INTERACTIVE MANIPULATION OF VIRTUAL FOLDED PAPER

by

JOANNE MARIE THIEL

B.Sc., The University of Western Ontario, 1996

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in

THE FACULTY OF GRADUATE STUDIES
DEPARTMENT OF COMPUTER SCIENCE

We accept this thesis as conforming to the required standard

_____

_____

_____

THE UNIVERSITY OF BRITISH COLUMBIA

October 1998

© Joanne Marie Thiel, 1998

ABSTRACT

The University of British Columbia

INTERACTIVE MANIPULATION OF VIRTUAL FOLDED PAPER

by Joanne Marie Thiel

Origami is the art of folding paper. Traditionally, origami models have been recorded as a sequence of diagrams using a standardised set of symbols and terminology. The emergence of virtual reality and 3D graphics, however, has made it possible to use the computer as a tool to record and teach origami models using graphics and animation.

Little previous work has been done in the field to produce a data structure and implementation that accurately capture the geometry and behaviour of a folded piece of paper. Many difficult and interesting modelling and animation problems arise when trying to develop a visually and geometrically satisfying origami simulation system.

In this thesis, a review of traditional origami and origami design is presented, including origami geometry and computational algorithms. A virtual origami system, including the user interface, has been designed and is described here. A partial implementation of the system has been completed. Modelling, animation and rendering issues are

recognised and solutions are outlined. Finally, future improvements to the system are suggested.

The partial implementation is reasonably successful; major difficulties have been identified and their solutions proposed. This means that a powerful virtual origami system may be possible, continuing work in the direction demonstrated by this thesis.

TABLE OF CONTENTS

Page

Abstract ..... ***ii***

Table of Contents ..... ***iv***

List of Tables ..... ***ix***

List of Figures ..... ***x***

Preface ..... ***xv***

Acknowledgements ..... ***xvii***

Chapter 1 Introduction ***1***

Chapter 2 Background ***5***

2.1 Traditional Diagramming Methods **5**

2.1.1 Overview 5

2.1.2 Terminology 6

2.1.3 Basic folds and symbols 6

2.1.4 Other basic procedures 14

Chapter 3 Framework of Modern Origami ***21***

3.1 Origami Design Basics **21**

3.1.1 Traditional Design 24

3.1.2 Box Pleating 24

3.1.3 Blintzing and Grafting 25
3.1.4 Technical folding 26
3.1.5 Modular Origami 28
3.1.6 Computational Algorithms 29
3.1.7 Origami Design by a Computer? 29
3.2 Origami Geometry 31
3.2.1 Geometry in Paper Folding 31
3.2.2 Huzita's Origami Axioms 32
3.2.3 Mathematics of Flat Origamis 34
3.2.4 Circle Tiling and Packing 36
3.2.5 Planar Graph Theory 38
3.3 Computational Geometry of Origami 39
3.3.1 The Complexity of Flat Origami 40
3.3.2 A Computational Algorithm for Origami Design 41
3.4 Computer Simulation of Paper Folding 42
3.4.1 Previous work 42
3.4.2 Related Areas 46
3.5 Current Research 51
Chapter 4 Program Design 53
4.1 Overview 53
4.2 Functional Requirements 54
4.2.1 Intended Users 54
4.2.2 Basic Functions 55

4.2.3 Desirable Functions 55

4.2.4 Additional Functions 56

4.2.5 Importance of the Basic Symbols and Procedures 58

4.3 Graphical User Interface 61

4.3.1 Overview of the Interface Design 61

4.3.2 General Layout 62

4.3.3 Central Window 63

4.3.4 Toolbars 66

1.1.5 Menu Bar 78

1.1.6 Status Bar 82

1.1.7 Instruction Text Box 83

1.1.8 Crease Pattern Display 83

1.1.9 Tool Tips 84

Chapter 5 Data Structures 85

5.1 Representation of the Origami Model 85

5.1.1 Overview 85

5.1.2 Origami Elements 86

5.1.3 Binary Tree Structure 86

5.1.4 Common properties 88

5.1.5 Faces 88

5.1.6 Edges 91

1.1.7 Vertices 94

1.2 Effects of Adding Creases 95

1.2.1 Joining Existing Vertices 96

1.2.2 Simultaneously Creating New Vertices 96

1.3 Effects of Folding 97

1.3.1 Determining the Moving Faces 97

1.3.2 Moving Faces in Simple Folds 100

1.3.3 Moving Faces in Complex Folds 100

Chapter 6 Results and Evaluation 103

6.1 Jo's Origami Interface (JOI) 103

6.1.1 Implementation Achieved 103

6.1.2 Limitations of the Implementation 106

6.1.3 Programming Language and Toolkits 107

6.2 Difficulties Encountered 108

6.2.1 Folding Flat 108

6.2.2 Rendering of Faces 110

6.2.3 Interpenetrating Faces 112

6.2.4 Stretching of Faces 113

6.3 Future Work 114

6.3.1 A Better Approach to Handling Paper Thickness 114

6.3.2 Animation Using Constrained Geometry 118

6.3.3 Animation Using Curved Surfaces 119

Chapter 7 Conclusion 122

References 123

Appendix A Java Programming *129*

A.1 Java 129

A.2 Java Abstract Windowing Toolkit (AWT) 130

A.3 Java Foundation Classes (Swing) 130

A.4 OpenGL 131

A.5 Magician OpenGL Interface 131

Index *133*

List of Tables

Number *Page*

Table 1. Importance of the basic folds and symbols.....59

Table 2. Importance of the basic procedures.....60

LIST OF FIGURES

Number *Page*

Figure 1. A valley fold. 7

Figure 2. A mountain fold. 7

Figure 3. Fold and unfold. 8

Figure 4. Push here. 8

Figure 5. Repeat. 9

Figure 6. Edge configurations. 9

Figure 7. Watch this spot. 10

Figure 8. Rotate. 10

Figure 9. Equal distances. 11

Figure 10. Equal angles. 11

Figure 11. Fold over and over. 11

Figure 12. Turn the paper over. 12

Figure 13. Pull paper out from here. 12

Figure 14. Cut-away view. 13

Figure 15. X-ray line. 13

Figure 16. Next view larger. 13

Figure 17. Hold here and pull. 14

Figure 18. Waterbomb base. 14

Figure 19. Preliminary fold. 15

Figure 20. Rabbit-ear.....16

Figure 21. Inside reverse.....16

Figure 22. Outside reverse.....17

Figure 23. Crimp.....17

Figure 24. Squash.....18

Figure 25. Petal fold a point.....19

Figure 26. Petal fold an edge.....19

Figure 27. Open sink.....20

Figure 28. Closed sink.....20

Figure 29. A simple crease pattern 22

Figure 30. A mountain crease (a) and a valley crease (b). 22

Figure 31. The four traditional origami bases, together with their crease patterns, are a) kite base, (b) fish base, (c) bird base and (d) frog base.....23

Figure 32. (a) The blintz fold. (b) Crease pattern for the blintzed bird base. 26

Figure 33. The triangle in (a) is found in successively smaller sizes in the crease patterns of the four fundamental bases: two in the kite base (b); four in the fish base (c); eight in the bird base (d); and sixteen in the frog base (e). 27

Figure 34. The true building blocks of technical origami, (b) and (c), can be found in the repeated triangle (a) of the traditional bases.....28

Figure 35. The traditional flapping bird (a) lies flat when complete. The crease pattern (b), for the figure (c), which does not lie flat, is not flat foldable.....34

Figure 36. (a) Stick figure for a six-legged base and (b) the preliminary crease pattern for the base, with tiled circles. 37

Figure 37. The cube (a) and its planar graph representation (b). 38

Figure 38. A capped cube (a) and its corresponding planar graph (b) 39

Figure 39. Two flat origamis with the same crease pattern (a), but different overlap orders (b) and (c) 40

Figure 40. A polyhedron (a) and its corresponding folding net (b). 45

Figure 41. In the final step of the traditional origami crane, the wings are curved and the body is stretched so that the model becomes 3-dimensional. 51

Figure 42. The graphical user interface 61

Figure 43. The main toolbar 67

Figure 44. The folding toolbars 68

Figure 45. The landmark tools 75

Figure 46. The camera controls 76

Figure 47. The animation controls 76

Figure 48. The menu bar 78

Figure 49. The status bar. 82

Figure 50. The instruction text box. 83

Figure 51. The crease pattern display. 83

Figure 52. A tooltip. 84

Figure 53. A piece of paper that has been folded twice. 87

Figure 54. The face tree for the folded paper in Figure 53. 87

Figure 55. The set of edge trees for folded paper in Figure 53. 87

Figure 56. The detailed face nodes for this example of a divided face.....89

Figure 57. The direction of an edge is independent of its order among all edges around a given face. Note edge e5.....90

Figure 58. Vertices are always specified in clockwise order around the white side of the face.....91

Figure 59. Two complete edge trees for this example. The three other edge trees are not shown.....92

Figure 60. The order of an edge's two adjacent vertices and two adjacent faces.....93

Figure 61. The vertex table for this example.....94

Figure 62. Animation of a simple fold.....100

Figure 63. A general reverse fold, done by reflecting through the common plane.....101

Figure 64. A squash fold using only a reverse fold followed by a valley fold.....102

Figure 65. A full screen capture of a completed origami model.....105

Figure 66. Three views of the paper airplane. Normal shaded paper airplane in (a), shown with vertices and edges in (b), and wireframe only in (c).....106

Figure 67. The original paper (a) is folded to exactly 180° in (b), but only 179° in (c).....109

Figure 68. Paper (a) folded to exactly 179° compared to 180° with simulated thickness in (b).....110

Figure 69. Convex polygons (a) can be rendered using OpenGL, but concave polygons (b) are not guaranteed to be rendered correctly.....111

Figure 70. Vertices of a convex polygon that are coplanar (a) can drift from the plane as in (b). If the resulting polygon (c) is viewed from the side, a concave shape may be formed (d).....112

Figure 71. Different cases illustrating possible penetration of non-contiguous faces. The faces marked "x" will be affected by the moving faces. 113

Figure 72. Paper (a) folded to 180° with simulated thickness, and a close-up view (b).....115

Figure 73. Paper folded in half twice (a) causes many intersections of strips (b) that have to be merged.....116

Figure 74. A close up of the corner of the folded paper shown in Figure 73(a).....117

Figure 75. Two-dimensional plan view of paper folded in half three times.....118

Figure 76. An internal angles around the central vertex is marked with a single line in (a). A dihedral angle is marked with a double line in (b). 119

PREFACE

This document summarises my research and is divided into the following sections:

Chapter 1, Introduction, discusses the motivation for this thesis, and is an introduction to the entire project.

Chapter 2, Background, gives an introduction to origami (i.e. paper folding). Traditional well-known origami diagramming methods are described. The information in this chapter is assumed knowledge through the rest of the thesis.

Chapter 3, Framework of Modern Origami, is a review of the literature pertaining to origami, including the development of origami design and analysis as well as computer simulation of paper folding and related problems.

Chapter 4, Program Design, describes the design of a program for simulating paper folding on a computer. This includes the functional requirements and user interface design of such a system.

Chapter 5, Data Structures, presents the data structures that were developed to implement the origami system, and explains what happens to the data structures as folds occur on the model.

Chapter 6, Results and Evaluation, describes the actual implementation achieved, problems and issues encountered during development, and future research possibilities.

Chapter 7, Conclusion, reflects on the results of the project.

Following the last chapter is **Appendix A, Java Programming**, which describes the various Java graphics libraries that were used in the implementation.

Java is a registered trademark of Sun Microsystems Inc.

OpenGL is a registered trademark of Silicon Graphics, Inc.

Windows 95 is a registered trademark of Microsoft Corp.

ACKNOWLEDGEMENTS

I would like to take this opportunity to thank my supervisors Maria Klawé and Alain Fournier for their support, help and patience. Thanks also to Jack Snoeyink (my second reader), Tom Hull, Robert Lang, Dave R. Howe, Christopher Moreno, Dave Forsey, Joseph Wu, the members of the local Vancouver origami club (PALM: Paperfolders Around the Lower Mainland), the rec.arts.origami newsgroup and the origami-l mailing list for their enthusiastic interest and helpful suggestions.

The Internet has been indispensable during research and development of this project. Joseph Wu's Origami Page* is the definitive resource for origami, and includes both traditional and mathematical paper folding. Tom Hull's Origami Math* pages are the best source for references to obscure origami mathematics papers. For quick and clear solutions to forgotten geometry and mathematics formulae, "Ask Dr. Math"*. The archives of this web site provided the answers to all the math questions that arose while I was writing code.

Many of the diagrams for origami symbols and folding procedures were made after figures in Robert Lang's "Origami Zoo" and "The Complete Book of Origami." Diagrams for Huzita's origami axioms were made after those presented in Tom Hull's Origami Math pages.

This research was funded in part through Graduate Scholarships from the Natural Sciences and Engineering Research Council (NSERC PGS-A), and the British Columbia Advanced Systems Institute (ASI).

Finally, thanks to my parents, Bill and Louise Thiel, for understanding when I left home to travel half-way across the country for my Master's degree. And for their support and encouragement, not only during the past two years, but all my life.

* The Internet address for this site is in the References section.

Chapter 1

INTRODUCTION

Ever since the very first sheet of paper was crumpled into a ball, people have been folding paper. Paper folding, or origami, is truly an art that is as old as the medium itself. A plain piece of paper can be transformed into a geometrically and aesthetically beautiful sculpture by the thoughtful placement of a few simple folds. And since origami is something to be shared, people have been using various methods to record the folding process involved in creating their models.

The preferred method of learning a new origami model is through demonstration by the inventor. However, due to the small number of origami designers, this severely limits the number of people who will ever learn a particular model. In addition, the model would not likely survive past the death of its inventor if this were the only method of passing a design along.

Many origami designers choose to record their models using a series of annotated diagrams. There exists a system of lines and arrows that is standardised and well

accepted throughout the origami community. This is the most common way to teach and learn origami, since books are generally easily accessible. However, two-dimensional diagrams, even with textual notation, can never capture the folding process of a three-dimensional object exactly.

Another way of passing origami models between people is through natural language descriptions only. This method is also known as "phone folding" (British Origami Society, 1998). With the introduction of text-based email on the Internet, a natural language description of origami has become even more desirable. Using a formal language would be even more advantageous, since its use would readily lend itself to implementation on a computer. However, natural language descriptions of origami have been found to be imprecise and verbose (Fisher, 1996).

Returning to the idea of using a visual means to teach and record origami, some attempts have been made to videotape people folding models. This method has the advantage of showing exactly what happens to a real three-dimensional piece of paper when it is folded. Unfortunately, the tape plays back at a set speed and the viewer may not follow along quickly enough to keep up the pace. People have to look up from their piece of paper in order to rewind or pause the tape.

The most recent work has been done to take advantage of the power of computer modelling and animation to diagram origami instructions. The computer allows an accurate description of the origami model, and can produce animation between steps.

Computer animation can be played at any speed, and either forward or backward. A computer program would also allow the user to change an existing folding sequence, or create a new one entirely from scratch.

The problem of designing such a system is a very interesting and challenging one. No results have yet been produced that completely model and animate folded paper in an acceptable manner. The fundamental problem is to model a specific class of two-dimensional surfaces embedded in three dimensions. A physically-based model seems necessary to simulate the thickness, elastic properties and other qualities of the paper. However, such an approach results in a model that is no longer truly two-dimensional. The model must also allow animation while avoiding self-penetration and allowing parallel layers to get arbitrarily close. The result will be a very complicated model, with complex manipulation and collision detection algorithms.

Other researchers have attempted to design and implement such a model, with varying results that are presented in the following pages. As an origami enthusiast, I found that regardless of the model chosen, the user interface to each such system was, in general, unsatisfactory. In each system, it seems that the user chooses part of the model with the mouse, drags it around on the screen, and releases the mouse button when the desired result is achieved. However, in practice folds require *pre-creasing*, i.e. indicating crease locations before the final fold is ever done. The system presented in this thesis takes this novel approach to the graphical user interface design.

The interface designed here for an origami manipulation system takes care to use the traditional diagramming symbols throughout. The symbols are well known and fairly intuitive, and comments from origami folders indicate that using familiar symbols is a good idea. As well, folds and creases may be added to the model by simply joining existing vertices, rather than dragging flaps of virtual paper around.

The chosen theoretical model for the folded piece of paper in this implementation is a variation on those presented previously in the literature. Several other areas of computer graphics and modelling have been explored to suggest an entirely new model and are also discussed in the following pages.

Chapter 2

BACKGROUND

Origami is the art of folding paper into sculpture. Traditionally, folding is restricted to a single square piece of paper, often coloured on one side, and without the use of scissors or glue. Despite these restrictions, objects of rich variety and complexity can be realised from a sequence of folds on the square.

2.1 Traditional Diagramming Methods

2.1.1 Overview

In the 1950s, Japanese master folder Akira Yoshizawa developed a systematic code of dots, dashes and arrows to diagram his origami models. Since then, this method of diagramming, with minor alterations, has become a standard in origami books. The process of folding a particular model is broken down into logical steps. The steps are explained with a diagram of the model at that point, and symbols indicating what folds are to be done next. The diagrams are often accompanied by textual descriptions of each step for clarification, but it is possible to work from the diagrams alone.

The following sections describe the basic terminology, folds and symbols used in origami. One example of each procedure is given, but the actual fold is general and may occur on any configuration (e.g. on multiple layers) of the paper.

Readers familiar with standard origami diagrams and symbols may wish to skip the rest of this chapter.

2.1.2 Terminology

Edges are straight lines that are formed when the paper is folded. The boundary edges of a piece of paper are called *raw edges*. A *flap* is a section of the paper that may be moved and folded without affecting the other parts of the paper.

2.1.3 Basic folds and symbols

2.1.3.1 Valley Fold

The valley fold is formed when a flap of paper is folded to form a trough, or valley shape relative to the viewer. The valley fold is the most common fold, and is represented with a dashed line locating the position of the crease, and a plain arrow indicating the direction of motion of the flap.

