Mathematical Origami

Geoff Hagopian

College of the Desert

ghagopian@collegeofthedesert.edu

October 25, 2016

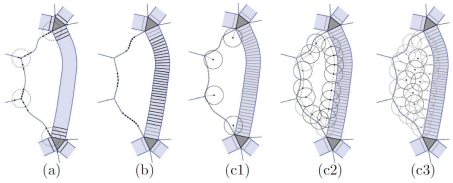
Fall 2016 : Math & Science Lecture Series

The College of the Desert School of Math and Science Presents:

The Mathematics and Algorithms of Origami

Figure 1: Robert Lang's Fiddler Crab.

On Tuesday, October 25^{th} in MSTC 250, College of the Desert Math and Computer Science Professor Geoff Hagopian will give an introduction to the exciting new science of computational origami.



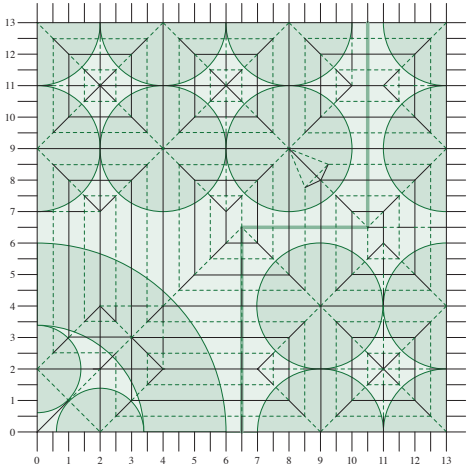


Figure 2: The crease pattern of the fiddler crab.

From simple folds on 1-dimensional paper to squash folds in hyperspace, learn how new computer algorithms are designed to create crease patterns to fold lizards and lions; elk, shrimp, praying mantises and the dollar velociraptor.

This is a potent new area of research and development with applications ranging from biology (folding proteins) to astrophysics (deploying NASA's gliders in the Martian atmosphere!)

Overview

1. Mathematical definitions for origami.

Overview

1. Mathematical definitions for origami.
2. Universality.

Overview

1. Mathematical definitions for origami.
2. Universality.
3. Turn Gadget

Overview

1. Mathematical definitions for origami.
2. Universality.
3. Turn Gadget
4. Example (dodecahedron)

Overview

1. Mathematical definitions for origami.
2. Universality.
3. Turn Gadget
4. Example (dodecahedron)
5. One Dimensional Flat Folding.

Overview

1. Mathematical definitions for origami.
2. Universality.
3. Turn Gadget
4. Example (dodecahedron)
5. One Dimensional Flat Folding.
6. Two Dimensional Flat Folding

Overview

1. Mathematical definitions for origami.
2. Universality.
3. Turn Gadget
4. Example (dodecahedron)
5. One Dimensional Flat Folding.
6. Two Dimensional Flat Folding
7. Single vertex Folding.

Overview

1. Mathematical definitions for origami.
2. Universality.
3. Turn Gadget
4. Example (dodecahedron)
5. One Dimensional Flat Folding.
6. Two Dimensional Flat Folding
7. Single vertex Folding.
   - Maekawa's Theorem

Overview

1. Mathematical definitions for origami.
2. Universality.
3. Turn Gadget
4. Example (dodecahedron)
5. One Dimensional Flat Folding.
6. Two Dimensional Flat Folding
7. Single vertex Folding.
   - Maekawa's Theorem
   - Kawasaki's Theorem

Overview

1. Mathematical definitions for origami.
2. Universality.
3. Turn Gadget
4. Example (dodecahedron)
5. One Dimensional Flat Folding.
6. Two Dimensional Flat Folding
7. Single vertex Folding.
   - Maekawa's Theorem
   - Kawasaki's Theorem
8. Tesselation Folds

Overview

1. Mathematical definitions for origami.
2. Universality.
3. Turn Gadget
4. Example (dodecahedron)
5. One Dimensional Flat Folding.
6. Two Dimensional Flat Folding
7. Single vertex Folding.
   - Maekawa's Theorem
   - Kawasaki's Theorem
8. Tesselation Folds
9. Straight Cuts

Overview

1. Mathematical definitions for origami.
2. Universality.
3. Turn Gadget
4. Example (dodecahedron)
5. One Dimensional Flat Folding.
6. Two Dimensional Flat Folding
7. Single vertex Folding.
   - Maekawa's Theorem
   - Kawasaki's Theorem
8. Tesselation Folds
9. Straight Cuts
10. Treemaker.

Overview

1. Mathematical definitions for origami.
2. Universality.
3. Turn Gadget
4. Example (dodecahedron)
5. One Dimensional Flat Folding.
6. Two Dimensional Flat Folding
7. Single vertex Folding.
   - Maekawa's Theorem
   - Kawasaki's Theorem
8. Tesselation Folds
9. Straight Cuts
10. Treemaker.
11. Origamizer.

Overview

1. Mathematical definitions for origami.
2. Universality.
3. Turn Gadget
4. Example (dodecahedron)
5. One Dimensional Flat Folding.
6. Two Dimensional Flat Folding
7. Single vertex Folding.
   - Maekawa's Theorem
   - Kawasaki's Theorem
8. Tesselation Folds
9. Straight Cuts
10. Treemaker.
11. Origamizer.
12. Hyperbolic Paraboloid

Origami Definitions

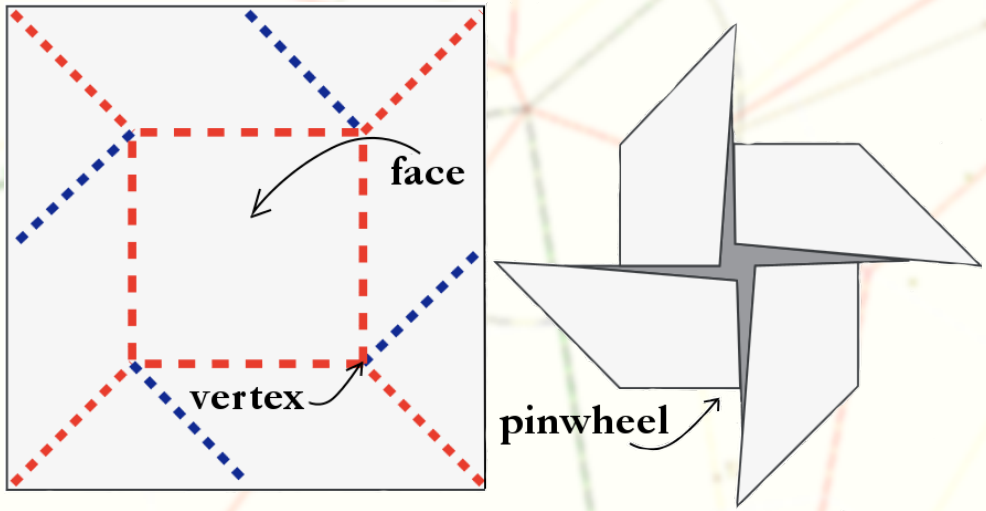
- Piece of paper: A 2-D polygon with distinguished top and bottom faces.

Origami Definitions

- Piece of paper: A 2-D polygon with distinguished top and bottom faces.
- Crease: line segment or curve on the paper.

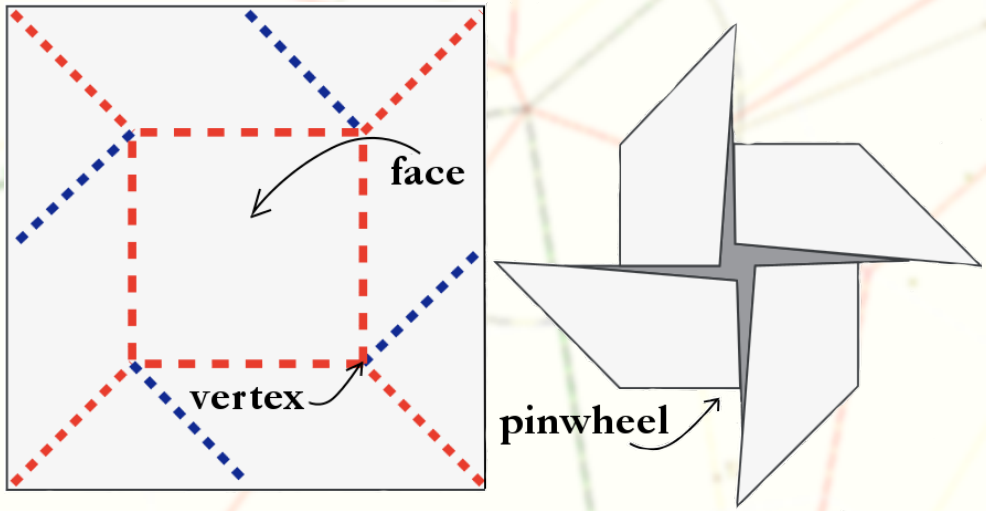
Origami Definitions

- Piece of paper: A 2-D polygon with distinguished top and bottom faces.
- Crease: line segment or curve on the paper.
- Crease pattern: a set of creases; that is, a planar graph drawn on the paper.



Origami Definitions

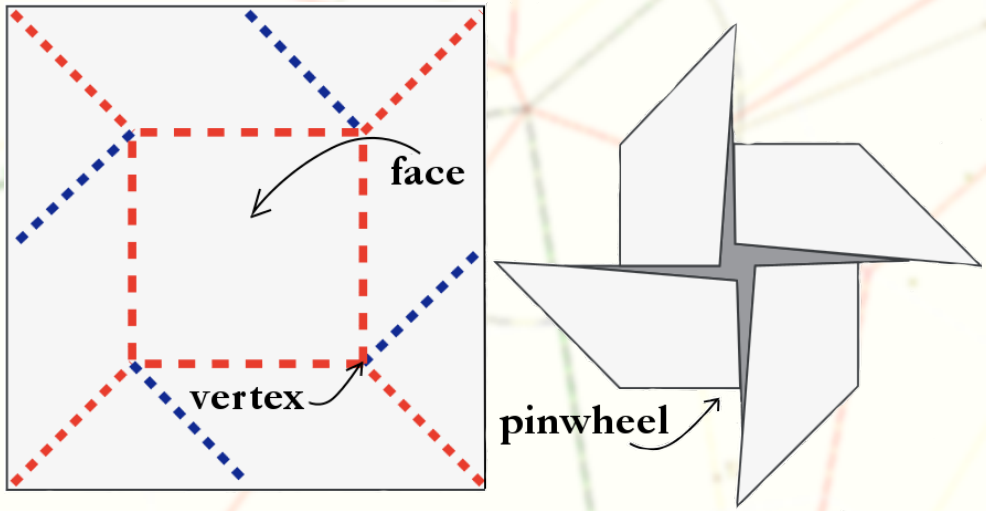
- Piece of paper: A 2-D polygon with distinguished top and bottom faces.
- Crease: line segment or curve on the paper.
- Crease pattern: a set of creases; that is, a planar graph drawn on the paper.



