CS3242 Lab 2

Triangulation Data Structure

- Point array

PIdx	x	y	z
1	20	10	20
2	30	15	12
…	…	…	…
n	…	…	…

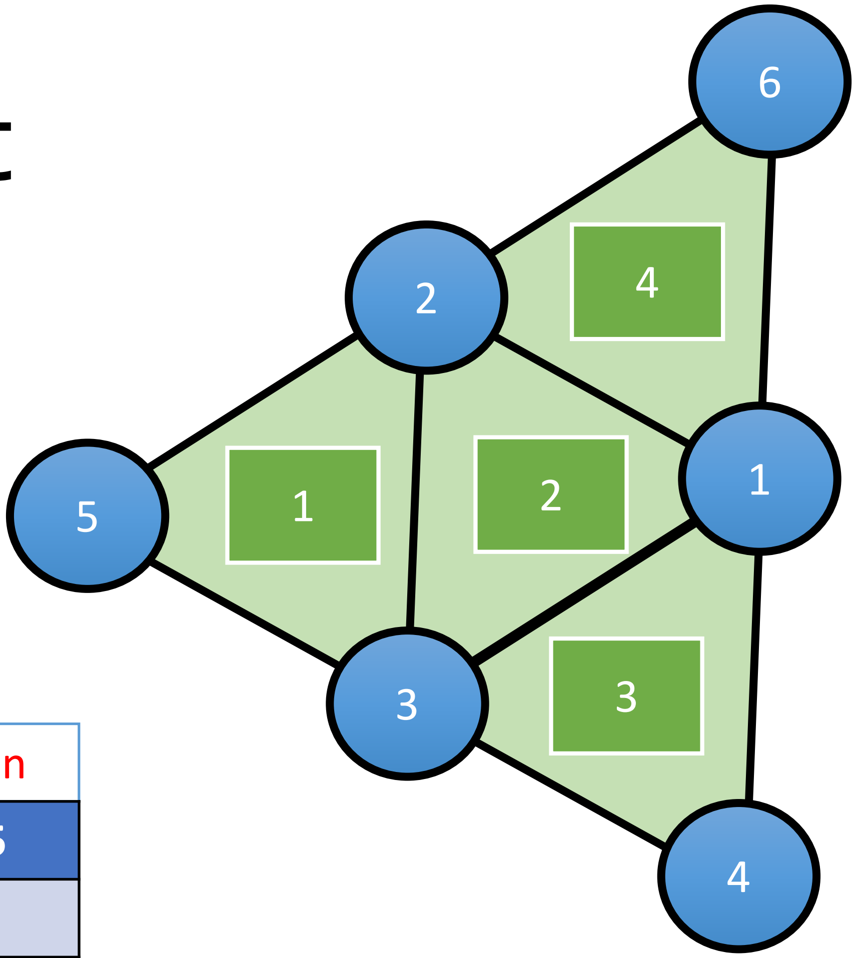
- Triangle array

	Vertex Indices			Triangle Indices for “fnext” WITH version					
TIdx	v0	v1	v2	ti0	ti1	ti2	ti3	ti4	ti5
1	3	2	5						
2	2	3	1						
3	4	1	3						
4	2	1	6						
…	…	…	…						
m	…	…	…						

Constructing `flist` for `fnext`

- Let's consider which triangle should be in the fnext WITHOUT version first
  - Namely, only the triangle index
- Triangle array

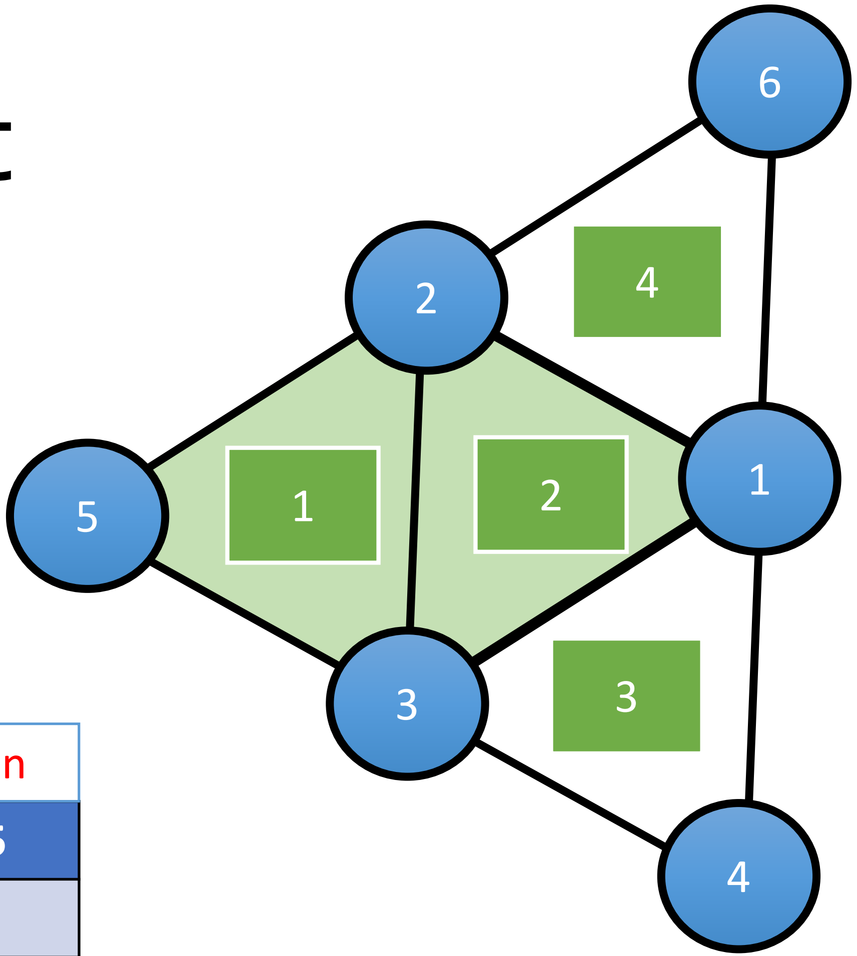
	Vertex Indices			Triangle Indices for "fnext" WITH version					
TIdx	**v0**	**v1**	**v2**	**ti0**	**ti1**	**ti2**	**ti3**	**ti4**	**ti5**
1	3	2	5						
2	**2**	**3**	1	**?**					
3	4	1	3						
4	2	1	6						
…	…	…	…						
m	…	…	…						



