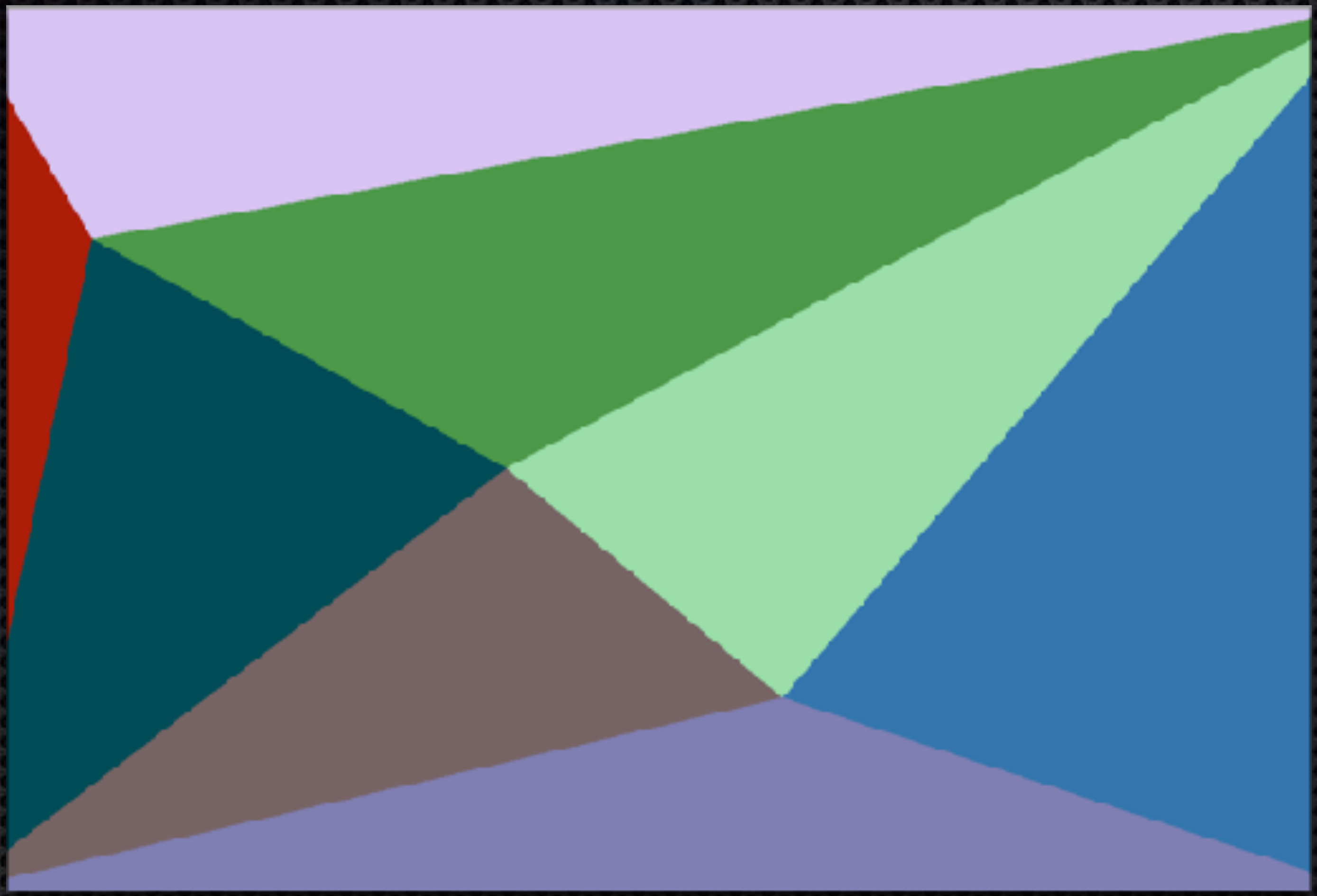


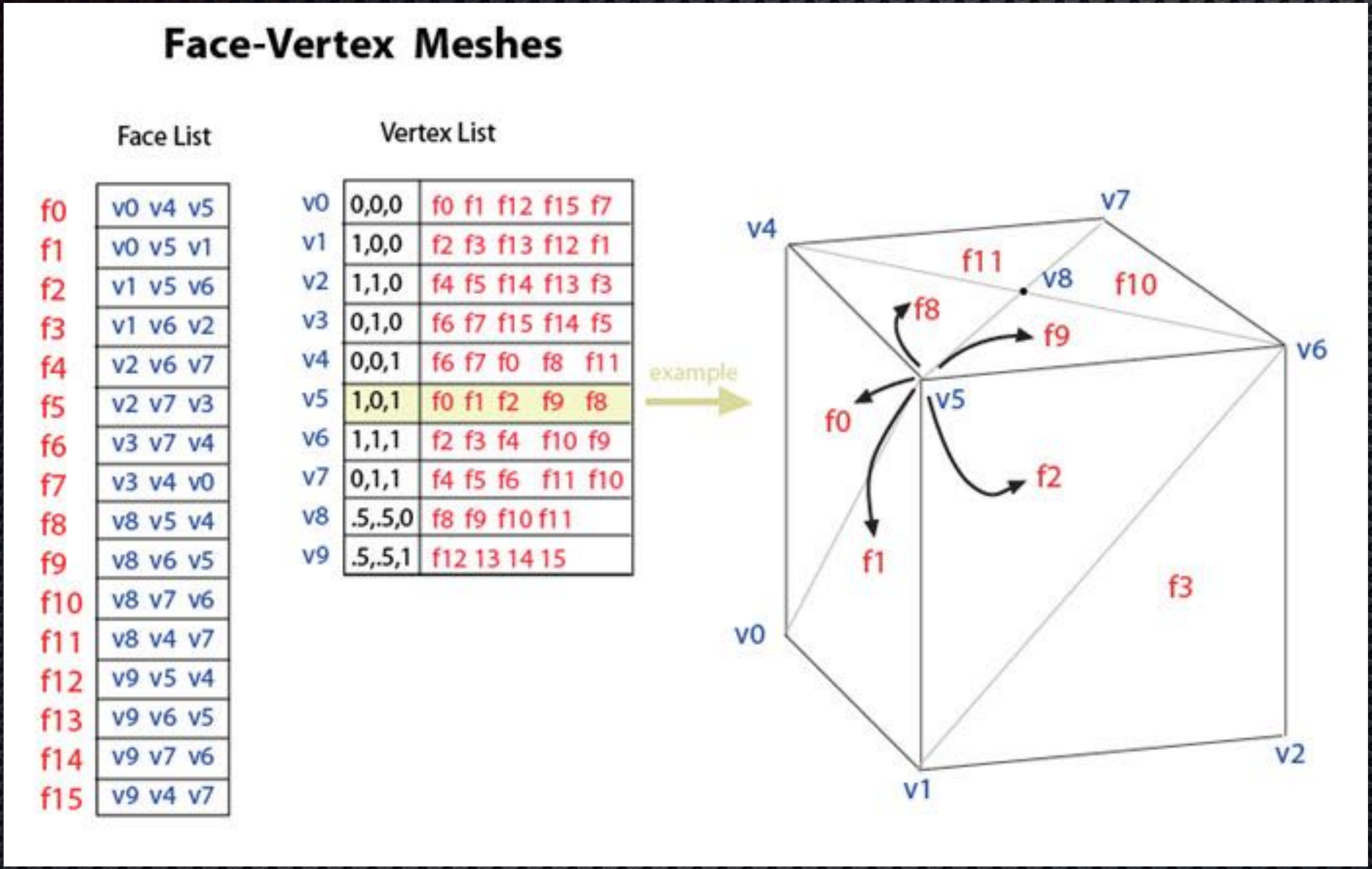
Half-Edge Data Structure

Pablo Burgos

Aug 1st

Surface Mesh Representation 3D

Face-Vertex



Face-Vertex Meshes

Face List

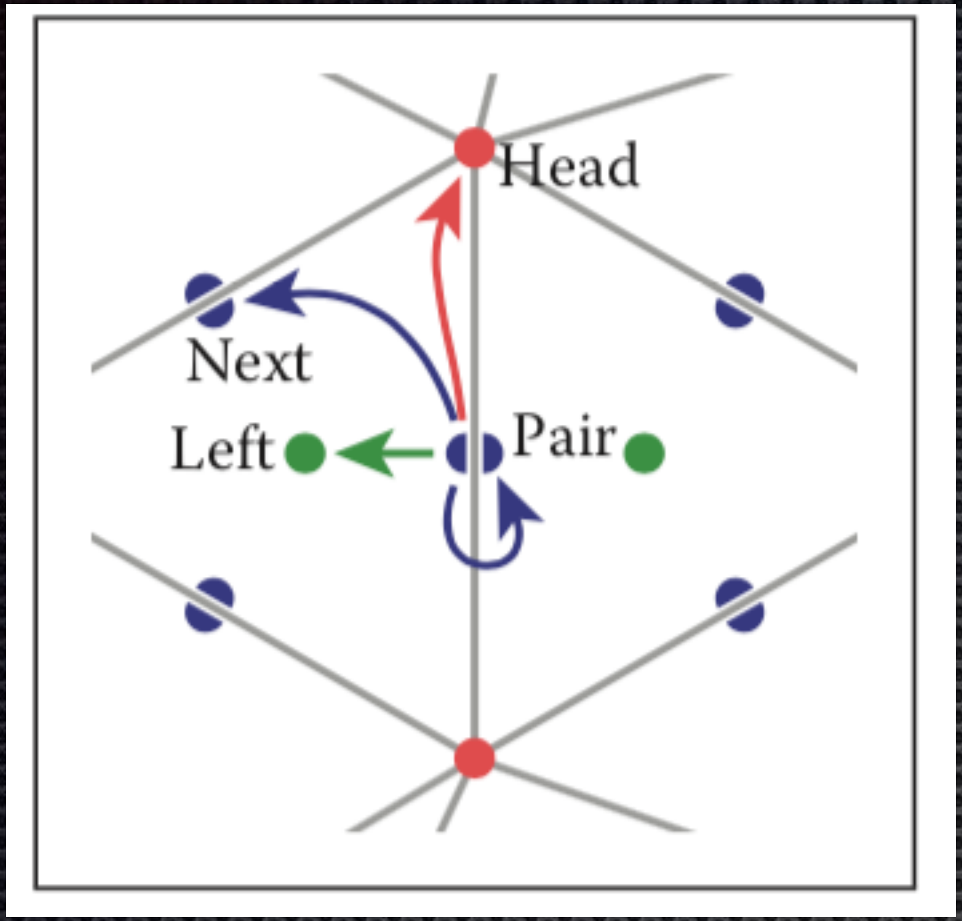
f0	v0 v4 v5
f1	v0 v5 v1
f2	v1 v5 v6
f3	v1 v6 v2
f4	v2 v6 v7
f5	v2 v7 v3
f6	v3 v7 v4
f7	v3 v4 v0
f8	v8 v5 v4
f9	v8 v6 v5
f10	v8 v7 v6
f11	v8 v4 v7
f12	v9 v5 v4
f13	v9 v6 v5
f14	v9 v7 v6
f15	v9 v4 v7