- Folded state: finished origami–unfolding→ crease pattern

Origami Definitions

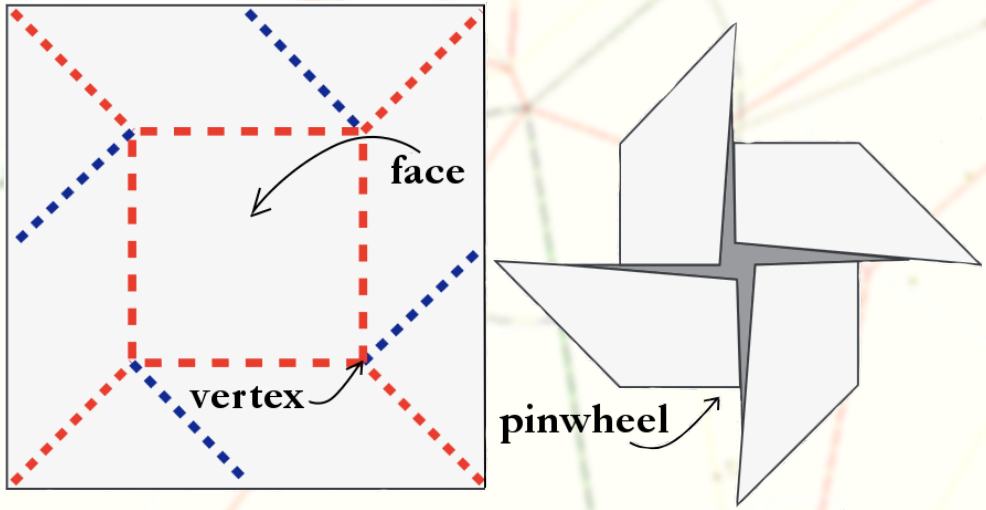
- Piece of paper: A 2-D polygon with distinguished top and bottom faces.
- Crease: line segment or curve on the paper.
- Crease pattern: a set of creases; that is, a planar graph drawn on the paper.



- Folded state: finished origami–unfolding→ crease pattern
- Flat folding: folded state lying in a plane–if all creases are folded, the crease pattern is called flat foldable

Origami Definitions

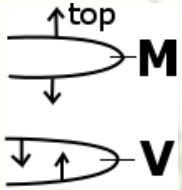
- Piece of paper: A 2-D polygon with distinguished top and bottom faces.
- Crease: line segment or curve on the paper.
- Crease pattern: a set of creases; that is, a planar graph drawn on the paper.



- Folded state: finished origami–unfolding→ crease pattern
- Flat folding: folded state lying in a plane–if all creases are folded, the crease pattern is called flat foldable

Valley crease: top sides touch

- Mountain crease: bottom sides touch



More Origami Terminology

- Mountain-Valley assignment: specify which creases are mountains and which are valleys.

More Origami Terminology

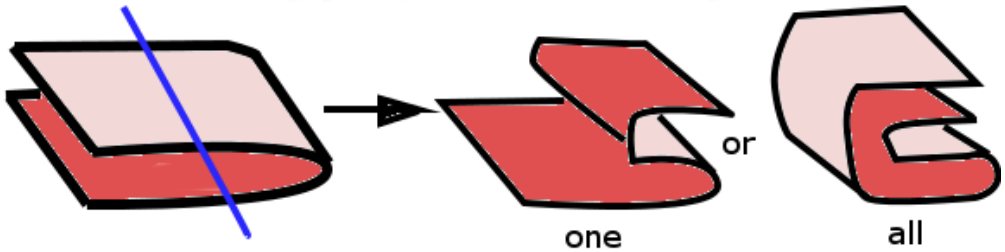
- Mountain-Valley assignment: specify which creases are mountains and which are valleys.
- Mountain-Valley pattern: crease pattern + mountain-valley assignment.

More Origami Terminology

- Mountain-Valley assignment: specify which creases are mountains and which are valleys.
- Mountain-Valley pattern: crease pattern + mountain-valley assignment.
- Simple fold: fold along a single mountain/valley line by $\pm 180^\circ$
  This presents the choice of how many layers to fold.

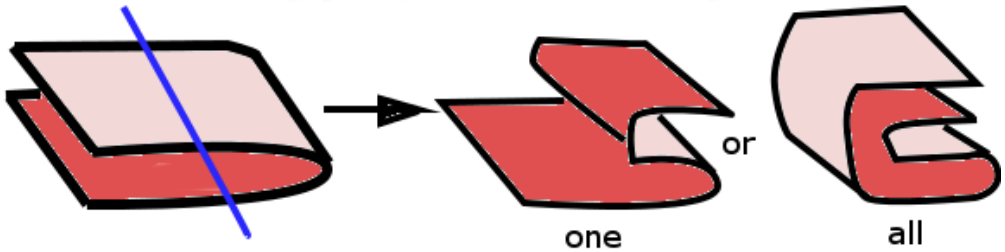
More Origami Terminology

- Mountain-Valley assignment: specify which creases are mountains and which are valleys.
- Mountain-Valley pattern: crease pattern + mountain-valley assignment.
- Simple fold: fold along a single mountain/valley line by $\pm 180^\circ$
  This presents the choice of how many layers to fold.
- One-layer simple fold: just top or bottom layer
  All-layers simple fold: fold all the layers



More Origami Terminology

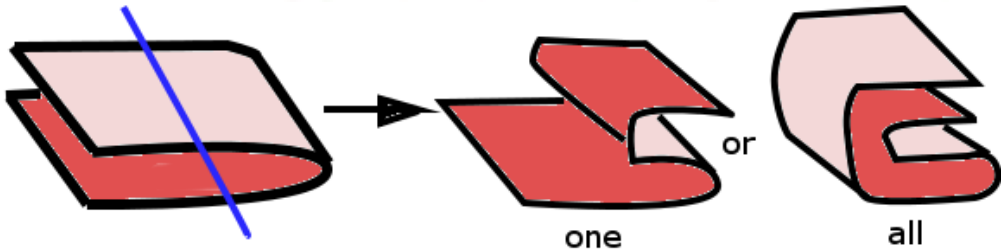
- Mountain-Valley assignment: specify which creases are mountains and which are valleys.
- Mountain-Valley pattern: crease pattern + mountain-valley assignment.
- Simple fold: fold along a single mountain/valley line by $\pm 180^\circ$
  This presents the choice of how many layers to fold.
- One-layer simple fold: just top or bottom layer
  All-layers simple fold: fold all the layers



- Strip: a long, narrow rectangle 

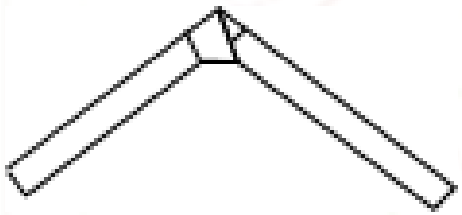
More Origami Terminology

- Mountain-Valley assignment: specify which creases are mountains and which are valleys.
- Mountain-Valley pattern: crease pattern + mountain-valley assignment.
- Simple fold: fold along a single mountain/valley line by $\pm 180^\circ$
  This presents the choice of how many layers to fold.
- One-layer simple fold: just top or bottom layer
  All-layers simple fold: fold all the layers



- Strip: a long, narrow rectangle

- Non-simple strip folding: a strip knot



Universality: Any shape is made by folding.

- Every connected union of polygons in 3D, each with a specified visible color (on each side), can be folded from a sufficiently large piece of bicolor paper of any shape (say, a square.) [Demaine, Mitchell 2000]

  Proof by algorithm:

Universality: Any shape is made by folding.

- Every connected union of polygons in 3D, each with a specified visible color (on each side), can be folded from a sufficiently large piece of bicolor paper of any shape (say, a square.) [Demaine, Mitchell 2000]

  Proof by algorithm:

  - fold paper down to long narrow strip (!)

Universality: Any shape is made by folding.

- Every connected union of polygons in 3D, each with a specified visible color (on each side), can be folded from a sufficiently large piece of bicolor paper of any shape (say, a square.) [Demaine, Mitchell 2000]

  Proof by algorithm:

  - fold paper down to long narrow strip (!)
  - triangulate the polygons

Universality: Any shape is made by folding.

- Every connected union of polygons in 3D, each with a specified visible color (on each side), can be folded from a sufficiently large piece of bicolor paper of any shape (say, a square.) [Demaine, Mitchell 2000]

  Proof by algorithm:

  - fold paper down to long narrow strip (!)
  - triangulate the polygons
  - choose a path visiting each triangle at least once

Universality: Any shape is made by folding.

- Every connected union of polygons in 3D, each with a specified visible color (on each side), can be folded from a sufficiently large piece of bicolor paper of any shape (say, a square.) [Demaine, Mitchell 2000]

  Proof by algorithm:

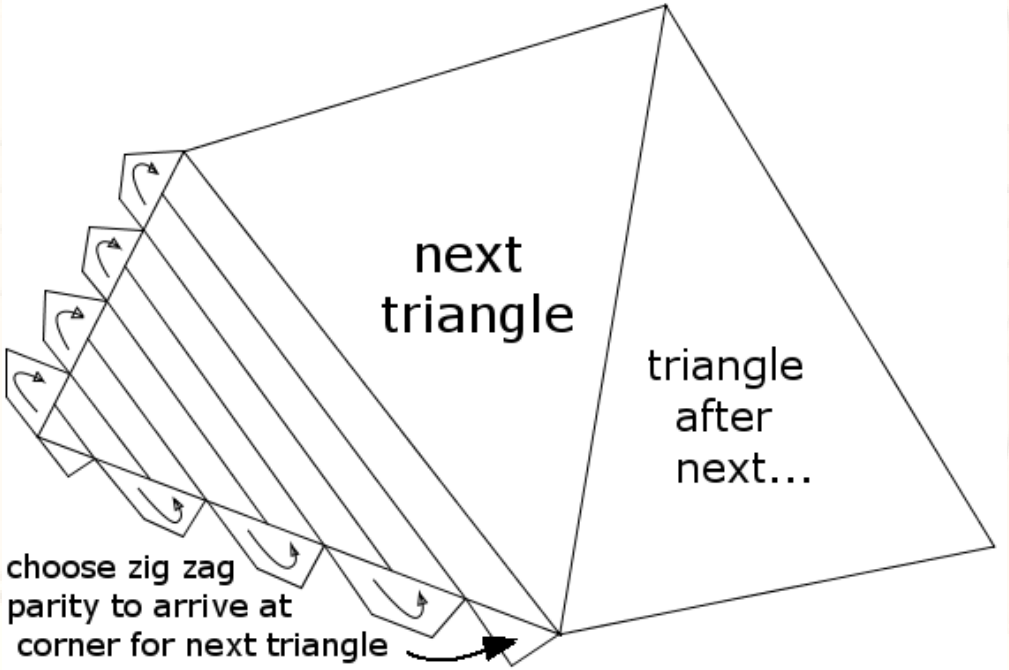
  - fold paper down to long narrow strip (!)
  - triangulate the polygons
  - choose a path visiting each triangle at least once
  - cover each triangle along the path by zig-zagging parallel to next edge, starting at opposite corner

Universality: Any shape is made by folding.

- Every connected union of polygons in 3D, each with a specified visible color (on each side), can be folded from a sufficiently large piece of bicolor paper of any shape (say, a square.) [Demaine, Mitchell 2000]

  Proof by algorithm:

  - fold paper down to long narrow strip (!)
  - triangulate the polygons
  - choose a path visiting each triangle at least once
  - cover each triangle along the path by zig-zagging parallel to next edge, starting at opposite corner



Turn Gadget.

A strip and be turned by any angle with the same face facing up.

- Algorithm:

Turn Gadget.

A strip and be turned by any angle with the same face facing up.

- Algorithm:
  - 1. Fold the strip back under itself.

Turn Gadget.

A strip and be turned by any angle with the same face facing up.

- Algorithm:
  - 1. Fold the strip back under itself.
  - 2. Fold the lower layer at an angle which is half the turn angle.