Constructing `flist` for `fnext`

- for tiN, it should be the triangle after applying fnext to version N of the triangle
  - For ti0, version 0 of triangle 2 should be 231
  - After `fnext`, it should be triangle 235
  - 235 is the `sym()` of 325, namely, version 3

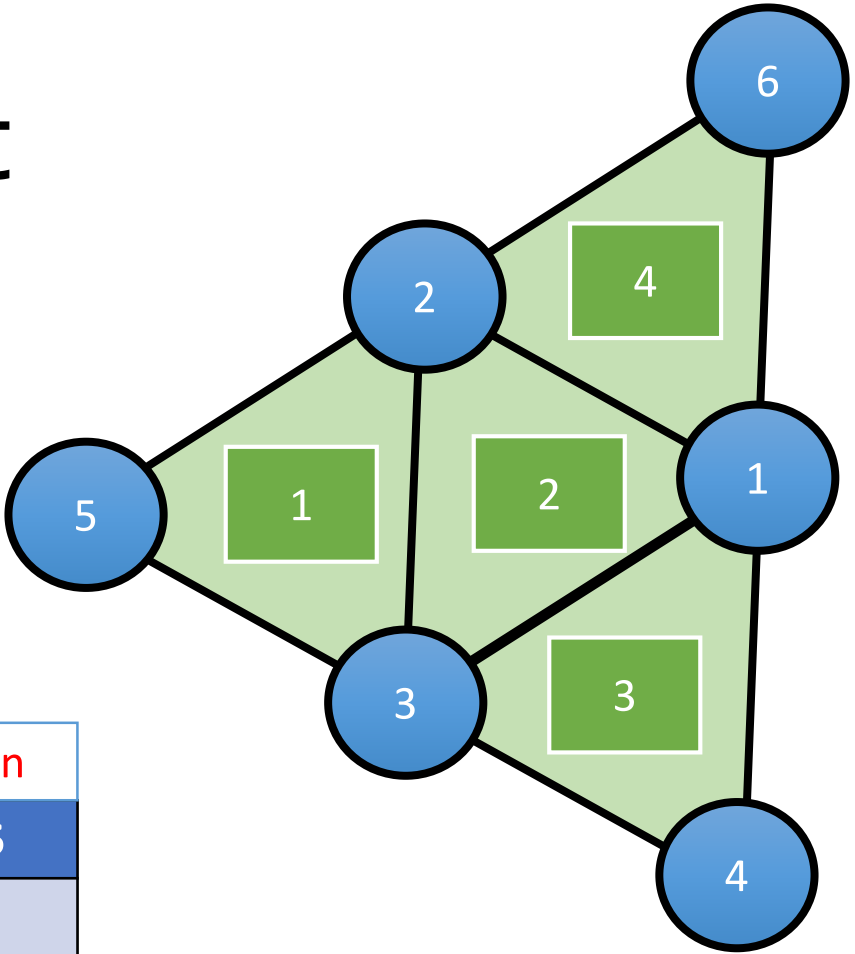
	Vertex Indices			Triangle Indices for "fnext" WITH version					
TIdx	**v0**	**v1**	**v2**	**ti0**	**ti1**	**ti2**	**ti3**	**ti4**	**ti5**
1	3	2	5						
2	**2**	**3**	1	**?**					
3	4	1	3						
4	2	1	6						
…	…	…	…						
m	…	…	…						



Constructing `flist` for `fnext`

- Index of triangle 325 is 1 and version is 3
  - `(1 << 3) | 3 ⇒ 11`
- Immediately, the fnext of 321 (aka sym(231)) is sym(325)
  - `fnext(321) = sym(fnext(231)) (OrTri:8)`

	Vertex Indices			Triangle Indices for "fnext" WITH version					
TIdx	**v0**	**v1**	**v2**	**ti0**	**ti1**	**ti2**	**ti3**	**ti4**	**ti5**
1	3	2	5						
2	**2**	**3**	1	**11**			?		
3	4	1	3						
4	2	1	6						
…	…	…	…						
m	…	…	…						



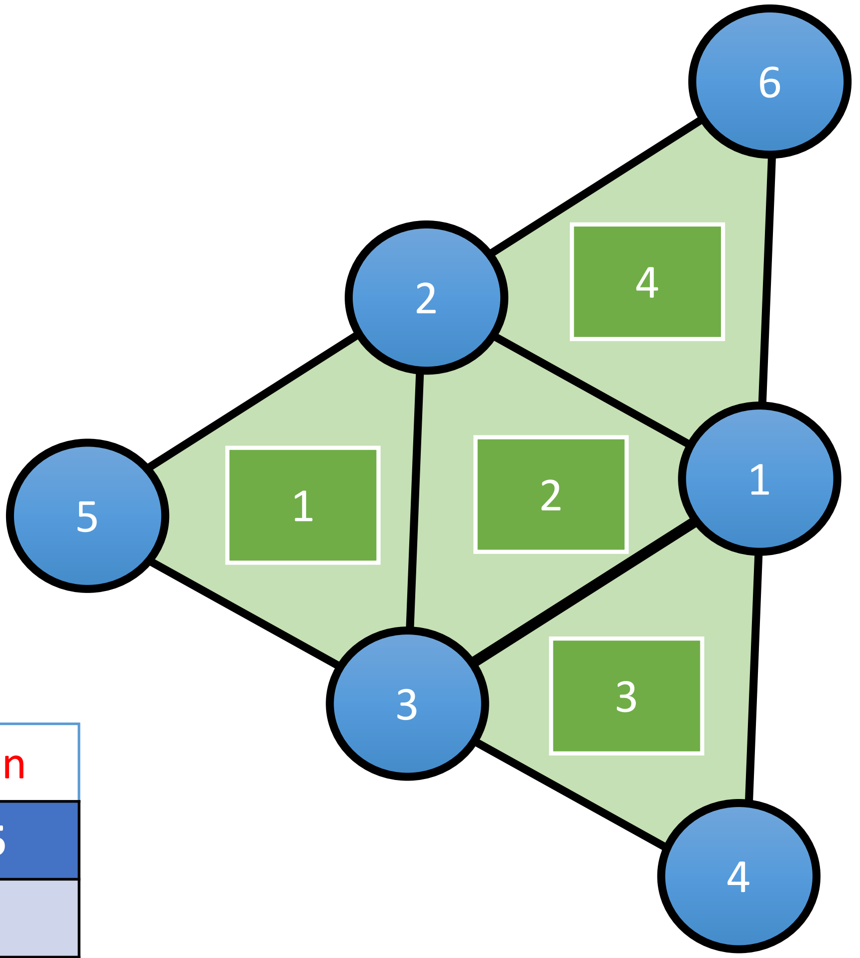
Two Steps for Constructing `flist` for `fnext`

