Planning

Start: 23 January
Deadline: 30 January
Time-span: 8 days including 30 January
Time available: 2 hours/day * 8 days = 16 hours total

EDIT: I ended up having a lot of fun on this problem and ended up working 40+- hours on it.

Approach

---

1. Work 1:30 hours a day on the problem and 30 minutes on writing my write-up afterwards.
2. Setup a basic Three.js test scene based on my Three.js Typescript GitHub template I've made previously to easily get started with new Three.js projects.
3. Try to solve the problem of capping clipped geometry, making sure to test potential solutions on the given 3D models.
4. If I'm able to solve the problem, start working on integrating the solution in the speckle viewer and using their sandbox project as a client for the viewer library.
5. During development keep a devlog describing my attempts at solving the problem.

Log

@Jan 23
Work duration: 6:30 hours

I got a basic scene setup with the house model and basic stencil clipping. I can see the problem Alex was having when the terrain was enabled. That is messing with the stencil clipping.

_
I can see the method working for basic geometry. For instance if the method was only applied to the building and not the surrounding terrain and trees etc. But speckle wants it to work on all geometries: "cannot assume anything about the scene".

_
I will research stencils a bit more tomorrow before ditching it. An alternative approach is to check every mesh triangle <-> clipping plane intersection and saving those **edges** and generate geometry out of them. This is probably how Autodesk does it because it only generates edges when you stopped moving the clipping plane.

_
Possible solutions:

Solution 1:
Filter out non-solid meshes before using the stencil capping method.

Solution 2:
Find closed loops in meshes intersecting with the clipping plane and generate real geometry.
https://www.scitepress.org/papers/2017/60972/60972.pdf

@Jan 24
Work duration: forgot to track

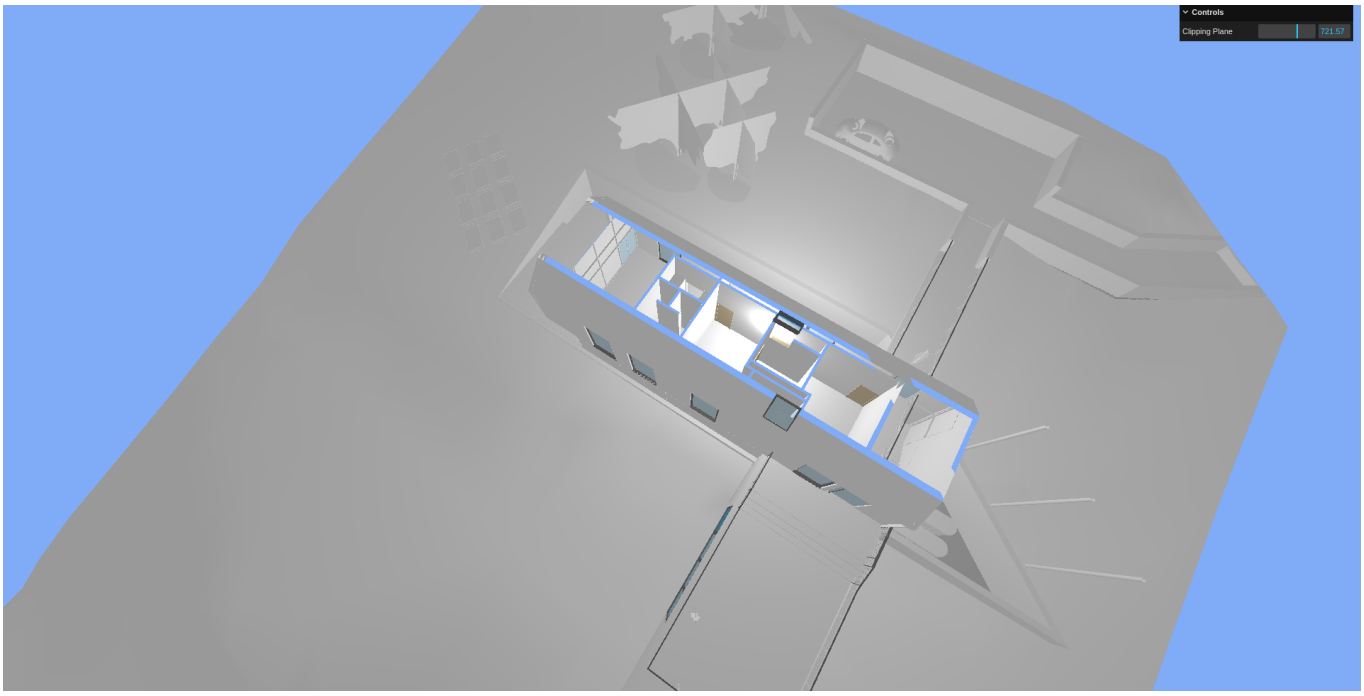
Found out about manifold geometries from the paper keywords last night. Made me think about stencil clipping and other techniques only working on manifold geometries. A plane and leaves on a tree etc are non-manifold geometries and that's why stencil clipping doesn't work correctly. My solution is to filter out all non-manifold meshes and only use stencil clipping on the remaining manifold meshes.

