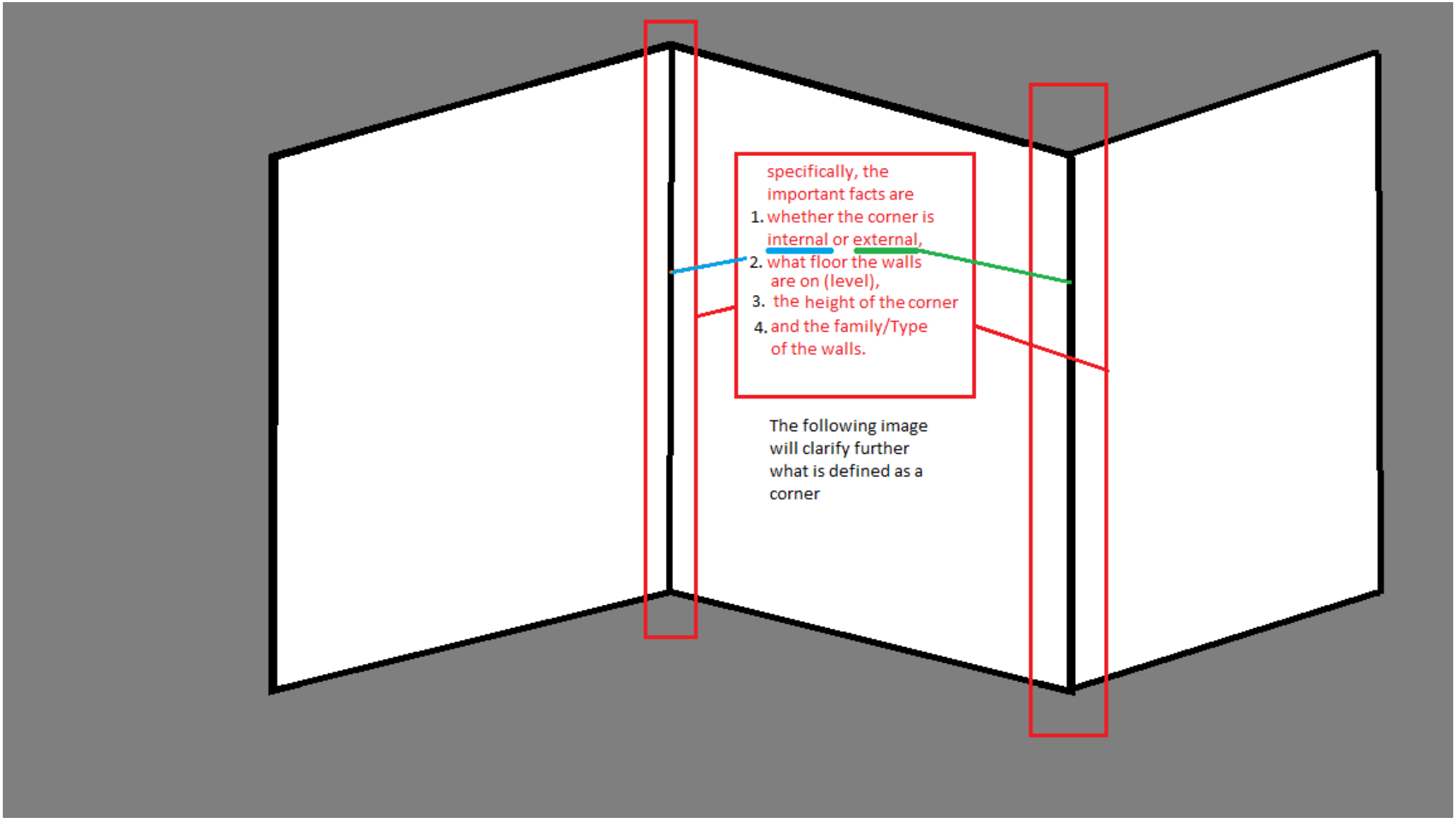
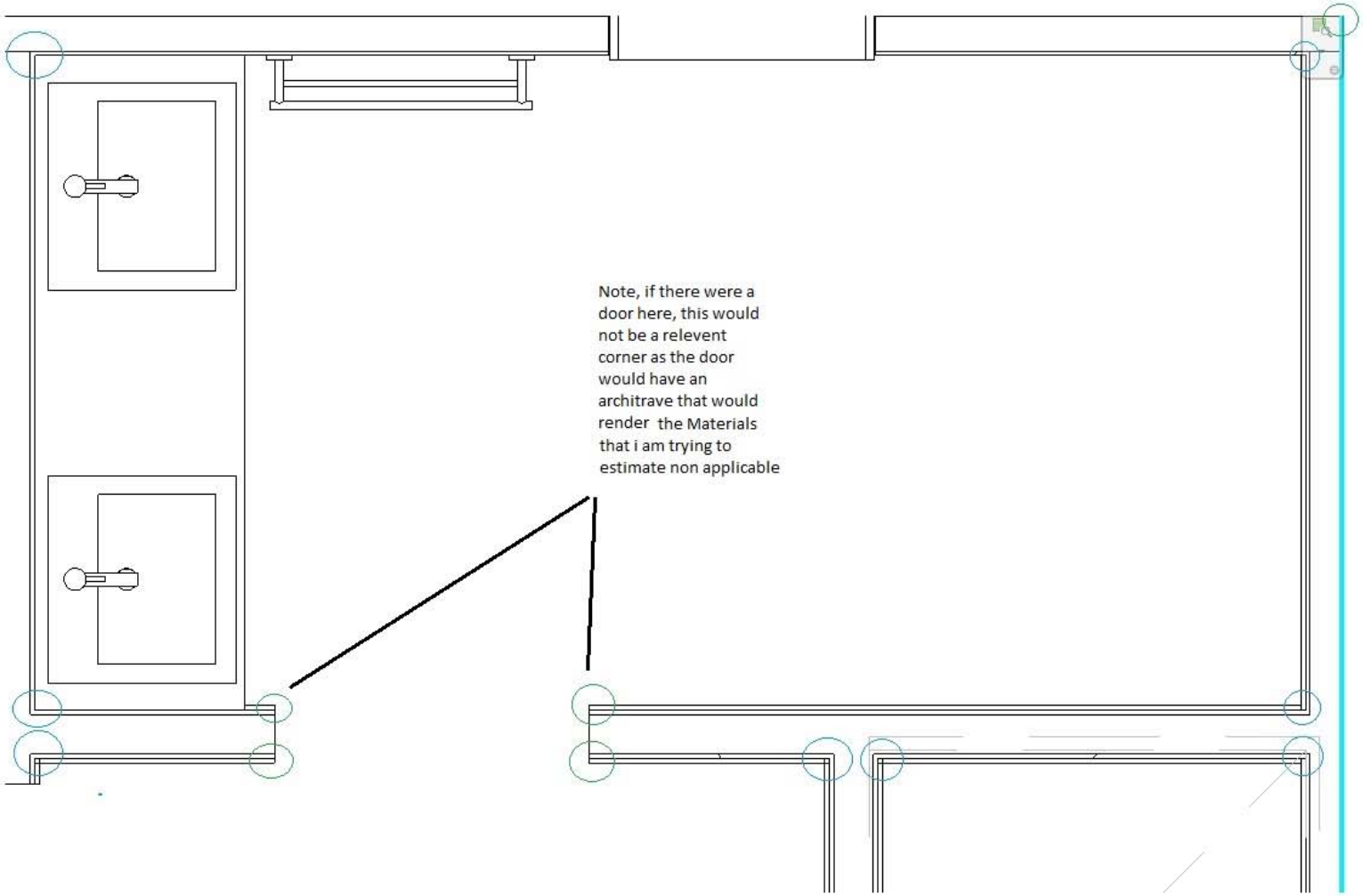
