University at Buffalo The State University of New York

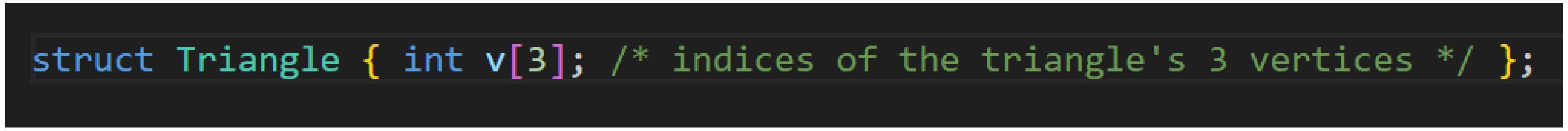
HPC SUBJECT: MESH ORIENTATION

Harvey Kwong, MS CSE, University at Buffalo

University at Buffalo The State University of New York

Orienting Simplices

Suppose we define a triangle by its 3 vertices. The orientation of that triangle is given by the sequence of these indices.



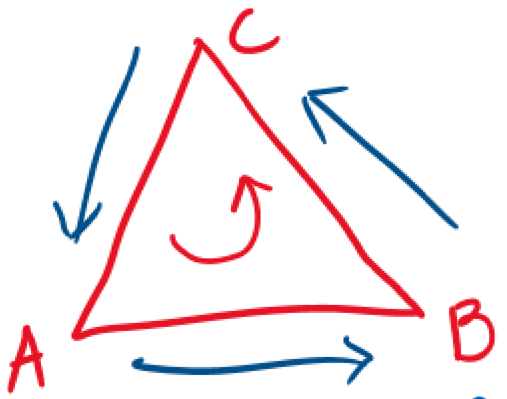
```
struct Triangle { int v[3]; /* indices of the triangle's 3 vertices */ };
```

We say a triangle is oriented once we arbitrarily fix an ordering for it. By doing so, we induce an orientation for the edges as well.

Flipping any two indices an odd number of times (ie. swapping v[1] and v[2]) will reverse the orientation (clockwise <-> counterclockwise)

Flipping them an even number of times preserves the orientation and the subsequent vertex order is considered equivalent.

$(A, B, C) == (B, C, A) \mathrel{!=} (C, B, A)$



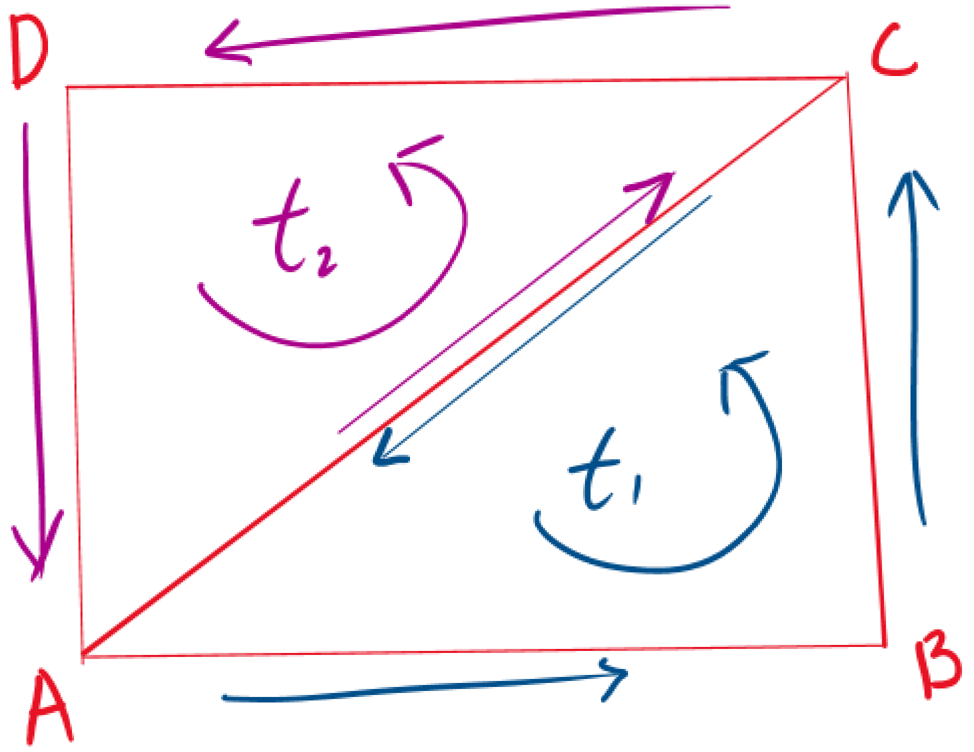
(A,B,C)

