A New File Standard to Represent Folded Structures

Erik D. Demaine* Jason S. Ku* Robert J. Lang†

Computational origami is a rapidly growing field. Each year, researchers are developing and implementing new algorithms and software to solve folding related problems. Many of these software can export folding geometry in standard 2D and 3D vector file formats. However, none of the currently available vector mesh standards (DWG, DXF, OBJ, etc.) support coplanar facet layer ordering, a critically necessary feature when representing many folded states. By adopting a standard file format, the idea is that one can write relatively simple tools to convert otherwise incompatible file formats from different software to/from the standard as an intermediate form, thereby allowing the output from one such tool be fed as input into another without having to write specialized pairwise converters.

In order to facilitate data interoperability between folding related software, we introduce the Flexible Origami List Datastructure (FOLD) format, built on top of the open-standard JavaScript Object Notation (JSON) format. One benefit of adopting this format is that JSON parsers are already available in many languages. The FOLD format seeks to balance generality and simplicity: able to represent a wide variety of folded structures in different dimensions including general codimensional layering information, but able to represent common folded structures simply.

The FOLD Format represents one or more frames containing linked geometric information describing folded and layered geometry in a flat listing of arrays, similar to information storage in the OBJ format. JSON structures store data in either zero-indexed arrays or key-value dictionaries. At the top level of every FOLD file is a dictionary containing keys linking to either geometric data for a folded structure (*frame properties*), or metadata about the file itself (*metadata properties*).

All properties have an **A_B** naming convention, where **A** represents some implicit object or objects and **B** represents some property of **A**. For example, if `students` represents an arbitrarily ordered list of students, then `students_name` might represent the name of each student, while `students_age` might represent each student's age. The `students_name` property would be associated with an array of names, while the `students_age` property would be associated with an array of integers, with element i of `students_name` corresponding to the name of student i, and element i of `students_age` corresponding to the age of student i. Laying out data in this flat representation decreases the depth of the object tree and makes it easy to add custom data onto existing objects.

Standard metadata properties in the FOLD format include `file_version`, `file_creator`, `file_author`, `file_classes`. `file_version` specifies the version of the FOLD spec that the file is written under, `file_creator` specify the software that created the file, and `file_author` specifies the human author. `file_class` specifies a subjective interpretation about what the file represents. Some standard classes are `creasePattern`, `foldedForm`, and `foldingMotion`.

Frame properties in the FOLD format include frame metadata like `frame_title`, `frame_author`, and `frame_attributes`, in addition to geometric properties like `vertices_coords`, `edges_vertices`, `edges_assignment`, `faces_vertices`, and `faceOrders`.

Property `frame_title` and `frame_author` respectively store the title and author of the folded structure represented by the frame. `frame_attributes` stores an array of attributes that objectively describe properties of the folded structure being represented. Examples of standard frame attributes include `2D`, `3D`, `manifold`, `nonManifold`, `selfIntersecting`, and `nonSelfIntersecting`.

Property `vertices_coords` is an array of coordinate arrays, one for each vertex of the folded structure. Coordinate arrays may represent coordinates in one, two, three or higher dimension, but all coordinate arrays within `vertices_coords` should be the same dimension. A reader can interpret the coordinates in a higher dimensional space by padding the provided coordinates with zeros. Note that no property is mandatory, including `vertices_coords`. If the property exists, the frame describes an embedded folded structure, or an unembedded folded structure if the property is missing.

Property `edges_vertices` is an array of vertex index pairs, one for each edge of the folded structure. Thus, the undirected pair $[1, 3]$ represents an edge between the second and fourth vertices, since arrays are zero-indexed. This property can be used to reference the individual creases of a folded structure, or in general, any directed graph on an abstract set of vertices. `edges_assignment` is an array of character labels from the set {V, M, F, U, B}, one for each edge of the folded structure. The labels correspond to the edge being a valley fold (V), mountain fold (M), flat (F), unknown (U), or boundary (B).

Property `faces_vertices` is an array of vertex index lists, one for each face of the folded structure. If a face has orientation, the convention is to list vertices in

*MIT, contact: `jasonku@mit.edu`
†Lang Origami

```
{
  "file_version": 1,
  "file_creator": "A text editor",
  "file_author": "Jason Ku",
  "file_class": "foldedForm",
  "frame_title": "Three fold 3D example",
  "vertices_coords": [
    [0, 1, 0],
    [0, 0, 1],
    [0,-1, 0],
    [1, 0, 0],
    [0, 0,-1],
    [0, 0,-1]
  ],
  "faces_vertices": [
    [0,1,2],
    [0,2,3],
    [0,4,1],
    [1,5,2]
  ],
  "edges_vertices": [
    [0,2],
    [0,1],
    [1,2],
    [2,3],
    [0,3],
    [1,4],
    [1,5],
    [0,4],
    [2,5]
  ],
  "edges_assignment": [
    "V",
    "M",
    "M",
    "B",
    "B",
    "B",
    "B",
    "B",
    "B"
  ],
  "faceOrders": [
    [0,2,-1],
    [2,0,-1],
    [0,3,-1],
    [3,0,-1]
  ]
}
```

Figure 1: An example of a simply folded structure in FOLD format.

counter-clockwise order about the face orientation.

Property `faceOrders` is an array of triples. Each triple $[i, j, k]$ represents a condition on the layer ordering between two faces. i and j represent indices of two orientable faces. The third element k is taken from the set $\{-1, 0, 1, \texttt{null}\}$ based on the orientation of face j relative to the orientation of face i, as in [1]. If face j lies above face i relative to face i's orientation, then $k = 1$, whereas we set $k = -1$ if face j lies below face i. A value of $k = 0$ is given for all other instances. We note that under the assumption of transitivity, not all pairwise face orders need to be defined to uniquely identify a folded state. Readers should infer a unique total face order if one is implied by the geometry and a given partial set of face orders. An example of a single frame folded form in FOLD format is shown in Figure 1.

An additional file level property `file_frames` can store an array of additional folded structure frames within the same file. Frames can be related via parent pointers, using the `frame_parent` property to store the index of another frame. Frames can be independent from one another, or inherit properties from their parent by setting the `frame_inherit` boolean property to true. This advanced capability allows the FOLD file format the ability to store folding sequences, or key-frames of a folding motion.

In addition, we have written an online web tool for viewing folded structures FOLD file type; see Figure 2. It is written in CoffeeScript, allows object rotation, renders geometry as SVG in the browser, dynamically computing layer order relative to the camera direction, and reordering faces in real time.

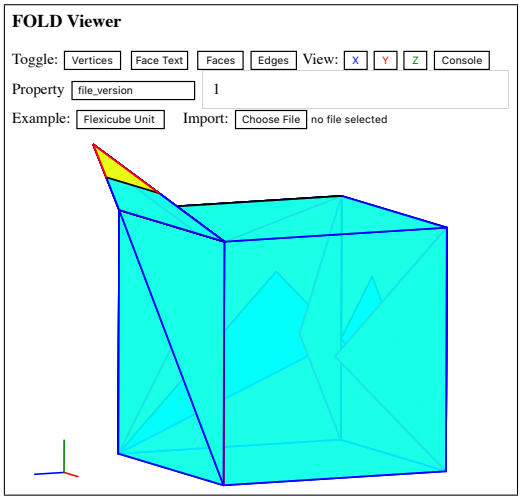

Figure 2: Web tool for viewing FOLD files.

References

[1] Erik D. Demaine and Joseph O'Rourke. *Geometric Folding Algorithms: Linkages, Origami, Polyhedra.* Cambridge University Press, July 2007.