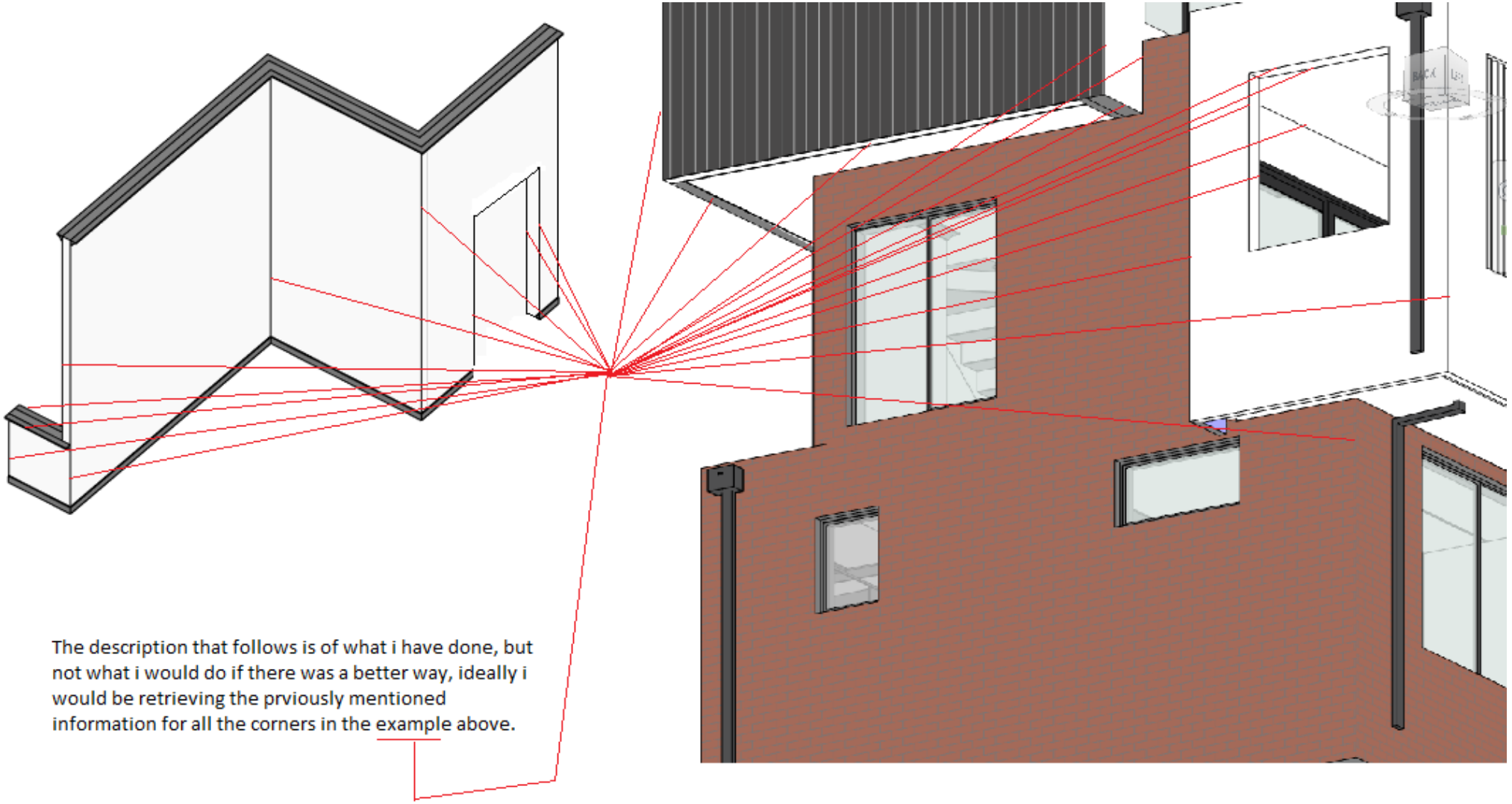
The Basic Idea is to extract from a model like this, a list of corners where cladding meets cladding, whether this be interior or exterior cladding.