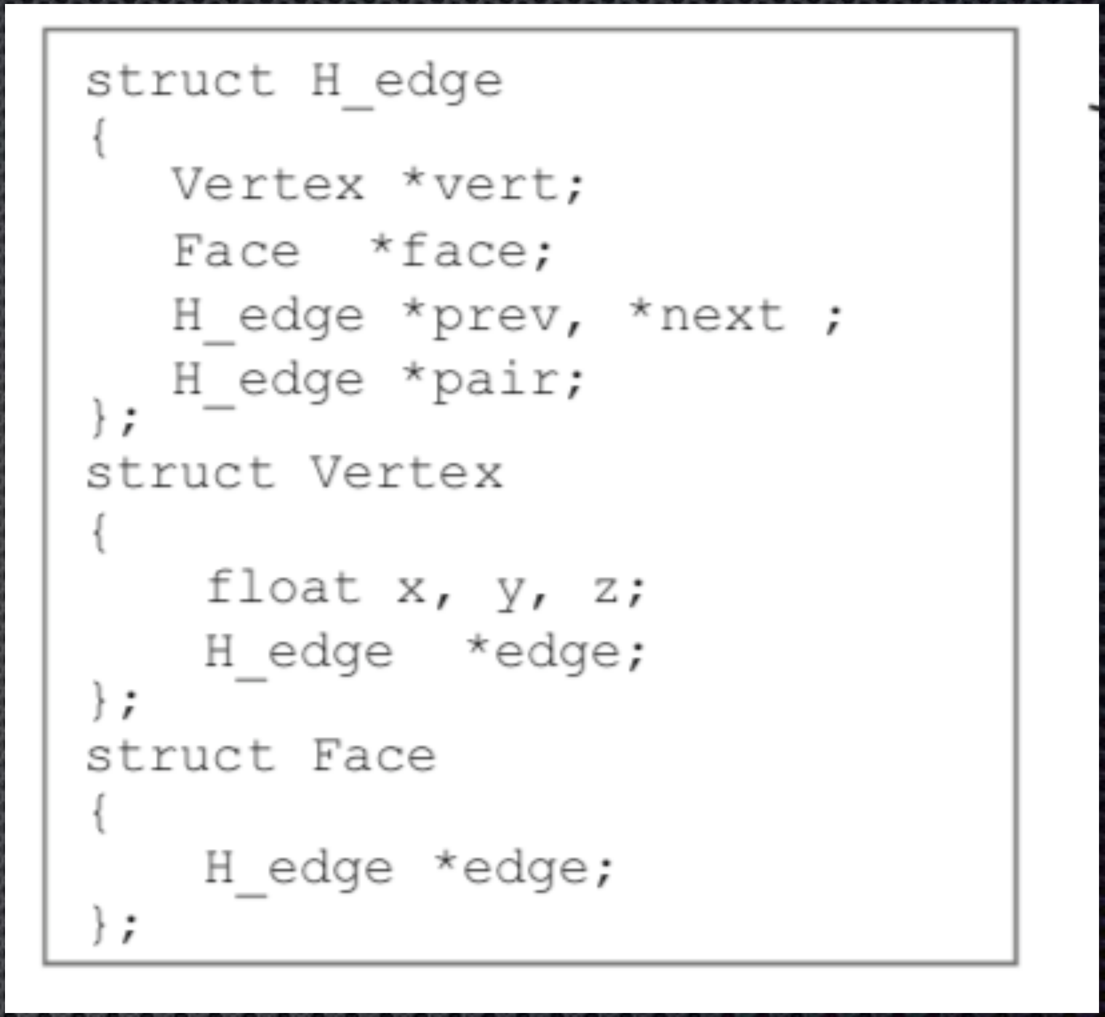
Vertex List

v0	0,0,0	f0 f1 f12 f15 f7
v1	1,0,0	f2 f3 f13 f12 f1
v2	1,1,0	f4 f5 f14 f13 f3
v3	0,1,0	f6 f7 f15 f14 f5
v4	0,0,1	f6 f7 f0 f8 f11
v5	1,0,1	f0 f1 f2 f9 f8
v6	1,1,1	f2 f3 f4 f10 f9
v7	0,1,1	f4 f5 f6 f11 f10
v8	.5,.5,0	f8 f9 f10 f11
v9	.5,.5,1	f12 13 14 15

https://en.wikipedia.org/wiki/Polygon_mesh

Surface Mesh Representation 3D
Half-Edge



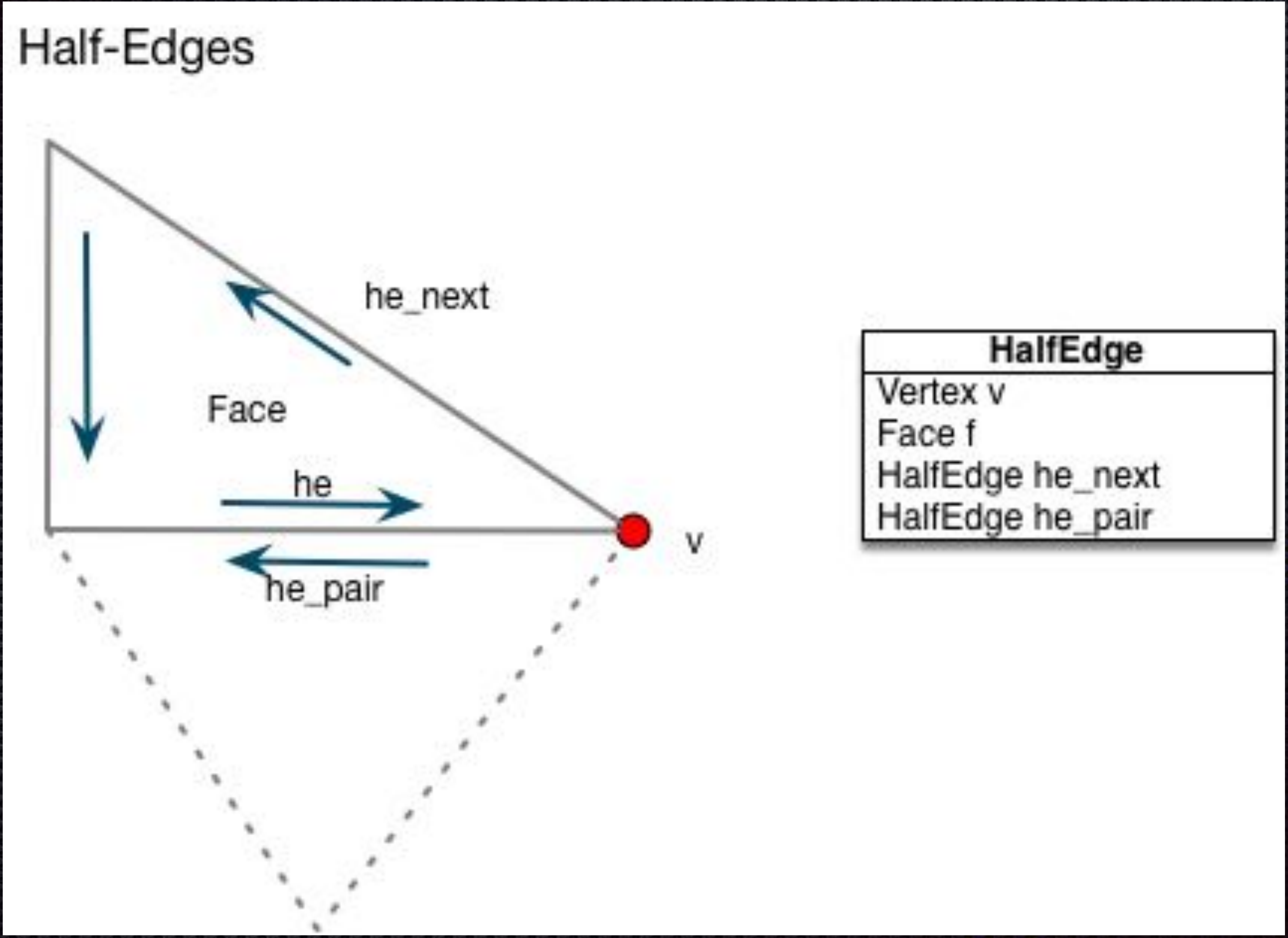


```
struct H_edge
{
   Vertex *vert;
   Face   *face;
   H_edge *prev, *next ;
   H_edge *pair;
};
struct Vertex
{
    float x, y, z;
    H_edge  *edge;
};
struct Face
{
    H_edge *edge;
};
```

- Allows for easier mesh modification (adding vertex, edges, cuts, etc)