_
I learned about DCEL data structures and about how to determine if a mesh is manifold or not.

_
Can I make DCEL of a non-manifold mesh? Try it with three.js DCEL addon.
Oke DCEL is always of a manifold mesh

Lecture 10: Meshes and Manifolds (CMU 15-462/662

_
Got pretty good results now by not using non-manifolds for the stencil pass.



_
My pipes in house.fbx don't work because there are 2 pipes on top of each other. Filter out vertices that are in the same location within epsilon so we don't have double faces messing with the stencil clipping. Uhm now that I think about it, it should still work tho. Do they have the correct materials? Filter out a pipe mesh and inspect it.

@Jan 25
Work duration: 12:00

Fixed pipes under building not capping. Was because I added back/front stencil meshes to the scene root instead of the mesh parent. So the position and rotation wasn't applied to the stencil meshes and thus failing. I still have a problem that the windmills and some other tubes like the fire place aren't capped. It's because the way I check if a mesh is manifold. Those meshes contain geometry that contains gaps for instance a block on a pillar. They aren't merged together to form a continuous mesh and thus not manifold. But they LOOK pretty much solid and the stencil clipping can work on those meshes just fine. How can I make a good check for those meshes? A mesh should not contain holes but it can contains gaps. I should do some more research and look at how other libraries do it. I haven't found a good library yet although I did find:

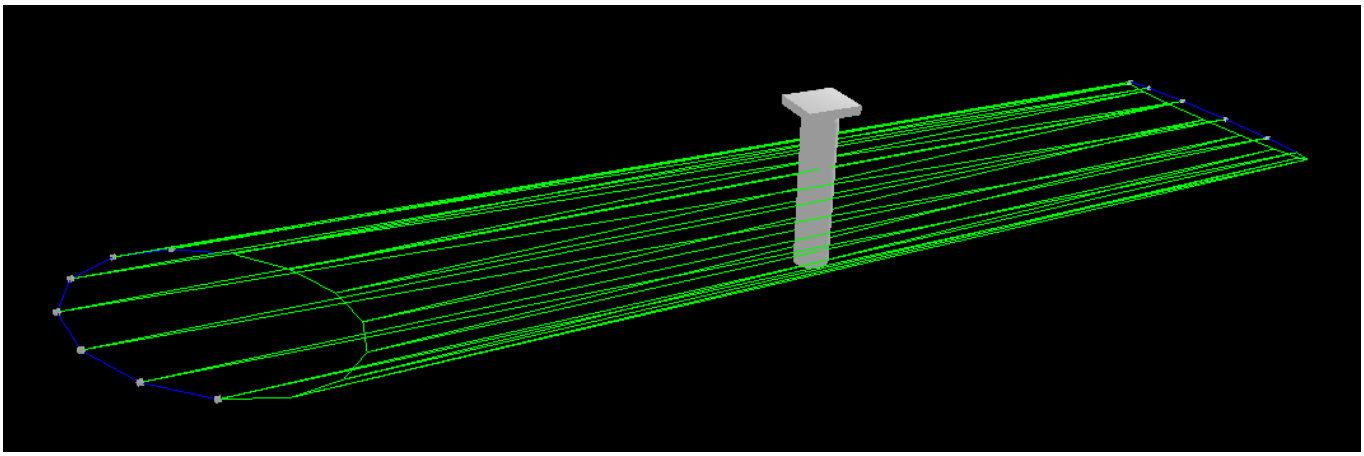
https://observablehq.com/@esperanc/mesh-data-structures-traversal
https://github.com/GeometryCollective/geometry-processing-js

How exactly is my manifold test failing on meshes with gaps? I check if all edges have twins... should it not still pass? Maybe the DCEL library I use just sucks or the indices I get from `BufferGeometryUtils.mergevertices` aren't good.

_
Oke gaps in a mesh are technically disconnected vertices/edges and thus non-manifold.
https://stackoverflow.com/questions/12968478/how-to-detect-a-hole-in-the-triangular-mesh

_
Tried to improve the manifold detection code... having problems with detecting planes. It should be easy... get all boundary edges, see if they loop. And if the mesh's boundary edges all form 1 loop in let's say CCW order, it's a mesh. This way I can ditch all the planes but keep all meshes with volume or even holes if I want to. But the terrain mesh has boundary 3 edges...

