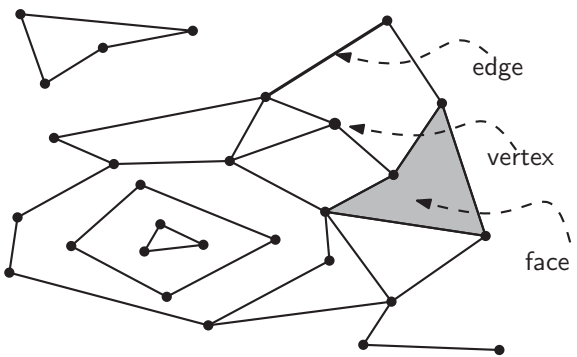
Subdivisions

A planar subdivision is a structure induced by a set of line segments in the plane that can only intersect at common endpoints. It consists of vertices, edges, and faces



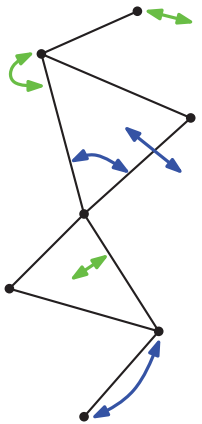
Subdivisions

Vertices are the endpoints of the line segments

Edges are the interiors of the line segments

Faces are the interiors of connected two-dimensional regions that do not contain any point of any line segment

Objects of *the same* dimensionality are adjacent or not; objects of *different* dimensionality are incident or not



adjacent

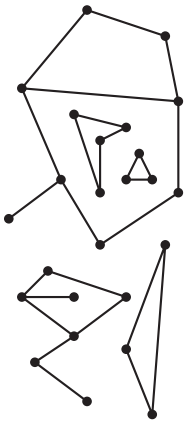
incident

Subdivisions

Exactly one face is unbounded, the outer face

Every other face is bounded and has an outer boundary consisting of vertices and edges

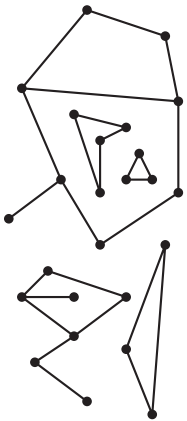
Any face has zero or more inner boundaries



Subdivisions

Vertices, edges, and faces form a partition of the plane

If a planar subdivision is induced by n line segments, it has exactly n edges, and at most $2n$ vertices

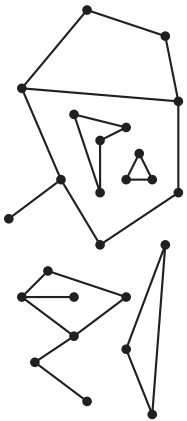


Subdivisions

And how many faces?

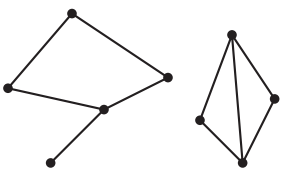
Observe: Every face is bounded by at least 3 edges, and every edge bounds at most 2 faces $\Rightarrow$
$F \leq 3 \cdot (E/2) = 3n/2 = O(n)$

Exception: there are less than 3 edges, and the only face is the outer face



Euler's formula

Euler's formula for planar graphs: If S is a planar subdivision with V vertices, E edges, and F faces, then $V - E + F \geq 2$, with equality iff the vertices and edges of S form a connected set



$V = 9,\ E = 10,\ F = 4$

$V - E + F = 3$



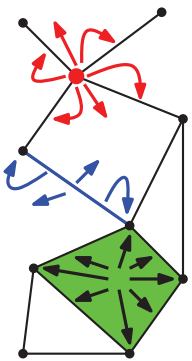
$V = 11,\ E = 12,\ F = 4$

$V - E + F = 2$

Representing subdivisions

A subdivision representation has a vertex-object class, an edge-object class, and a face-object class

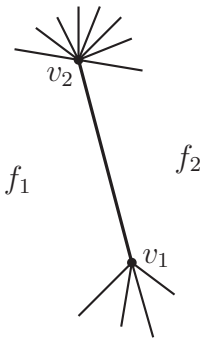
It is a pointer structure where objects can reach incident (or adjacent) objects easily



Representing subdivisions

Use the **edge** as the central object

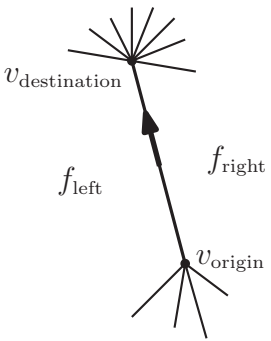
For any edge, exactly two vertices are incident, exactly two faces are incident, and zero or more other edges are adjacent