1. For each version of an OrTri abc, search for a triangle $\boldsymbol{\tau}$ with vertices a and b
   - Brute force?
   - Hashmap
2. Find the correct version of $\boldsymbol{\tau}$ and put its OrTri representation into the right entry in `fnlist`.

Flipping Triangles Version 0

- What is direction of the triangle normal vector facing for all the triangle with version 0
- Very lucky

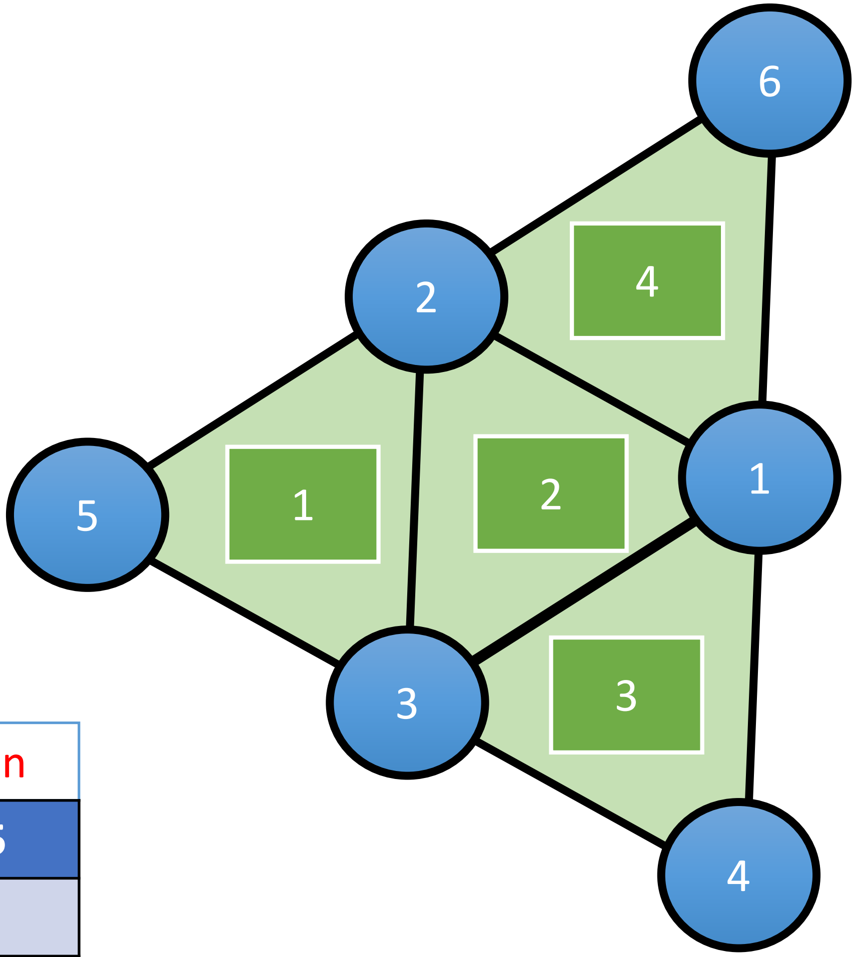
	Vertex Indices			Triangle Indices for “fnext” WITH version					
TIdx	**v0**	**v1**	**v2**	**ti0**	**ti1**	**ti2**	**ti3**	**ti4**	**ti5**
1	3	2	5						
2	2	3	1						
3	4	1	3						
4	2	1	6						
…	…	…	…						
m	…	…	…						



Flipping Triangles Version 0

- What is direction of the triangle normal vector facing for all the triangle with version 0
- Not so lucky, better flip

	Vertex Indices			Triangle Indices for “fnext” WITH version					
TIdx	**v0**	**v1**	**v2**	**ti0**	**ti1**	**ti2**	**ti3**	**ti4**	**ti5**
1	3	2	5						
2	3	2	1						
3	4	1	3						
4	2	1	6						
…	…	…	…						
m	…	…	…						



Accessing Different Types of Simplices

- Iterate all the vertices and triangles
  - Simple

- Iterate all the edges?
  - Not a very popular operation
  - If we really need to do so, maybe the best way is to do a BFS/DFS on the mesh, because every edge will be “crossed” only once