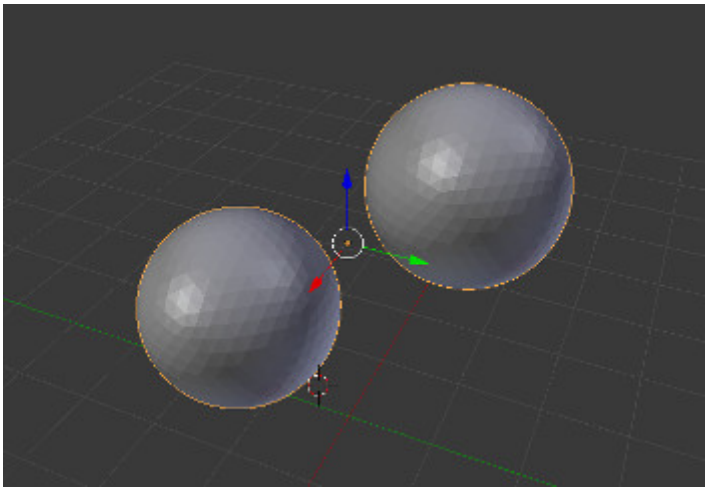
_
Took a quick break and went back to the basics: drawing debug information to see how things work. Tomorrow I will give it another shot (filter out geometries based on boundary edges).



_
My plan is to keep track of the boundaries of each mesh. CCW bounary is outside CW boundary is a hole. 1 CCW boundary and nothing else means a plane. Multiple CCW boundaries means a non-manifold solid/hollow mesh. And if they also have CW boundaries that means they have holes. I can then decide on which meshes I want to apply the stencil capping and filter out the planes because they don't work with stencil capping. Non-manifold solid/hollow meshes do work though so that's the reason I go through the trouble of doing all this. So it works with them too as well as manifold meshes. I hope this works out. Otherwise I can always implement the https://www.scitepress.org/papers/2017/60972/60972.pdf paper. Not that much trouble anyway as they also do loop (boundary) construction but they use it to generate meshes.

@Jan 26
Work duration 00:30

https://github.com/rlguy/Blender-FLIP-Fluids/wiki/Manifold-Meshes (picture below) tells me two joined spheres with a gap in between them in the same mesh is a still a manifold mesh. Does that mean that my pillar + block in the same mesh is supposed to be manifold as well? My DCEL library tells me it isn't. Maybe I should implement my own DCEL class, from taking a look at the library it isn't that much work. Wikipedia says a DCEL can hold multiple disjoint graphs. In the DCEL I could keep track of all the outer boundaries and for each also keep track of the inner boundaries (holes).



@Jan 27
Work duration: 9:00

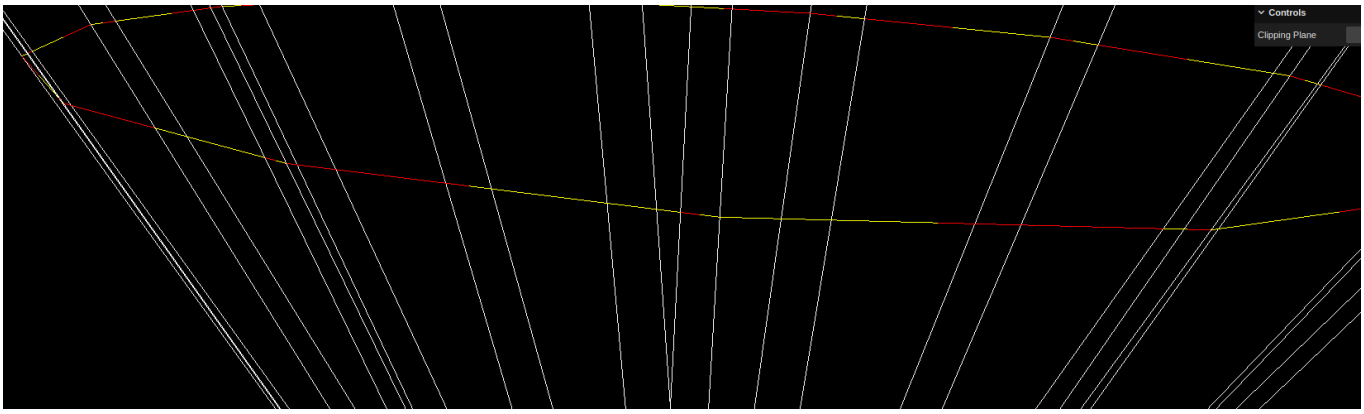
I refined my code to check for loops in the boundaries given by the DCEL library. It seems it gives me boundary of meshes when there shouldn't be a boundary. So either the mesh is wrong or the DCEL code has some bugs in it. Maybe I should implement my own DCEL data structure? Then at least I know exactly what it is doing. Also, I thought the DCEL library didn't handle gaps, but it seems to handle them just fine. If I only apply the stencil capping on meshes without any boundaries it works but skips some meshes (as I just described, buggy mesh or DCEL?). And it doesn't work on things like hollow meshes (because it has boundaries). Also the blizzard scene doesn´t work because it's full of non-solid geometry it seems. If I really want it to work on all kinds of meshes I have to implement the method of the research paper mentioned before: for every face -> check intersection with plane -> collect intersection segments -> construct loops -> for every closed loop generate a cap geometry. Then depending on the CW or CCW direction of the loop it's a boundary or hole. Then you can use a triangulation algorithm like earcut or constrained delaunay to generate the cap geometries. Constrained delaunay is