Representing subdivisions

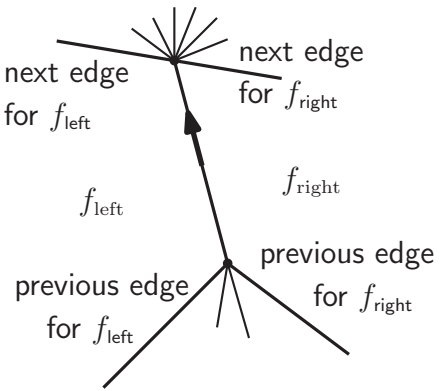
Use the **edge** as the central object, and give it a direction

Now we can speak of Origin, Destination, Left Face, and Right Face



Representing subdivisions

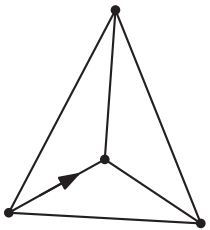
Four edges are of special interest

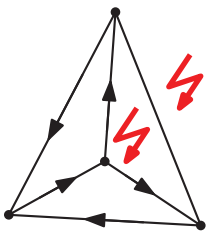


Representing subdivisions

It would be nice if we could traverse a boundary cycle by continuously following the next edge for f_{left} or f_{right}

... but, no consistent edge orientation needs to exist



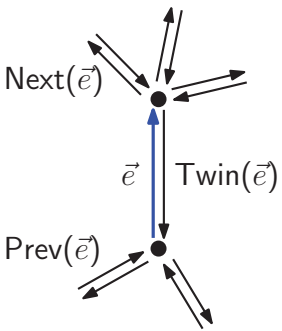


Representing subdivisions

We apply a trick/hack/impossibility: split every edge *length-wise(!)* into two half-edges

Every half-edge:

- has exactly one half-edge as its Twin
- is directed opposite to its Twin
- is incident to only one face (left)



The doubly-connected edge list

The doubly-connected edge list is a subdivision representation structure with an object for every vertex, every half-edge, and every face

A vertex object stores:

- Coordinates
- **IncidentEdge** (some half-edge leaving it)

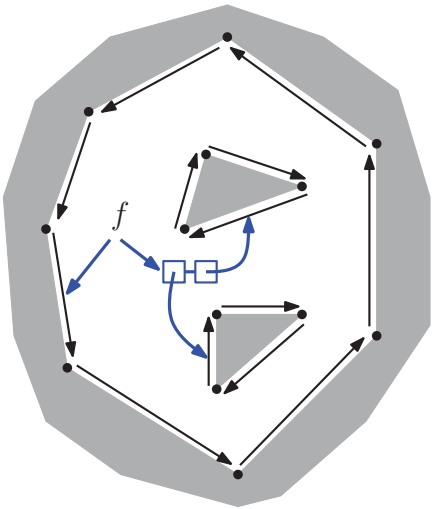
A half-edge object stores:

- **Origin** (vertex)
- **Twin** (half-edge)
- **IncidentFace** (face)
- **Next** (half-edge in cycle of the incident face)
- **Prev** (half-edge in cycle of the incident face)

The doubly-connected edge list

A face object stores:

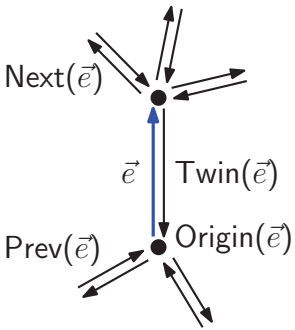
- **OuterComponent** (half-edge of outer cycle)
- **InnerComponents** (list of half-edges for the inner cycles bounding the face)



The doubly-connected edge list

Question: A half-edge $\vec{e}$ can directly access its **Origin**, and get the coordinates of one endpoint. How can it get the coordinates of its other endpoint?

Question: For a vertex v, how do we find all adjacent vertices?





The doubly-connected edge list

A vertex object stores:

- Coordinates
- **IncidentEdge**
- *Any attributes, mark bits*

A face object stores:

- **OuterComponent** (half-edge of outer cycle)
- **InnerComponents** (half-edges for the inner cycles)
- *Any attributes, mark bits*

A half-edge object stores:

- **Origin** (vertex)
- **Twin** (half-edge)
- **IncidentFace** (face)
- **Next** (half-edge in cycle of the incident face)
- **Prev** (half-edge in cycle of the incident face)
- *Any attributes, mark bits*