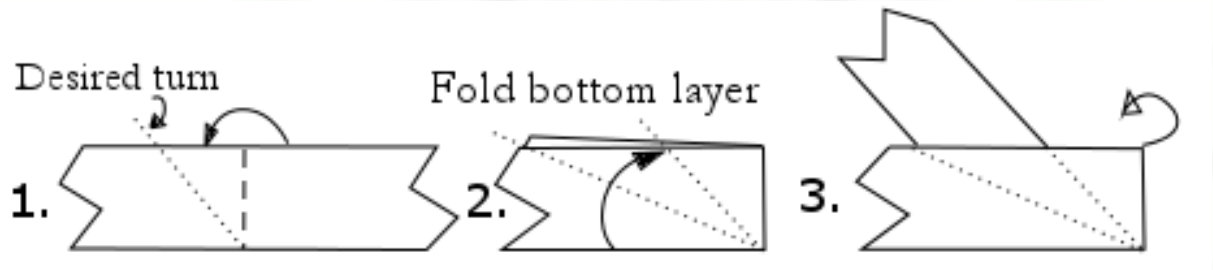
Turn Gadget.

A strip and be turned by any angle with the same face facing up.

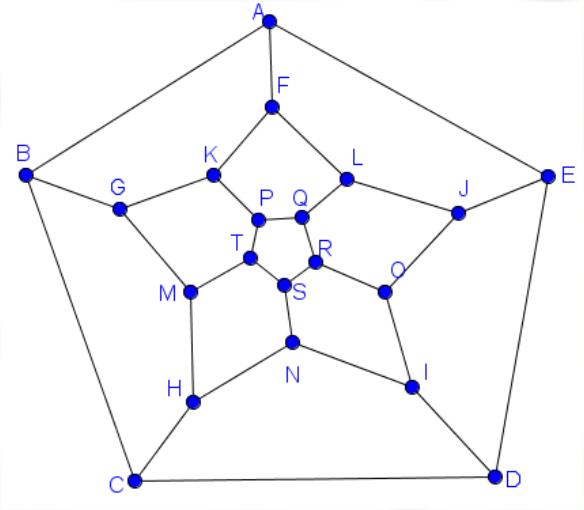
- Algorithm:
  - 1. Fold the strip back under itself.
  - 2. Fold the lower layer at an angle which is half the turn angle.
  - 3. Fold the extra paper out of the way.

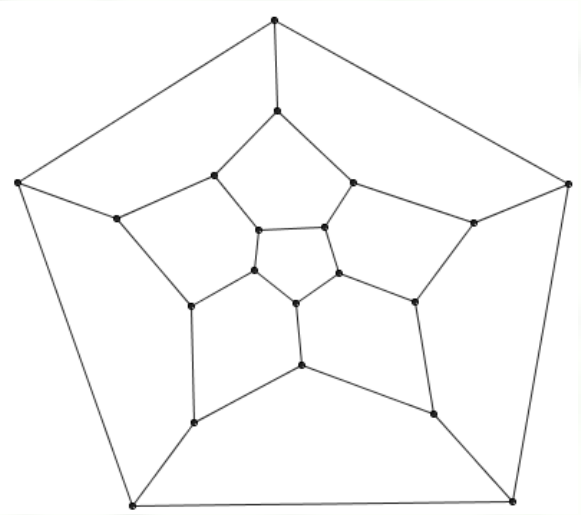
Turn Gadget.

A strip and be turned by any angle with the same face facing up.

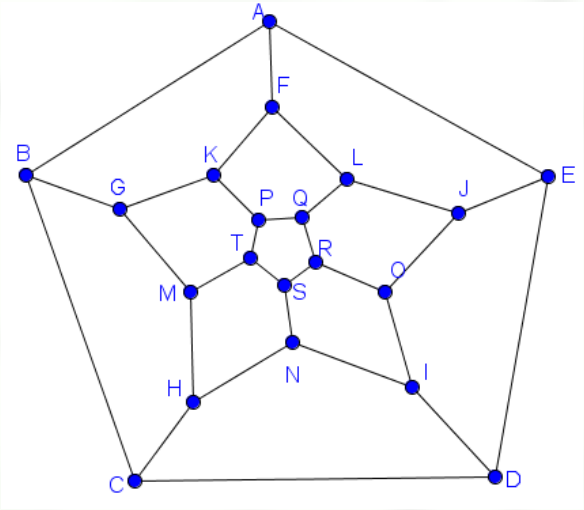
- Algorithm:
  - 1. Fold the strip back under itself.
  - 2. Fold the lower layer at an angle which is half the turn angle.
  - 3. Fold the extra paper out of the way.
  - 

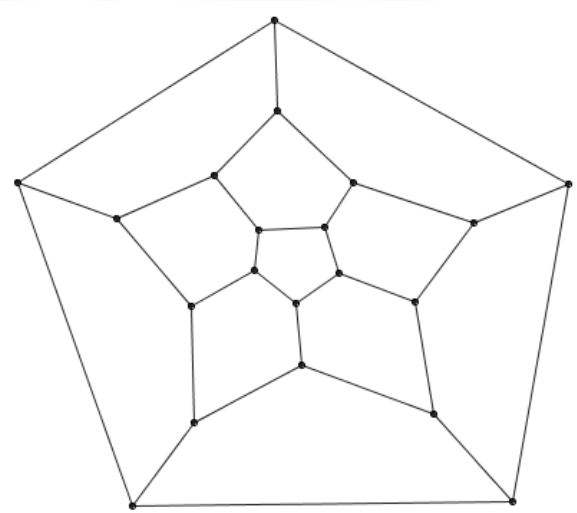
Example: Planar Graph for Dodecahedron.

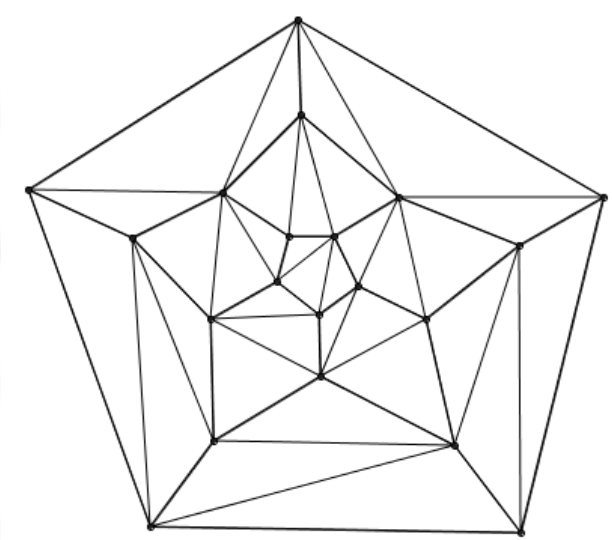




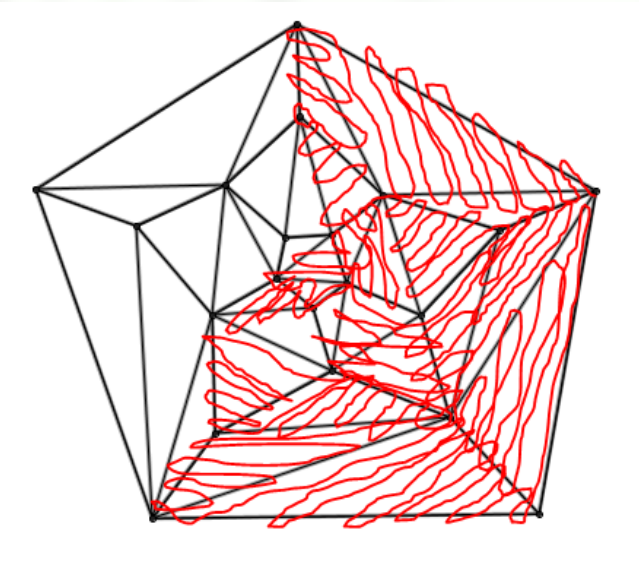
Example: Planar Graph for Dodecahedron.





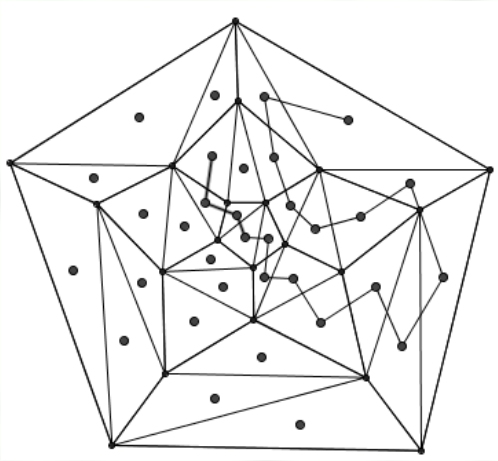


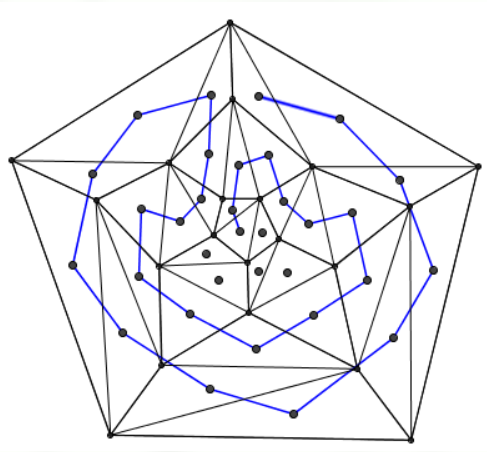
Triangulate



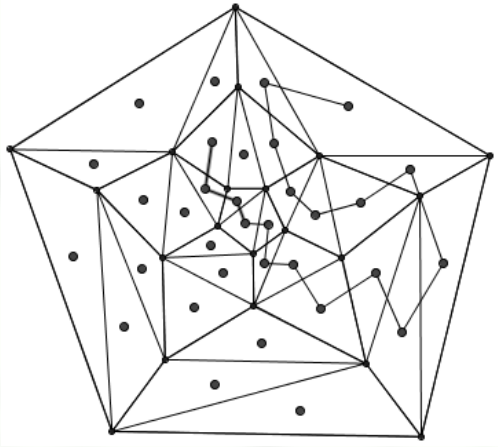
Choose a path and cover each triangle

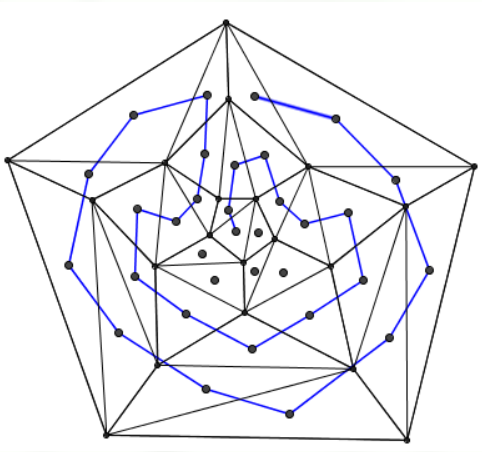
Example: Can we find an efficient path?

