1) For each vertex of the mesh calculate:
   - Connected Components
   - Number of labels of Connected Components

2) Test

   if Connected.Component.lenght > Number of labels of Con.Comp
   Vertex.manifoldness = false

   else
   Vertex.manifoldness = true

3) Calculate Tunnel to achieve manifoldness

Both triangular mesh and thetrahedral mesh derive from abstract_mesh.h, where my function is.
I can correctly detect non_manifoldness but I cannot cut edges inside of abstract_mesh.cpp

---

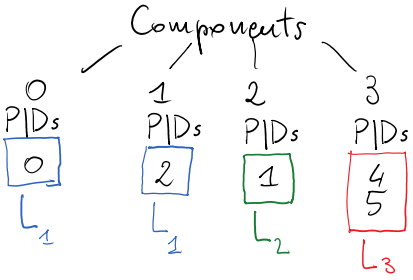
Steps Description:

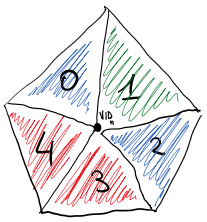
1) Mesh VID:

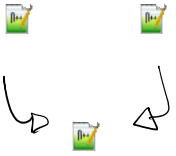
Given a VID we explore every single poly adjecent to it

If current poly hasn't been visited yet, add its label to counter

Exploring is done through a Breadth-Search-First algorithm







L_1 L_1 L_2 L_3

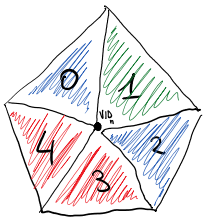
2) Testing

Components.lenght → 4 $\{0, 1, 2, 3\}$

Labels.lenght → 3 $\{L_1, L_2, L_3\}$

$4 > 3$

VID_n is NOT manifold →



3A) Tunnel :

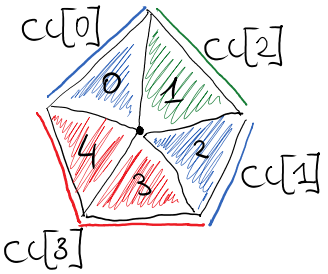
"CC"

We now have 1 vector of vectors for the Connected Components and 1 vector with all the unique Labels.

From CC we can get the label with the highest count.

For this case is L_1.

So starting from CC[0] we can try to 'travel to CC[2] and CC[3]



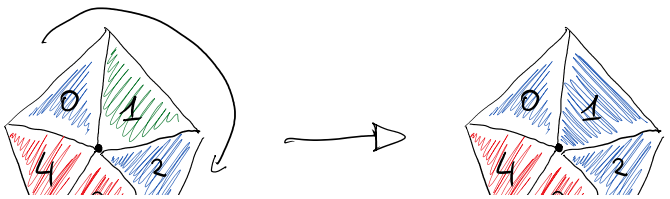
- From each poly in CC[2] can I reach another component with label L_1? ✓
- From each poly in CC[3] can I reach another component with label L_1? ✓

Both ways, we can reach another CC of the same label.

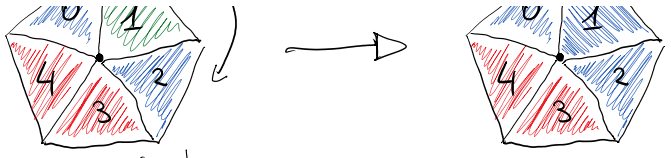
Which one is preferable?

Checking the numbers of polys in both way round we find:

CC[2].lenght < CC[3].lenght => Tunnel = CC[0] → CC[2] → CC[1]



outputs the index of



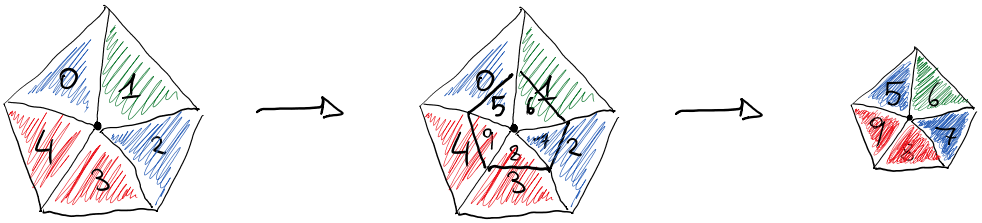
↓ outputs the index of the biggest component with the most used label around VID_n

← In this case cc[0] and cc[1] are both size 1, so the tunnel will start from cc[0]

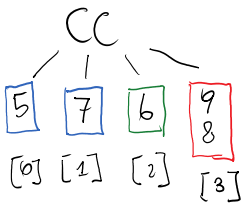
3B) Tunnel with cutting:

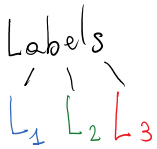
VID_n is NOT manifold.