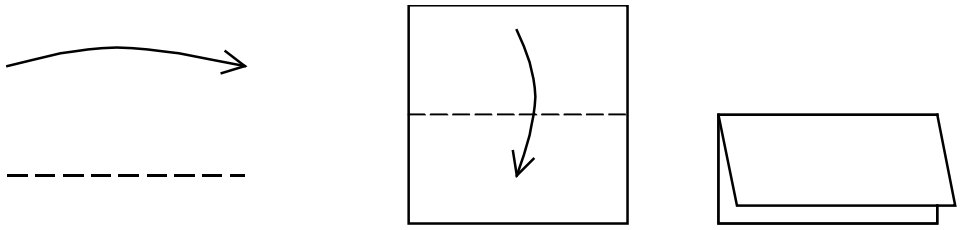


Figure 1. A valley fold.

2.1.3.2 Mountain Fold

A mountain fold is formed when a flap of paper is folded away from the viewer, forming a peak, or mountain-shaped crease. The mountain fold is indicated with a dot-dot-dash line, and an arrow with a one-sided hollow head to indicate the flap and direction to be folded. A mountain fold looks the same as a valley fold from the other side of the paper.

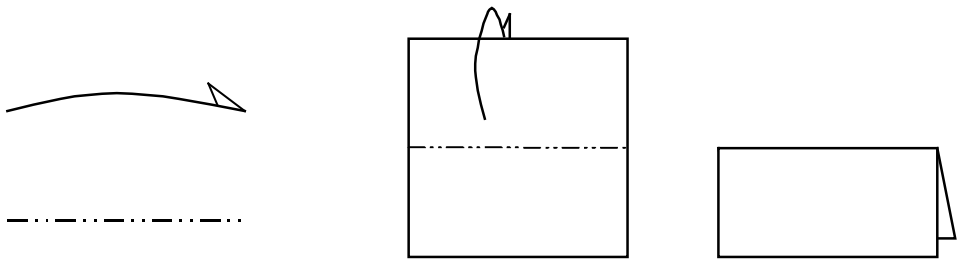


Figure 2. A mountain fold.

2.1.3.3 Fold and unfold

An arrow that doubles back upon itself means to fold and unfold as indicated, leaving a crease. The crease is marked with a finer line that does not quite reach the adjacent edges at its ends.

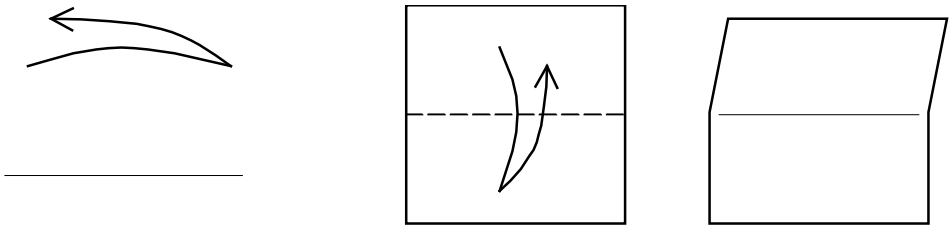


Figure 3. Fold and unfold.

2.1.3.4 Push here

A small, hollow arrow with a flat tail indicates "push here." This type of symbols is used with such folds as the Reverse Fold and Petal Fold (e.g. see §2.1.4.4 and §2.1.4.8), and means that the paper is pushed in symmetrically, rather than being folded towards or away from you.

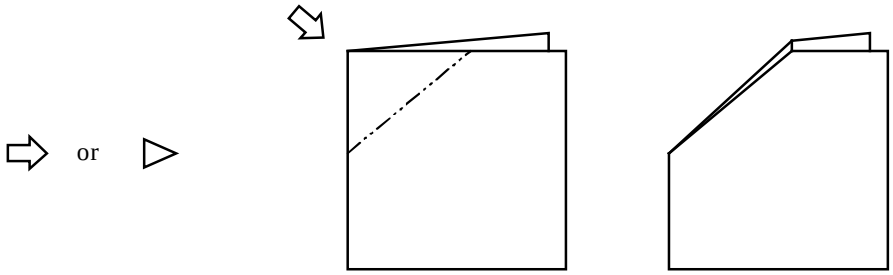


Figure 4. Push here.

2.1.3.5 Repeat

Instead of explicitly diagramming every fold, sometimes the repeat symbol is used to indicate a repetition of a fold on other similar layers. The repeat symbol is an arrow pointing at the location of the new folds, crossed with hatch marks to indicate the number of times to do the repetition.

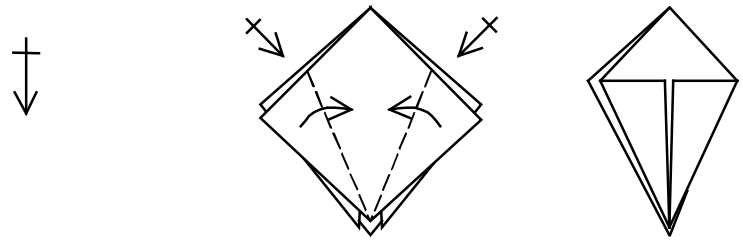


Figure 5. Repeat.

2.1.3.6 Edge configurations

When working with a flap with several layers, there are sometimes several ways in which they can be folded. In these cases, zigzag lines are drawn beside the model to indicate the order of folding.

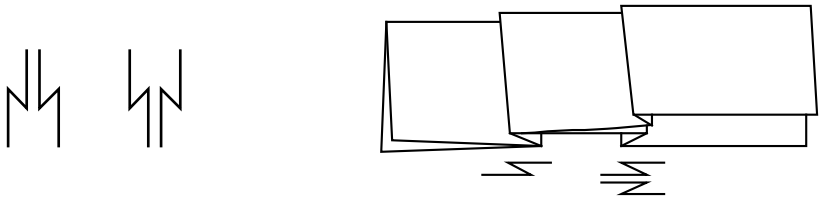


Figure 6. Edge configurations.

2.1.3.7 Watch this spot

If a new or unusual step is occurring, the "x" marks are used to mark a certain part of the paper, and show where it ends up in the next diagram. This helps to clarify the movement of the paper.

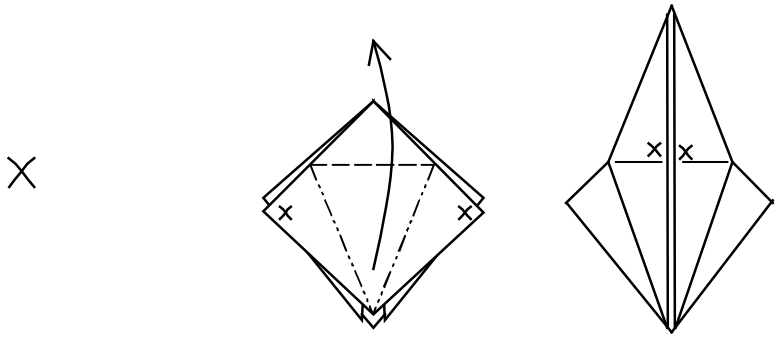


Figure 7. Watch this spot.

2.1.3.8 Rotate

Rotation of the model, in the plane of the page, is indicated with a circle and two arrowheads on it. The arrows show the direction of the rotation, and the spacing of the arrowheads indicates the angle (e.g. either "90°" or "½ turn" (180°)).

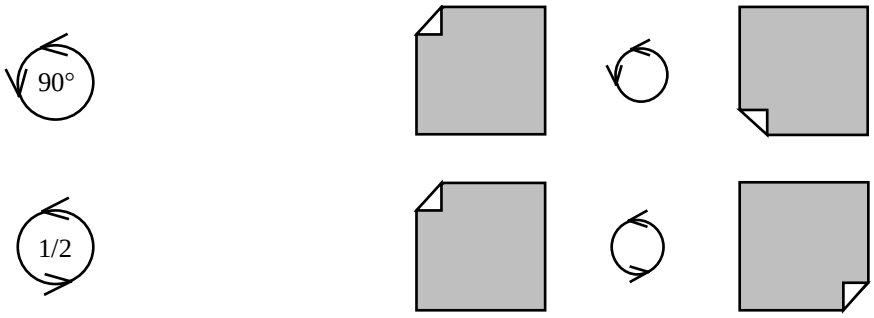


Figure 8. Rotate.

2.1.3.9 Equal distances

This symbol indicates when folds are to be made so that two or more distances are equal.

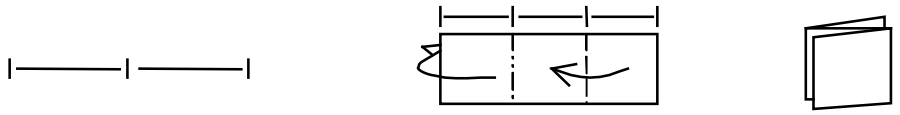


Figure 9. Equal distances.

2.1.3.10 Equal angles

If two or more angles are to be made equal, they are indicated with similar marks.

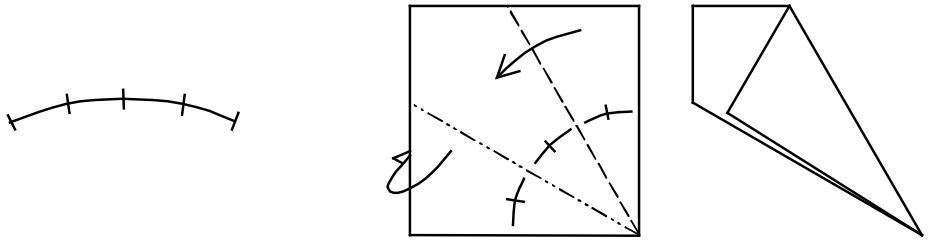


Figure 10. Equal angles[1].

2.1.3.11 Fold over and over

An arrow that touches down on the paper more than once indicates to make one fold, then fold the paper over and over as many times as the arrow touches.

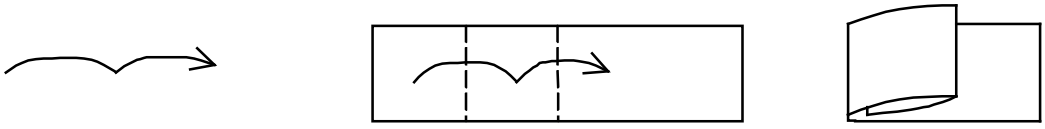


Figure 11. Fold over and over.

[1] Note that it is possible to trisect an angle exactly using origami techniques. For more details, see Hull's explanation of Hisashi Abe's paper folding method of trisecting an angle developed in the 1970s (Hull, 1996).

2.1.3.12 Turn the paper over

An arrow that makes a loop means to turn the entire model over. The direction of the arrow indicates the direction of the flip.

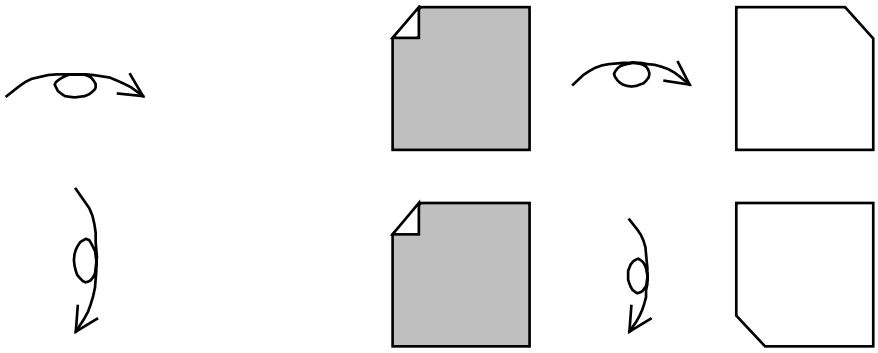


Figure 12. Turn the paper over.

2.1.3.13 Pull paper out from here

A hollow arrow with an indented tail indicates to pull some paper out from an interior pocket or unfold some paper.

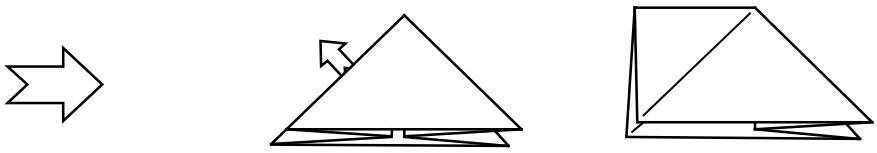


Figure 13. Pull paper out from here.

2.1.3.14 Cut-away view

A jagged outline is used to show hidden layers of paper, as if the top layers were torn away.

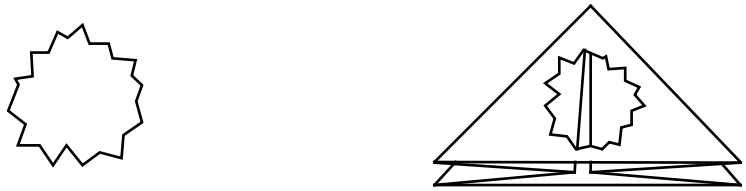


Figure 14. Cut-away view.

2.1.3.15 X-ray line

A dotted line is used to show a fold or edge that is hidden. Sometimes the X-ray line will also be used to indicate the position an edge will take in the next step.

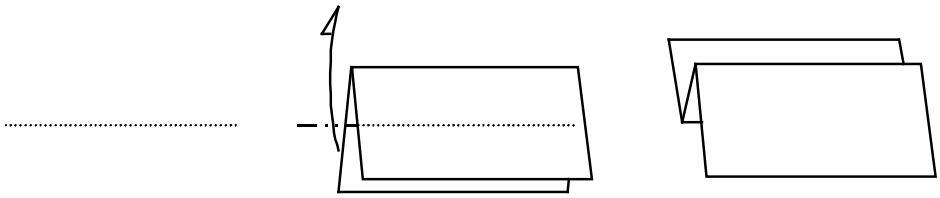


Figure 15. X-ray line.

2.1.3.16 Next view larger

A hollow arrow with a large head and tiny tail is used to indicate that the next drawing represents an enlarged view of the model.

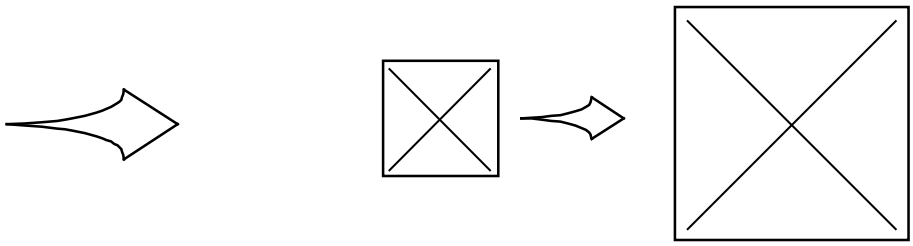


Figure 16. Next view larger.

2.1.3.17 Hold here and pull

A small circle with an arrow attached to it means to hold the paper at the indicated spot, and pull in the direction of the arrow.

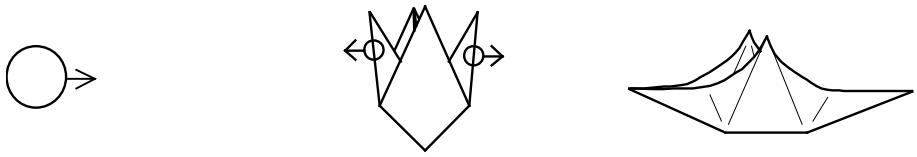


Figure 17. Hold here and pull.

2.1.4 Other basic procedures

2.1.4.1 Waterbomb base

The waterbomb base is a basic shape used in many traditional folds. First, the paper is folded in half vertically and horizontally. Next, the paper is turned over and folded along both diagonals. The middle of all four sides are then brought together, and the paper is flattened with two flaps going in each direction.

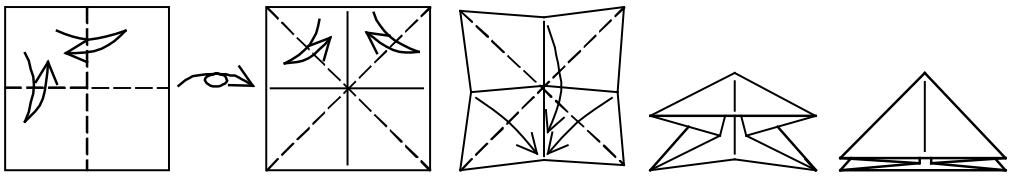


Figure 18. Waterbomb base.

2.1.4.2 Preliminary fold

The preliminary fold is another basic fold, from which other bases are made. It can be made by simply turning a waterbomb base inside out. Crease the diagonals of the paper first. Then turn the paper over and crease the paper horizontally and vertically. This time, bring the corners of the paper together and flatten symmetrically.

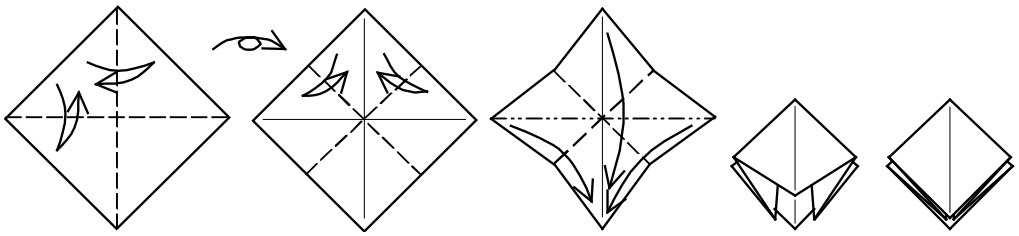


Figure 19. Preliminary fold.

2.1.4.3 Rabbit-ear

The rabbit-ear fold is a way of making a flap narrower, and changing its direction. It is made with three valley folds that meet at a point, and a mountain fold that also joins the point. Often, the first three creases are made, and the mountain crease forms as the paper is flattened. The new flap may be made in either direction, depending on the location of the mountain crease. Rabbit-ears may be made on any flap of paper, not necessarily a triangular one.

