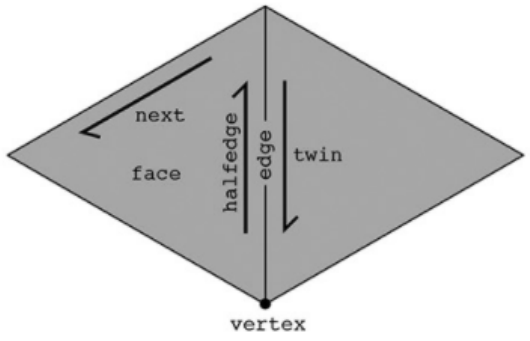
HALF-EDGES AND RAY-SURFACE INTERSECTION
4

CS184: COMPUTER GRAPHICS AND IMAGING

Feb 14 - Feb 15, 2023

1 The Half-edge Data Structure

The half-edge data structure is a powerful representation used for polygon meshes. In addition to the familiar vertices, edges, and faces, this data structure includes the half-edge, an entity that serves as the ”glue” that connects the entities together. Using this data structure, we can easily navigate the mesh and perform various mesh operations, which include local operations like edge flip, edge split, edge collapse and mesh-scale operations like Loop subdivision, mesh simplification, and mesh regularization.

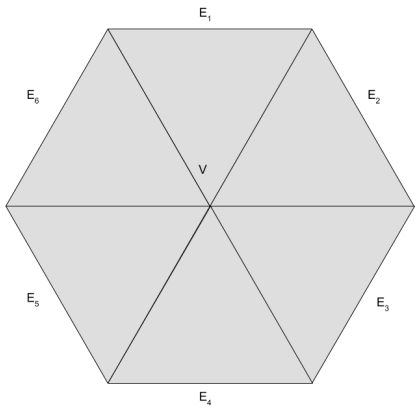


The mesh elements have the following relationships:

Mesh element	Pointers
Vertex	one halfedge
Edge	one halfedge
Face	one halfedge
Halfedge	twin, next, vertex, edge, face

As the table might suggest, the **halfedge** connects the other mesh elements together. From any vertex, edge, or face, we can get a reference to a halfedge; from that halfedge, we can then access other halfedges, vertices, edges, and faces. These relationships allow us to easily navigate the mesh.

1. Starting from a given vertex, traverse the mesh and return a `std::vector` containing all of the edges that are opposite that vertex. In the diagram below, the opposite edges are labelled. **Hint:** Start with the vertex's halfedge, then perform a do-while loop until we return to the original halfedge.




```
std::vector<EdgeIter> getOppositeEdges(VertexIter v)
```

Solution:

```
{
  std::vector<EdgeIter> edges;
  HalfedgeIter h = v->halfedge();
  HalfedgeIter start_h = h;
  do {
    edges.push_back(h->next()->edge());
    h = h->next()->next()->twin();
```

```
  } while (h != start_h);
  return edges;
}
```

2. Let's write a function that locally attenuates noise in a mesh through a diffusion process. To do so, we will approximate the Laplacian operator using the umbrella operator at a specific vertex. Then, we will apply the approximated Laplacian to a given vertex to locally smooth the mesh around that vertex. This void function **should directly modify** the vertex v's position.

   More specifically, given a vertex v at position x, access the vertex's neighboring vertices and compute the following value:

$$L(v) = \frac{1}{n} \sum_{v_j \in N(v)} x_j - x$$

   where $N(v)$ is the set of neighboring vertices of vertex v, x_j is the position of neighboring vertex v_j, and n is the number of neighboring vertices.

   Then, apply $L(v)$ to slightly move the vertex v from position x to a new position x' that makes the mesh slightly smoother around v:

$$x' = x + kL(v)$$

   where k is a weight on the approximated Laplacian effects.

   Note: When repeatedly applied throughout an entire mesh, this operation smoothes out high frequencies while maintaining the mesh's relative shape.

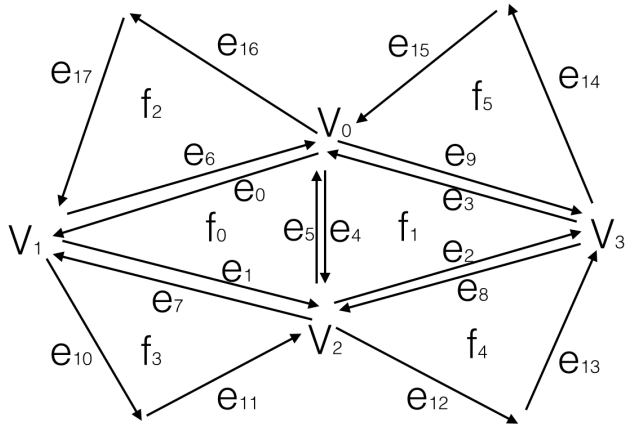
```
void diffuse(VertexIter v, float k)
```

Solution:

```
{
  Vector3D L(0, 0, 0);
  HalfedgeIter h = v->halfedge();
  HalfedgeIter start_h = h;
  int n = 0;
  do {
    VertexIter v_j = h->next()->vertex();
    Vector3D dir = v_j->position() - v->position();
    L = L + dir;
    n++;
    h = h->twin()->next();
  } while (h != start_h);
  v->position() = v->position() + k * L / n;
}
```

3. The figure below shows some local structure of a mesh. To contract the edge connecting vertices V_0 and V_2, we need to first update some fields of those elements shown in the figure, then delete vertex V_2, halfedges $e_1, e_2, e_4, e_5, e_7, e_8$, and faces f_0, f_1. Fill in the blanks for how the following fields should be updated in order to minimize changes in the whole operation.