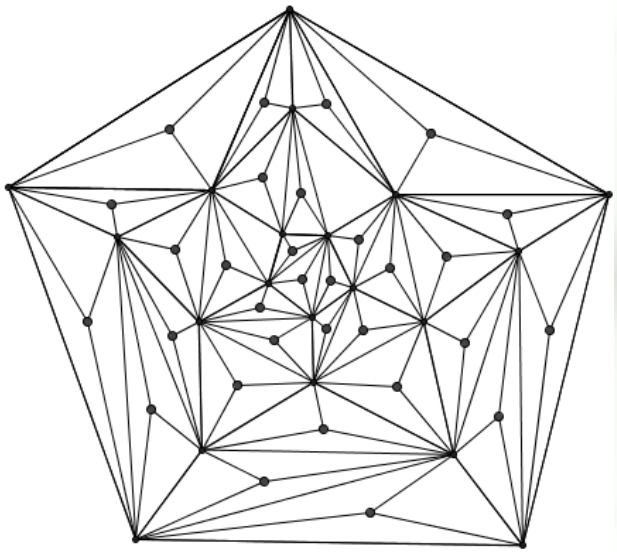




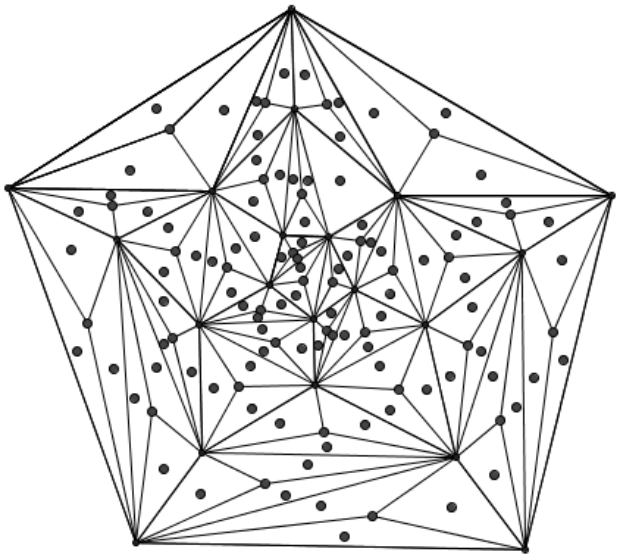
Example: Can we find an efficient path?





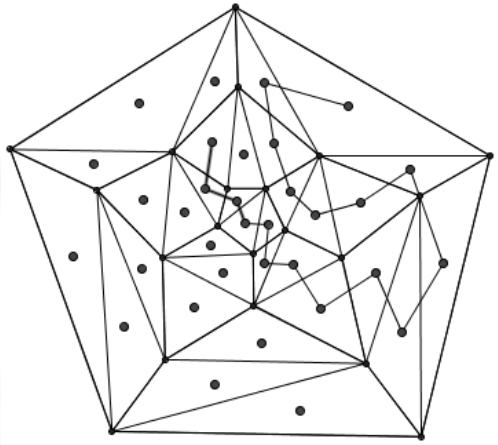


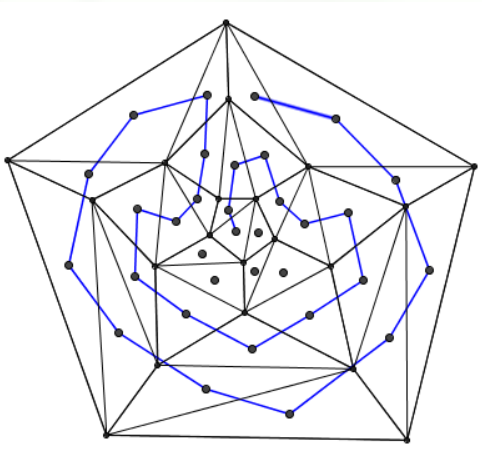
Triangulate the triangulation

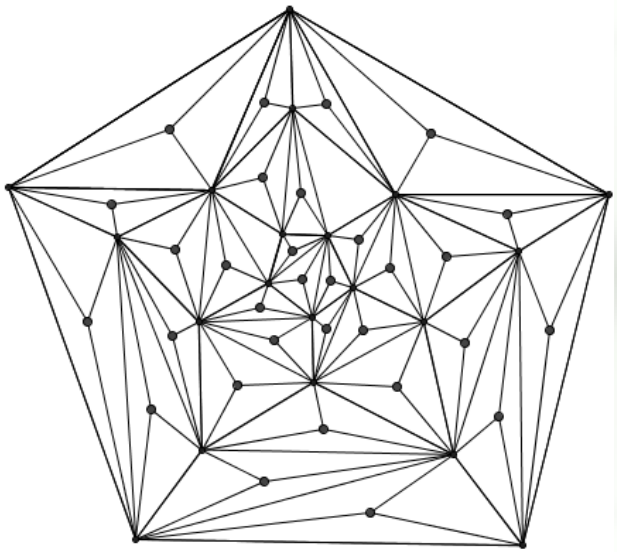


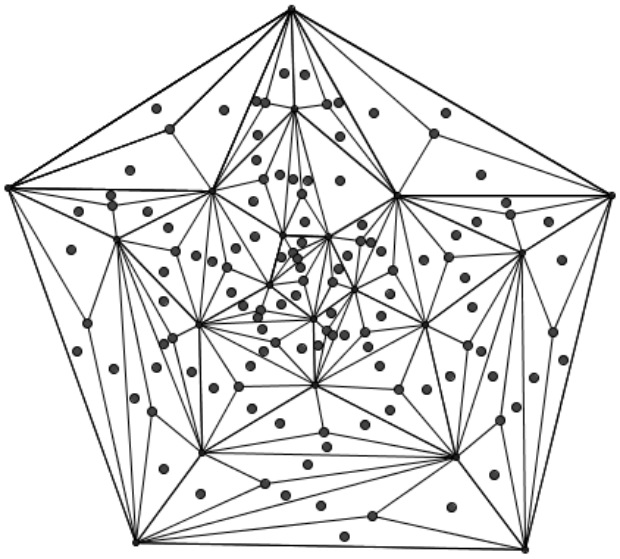
Dot's Hamiltonian if it has a 2-coloring.

Example: Can we find an efficient path?









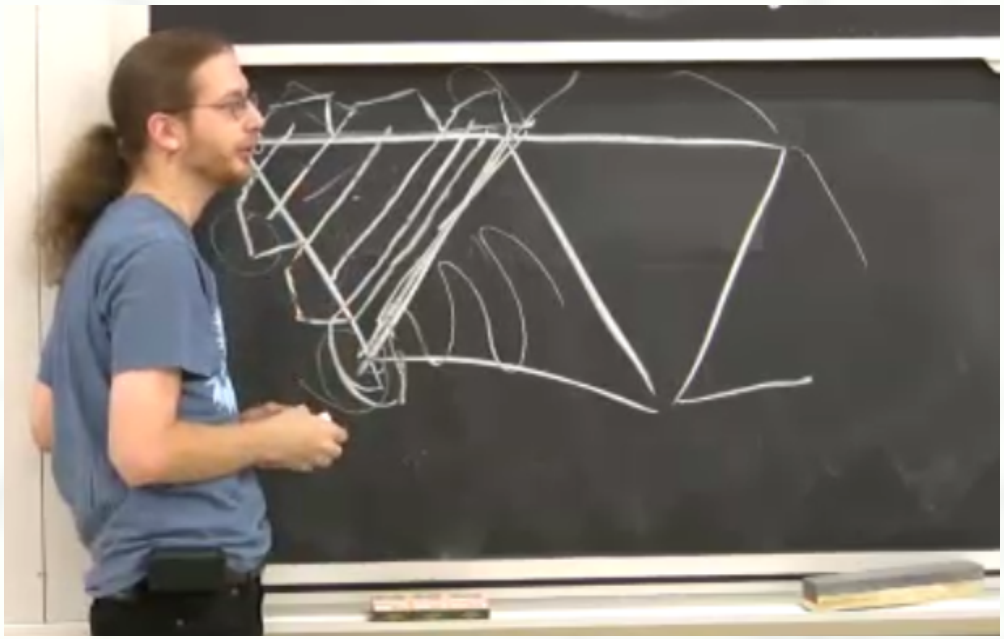
Triangulate the triangulation

Dot's Hamiltonian if it has a 2-coloring.

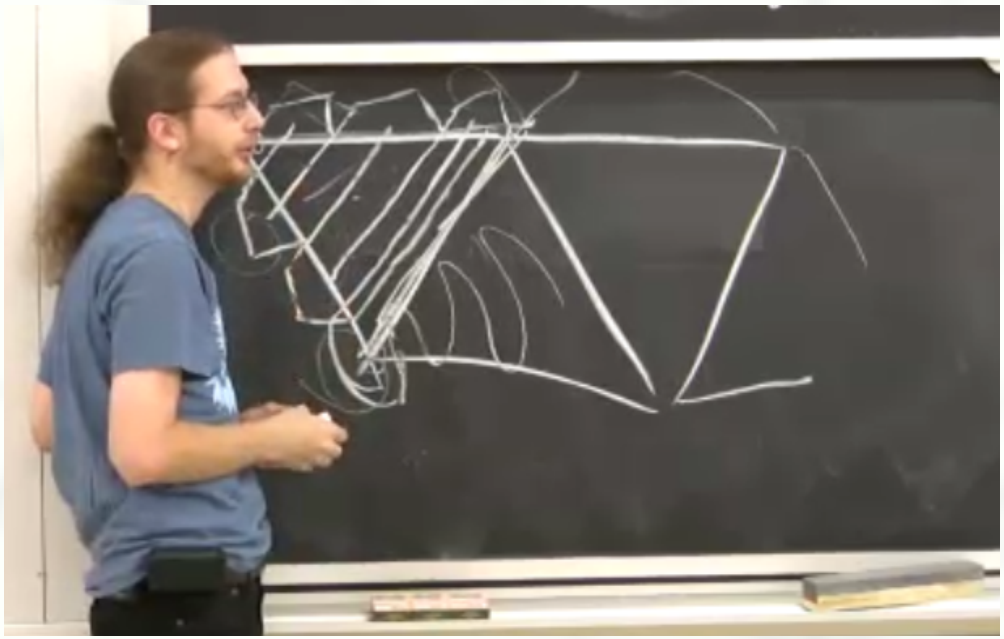
"Boom! You've made anything!"–Erik Demaine

Is tucking under/over here a simple fold?

"Hmmm. I hadn't thought of that." ED



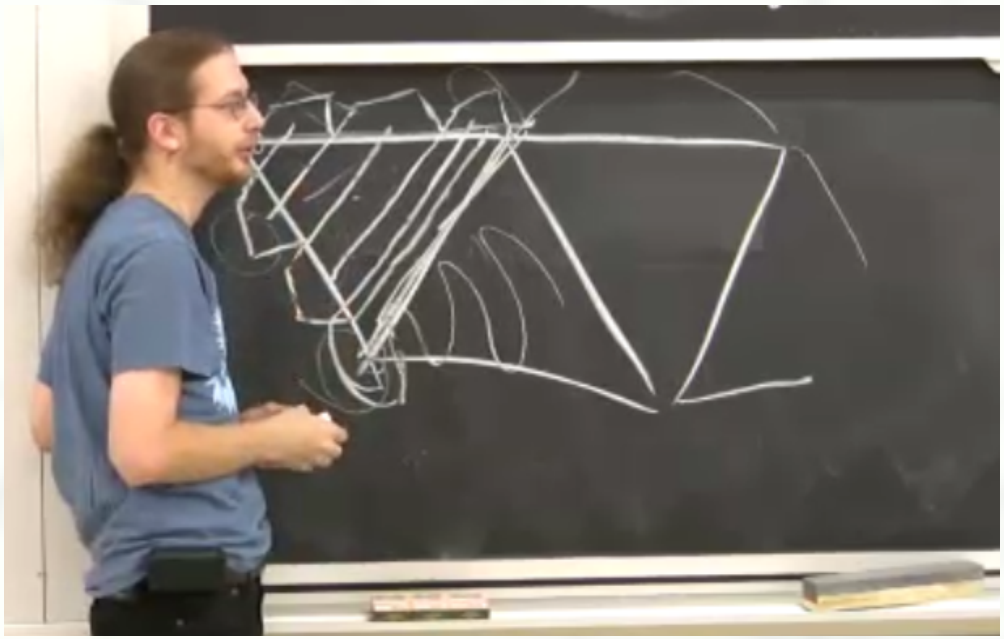
Is tucking under/over here a simple fold? "Hmmm. I hadn't thought of that." ED



Pseudo-efficiency: if allowed to start with any long-enough strip, then we can achieve

$$\texttt{area}(\texttt{paper}) = \texttt{area}(\texttt{surface}) + \epsilon$$

Is tucking under/over here a simple fold? "Hmmm. I hadn't thought of that." ED



Pseudo-efficiency: if allowed to start with any long-enough strip, then we can achieve

$$\texttt{area}(\texttt{paper}) = \texttt{area}(\texttt{surface}) + \epsilon$$

Open: pseudopolyonomial time upper bound? Erick Demaine on this.