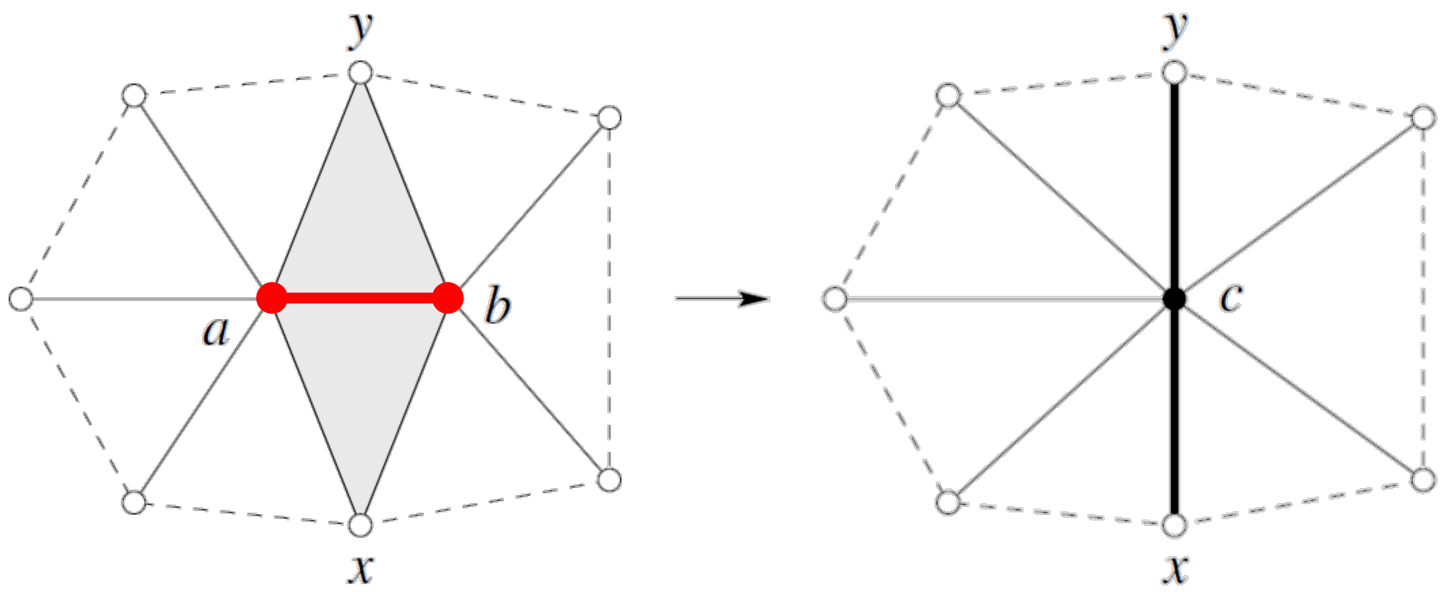
Accessing Different Types of Simplices

- an OrTri can represent a vertex and an edge also
    - if an OrTri `t` is an edge, then it is the edge `(org(t) , dest(t))`
    - If an OrTri `t` is a vertex, then it is the vertex `org(t)`.

- However, if you only have a vertex index `vi`, how do you find any triangle that shares that vertex?
    - Create one more list vflist that stores any one of the triangles that share a vertex
    - Need to update this whenever there is mesh modification as well

Recap: Edge Contraction

- For an edge ab, that is shared by two triangles
- Merge the two vertices a and b into a new one c
- Delete the two triangles
- And "stitch" the empty hole