For each EID of every PID around VID_n, if is_incident cut in half and split the poly into 2.



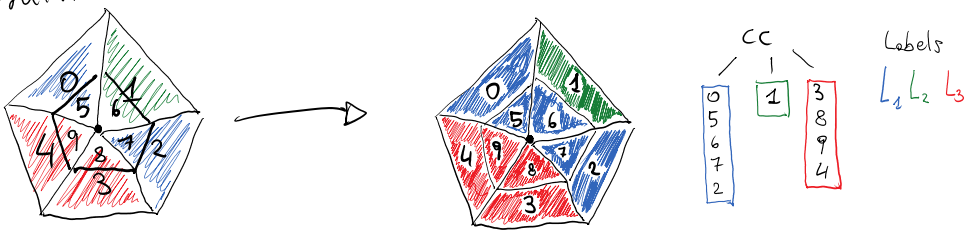
We can now search for a tunnel in the set of polys {5, 6, 7, 8, 9}





Tunnel => cc[0] → cc[2] → cc[1]

Result:



CC.lenght ~~>~~ Labels.lenght

VID_n IS manifold

What's a good order to implement it?

- Connected Components and labels vectors are calculated inside abstract_mesh.cpp, as they are general, both for tris and thetre.
- Cutting is done only with mesh specific functions, so inside the main.cpp file.

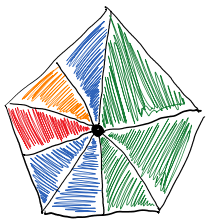
- Tunnel creation requires the same calculations as the first step, therefore can be done in abstract_mesh.cpp aswell.

Summary.

Algorithm logic is mesh type independent, but code isn't.

Also tunnel creation is almost the exact copy of the detection steps.

- Should tunnel be created for the lower count of polys or components??

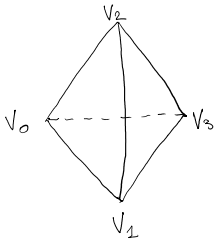


Blue cluster is NOT manifold.

Should green be relabled, as its poly count is 3,

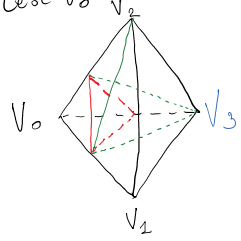
or should red and orange both be relebled as their polys sum is 2 ??

Tet Reference:

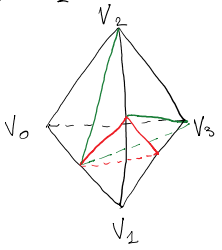


4 Possible cases.

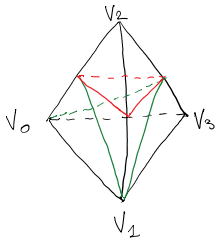
1) Case V_0 NOT manifold



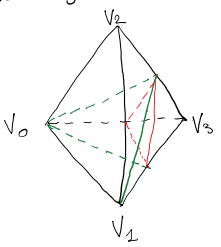
Case V_1 NOT manifold



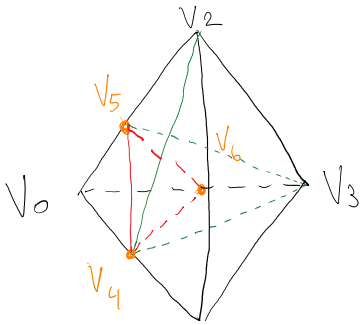
Case V_2 NOT manifold



Case V_3 NOT manifold



V_0 not Manifold



$V_4 = \frac{V_0 + V_1}{2}$

$V_5 = \frac{V_0 + V_2}{2}$

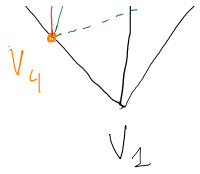
$V_6 = \frac{V_0 + V_3}{2}$

4 new polys:

1) {START V_0, V_4, V_5, V_6}

2) { V_1, V_2, V_3, V_4 }

3) { V_2, V_3, V_4, V_5 }

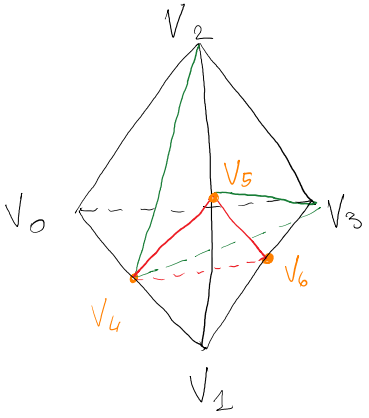


$$V_6 = \frac{V_0 + V_3}{2}$$

3) $\{V_2, V_3, V_5\}$

4) $\{V_3, V_4, V_5, \}$

V_1 NOT Manifold



$$V_4 = \frac{V_1 + V_0}{2}$$

$$V_5 = \frac{V_1 + V_2}{2}$$

$$V_6 = \frac{V_1 + V_3}{2}$$

4 new polys:

START

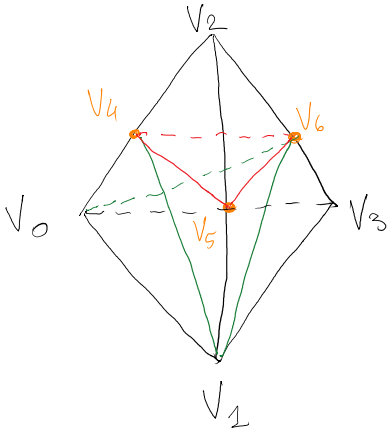
1) $\{V_1, V_4, V_5, V_6\}$

2) $\{V_0, V_2, V_3, V_4\}$

3) $\{V_2, V_3, V_4, V_5\}$

4) $\{V_3, V_4, V_5, V_6\}$

V_2 not Manifold



$$V_4 = \frac{V_2 + V_0}{2}$$

$$V_5 = \frac{V_2 + V_1}{2}$$

$$V_6 = \frac{V_2 + V_3}{2}$$

Start

4 new polys:

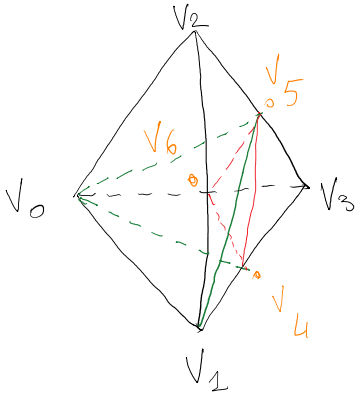
1) $\{V_2, V_4, V_5, V_6\}$

2) $\{V_0, V_1, V_4, V_6\}$

3) $\{V_1, V_4, V_5, V_6\}$

4) $\{V_0, V_1, V_3, V_6\}$

V_3 not Manifold



$$V_4 = \frac{V_3 + V_1}{2}$$

$$V_5 = \frac{V_3 + V_2}{2}$$

$$V_6 = \frac{V_3 + V_0}{2}$$

4 new polys

START

1) $\{V_3, V_4, V_5, V_6\}$

2) $\{V_0, V_1, V_4, V_5\}$

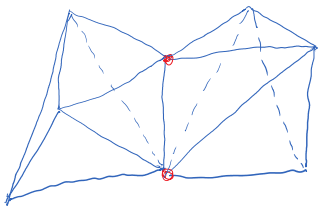
3) $\{V_0, V_4, V_5, V_6\}$

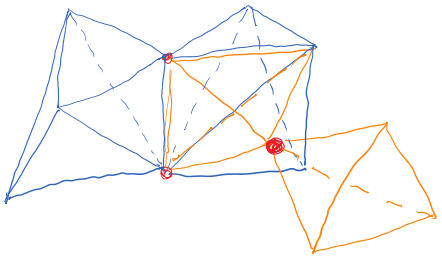
4) $\{V_0, V_1, V_2, V_5\}$

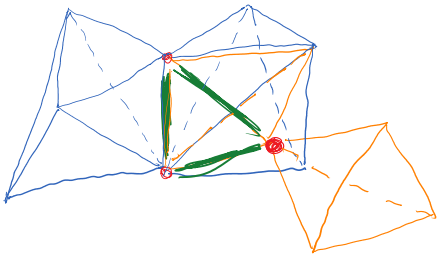
Algorithm edge case: 3 VIDs on 2 connecte Edges.

The edges are of different polys

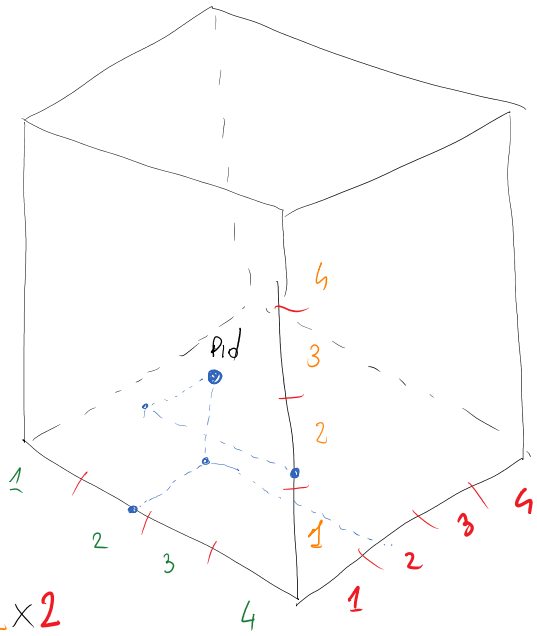
4 blue poly, connecte by 1 Edge.







BBox (mesh) →



label_x = 2

label_y = 2

label_z = 2

poly_delete(pid) = 2×2×2