# Line Folding

## General problem description and “total” kind of fold

A "line" fold is some variation of folding a figure along a line. Of course it is only one of the "bundles or a subset of layers in a bundle that gets folded, as that’s the only way to fold along a line; ie: no more than one bundle (which belong in different planes) can be folded along a line.

What we call bundle are layers of faces that occupy, theoretically, the same plane in the three-dimensional space.

The simplest way to fold along a line is folding every face of a bundle. That’s it. We are only concerned with what faces are intersected by the line and keeping the connections between faces after breaking and rotating the faces.

That kind of fold, however, is not, strictly speaking, considered by this application, as it is not possible to fold along a line, only along a segment. However manually picking the ends of a fold segment that intersects the whole bundle of layers side to side, obviously, accomplishes the same thing.

Why not have the “line fold” and have this instead? The line fold is the more common case, but as it is a special case of “segment fold” and it does not take more effort for the user to define a line than a segment, it’s preferable to create this kind of fold which also cover the more unusual needs. Defining a line would be more complicated for any kind of interface than just defining a segment, with an option for docking a fold segment end point to a side or a vertex of the figure.

While this fold seems common and very simple to understand and implement, considering every other kind common folds along a line segment in origami is unexpectedly more complicated with many corner cases to consider.

We tackle the problem by restricting and clearly specifying which other kind of folds are considered valid by the application.

The second usual case in origami is wanting to fold a single “flap”. For lack of a better word.

A "flap" will be a single face or a set of faces connected to one another that need to be folded together because of their connectedness.

At the time of writing there are only two types of fold considered by the application:

1. The "total" fold, that folds every face intersected by the segment up to the face that is  
   partially intersected (or totally intersected by connected through the side to a partially intersected face) and thus, cannot be folded, if there is any, or otherwise the whole bundle.
2. The single flap fold, that folds the minimum number of faces given  
   a segment and a “GUI user looking direction” from which a set of visible faces intersected by the segment can be found. Note that this kind of fold may also require specifying the side to fold, as it can be asymmetric.

These two cases should cover most desired kind of folds. Trying to cover other different kinds of fold is not easy. Folding flaps is deceptively complex as there are many corner cases.

I am quite confident that these two cases cover quite well most, but certainly not all necessities. Note that, normally, in origami, we are doing one of two things: folding the whole figure over along a line o folding a single layer or flap of the figure. Even more unusual cases like folding a flap and every flap behind it, if there is any, is still covered the “total” line fold.

Line folding cases not covered:

* Fold and “pop”. When there are two bisected faces, and we want to fold one of them, let’s say the front one, but not the one behind it, which we just want to “pop” out without folding it.
* Fold behind layers. Sometimes in origami instructions, there is an “x-ray” type of drawing showing a fold to be made on a face hidden behind another.
* Anything else?

# Error Checking

At the time of writing, and predictably nowhere in the near future, will error checking be very sophisticated.

The purpose of the folding algorithms is that you can do what you want to do and there are no errors. Caring about “malicious” users is secondary. In other words, it’s letting the user do a bona fide fold and considering all the corner cases for it.