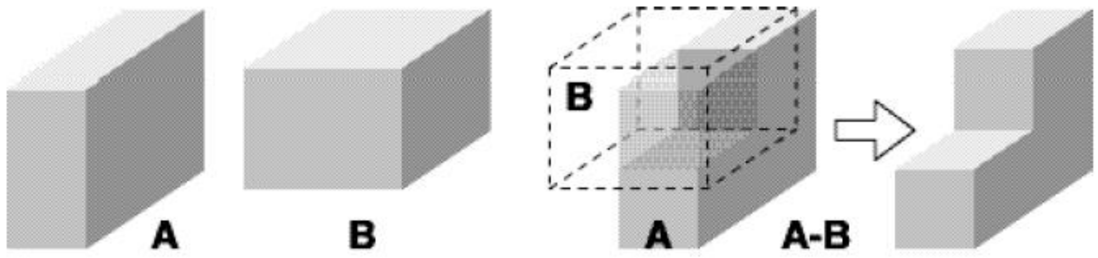
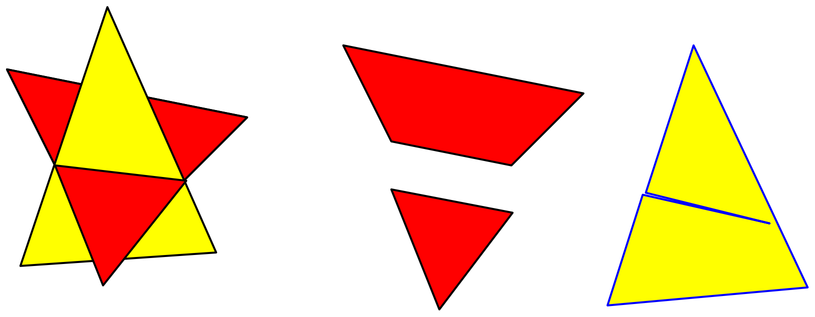
CSG (Constructive Solid Geometry) is a representation format for data that performs set operations from basic solids (primitives) as a shape.

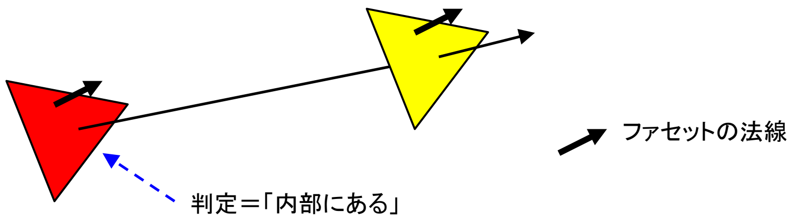


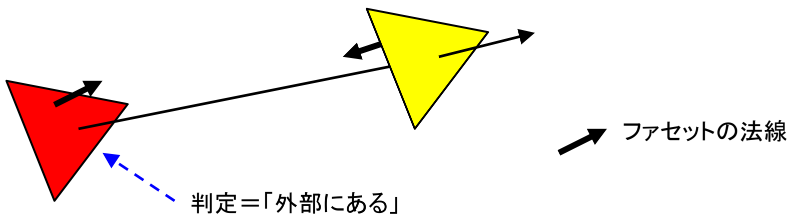
I think it's beautiful to use a Winged Edge Data structure to handle solids, but it's not possible if the phase is not always maintained. It is quite troublesome to think about calculating in floating point. Moreover, it is even more problematic that it requires a high level of code writing ability.
What's more straightforward is that you don't know about the phases, and you can't do the calculations.
So, I can simply think of the following method (well, everyone will come up with first, but...)
Split the facets of a solid (polygon) by checking the intersection of each facet (triangle) on the other solid facet.
At this point, solid A and solid B are separable.



After that, it is enough to determine whether each facet is inside or outside the solid, and select according to the set operator. This method is cheap, so it is possible to perform a set operation even if the phase of the solid is somewhat collapsed, but on the other hand, as you can see from the algorithm, the result is output as a disjointed facet set.

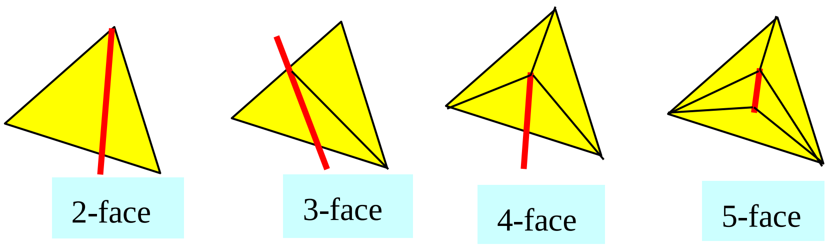
It is simple to determine whether each facet is inside or outside the solid.