Orientation (Consistency)

For the whole mesh to have a consistent orientation, whenever two triangles share an edge, that edge's orientation should be opposite in the two triangles. If two adjacent triangles accidentally had the shared edge oriented the same way, they would form an inconsistent loop (one triangle's orientation would be flipped relative to the other), leading to a conflict

$t_1 = (A, B, C)$

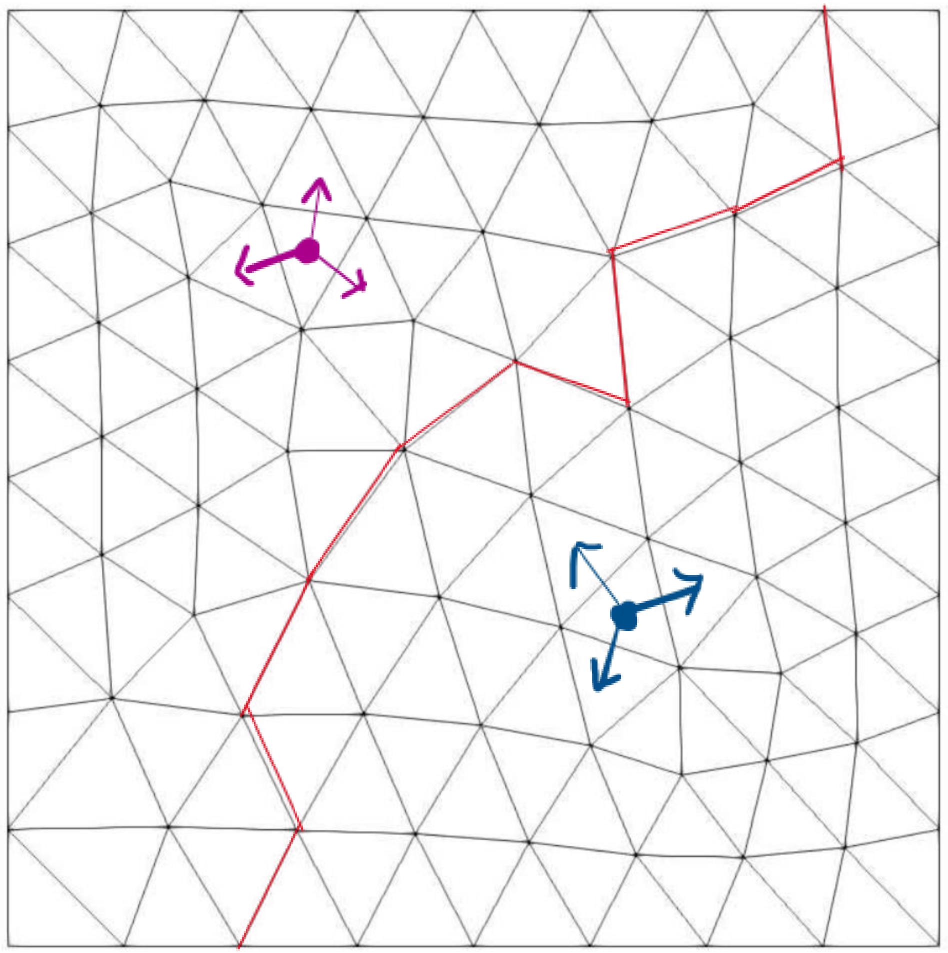
$t_2 = (A, C, D)$



Parallel Orientation

Assume the mesh is partitioned such that each processor has a list of triangles and knows which edges on its triangles are shared with triangles on neighboring processors.

- **Local Orientation**: Each processor orients their local submesh using the algorithm from last slide.
- **Boundary communication**: Each processor reports to neighboring processors regarding the orientation of shared edges.
- **Conflict resolution:** If two waves of opposing orientations collide, we use some scheme such as processor rank to determine who has the master orientation. The other processor will have to flip its orientations on the boundary and those changes propagate throughout its local submesh. Then, we repeat from step 2.

