

Algorithm Logic

1) For each vertex of the mesh calculate :

- Connected Components

- Number of Labels of Connected Components

2) Test

```
if Connected Component.height > Number of Labels of Con. Comp
    vertex.manifoldness = false
```

else

```
    vertex.manifoldness = true
```

3) Calculate Tunnel to achieve manifoldness

Both triangular mesh and the rehedrel 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

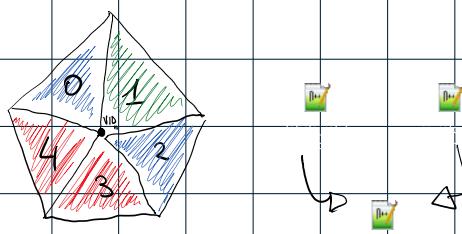
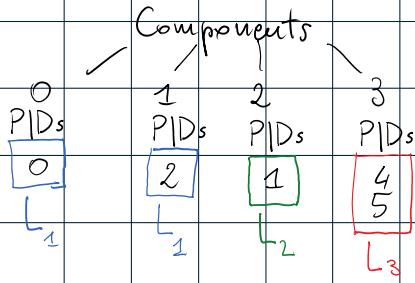
Steps Description:

1) Mesh VID:

Given a VID we explore every single poly adjacent 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 $\underline{L_3}$

~~VID_n~~

4D

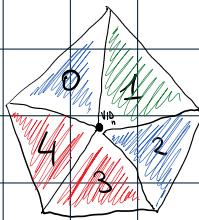
2) Testing

Components.length $\rightarrow 4 \quad \{0, 1, 2, 3\}$

Labels.length $\rightarrow 3 \quad \{L_1, L_2, L_3\}$

$$4 > 3$$

VID_n is NOT manifold



3) Tunnel:

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 .

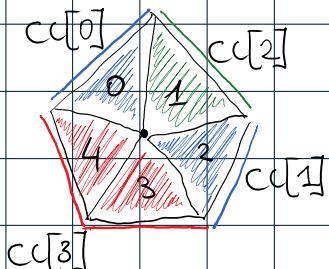
So starting from $CC[0]$ we can try

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 ? ✓

'CC'

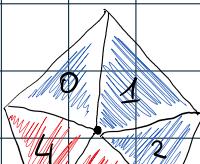


Both ways, we can reach another CC of the same label.

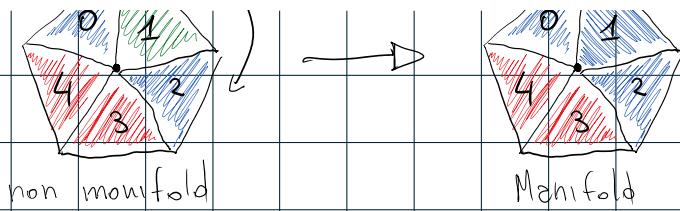
Which one is preferable?

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

$CC[2].length < CC[3].length \Rightarrow$ Tunnel = $CC[0] \rightarrow CC[2] \rightarrow CC[1]$



↓
outputs the index of



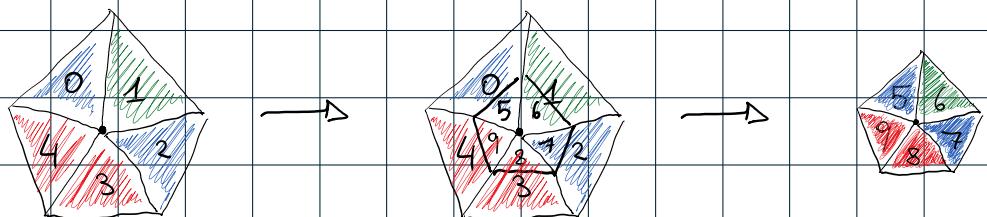
↑
outputs the index of
the biggest component
with the most used
label around VID_n

Manifold

3B) Tunnel with cutting:

VID_n is NOT manifold.

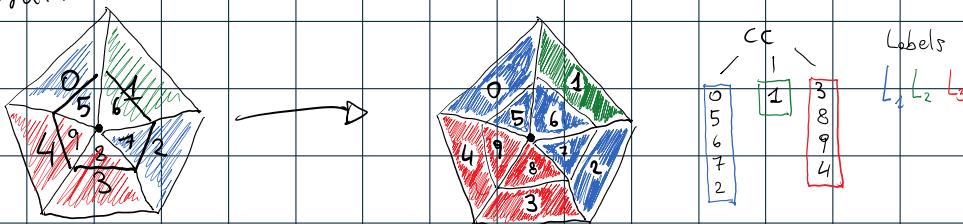
For each EID of every PID around VID_n , if is incident cut in half and split the poly into 2.



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

CC	Labels	Tunnel
$\begin{matrix} 1 & 1 & \backslash \\ \boxed{5} & \boxed{7} & \boxed{6} \\ [0] & [1] & [2] \end{matrix}$	$\begin{matrix} 1 & 1 \\ L_1 & L_2 & L_3 \end{matrix}$	$\rightarrow CC[0] \rightarrow CC[2] \rightarrow CC[1]$

Result:



$CC.length > Labels.length$

VID_n is non-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 thets.
- Cutting is done only with mesh specific functions, so inside the main.cpp file.

- Tunnel creation requires the same calculation as the first step, therefore can be done in `abstract_mesh.cpp` as well.

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 relabeled, as its polycount is 3,

or should red and orange both be relabeled as their polycount is 2 ??