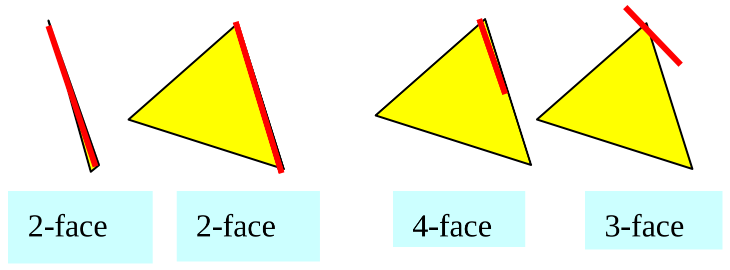


You can skip Ray in the normal direction and compare the direction with the normal of the pierced facet.
If it pierces the front, it can be judged to be "outside", and if it pierces the back, it can be judged to be inside.
Naturally, it is pierced with multiple facets, but compared to the facet that pierces the front.
That's why I skip the Ray (find a place to block the light).

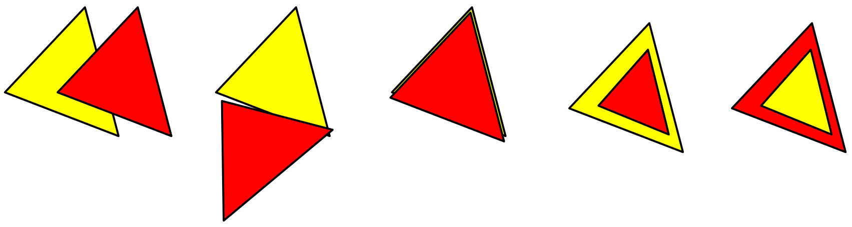
So far, so algorithmic.
It's a very simple implementation, but there are a lot of problems.
First, splitting the facets. There are only four basic patterns. This allows you to decide in advance how to split the facets (yellow).
The N-face represents how many splits are to be made.



Now, this is just a basic pattern. The troublesome pattern is as follows.



Even more troubling is the following pattern

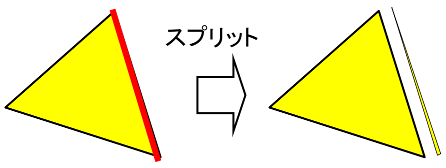


These are cases that come into contact with petali without intersecting.
However, if we focus on the intersection relationship between edges and facets, we can classify them into one of the 4+1 patterns described above. In other words, splitting is possible, but the problem is internal and external judgment.
Since it is in contact with Petari, it is not possible to skip the Ray, but it is possible to judge inside and outside because it can be checked whether it is back-to-back by looking at each other's normal vectors.

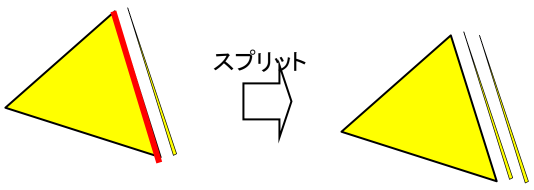
And the most troublesome thing is the threshold that determines whether it is a basic 4+1 pattern or something else. If it is exact, it can simply be divided into patterns, but for example, if the distance is 0, it is contact, otherwise it is crossing、、、 but since the calculation is done in floating point, it is impossible to make such a judgment.
In other words, if it is less than 0.0001, it is a contact, and if it is not, it is a process called crossing.
But what about 0.000100001?
I can't write about it in detail, but the ingenuity at such times determines the quality of the algorithm.

However, there are always cases that cannot be dealt with in any way. An error correction mechanism should also be included for that time. The question is, what should be an error?
The algorithm described above does not break in the implementation even if it is wrong in classifying patterns, so it returns the result with a blank face.

Therefore, while the processing was going well, we did not include an error correction mechanism, but when there was an abnormality in the output result, we investigated why it was abnormal, extracted the conditions, and added an error correction mechanism.



Normally, this completes the split, but if you check if it is still split, it will split again.



In other words, this process is an infinite loop.