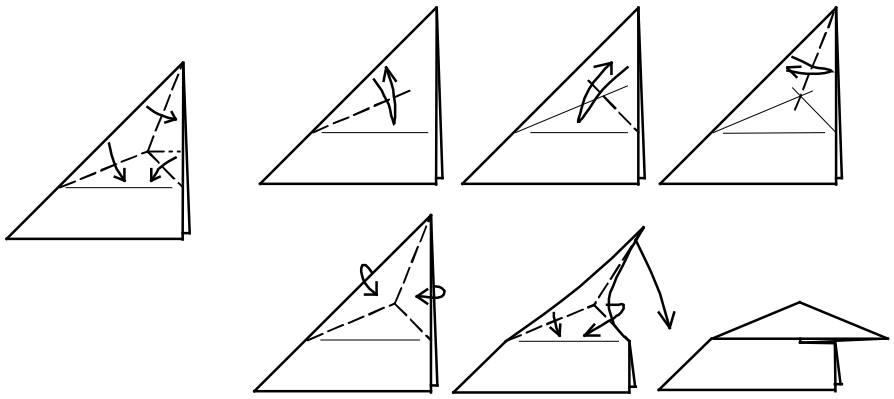


Figure 20. Rabbit-ear.

2.1.4.4 Inside reverse

An inside reverse fold is a way of changing the direction of a flap of paper in a way that makes the fold more permanent than a simple valley or mountain fold. An inside reverse fold is indicated by a mountain fold on the near layer of paper, and a matching valley fold on the far layer. There is also a push arrow pointing to the spine of the fold.

In general, the paper is pre-creased, and then the tip is pushed inside of the flap, turning inside-out in the process. The paper is flattened.

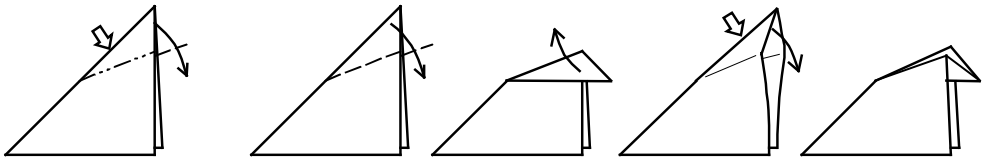


Figure 21. Inside reverse.

2.1.4.5 Outside reverse

An outside reverse fold is similar to an inside reverse, but the flaps wrap around the outside of the tip, rather than being pushed inside. The near layer of the paper is creased with a valley fold, and the far layer is indicated with a mountain fold. The arrows show the direction of motion of the paper.

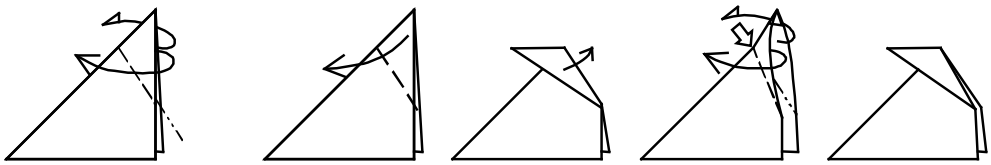


Figure 22. Outside reverse.

2.1.4.6 Crimp

A crimp is used to change the direction of a flap or point, without turning the paper inside-out. They are often used to make features such as beaks, knees, ankles and ears. This is just one version of the crimp fold. They may also be made to be straight or to curve upwards.

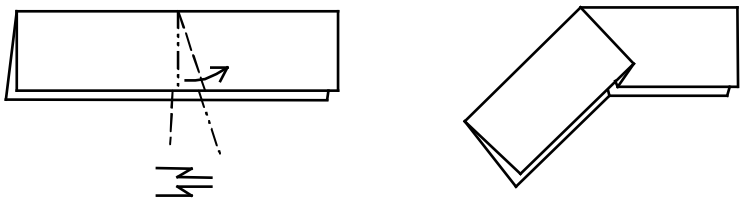


Figure 23. Crimp.

2.1.4.7 Squash

The squash fold is a way of converting one flap into two. It is indicated with valley and mountain folds, and an arrow indicating the folded edge to be squashed. The folded edge often becomes unfolded during a squash fold, and is lined up with the fold line at the base of the flap.

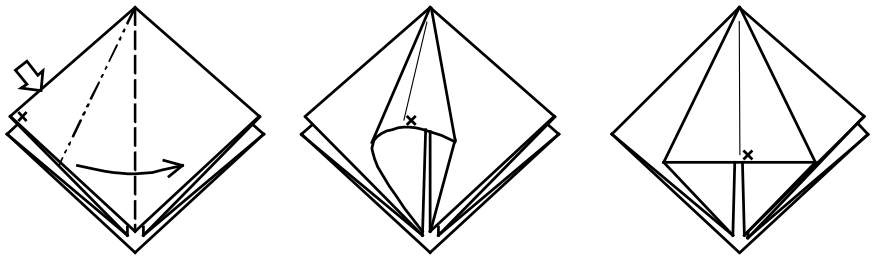


Figure 24. Squash.

2.1.4.8 Petal fold a point

A petal fold simultaneously narrows and lengthens a point. It is indicated by two mountain folds and a valley fold which form a triangle. There are also two push arrows indicating which flaps are pushed in. The two mountain folds are nearly always angle bisectors, and should be creased first. Next, the valley fold crease is made to connect the ends of the mountain creases. The lower tip is then pulled upward along the valley fold just made, and the two side flaps turn inside out and move inward. As the new flap is flattened, the mountain folds on the top flap become valley folds.

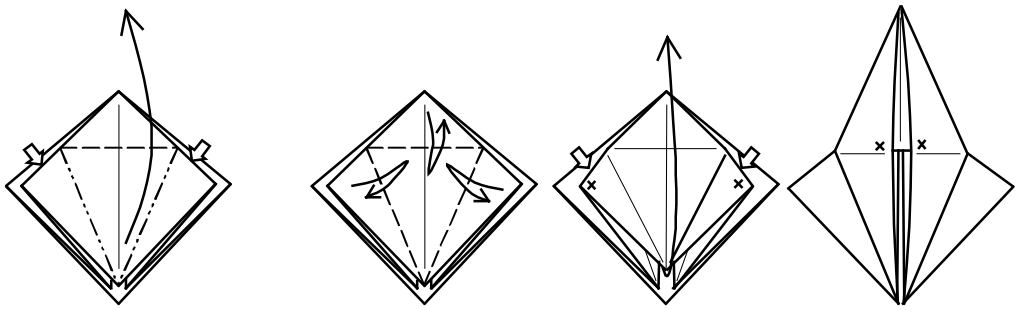


Figure 25. Petal fold a point.

2.1.4.9 Petal fold an edge

The petal fold is used to make a point out of an edge. The angle bisectors at the bottom are made first, then the valley fold to join them (as when petal folding a point). Then fold up along the valley fold, allowing the two sides to come inward. The two other creases appear as you flatten the flap to a point.

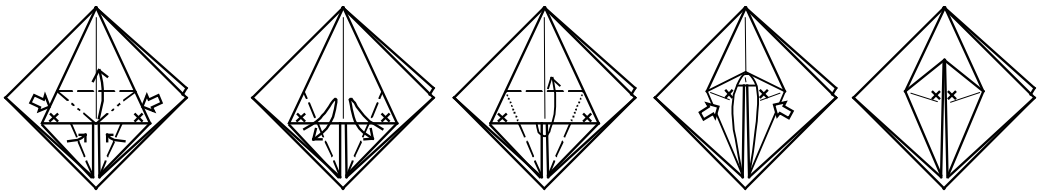


Figure 26. Petal fold an edge.

2.1.4.10 Open sink

An open sink is a way to blunt a point that has no raw edges. It is indicated with a valley fold and an arrow locating the point that is to be sunk. To make an open sink, first make the valley fold, then partially open the point until the middle of the point is

flattened. Push the tip downwards and close the paper. Some of the folds change from valley to mountain, and should be adjusted as needed.

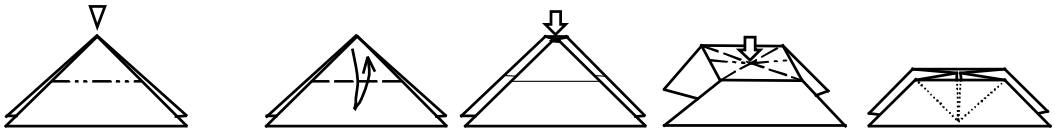


Figure 27. Open sink.

2.1.4.11 Closed sink

A closed sink is another way of blunting a point, but this time the interior edges are locked together inside the model. The point is creased as in the open sink, but instead of opening the paper out flat, all layers except one are held together, and the point is tucked in the last layer. This is the most difficult type of fold described here, since most creases are reversed in type and the paper must be bent and coaxed to fit into the pocket.

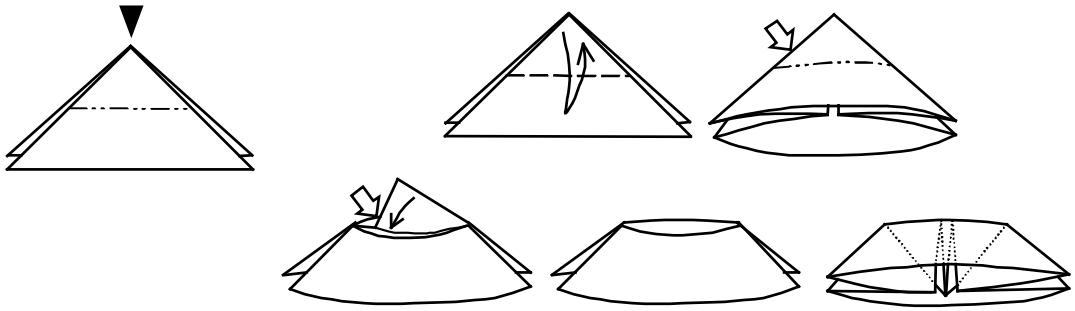


Figure 28. Closed sink.

Chapter 3

FRAMEWORK OF MODERN ORIGAMI

Origami has traditionally been designed by trial and error, creating variations on existing models. More recently, computational algorithms, notably by Lang (Lang, 1996) have been implemented to let the computer help design origami.

3.1 Origami Design Basics

There are two main parts to the creation of an origami model: fold the base, and then fold the details. A *base* is a regular geometric shape that has an overall structure similar to that of the final model. For example, the base for an animal may have six major flaps: four for the legs, one for the head, and one for the tail. Additional folds called *detail folds* are used to transform the base into the final model.

If we unfold a completely folded base (or any other origami model), a division of the paper into a set of polygonal regions by a set of line segments[2] is left on the paper. Each line segment is called a *crease*. This pattern of lines on the paper is called the *crease*

2 This is a specific case. In general, some models involve curved creases, but those are not discussed here.

pattern (Figure 29) of the base. Each crease may be either "mountain" or "valley" (Figure 30), but when looking at the crease pattern the orientation often does not matter, it is just the pattern of lines that is interesting.

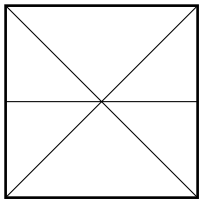


Figure 29. A simple crease pattern

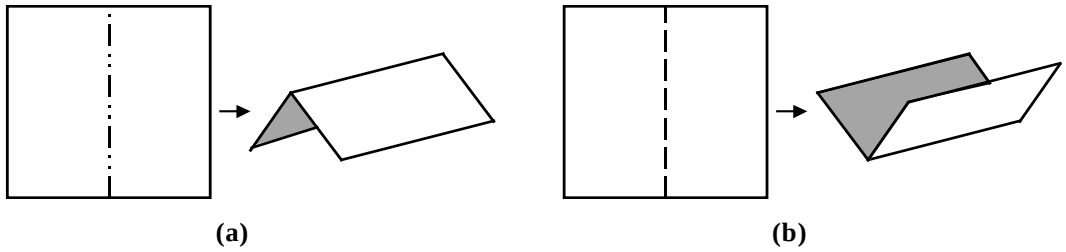


Figure 30. A mountain crease (a) and a valley crease (b).

The four traditional fundamental bases are shown in Figure 31. The major points of a base become major appendages of the final model. In English, the bases are named for the general animal shape best represented by the number of flaps. The kite, fish, bird and frog bases have one, two, four and five large points and one, two, one and four small points respectively. As an example, the fish base's two large points would correspond to a fish's head and tail, and the two smaller points may get turned into

pectoral fins. Most of the two to three hundred well-known designs up until the 1960s were created from these four bases.

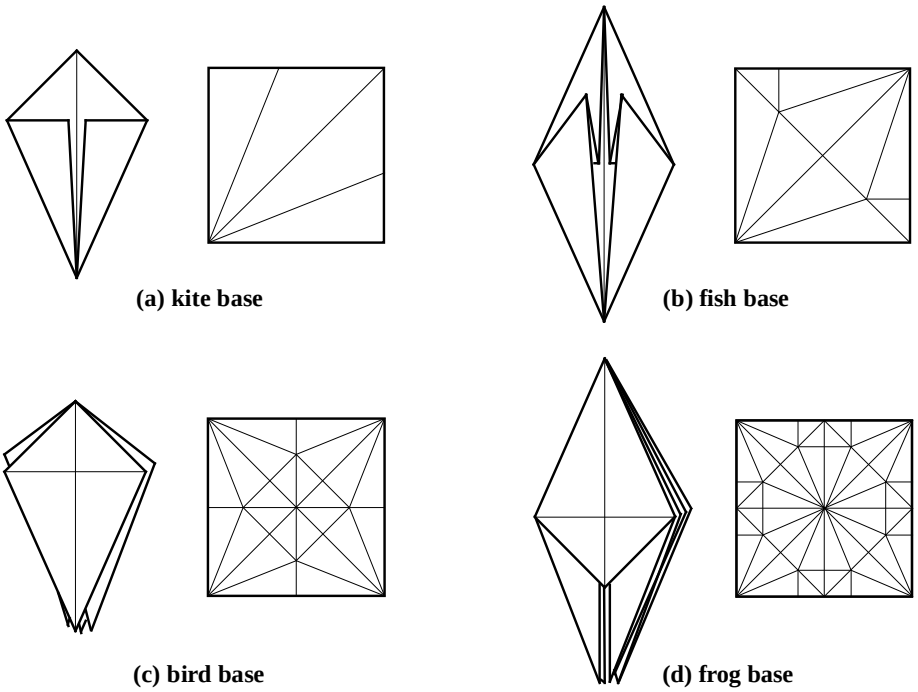


Figure 31. The four traditional origami bases, together with their crease patterns, are a) kite base, (b) fish base, (c) bird base and (d) frog base.

However, the four traditional bases do not cover all classes of figures to be modelled. Consider modelling a spider, which requires eight large points for the legs and one small point for the head. To solve design problems such as these, origami enthusiasts began to explore new systematic design methods. Modern advancements in the complexity of origami reveal a rich geometric structure governing the possibilities of paper folding.

3.1.1 Traditional Design

The traditional design method for origami models depends on the four fundamental bases, and the creativity of the designer to add the features or details particular to the subject being modelled (Randlett, 1966; Kasahara, 1967; Maekawa and Kasahara, 1983; Kasahara, 1987; Lang, 1988b). In general, designers make alterations to existing models.

In the traditional sense, a good model is one that captures the essence of the subject with a few carefully placed detail folds and an uncomplicated folding sequence. For examples of renowned traditional origami design, see the work of origami master Akira Yoshizawa (Yoshizawa, 1987, 1997). Pioneers of origami in the Western Hemisphere include Robert Harbin, Samuel Randlett, Neal Elias, Ligia Montoya and Florence Temko (Harbin, 1970, 1978).

3.1.2 Box Pleating

The first significant development in origami design was described in the 1960s and was called *box pleating* (Kasahara, 1967). In this method, a piece of paper, not necessarily square, is initially pleated horizontally, vertically and diagonally. The design is then built upon these pre-creases. Engel explores box-pleating with some of his designs (Engel, 1994).

3.1.3 Blintzing and Grafting

Other authors have addressed the problem of origami design on a conceptual level (Kasahara and Takayama, 1987; Kasahara, 1988; Engel, 1994). The four fundamental bases can be improved by using repetitions of the entire base on a single sheet. Two such methods are *blintzing* and *grafting*.

Engel (Engel, 1994) describes blintzing, but does not know the inventor. To blintz a square of paper, the four corners are folded to the centre. The resulting figure is a smaller square with the addition of four triangular flaps shown in Figure 32(a). This new square, (which is actually the original square divided in two), is then folded like one of the regular bases. Because of the four extra flaps, the resulting figure has twice as many points as the regular base. The blintz was the first widespread innovation in origami since the invention of the frog base. An example of a blintzed bird base is shown in Figure 32(b). The crease pattern for the standard bird base can be seen in the shaded portion of Figure 32.

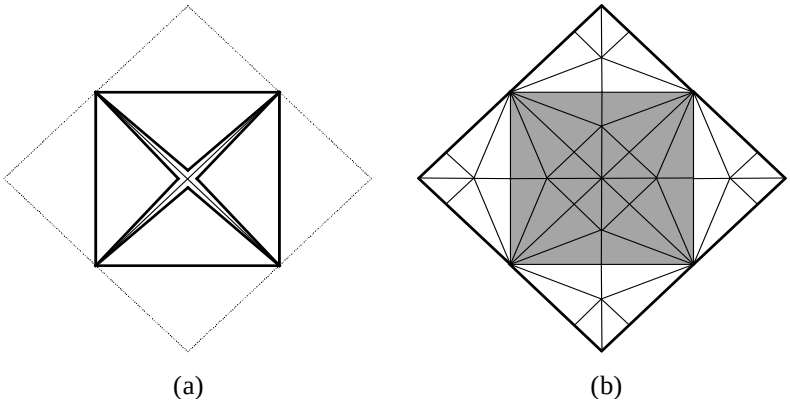


Figure 32. (a) The blintz fold. (b) Crease pattern for the blintzed bird base.

Grafting, a term coined by Engel (Engel, 1994), is done by inserting one base into another. For his first attempt, Engel grafted a frog base into the centre of a bird base to make a baby for his kangaroo model. Repeating the procedure allows the introduction of even more points. Blintzing and grafting led to further examination of the repeating patterns in the traditional bases.

3.1.4 Technical folding

In the 1970s, four origami designers independently found a set of techniques and symmetries suitable for folding complex models from single, uncut squares. These techniques form a research area of origami design called *technical folding*, or *origami sekkei*, combining both art and engineering. In developing these methods, Montroll (Montroll and Lang, 1990), Engel (Engel, 1988, 1994), Maekawa (Maekawa, 1992) and Lang (Lang, 1988a, 1989, 1994) examined the four traditional bases. They noticed the triangular shape that appears in multiples of two, four, eight and sixteen in the crease

patterns of the bases shown in Figure 33. By using multiple copies of varying size and orientation of the basic triangle shape, they noticed that much more complex bases could be formed. For example, Lang's "Sea Urchin" (Montroll and Lang, 1990) incorporates 128 copies of the triangle in its crease pattern, and results in 25 equal-length points in the 3D model.

