Mesh Basics

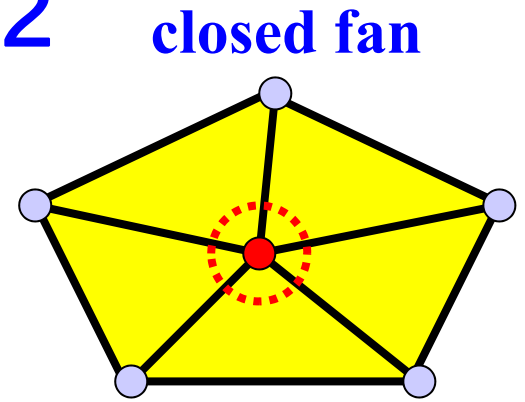
Spring 2010

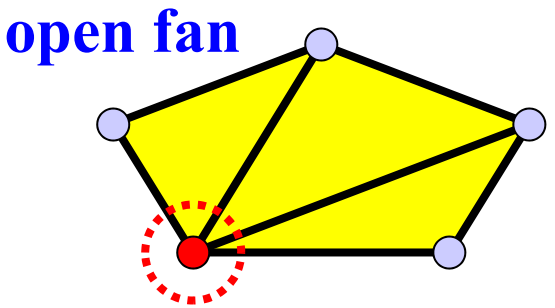
Definitions: 1/2

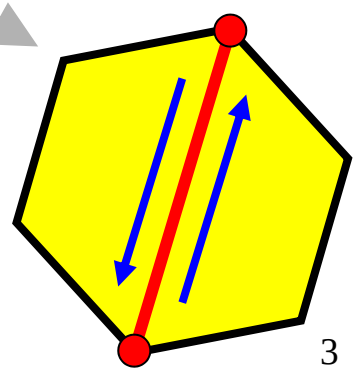
- **A *polygonal mesh* consists of three kinds of *mesh elements*: vertices, edges, and faces.**
- **The information describing the mesh elements are *mesh connectivity* and *mesh geometry*.**
- **The *mesh connectivity*, or topology, describes the incidence relations among mesh elements (*e.g.*, adjacent vertices and edges of a face, etc).**
- **The *mesh geometry* specifies the position and other geometric characteristics of each vertex.**

Definitions: 2/2

- A mesh is a *manifold* if (1) each edge is incident to only one or two faces and (2) the faces incident to a vertex form a *closed* or an *open* fan.
- The *orientation* of a face is a cyclic order of the incident vertices.
- The orientation of two adjacent faces is *compatible*, if the two vertices of the common edge are in opposite order.
- A manifold mesh is *orientable* if any two adjacent faces have compatible orientation.

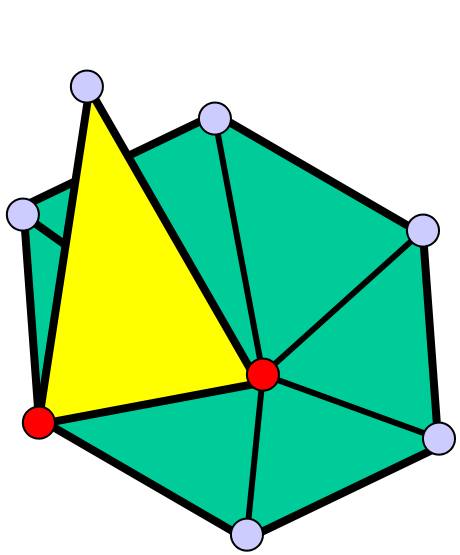


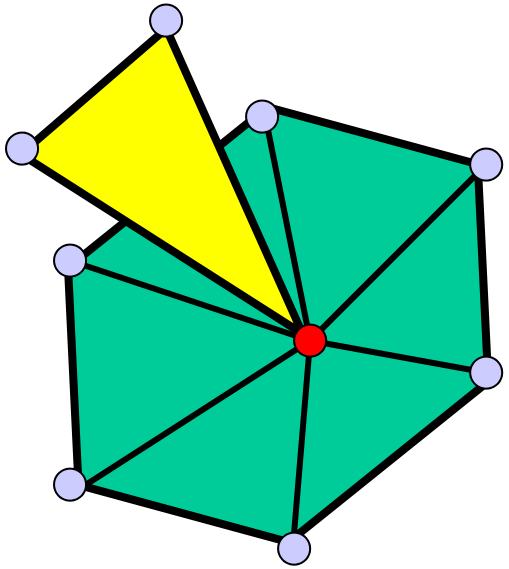


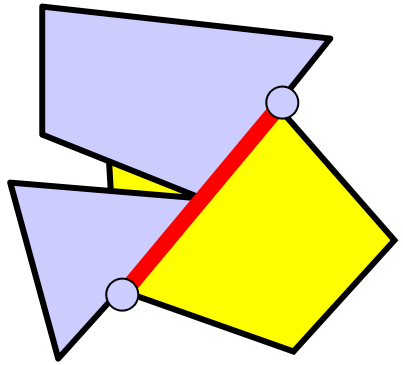


Non-Manifold Meshes

- **Manifold Conditions:** (1) **Each edge is incident to only one or two faces and** (2) **the faces incident to a vertex form a closed or an open fan.**
- **The following examples are not manifold meshes!**

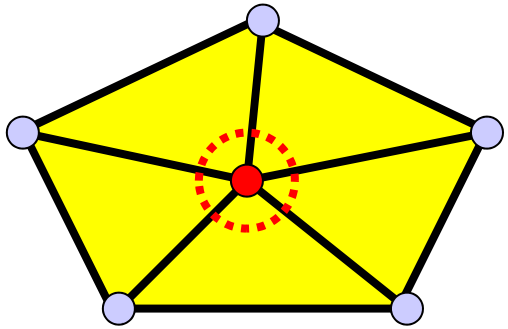


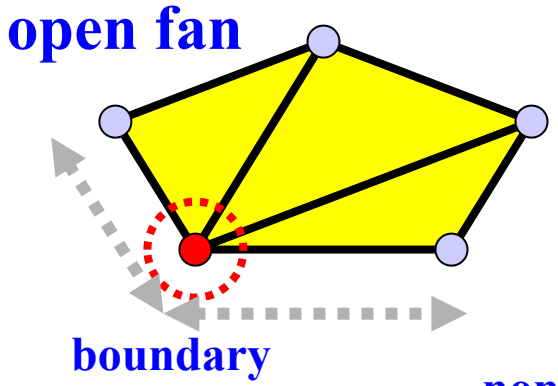


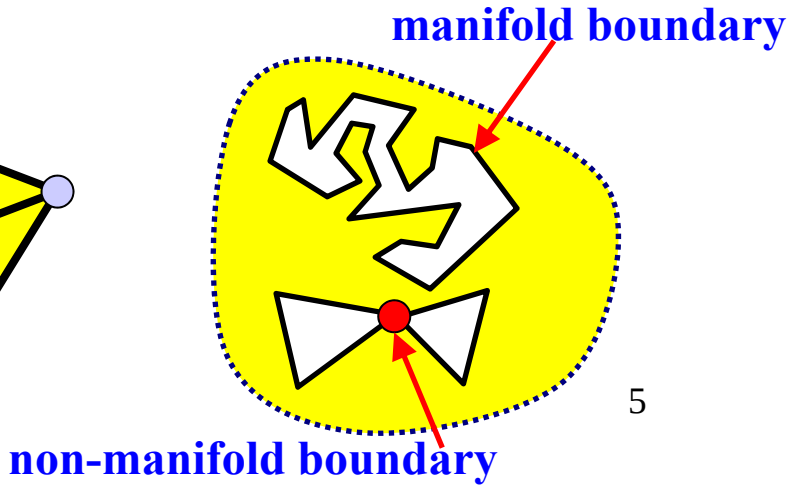


Manifolds with/without Boundary

- If every vertex has a closed fan, the given manifold has no boundary. Edges only incident to one face form the boundary of the manifold.
- Boundary is a union of simple polygons.
- We only consider orientable manifolds without boundary in this course.

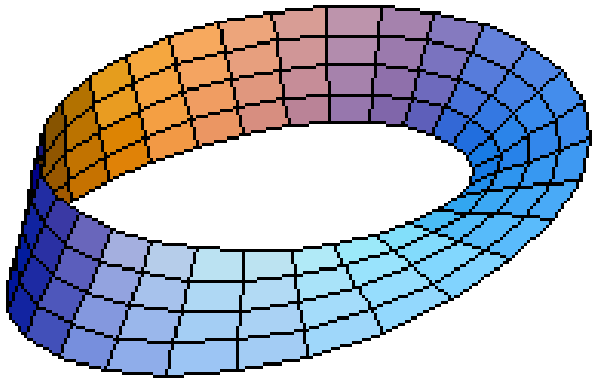


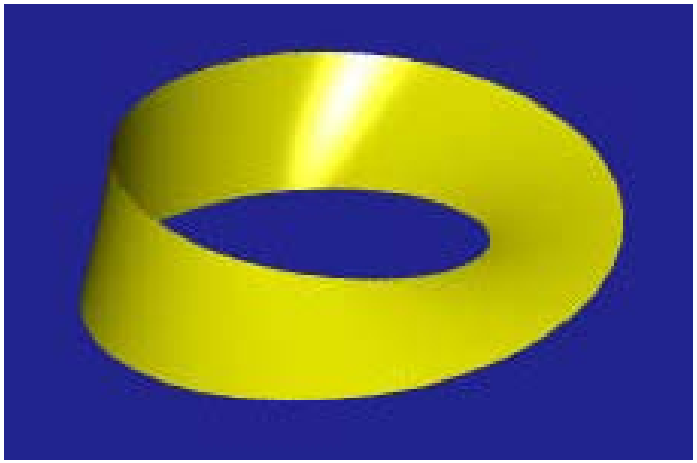




Non-Orientable Manifolds: 1/2

- **Not all manifolds are orientable. The most well-known ones are Möbius band and Klein bottle.**
- **The Möbius band is shown below, and is an *one-sided* manifold with boundary (*i.e.*, a circle).**



