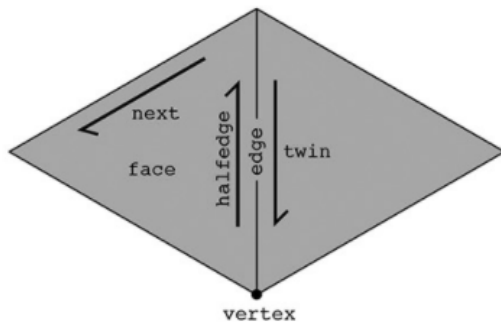


HALF-EDGES AND RAY TRACING 5

CS 184: FOUNDATIONS OF COMPUTER GRAPHICS

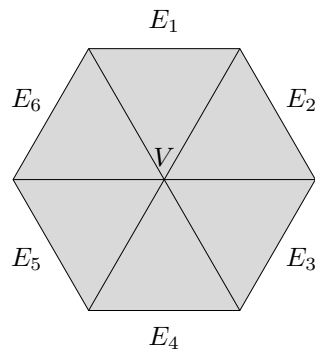
1 The Half-edge Data Structure

The half-edge data structure is a powerful representation that enables us to navigate a polygon mesh and perform various operations. The mesh elements have the following relationships:



Data structure	Attributes
Vertex	one halfedge
Edge	one halfedge
Face	one halfedge
Halfedge	twin, next, vertex, edge, face