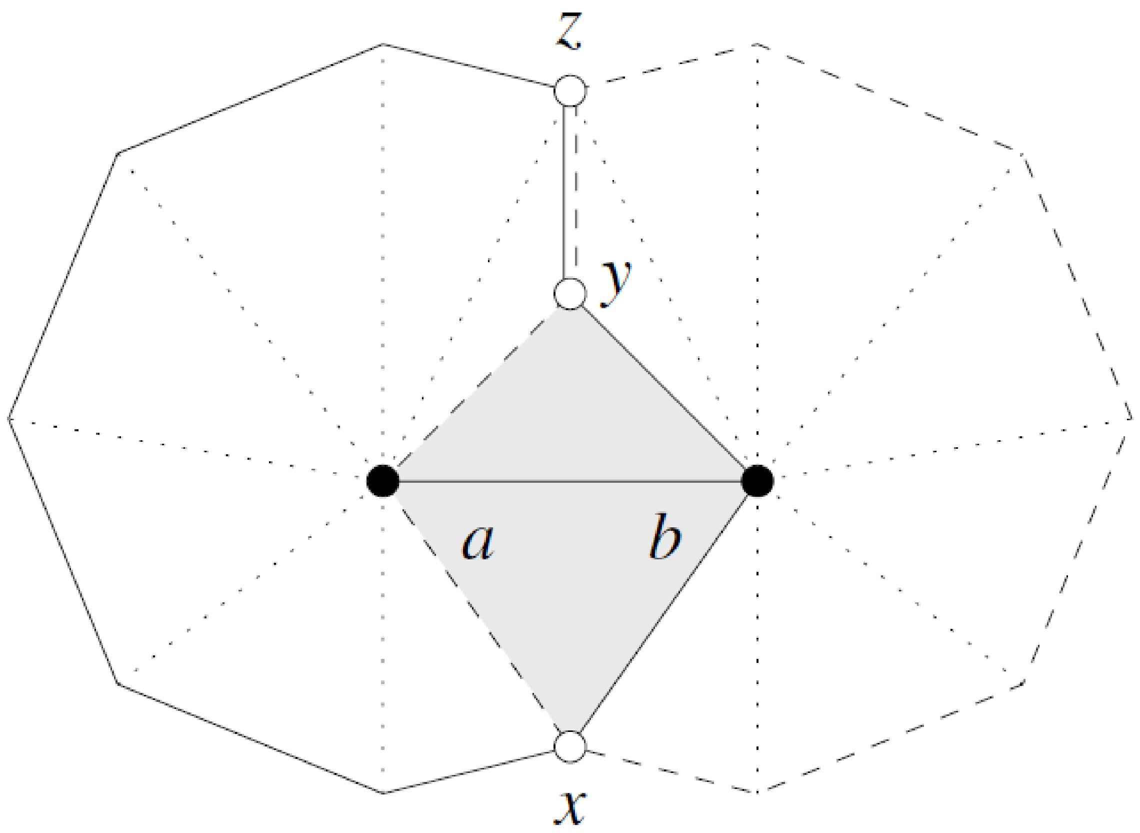
Recap

- We can only contract an edge ab only if

$$\text{Lk}(a) \cap \text{Lk}(b) = \text{Lk}(ab)$$

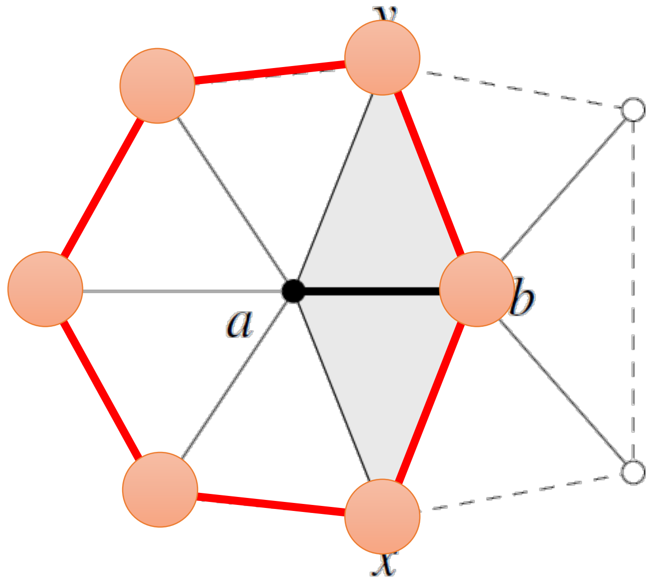
- so that the topology of the mesh will be preserved

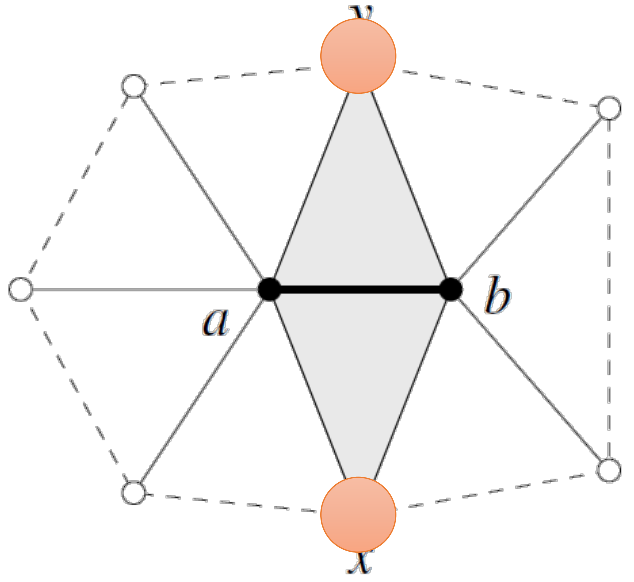
- We try to explain the Lk operations in very layman term
  - Then give the formal definition



The Lk (Link) Function in General

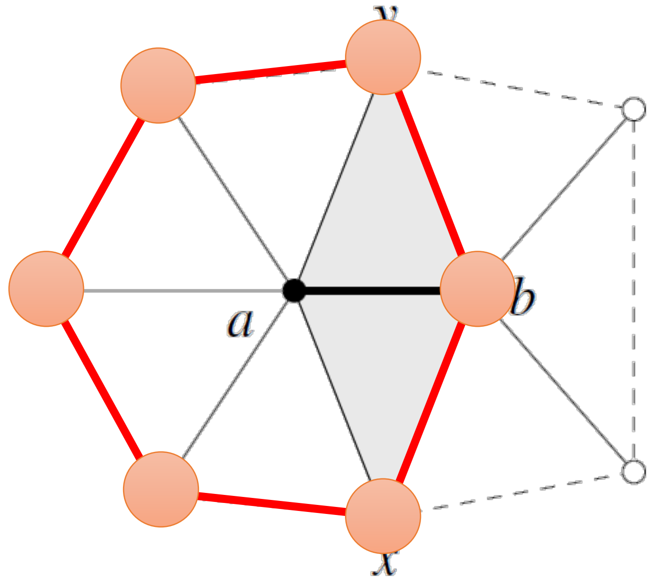
- This is a layman explanation in general, but it may have *exceptions*
- The link of a vertex a, $\mathrm{Lk}(a)$, is all the vertices and edges that “surround” a
- The link of an edge ab, $\mathrm{Lk}(ab)$, is set of vertices on the neighboring triangles excluding a and b