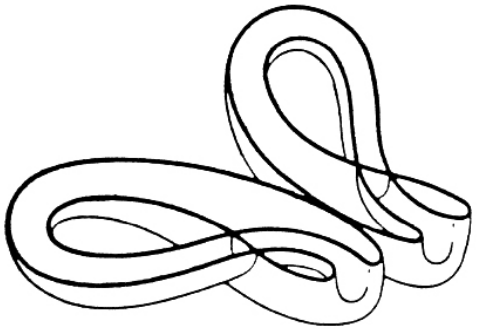


http://www.jcu.edu/math/vignettes/Mobius.htm

Non-Orientable Manifolds: 2/2

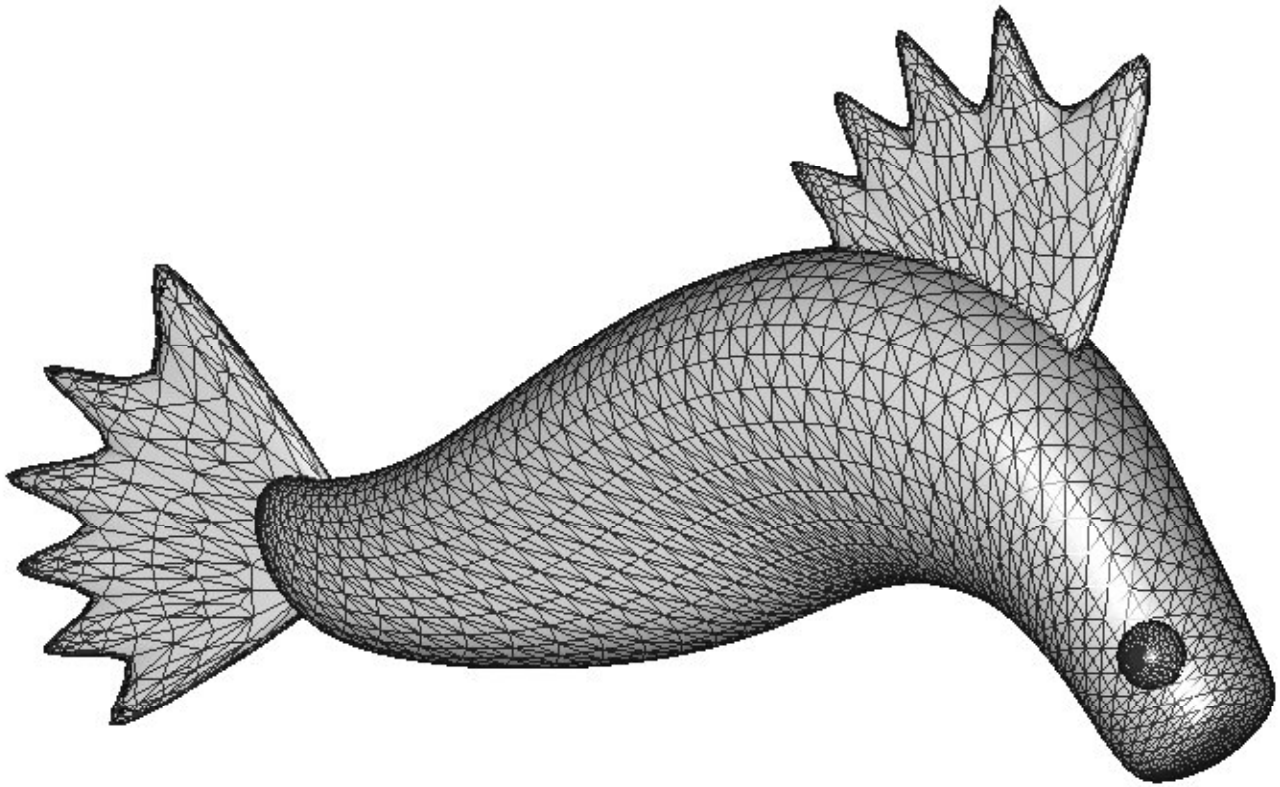
- The Klein bottle is a manifold without boundary.
- Slicing a Klein bottle properly yields two Möbius bands.

A Klein bottle sliced to show its interior.
However, Klein bottles have no interior.



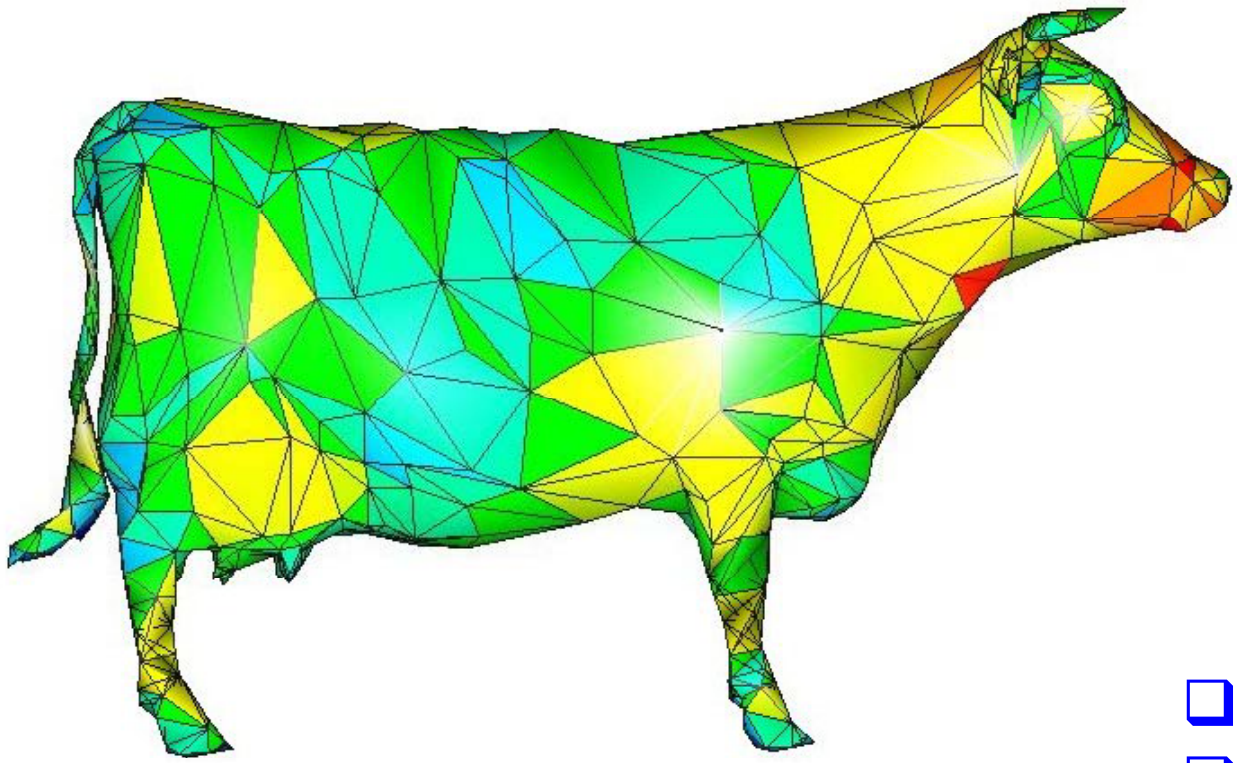
Maurice Fréchet and Ky Fan,
Invitation to Combinatorial Topology

Mesh Examples: 1/2



- **Vertices:** 4,634
- **Edges:** 13,872
- **Faces:** 9,248

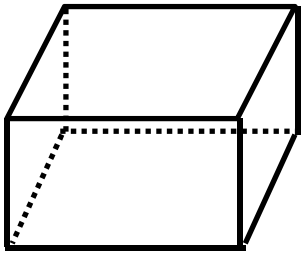
Mesh Examples: 2/2



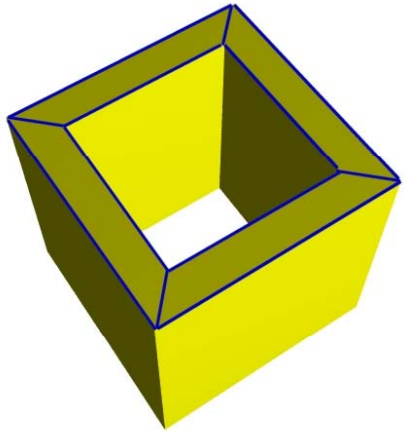
- **Vertices: 703**
- **Edges: 2106**
- **Faces: 1401**

Euler-Poincaré Characteristic: 1/5

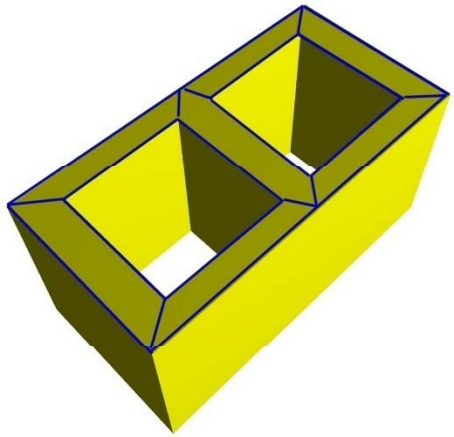
- **Given a 2-manifold mesh M without boundary, the Euler-Poincaré characteristic of M is $\chi(M) = V-E+F$, where V, E and F are the number of vertices, number of edges, and number of faces.**