kind of slow tho from previous experience (I used https://github.com/r3mi/poly2tri.js before, the only one I was able to find that worked). Earcut is supported by Three.js itself.

_
19:36
Found out that `BufferGeometryUtils.mergeVertices(mesh.geometry);` doesn't work with meshes that have gaps. I used it to create indices for the mesh but that messed up the indices. I used this broken mesh to init the DCEL data structure. No wonder I was getting false positives when checking from boundaries. I figured this one by searching for face/plane intersections and rendering the resulting segments. It looks good on the original non-indexed mesh but not the indexed mesh.

EDIT: **Wrong!** I messed up the way I iterate over indices. Now I loop correctly over indexes meshes and mergeVertices is NOT bugged. Good to know.

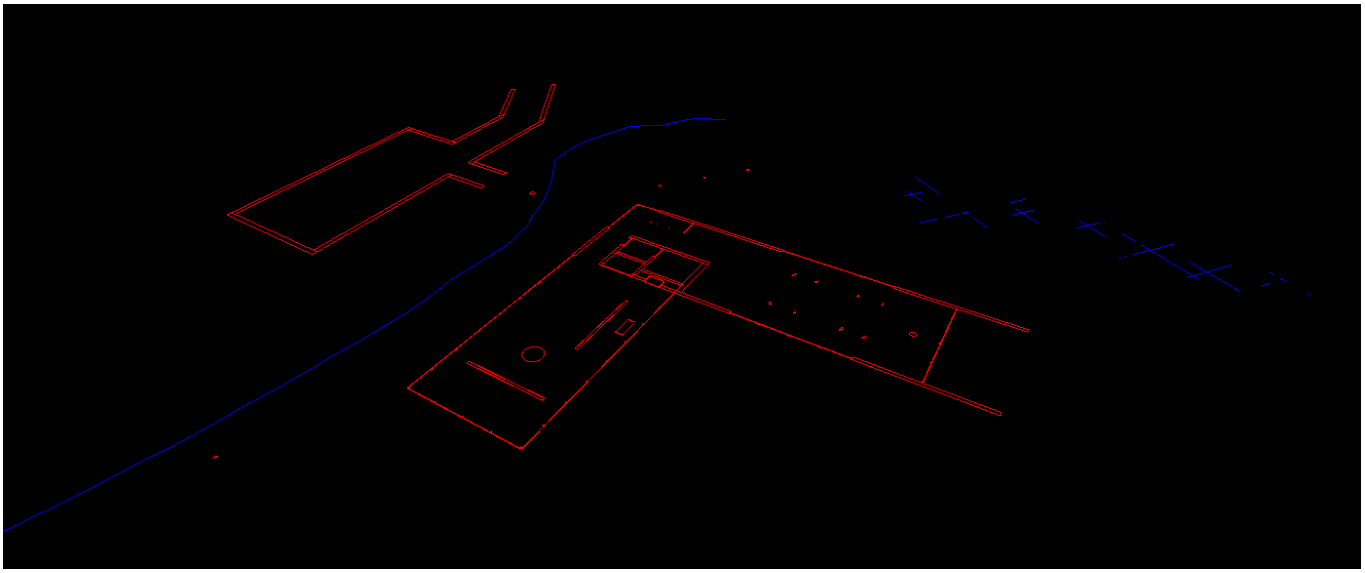
_
23:02
I started implementing the technique from the paper. I noticed my code to detect loops in segments is bugged (some segments face the wrong way). I am working on fixing it but I don't know the right technique for it. The line segments should be filled in the right order and currently that's not happening (see picture bellow, it should alternate from yellow to red but sometimes it does not). I am asking the question in the Graphics Programming discord on how to solve it / keywords I can google to find a solution. I know it's something pretty basic and it seems like I've done something like this before when I wrote a software rasterizer. The segment should be inverted depending on certain criteria. But what criteria exactly?