1-Dimensional Flat Folding

Piece of Paper = line segment

Crease = point on paper (in general, 1 dimension smaller than dimension of paper)

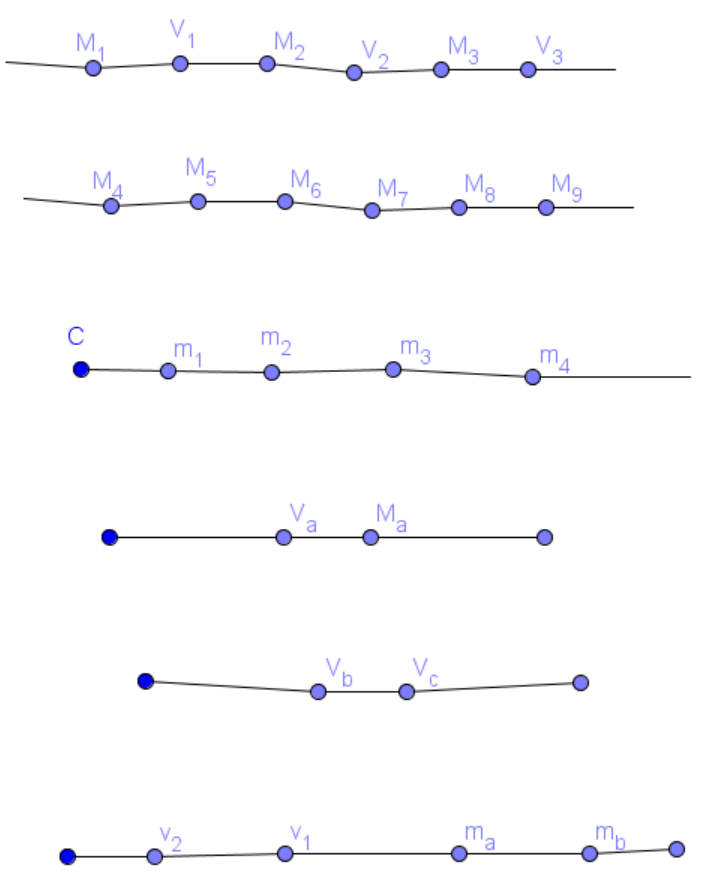
Flat folding lies on a line

All crease patterns are flat-foldable: use an alternating M-V assignment

Not all mountain-valley patterns are f

Two folding operations:

- endfold (if end length$\leq$neighbor)
- pleat MV or VM contains length less than neighbors



- Algorithm: (NEW) (covered in C3)
  - search (left to right) for segment that's crimpable  OR end foldable 
  - if none found: STOP ~ not flat foldable
  - else: do fold 

    merge segments x, y, z → x-y+z
    go back one segment (left of x)
    continue search

C++ program

```
const int maxLength = 99;

class Segment {
public:
    Segment(short len, char ch) {
        length = len;
        fold = ch;
    }
    short length;
    char fold; /// M = mountain, V = valley, E = end
};

vector<Segment> createChain(int);

void print(vector<Segment> S) {
    for(int i = 0; i < S.size()-1; ++i)
        cout << S[i].length << '-' << S[i].fold << '-';
    cout << S[S.size()-1].length << '-' << 'E';
}
bool doFold(int &position,
            vector<vector<Segment> > &Seq);


int main() {
    srand(time(0));
    int n{0};
    cout << "\nEnter the number of segments in your sequence: ";
    do {
        cin >> n;
        if(n<=1) cout << "\nn must be at least 2."  << endl;
    } while(n<=1);
    vector<Segment> S = createChain(n);
    /// add initial sequence to a vector of sequences
    vector<vector<Segment> > Seq;
    Seq.push_back(S);
    bool foldable{true};
    int position{0};
    while(foldable) {
        print(Seq[Seq.size()-1]); cout << endl;
        foldable = doFold(position,Seq);
        cin.get();
    }
    for(int i = 0; i < Seq.size(); ++i) {
        print(Seq[i]);
        cout << endl;
        cin.get();
    }
}
```

```
vector<Segment> createChain(int n) {
    vector<Segment> vS;
    char ch;
    for(int i = 0; i < n-1; ++i) {
        if(rand()%2==0)
            ch = 'M';
        else ch = 'V';
        vS.push_back(Segment(1+rand()%maxLength,ch));
    }
    vS.push_back(Segment(1+rand()%maxLength,'E')); ///end
    return vS;
}
```

```
bool doFold(int &position,
            vector<vector<Segment> > &Seq) {
    bool didFold{false};
    vector<Segment> S = Seq[Seq.size()-1];
    ///if foldable at position,
    if(position==0 && S[0].length < S[1].length)
    {
        ///erase the end Segment
        S.erase(S.begin(),S.begin()+1);
        cout << "\nposition = " << position << endl;
        Seq.push_back(S);
        return true;
    }
    else while(!didFold && position<S.size()-1) {
        ///if pleatable
        if(S[position].length <= S[position-1].length &&
           S[position].length <= S[position+1].length &&
           S[position-1].fold != S[position].fold)
        {  /// compute the length of the replacement segment
            S[position-1].length = S[position-1].length
                                 - S[position  ].length
                                 + S[position+1].length;
            /// fold type will be of the piece on the right
            S[position-1].fold = S[position+1].fold;
            /// delete two segments
            S.erase(S.begin()+position,S.begin()+position+2);
            Seq.push_back(S);
            --position;  ///back up one
            cout << "\nposit = " << position << endl;
            didFold=true;
            return true;
        } /// if it's not pleatable, move on to next position
        ++position;
        cout << "\npos = " << position << endl;
        ///if at the right end, do endfolds while you can
        cout << "\nS.size()-1==" << S.size()-1 << endl;
        while(position==S.size()-1 &&
              S[position].length <= S[position-1].length) {
            cout << "\nhaha";
            S.erase(S.end()-1,S.end());
            S[position-1].fold = 'E';
            Seq.push_back(S);
            --position;
        }
        //didFold=false;
        if(position==S.size()-1) return false;
    }
}
```

Random foldings:

```
Enter the number of segments in your sequence: 15
66-M-43-V-85-M-29-V-65-M-31-M-38-M-41-M-42-V-38-V-58-V-24-V-25-V
108-M-29-V-65-M-31-M-38-M-41-M-42-V-38-V-58-V-24-V-25-V-37-V-92-
144-M-31-M-38-M-41-M-42-V-38-V-58-V-24-V-25-V-37-V-92-E

Enter the number of segments in your sequence: 12
98-M-5-V-89-M-29-V-69-M-53-V-92-V-65-M-83-V-87-M-29-V-98-E
182-M-29-V-69-M-53-V-92-V-65-M-83-V-87-M-29-V-98-E
222-M-53-V-92-V-65-M-83-V-87-M-29-V-98-E
261-V-65-M-83-V-87-M-29-V-98-E
279-V-87-M-29-V-98-E
279-V-156-E
279-E
```

2D Map Folding:

A rectangular paper with axis-parallel creases

2D Map Folding:

A rectangular paper with axis-parallel creases

Every crease pattern is flat foldable by zig-zagging in x then y

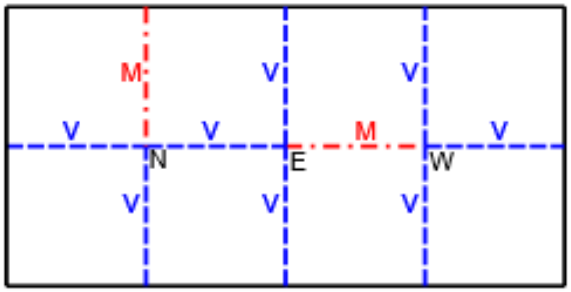
2D Map Folding:

A rectangular paper with axis-parallel creases

Every crease pattern is flat foldable by zig-zagging in x then y

OPEN PROBLEM:

characterize flat-foldable mountain-valley patterns–even in $2 \times n$!



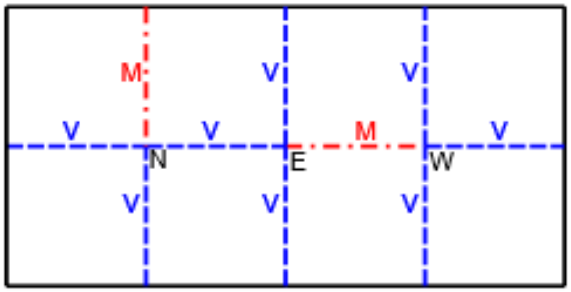
2D Map Folding:

A rectangular paper with axis-parallel creases

Every crease pattern is flat foldable by zig-zagging in x then y

OPEN PROBLEM:

characterize flat-foldable mountain-valley patterns–even in $2 \times n$!



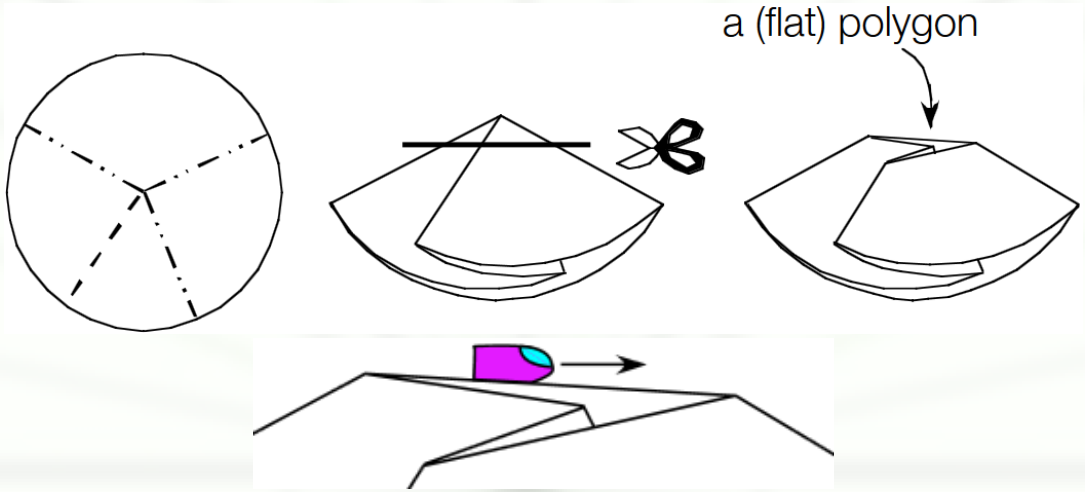
To be simply foldable, a map folding must have at least one M/V fold which cuts across the whole paper. Fold in 1-dimension (either vertically or horizontally) until you can't anymore and then fold 1-dimensionally in the other direction, back and forth between the dimensions until flat-folded.