V=8, E=12, F=6
$\chi(M) = V-E+F=2$



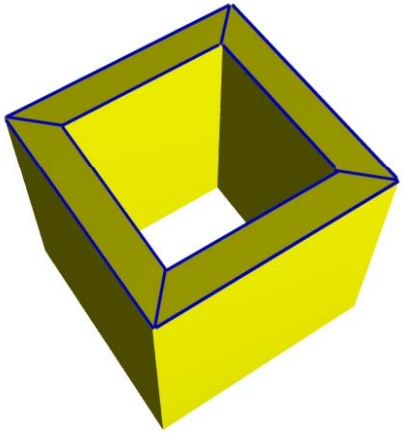
V=16, E=32, F=16
$\chi(M) = V-E+F=0$



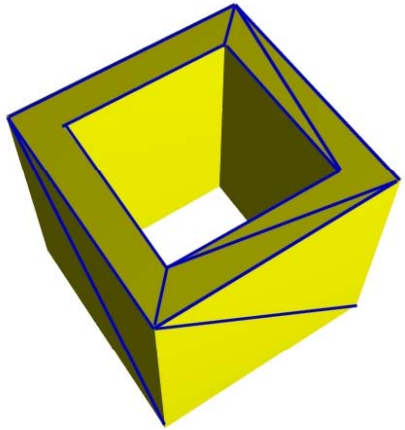
V=28, E=56, F=26
$\chi(M) = V-E+F=-2$

Euler-Poincaré Characteristic: 2/5

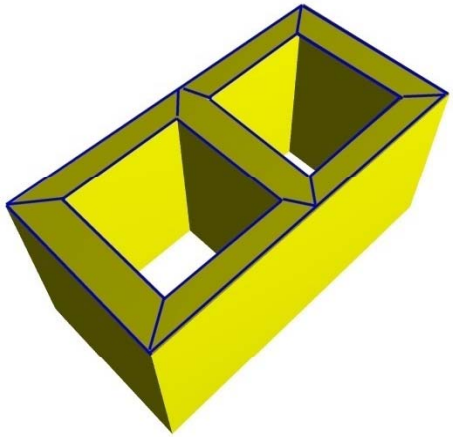
- **Euler-Poincaré characteristic $\chi(M) = V-E+F$ is independent of tessellation.**



V=16, E=32, F=16
$\chi(M) = V-E+F=0$

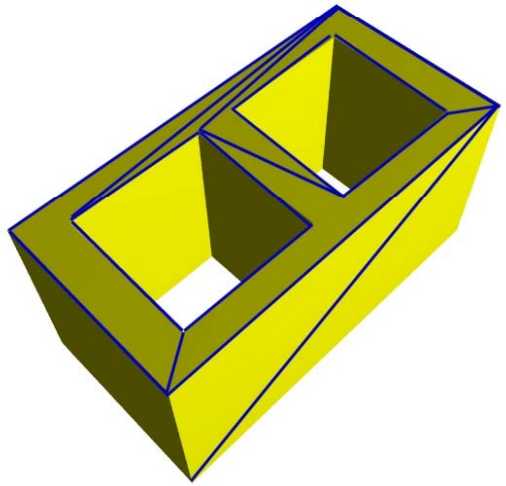


V=16, E=36, F=20
$\chi(M) = V-E+F=0$



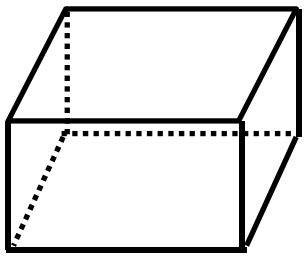
V=28, E=56, F=26
$\chi(M) = V-E+F=-2$

V=24, E=48, F=22
$\chi(M) = V-E+F=-2$

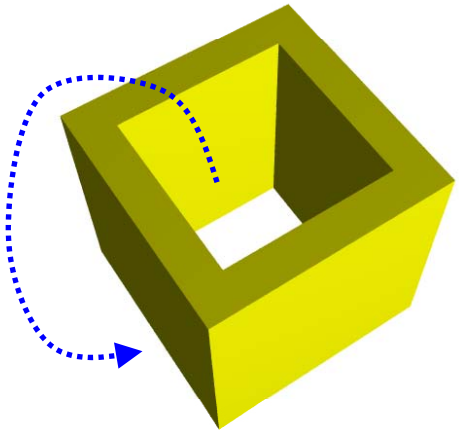


Euler-Poincaré Characteristic: 3/5

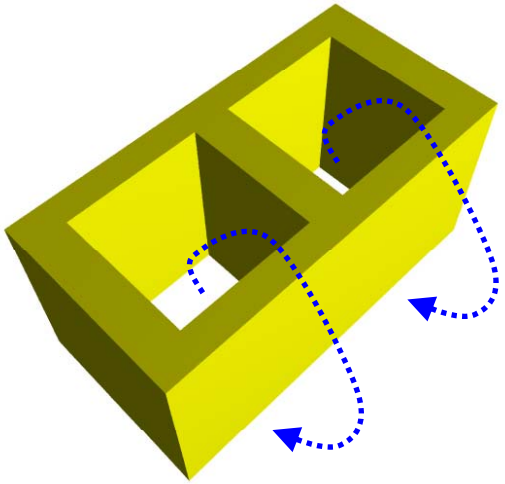
- **An orientable 2-manifold mesh M with *g* "handles" (*i.e.*, genus) has Euler-Poincaré characteristic $\chi(M) = V-E+F = 2(1-g)$.**
- **Spheres, boxes, tetrahedrons and convex surfaces have $g = 0$; but, tori have $g = 1$.**



$g=0 \Rightarrow \chi(M) = 2(1-0)=2$



$g=1 \Rightarrow \chi(M) = 2(1-1)=0$



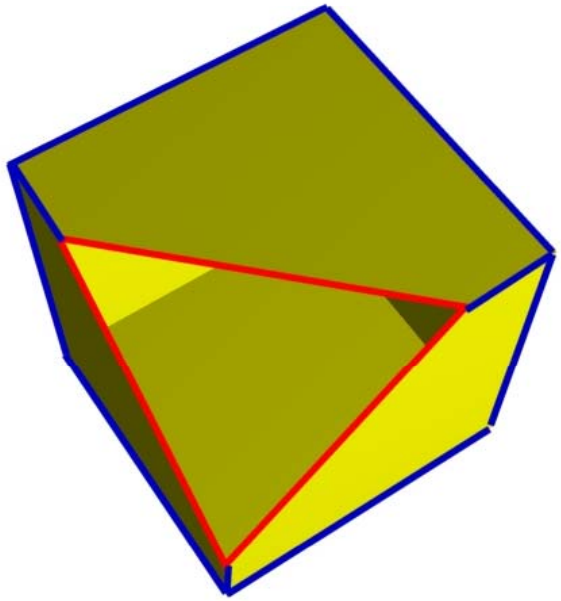
$g=2 \Rightarrow \chi(M) = 2(1-2)=-2$

Euler-Poincaré Characteristic: 4/5

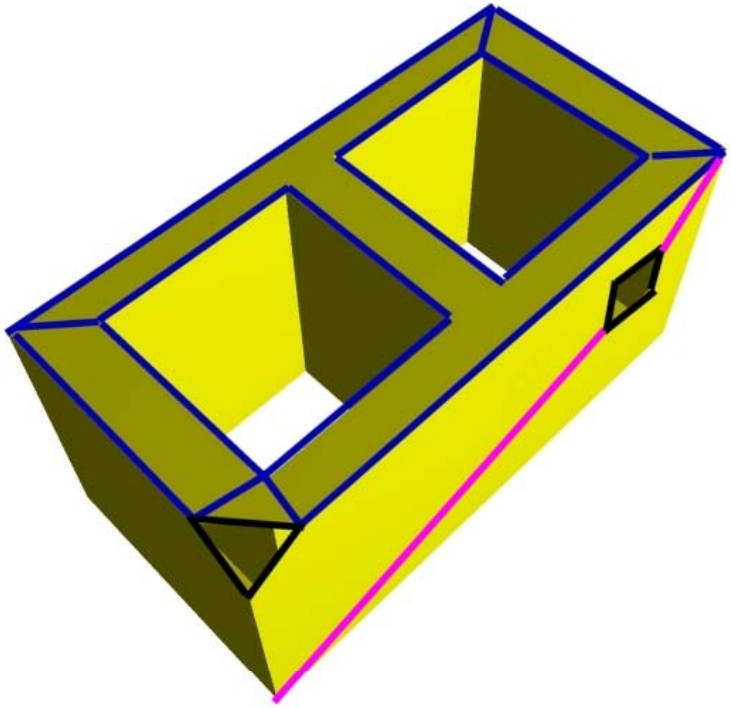
- **The boundary of an orientable 2-manifold is the union of a set of simple polygons.**
- **Since each polygon bounds a face, these "boundary faces" may be added back to form a manifold without boundary so that Euler-Poincaré characteristic can be applied.**
- **The Euler-Poincaré characteristic of an orientable 2-manifold with boundary is $\chi(M) = 2(1-g)-\partial$, where ∂ is the number "boundary polygons".**

Euler-Poincaré Characteristic: 5/5

- **Two Examples:**



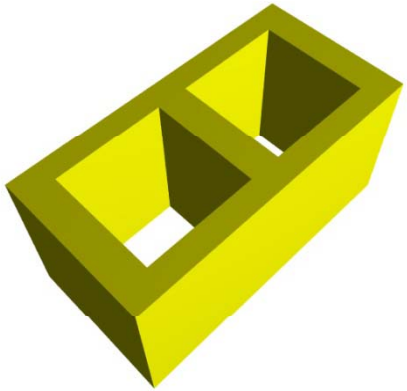
V = 10, E = 15, F = 6
g = 0, ∂ = 1
χ(M) = V-E+F = 1
χ(M) = 2(1-*g*)-∂ = 1



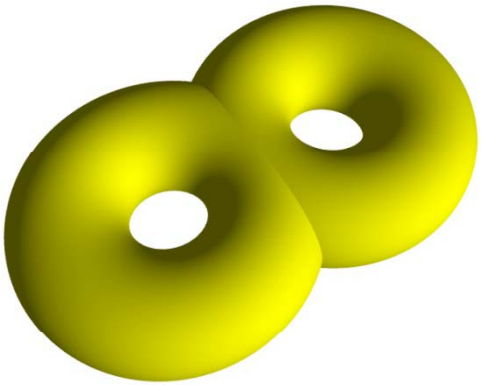
V = 30, E = 54, F = 20
g = 2, ∂ = 2
χ(M) = V-E+F = -4
χ(M) = 2(1-*g*)-∂ = -4