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, E_1, \dots, E_6 are the edges opposite V . **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) {
```

2. Let's write a function that locally attenuates noise in a mesh. Given a vertex v at position x , let

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

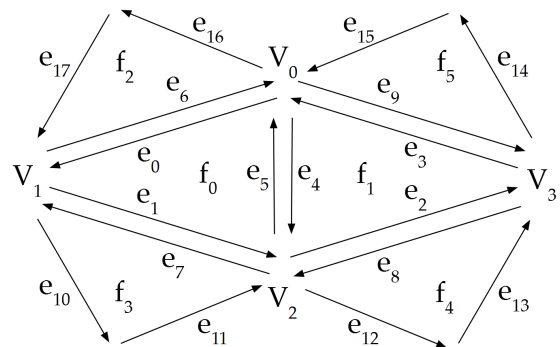
where $N(v)$ is the set of neighboring vertices of vertex v , x_j is the position of neighboring vertex v_j , and n is the number of neighboring vertices.

Apply $L(v)$ to slightly move v from position x to a new position x' , making the mesh slightly smoother around v : $x' = x + kL(v)$, where k is a weight. We call this a diffuse operation on v 's position.

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

3. Fill in the blanks so that the resulting procedure collapses the edge connecting vertices V_0 and V_2 .

- (i) $e_{11} \rightarrow \text{next}() =$ _____
- (ii) $f_3 \rightarrow \text{halfedge}() =$ _____
- (iii) $V_0 \rightarrow \text{halfedge}() =$ _____
- (iv) $e_3 \rightarrow \text{face}() =$ _____
- (v) $e_{12} \rightarrow \text{vertex}() =$ _____
- (vi) $V_1 \rightarrow \text{halfedge}() =$ _____
- (vii) _____
- (viii) _____
- (ix) _____
- (x) _____
- (xi) _____
- (xii) _____



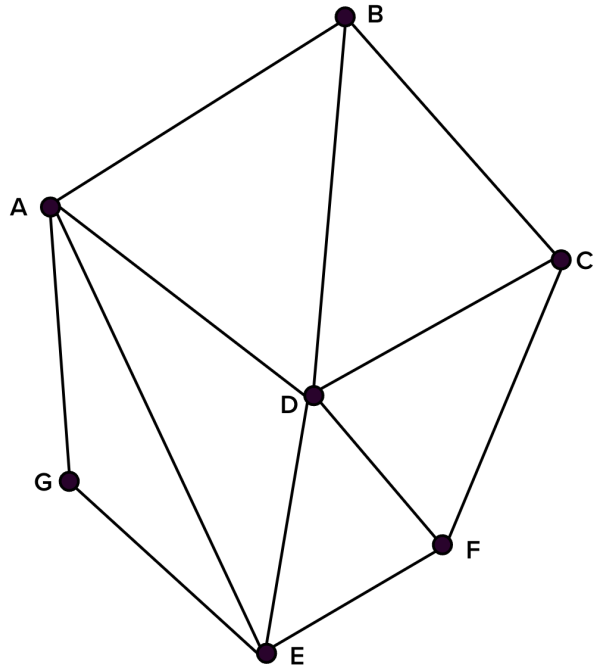
- (xiii) Delete vertex V_2
- (xiv) Delete half-edges $e_1, e_2, e_4, e_5, e_7, e_8$
- (xv) Delete faces f_0, f_1

2 Loop-de-Loop

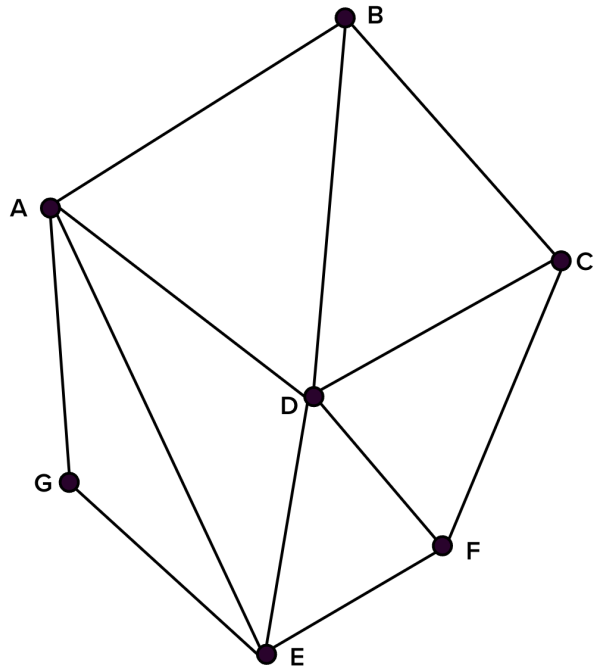
Loop subdivision is a method for upsampling triangular meshes. A single Loop subdivision involves:

1. 4-1 subdividing each triangle (face splitting).
2. Updating each vertex position (smoothing).

1. Draw the midpoint of each edge. Connect each edge's midpoint to the vertex (or vertices) opposite to that edge, effectively *splitting* on each edge.



2. For each new edge that you created in Part 1, if the edge connects a *new* vertex to an *old* vertex, perform an edge *flip*. Draw the updated mesh.



3. Now, let's smooth this new mesh using the Loop subdivision update rules.

Suppose $A = (2, 4, 0)$, $B = (5, 7, 0)$, $C = (7, 3, 0)$, and $D = (4, 2, 0)$. For the *new* vertex that is initialized as the midpoint of edge BD , calculate its updated position.

4. In general, when is the position of a *new* vertex unchanged by the Loop subdivision update rule? In other words, when is the updated position equal to the midpoint of the original edge?

5. Vertex D is an *old* vertex that has neighbors A, B, C, E , and F . $E = (5, 0, 0)$ and $F = (6, 1, 0)$. Calculate the updated position of vertex D .

6. In general, in what direction does the Loop subdivision update rule move an *old* vertex?

3 Ray-Surface Intersection

When rendering a 3D object, we must determine which parts of it are visible, where shadows fall, and how lighting interacts within the scene. A straightforward yet computationally expensive approach is to cast rays from the camera through each pixel, finding where those rays intersect the object's mesh. In general, a ray can intersect the mesh zero times, once, multiple times, or—if it lies exactly in a triangle's plane—infinately many times.

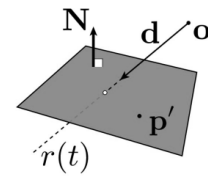
Recall that a ray is defined by its origin \mathbf{o} and its direction vector \mathbf{d} , and is parameterized by $t \geq 0$:

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

1. As a warm-up, let's re-derive the equation for a ray intersecting an arbitrary plane. Recall that a plane can be defined as the set of all points \mathbf{p} satisfying

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

where \mathbf{p}' is any point on the plane and \mathbf{N} is the plane's normal vector. Set \mathbf{p} equal to $\mathbf{r}(t)$ and solve for t .



2. What does it mean if we get a value of $t < 0$?

3. What does it mean if $\mathbf{d} \cdot \mathbf{N} = 0$?

4. Given the following implicit representation of an ellipsoid and the definition of a ray, compute where (and at what parameter value(s) of t) the ray intersects the ellipsoid:

$$f(x, y, z) = \frac{(x - 2)^2}{4} + (y - 2)^2 + \frac{z^2}{4} - 1$$

$$\mathbf{r}(t) = (0, 0, 0) + t(1, 1, 0)$$

Start by substituting the ray $\mathbf{r}(t)$ into the function $f(x, y, z)$ to obtain $f(\mathbf{o} + t \mathbf{d})$.