Single Vertex Folding Theorems

Maekawa's Theorem (1986): The difference between the number of mountain and valley creases in a flat vertex fold is always two.($|M - V| = 2$)

Single Vertex Folding Theorems

Maekawa's Theorem (1986): The difference between the number of mountain and valley creases in a flat vertex fold is always two.($|M - V| = 2$)

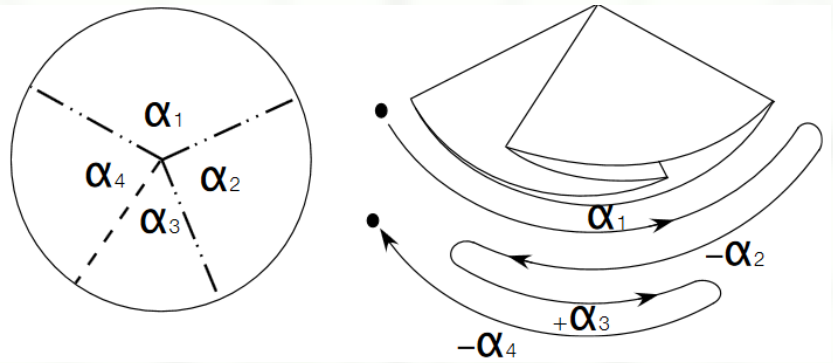


The monorail rotates $180°$ at each M and $-180°$ at each V. Thus $180M - 180V = 360$ (sum of the exterior angles of any polygon), or $M - V = 2$

Kawasaki's Theorem (1989): A collection of creases meeting at a vertex are flat-foldable if and only if the sum of the alternate angles around the vertex is π.

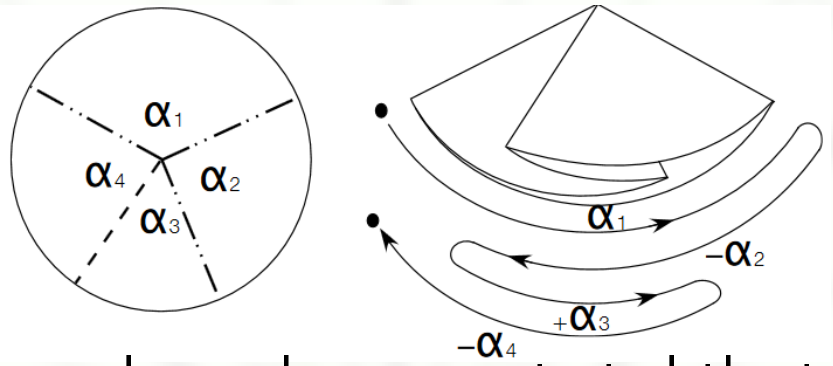
Kawasaki's Theorem (1989): A collection of creases meeting at a vertex are flat-foldable if and only if the sum of the alternate angles around the vertex is π.

Proof: ($\Rightarrow$): Walk around the vertex, starting at a crease on the flat-folded object.



Kawasaki's Theorem (1989): A collection of creases meeting at a vertex are flat-foldable if and only if the sum of the alternate angles around the vertex is π.

Proof: ($\Rightarrow$): Walk around the vertex, starting at a crease on the flat-folded object.

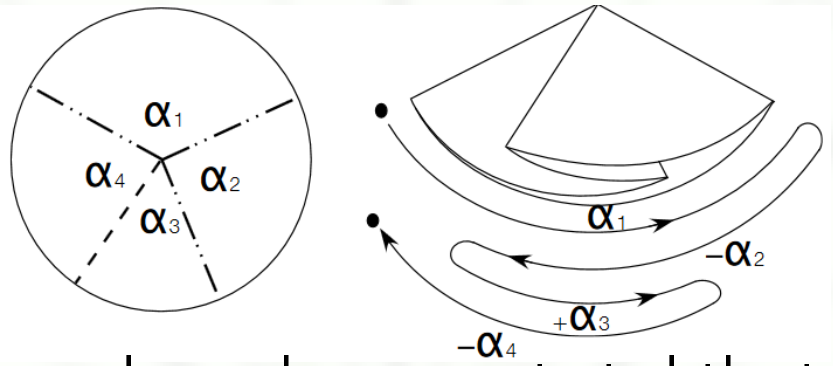


Since you have to end up where you started the total of the back-and-forth angles must be zero:

$$\alpha_1 - \alpha_2 + \alpha_3 - \alpha_4 + \cdots - \alpha_{2n} = 0$$

Kawasaki's Theorem (1989): A collection of creases meeting at a vertex are flat-foldable if and only if the sum of the alternate angles around the vertex is π.

Proof: ($\Rightarrow$): Walk around the vertex, starting at a crease on the flat-folded object.



Since you have to end up where you started the total of the back-and-forth angles must be zero:

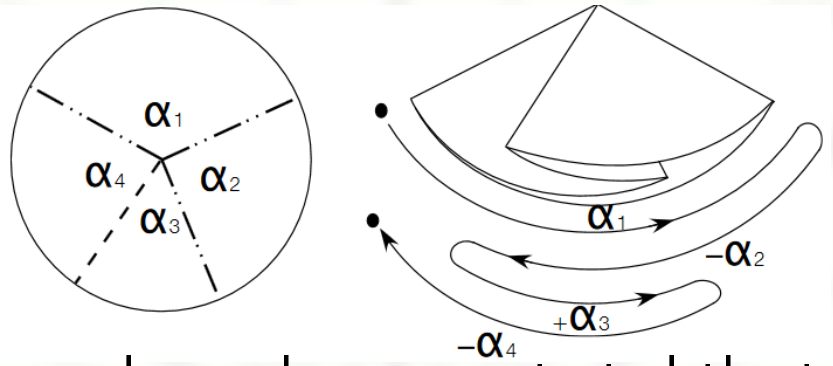
$$\alpha_1 - \alpha_2 + \alpha_3 - \alpha_4 + \cdots - \alpha_{2n} = 0$$

Since the sum of interior angles must be 2π, we can add to this

$$\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 + \cdots + \alpha_{2n} = 2\pi$$

Kawasaki's Theorem (1989): A collection of creases meeting at a vertex are flat-foldable if and only if the sum of the alternate angles around the vertex is π.

Proof: ($\Rightarrow$): Walk around the vertex, starting at a crease on the flat-folded object.



Since you have to end up where you started the total of the back-and-forth angles must be zero:

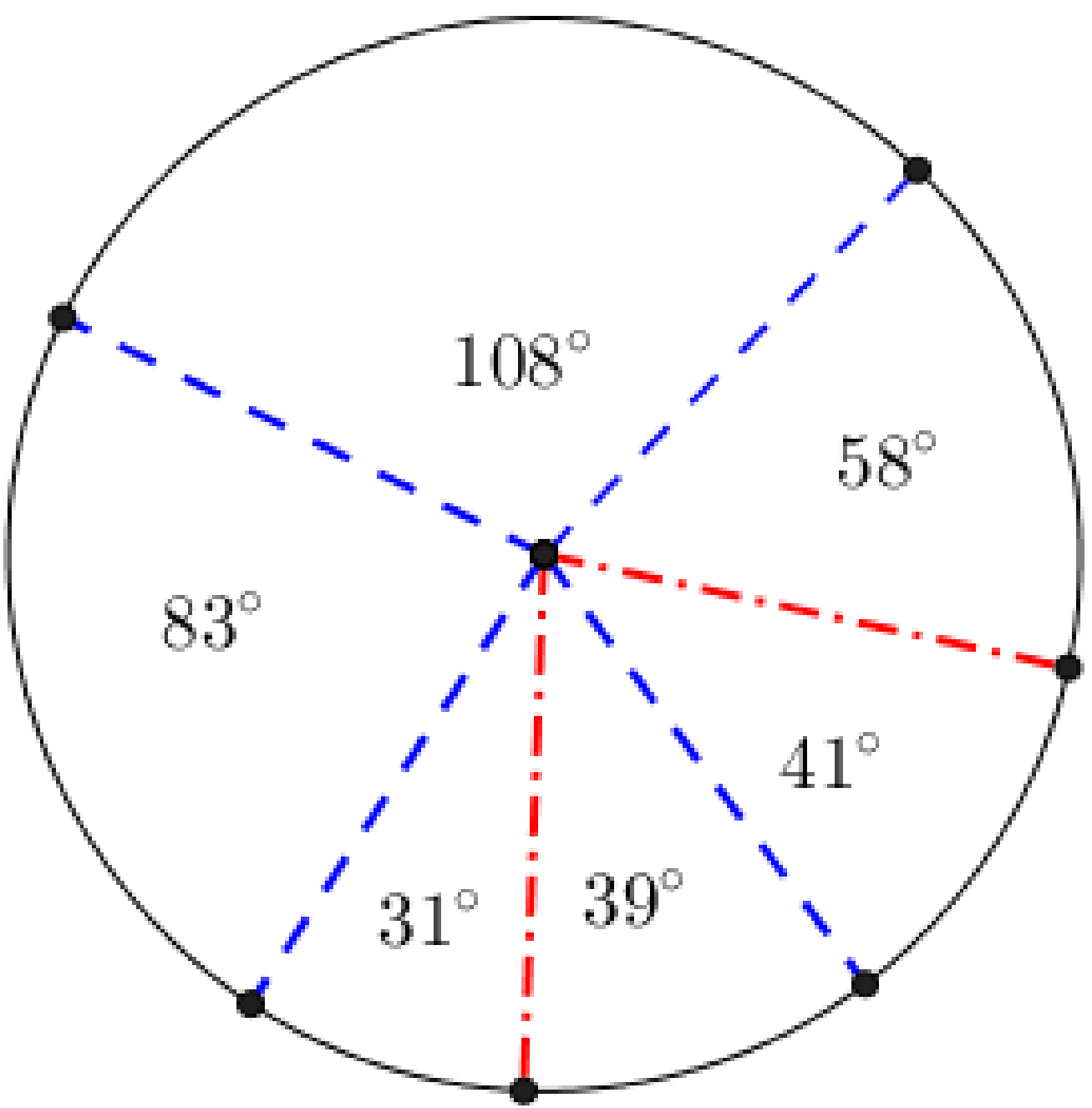
$$\alpha_1 - \alpha_2 + \alpha_3 - \alpha_4 + \cdots - \alpha_{2n} = 0$$

Since the sum of interior angles must be 2π, we can add to this

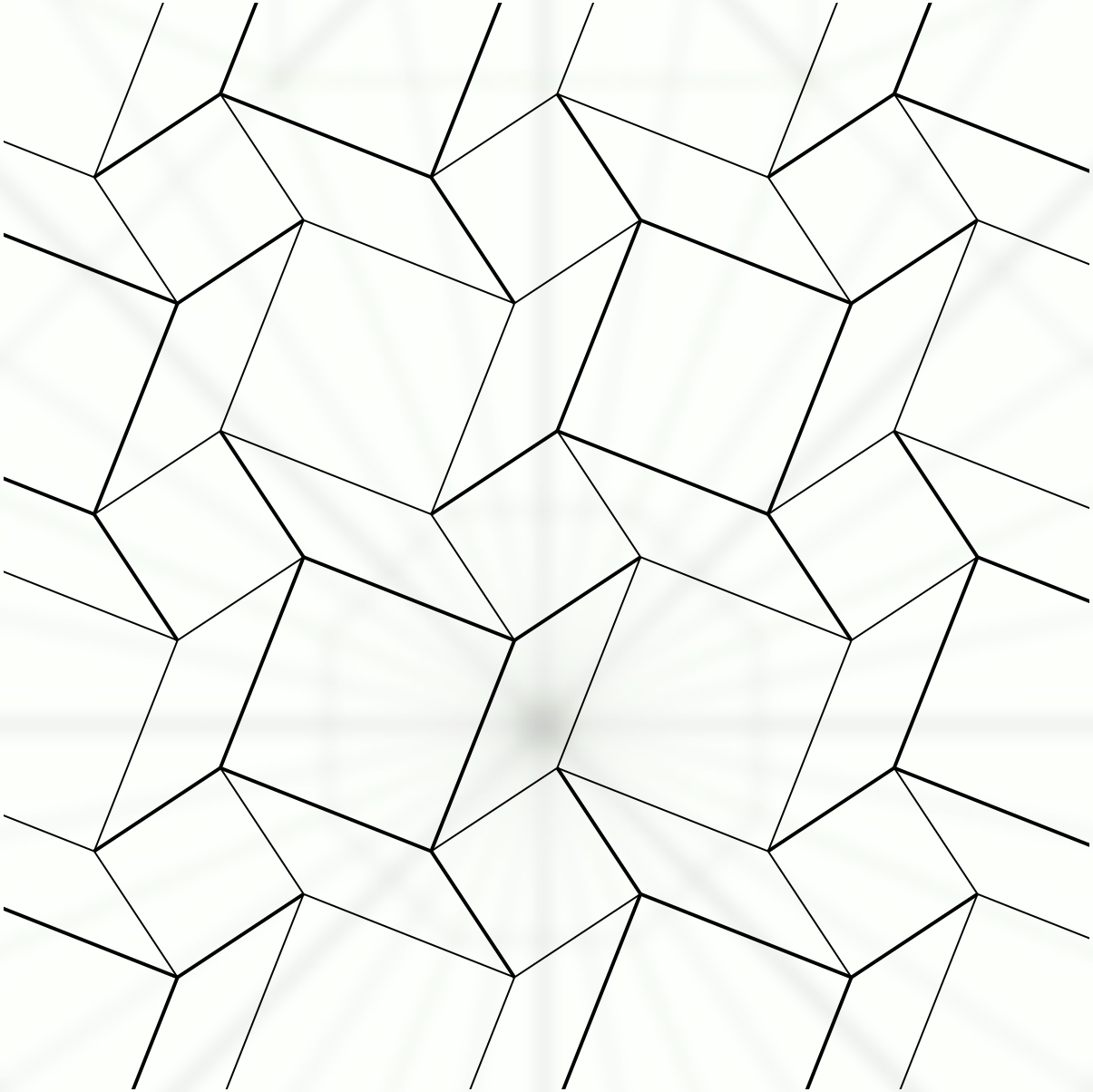
$$\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 + \cdots + \alpha_{2n} = 2\pi$$