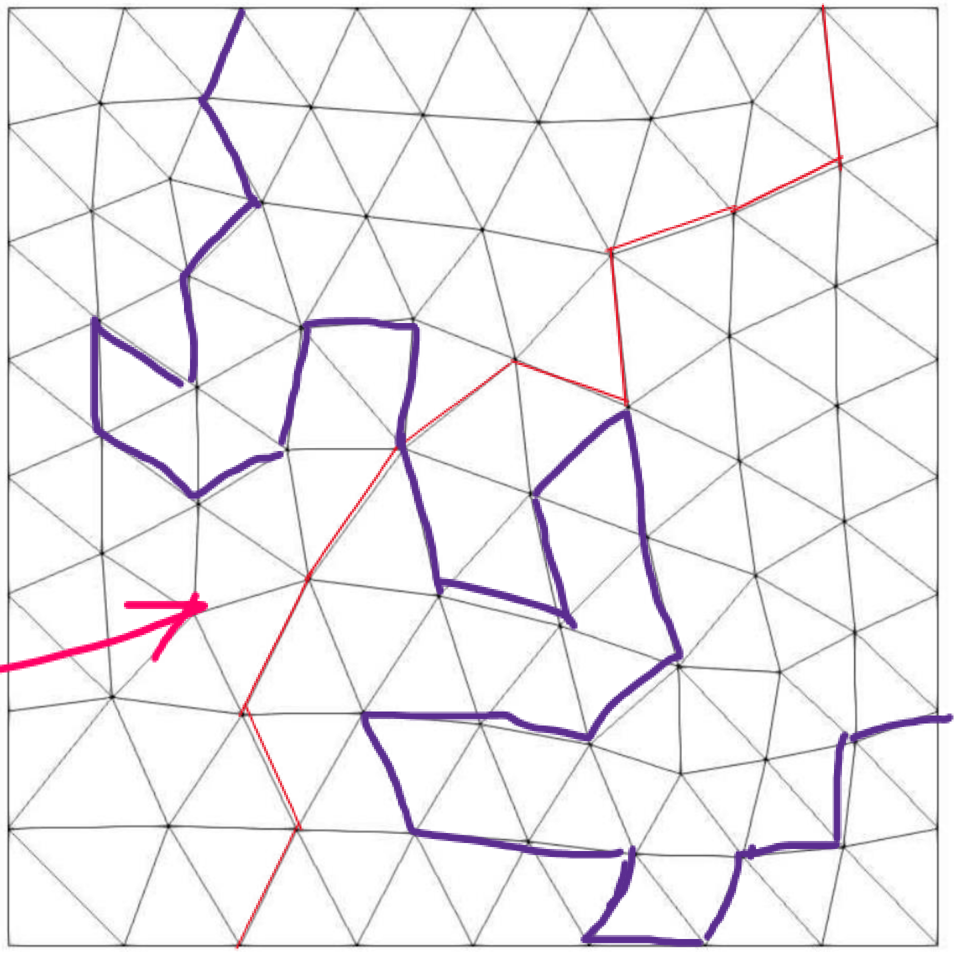


Parallel Orientation Communications

When the mesh (and submeshes) is balanced, the ratio of boundary elements to internal elements is very small. The cost of communicating orientations is small as well. Since the bulk of computation occurs locally within each processor's submesh, this is a highly parallelizable algorithm.