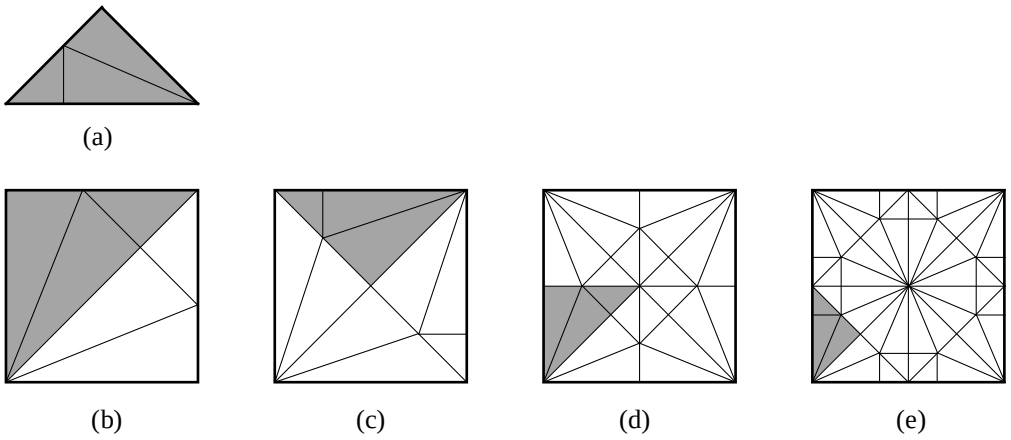


Figure 33. The triangle in (a) is found in successively smaller sizes in the crease patterns of the four fundamental bases: two in the kite base (b); four in the fish base (c); eight in the bird base (d); and sixteen in the frog base (e).

This triangle is still not the most fundamental unit of technical folding. The triangle found in the traditional bases can be separated into three smaller triangles (see Figure 34)— one isosceles right triangle with sides in the ratio of 1:1, and two identical scalene right triangles with sides in the ratio $1:1+\sqrt{2}$. These are the true building blocks of technical origami. Lang (Lang, 1988a, 1989) and Engel (Engel, 1994)

assembled these two triangles into larger and larger triangles and rectangles to create bases with more and more points.

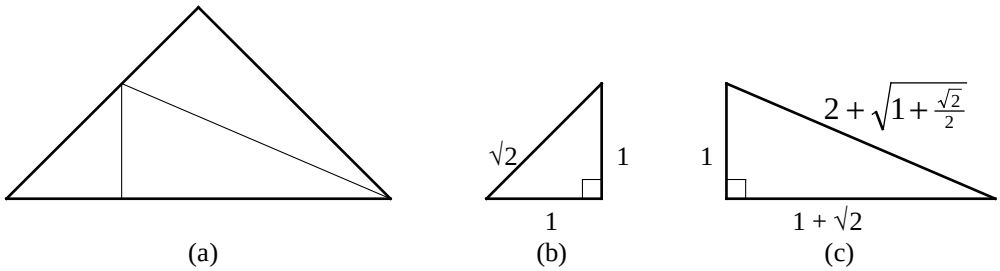


Figure 34. The true building blocks of technical origami, (b) and (c), can be found in the repeated triangle (a) of the traditional bases.

Note, however, that this method of copying the triangle shape cannot go on indefinitely. Real paper has a real thickness; although a generalisation of a complex model such as the sea urchin (Lang, 1988a) to 36 or 49 points is conceivable, it is not foldable. As more flaps are made, they each grow simultaneously thicker and shorter, and move closer together. The paper can no longer collapse compactly, and rips.

3.1.5 Modular Origami

Modular origami goes beyond the limitations of a single sheet of paper and uses several, sometimes hundreds, of separate pieces. Each small square is folded into a *unit* and identical units are linked together to create the object. The resulting model is usually a geometric solid, with the units making up the faces or edges of the polyhedron.

However, other designs such as architectural structures and boxes with lids can also be constructed using this method (Fuse, 1990, 1997, 1998).

One advantage of modular origami is that units are relatively easy to fold, but can result in an object with very complex structure. Fuse describes the basics of modular origami in *Unit Origami* (Fuse, 1990). Andrew Glassner has recently discussed modular origami in the context of 3D graphics and 3D visualisation (Glassner, 1996a, 1996b). He suggests that folding solids from flat paper is a useful experience for building visualisation skills and 3D intuition.

3.1.6 Computational Algorithms

The introduction of technical folding has made it possible to extend origami design to models with arbitrary numbers of points. Technical folders brought focus to the inherent geometry in paper folding, and have reduced much of the process of design to algorithms. These algorithms naturally lend themselves to computer implementation, thus perhaps making it possible for a computer to design origami.

3.1.7 Origami Design by a Computer?

People relate origami to geometry, because of the angles and straight lines that can be seen in the creation of models and their crease patterns. This underlying geometry has recently been exploited by origami designers to produce a computational approach to origami design.

The first attempt to program a computer to aid the design of origami was in 1971 (Engel, 1994; Lang, 1988a). Arthur Appel, a researcher at IBM, programmed a computer to print out pseudo-random, simple geometric configurations of crease lines on a square piece of paper. About 90% of these crease patterns were considered unsuccessful, or non-foldable. However, the other 10% suggest that it may be possible to program a computer to design origami models more complex than any designed by a human.

There are some practical problems in origami design by computer, however. A computer could print out a geometric crease pattern of a foldable model, but could anyone figure out how to fold it? Origami models generally require a long sequence of folding diagrams or instructions to describe the final model. The thickness of physical paper, though small, makes a difference between real and computer folding. In some models, the thickness of the paper is actually used to help make the model appear three-dimensional. But paper, defined as a finite 2D surface, has no thickness in the computer.

Fisher (Fisher, 1996) describes an algorithm to fold the paper given a crease pattern with the location and kind of each fold (mountain or valley) and some layering relationships between faces. However, the algorithm makes the assumption that the crease pattern will fold flat, and is only correct for the case where the paper has no thickness. See §3.3.1 for further discussion of the complexity of such an algorithm.

3.2 Origami Geometry

Martin Gardner was one of the first to introduce Westerners to origami through his article in Scientific American (Gardner, 1959). Engineers, architects, scientists and mathematicians are intrigued by the mystery and complexity that lies within the simple square. They have studied the characteristics of existing origami models and folding operations. This includes a characterisation of the angles, faces, edges and vertices of origami crease patterns. There are interesting purely mathematical properties of a finite, convex, foldable (along a straight line), non-self-intersecting plane (Row, 1966; Maekawa and Kasahara, 1983). Some of these properties have been explained and turned into theorems, but there are still many unanswered questions.

3.2.1 Geometry in Paper Folding

Paper folding is a novel and entertaining way to introduce children and adults to geometry. In 1905, Row (Row, 1966) published a small book of geometric exercises to be done using only a penknife and paper squares. He showed that some important geometric processes can be effected better with paper folding than with a compass and straightedge, the usual tools. For example, the construction of bisections of lines and angles, perpendiculars and parallels can be built easily using origami.

Row provides methods for folding all the regular polygons, including the nonagon, on a single square of paper. Arithmetic, geometric and harmonic progressions and summation of certain arithmetic series are developed, as is the calculation of π. A

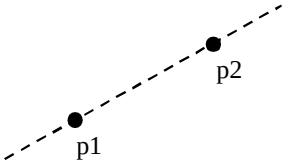
folded sheet of paper illustrates concepts such as congruence, symmetry, similarity, concurrence of straight lines and collinearity of points. Conic sections are constructed, a duplication of the cube, and a trisection of the angle is achieved.

Several other authors have examined the geometry of paper folding. Auckly and Cleveland give a system of origami constructions in terms of mathematics (Auckley and Cleveland, 1995). Geretschlager discusses Euclidean constructions and the geometry of origami (Geretschlager, 1995). Fisher shows that if the initial piece of paper is convex then every face in a flat origami model folded from that piece of paper will be convex (Fisher, 1996). Huzita describes an axiomatic system of origami geometry (Huzita, 1992) that is summarised in the following section.

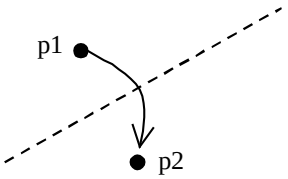
3.2.2 Huzita's Origami Axioms

Huzita has developed the most powerful set of origami axioms to date (Hull, 1998b). Some geometrical constructions that cannot be performed with the standard straightedge and compass are shown to be possible using paper folding. The first five axioms can be done using a straightedge and compass, but the final axiom shows that origami also has the power to solve cubic equations. Thus we can trisect the angle and double the cube using origami.

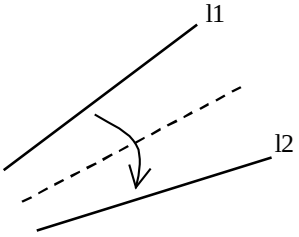
Axiom 1: Given two points *p1* and *p2* we can fold a line connecting them.



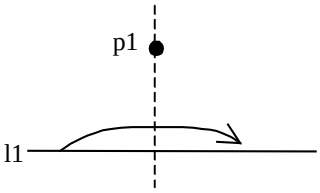
Axiom 2: Given two points *p1* and *p2* we can fold *p1* onto *p2*.



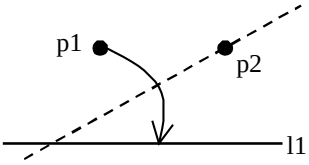
Axiom 3: Given two lines *l1* and *l2*, we can fold line *l1* onto line *l2*.



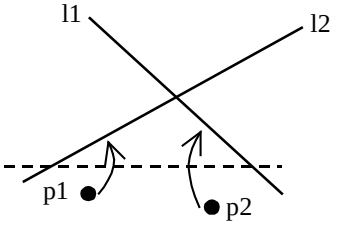
Axiom 4: Given a point *p1* and a line *l1* we can make a fold perpendicular to *l1* passing through the point *p1*.



Axiom 5: Given two points *p1* and *p2* and a line *l1* we can make a fold that places *p1* onto *l1* and passes through the point *p2*.



Axiom 6: Given two points *p1* and *p2* and two lines *l1* and *l2* we can make a fold that places *p1* onto line *l1* and places *p2* onto line *l2*.



3.2.3 Mathematics of Flat Origamis

A *flat foldable* crease pattern is any crease pattern that can be folded into a base so that all layers of the base lie in a common plane (Lang, 1996). Some crease patterns, such as the one in Figure 35(b), cannot be folded flat.

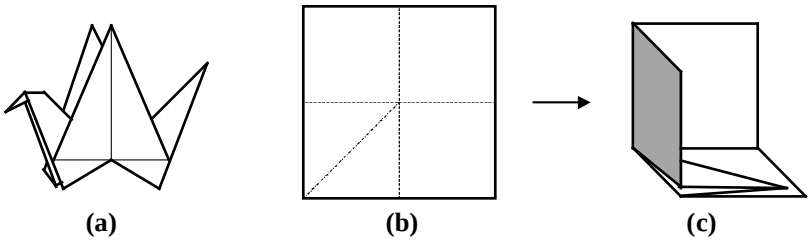


Figure 35. The traditional flapping bird (a) lies flat when complete. The crease pattern (b), for the figure (c), which does not lie flat, is not flat foldable.

Hull presents a mathematical study of origami geometry, particularly of origamis that are flat foldable (Hull, 1994). He explores the possibilities of a graph theoretic model of origami. Hull presents the following theorems, including proofs, of local properties of a flat origami *C*. These theorems describe the properties of the crease pattern around a single interior vertex *v* in *C*, where *C* is the crease pattern of a flat origami with any number of vertices. These theorems are also discussed in (Kasahara and Takahama, 1987; Kawasaki, 1989).

> **Theorem 2.1** (Maekawa): The number of mountain folds *M* minus the number of valley folds *V* meeting at vertex *v* in *C* is either 2 or -2.

> **Corollary 2.1** (Hull): The number of creases meeting at *v* is even.

Corollary 2.2 (Hull): If we think of an origami crease pattern as a graph, then every flat origami crease pattern is 2 face-colourable.

Theorem 2.2 (Kawasaki): The sum of the alternate angles about vertex v in C is π.

Theorem 2.3 (Kawasaki): If the sum of the alternate angles about vertex v in C is π, then C generates a flat origami.

Unfortunately, there are problems in extending these theorems globally (i.e. to crease patterns with more than one vertex). For example, for the traditional flapping bird shown in Figure 35(a), $M - V = 15$. Hull also shows that there exist some non-foldable crease patterns that are locally foldable. He ends the study with an attempt at a conjecture:

Conjecture 2.1 (Incomplete) (Hull): A collection C of crease lines in the square (possibly with more than one vertex) can generate a flat origami if and only if:

(i) each vertex in C satisfies the 180° condition and
(ii) the origami line graph of C is 2 vertex-colourable

where,

An *origami line graph* G is constructed from C as follows:

(1) The vertices of G are the crease lines in C.

(2) Two crease lines l_i and l_j form an edge in G if and only if

(i) l_i and l_j are adjacent in C, and

(ii) l_i and l_j are not both mountains nor both valleys.

However, Hull also shows a crease pattern that satisfies the conditions of the conjecture, but is non-foldable. This disproves the current conjecture, and Hull gives an example suggesting that it is quite difficult to add a condition that would make the conjecture true.

3.2.4 Circle Tiling and Packing

One geometric heuristic applied to origami design uses the centres of non-overlapping circles to determine the tips of flaps of the base. Several authors, including Maekawa, Kawahata, Kawasaki and Lang use a stick figure of the target shape, and represent the flaps of the base by non-overlapping circles and/or circular contours (Maekawa, 1992; Kawahata, 1994; Kawasaki, 1989; Lang, 1994). An example is shown in Figure 36.

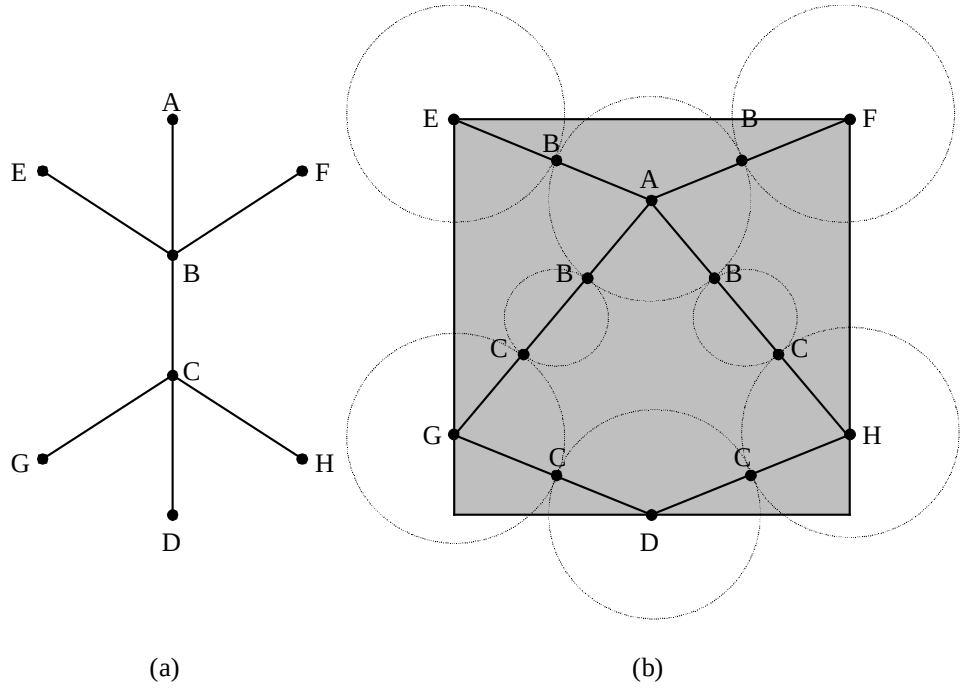


Figure 36. (a) Stick figure for a six-legged base and (b) the preliminary crease pattern for the base, with tiled circles.

Their conjecture is:

> **Conjecture 2.2**: If a pattern of non-overlapping circles, centres (x_i, y_i), radii r_i, are laid out on a sheet of paper, then there is always a way of folding the paper into a flat origami with flaps of length r_i, whose tips correspond to the points (x_i, y_i).

Ewins (Ewins, 1996) attempts to prove the conjecture for a design packed with circles, and in doing so, provides better justification for a belief in the proposition. He designs a set of triangular tiles which are generated by joining the centres of packed circles, and describes special placement of these tiles which will guarantee that Maekawa's theorem

holds at every vertex simultaneously. However, it has been shown that an origami crease pattern can be locally foldable flat and satisfy Maekawa's theorem everywhere, and still not be foldable, so the proof is incomplete. Gardner also discusses tangent circles (Gardner, 1992).

3.2.5 Planar Graph Theory

Modular origami uses many sheets of paper together to create various geometric forms. Since more than one piece of paper is involved, one can also experiment with colour patterns. Hull explores the relationship of modular origami to planar graph theory, and shows how the study of each provides insight on the other (Hull, 1996).

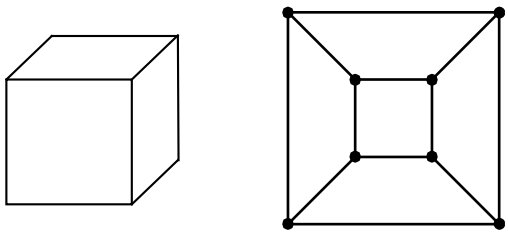


Figure 37. The cube (a) and its planar graph representation (b).

Many modular folds produce stellated or capped polyhedra (Kasahara, 1988; Fuse, 1990; Gurkewitz and Arnstein, 1995). Any polyhedra can be embedded in the plane, as in Figure 37, resulting in a *planar graph* of the solid. For capped polyhedra, every face of the planar graph will be a triangle. Hence every vertex of the dual of this planar graph has degree three.