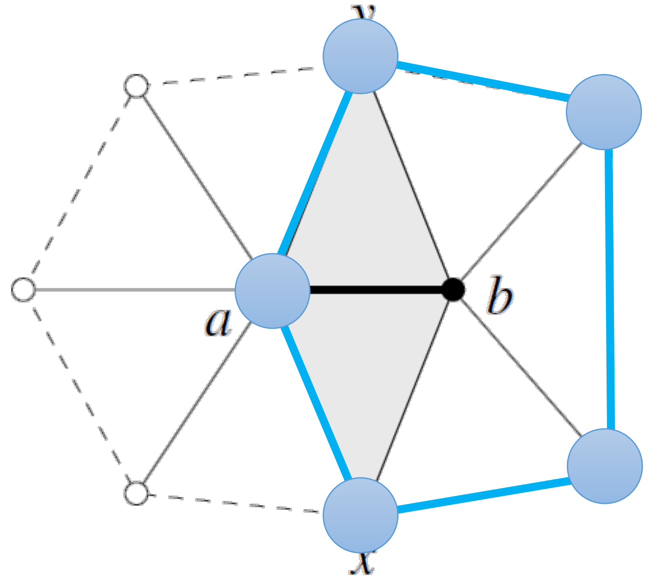


For Contractable case

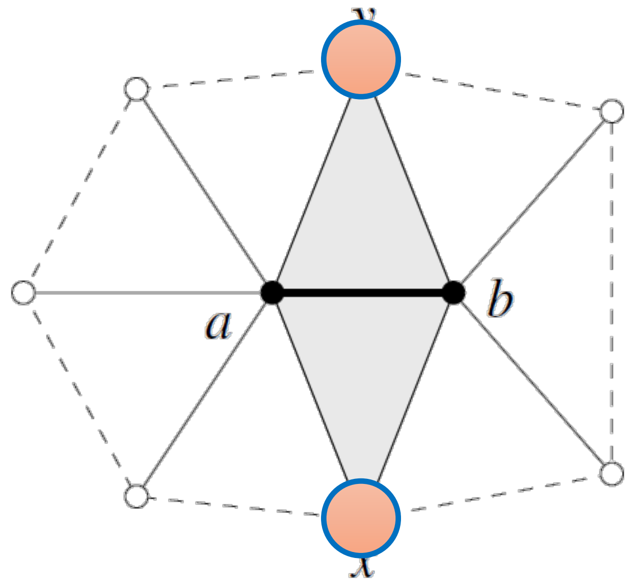
- $\mathrm{Lk}\,(a) \cap \mathrm{Lk}\,(b) = \mathrm{Lk}\,(ab)$



$\mathrm{Lk}(a)$



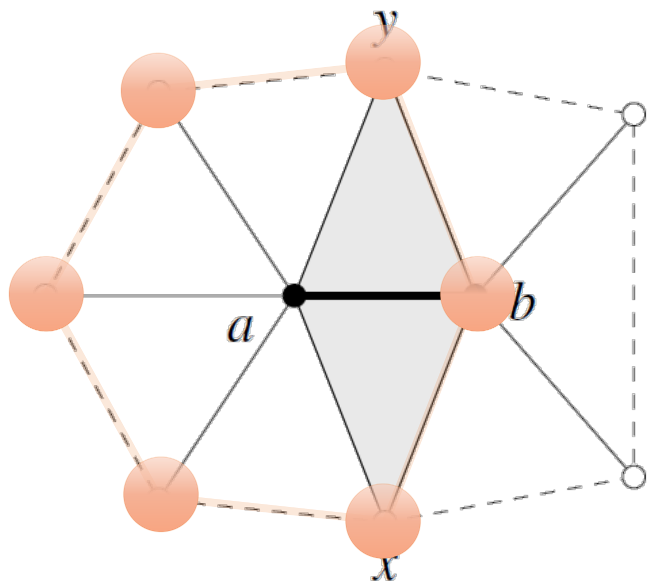
$\mathrm{Lk}(b)$



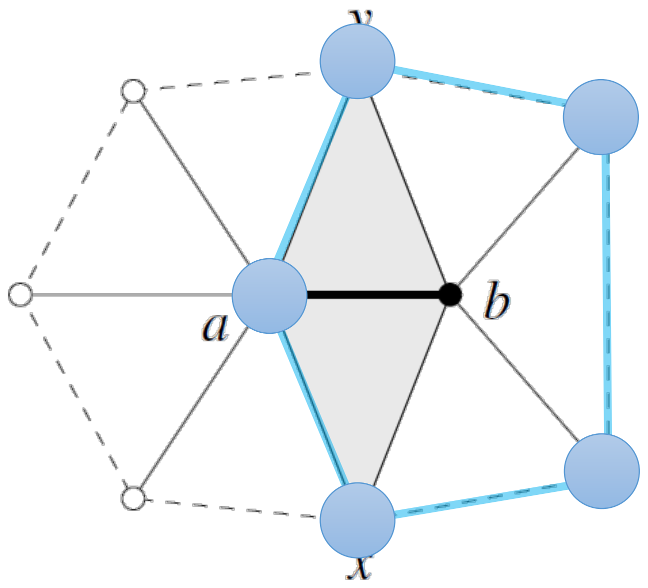
$\mathrm{Lk}(ab)$

For Contractable case

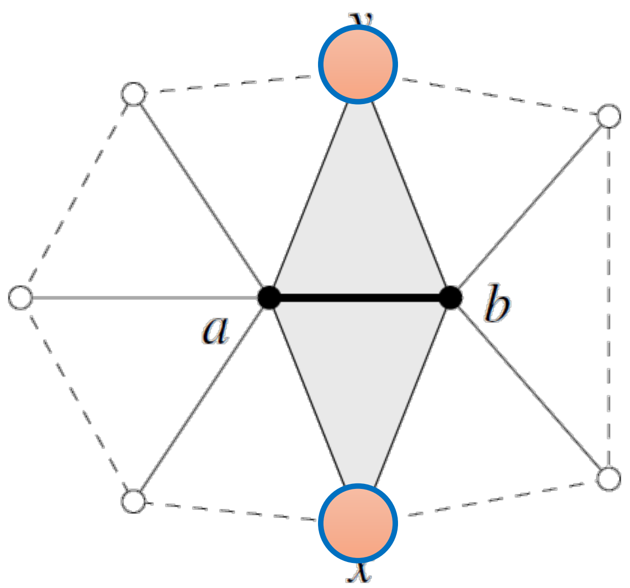
- $\mathrm{Lk}\,(a) \cap \mathrm{Lk}\,(b) = \mathrm{Lk}\,(ab)$