Half-Edge Data Structure

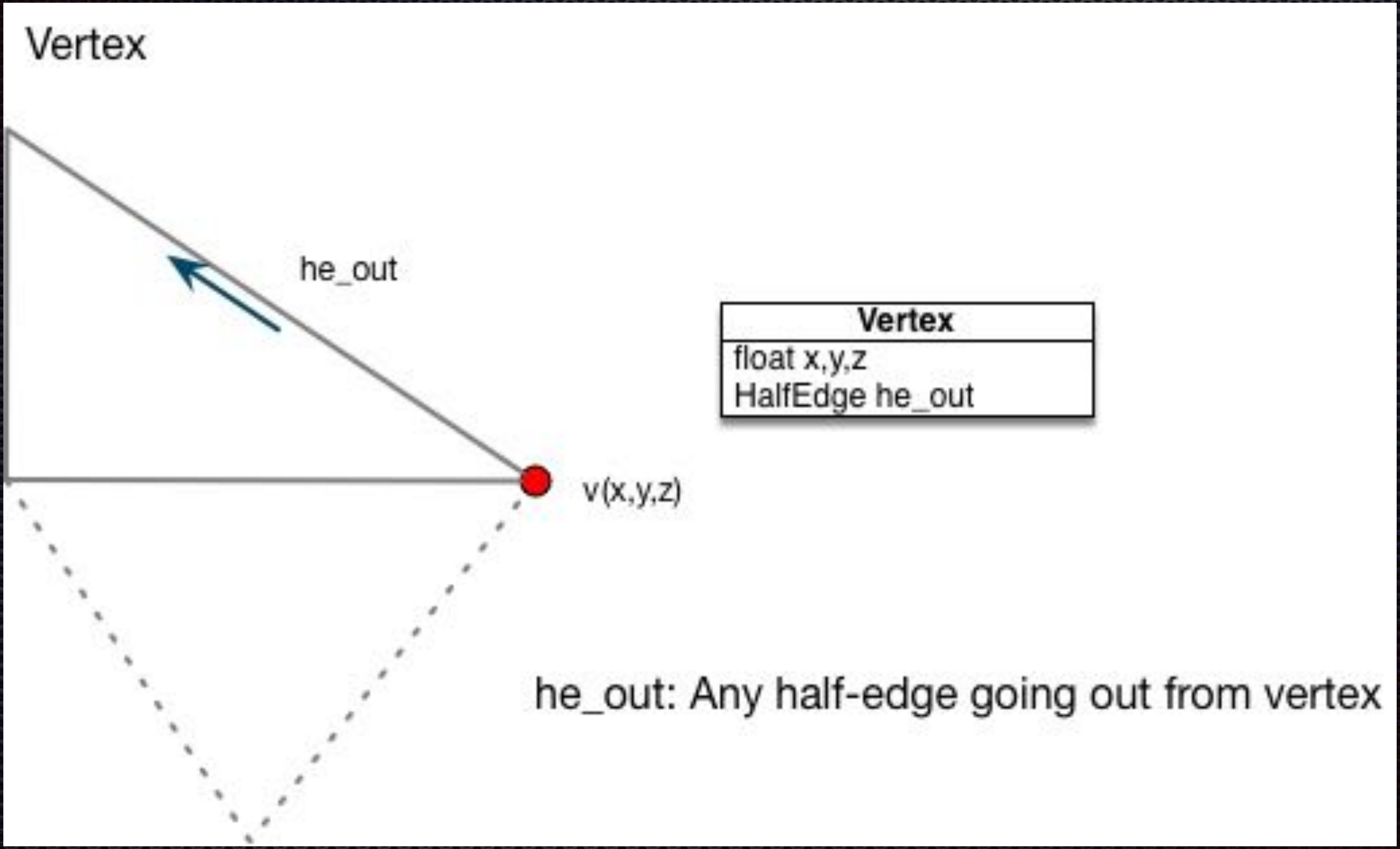
HalfEdge



HalfEdge
Vertex v
Face f
HalfEdge he_next
HalfEdge he_pair

Half-Edge Data Structure

Vertex



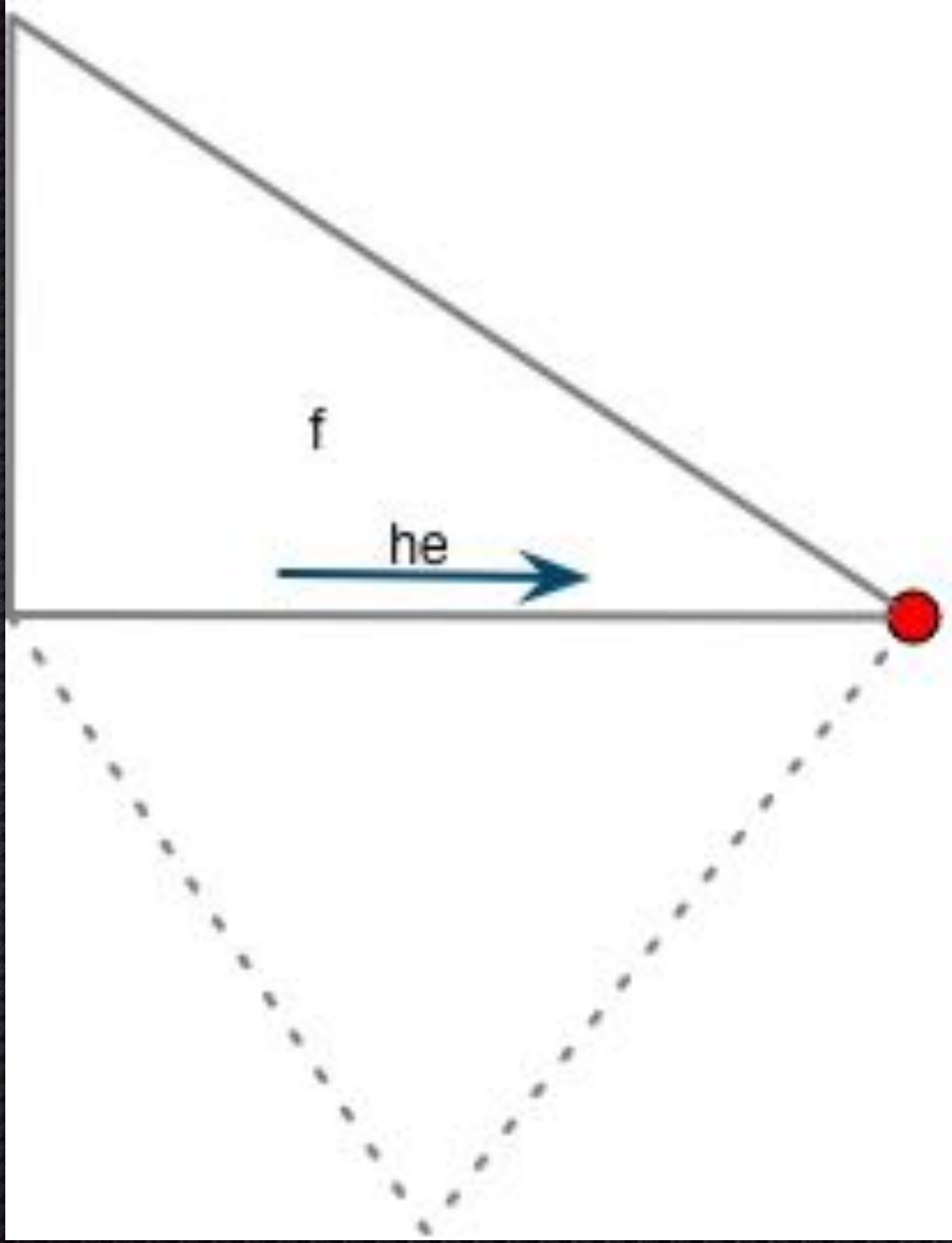
Vertex
float x,y,z HalfEdge he_out

he_out: Any half-edge going out from vertex

Half-Edge Data Structure

Face

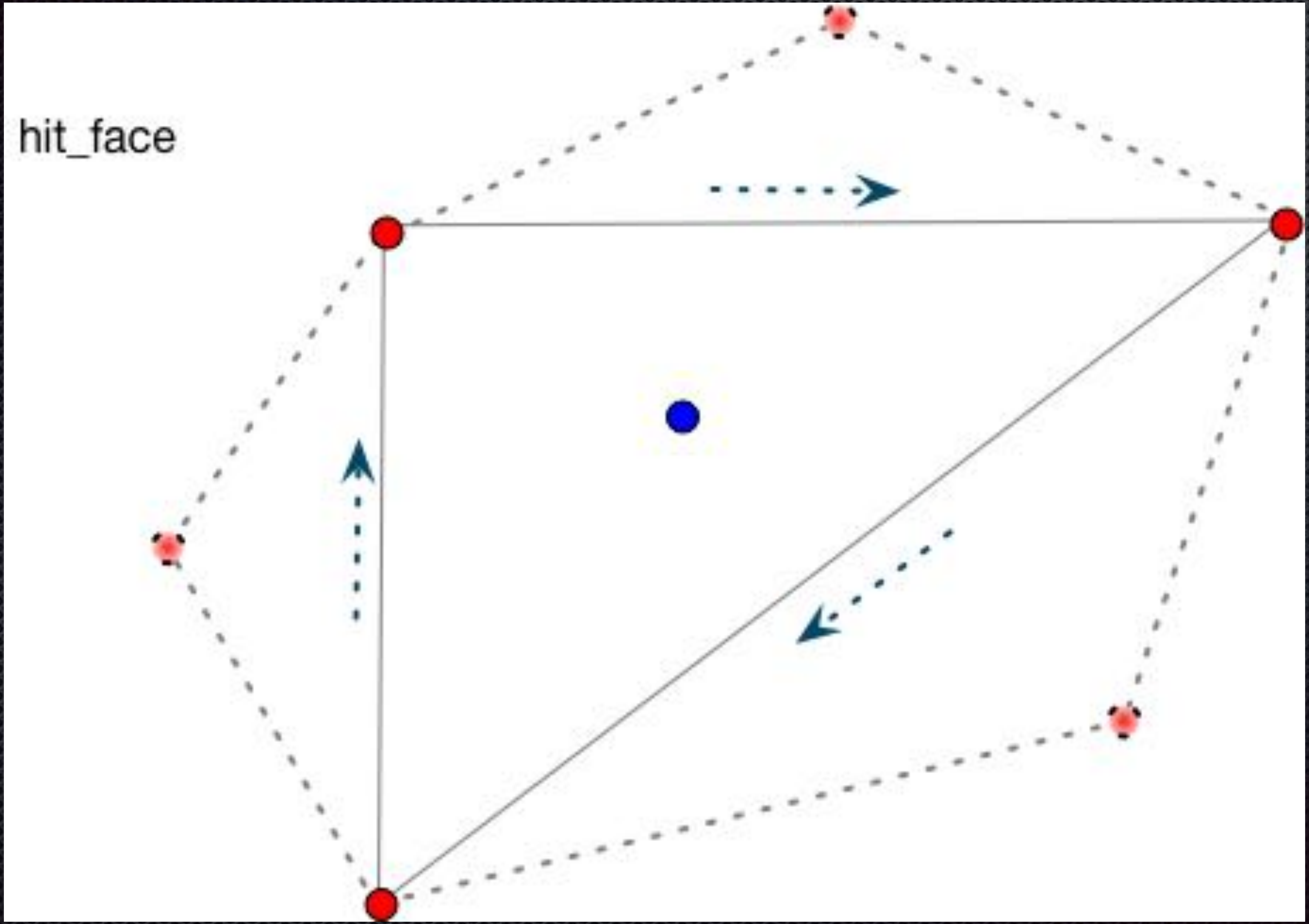
Face



Face
HalfEdge he

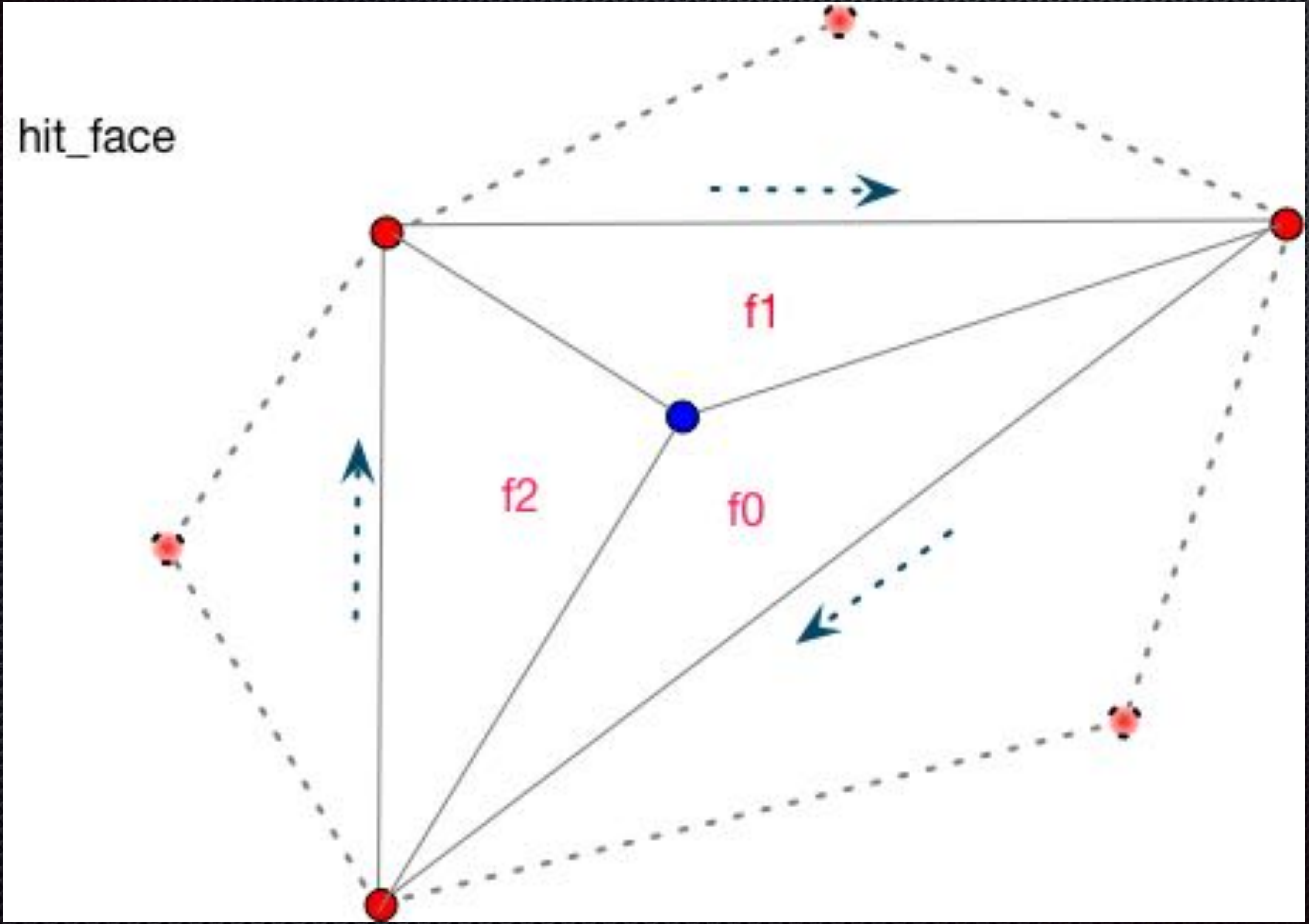
he: Any Half-Edge inside face