Theorem 2.4 (Brook): A graph is vertex-colourable in three colours if its maximum vertex degree is 3, unless it is the tetrahedron.

Then Brook's Theorem tells us that the dual is three vertex-colourable. This proves that the original planar graph, and thus the capped polyhedra, can be face-coloured using three colours.

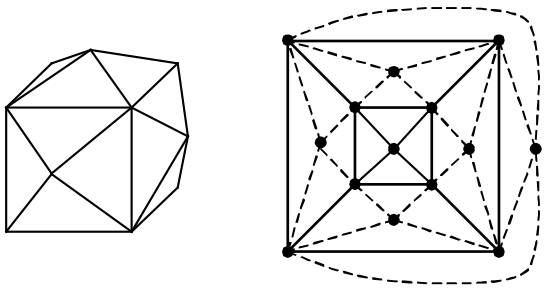


Figure 38. A capped cube (a) and its corresponding planar graph (b).

Finally, Hull suggests that by examining the colourings of 3D origami capped polyhedra, some new insight to the famous Four Colour Theorem might be found.

3.3 Computational Geometry of Origami

Computational algorithms for origami design are rare in the literature. The Second International Meeting on Origami Science and Scientific Origami is a good source for origami mathematics and geometry. Consideration of computational algorithms has only emerged recently, notably by Bern and Hayes and Lang (Bern and Hayes, 1996; Lang, 1996).

3.3.1 The Complexity of Flat Origami

Bern and Hayes investigate the problem of determining if a given crease pattern is flat foldable, and answer three versions of the question (Bern and Hayes, 1996). First, they determine that local flat foldability of a crease pattern is easy, and give a linear-time algorithm that assigns mountain and valley orientations to the creases and determines an overlap order (Figure 39) for flaps. Next they show that it is NP-complete to assign orientations and overlap order globally (i.e., for a crease pattern with more than one vertex). Finally they show that given the location and orientations of all creases in the pattern, simply finding an overlap order is NP-complete.

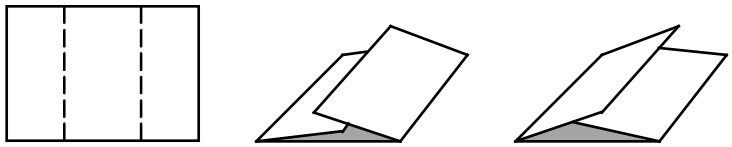


Figure 39. Two flat origamis with the same crease pattern (a), but different overlap orders (b) and (c)

In practice, a paper folder can find the assignments of creases on a crease pattern by empirical means. However, only the most experienced origamist is capable of folding complicated models from the crease pattern only. Origami instructions are usually presented as a sequence of folding steps and diagrams. Some authors give only the crease pattern (including mountain and valley assignments) and a photograph or drawing of the finished model (Engel, 1994).

Bern and Hayes conclude their investigation with some suggested open problems including:

> **Open Problem 2.1**: How many different flat origamis are there for a given crease pattern?
>
> **Open Problem 2.2**: Is every simple polygon the silhouette of a flat origami? If so, how many creases are needed for an n-vertex polygon, and how thick (how many layers) will the origami be?
>
> **Open Problem 2.3**: Determine, for a given crease pattern, if there exists a flat origami which is at most k layers thick. Bern and Hayes' reduction shows that this problem is NP-complete for $k \geq 7$, but what about k from 2 to 6? This is called k-layer flat foldability.

3.3.2 A Computational Algorithm for Origami Design

A useful computational algorithm for origami design is by Lang and is implemented in C++ in a program called *Treemaker* (Lang, 1996). The program designs a crease pattern that folds flat into a base with any desired number of flaps of arbitrary length. Most of the crease assignments are also given. The folder must determine the rest of the crease assignments, and how to transform the crease pattern into a base. Lang emphasises that there is nothing inherent in the algorithm that would give a sequence of folding steps. In fact, many of the resulting bases cannot be folded one crease at a time; instead, all pre-creases must be made and then brought together at once. Also,

the result of the algorithm is a base, and not the final figure. It is up to the folder to add the details of the model.

The algorithm is based on a set of mathematical definitions of an origami base and an abstraction of that definition to the tree graph of a base. Lang defines several other useful measures of the properties of this base and its tree graph, including projectability, completeness, connectedness and orientability.

3.4 Computer Simulation of Paper Folding

3.4.1 Previous work

Modelling a folded piece of paper may seem like a trivial task, but there are several obstacles to creating a realistic, accurate model. These include modelling the thickness and texture of the paper, modelling non-planar surfaces that arise during folding, and maintaining the order of layers in a flat folded model. Only three papers and two pieces of commercial software are directly related to computer modelling and animation of origami. These are described below.

3.4.1.1 Origami Playing Simulator

Miyazaki et al. have designed and implemented an origami-playing simulator in virtual space (Miyazaki and others, 1996). It includes folding up, bending and curving of the paper, and uses a physically-based model. As the user manipulates the virtual paper it is continuously deformed and updated on the screen. However, their implementation

is limited to certain basic types of origami folds since simultaneous manipulation of more than one point is not possible in their model. Their implementation is a fundamental study of software development in virtual reality.

3.4.1.2 Origami Animation from Keyframes

Agui et al. have developed a method of producing an animation of an origami model based on 3D databases (Agui and others, 1983). The method creates pictures between key frames of the folding process according to the origami rules. However, key frames must be provided by hand, and are very difficult to describe.

3.4.1.3 Origami Language

Fisher has developed a language syntax for describing the folding sequences of flat origami models (Fisher, 1996). He presents algorithms and data structures for performing origami operations, and discusses many issues involved in using computers to do origami. He describes a partial implementation of a computer program for entering and executing instructions given in the language. However, several portions of the language are not implemented due to the difficulty of the problems.

3.4.1.4 Paper Animal Workshop

Kittyhawk Software Inc. has developed an origami program, "Paper Animal Workshop" available on CD-ROM (Kittyhawk, 1995). The program presents several built-in models with a useful, controllable animation interface. The interface includes a

display of the crease pattern, the 3D model, the set of folding instructions, animation and camera control, and the ability to print the crease pattern to follow along with the instructions. Curved surfaces are not represented. It is an excellent tool for teaching origami models using animation. Unfortunately, the user is only allowed to fold the given models, and no interaction with the virtual origami model is provided. This means that no new folds or creases can be added to any of the models, and only the programmer can create new models using their system. Attempts have been made to find details about the method used for modelling paper in their system, but Kittyhawk does not seem to have made this information available.

3.4.1.5 HyperGami

HyperGami is a computer program that allows the user to modify three-dimensional polyhedra and then automatically generates the corresponding *folding net*. The folding net is a two-dimensional version of the given polyhedron, i.e. the shape you get if you cut along some of the edges and open up the polyhedron to flatten it on a piece of paper as in Figure 40.

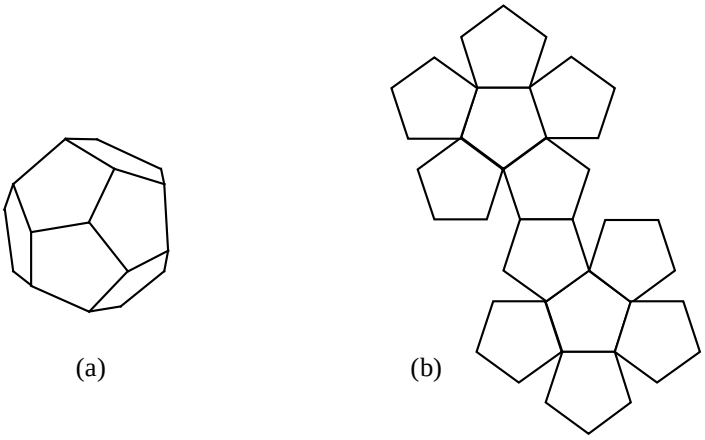


Figure 40. A polyhedron (a) and its corresponding folding net (b).

HyperGami allows the user to modify the given polyhedra by various transformations including stretching along an axis, truncating vertices or capping faces. The user can also decorate the resulting shape and print the folding net on a colour printer. The net is then cut out and assembled. Several examples of HyperGami sculptures can be see on their web pages at http://www.cs.colorado.edu/~eisenbea/hypergami/ (the research site) or http://www.hypergami.com/ (a commercial site). The HyperGami system is available free of charge for download from the research site given above.

The HyperGami program does not, however, allow the user to fold virtual paper. It is purely a CAD design system for paper sculpture. Manipulations can only be done on the provided polyhedra, so traditional origami cannot be realised with the HyperGami system.

3.4.2 Related Areas

Since relatively little work has been done on modelling a folded piece of paper specifically, work from several other areas of computer graphics and modelling may be looked at for reference. Some of these areas are described below.

3.4.2.1 Polygon mesh

Polygonal modelling is a common topic in computer graphics. Most current research focuses on efficient ways to increase the number of polygons computationally allowable in a model, or modelling smooth surfaces.

Turk describes a method of creating surface models at several levels of detail from an original polygonal description of a given object (Turk, 1992). He does this by re-tiling the polygonal surface automatically in a way that is faithful to both the geometry and the topology of the original surface. By examining the methods Turk uses to achieve the re-tiling, we can learn about direct manipulation of polygon meshes.

Further work has been done on direct manipulation of meshes, but not in the way that would be used for modelling paper. In most cases, the polygon mesh is used to define a malleable, stretchable surface. The surface of a piece of paper should appear to have no stretch at all. Allan et al. present a method for direct manipulation of polygon meshes, where a single vertex may be moved without altering the topology of the mesh

(Allan and others, 1989). This powerful technique is useful for smoothly sculpting surfaces, but not as effective for folding them.

Another method for sculpting polygonal surfaces is discussed by Bill and Lodha in their paper (Bill and Lodha, 1995). They describe a system for designing free-form polygonal models using virtual sculpting tools and mesh refinement operations. The user controls a virtual tool (a geometric object) to push, pull and deform the initial mesh in a variety of ways. The user may also refine the mesh to obtain more detail. This method is very effective for smooth, solid models, but does not apply directly to modelling a sheet of folded paper.

Unlike the polygon mesh implementations described here, a piece of real paper does not (perceivably) stretch or change in size or shape. Ideally, we can imagine the paper as a simple planar polygon, say, a square. As the piece of paper is folded, it is divided into separate regions (called faces) by the creases (called edges). Many of the faces in traditional paper folding are triangular, and most are convex.

A special polygon mesh seems like a reasonable way to model paper. A simple dynamic polygon mesh would be able to represent the crease pattern and position of the paper in 3-space. However, this simple mesh will not represent the thickness of the paper, nor allow for true curves in the paper. Curves may be approximated with smaller polygons in the curved area. The method providing for user manipulation of

the paper must not allow stretching of the paper surface (i.e. arbitrary positioning of vertices).

The data structure describing the polygon mesh would have to include not only the details about vertices, edges and faces, but also layering information (e.g. for arbitrarily close parallel flaps of the paper). A single, straight crease may involve many layers of paper as the model becomes more complicated. A method must be devised to select and manipulate only the desired layers.

The user must be allowed to refine the mesh, in the same way that creases are added throughout the folding of a paper model. The final configuration of the mesh would look like the crease pattern on the real paper if it were laid out flat.

3.4.2.2 Simple polygonal model

A simple polygonal model, as opposed to a polygon mesh, is just a set of simple polygons in space. An advantage is that several distinct polygons may be more easily manipulated than a polygon mesh. Rather than moving vertices and edges about, we would perform 3D transformations on entire polygons, where each polygon would represent one face of the crease pattern as described above.

However, the piece of paper is supposed to be continuous. So dividing the surface of the paper into several distinct pieces is not satisfactory. Two adjacent faces may appear to have a tiny space between them as numerical inaccuracies occur during the

manipulation. This would destroy the illusion of a single folded piece of paper. This method would also be inefficient in terms of space required to store the set of polygons.

3.4.2.3 Cloth modelling

A considerable amount of work has been done recently on modelling the draping of cloth. However, cloth is generally more flexible than paper, and shows creases in a much more relaxed way. Ng and Grimsdale present a survey of nineteen cloth modelling techniques in their paper (Ng and Grimsdale, 1996). See also results in (Breen, 1994), (Eischen, 1996), (Eberhardt, 1996), and (Baraff and Witkin, 1998). Many of the models presented are physically-based, but physical techniques involve solving a large set of differential equations or iterating many times, and so are not appropriate for interactive manipulation and animation.

The authors make it clear that in computer graphics or animation, appearance is generally more important than physical accuracy. In the case of paper folding, most of the surfaces are not curved and so can be modelled with polygons. Curved surfaces may be physically inaccurate, but approximated with smaller polygons well enough to be visually acceptable.

3.4.2.4 Drawing With Constraints

Drawing with constraints is a method of augmenting a basic 2D drawing program with constraints placed interactively by the user (Gleicher, 1994). The constraints are then maintained as the user drags the model.

This concept can be extended into 3D. In the case of modelling folded paper, we could represent the creases by lines of a specified length and width, joined by their ends. The constraints would determine acceptable angles that may occur between two adjacent lines. As the user adds more creases, the constraints between lines would change. The result would be a set of lines constrained as if lying on a piece of paper, which the user could drag around. The surface of the paper could be added later during rendering. One difficulty is how to decide and specify which parts of the paper remain stationary while the rest of the model is moved around.

3.4.2.5 Interpretation of Line Drawings

In the research areas of artificial intelligence and computer vision, methods to understand line drawings are studied. The problem is to reconstruct the 3D structure of a scene from a single image containing lines with no texture, colour or shading. One class of these line drawings is known as the *Origami world* (Kanade, 1980), that is scenes that are built up from planar panels of negligible thickness. Parodi presents a study of the complexity of interpreting Origami scenes in his paper (Parodi, 1996).

3.5 Current Research

Several origami enthusiasts have attempted to develop programs that will allow simple representation and animation of virtual origami. The results described in §3.4.1 are the first published attempts at implementations, with limited results.

The traditional crane is shown in Figure 41 and is a simple 3D origami model in which curved surfaces appear in the final steps. No one has yet created a model of a folded sheet of paper that handles the nuances of curves, folds, thickness and three-dimensionality of origami. Various graph theory and computational geometry approaches have been taken, but these ignore fundamental intricacies of the behaviour of paper. A physically-based model of origami has the most promise of working correctly, but is much more difficult to implement than might be first assumed.

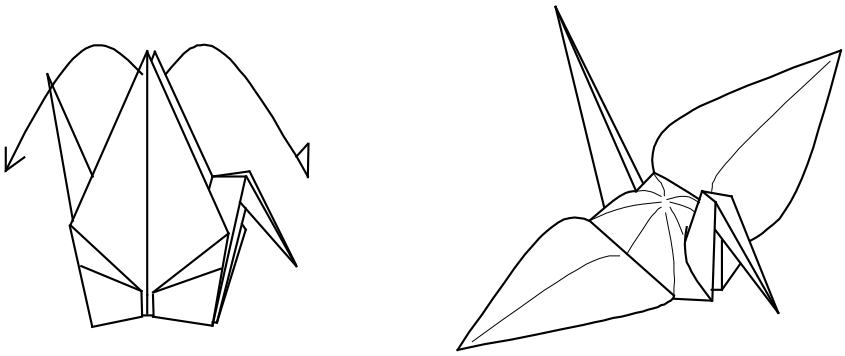


Figure 41. In the final step of the traditional origami crane, the wings are curved and the body is stretched so that the model becomes 3-dimensional.

Once an effective model has been found, perhaps the problem of finding algorithms for folding it will become easier. Current systems have difficulty manipulating more than one point or vertex on the model at a time.

Chapter 4

PROGRAM DESIGN

4.1 Overview

The program that has been developed and is described in this section will allow the user to manipulate a computer-generated piece of paper in what appears to be three dimensions, similarly to folding a real piece of paper. The paper may be viewed from all directions, and at any reasonable zoom factor.

The interface will allow the user to indicate creases on the paper, and then perform the folds. Folds may be constructed using natural landmarks on the paper that are the result of previous operations, or by using some built-in mathematical tools such as midpoint of a line. The program will animate the behaviour of the paper between steps. The resulting set of steps may be saved for later use.

At this time, a partial implementation of this design has been completed. Details of the implementation are presented in Chapter 6, Results and Evaluation.

4.2 Functional Requirements

4.2.1 Intended Users

Several members of the rec.arts.origami newsgroup have expressed strong interest in an easy-to-use program for teaching and diagramming origami models. A few attempts have been made to model a folded piece of paper using computer graphics, but never in such a way that users familiar with traditional diagramming methods would find very useful.

Ideally, this program would be used as a tool to teach origami models in a clearer and more intuitive way than with diagrams alone. Some people, especially children, find traditional diagrams hard to understand. Animated diagrams would be easier to follow.

This program would allow origami designers to work out new models on the computer before committing them to paper. Besides saving reams of paper, the computer program would provide the ability to go back a few steps or review an entire model from start to finish. Once a paper model is completed, it is usually quite difficult to discover the sequence of operations that led to the final product. By using this tool, designers would not have to remember every step to re-create the model.

This program would also be useful for teaching mathematics and geometry since there are many geometrical principles that are demonstrated by a folded piece of paper.

4.2.2 Basic Functions

This section describes the functions that are essential for the program to be considered "working".

- user-defined valley & mountain folds, user may pick vertices and edges to build folds
- use traditional diagramming symbols
- use square paper
- flip the model over
- "all at once" folding (i.e. no animation, just show "before" and "after" a pre-defined folding operation)
- undo to previous step (just one-step undo)
- 3D view of current state of the model, including zoom, rotate, pan (a virtual trackball might be best, but buttons could be easier to implement)
- save a "snapshot" of the model (e.g. when it is completed)

4.2.3 Desirable Functions

This section describes the functions that are necessary in the implementation to make it interesting and useful.

- all complex/combination folds (see Table 1)
- user may use various tools to choose landmarks for creating folds
- animation of the fold as the user drags a flap of paper around
- flat-surface animation between steps
- replay the entire model backwards and forwards
- infinite undo while constructing a model
- start with any shape of paper
- set the colours for vertices, edges and faces, to watch where they go between steps
- allow the user to remove creases and vertices that will no longer be used
- show the "type" of creases currently being used, but just show a faint line for creases not involved in the current fold
- print out the final crease pattern on paper so that it can be folded

4.2.4 Additional Functions

This section describes the "bells and whistles" that may be added to the implementation if time permits. They are not really essential to the functionality of the program.

