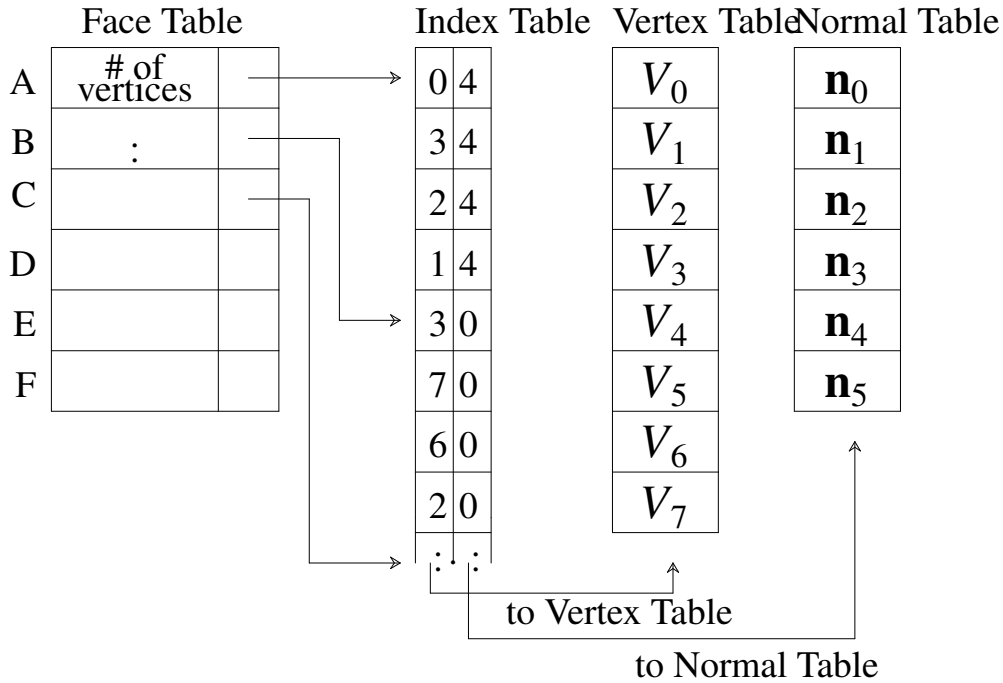


# 13 Data Structures for Graphics

- With the following representation,

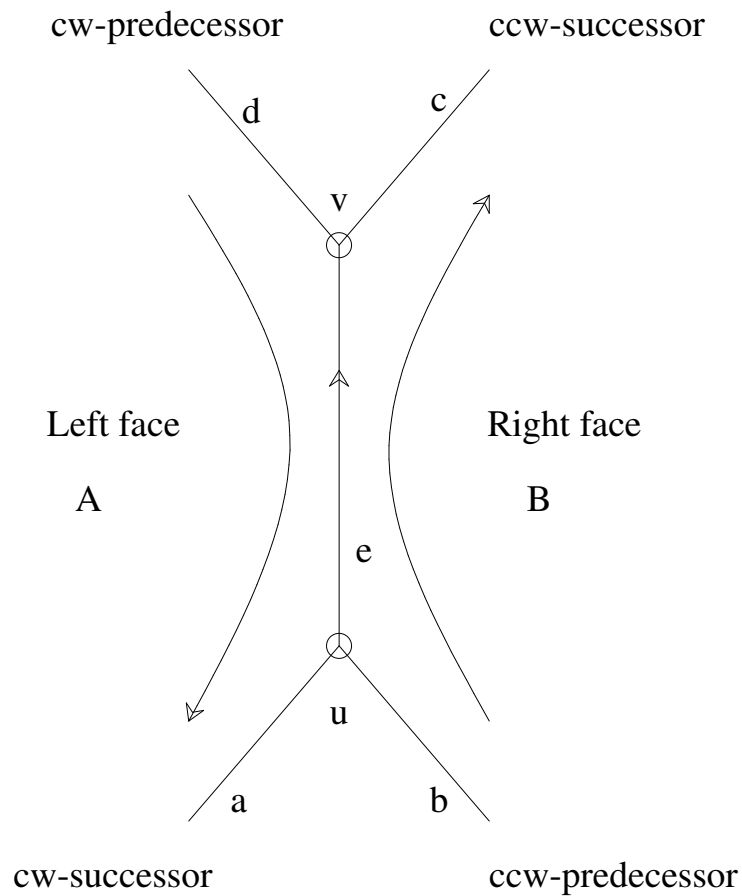


would you be able to answer the following queries in constant time?

- Given a triangle, what are the three adjacent triangles?
- Given an edge, which two triangles share it?
- Given a vertex, which faces share it?
- Given a vertex, which edges share it?

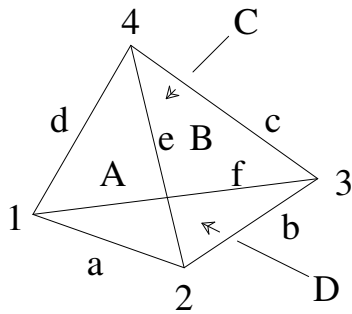
## 13.1 Winged-Edge Data Structure

- To avoid using variable-length data structures
- Hide the implementation behind a class interface



Edge	Vertices		Faces		Clockwise		Counter-Clockwise	
	from	to	left	right	pred	succ	pred	succ
$e$	$u$	$v$	$A$	$B$	$d$	$a$	$b$	$c$

- Example: representing a tetrahedron



Edge Name	Vertices		Faces		Clockwise		Counter-Clockwise	
	from	to	left	right	pred	succ	pred	succ
a	1	2	A	D	e	d	f	b
b	2	3	B	D	c	e	a	f
f	3	1	C	D	d	c	b	a
c	3	4	B	C	e	b	f	d
d	1	4	C	A	c	f	a	e
e	2	4	A	B	d	a	b	c

Vertex	Edge
1	d
2	b
3	b
4	c

Face	Edge
A	d
B	e
C	d
D	b

## 13.2 Scene Graphs

- A tree structure with each node representing an arm and each edge representing a joint
- Matrix to apply to an object is the product of the matrices in the chain from the object to the root of the tree
- Use *matrix stack* to achieve efficient implementation

