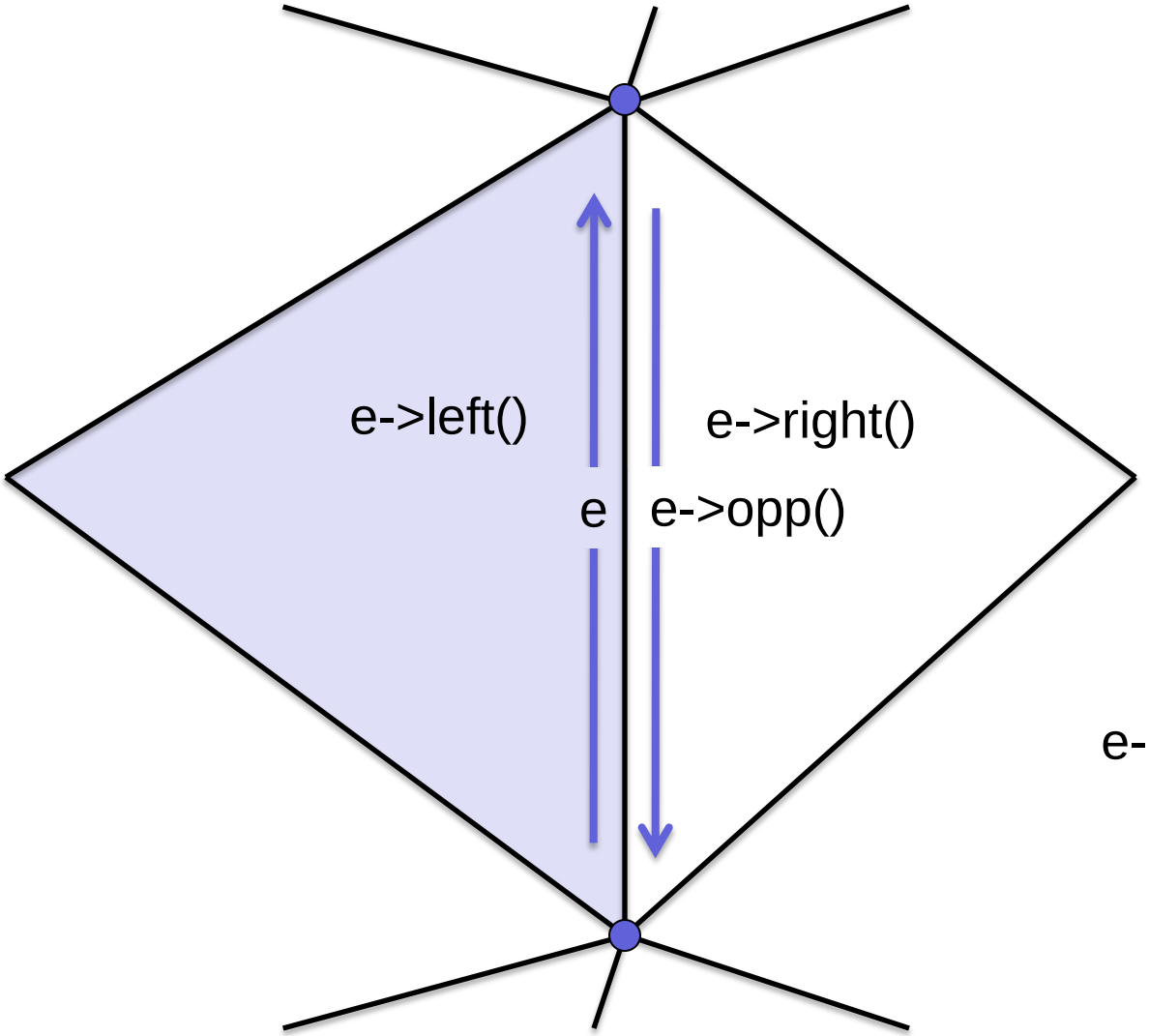


e->start() = e->opp()->end();