To get $2\alpha_1 + 2\alpha_3 + \cdots + 2\alpha_{2n-1} = 2\pi$

Vertex Fold example



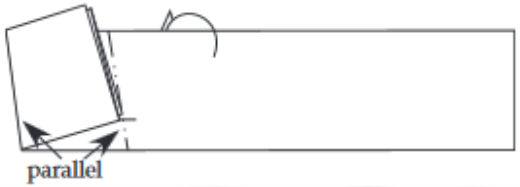
Tesselation folds and Tess



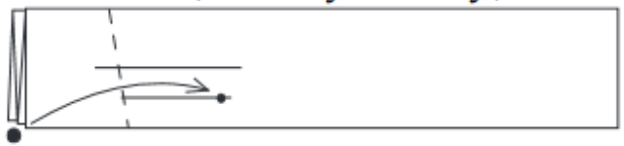
The Miura Map Fold

Japanese astrophysicist Koryo Miura wanted a way to unfold large solar panels in outer space. His fold also makes a great way to fold maps.

(1) Take a rectangle of paper and mountain-valley-mountain fold it into 1/4ths lengthwise.

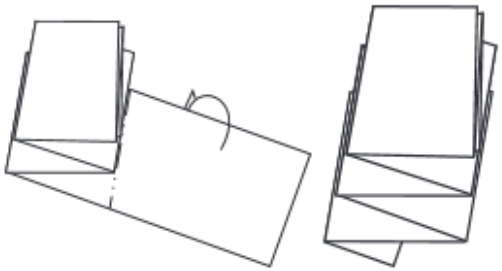


(2) Make 1/2 and 1/4 pinch marks on the side (one layer only) as shown.

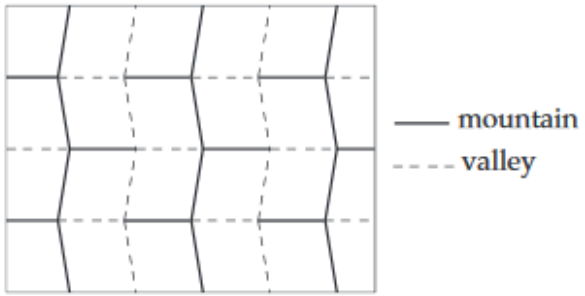


(3) Folding **all layers**, bring the lower left corner to the 1/4 line, as in the picture.

(4) Fold the remainder of the strip behind, making the crease parallel to the previous crease.

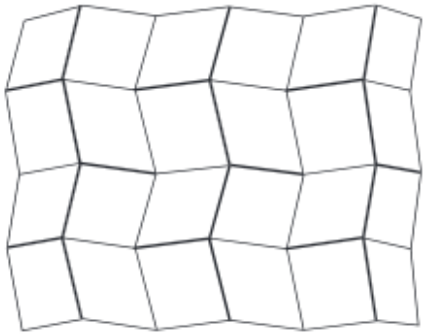


(5) Repeat, but this time use the fold from step (3) as a guide.

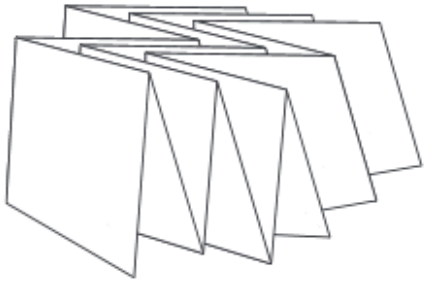


(6) Repeat this process until the strip is all used up. Then **unfold everything**.

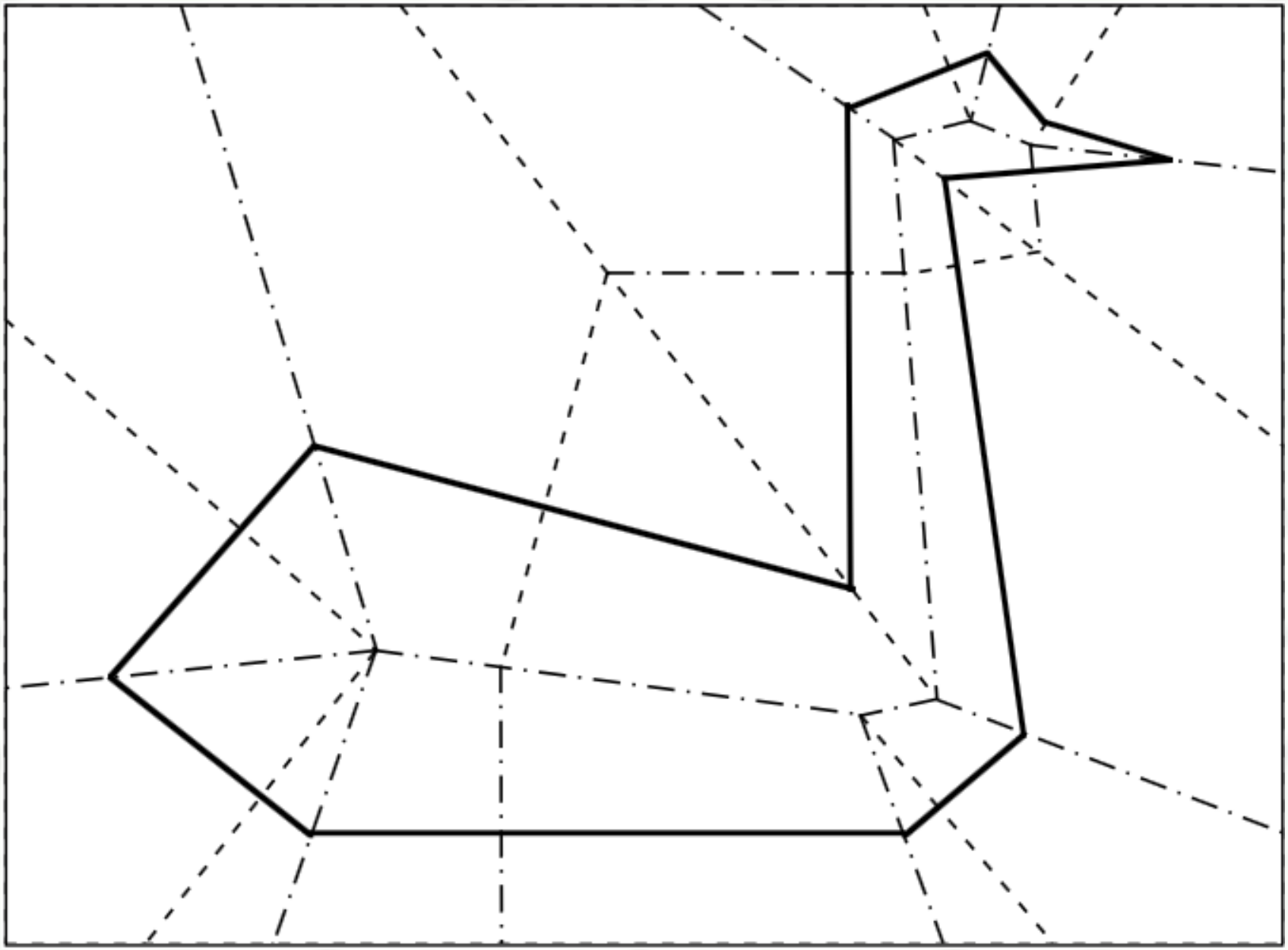
(7) Now re-collapse the model, but change some of the mountains and valleys. Note how the zig-zag creases alternate from all-mountain to all-valley. Use these as a guide as you collapse it...



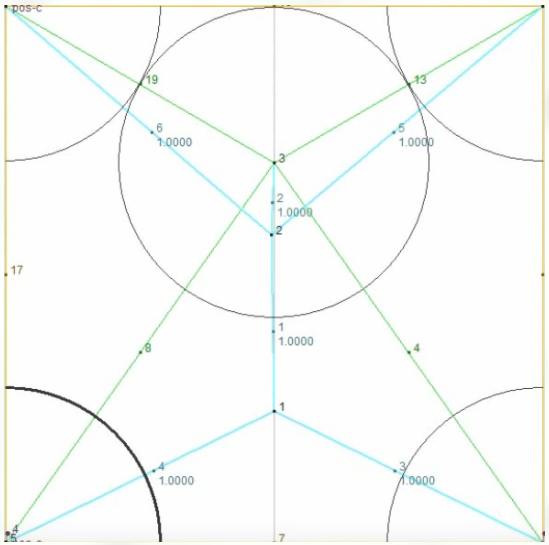
...In the end the paper should fold up neatly as shown to the right. You can then pull apart two opposite corners to easily open and close the model.