Delaunay
Insert Vertex

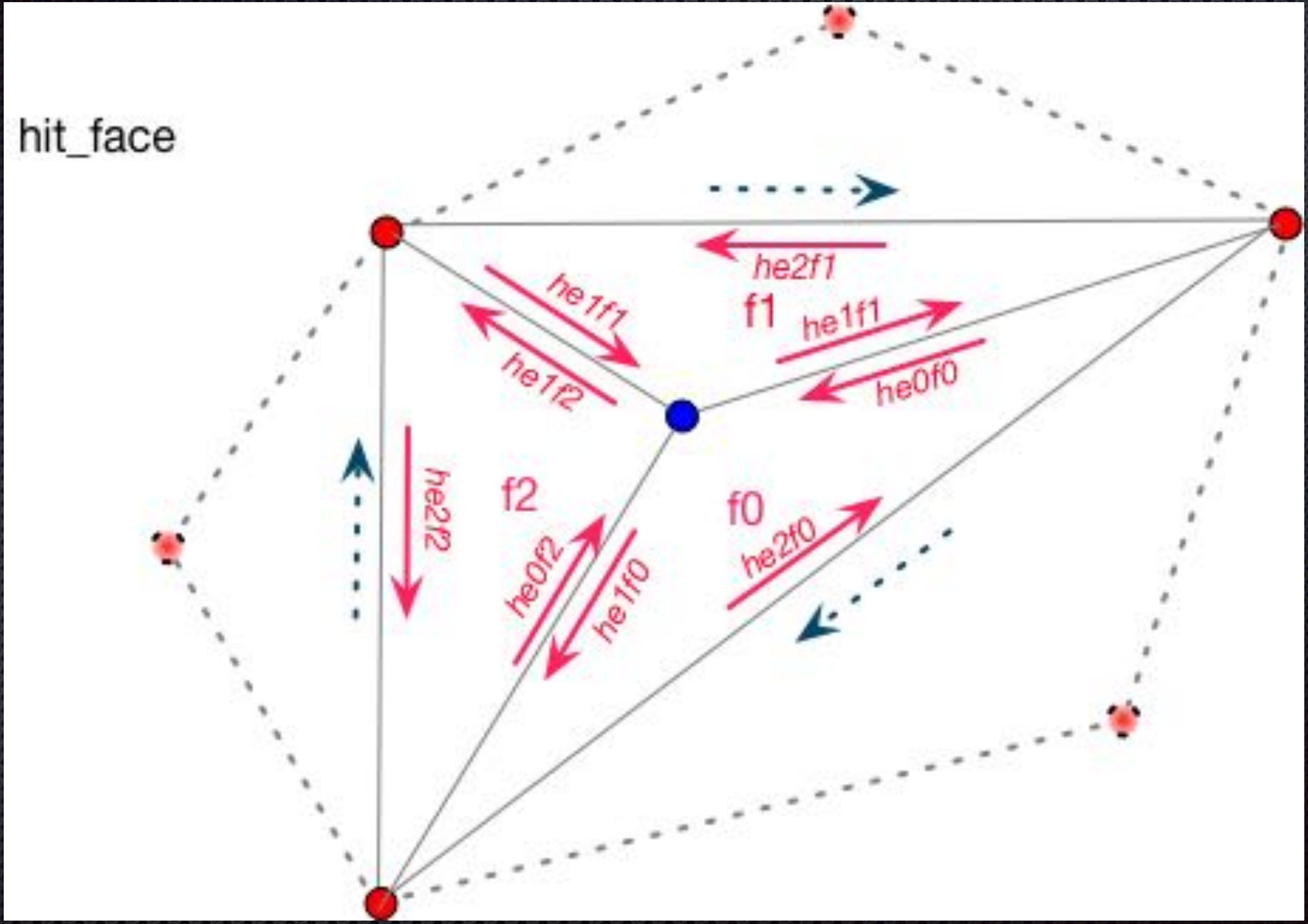


Delaunay

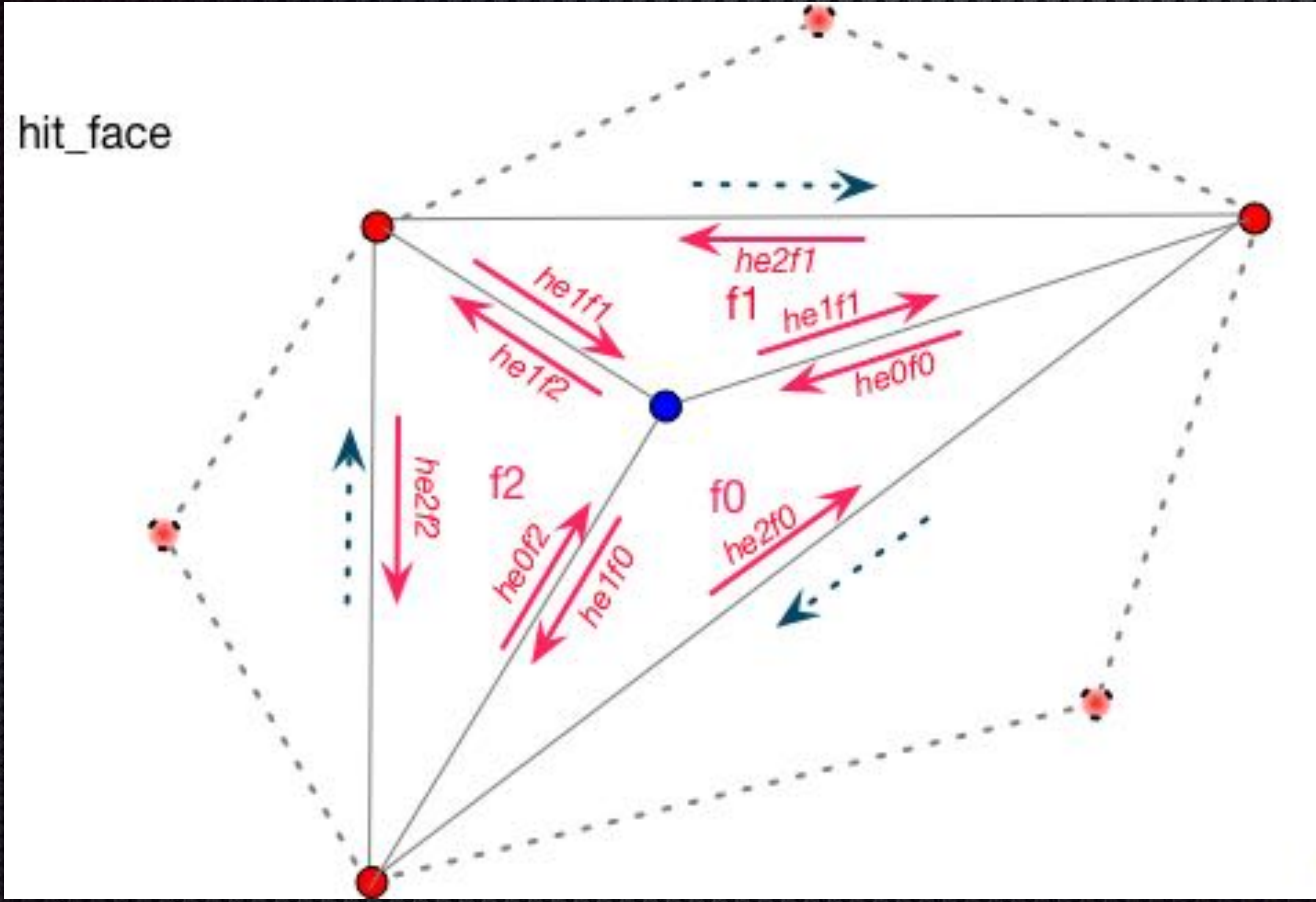
Insert Vertex



Delaunay
Insert Vertex

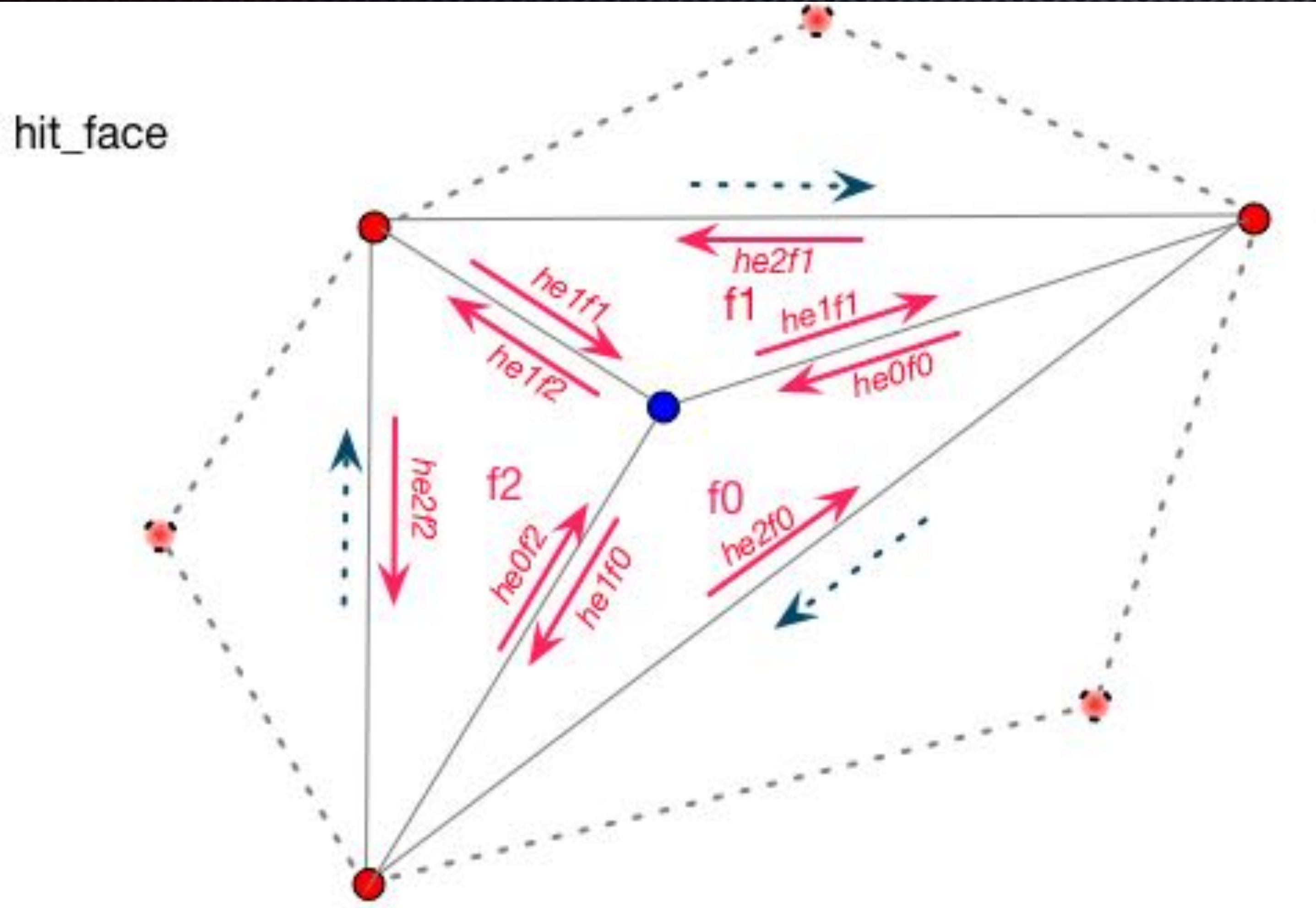


Delaunay
Insert Vertex



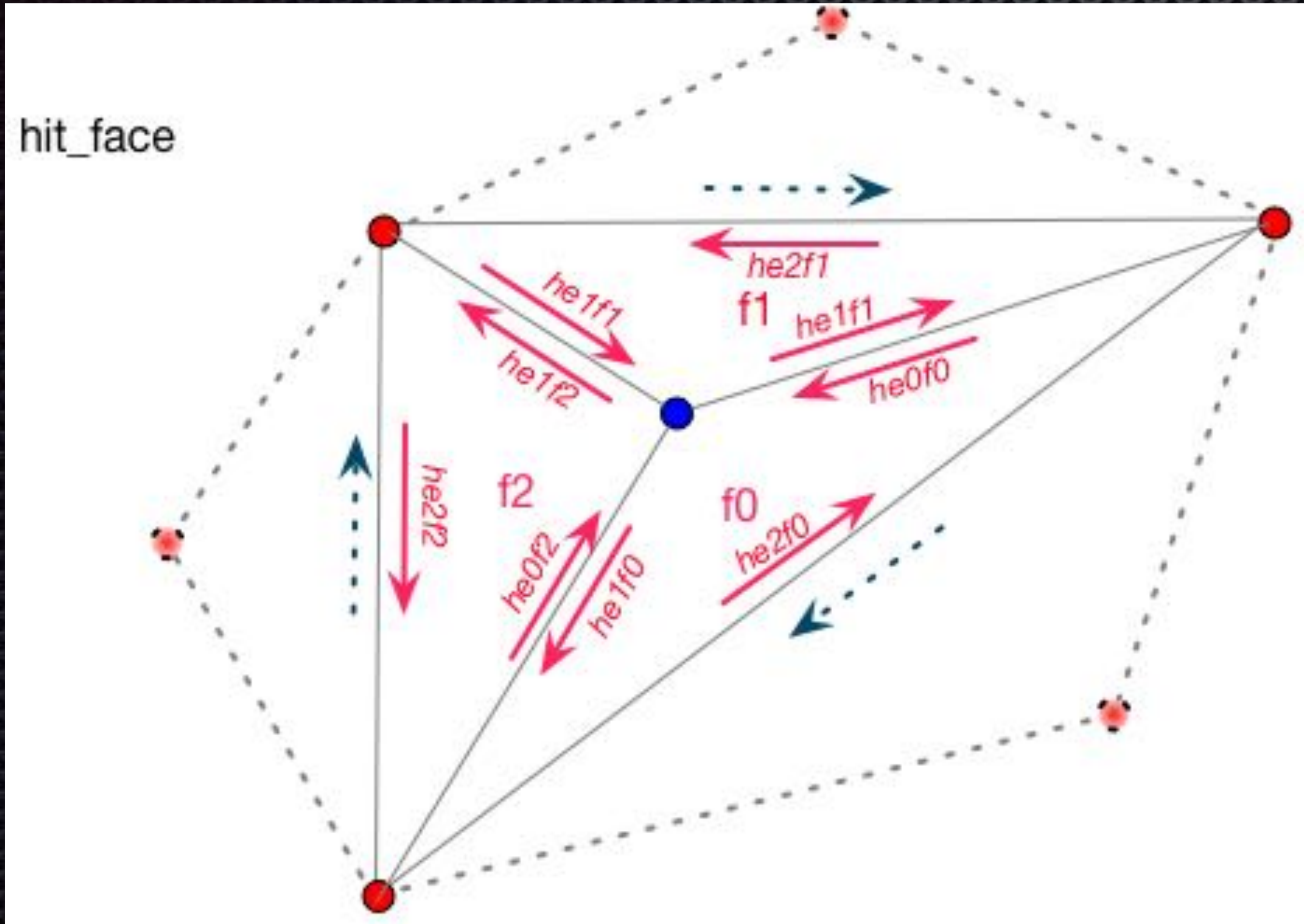
```
f0 = Face()
f1 = Face()
f2 = Face()

he0f0 = HEdge()
he1f0 = HEdge()
he2f0 = HEdge()

he0f1 = HEdge()
he1f1 = HEdge()
he2f1 = HEdge()

he0f2 = HEdge()
he1f2 = HEdge()
he2f2 = HEdge()
```