Half Edge

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

HalfEdge e;
```

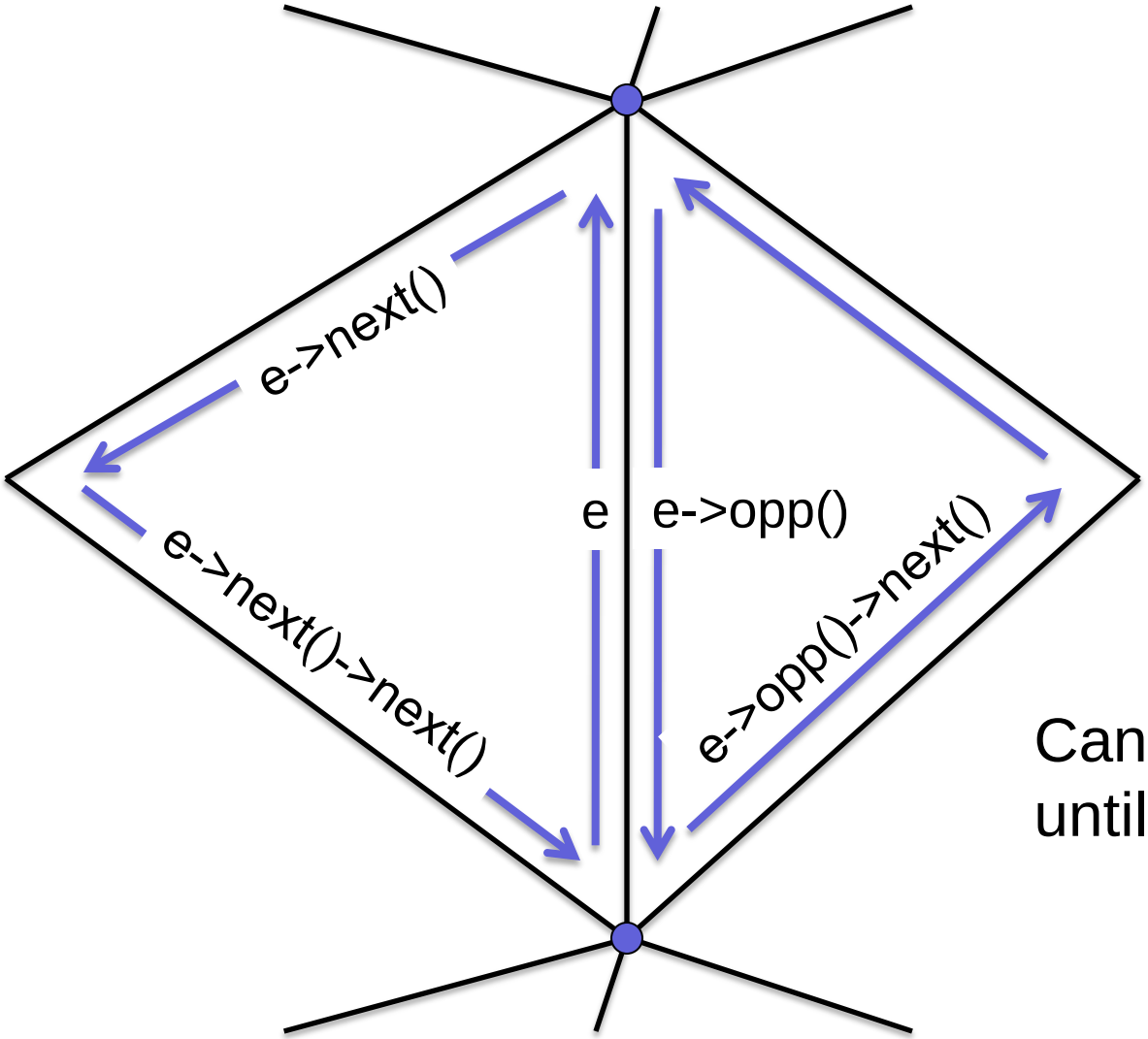


e->right() = e->opp()->left();

Half Edge

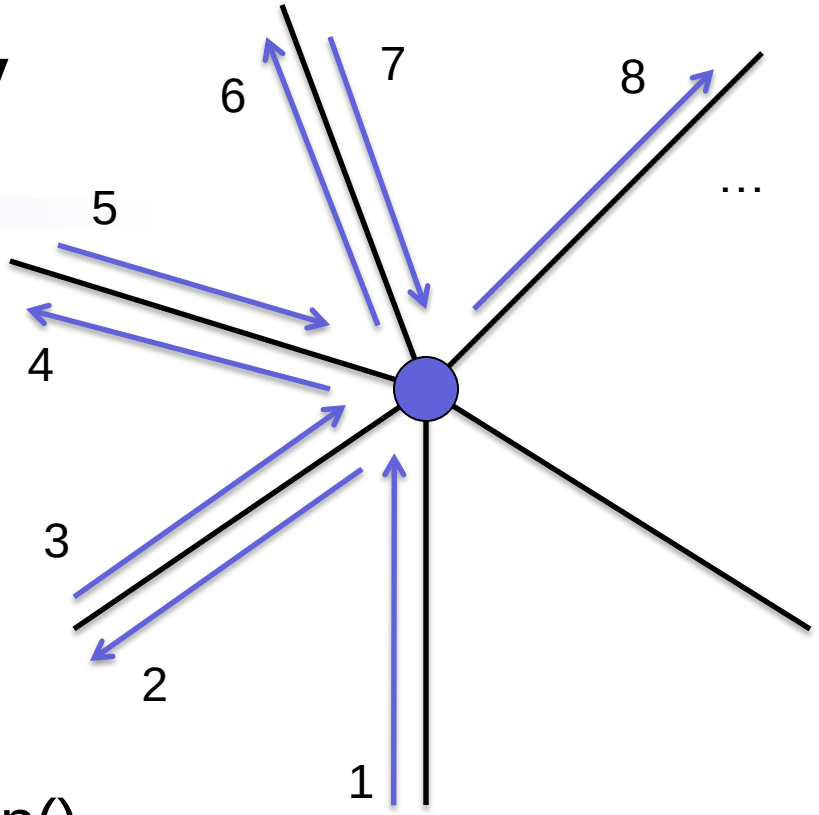
```
class HalfEdge {
  HalfEdge *opp;
  Vertex   *end;
  Face     *left;
  HalfEdge *next;
};