Homeomorphisms: 1/3

- **Two 2-manifold meshes A and B are homeomorphic if their surfaces can be transformed to the other by twisting, squeezing, and stretching without cutting and gluing.**
- **Thus, boxes, spheres and ellipsoids are homeomorphic to each other.**



is homeomorphic to

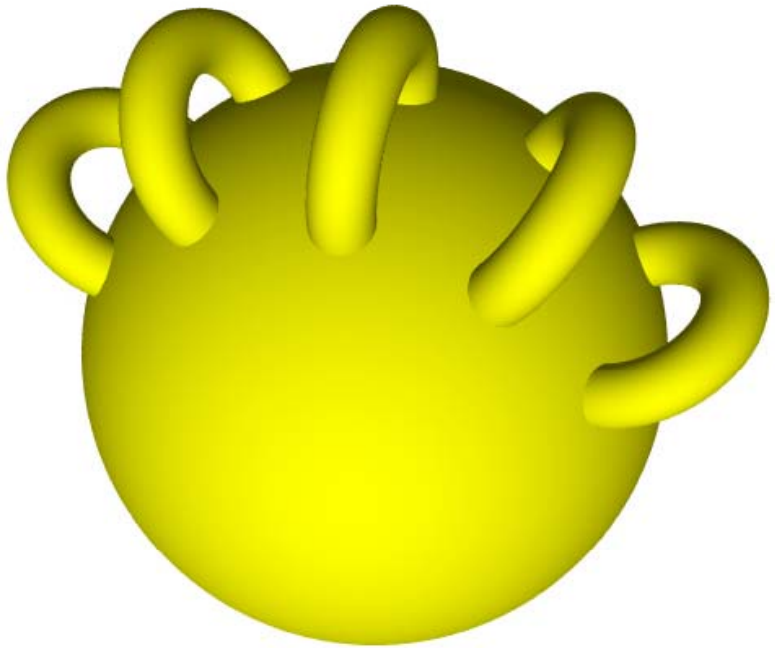


Homeomorphisms: 2/3

- **Two orientable 2-manifold meshes without boundary are *homeomorphic* if and only if they have the same Euler-Poincaré characteristic.**
- **Thus, a *m*-handle (*i.e.*, genus *m*) orientable mesh is homeomorphic to a *n*-handle (*i.e.*, genus *n*) orientable mesh if and only if *m* = *n*.**
- **Two orientable 2-manifold meshes with the same number of boundary polygons are *homeomorphic* if and only if they have the same Euler-Poincaré characteristic.**

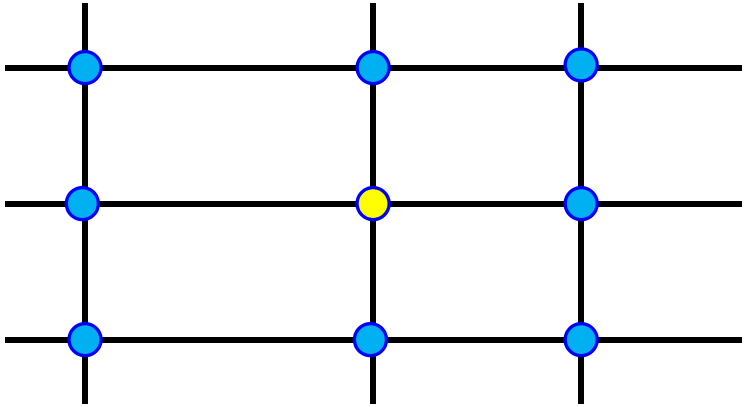
Homeomorphisms: 3/3

- **Hence, any orientable 2-manifold mesh without boundary is *homeomorphic* to a sphere with m handles (*i.e.*, genus m), where $m \geq 0$.**



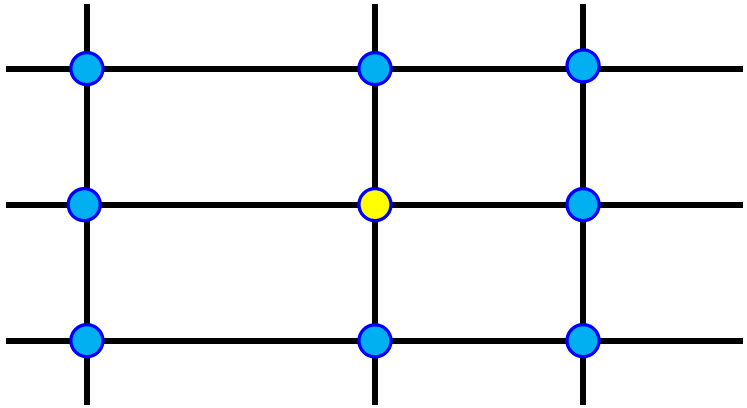
Applications: 1/3

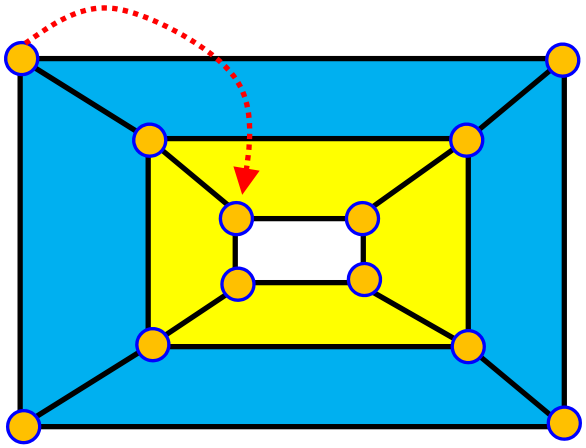
- **A mesh is *regular* if all faces have the same number edges, and all vertices are incident to the same number of edges (*i.e.*, valence).**
- **Each face of a *regular quad mesh* is a quadrilateral (*i.e.*, four-sided) and each vertex is incident to four edges (*i.e.*, valence = 4).**



Applications: 2/3

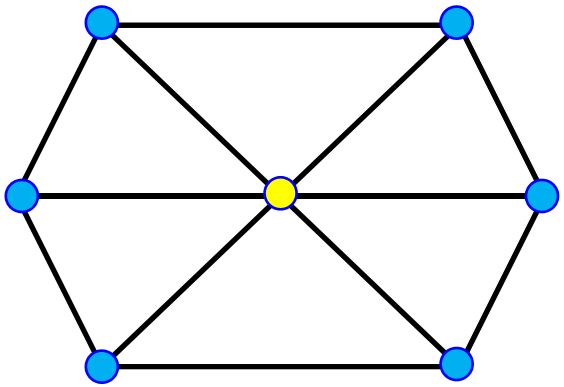
- Only a torus can be a regular quad mesh!
- Since each vertex has 4 edges and each edge is counted twice, we have $4V = 2E$ (*i.e.*, $V=E/2$).
- Since each face has 4 edges and each edge is counted twice, we have $4F = 2E$ (*i.e.*, $F = E/2$).
- Thus, $\chi(M) = V-E+F = 0$ means a torus!

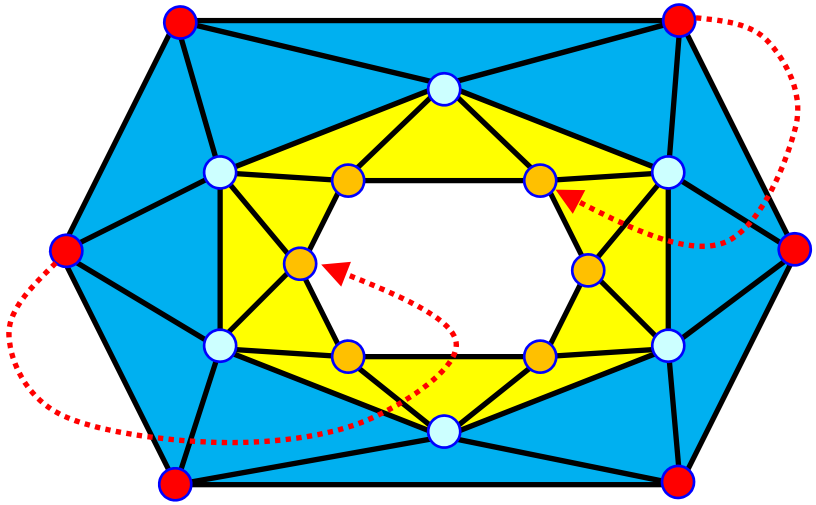




Applications: 3/3

- Only tori can be regular triangle mesh of valence 6!
- Since each vertex has 6 edges and each is counted twice, we have $6V = 2E$ (*i.e.*, $V=E/3$).
- Since each face has 3 edges and each edge is counted twice, we have $3F = 2E$ (*i.e.*, $F = 2E/3$).
- Thus, $\chi(M) = V-E+F = 0$ means a torus!



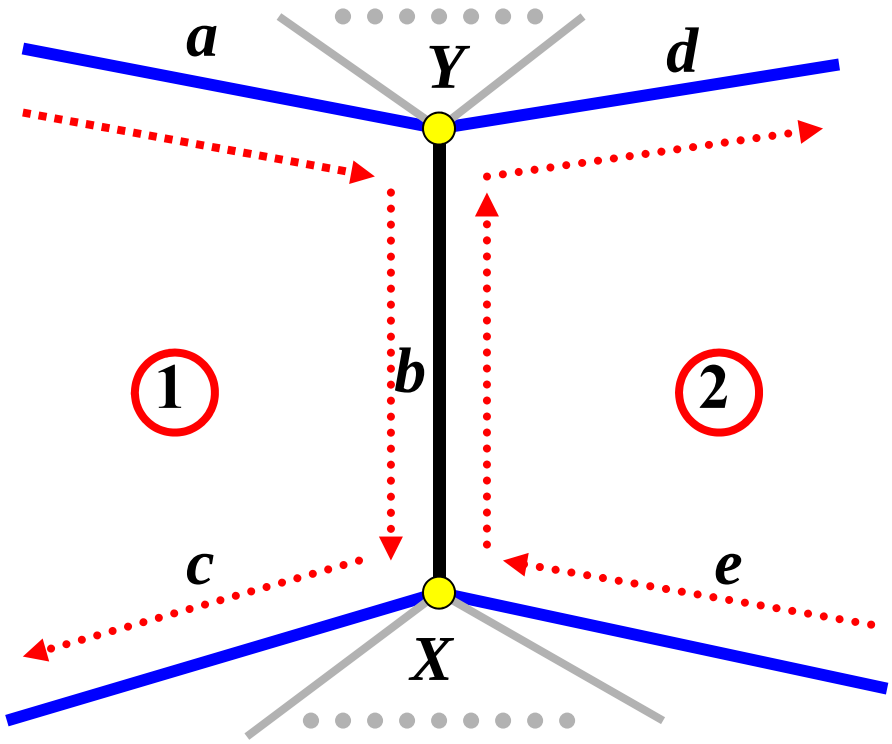


Data Structures for Meshes

- **Since meshes are usually large and complex and since many operations are performed on meshes, compact data structures that support efficient algorithms are needed.**
- **Depending on the applications in hand, one may use vertex- (or point-) based, edge-based, face-based, or other data structures.**
- **One of the earliest edge-based data structure is the *winged-edge* data structure. Its new variant is the *half-edge* data structure.**