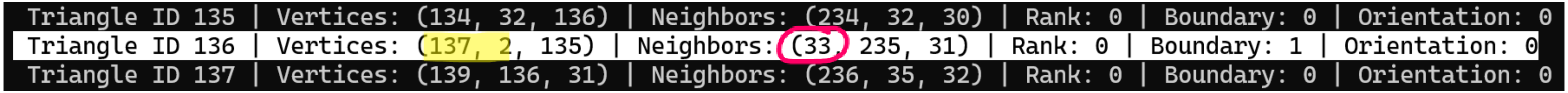
10 communications needed

Not Balanced partition



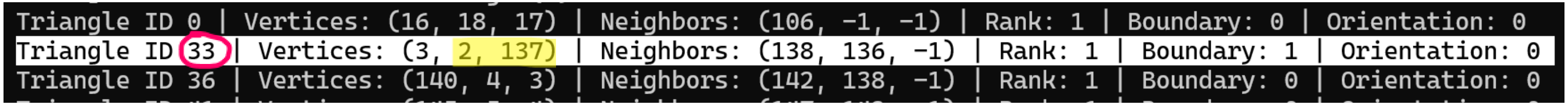
Close look at Parallel Output

Triangle 136 owned by processor 0

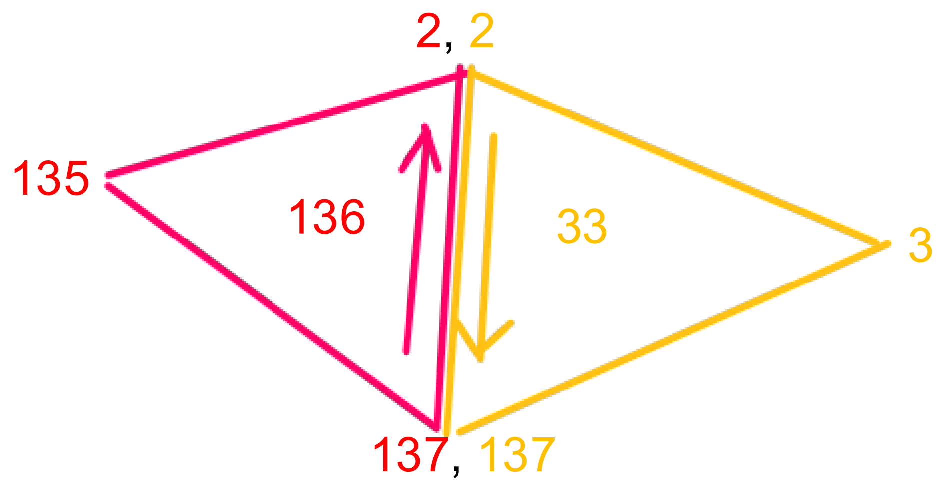


```
 Triangle ID 135 | Vertices: (134, 32, 136) | Neighbors: (234, 32, 30) | Rank: 0 | Boundary: 0 | Orientation: 0
 Triangle ID 136 | Vertices: (137, 2, 135) | Neighbors: (33, 235, 31) | Rank: 0 | Boundary: 1 | Orientation: 0
 Triangle ID 137 | Vertices: (139, 136, 31) | Neighbors: (236, 35, 32) | Rank: 0 | Boundary: 0 | Orientation: 0
```