$\mathrm{Lk}(a)$



$\mathrm{Lk}(b)$



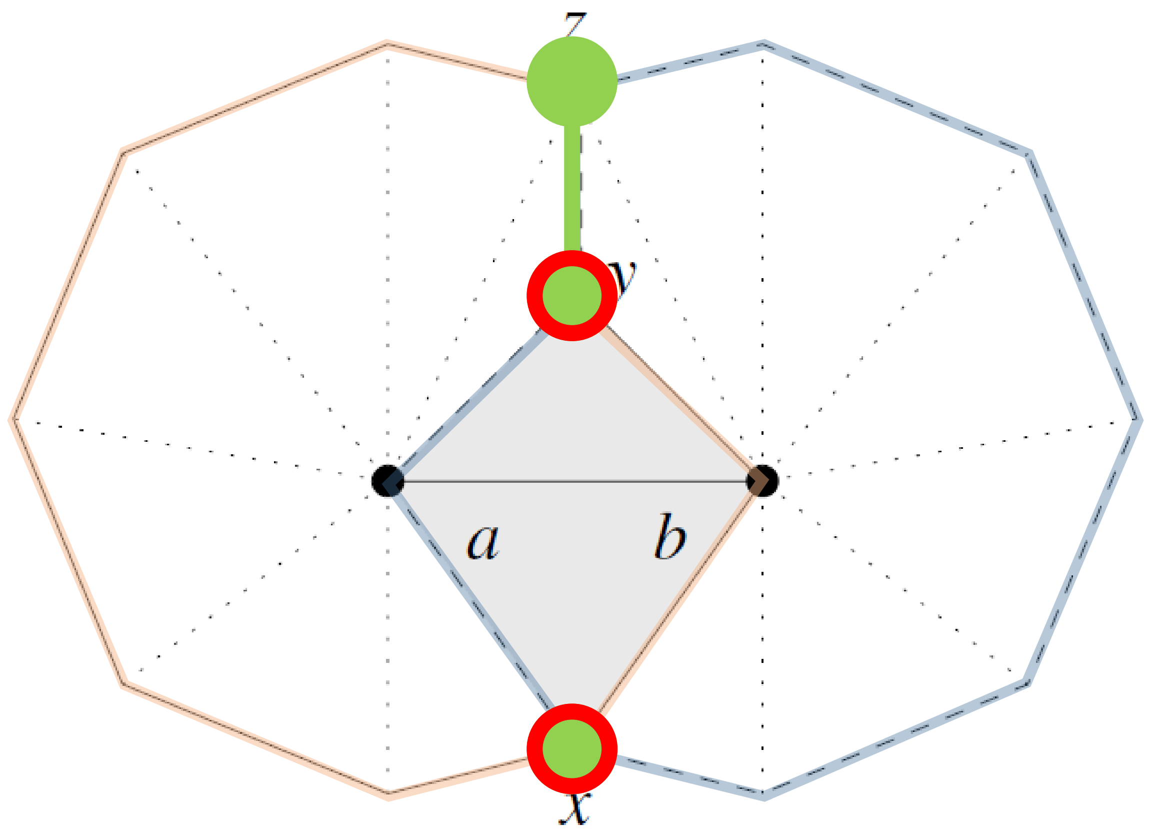
$\mathrm{Lk}(ab)$

For a Non-contractable Case

- $\mathrm{Lk}\,(a) \cap \mathrm{Lk}\,(b) = \{x, y, z, yz\}$
- $\mathrm{Lk}\,(ab) = \{x,\ y\}$

- So

$$\mathrm{Lk}\,(a) \cap \mathrm{Lk}\,(b) \neq \mathrm{Lk}\,(ab)$$



Formal Definition of Lk

- Given a simplicial complex K, and a subset $S \subseteq K$
- The *star* of a set of simplices S, $\mathrm{St}\,(S)$, is the set of all the simplices in K such that each simplex has a face in S

$$\mathrm{St}\,(S) = \{\, \sigma \in K \mid \sigma \geq \tau \in S \,\}$$

- The *closure* of a set of simplices S, $\mathrm{Cl}\,(S)$, is the set of all the faces of the simplices in S.

$$\mathrm{Cl}\,(S) = \{\, \tau \in K \mid \tau \leq \sigma \in S \,\} \ .$$

- The *link* of a set of simplices S, $\mathrm{Lk}\,(S)$ is

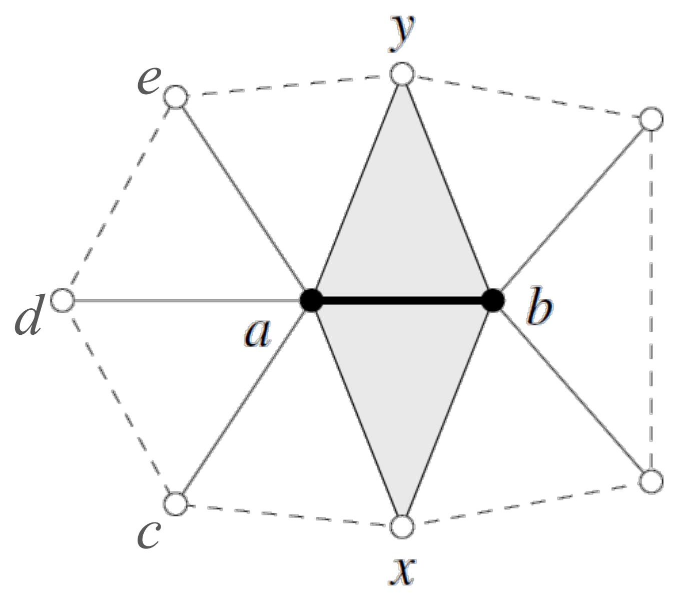
$$\mathrm{Lk}\,(S) \ = \ \mathrm{Cl}\,(\mathrm{St}\,(S)) - \mathrm{St}\,(\mathrm{Cl}\,(S)).$$

Star

- The *star* of a set of simplices S, $\mathrm{St}\,(S)$, is the set of all the simplices in K such that each simplex has a face in S

$$\mathrm{St}\,(S) = \{ \, \sigma \in K \mid \sigma \geq \tau \in S \, \}$$

- For example, the star of the vertex a is
  - $\{a, ab, ax, ay, ac, ad, ae, abx, aby, aey, aed, adc, acx\}$
- $\mathrm{St}(ab) = \{ab, aby, abx\}$
- Note that the $\mathrm{St}(a)$ and $\mathrm{St}(ab)$ are NOT simplicial complexes
  - What rules did they violate?