$e_{11} \rightarrow next() =$ _____
$f_3 \rightarrow halfedge() =$ _____
$V_0 \rightarrow halfedge() =$ _____
$e_3 \rightarrow face() =$ _____
$e_{12} \rightarrow vertex() =$ _____



Solution: $e_{11} \rightarrow next() = e_0$
$f_3 \rightarrow halfedge() = e_0/e_{10}/e_{11}$
$V_0 \rightarrow halfedge() = e_0/e_9/e_{12}/e_{16}$
$e_3 \rightarrow face() = f_4$
$e_{12} \rightarrow vertex() = V_0$

2 Ray-Surface Intersection

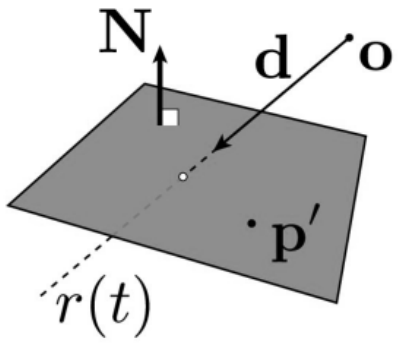
Given a mesh representation of an object, we would like to render it onto a display. To do so, we need to know which parts of the object are visible, where to put shadows, how to apply the scene's lighting, and more. The simplest (but rather slow) idea to handle these problems is to take a ray and intersect it with each triangle in the mesh. There could be 0, 1, or multiple intersections.

Recall that a ray is defined by its origin $\mathbf{o}$ and a direction vector $\mathbf{d}$ and varies with "time" t for $0 <= t < \infty$.

$$\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$$

1. (a) As a warm-up, let's rederive the equation for a ray intersection with an arbitrary plane. Recall that a plane can be defined as any point $\mathbf{p}$ that satisfies the following equation:

$$\mathbf{p} : (\mathbf{p} - \mathbf{p}') \cdot \mathbf{N} = 0$$



Set $\mathbf{p}$ equal to $\mathbf{r}(t)$ and solve for t.

Solution: $(\mathbf{p} - \mathbf{p}') \cdot \mathbf{N} = 0$
$(\mathbf{o} + t\mathbf{d} - \mathbf{p}') \cdot \mathbf{N} = 0$
$(\mathbf{o} - \mathbf{p}') \cdot \mathbf{N} + t\mathbf{d} \cdot \mathbf{N} = 0$
$t\mathbf{d} \cdot \mathbf{N} = (\mathbf{p}' - \mathbf{o}) \cdot \mathbf{N}$
$t = \frac{(\mathbf{p}' - \mathbf{o}) \cdot \mathbf{N}}{\mathbf{d} \cdot \mathbf{N}}$

(b) What does it mean if we get a value of $t < 0$?

> **Solution:** This means that the intersection point with the plane is behind the ray's origin. Since the ray only moves forward in the direction of $\mathbf{d}$ for positive values of t, $t < 0$ indicates that this value of t is not a valid intersection with the ray.
>
> Note that this is true for any ray-surface intersection problem, not just with planes.

(c) What does it mean if $\mathbf{d} \cdot \mathbf{N} = 0$?

> **Solution:** When $\mathbf{d} \cdot \mathbf{N} = 0$, then the ray's direction vector is perpendicular to the plane's normal vector $\mathbf{N}$. Therefore, the ray is parallel to the plane and either intersects for all values of t or has 0 intersections.
>
> To figure out which one, plug in $\mathbf{o}$ into the plane equation and see if it satisfies the equality.

2. Given the following implicit representation of an ellipsoid and the following ray, compute the position(s) at which the ray intersects the ellipsoid.

$$f(x, y, z) = \frac{x^2}{4} + y^2 + \frac{z^2}{4} - 9$$
$$\mathbf{r}(t) = < 0, 1, 0 > + t < 1, 0, 2 >$$

Start by substituting the ray $\mathbf{r}(t)$ into the function $f(x, y, z)$ to get an expression $f(\mathbf{o} + t\mathbf{d})$. Set $f(\mathbf{o} + t\mathbf{d}) = 0$ and solve for t. Then plug t back into the ray equation to find the point(s) of intersection.

> **Solution:** To get $f(\mathbf{o} + t\mathbf{d}) = 0$, we substitute $x = 0 + t \cdot 1$, $y = 1 + t \cdot 0$, and $z = 0 + t \cdot 2$. This gives us:
>
> $$\frac{t^2}{4} + 1^2 + \frac{(2t)^2}{4} - 9 = 0$$
> $$\frac{t^2}{4} + \frac{4t^2}{4} = 8$$
> $$\frac{5t^2}{4} = 8$$
> $$t^2 = \frac{32}{5}$$

$$t = -4\sqrt{\frac{2}{5}}, 4\sqrt{\frac{2}{5}}$$

Only the positive t result is a valid intersection with this ray, as the negative t result is behind the ray's origin. Plugging in $t = 4\sqrt{\frac{2}{5}}$, we see that the ray intersects the ellipsoid at $< 0, 1, 0 > +4\sqrt{\frac{2}{5}} < 1, 0, 2 >=< 4\sqrt{\frac{2}{5}}, 1, 8\sqrt{\frac{2}{5}} >$.