specifically, the
important facts are
1. whether the corner is
internal or external,
2. what floor the walls
are on (level),
3. the height of the corner
4. and the family/Type
of the walls.
The following image
will clarify further
what is defined as a
corner

internal corner

external corner





The description that follows is of what i have done, but not what i would do if there was a better way, ideally i would be retrieving the prviously mentioned information for all the corners in the example above.

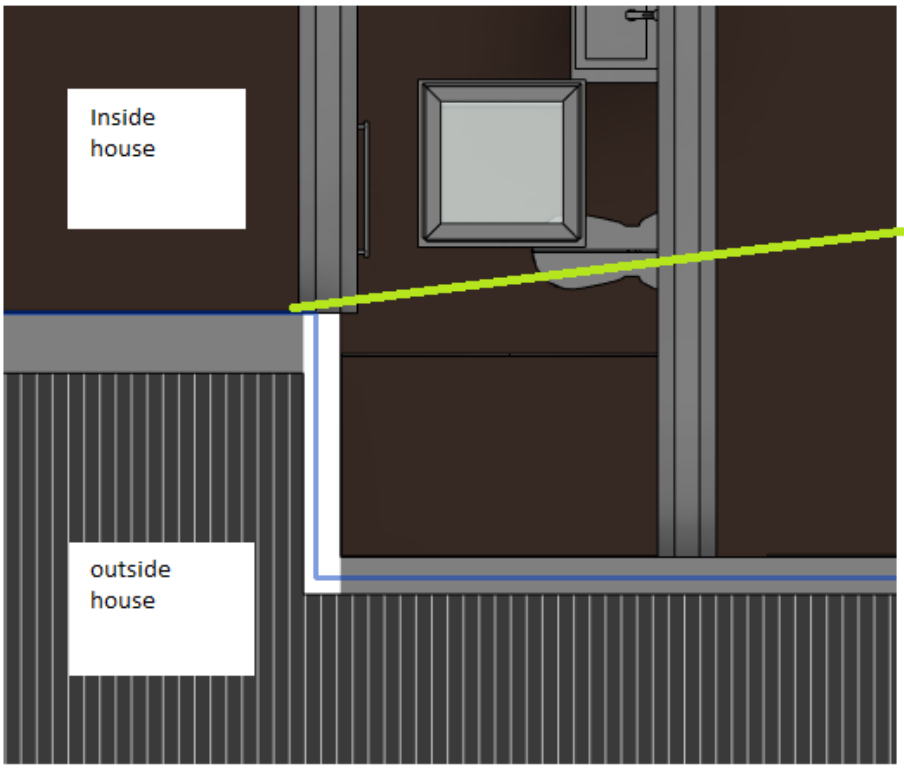
Note: this method strictly provides information relevant to internal cladding, it only stores information for external cladding to be used for later.

Description of method:

1. Take list of all walls in the model,
2. Loop Through all of these walls.
3. Disregard any walls that are parapets.

The rest of this description will be from the perspective of the current wall.

1. Use the bounding box functionality to find all intersecting elements.
2. Utilise a filter to reduce this result to a list of elements that are: Walls, Share the same base constraint as the current wall, are not dwarf walls, are not parapets.
3. (store dwarf walls for later use)
4. Loop through list of intersecting walls
5. NOTE: from here we assume that there is an intersection (The Current Intersection) between the current wall and the current intersecting wall. this intersection is either a perpendicular T intersection, a corner or a Parallel intersection. The later most of which will be ignored. We instantiate a Class of “intersection” to represent the Current intersection.
6. If the current intersection is not parallel we continue,
7. First we take the start and end points of the 2 walls and check if any of these points are almost equal, if so we take this to mean that the intersection is a corner and not a T.
8. We then check for a third wall abutting on to this intersection as this may mean there are more corners.
9. We then store all the walls as properties of the intersection object.
10. We then get the functions of the walls involved in the intersection, we do this because there is a distinct difference between a corner that involves an exterior wall and a corner that involves no exterior wall.
11. NOTE: it is assumed that any walls in the model will have internal cladding on at least one side.
12. IF CORNER: (we assume that if the corner is an internal wall meeting another internal wall, that the corner has 2 sides. This being so we only record the external corner, because this type of internal corner does not actually require any additional material to build. If the corner is internal meeting external it would have to mean that there is a third wall and would look something like this,