- toggles:
  - "diagram mode" (see symbols on paper) vs. "natural mode" (see indication of crease location using some sort of shading)
  - "fold mode" (to make models) vs. "view mode" (to watch steps of saved models)
  - "beginner mode" (show every pre-crease fold in a complex fold, perhaps show location of fingers in animation) vs. "experienced mode" (name the fold and the flap and just see the results)
- customisation of available tools (i.e. a teacher may turn off certain mathematical constructs so that a student is required to use other methods to achieve the same thing. E.g. find the midpoint of a line without using the "midpoint" tool.
- put a texture on the paper
- start with a base (preliminary fold, bird base, etc.) instead of flat paper
- curved surfaces and curved surface animation
- allow to cut the paper
- pre-defined animations of some difficult sections
- some sample saved models

4.2.5 Importance of the Basic Symbols and Procedures

Some of the basic symbols and traditional procedures are more common and useful than others. This chart outlines the applicability of each symbol and procedure described previously. These ratings are informal, and are based on personal experience, memories of first learning origami, and comments made by members of the rec.arts.origami newsgroup. The overall rating in the last column is based on arbitrary weighting of the other four columns.

Ratings (in order):

> How common or how easy? Very – quite – somewhat – not (A "usually" qualifier means that in most cases the given rating applies, but there may be a few exceptions.)
>
> Can it be done without curved surfaces? Yes – usually – not usually – no
>
> Overall rating: Essential – desirable – useful – not very useful

Table 1. Importance of the basic folds and symbols.

Basic fold or symbol	How common is the fold or symbol in general?	How easy is it to perform the fold on real paper?	How easy is it to learn or understand the fold from traditional diagrams?	Can the fold be done without curved surfaces?	Overall importance of the procedure to this program?
Valley fold	very	very	very	yes	essential
Mountain fold	very	very	very	yes	essential
Fold and unfold	very	very	very	yes	essential
Push here	quite	quite	quite	not usually	desirable
Edge configurations	somewhat	-	quite	-	useful
Watch this spot	somewhat	-	very	-	desirable
Rotate	somewhat	very	very	yes	desirable
Equal distances	somewhat	quite (usually)	very	-	useful
Equal angles	somewhat	quite (usually)	very	-	useful
Fold over and over	not	very	very	yes	useful
Turn the paper over	quite	very	very	yes	essential
Pull paper out from here	somewhat	somewhat	somewhat	not usually	useful
Cut-away view	quite	-	quite	-	desirable
X-ray line	quite	-	quite	-	desirable
Next view larger	somewhat	-	quite	-	useful
Hold here and pull	somewhat	quite (usually)	somewhat	no	useful

A dash (-) indicates not applicable.

Table 2. Importance of the basic procedures

Basic procedure	How common is the procedure in general?	How easy is it to perform the fold on real paper?	How easy is it to learn or understand the fold from traditional diagrams?	Can the fold be done without curved surfaces?	Overall importance of the procedure to this program?
Preliminary fold	very	very	quite	yes	desirable
Rabbit-ear	quite	quite	somewhat	yes	desirable
Inside reverse	very	quite	somewhat	not usually	desirable
Outside reverse	quite	somewhat	somewhat	not usually	desirable
Crimp	quite	somewhat	somewhat	not usually	desirable
Squash	quite	somewhat	somewhat	usually	desirable
Petal fold a point	quite	somewhat	somewhat	usually	desirable
Petal fold an edge	somewhat	somewhat	somewhat	usually	desirable
Open sink	somewhat	not	not	no	useful
Closed sink	not	not	not	no	not very useful

4.3 Graphical User Interface

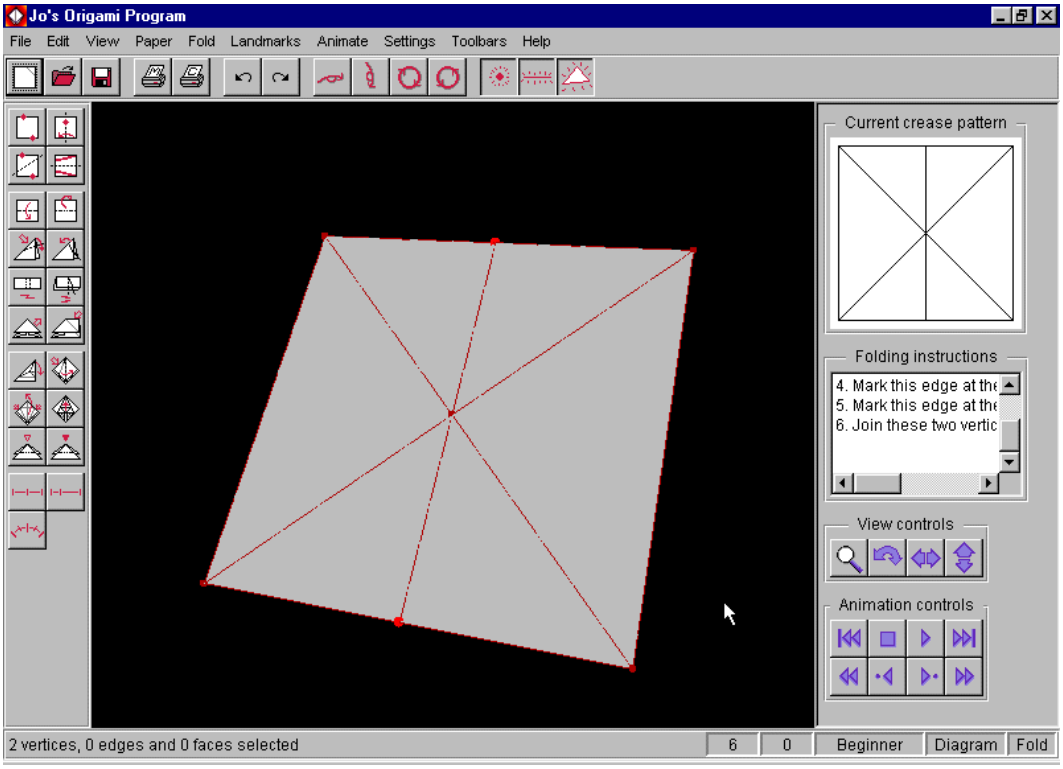


Figure 42. The graphical user interface

The screen capture shown here in Figure 42, and those presented throughout this section, are from the current implementation written in Java and running under Windows 95. The current implementation does not have the complete functionality described in this section.

4.3.1 Overview of the Interface Design

Interaction is primarily with the mouse, with some input using the keyboard. The interface allows the user to interact with the paper model similarly to how the paper is

folded in reality. Unlike previously implemented origami interfaces (Fisher, 1996; Miyazaki and others, 1996), this one allows the user to describe the location of the intended crease on the folded piece of paper, and then bend the paper along the crease until it is in its desired place. Other implementations only allow the user to click and drag a corner of the paper around until it is in place. However, pre-creasing is essential when constructing real origami models and so should be included in a user interface. Once a crease is made, the flap of paper may be dragged about the crease line until it is in the correct position. Alternatively, the paper may be folded flat automatically, as this is the most common position for the flap.

The user is able to move the viewpoint around in space while folding the paper, in order to see the model from any angle. This includes zooming in and out, rotating or flipping the paper over. The user is able to save the final model for viewing later.

The user may add curved surfaces by highlighting an existing region of the model to subdivide it into smaller polygons. The user may then drag the curved region into place.

4.3.2 General Layout

The graphical user interface has the following standard areas:

- Menu bar at the top

- Status bar at the bottom
- Window in the centre where the 3D model is drawn
- Other toolbars and displays on the left and right of the central display
- The central window is the main working window. If the user resizes the application, the central window and the 3D model are resized to take advantage of all available space.

4.3.3 Central Window

The central window displays the 3D model of the virtual sheet of paper and allows the user to manipulate it directly using the mouse.

4.3.3.1 Selection of Objects

We assume a one-button mouse so as to allow the most portability to various platforms. The mouse button is used to select and deselect menu items, buttons and landmarks on the virtual paper. If a face, edge or vertex on the paper is selected, it is coloured brighter than unselected items. In the case of edges and vertices, a selected object is also drawn larger than the others.

Any function that can be performed with a multiple-button mouse can be achieved by using a modifier key with a one-button mouse. Java allows the second (and third)

mouse button, if available, to be interpreted as a one-button mouse click with a keyboard key modifier.

If the META[3] key is held down and the mouse button is clicked, then the selected object is added to the currently selected items. When the user clicks on a face, point or edge that is already selected, it is deselected. In systems with a two- or three-button mouse, the right mouse button is used in place of the META key.

If the CONTROL key is held down and the mouse button is clicked, a context-sensitive pop-up menu is presented that will allow changes to the selected item. For example, when a face, edge or vertex is selected, the CONTROL key may be used to select a pop-up menu item to change its colour. If a face is selected, the CONTROL key may also be used to make the face transparent to allow the view and selection of hidden objects.

A face of the model is a flat section of paper bordered by edges, and containing no other edges. Selection of a face on the model is straightforward; the user simply clicks on the face with the mouse button. The user may also select several adjacent faces to be treated as a single face by holding down the META key (or using the right mouse button) and clicking with the mouse pointer on each face. The face that is closest to

[3] The META key is the Apple key on MacIntosh computers. Windows and UNIX-based systems do not have a META key, and use a second button on the mouse instead. Although the general convention is to use the SHIFT key to select multiple items, META is used here in order to take advantage of the second mouse button on other systems using the Java language.

the viewer and lies under the mouse pointer will be selected. The user may also select contiguous faces that lie in a single plane of the model by double-clicking anywhere on one of the faces with the mouse button.

Selection of vertices and edges is more involved, since several edges may share an endpoint. When the user clicks on a vertex with the mouse button, only the vertex is selected (and not any of the adjacent edges). When the user clicks on an edge with the mouse button, just the single edge (between the two closest endpoints) is selected. If the user double-clicks an edge, then all coincident edges are selected. Again, the META key (or second mouse button) may be used with the mouse to select multiple edges and/or vertices.

If the user clicks on a point that is shared by more than one object, only the object that is closest to the viewer will be selected. If the point is common to more than one type of object (e.g. an edge and a vertex), only one object will be selected in the priority:

(1) vertex

(2) edge

(3) face

If the user wants to select an object that is not nearest to the viewpoint, he or she can rotate the model around until the desired object is closest. Alternatively, if the user clicks at the same position repeatedly, selection will cycle through all objects that are located at that position.

4.3.4 Toolbars

Toolbars are sets of buttons with images on them. When the user clicks on a button on the toolbar, the specified default function is performed. The items on the toolbars are also available on the Menu Bar where parameters may be supplied (see §4.3.5). The toolbars surrounding the central window have buttons for:

- main functions
- simple and complex folds
- tools for choosing landmarks on the paper
- camera control
- animation control

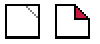
The buttons for simple folds, complex folds and landmarks use the traditional diagramming symbols. Every button consists of a two-frame animation that is activated when the user passes the mouse over the button. This is called a *rollover icon*. Rollover icons give a quick visual description of what will occur if the user chooses to use the button.

There is also a text box, which allows textual descriptions to be associated with folding steps. A window containing the current crease pattern is also visible. The user may select to hide or show any of the toolbars and displays (except the main central display) as desired.

4.3.4.1 Main Toolbar

Figure 43. The main toolbar

The main toolbar has several buttons for main, frequently used functions. They are presented here with their rollover icons.



Start a new model with a square sheet of paper, white side up.

Open an existing model.

Save the current model.

Print out the main display.

Print out the current crease pattern.

Undo the last step.

Redo the last undone step.

Flip the paper over horizontally.

Flip the paper over vertically.

Rotate the model clockwise (default 90°).

Rotate the model counter-clockwise (default 90°).

Toggle the visibility of vertices on/off.

Toggle the visibility of edges on/off.

Toggle the visibility of faces on/off.

4.3.4.2 Folding Tools

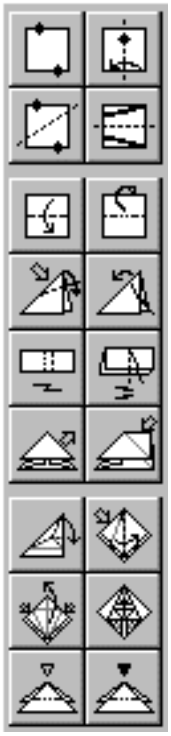
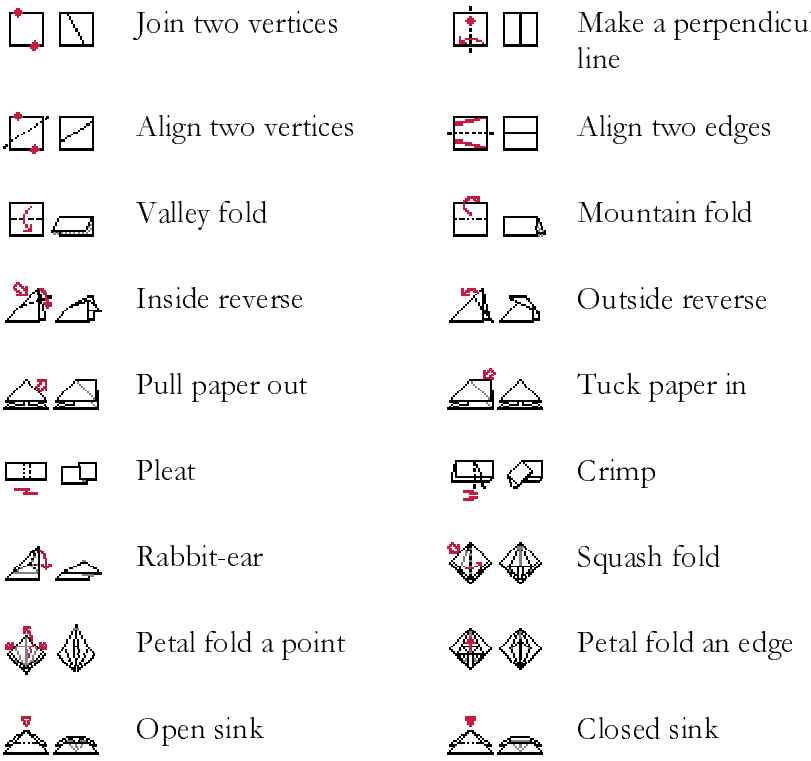


Figure 44. The folding toolbars

The folding toolbar has buttons for creating new creases and folding along existing creases. Each of the simple and complex fold types described previously has its own button (e.g. valley fold, mountain fold, rabbit-ear fold, etc.). The user selects the vertices, edges or faces required for a certain type of fold, and then the user clicks on the appropriate button to select the type of fold that is desired. Each type of fold will expect different landmarks to be provided. These are indicated using the mouse to directly select the landmarks from the model displayed on the screen.

If the user does not select the correct number and type of items required for a certain type of crease or fold, an audible alert will be sounded and the status bar will indicate the problem. The user will then have the opportunity to correct the problem and try again.

4.3.4.2.a Rollover icons for the folding tools



4.3.4.2.b Creasing

The location of creases (valley or mountain folds) may be indicated in several ways as described by the first four of Huzita's origami axioms in §3.2.2. It is important to include the fifth and sixth axioms in a complete origami modelling system, but they are not discussed in this implementation. The four methods of making a fold are then:

- Fold through two points (e.g. join two vertices with a crease)
- Align two vertices
- Align two edges

- Fold an edge to itself through a certain point so that the fold is perpendicular to the edge.

Each of the four different types of folds has its own button on the toolbar. To create a crease that joins two points, the user selects the two vertices to be joined. Then the user chooses the "Join two vertices" button from the Creases toolbar or menu. A new crease created in this way may cross other edges, and if so, new vertices will be added at the intersections. The new crease may also pass though already existing vertices.

To create a crease that is the result of aligning two points, the user chooses the two points that are to be aligned and clicks the "Align two vertices" button from the Creases toolbar or menu. The bisector of these two points, if one exists in the plane of the paper, is calculated. New vertices that arise from this crease line crossing other creases are created, and the new edges are added.

To align two creases (not necessarily parallel), the user chooses the two creases that are to be aligned, and then selects the "Align two edges" button or menu item. The program will find the bisector of these two edges in the plane of the paper, if one exists, and add the crease(s) and any new vertices to the model as before.

To create a crease that is perpendicular to an edge, and passes through a given point, the user chooses the edge and the point that are to be used in this construction. Then the user clicks the "Make a perpendicular edge" button from the Creases toolbar or

menu. The program will find the perpendicular to the given line, which also passes through the given point. The new edge(s) will be added, and new vertices added as in the case of joining two vertices.

To bend the paper along crease(s) defined previously, the user must choose the edge(s) to bend along, and then press the "Bend" button. Next, the status bar will show "Click and drag the flap", and the user selects a vertex, edge or face to be dragged with the mouse. As the user moves the mouse with the left button depressed, the flap will be bent along the selected edge(s), and the other faces will not move. The flap will not move if the chosen set of edges does not allow folding without curved surfaces. When the button is released, the flap will stop moving. If the flap is very close to being folded exactly 90° or 180°, the program will automatically correct the position to 89.5° or 179°, respectively. This is a temporary solution, to compensate for the lack of thickness in the paper representation, and to allow the user to differentiate between vertices and edges that lie very close together. For a more rigorous solution to this problem, see the discussion in §6.3.1.

4.3.4.2.c Simple Folds

A *simple fold* is one that involves folding along one fold-line, or a set of coincident lines, only. For each of the pre-defined folds, the user must first indicate the locations of all creases that will be involved in the fold. This is called *pre-creasing* and is standard

practice in real origami. That is, the user must use a combination of the creasing tools described in §4.3.4.2.b to do the pre-creasing required for any of the following folds.