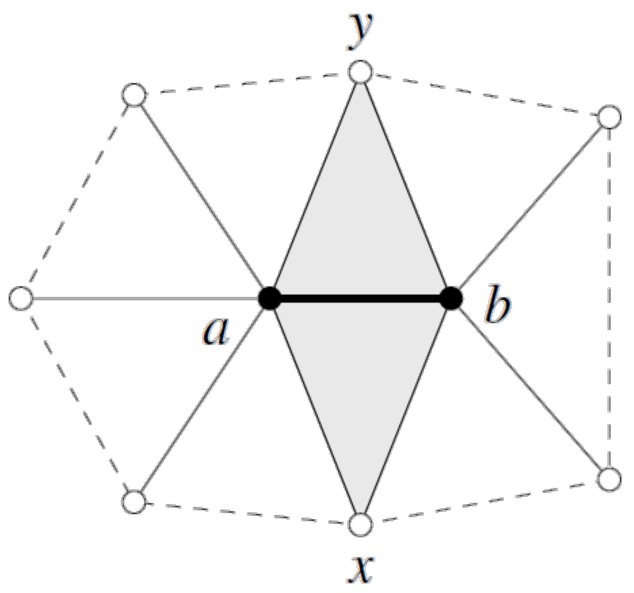


Closure

- The *closure* of a set of simplices S, $\mathrm{Cl}(S)$, is the set of all the faces of the simplices in S.

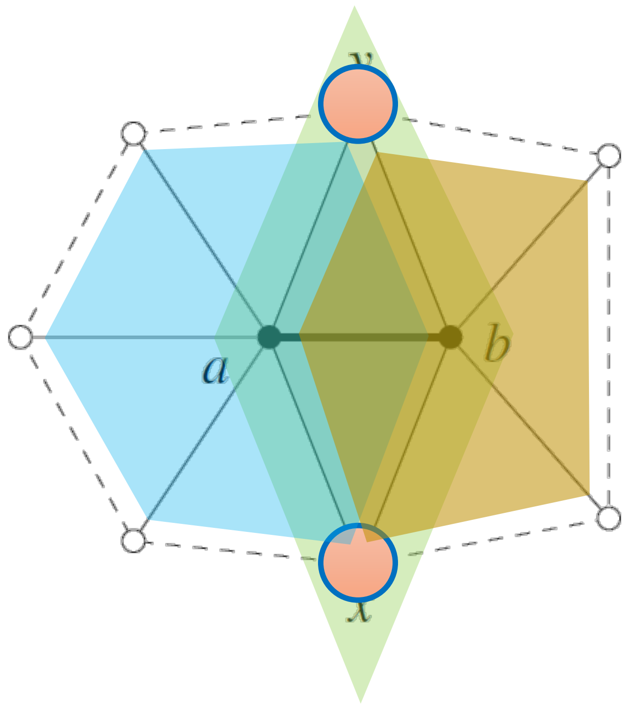
$$\mathrm{Cl}\,(S) = \{\, \tau \in K \mid \tau \le \sigma \in S \,\} .$$

- For example, if S is $\mathrm{St}(ab)$
  - $\mathrm{St}(ab) = \{ab, aby, abx\}$
  - $\mathrm{Cl}(S) = \{a, b, x, y, ab, ay, ax, bx, by\}$



Example: Link of ab

- The *link* of a set of simplices S, $\mathrm{Lk}\,(S)$ is
$$\mathrm{Lk}\,(S) \;=\; \mathrm{Cl}\,(\mathrm{St}\,(S)) - \mathrm{St}\,(\mathrm{Cl}\,(S)).$$
- $S = \{ab\}$
- $\mathrm{St}(S) = \{ab, aby, abx\}$
- $\mathrm{Cl}(\mathrm{St}(S)) = \{a, b, \boldsymbol{x}, \boldsymbol{y}, ab, ay, ax, bx, by, abx, aby\}$
- $\mathrm{Cl}(S) = \{a, b, ab\}$
- $\mathrm{St}(\mathrm{Cl}(S)) = \{a, ab, ax, ay, ac, ad, ae, abx, aby, aey, aed, adc, acx\} \cup \mathrm{St}(b) \cup \mathrm{St}(ab)$

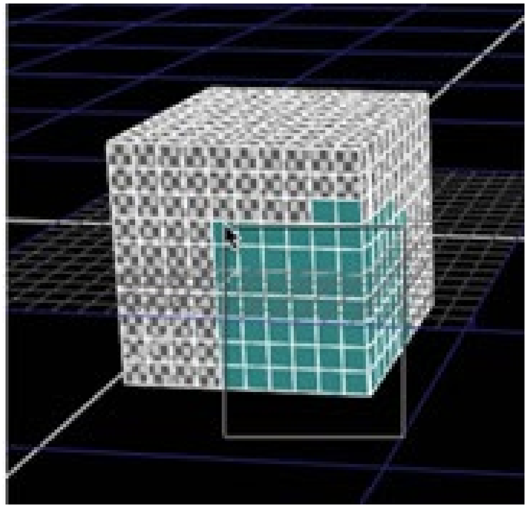


Lk(*ab*)

Tasks

- For main tasks, we expect you to try all of them. However, for optional tasks, we expect you only to try %80 of them. For example, if there are a total of 200 marks for all the optional quests (we are still adjusting that), we pick the optional questions from you (or you can choose them by yourself) the optional tasks you did that the maximum mark worth 160.
- For example, if we have 5 optional tasks with 40 marks each. And a student did all of them and the marks are 20, 40, 35, 40, 30 respectively. Then his mark for optional task will be 145 out of 160 (ignoring the 20).
- We will have a list of all main and optional tasks at the end of the semester for you to check.

Tasks



- Implement enext(), sym() (20 marks, main)
- Implement org(), dest() (20 marks, main)
- Implement fnext() (80 marks, main)
- Compute the Number of Components (20 marks, optional)
- Implement orientTriangles() (20 marks, optional)
- Compute Vertex Normal Vectors for Smooth Shading (10 marks, optional)
- Visualize boundary edges (10 marks, optional)
- Implementing Selection of Triangle by User Marquee (20 marks, optional)