Delaunay
Insert Vertex



```
f0.set_h_edge(he0f0)
f1.set_h_edge(he0f1)
f2.set_h_edge(he0f2)

he0f0.set_vertex(vertex)
he0f0.set_face(f0)
he0f0.set_he_next(he1f0)
he0f0.set_he_pair(he1f1)

he1f0.set_vertex(hit_face.h_edge.he_next.he_next.vertex)
he1f0.set_face(f0)
he1f0.set_he_next(he2f0)
he1f0.set_he_pair(he0f2)

he2f0.set_vertex(hit_face.h_edge.vertex)
he2f0.set_face(f0)
he2f0.set_he_next(he0f0)
he2f0.set_he_pair(hit_face.h_edge.he_pair)
```

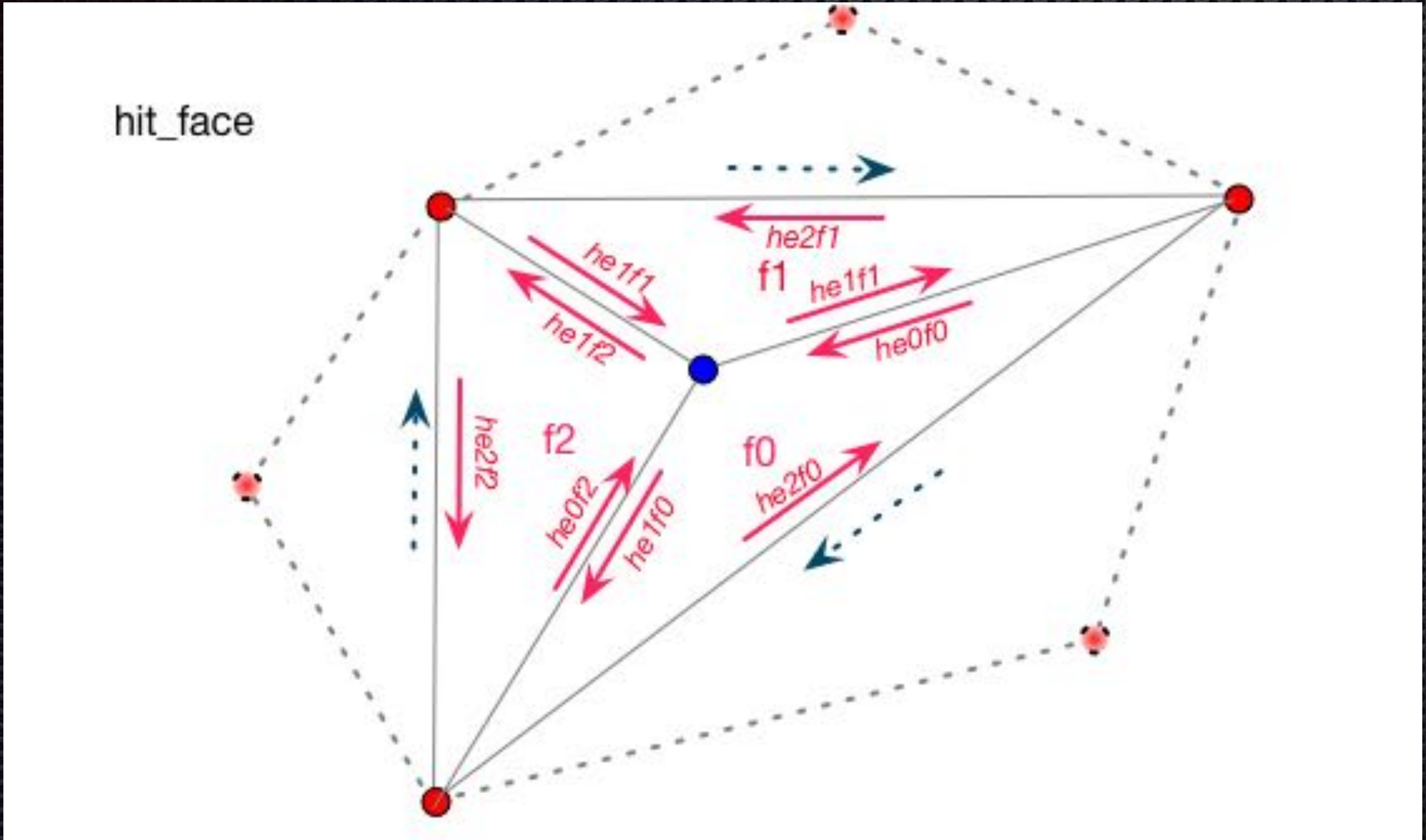
Delaunay
Insert Vertex



```
# add new faces

self.faces[f0.uuid] = f0
self.faces[f1.uuid] = f1
self.faces[f2.uuid] = f2

# add new vertex

self.vertexes[vertex.uuid] = vertex
self.vertexes[vertex.uuid].set_h_edge(he1f0)

# add new half edges

self.h_edges[he0f0.uuid] = he0f0
self.h_edges[he1f0.uuid] = he1f0
self.h_edges[he2f0.uuid] = he2f0
self.h_edges[he0f1.uuid] = he0f1
self.h_edges[he1f1.uuid] = he1f1
self.h_edges[he2f1.uuid] = he2f1
self.h_edges[he0f2.uuid] = he0f2
self.h_edges[he1f2.uuid] = he1f2
self.h_edges[he2f2.uuid] = he2f2
```