What Is a Winged-Edge? 1/7

- **If all faces are oriented clock-wise, each edge has *eight* pieces of incident information.**

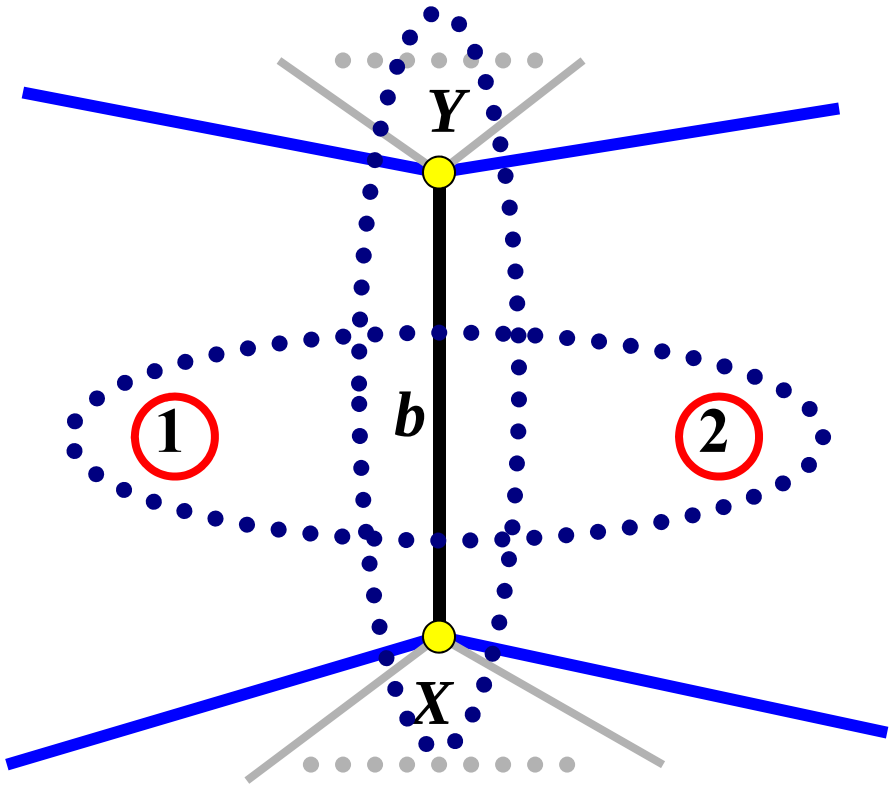


- **Given edge: *b*=*XY***
- **Incident faces: 1 and 2**
- **Pred. & succ. edges of 1**
- **Pred. & succ. edges of 2**

- **The *wings* of edge *b*=*XY* are faces 1 and 2.**
- **Edge *b* is a *winged-edge***

What Is a Winged-Edge? 2/7

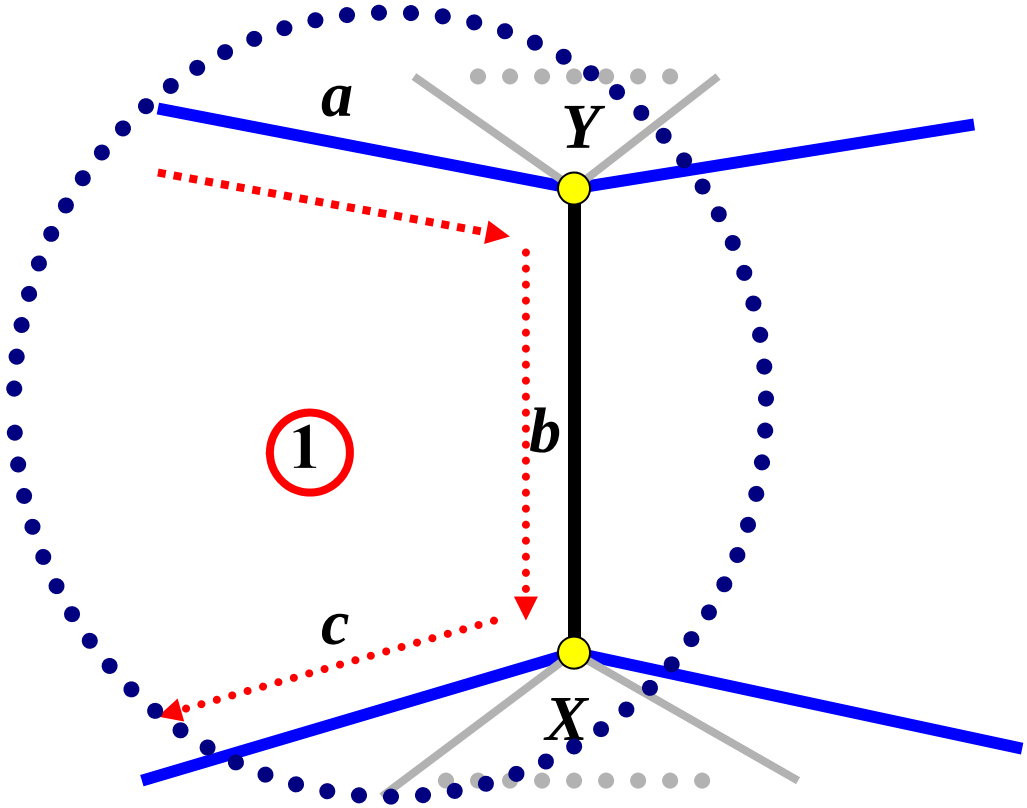
- **If all faces are oriented clock-wise, each edge has *eight* pieces of incident information.**



- **The first *four* pieces:**
  - **The *two* vertices of *b*: *X* and *Y***
  - **The *two* incident faces: 1 and 2**

What Is a Winged-Edge? 3/7

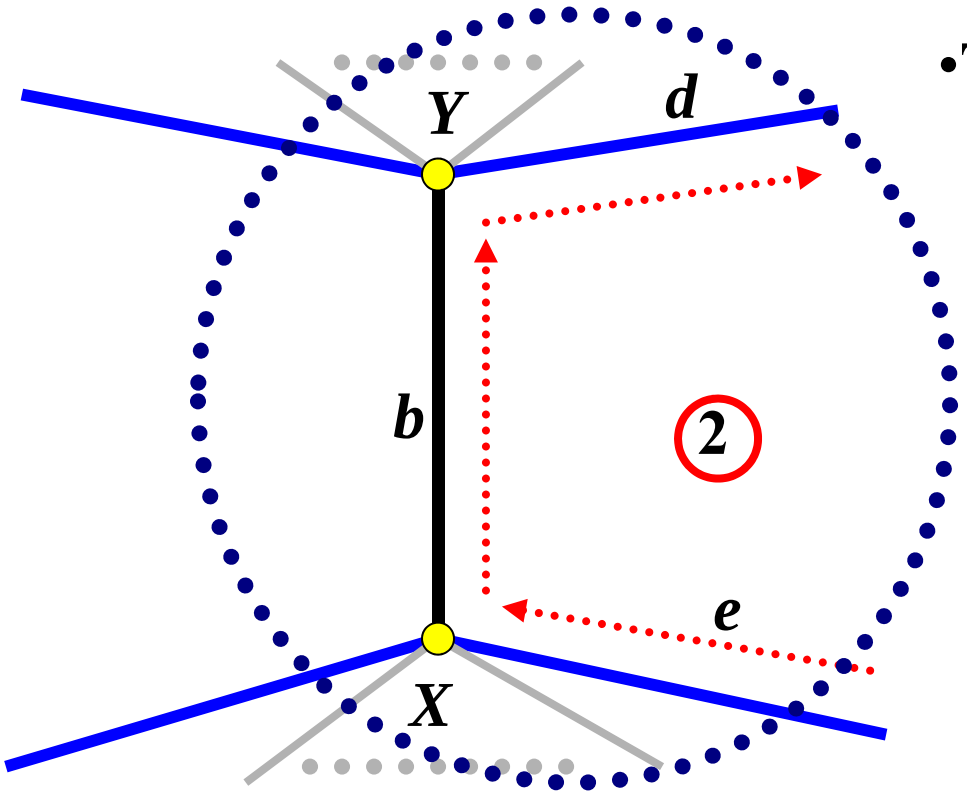
- If all faces are oriented clock-wise, each edge has *eight* pieces of incident information.



- The pred. and succ. edges of *b* with respect to face 1: *a* and *c*

What Is a Winged-Edge? 4/7

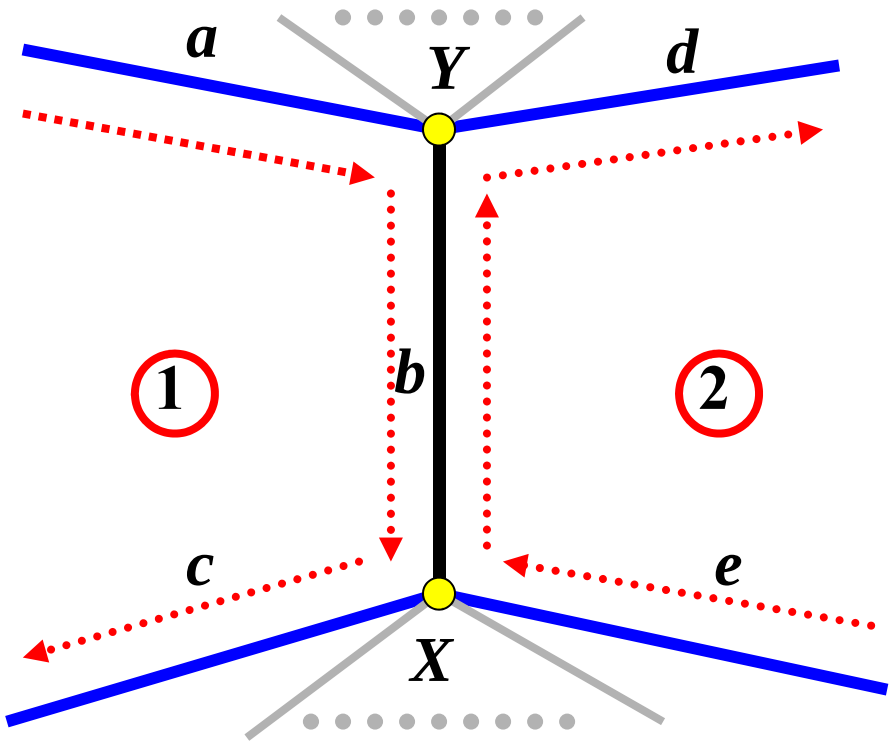
- If all faces are oriented clock-wise, each edge has *eight* pieces of incident information.