HalfEdge e;
```



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

Vertex Star Query

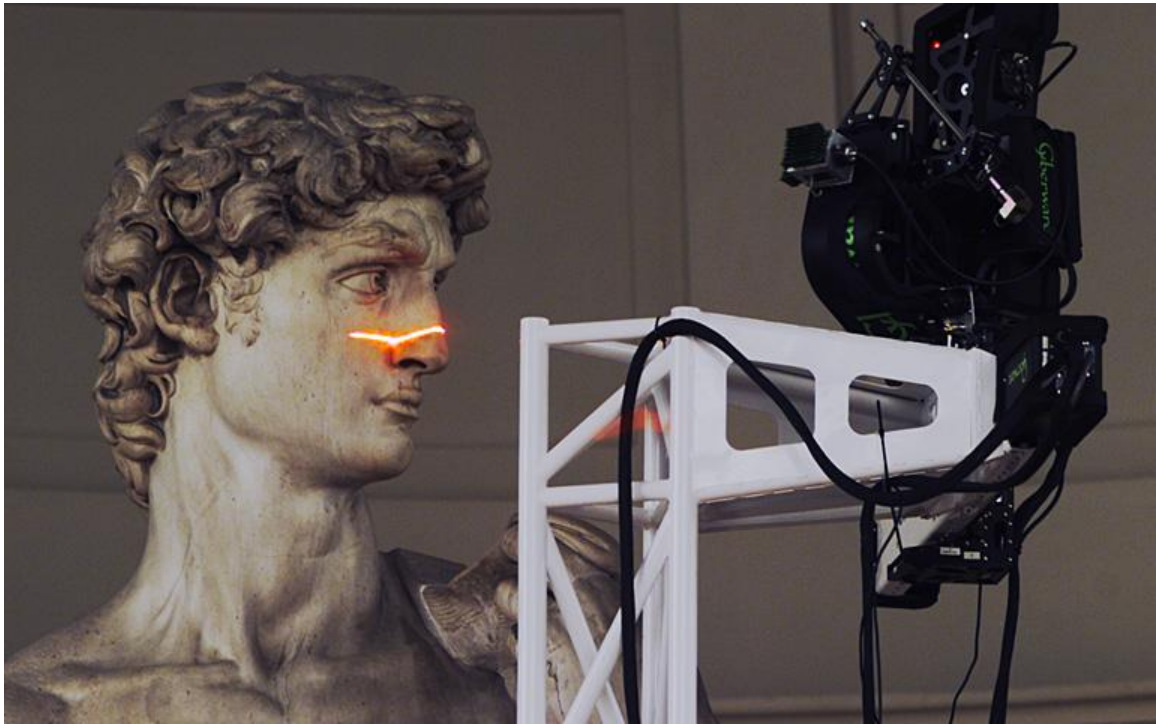


1. e
2. e->next()
3. e->next()->opp()
4. e->next()->opp()->next()
5. e->next()->opp()->next()->opp()
6. e->next()->opp()->next()->opp()->next()
7. e->next()->opp()->next()->opp()->next()->opp()
8. e->next()->opp()->next()->opp()->next()->opp()->next()

… until $e(->next()->opp())^n == e$

Digital Michelangelo

- In 1998 Marc Levoy takes a sabbatical year in Florence to scan a bunch of Michelangelo sculptures
- David at 1mm resolution
- St. Matthew at 290μm resolution



Edge-Face-Vertex

Half Edges

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

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

Faces

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

Vertices

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