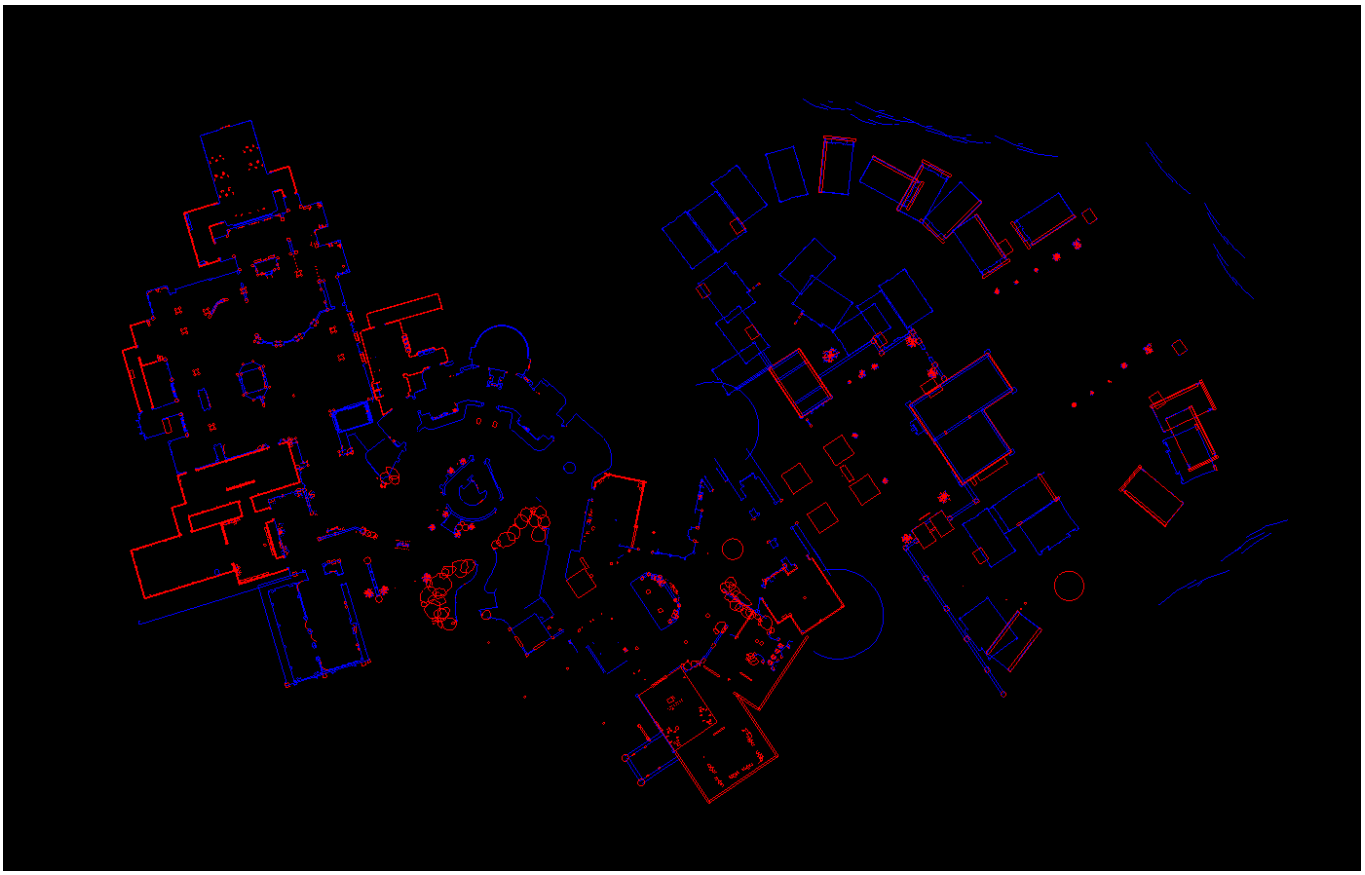


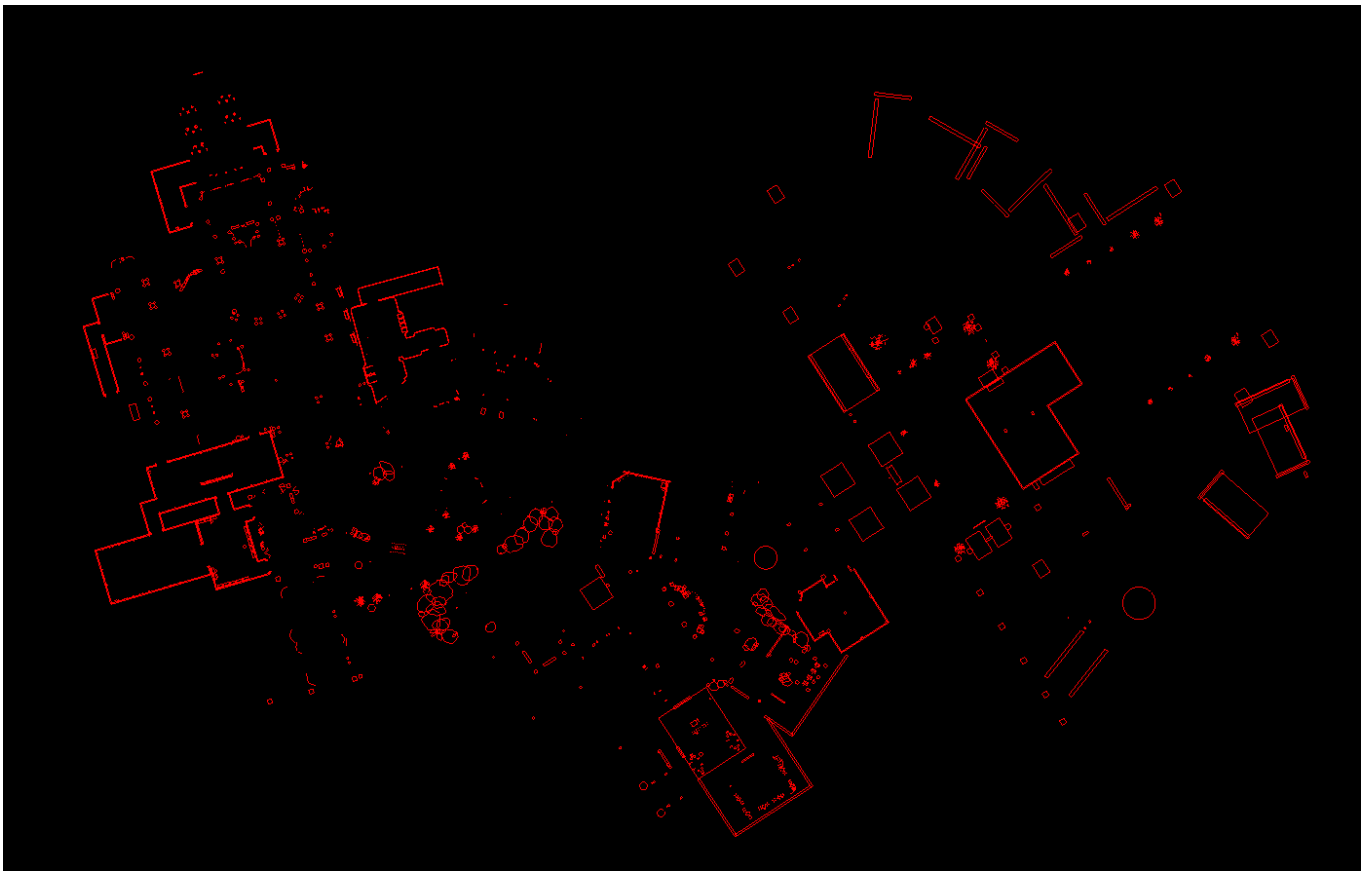
I am now reading the paper by McGuire, 2011 referenced by the previous paper on how to clip triangles. Maybe it has some info on how I should tackle the problem.

EDIT: Yeah the paper provided the answer. I had to make sure to rotate the triangle CCW or CW depending on which vertices was above the clipping plane to get the segments going in the right direction.

_
23:57
I freaking did it! Everything happens on the CPU mind you so it's rather slow. But it successfully generates segments and does the loop construction. In the image below the loops are displayed in red and the dead-end lines are displayed in blue. Next up is checking if a loop is a border or a hole and then triangulation. After that I need to move the clipping code to a shader to improve performance. Depending on the speed I can also use the information to feed the stencil technique (on what meshes to apply stencil capping).







@Jan 28
Work duration: 5:00

The challenge for today is figuring out which closed loops of a mesh are the contour loops and which ones are the holes loops. So I can use that information for triangulation. Right now I can only detect loops but I don't know which are which. Currently I don't know how to do it and the paper refers to the original paper from 1990 describing it but it's rather cryptic. I know the https://www.scitepress.org/papers/2017/60972/60972.pdf paper mentions a `Plane Sweep` and `Sweep Line` algorithm though. So I will first learn about how that works and go from there. I am watching some lectures about it on YouTube:

Sweep-Line Algorithm for Line Segment Intersection (2/5) | Computational Geometry - Lecture 02
https://www.youtube.com/watch?v=qkhUNzCGDt0