- The pred. and succ. edges of *b* with respect to face 2: *e* and *d*

What Is a Winged-Edge? 5/7

- How do we name faces 1 and 2?
- We use *left* and *right* faces!

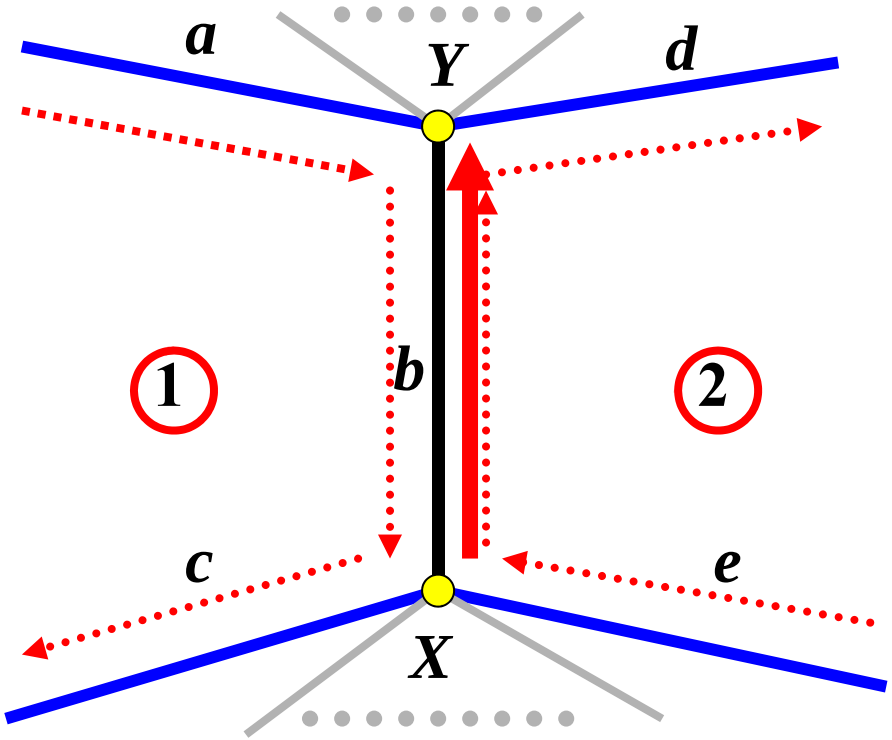


- **Which one is left, 1 or 2?**
- **Choose a direction for edge *b*, say from *X* to *Y* or from *Y* to *X*.**
- **Going from the start vertex to the end vertex, we know which face is the left one!**
- **If the start vertex is *X*, the *left* face is face 1.**

What Is a Winged-Edge? 6/7

Information for Edge *b* (from *X* to *Y*)

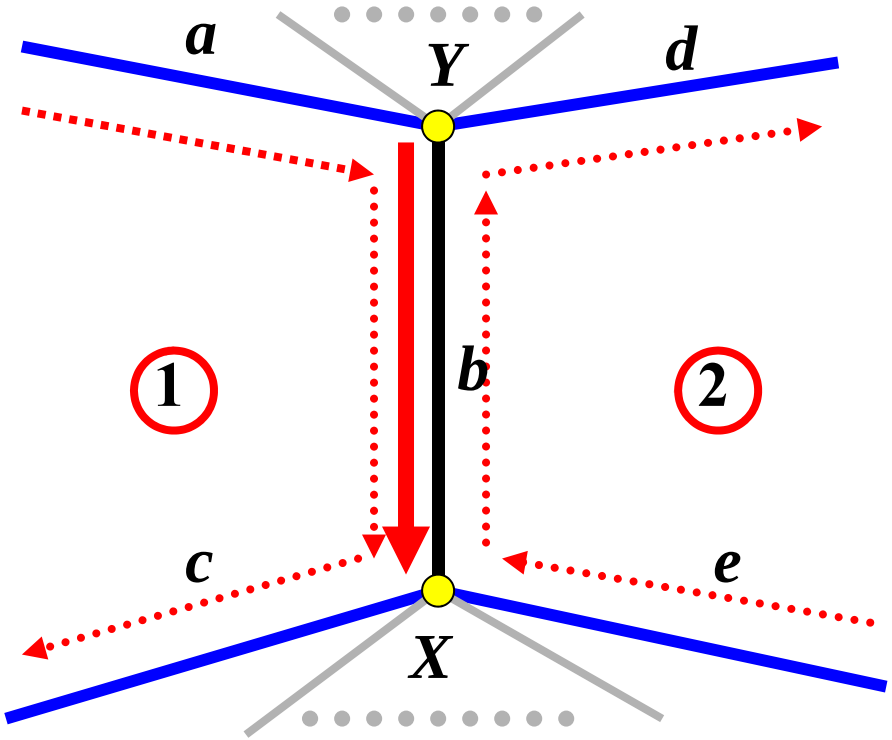
Start Vertex	End Vertex	Left Face	Right Face	Left Pred.	Left Succ.	Right Pred.	Right Succ.
X	*Y*	1	2	*a*	*c*	*e*	*d*



What Is a Winged-Edge? 7/7

Information for Edge *b* (from *Y* to *X*)

Start Vertex	End Vertex	Left Face	Right Face	Left Pred.	Left Succ.	Right Pred.	Right Succ.
Y	*X*	2	1	*e*	*d*	*a*	*c*

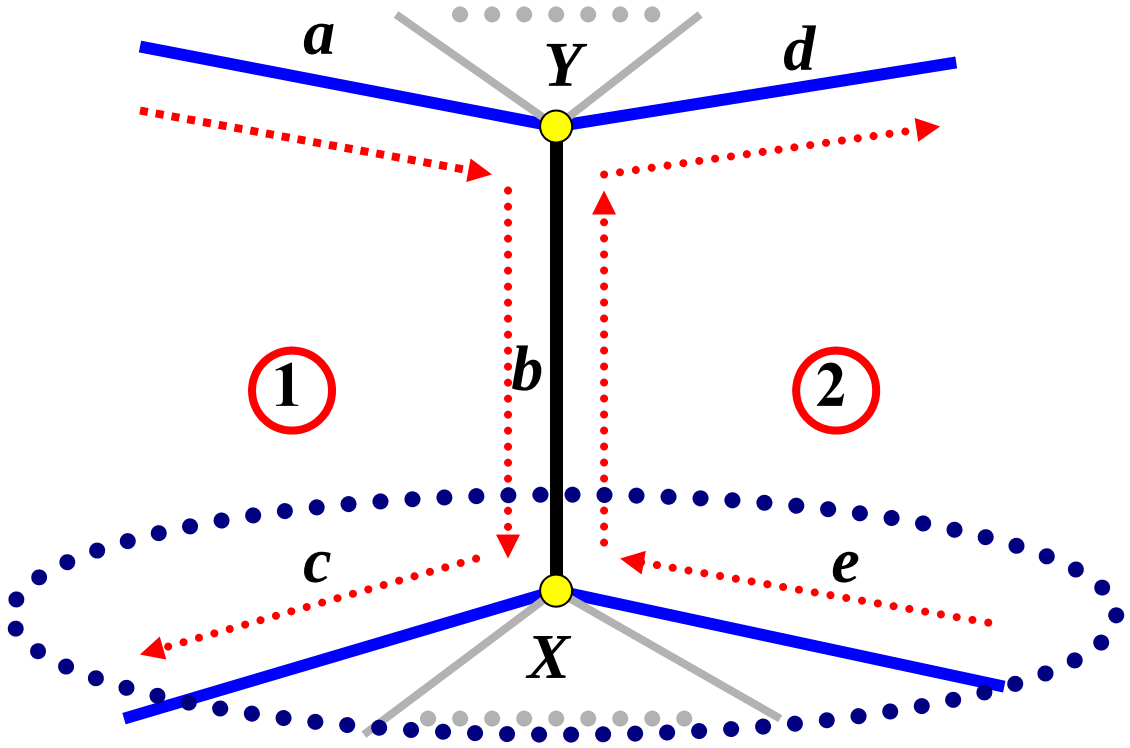


The Winged-Edge Data Structure: 1/6

- **The winged-edge data structure has three tables, *edge* table, *vertex* table, and *face* table.**
- **Each edge has one row in the *edge* table. Each row contains the eight pieces information of that edge.**
- **Each vertex has one entry in the *vertex* table. Each entry has a pointer to an incident edge (in the edge table) of that vertex.**
- **Each face has one entry in the *face* table. Each entry has a pointer to an incident edge (in the edge table) of that face.**