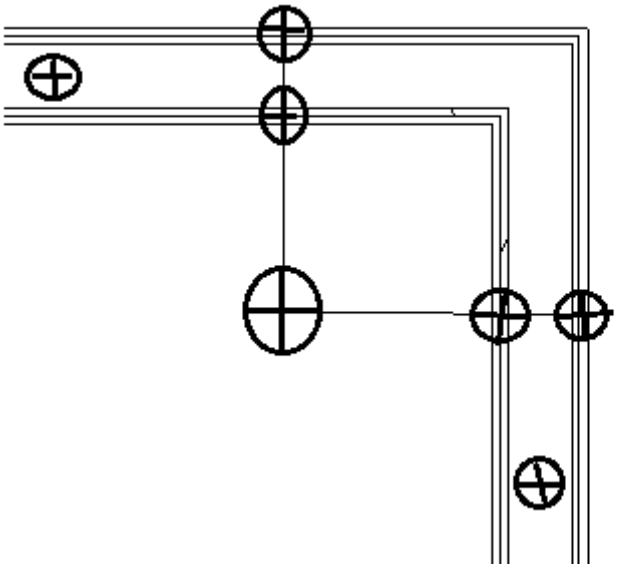


And as such only one corner is recorded. Now we know what to do with internal meeting internal, internal meeting external But what to do when the external meets the external, This is the only tricky one because we have to know whether the corner is external or internal)

UP to here every thig seems to work.

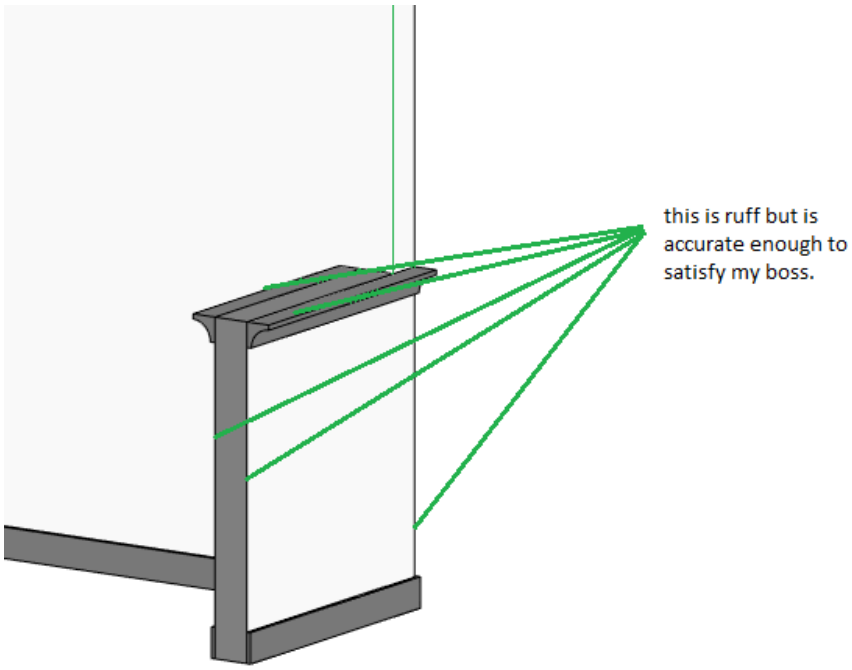
Its this next bit that doesn't work at all. I'm not sure whether it is because of units of measure not matching or something, but it is way out.

1. If the current wall is external and the intersector as well, we take the points that meet in the corner and process them one by one.
2. We find a point along the location curve of each wall that is a certain distance from the corner,
3. We get the mid point between these two points.
4. We get a point on the internal face of the current wall
5. We then get a point on the external face of the current wall
6. We measure the distance between that midpoint and the points on either side of the wall.
7. Based on which is closer we should be able to tell whether or not the corner is internal or external. The picture below describes what this should look like when there is just 2 walls.



This doesn’t work unfortunately, and it seems far too complicated to be the best way of doing things.

13. If the intersection is a T intersection we make similar assumptions as those described above based on the combination of walls involved .
14. After this is all done we have a list of intersection that are populated by a list of corners and the walls that they belong to,
15. We then remove any intersections that have duplicated themselves.
16. We then collect up a list of corners from these intersections,
17. Following that we add 5 external corners to the list for every dwarf wall in the model. Based on the logic described by image below.



So yeah, i just can’t believe that this is the best way to go.

It doesn’t record the corners made by voids in internal walls, or the corners made at the free standing ends of the walls, I couldn’t figure out how to do that.

Any way, thanks again.