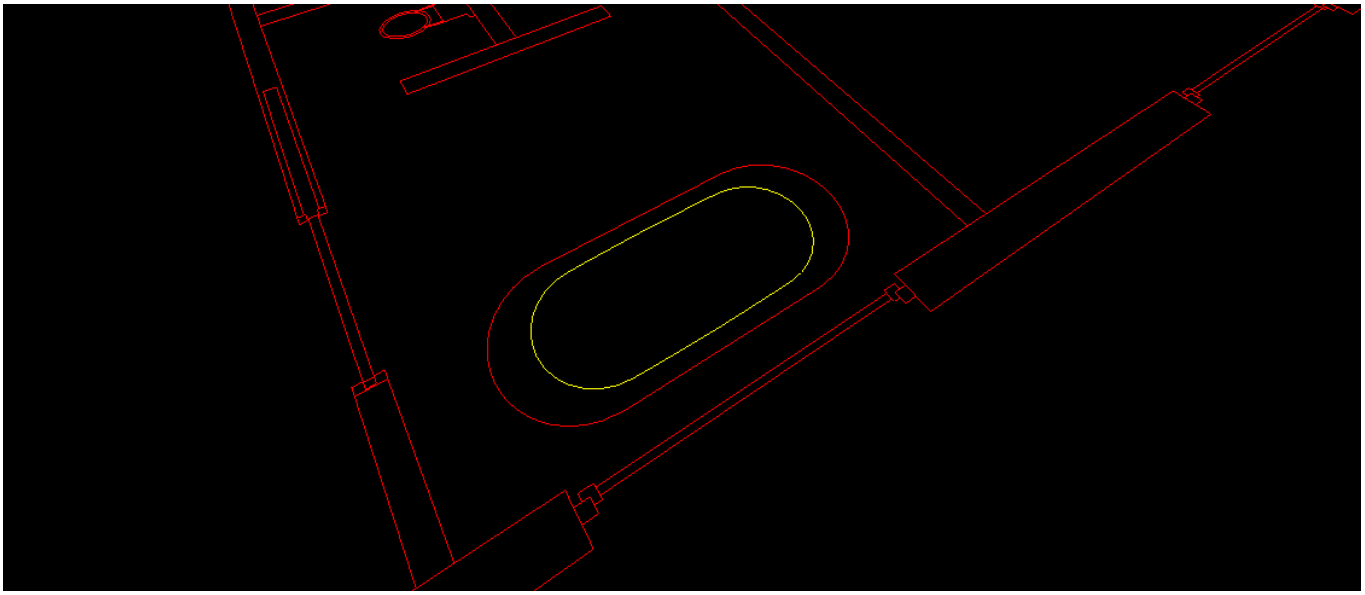
Plane Sweep Algorithm for finding Line Segment Intersections
https://www.youtube.com/watch?v=0mwScBSTHB4

–
So I found some information about how to test if a polygon is inside another polygon here. It tells me a simple way to test it and that it CAN be improved using the sweep line algorithm to

speed things up. So to start out with and get something working I will start implementing the brute force method:

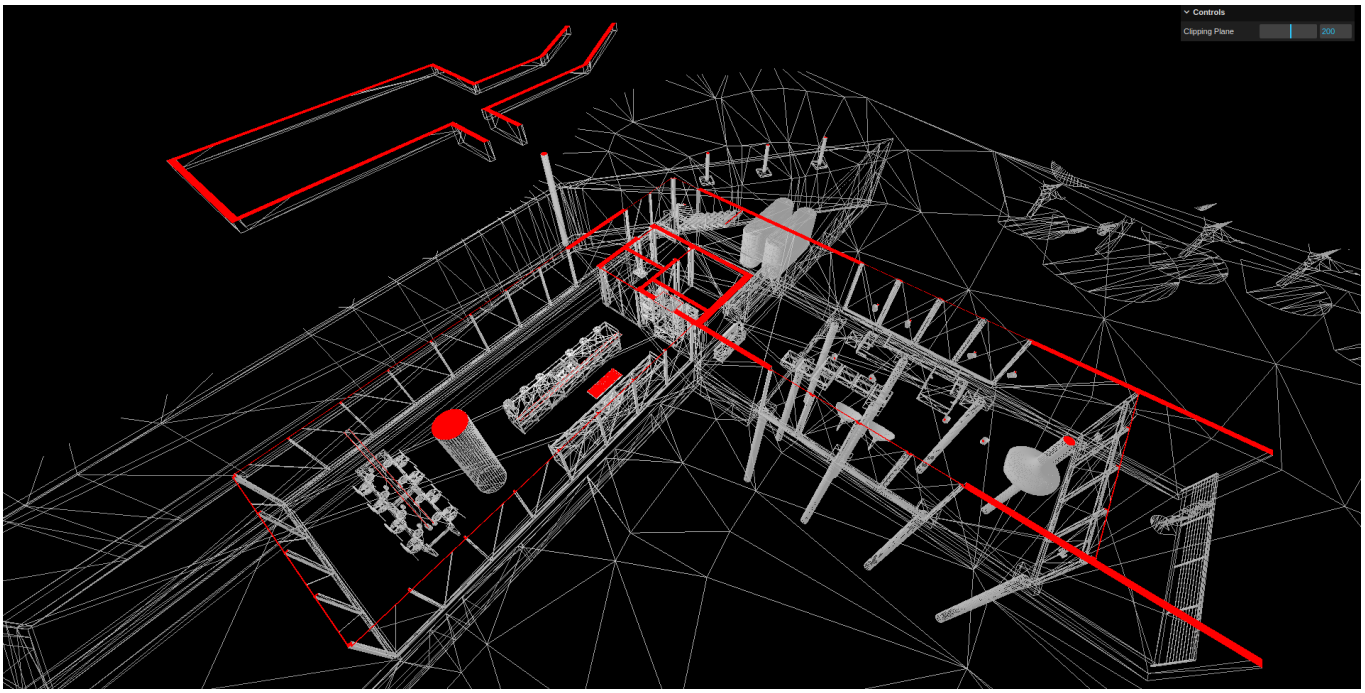
"Perform line intersection tests for each pair of lines, one from each polygon. If no pairs of lines intersect and one of the line end-points of polygon A is inside polygon B, then A is entirely inside B."