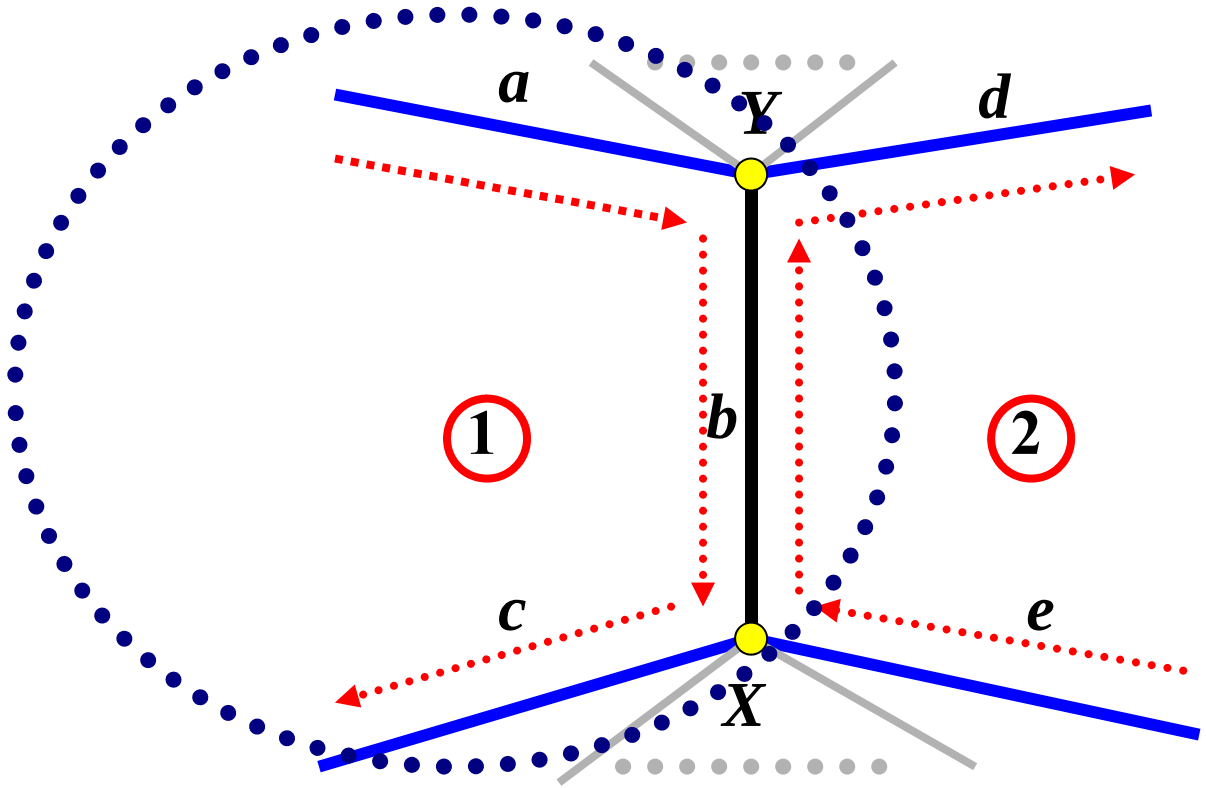
The Winged-Edge Data Structure: 2/6

- **The vertex table entry for vertex *X* may be the edge table entry of edges *c*, *b*, *e*, or any other incident edge.**



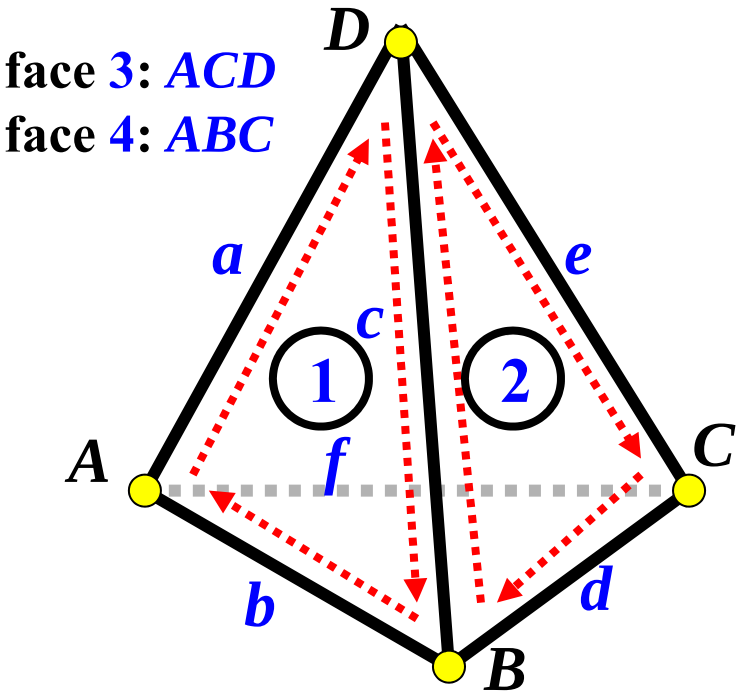
The Winged-Edge Data Structure: 3/6

- The face table entry for face 1 may be the edge table entry of edges *a*, *b*, *c*, or any other incident edge.



The Winged-Edge Data Structure: 4/6

- The following tetrahedron has four vertices *A*, *B*, *C* and *D*, six edges *a*, *b*, *c*, *d*, *e*, *f*, and four faces 1, 2, 3 and 4.



Vertex Table

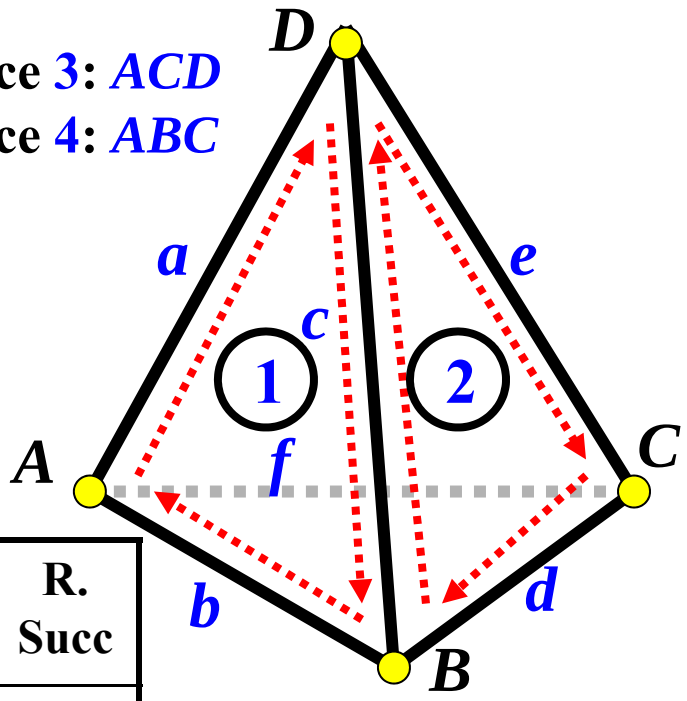
Vertex	Edge
A	*a*
B	*b*
C	*d*
D	*a*

Face Table

Face	Edge
1	*a*
2	*c*
3	*a*
4	*b*

The Winged-Edge Data Structure: 5/6

face 3: *ACD*
face 4: *ABC*



Edge Table

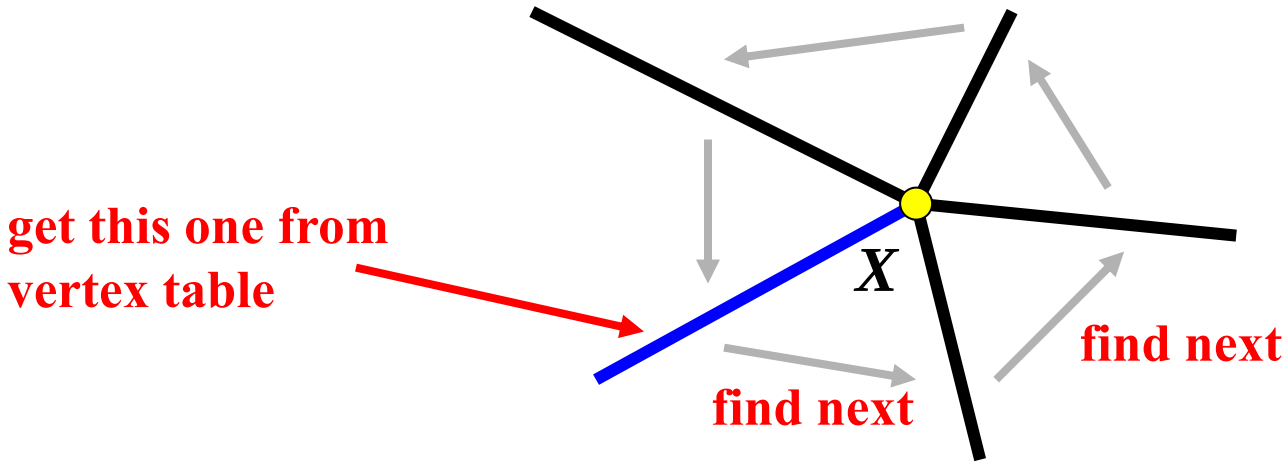
Edge	Start Vtx	End Vtx	L. Face	R. Face	L. Pred	L. Succ	R. Pred	R. Succ
a	*A*	*D*	3	1	*e*	*f*	*b*	*c*
b	*A*	*B*	1	4	*c*	*a*	*f*	*d*
c	*B*	*D*	1	2	*a*	*b*	*d*	*e*
d	*B*	*C*	2	4	*e*	*c*	*b*	*f*
e	*C*	*D*	2	3	*c*	*d*	*f*	*a*
f	*A*	*C*	4	3	*d*	*b*	*a*	*e*

The Winged-Edge Data Structure: 6/6

- **The winged-edge data structure seems to be very "unstructured;" however, it does record the incidence relations in a clever way.**
- **This clever way permits a program to answer many topological inquires very efficiently.**
- **If (1) *V*, *E* and *F* are the numbers of vertices, edges, and faces and (2) each entry in the table uses one memory unit, the vertex table, edge table, and face table require *V*, 8*E* and *F* memory units, respectively.**