To create a valley fold (or mountain fold, respectively), the user chooses the edge(s) to fold along, and a vertex that is on the moving part of the paper. If more than one edge is chosen, then these edges must be coincident. Next, the user chooses either the "mountain fold" button or the "valley fold" button. The paper is then automatically folded along the chosen edge(s) to 89.5° by default. If the user wishes to continue the fold to 179°, he or she need only click on the "mountain" or "valley" button again. The paper is animated during mountain and valley folds, so that it appears that the paper is actually folding. The paper is not allowed to fold beyond 180°, since this would mean that the paper is passing through itself. However, other means of collision detection are not used (i.e. if two non-adjacent faces of the paper happen to intersect during the fold, the intersection will be allowed to happen). For a discussion of the difficulties of collision detection under the scope of this project, see §6.2.3.

4.3.4.2.d Complex Folds

Complex folds are those that involve the simultaneous manipulation of more than one vertex or edge. This also means that some faces move in different directions. This is in contrast to simple folds, where all faces can be moved similarly. Thus an origami system implementing complex folds must be able to handle the manipulation of more than one vertex at a time. This implementation achieves basic complex folding, but

does neither animation nor collision detection for interpenetrating faces. See §6.2.3 and §6.3.2 for more details.

Pre-creasing is essential for complex folds. All complex folds (e.g. reverse, squash, rabbit-ear, petal) are done by first creating all necessary creases using the crease buttons described previously, and then choosing the appropriate button.

To create a reverse fold, the user chooses two distinct edges (or any number of edges, as long as they are coincident to two distinct edges), and a vertex that is to be moved in the fold. Then the user chooses either the "Inside reverse fold" button or "Outside reverse fold" button. The program will calculate the final locations of the faces involved in the reverse fold, and move them. No animation is performed here, since reverse folds cannot generally be done without bending some faces, and unfolding some of the model slightly. Just the final result of the reverse fold is displayed.

To create a squash fold, the user chooses the edges bordering the flap to be squashed, and the vertex at the tip of the flap. The program can then automatically determine the location of the two new creases that appear during a squash, and then perform the fold. The squash fold is animated as it is folded, if doing so would not require curving the paper.

A petal fold on a point is essentially two reverse folds (on the sides) and then a valley fold to bring the point up to the top. To petal fold a point, the user doesn't need to

define all the creases involved. The user chooses the point that is to move during the fold, and the two folded edges that are to be used. Then the program automatically determines the location of all creases (if the fold is possible) and completes the fold.

To petal fold an edge, the process is the same as for petal folding a point, but this time the user selects the edge that is to be moved instead of the point.

For a rabbit-ear fold, the user selects the flap that is to be folded, and presses the "Rabbit-ear fold" button. The program automatically does the pre-creasing of the three valley folds, and adds these creases to the model. A rabbit-ear fold is essentially a valley fold followed by a reverse fold, so the program determines the final positions of the faces involved in the fold, and moves them to their final location. No animation is performed for rabbit-ear folds.

Open and closed sinks will be created similarly to reverse folds. First, the user creates and selects all the creases needed in the open or closed sink. Then the user selects the "Open sink" button or the "Closed sink" button. The program calculates the location of the sunken faces, and automatically creates the correct order of the interior layers of paper to complete the sink fold. Animation is not performed for sink folds.

4.3.4.3 Landmark Tools

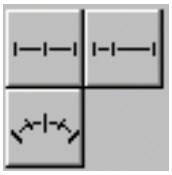


Figure 45. The landmark tools

The tools that are available to define landmarks on the paper and their rollover icons are:

 Make a vertex at the midpoint of an edge.

 Split an edge into two pieces at a certain proportion. The user provides the proportion as a ratio via a dialog box.

Split the angle between two edges into several equal angles and add edges at split points. The user provides the number of pieces via a dialog box.

- Split an edge into x equal sized pieces and add new vertices at split points. The user provides x via a dialog box. (This button not yet implemented.)

4.3.4.4 Camera and Virtual Trackball

Figure 46. The camera controls

The user is able to control the "camera" or viewpoint so that the model may be viewed from any angle and zoom factor. A virtual trackball allows the user to manipulate the view of the model using the mouse directly. The user clicks anywhere on the 3D display, and drags the mouse to rotate the model. The camera control also includes these functions, shown here with their rollover icons:

Zoom in/out, or in to a specific area

Tilt up/down.

Pan left/right.

Rotate the model.

4.3.4.5 Animation

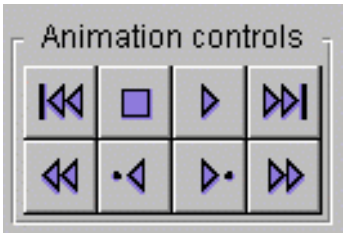


Figure 47. The animation controls

In the first version, animation occurs for simple folds only (i.e. mountain and valley folds). As a mountain or valley fold is created, the model is animated to show the

folding process. This is possible because the only parts of the model that need to be animated are the ones directly involved in the fold.

However, achieving a complex fold is more challenging. As a complex fold is performed, it is possible that other parts of the model need to move or be unfolded and refolded. Correct animation of this requires applying and solving geometric constraints on all parts of the model. Thus, complex folds happen all at once, i.e., the fold can occur, but animation of the folding process is not done. For further discussion of this, see §6.3.2.

The user has the following buttons to control the pace of steps through the model:

Play the animation.

Stop the animation.

Go to the beginning of the animation.

Go to the end of the animation (final model).

Go forward one step in the animation.

Go back one step.

Go forward one frame of the animation.

Go back one frame.

- A frame number indicator and step number indicator (on the status bar).

4.3.5 Menu Bar

Figure 48. The menu bar

The main menu bar has the following functions:

4.3.5.1 File

New: Start a new model from a flat piece of paper. The default will use a square of paper that is coloured on one side and white on the other. In general, the user may choose the paper shape and proportions, a starting base (e.g. preliminary fold, waterbomb base, bird base, etc.), colour of the paper, and texture on the paper.

Open: Open a previously saved model file for viewing and/or augmentation.

Save: Save the current model including the steps to make it.

Take snapshot: Save an image of the current contents of the central window.

Print model: Print the image of the current contents of the central window.

Print crease pattern: Print the image of the current crease pattern.

Quit: Quit the program. Prompt for save if necessary.

4.3.5.2 Edit

Undo: Undo the last step or any number of steps (if implemented).

Redo: Redo the last undone step.

Mark: Add a colour marking to an edge, vertex or face. Prompt for the object to be marked.

Remove: Delete a crease (edge) or vertex from the model that is no longer needed. A vertex may only be deleted if its adjacent edges can be combined into a single straight edge.

Invert: Change a valley crease into a mountain crease, or vice-versa.

4.3.5.3 View

Zoom in/zoom out: Set the amount of zoom on the model display window.

Pan left/pan right: Move the camera left or right in the main display window.

4.3.5.4 Paper

Flip horizontally/flip vertically: Flip the entire model over, as if it were lying on a table.

Rotate clockwise/rotate counter-clockwise: Rotate the entire model by a certain number of degrees (90°, 180°, or user-specified) in the plane of the table.

X-Ray: Select a face to make invisible, to allow viewing and selection of hidden points.

Change colour: Change the overall colour of the displayed paper.

Change texture: Apply an overall texture map to the paper.

4.3.5.5 Fold

The following folding functions are available under the Fold menu. For descriptions of these functions, please see §4.3.4.2.

Join two vertices	**Make a perpendicular line**	**Align two vertices**
Align two edges	**Valley fold**	**Mountain fold**
Inside reverse	**Outside reverse**	**Pull paper out**
Tuck paper in	**Pleat**	**Crimp**
Rabbit-ear	**Squash fold**	**Petal fold a point**
Petal fold an edge	**Open sink**	**Closed sink**

4.3.5.6 Landmarks

Divide an edge equally: Divide the selected edge into *x* equal parts.

Divide an edge using a ratio: Divide the edge into 2 parts using a ratio.

Divide an angle equally: Divide the angle into *x* equal parts.

4.3.5.7 Animate

Play: Start the animation.

Stop: Stop the animation.

Forward: Move forward one step in the animation.

Backward: Move backward one step in the animation.

Go to step: Go to the specified step in the animation.

4.3.5.8 Settings

Each of the following is an on/off toggle.

Show vertices/edges/faces: Show or hide the vertices, edges and/or faces.

Advanced mode: Choose advanced or beginner mode.

Shaded mode: Choose diagram or shaded mode.

View only mode: Choose view only or fold mode.

Show crease pattern: Show or hide the crease pattern.

Show instructions: Show or hide the folding instructions text box.

Show camera controls: Show or hide the camera controls.

Show animation controls: Show or hide the animation controls.

4.3.5.9 Toolbars

The following are also on/off toggles.

Main toolbar: Show or hide the main toolbar.

Simple folds toolbar: Show or hide the simple folds toolbar.

Complex folds toolbar: Show or hide the complex folds toolbar.

Landmarks toolbar: Show or hide the landmarks toolbar.

4.3.5.10 Help

How to...: General instructions about how to use the program. The help function will eventually explain how to start a new model, use all creasing and folding tools, save a folding sequence for later use, and view a previously saved model, among other things.

About: Version, date and author information.

4.3.6 Status Bar

Figure 49. The status bar.

The *status bar* is used to provide various messages to the user. The status bar indicates when the program is loading, saving, or otherwise doing some calculations that take control away from the user momentarily. It also prompts the user for input when needed, and indicates the nature of any problems due to improper input.

Labels at the right end of the status bar indicate the step number of the current model procedure and the current step number of the animation. Toggle indicators for the modes described in §4.2.4 also appear on the status bar.

4.3.7 Instruction Text Box

Figure 50. The instruction text box.

A set of general instructions for folding the model is generated automatically by the program, based on the folding sequence provided by the user. These instructions will be displayed in an editable, scrollable text box. The user may change the instructions to be more specific by clicking in the text box and typing the changes. This function will only be allowed when a user is designing a new model, not when he or she is playing back a previously saved model. The user may choose to hide or show the text box at any time.

4.3.8 Crease Pattern Display

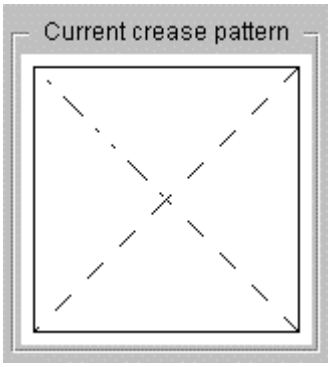


Figure 51. The crease pattern display.

The crease pattern of the current state of the model is displayed in a box. The crease pattern is displayed using the standard dash-dash patterns for valley creases and dot-dot-dash pattern for mountain creases. Creases on the paper that are not folded or do not yet have an orientation are displayed as a solid line. The user may choose to hide or show the crease pattern at any time.

4.3.9 Tool Tips

Figure 52. A tooltip.

Whenever the user holds the mouse over an element of the user interface (e.g. a button or label) for a few seconds, a short message called a *tooltip* appears to give a hint of what the tool does, or how it is used. An example is shown in Figure 52.

Chapter 5

DATA STRUCTURES

5.1 Representation of the Origami Model

Although various data structures for representing planar graphs (like our crease pattern) already exist, there is an inherent structure within the folding of paper that compels us to use something unique. A standard winged-edge model (Glassner, 1991) probably is not the best choice. This section describes the design of an origami model data structure that is a modification of the one described by Miyazaki et al. in their paper (Miyazaki and others, 1996).

5.1.1 Overview

The data structure that represents the folded piece of paper must be dynamic to allow the addition of new creases at any time. Folds in any direction, and through any number of layers of the paper must be possible. The paper must not appear to stretch or have holes in it. The data structure should be time and space efficient since it will be manipulated and displayed in real time. Fortunately, due to the nature of paper

folding, the total number of faces in a model should not grow so quickly as to cause a problem.

5.1.2 Origami Elements

The origami model is represented by a complex data structure representing the faces, edges, and vertices of the model. Each element (vertex, edge or face) is represented only once in the system, with relationships between elements indicated by using pointers. An object-oriented design methodology has been used. Thus faces are linked to their surrounding vertices and edges, edges are linked to their endpoints and adjacent faces, and vertices are linked to their surrounding faces. Using these relationships, each element can access every other element. The structures have been designed to trade off space for simplicity and efficiency of the algorithms that manipulate the data.

5.1.3 Binary Tree Structure

Generally in origami, and specifically in this implementation, faces are convex and fold lines are straight. Thus when a fold occurs on a face, that face will be divided into exactly two parts. And when a fold occurs on an edge, the edge can be broken into, at most, two parts.

Since each folding operation naturally divides faces and/or edges into two parts each, faces and edges are represented using binary trees. A node represents a face (or edge),

and its two child nodes are the two faces (edges) that the original node is divided into by a folding operation.

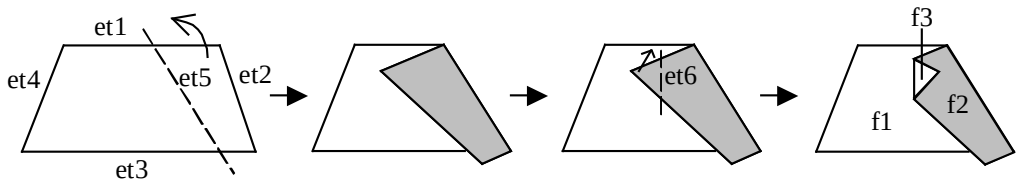


Figure 53. A piece of paper that has been folded twice.

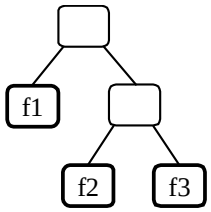


Figure 54. The face tree for the folded paper in Figure 53.

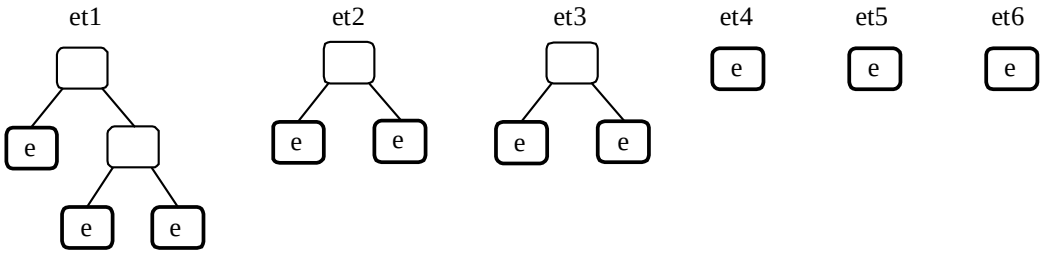


Figure 55. The set of edge trees for folded paper in Figure 53.

Each node in the tree has a unique identification number, and elements are labelled and inserted into the tree so that the resulting binary tree is a search tree. This makes

searching through the tree quite efficient when inserting new nodes or looking for previously created nodes.

Each node is also given a step number, indicating the step in which it was created or changed. Thus the binary trees record the face and edge division processes. Retaining all previous folding operations in this way is important for playing back the folding sequence.

5.1.4 Common properties

There are some properties that are common to every origami element in the program in addition to the identification number and step number as just described. Every face, edge and vertex has a colour. Each element also has a function to calculate the identification numbers of its left and right children (if any), and a function to draw itself to the 2D and 3D display windows.

5.1.5 Faces

A single binary tree contains the face information of the entire origami model. The root of the tree is the initial sheet of paper, and the leaves of the face tree represent the current states of all the sub-faces. In Figure 54 the highlighted nodes are the only faces that are part of the current state of the model. These are the only nodes that are rendered to the display.

The default colour of a face is a static variable that is shared by all the faces and describes the colour of one side of the paper. The other side is always white by convention.

Each face node also contains the following information, as illustrated in Figure 56:

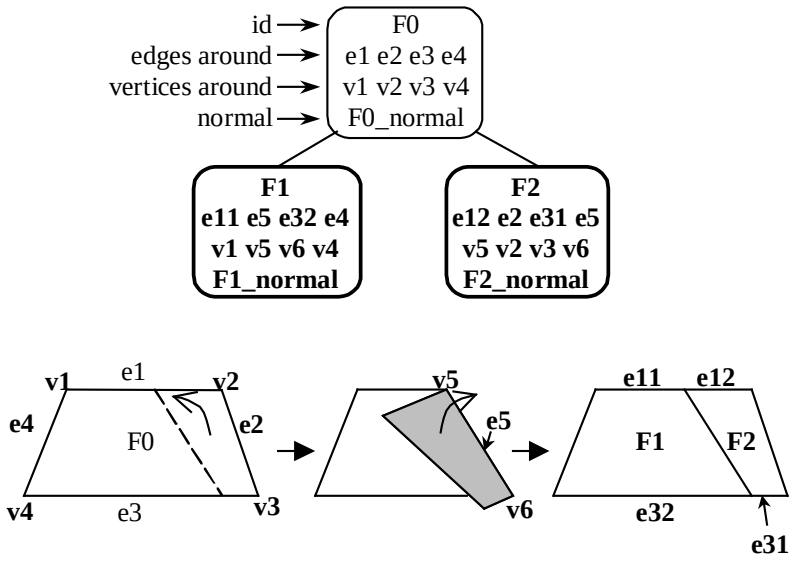


Figure 56. The detailed face nodes for this example of a divided face.

edgeList: The list of pointers to the edges that surround this face, in clockwise order around the white side of the paper. Note that each of these edges has a direction independent of its order among all edges around the face. The direction of an edge relative to any face depends on the circumstances when that edge was created. In Figure 57, edge e5 is defined in a clockwise direction around the white side of face F1. However, since each edge in the model is defined only once, it is inconsistent for edge

e5 to also be defined as clockwise around face F2. Thus we cannot assume that the edges are in one direction around any given face.

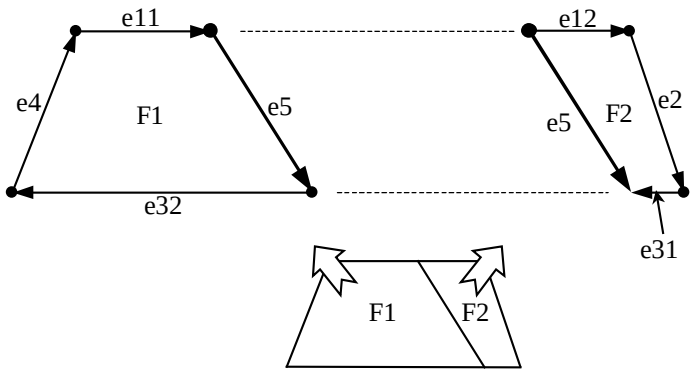


Figure 57. The direction of an edge is independent of its order among all edges around a given face. Note edge e5.