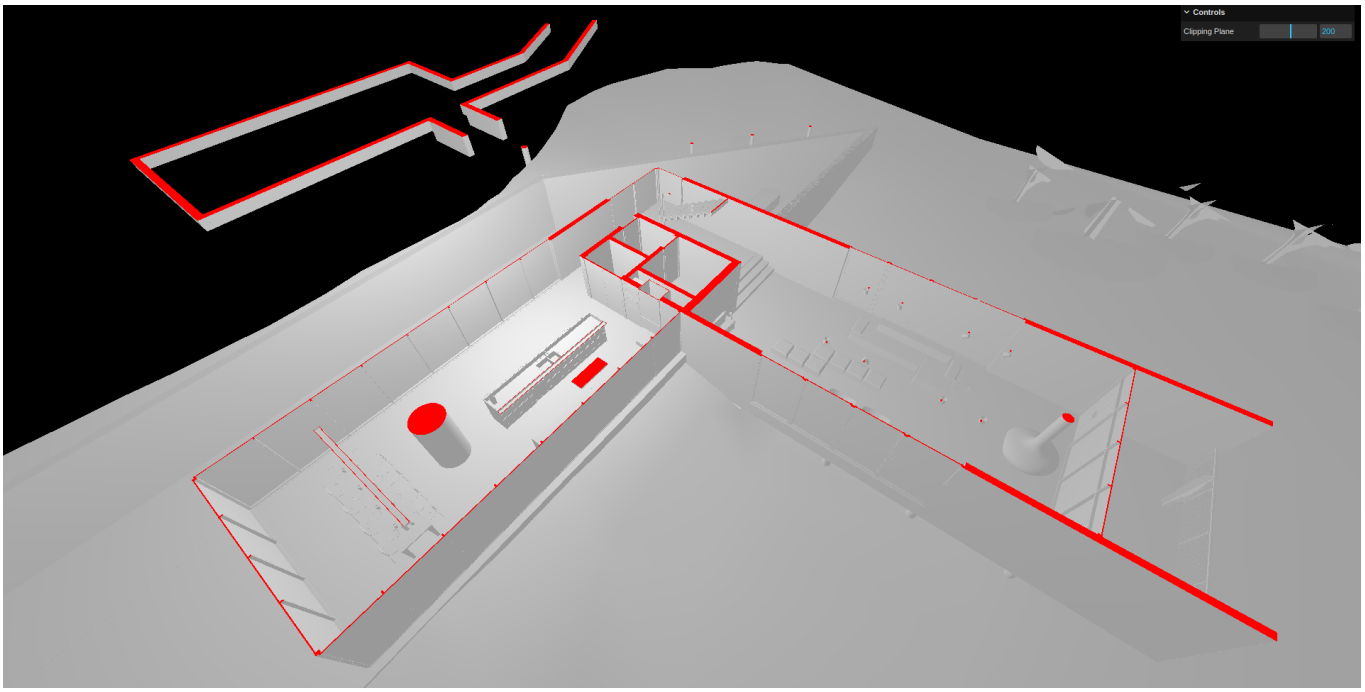
_
The brute-force approach seems to work! Now I need to create a nesting structure in the code for loops contained in loops contained in loops etc. Then the first loop is the boundary, one level deeper is a hole, one level deeper again a boundary etc.



_
I implemented Earcut triangulation on loops without holes in them (for now... I am not sure I have enough time before the deadline to also implement this for loops with inner loops/holes)!

EDIT: I didn't implement the loops tree structure due to time constraints. Right now I only render meshes without holes in them. This is something to improve later on.





Demo

https://nickvanurk.com/capping/