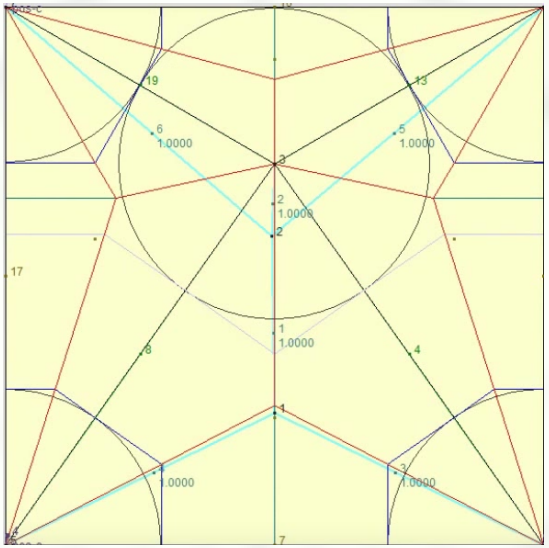


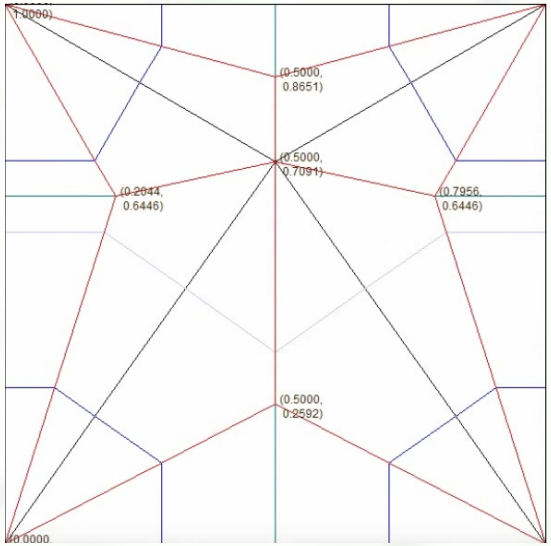
Single Cut Folds

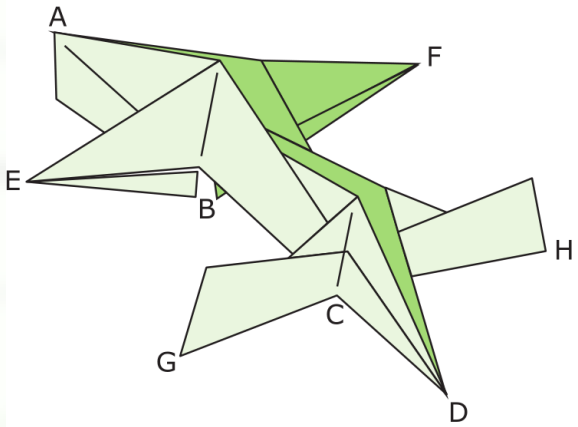


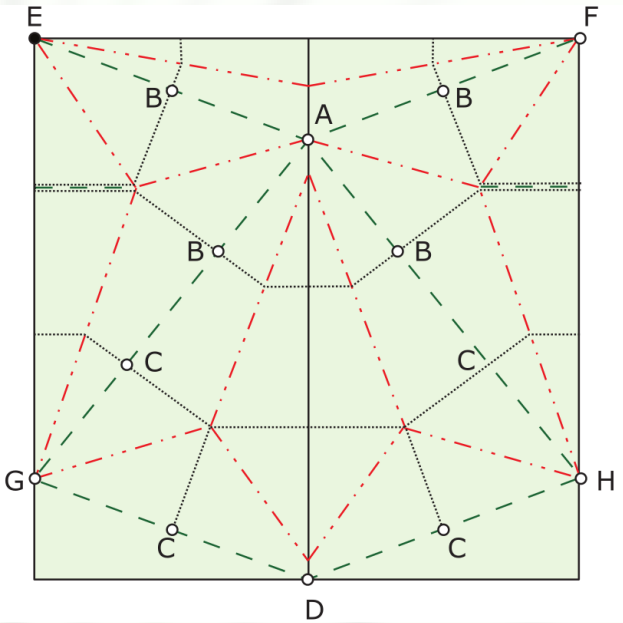
Robert Lang's Treemaker –tutorial





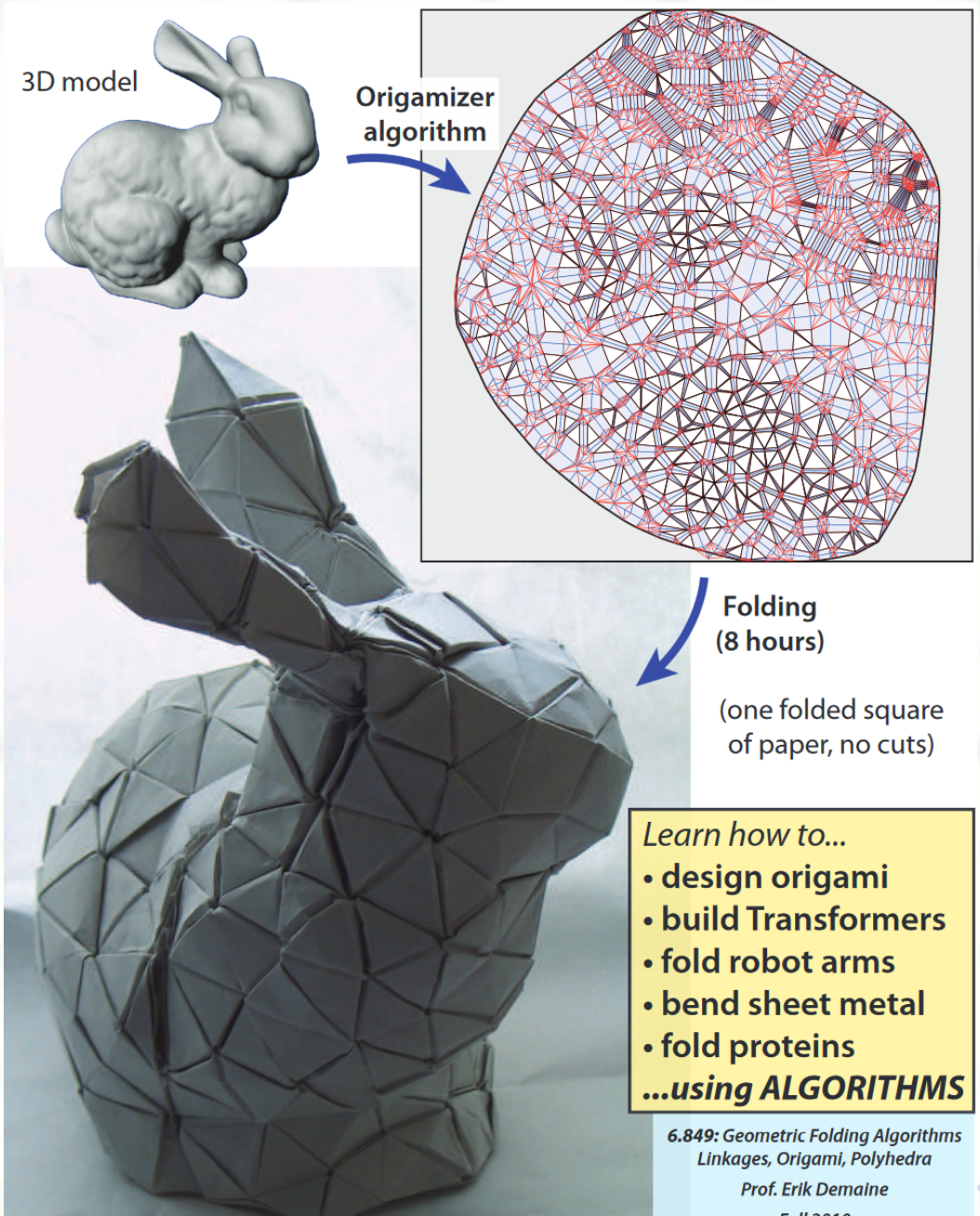






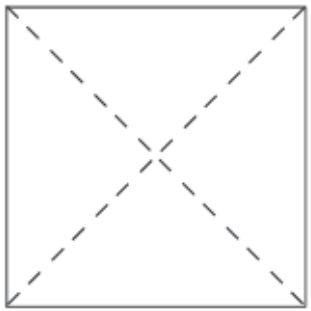
Origamizer:

A practical algorithm to fold any polyhedron.

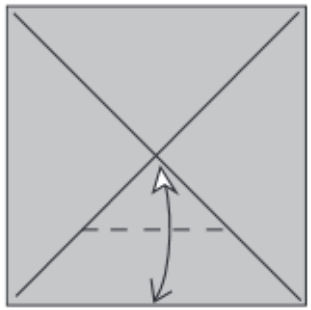


The Hyperbolic Paraboloid

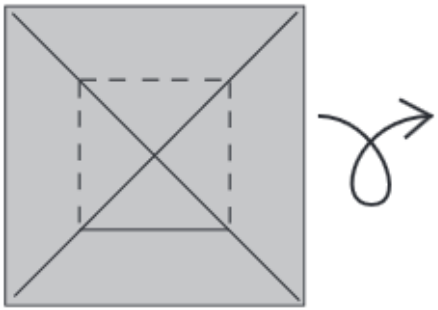
This unusual fold has been rediscovered by numerous people over the years. It resembles a 3D surface that you may recall from Multivariable Calculus.



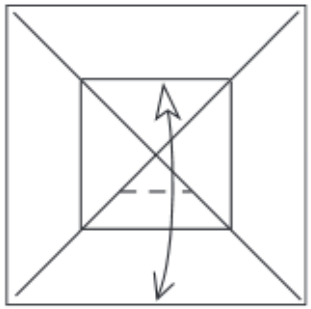
(1) Take a square and crease both diagonals. Turn over.

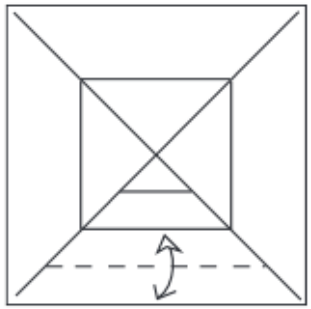


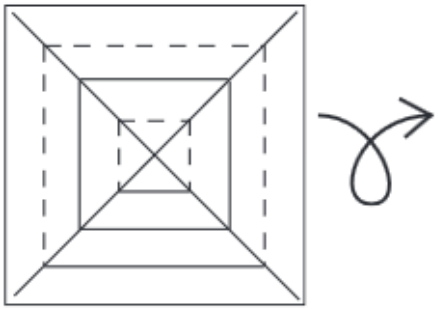
(2) Fold the bottom to the center, but **only** crease in the middle.



(3) Repeat step (2) on the other three sides. Turn over.



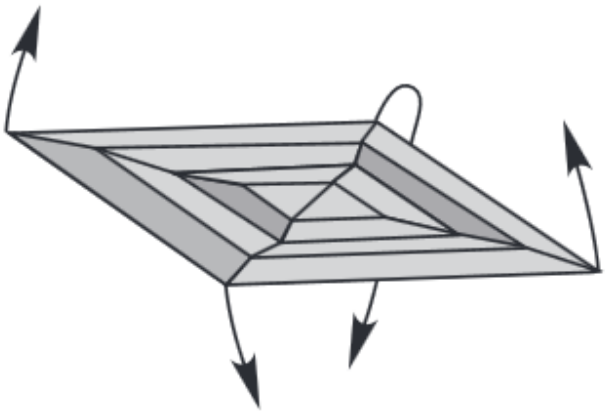


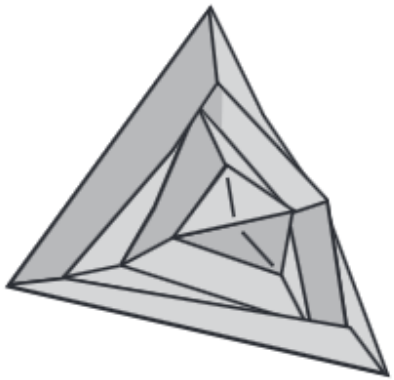


(4) Bring the bottom to the top crease line, creasing **only** between the diagonals.

(5) Then bring the bottom to the nearest crease line. Again, do not crease all the way across.

(6) Repeat steps (4) and (5) on the other three sides. Turn over.





(7) Now make all the creases at once. It may help to fold the creases on the outer ring first and work your way in.

(8) Once the creases are folded, the paper will twist into this shape, and you're done!

John Horton Conway

Erik Demaine (left), Martin Demaine (center), and Bill Spight (right) watch John Horton Conway demonstrate a card trick (June 2005)

Links related to Mathematical Origami

- Pseudopolynomial time.
- Robert Lang's Site
- National Museum of Mathematics Address by Demaine
- Erik Demaine's Papers
- Erik Demaine's: Algorithms Meet Art, Puzzles and Magic
- MIT OCW 6.849
- Handouts
- Tess