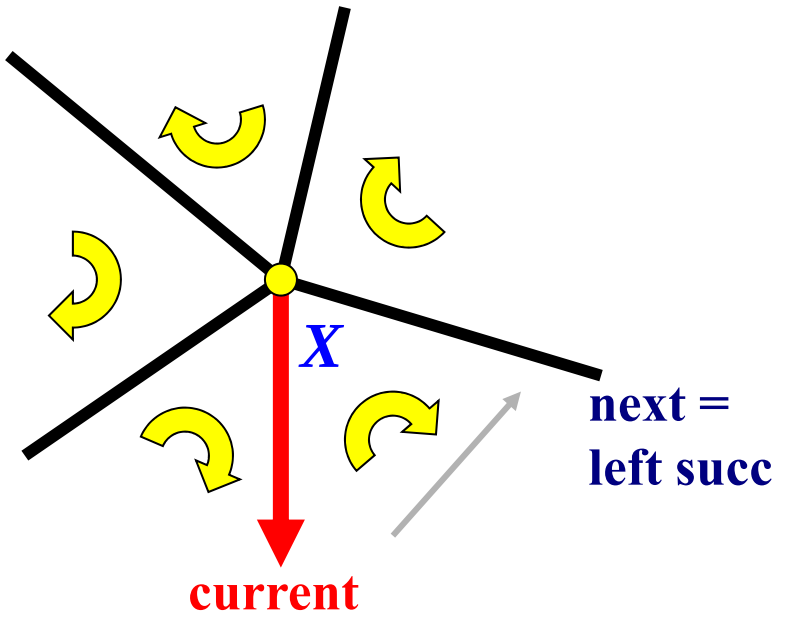
Inquiry Example 1: 1/4

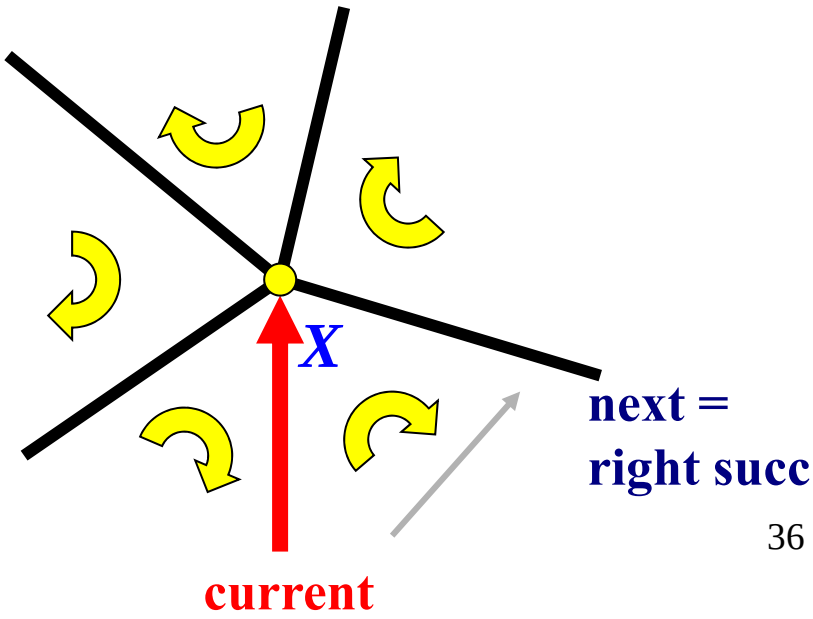
- Find all incident edges of a given vertex *X*.
- **Find *one* incident edge of *X* from its vertex table entry.**
- **Then, find the next incident edge, and "loop around" to find other incident edges.**



Inquiry Example 1: 2/4

- How do we find the next edge from the current one?
- **Let us use the counter clock-wise order.**
- **We have *two* cases to consider:**





Inquiry Example 1: 3/4

- Here is a possible algorithm:

```
Given vertex X;
Retrieve an incident edge e of X from vertex table;
Let s be e;                          // working variable
do                                   // move to next edge
    Output s;                        // found one incident edge
    if start vertex of s is equal to X then
        s = the successor of the left face of s
    else
        s = the successor of the right face of s;
    end if
while s ≠ e;                         // loop back if not equal
```

Inquiry Example 1: 4/4

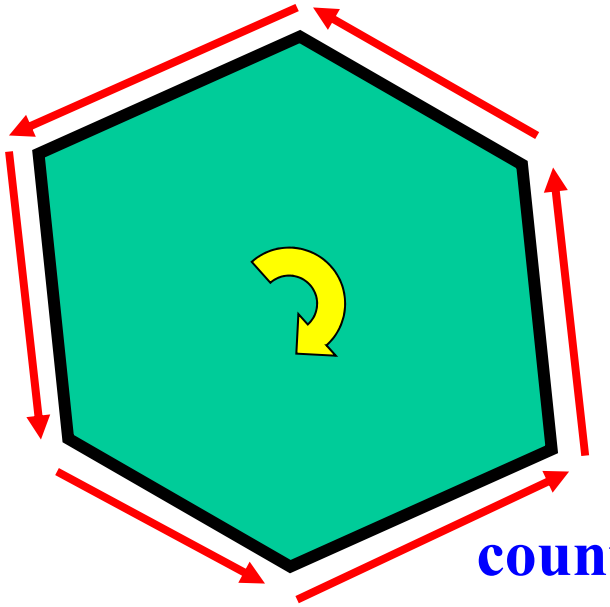
- **This algorithm finds all incident edges of a vertex by scanning those edges in the neighborhood of the given vertex. In fact, this algorithm scans *exactly* the incident edges.**
- **Since the number of incident edges of a vertex is usually small for most vertices, this algorithm is fast and of (almost) constant complexity!**
- **Compared with a conventional data structure, the winged-edge data structure wins hands-down in this case.**

Inquiry Example 2: 1/4

- Find all edges (and vertices, of course) of a given face *f*
- **Find one edge of *f* from its face table entry.**
- **Then, find the next incident edge, and "move forward" edge by edge to find all other edges.**
- **Keep in mind that the vertices of each face is oriented clock-wise.**

Inquiry Example 2: 2/4

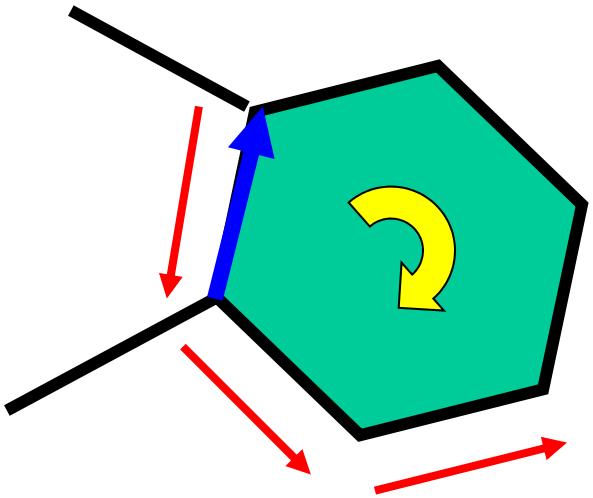
- Suppose we wish to list the edges in counter clock-wise order.
- The given face should always be on the left hand side of each edge.



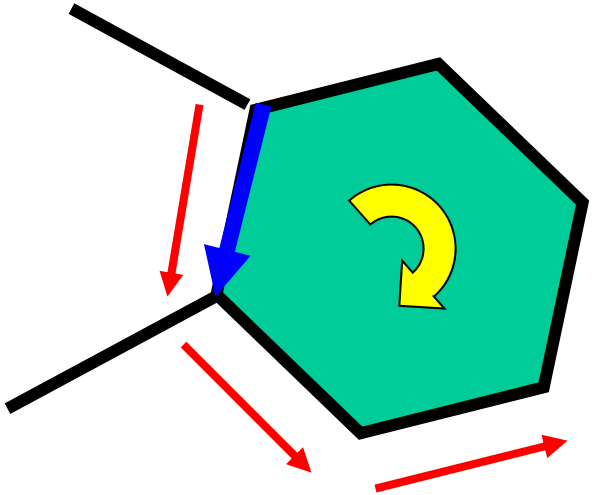
counter clock-wise

Inquiry Example 2: 3/4

- **Since the meaning of "left" and "right" face of an edge is based on that edge's orientation, we have to find the true meaning of "to the left" of an edge when traversing.**



right face of current edge
use right predecessor



left face of current edge
use left predecessor

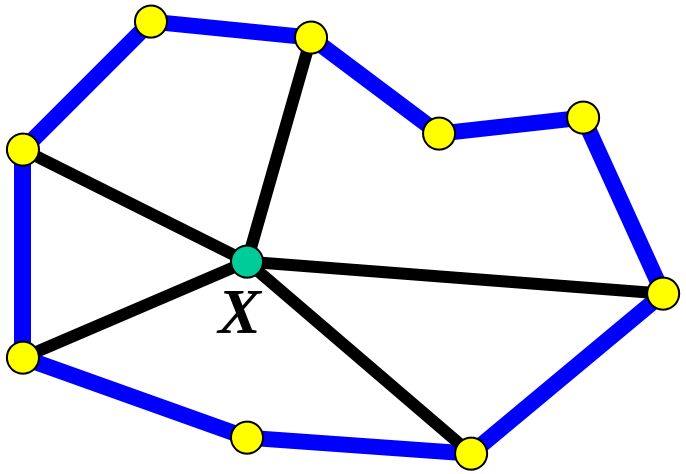
Inquiry Example 2: 4/4

- Here is a possible algorithm:

```
Given face f;
Retrieve an incident edge e of f from face table;
Let s be e;                          // working variable
do                                   // move to next edge
    Output s;                        // found one incident edge
    if f is the left face of s then
        s = the predecessor of the left face of s
    else
        s = the predecessor of the right face of s;
    end if
while s ≠ e;                         // loop back if not equal
```

Discussions

- Both examples list elements in counter clock-wise order. It is easy to change to clock-wise.
- The second example obviously requires more processing than a conventional data structure does; however, the first one is much faster.
- Based on what you know, do this inquiry:



Given a vertex *X*, find all "outer" edges of *X* (*i.e.*, the link) in counter clock-wise order.

Exercise

- **The half-edge data structure is an extension to the winged-edge data structure, and is more popular and widely used today.**
- **It splits each edge into two, each of which is referred to as an *half-edge*.**
- **Search the web to learn more about this data structure.**
- **An open source software, OpenMesh, developed by Prof. Leif Kobbelt, is available here: www.openmesh.org**

The End