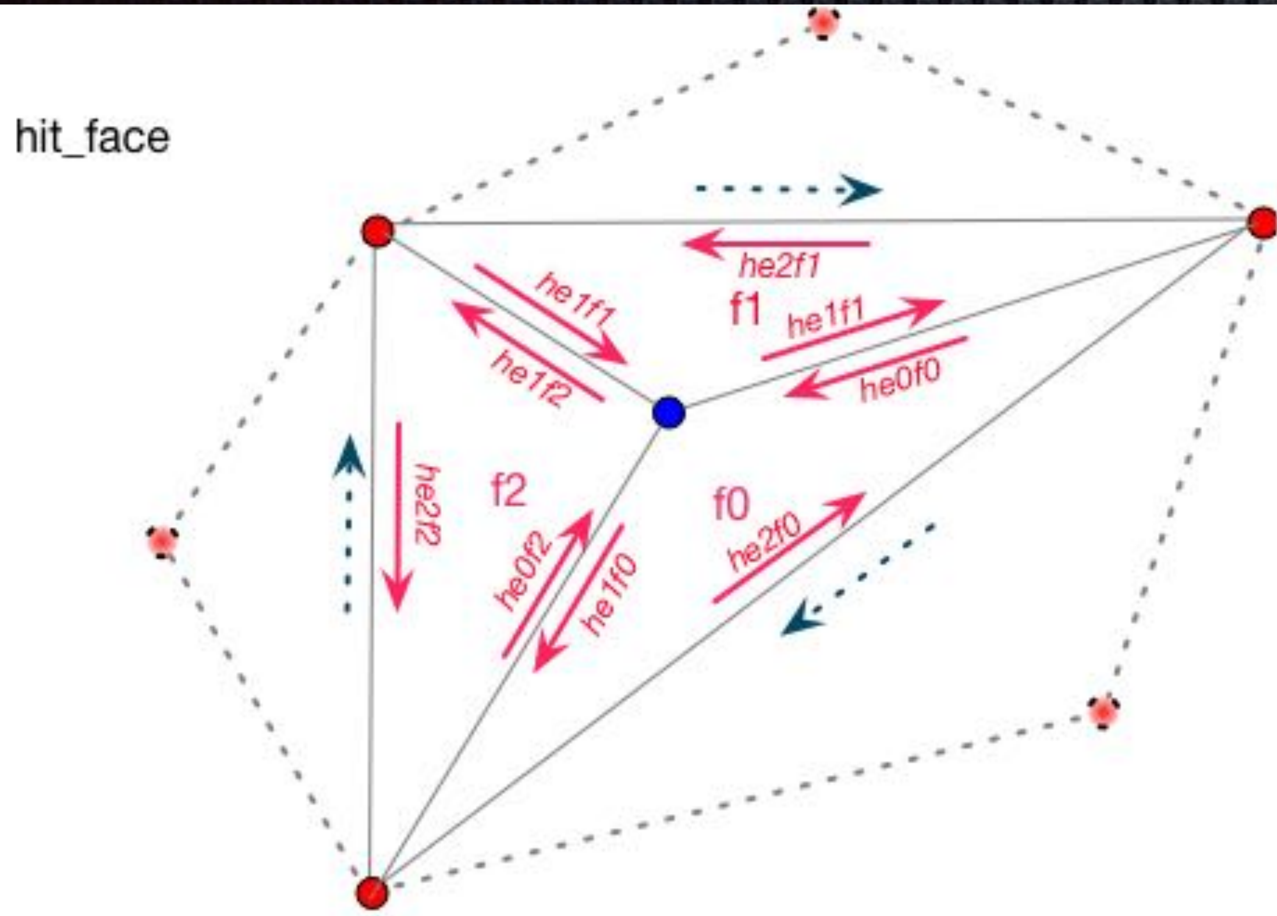
Delaunay

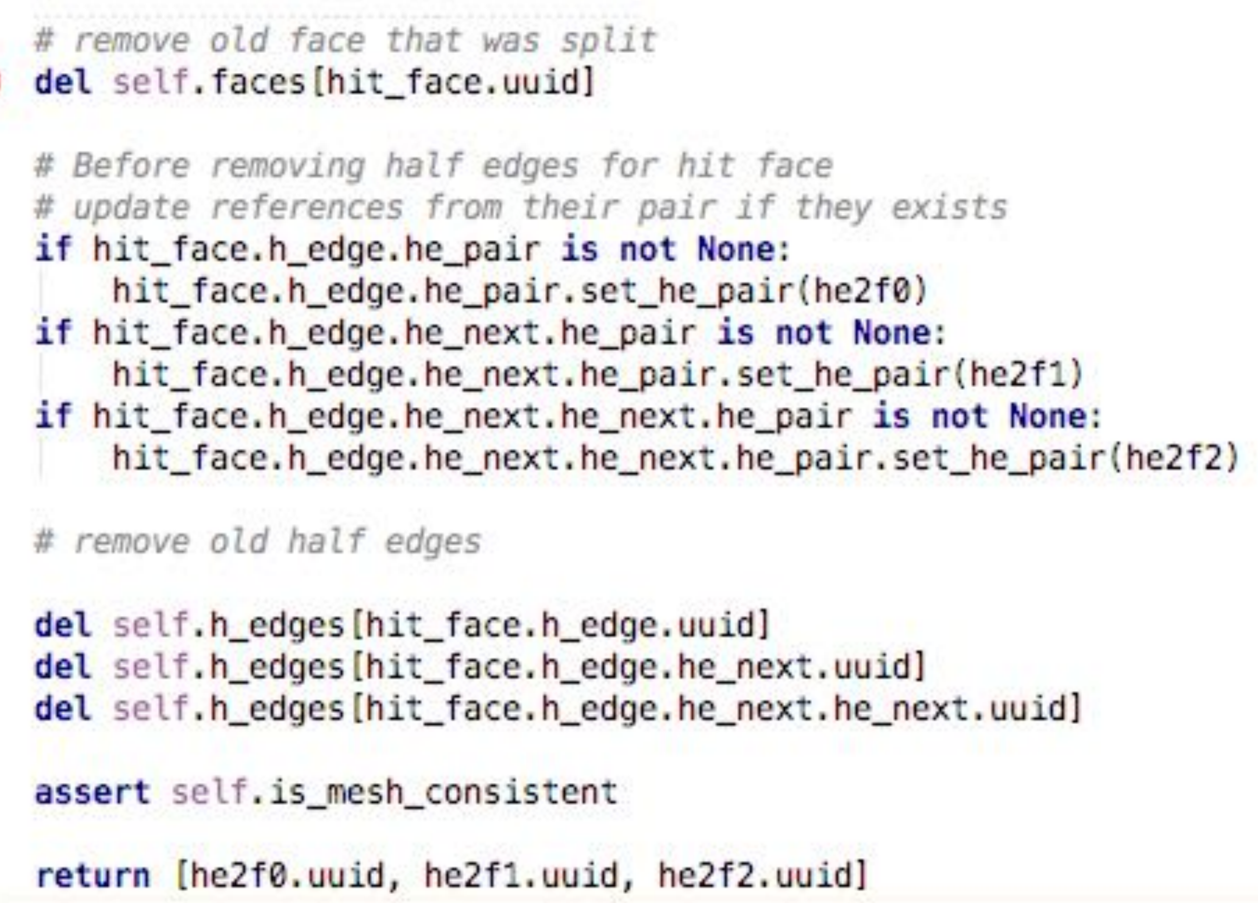
Insert Vertex



```
# update vertexes reference

self.vertexes[hit_face.h_edge.vertex.uuid].set_h_edge(he0f0)
self.vertexes[hit_face.h_edge.he_next.vertex.uuid].set_h_edge(he0f1)
self.vertexes[hit_face.h_edge.he_next.he_next.vertex.uuid].set_h_edge(he0f2)
```

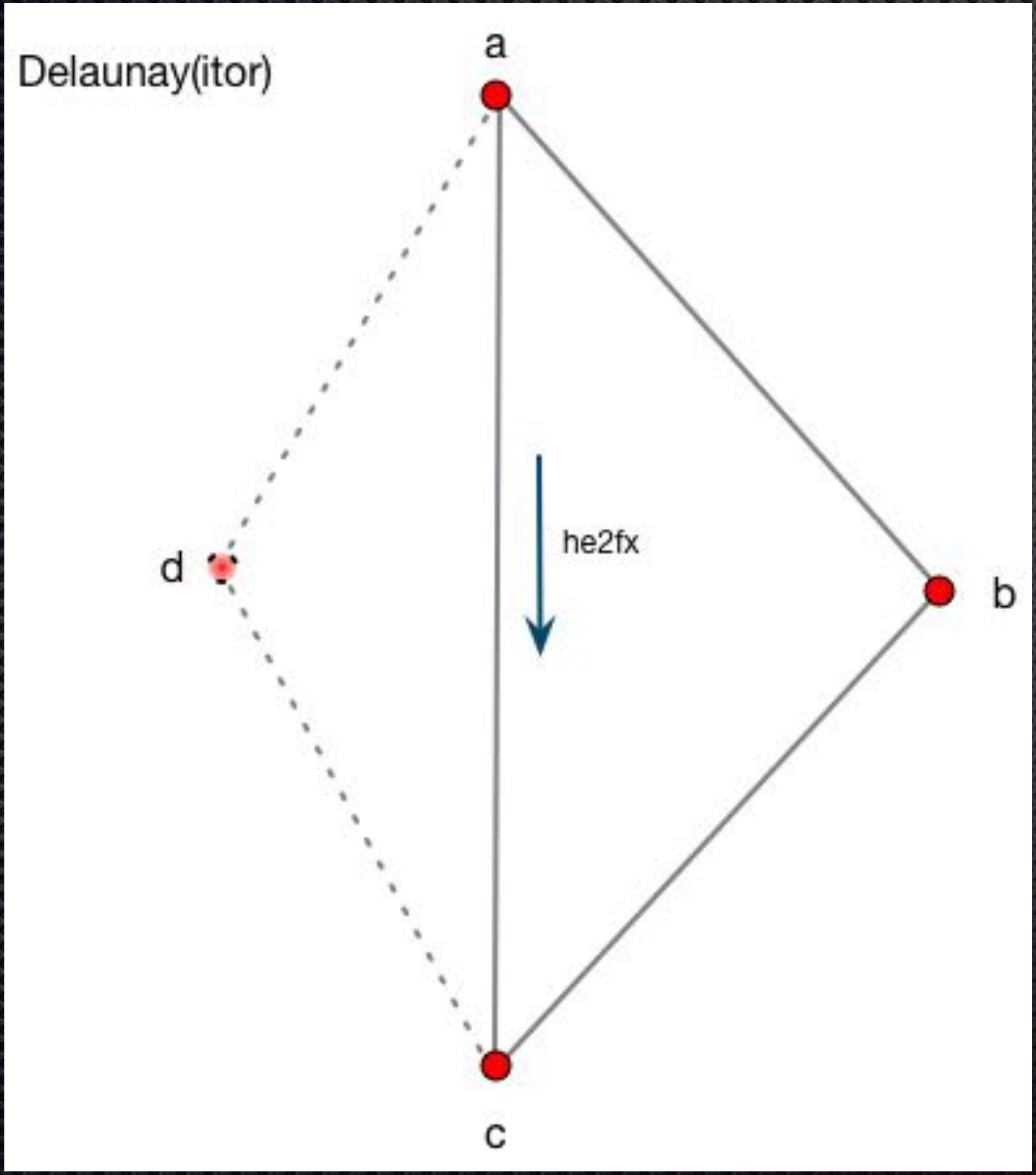
Delaunay
Insert Vertex



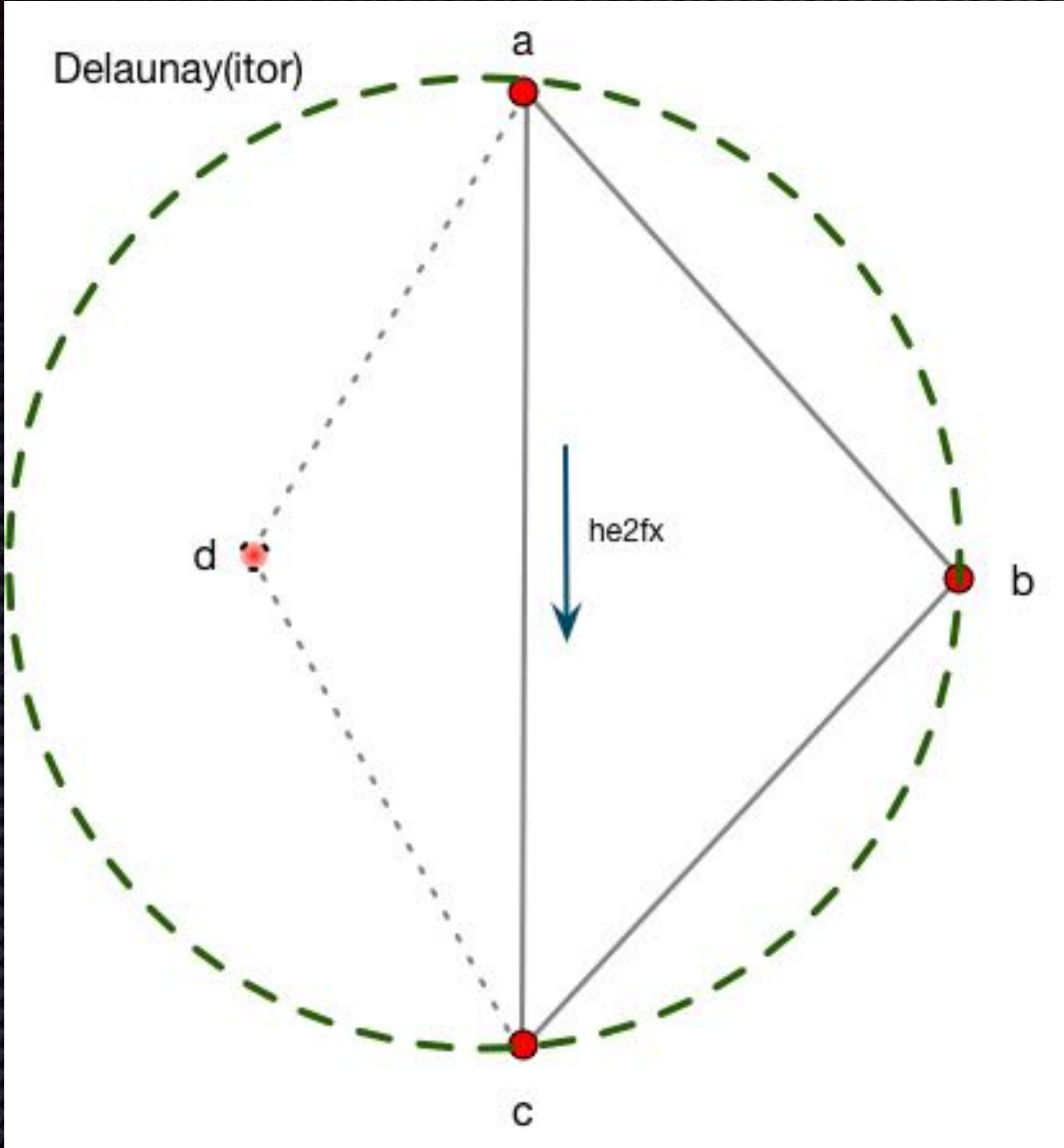


```
# remove old face that was split
del self.faces[hit_face.uuid]

# Before removing half edges for hit face
# update references from their pair if they exists
if hit_face.h_edge.he_pair is not None:
    hit_face.h_edge.he_pair.set_he_pair(he2f0)
if hit_face.h_edge.he_next.he_pair is not None:
    hit_face.h_edge.he_next.he_pair.set_he_pair(he2f1)
if hit_face.h_edge.he_next.he_next.he_pair is not None:
    hit_face.h_edge.he_next.he_next.he_pair.set_he_pair(he2f2)

# remove old half edges

del self.h_edges[hit_face.h_edge.uuid]
del self.h_edges[hit_face.h_edge.he_next.uuid]
del self.h_edges[hit_face.h_edge.he_next.he_next.uuid]

assert self.is_mesh_consistent

return [he2f0.uuid, he2f1.uuid, he2f2.uuid]
```