Triangle 33 owned by processor 1

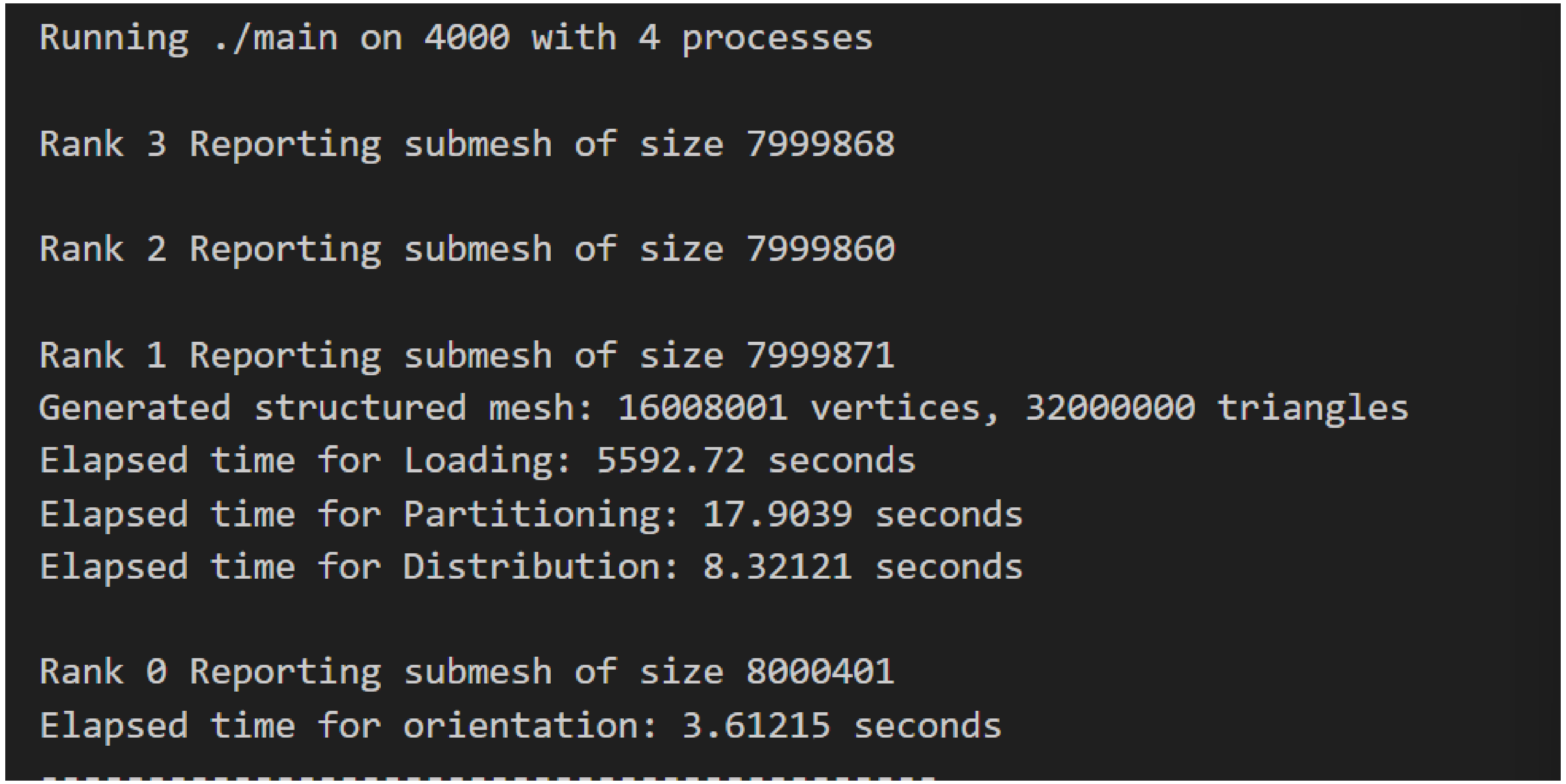


```
Triangle ID 0 | Vertices: (16, 18, 17) | Neighbors: (106, -1, -1) | Rank: 1 | Boundary: 0 | Orientation: 0
Triangle ID 33 | Vertices: (3, 2, 137) | Neighbors: (138, 136, -1) | Rank: 1 | Boundary: 1 | Orientation: 0
Triangle ID 36 | Vertices: (140, 4, 3) | Neighbors: (142, 138, -1) | Rank: 1 | Boundary: 0 | Orientation: 0
```



University at Buffalo The State University of New York

Comments on Triangle Generation



```
Running ./main on 4000 with 4 processes

Rank 3 Reporting submesh of size 7999868

Rank 2 Reporting submesh of size 7999860

Rank 1 Reporting submesh of size 7999871
Generated structured mesh: 16008001 vertices, 32000000 triangles
Elapsed time for Loading: 5592.72 seconds
Elapsed time for Partitioning: 17.9039 seconds
Elapsed time for Distribution: 8.32121 seconds

Rank 0 Reporting submesh of size 8000401
Elapsed time for orientation: 3.61215 seconds
```

Benchmark results – How well do we scale?

Triangle Count	2 million	4 million	8 million	16 million	32 million
Processor Count	Orientation Time (Seconds)				
2	0.339	0.734	1.604	3.139	6.711
4	0.165	0.407	0.850	1.778	3.612
8	0.097	0.221	0.427	0.919	2.016
16	0.058	0.122	0.268	0.489	1.995
32	0.069	0.133	0.252	0.397	0.949

Triangle Count	2 million	4 million	8 million	16 million	32 million
Processor Count	Speedup (T_2/T_P)				
2	-	-	-	-	-
4	2.06	1.81	1.89	1.77	1.86
8	3.49	3.33	3.76	3.42	3.33
16	5.84	6.00	5.99	6.42	3.36
32	4.93	5.52	6.38	7.91	7.07

University at Buffalo The State University of New York

Triangle Count	2 million	4 million	8 million	16 million	32 million
Processor Count	Efficiency (Speedup/P)				
2	-	-	-	-	-
4	0.51	0.45	0.47	0.44	0.46
8	0.44	0.42	0.47	0.43	0.42
16	0.37	0.38	0.37	0.40	0.21
32	0.15	0.17	0.20	0.25	0.22

That is all, Thank You

Questions?