Delaunay
FlipEdge

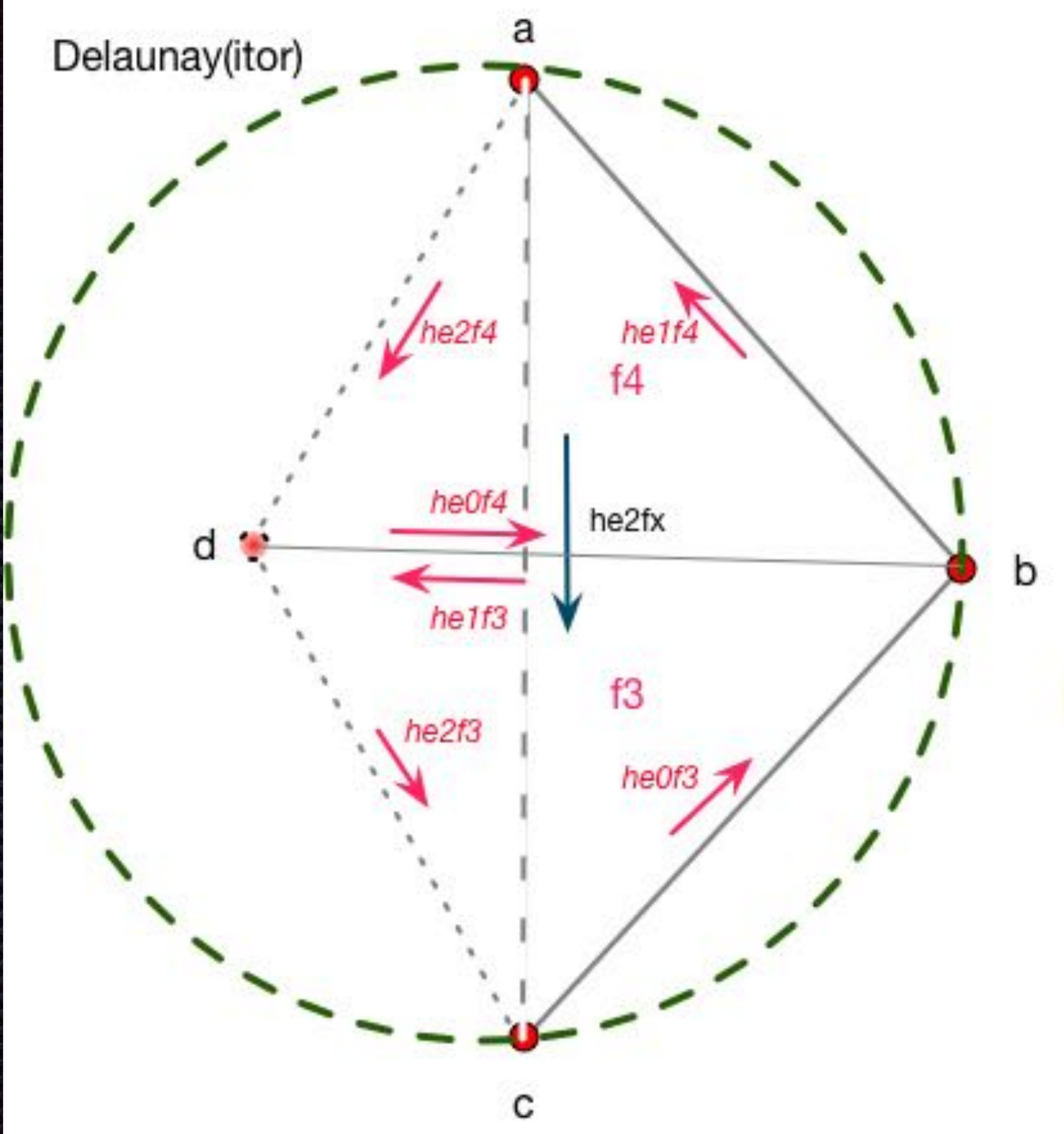


Delaunay
FlipEdge



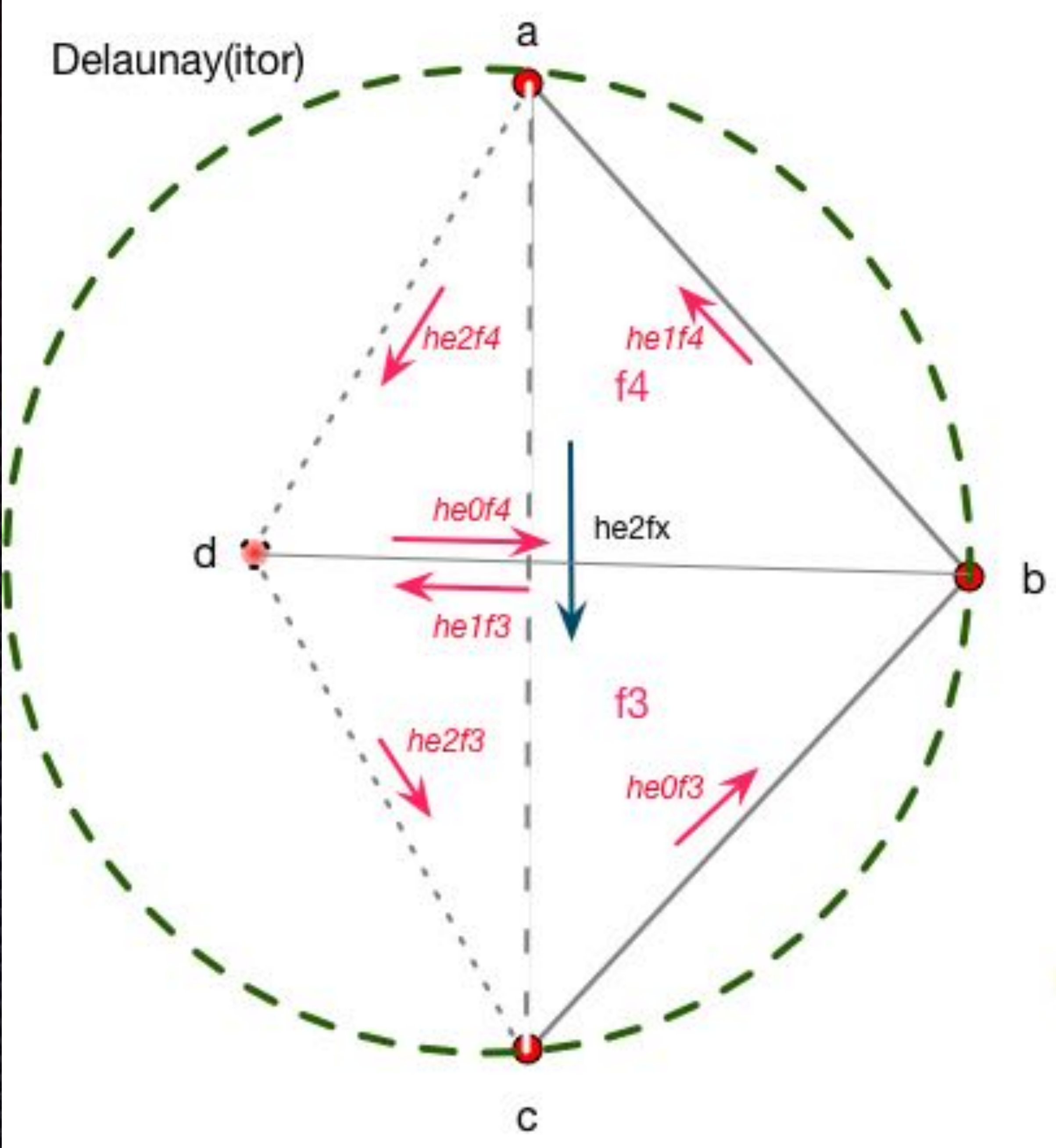
```python
if h_edge.he_pair is None:
    print("Half edge with no pair....boundary")
elif h_edge.is_in_circle():
    print("How we need to delaunize")
    # This step creates two new faces and del two old ones
    f3 = Face()
    f4 = Face()

    he2f3 = HEdge()
    he0f3 = HEdge()
    he1f3 = HEdge()

    he2f4 = HEdge()
    he0f4 = HEdge()
    he1f4 = HEdge()
```

Delaunay
FlipEdge



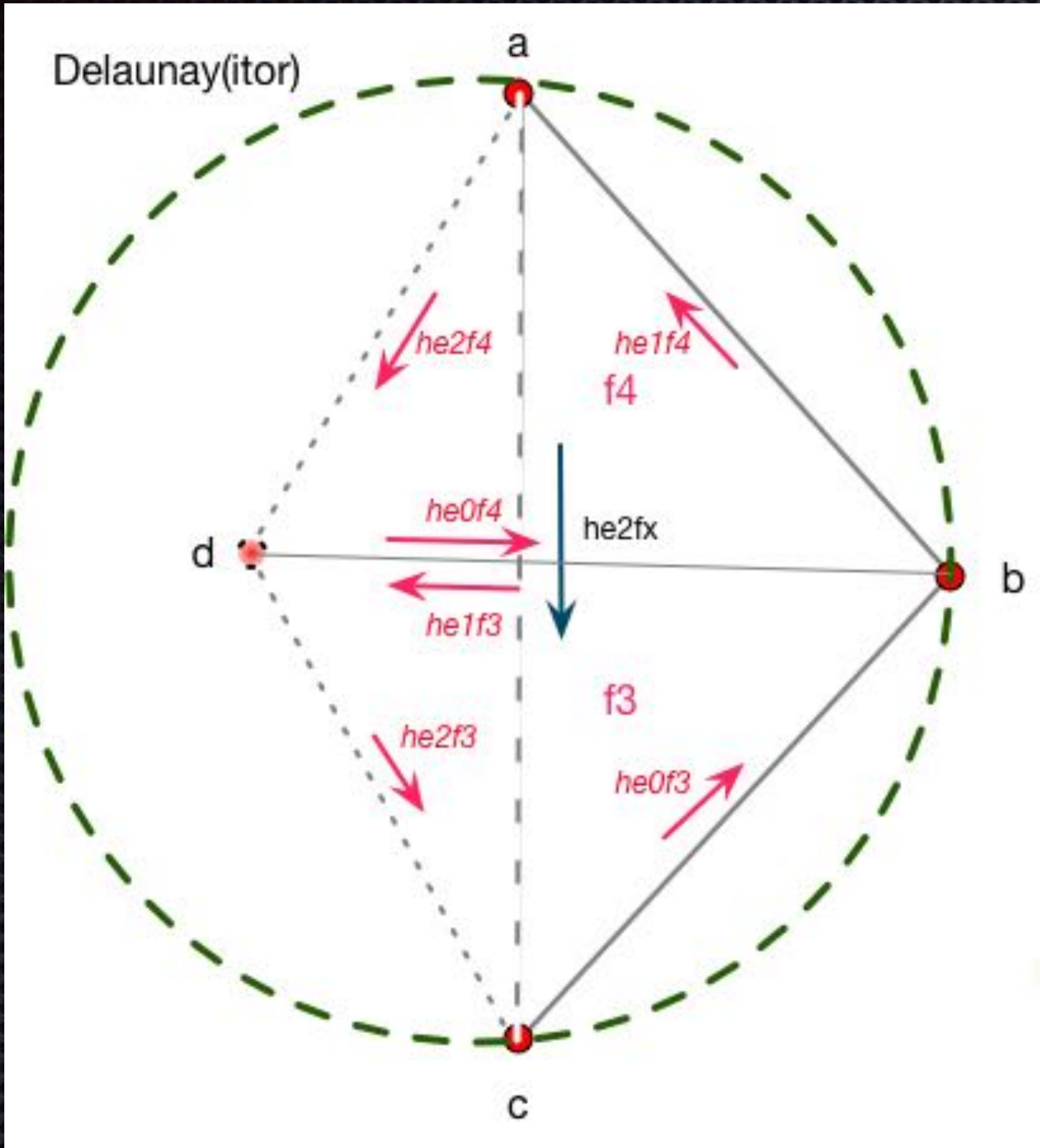
```
f3 = Face()
f4 = Face()

he2f3 = HEdge()
he0f3 = HEdge()
he1f3 = HEdge()

he2f4 = HEdge()
he0f4 = HEdge()
he1f4 = HEdge()

# Associate h_edges references to faces
f3.set_h_edge(he0f3)
f4.set_h_edge(he0f4)
    he1f4 = HEdge()
```

Delaunay
FlipEdge



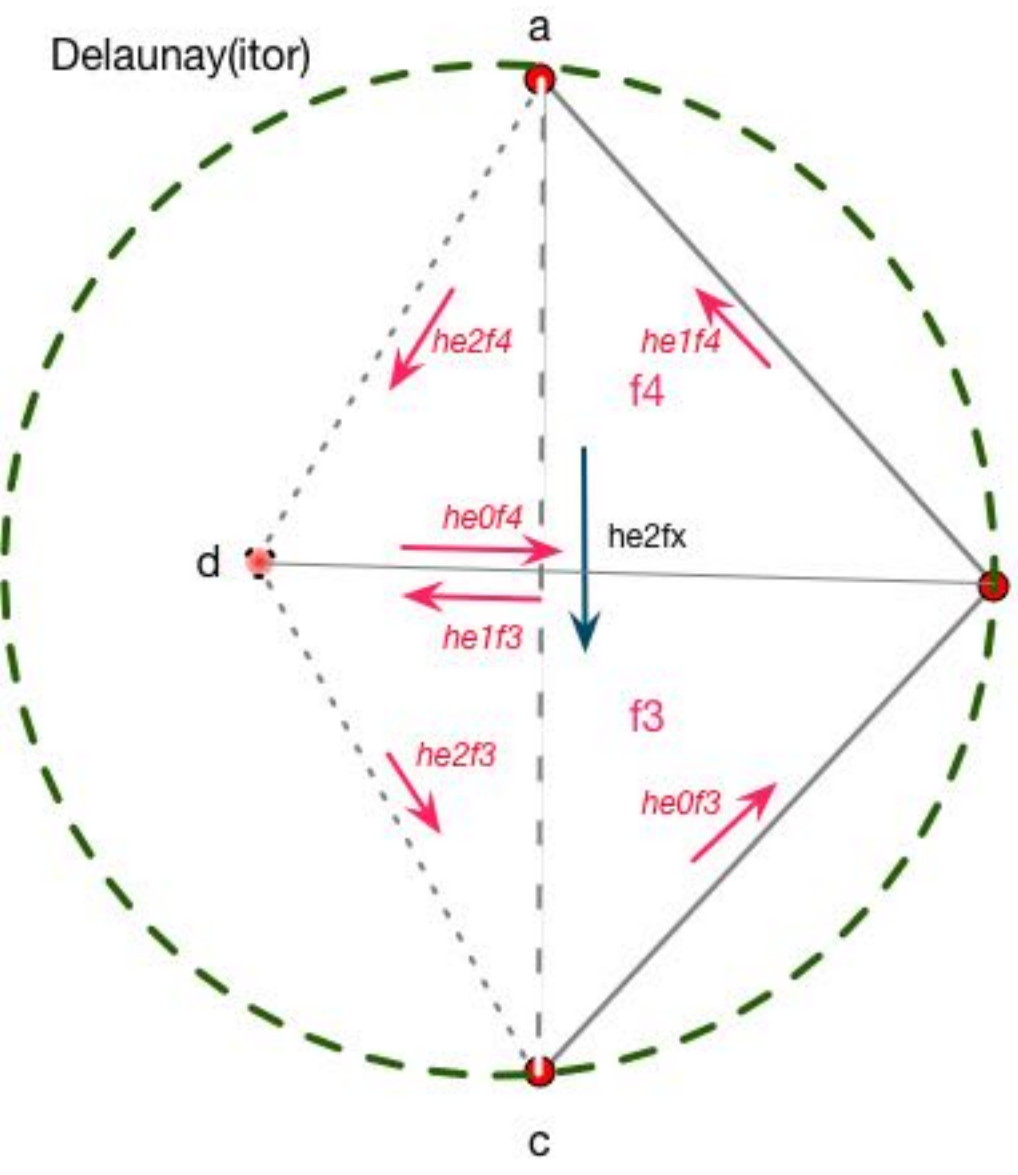
```
# Let's build half edges
he2f3.set_vertex(h_edge.he_pair.he_next.he_next.vertex)
he2f3.set_face(f3)
he2f3.set_he_next(he0f3)
he2f3.set_he_pair(h_edge.he_pair.he_next.he_next.he_pair)
if h_edge.he_pair.he_next.he_next.he_pair is not None:
    h_edge.he_pair.he_next.he_next.he_pair.set_he_pair(he2f3) #

he0f3.set_vertex(h_edge.he_next.vertex)
he0f3.set_face(f3)
he0f3.set_he_next(he1f3)
he0f3.set_he_pair(h_edge.he_next.he_pair)
if h_edge.he_next.he_pair is not None:
    h_edge.he_next.he_pair.set_he_pair(he0f3) #important to upc

he1f3.set_vertex(h_edge.he_pair.he_next.vertex)
he1f3.set_face(f3)
he1f3.set_he_next(he2f3)
he1f3.set_he_pair(he0f4)
```

Delaunay
FlipEdge



```
# Now vertices needs to be updated with new references
# 4 vertices to be updated with outgoing h_edges

va = h_edge.he_pair.vertex
vb = h_edge.he_pair.he_next.vertex
vc = h_edge.vertex
vd = h_edge.he_next.vertex

self.vertexes[va.uuid].set_h_edge(he2f4)
self.vertexes[vb.uuid].set_h_edge(he2f3)
self.vertexes[vc.uuid].set_h_edge(he0f3)
self.vertexes[vd.uuid].set_h_edge(he1f3)

# Add new faces to dictionary
self.faces[f3.uuid] = f3
self.faces[f4.uuid] = f4

# Add new half edges to dictionary
self.h_edges[he0f3.uuid] = he0f3
self.h_edges[he1f3.uuid] = he1f3
self.h_edges[he2f3.uuid] = he2f3
self.h_edges[he0f4.uuid] = he0f4
self.h_edges[he1f4.uuid] = he1f4
self.h_edges[he2f4.uuid] = he2f4
```

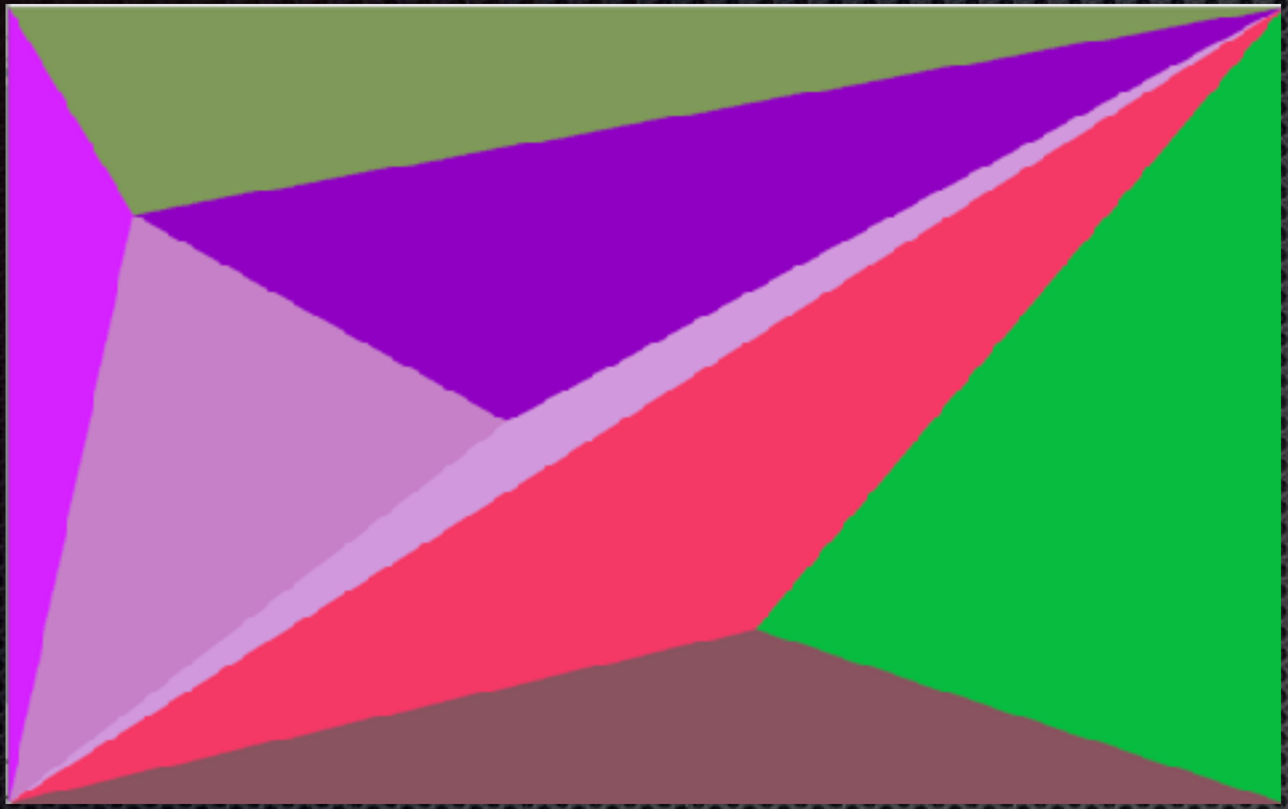
Delaunay
FlipEdge

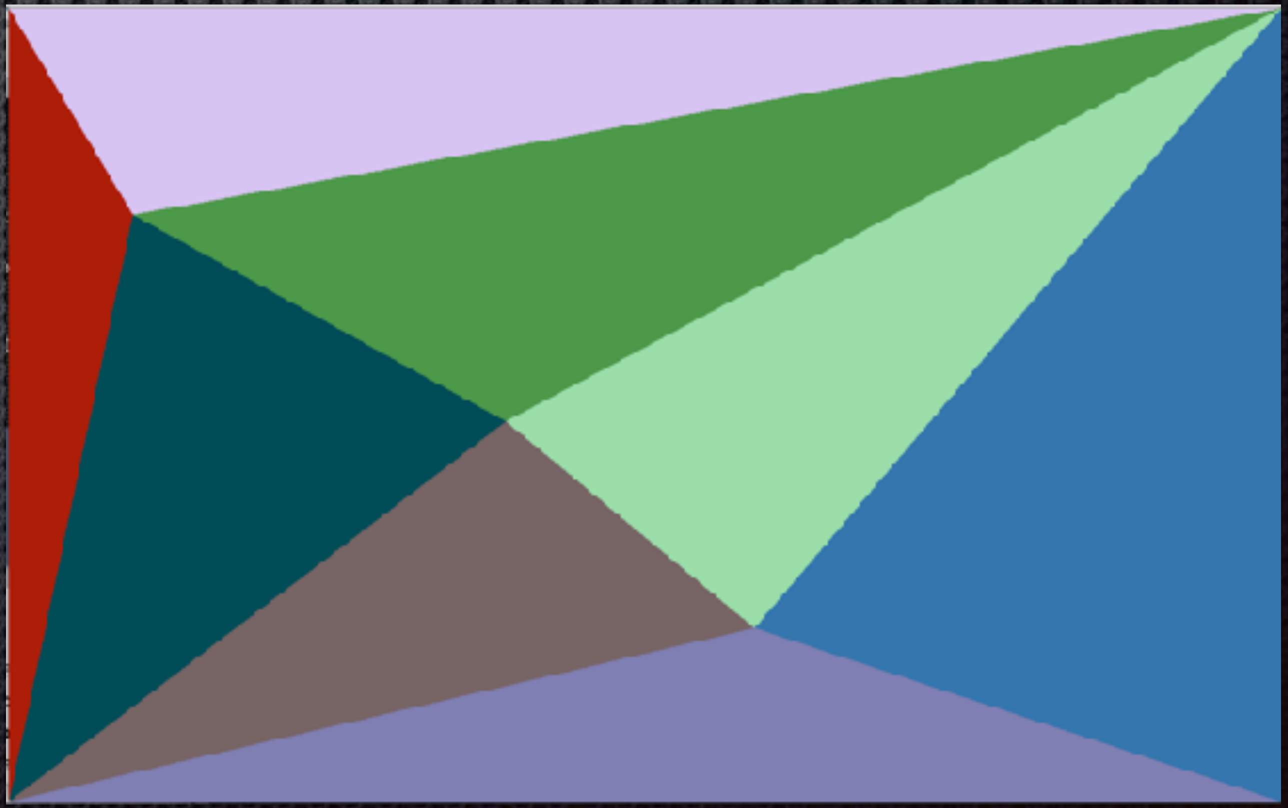


```
#Well all set...
# Let's remove old faces and h_edges destroyed by new faces
del self.faces[h_edge.face.uuid]
del self.faces[h_edge.he_pair.face.uuid]
# Let's remove old  half edges
del self.h_edges[h_edge.he_next.he_next.uuid]
del self.h_edges[h_edge.he_next.uuid]
del self.h_edges[h_edge.he_pair.he_next.he_next.uuid]
del self.h_edges[h_edge.he_pair.he_next.uuid]
del self.h_edges[h_edge.he_pair.uuid]
del self.h_edges[h_edge.uuid]


assert self.is_mesh_consistent

# Finally let's see whether delaunnization needs to be propagated
self.delaunize([he2f3.uuid,he2f4.uuid])
```

Mesh Data Structures 3D

Half-Edge Rocks!





Gracias