Edges are guaranteed however, to be listed in clockwise order around the face. See the "edges around" for face nodes F1 and F2 in Figure 56.

verticesAround: The list of pointers to the vertices that surround this face, and the value of the internal angle at each vertex, relative to the current face. The vertices are specified in clockwise order around the white side of the paper. These vertices are used as the corners of the polygon when the face is rendered. Compare Figure 58 with the "vertices around" for face node F1 in Figure 56.

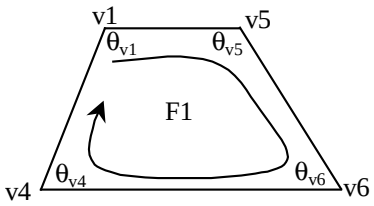


Figure 58. Vertices are always specified in clockwise order around the white side of the face.

normal: The calculated normal for this face. This variable is stored rather than calculating the normal each time the face is displayed. The normal is recalculated whenever the face is moved.

The face object also has functions that it may perform on the faces or other parts of the data structure.

5.1.6 Edges

The edge tree structure is similar to the face tree structure, except that a new tree is created every time a new straight edge is added to the model. Therefore, the root of each edge tree represents a single straight line segment, and the rest of the tree represents the results of folding operations on that edge. The leaves of the tree contain the complete current state of the edge. Thus to draw the edge at any time we draw only the set of leaf nodes.

The variables that are shared by all the edges are the default colour of an edge and some constants indicating the type of the edge (valley, mountain, flat or raw).

Each edge node contains the following information, as illustrated in Figure 59:

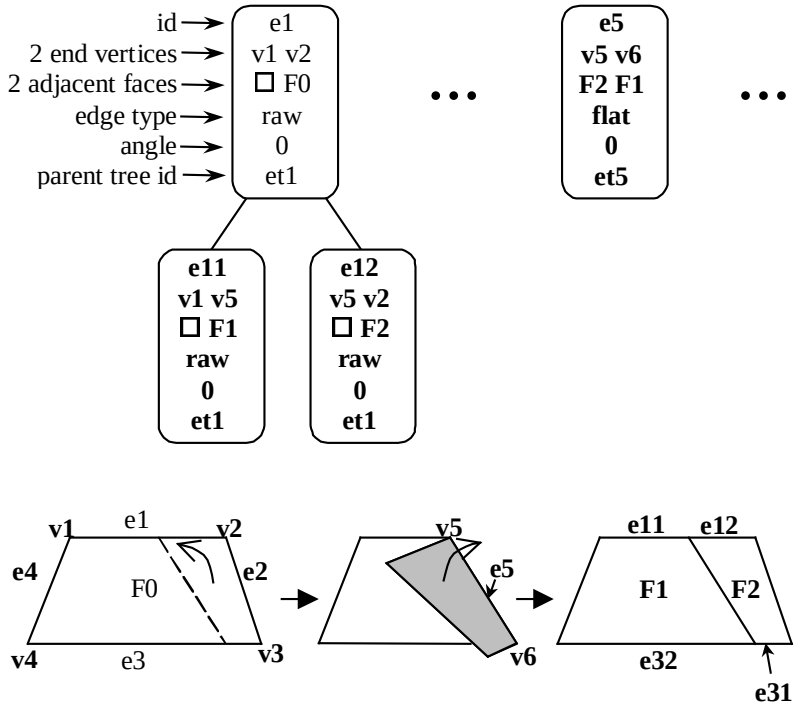


Figure 59. Two complete edge trees for this example. The three other edge trees are not shown.

vertex1, vertex2: Pointers to the two endpoints of this edge. Having an order to these vertices gives the edge a direction. See Figure 60.

face1, face2: Pointers to the two faces that are adjacent to this edge. Face1 is to the left of the edge *(vertex1, vertex2)*, and Face2 is to the right of the edge, relative to the white side of the paper. See Figure 60.

The order in which the adjacent vertices and faces are specified is very important, and must be maintained throughout. We assume that the white side of the paper is facing up in Figure 60.

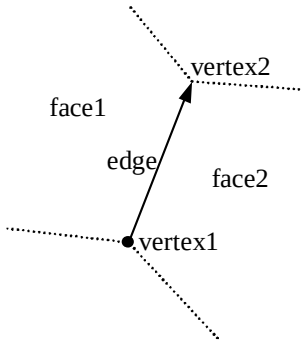


Figure 60. The order of an edge's two adjacent vertices and two adjacent faces.

edgeType: The type of this edge relative to the white side of the paper: valley, mountain, flat or raw.

length: The length of the edge. This value is set when the edge is created and is constant.

angle: The angle that the edge is folded, relative to the white side of the paper, i.e., the dihedral angle between the two faces that meet at this edge. The angle is in radians and lies within the range $[-\pi, \pi]$.

edgeTreeId: The identification number of the edge tree that this edge belongs to.

Each edge object also has functions that it may perform on itself or other parts of the data structure.

5.1.7 Vertices

Vertices are represented using a dynamic list of lists. The main list has one entry for every vertex. In order to store the vertices as they change location in space, each of the entries in the main vertex list is another list. Each element of this list is a vertex node. When accessing a particular vertex, we access the last element of that vertex's list.

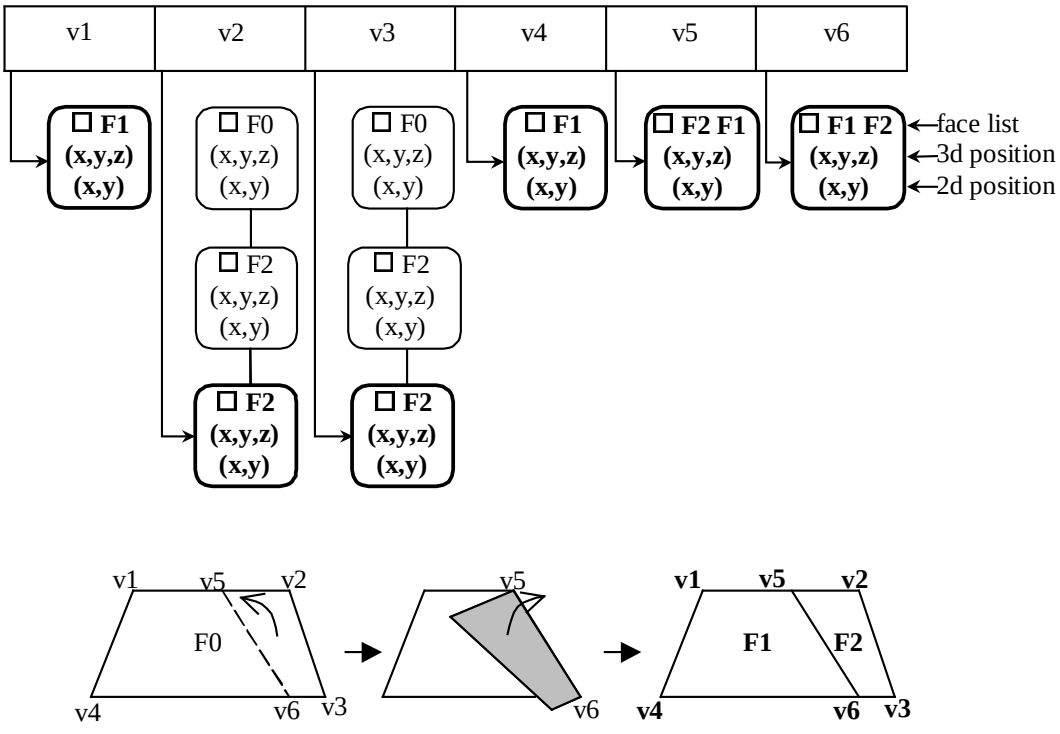


Figure 61. The vertex table for this example.

The static variables that are shared by all the vertices are the default colour of a vertex and the last used identification number for the vertices.

Each vertex node contains the following information:

faceList: Pointers to the faces adjacent to this vertex, in clockwise order.

location3d: A three-dimensional vector representing the location of this vertex in 3-space.

location2d: A two-dimensional point representing the location of this vertex on the crease pattern display.

Each vertex object also has functions that it may perform on itself or other parts of the data structure.

5.2 Effects of Adding Creases

When a new crease (edge) is added to the model, some faces and edges are divided into two pieces (creating some new faces and edges), some entirely new edge trees are created, and some vertices are created. No elements of the model move when an edge is added without folding. However, locations of new vertices must be calculated from intersections of existing edges and the new edge(s). Finding these intersections is non-trivial, and uses an algorithm based on ray-tracing.

5.2.1 Joining Existing Vertices

When a crease is added to a single face, two new faces are created and added to the face tree as children of the node containing the initial face. The endpoints of new edges must have been existing vertices. The new faces and all adjacent edges and vertices are updated to reflect the changes.

When an edge is added which crosses several faces (but still joins existing vertices), each of the edges that is crossed by the new edge is split into two pieces and adjacent vertices and faces are updated. The new edge is also split into pieces as it crosses the existing edges. Then each face that is crossed by part of the new edge is treated as a single face being split into two pieces by a portion of the new edge.

5.2.2 Simultaneously Creating New Vertices

For all creasing tools other than "Join two vertices", a crease line is formed implicitly. Two landmarks (vertices or edges) are chosen, and the location of the new crease line is calculated based on these. To add the new crease(s), each intersection of the new crease line with an existing edge is found, and marked with a new vertex. These vertices are then joined with creases as described in the previous section (§5.2.1).

5.3 Effects of Folding

Folds are performed on existing edges and vertices. Only the vertices are explicitly moved. The paper on one side of the fold line is held still while the selected part of the paper moves.

To fold the paper, the user selects the edge(s) to fold along, and a vertex on the moving part of the paper. In real origami when a selected face is moved, some other faces in the model are forced to move along with it. These faces are those that are attached to the selected moving face directly by sharing an edge, or by simply overlapping a moving face.

5.3.1 Determining the Moving Faces

Once the user has selected the moving part of the paper and the line(s) to fold along, the origami system determines which faces and vertices will be moved during this fold.

Fisher (Fisher, 1996) presents an algorithm for finding faces that move during simple folds. The algorithm assumes that we are given the part of the paper that moves (a vertex or an edge), the fold line along which the creases are made, and the type of fold (valley or mountain) along that edge. Note that his algorithm also assumes that there is just a single fold line. Fisher claims that this algorithm is able to find all moving faces, including those that overlap a known moving face, but are not directly attached to a moving face.

Miyazaki et al. also present an algorithm similar to Fisher's to determine moving (and folded) faces (Miyazaki and others, 1996). Their algorithm is correct only for three types of folds: folding up (to 180°), bending (less than 180°) and tucking in (a reverse fold at exactly 180°). Their algorithm does not find moving faces for folds involving more than one fold line at a time.

The user must select one vertex that is to be moved, and the edges to fold along. When choosing the edges to fold along, the user must be sure to select the edges on all layers of the model that are to be folded. In order to do this easily, the user only needs to double-click one of the edges. All collinear edges will then be selected.

The algorithm used in the current implementation finds all faces that are attached to a moving face, and thus are also moving, in the following manner:

Algorithm 5.1

1. The vertex that the user selected is called a *moving vertex*. Mark it.

2. Any face attached to a "moving vertex" is a *moving face*. Mark each of them. Unmark this "moving vertex" once all its faces have been marked.

3. Go through all the vertices that surround (are adjacent to) each of the marked faces. Any vertex from this set is a "moving vertex" if and

only if it does not lie on any of the fold lines. Mark these "moving vertices".

4. Repeat from step 2 until there are no more "moving vertices".

5. If all of the faces in the model are marked as "moving faces" after this process, then the user did not choose enough lines to fold along. Unmark everything and indicate the error to the user.

This algorithm assumes that the user chooses one vertex and any number of edges to fold along. The algorithm can easily be extended to allow the user to choose any number of moving vertices as well.

Once the "moving faces" have been determined, they may be moved using one of the folding tools. The new locations of moving vertices are then calculated and the display of the model is updated.

Note that Algorithm 5.1 does not determine all faces that should move. It handles only those faces that are directly or indirectly attached to a moving face. It does not handle faces that simply overlap a moving face. Overlapping faces cannot be determined using the relationships provided in the current data structure, and so further work must be done to handle them. A discussion of this is presented in §6.2.3.

5.3.2 Moving Faces in Simple Folds

The simple folds are mountain folds and valley folds, i.e. those that involve simple rotation of faces around a single fold line. In a simple fold the user selects one moving vertex and one line to fold along. The chosen fold line may consist of several collinear edges, and may be through several closely parallel layers of the paper.

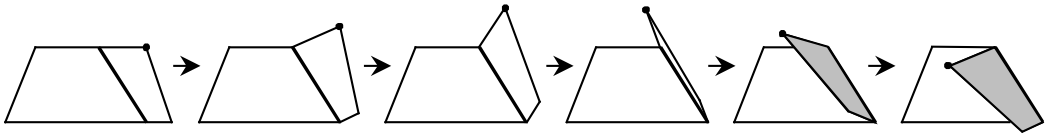


Figure 62. Animation of a simple fold.

Once the moving faces are determined as described in §5.3.1, the vertices belonging to the "moving faces" are rotated about the fold line. Since this process involves motion of all moving faces in the same way, animation of simple folding operations is possible. This is accomplished by simply rotating in small increments, and redisplaying the model at every stage. The current implementation uses 20 steps through 90° of rotation.

5.3.3 Moving Faces in Complex Folds

Complex folds involve the manipulation of more than one point at a time, or the use of two or more non-collinear fold lines. An example of a complex fold is the reverse fold, which has been implemented in the current system. Generally in a reverse fold, the user selects the moving vertex and two distinct lines to fold about. In special

circumstances (such as when the paper is already folded to about 180°) the two distinct lines may actually be almost collinear. This special type of reverse fold is described as "tucking in" by Miyazaki et al. (Miyazaki and others, 1996) and is implemented in their system. The current system handles not only this special type of reverse, but all types of reverse folds.

To begin a reverse fold, we again determine the "moving faces" using Algorithm 5.1. To perform a reverse fold correctly, it is important to notice that the final resting place of each moving face is actually its reflection through the plane determined by the two distinct fold lines. So given two distinct fold lines, we find equation of the unique plane that passes through both of them. Then for every vertex that belongs to a "moving face", its perpendicular distance from this plane is calculated. Using this distance, we can find the reflection of the vertex through the plane. See Figure 63.

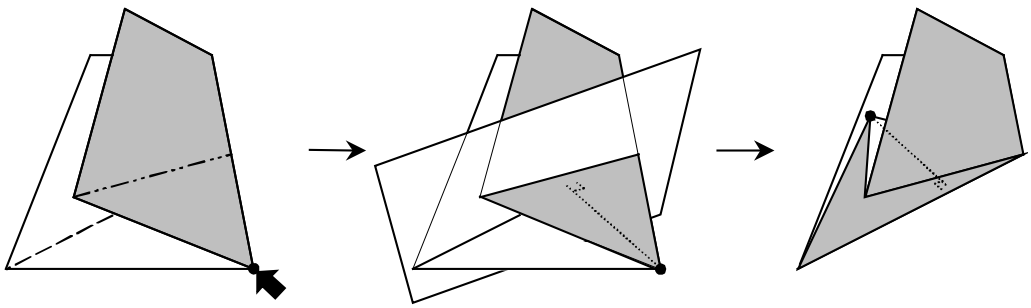


Figure 63. A general reverse fold, done by reflecting through the common plane.

Since the process of forming a reverse fold cannot generally be done without bending some faces or unfolding part of the model, animation is not done during reverse folds.

Instead, the fold appears to be done all at once, i.e. the display is updated before and after the reverse fold. Producing animation for reverse folds (and other complex folds) would involve constrained geometry and is beyond the scope of this project. See the discussion in §6.3.2 for more details.

The other complex folds can be handled with the help of a reverse fold. A squash fold is essentially a reverse fold followed by one or two simple folds (see Figure 64). A petal fold consists of two reverse folds followed by a simple fold. A rabbit-ear can be achieved using two valley folds followed by a reverse. The sink folds can be thought of as a special case of reverse folds, where more than two distinct edges determine the plane of reflection.

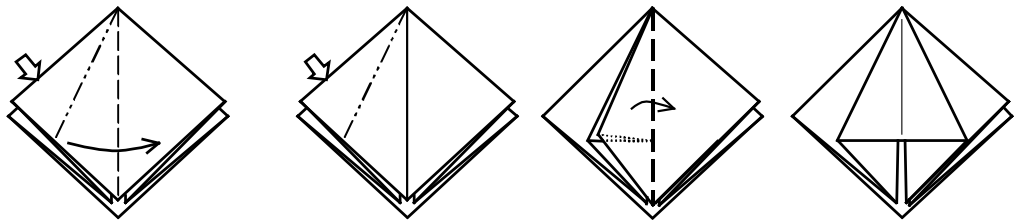


Figure 64. A squash fold using only a reverse fold followed by a valley fold.

Since we have mechanisms for simple folds and reverse folds, it is easy to extend the design to other complex folds. An example of the breakdown of one complex fold is shown in Figure 64. Thus the current system is able to model all types of complex folds previously described. Note, however, that accurate animation of these complex folding processes is not yet possible.

Chapter 6

RESULTS AND EVALUATION

6.1 Jo's Origami Interface (JOI)

The origami system described in Chapter 4 has been partially implemented in a program called "Jo's Origami Interface (JOI)". The functionality of that system and fundamental implementation issues are presented in this chapter.

6.1.1 Implementation Achieved

The complete graphical user interface for the origami program has been implemented as described in §4.3. The data structure that represents the origami model has also been implemented with several functions for manipulating the model including adding edges, splitting edges, splitting faces and adding vertices at given locations. As the user manipulates the paper model, editable textual instructions are automatically generated and appear in the text box. Picking and selection are working, and the user is permitted to select and deselect any combination of vertices, edges and faces using the mouse button. A double-click of the mouse button causes all coincident edges to be

selected or deselected. The virtual trackball allows the user to click and drag the mouse in order to view and manipulate the origami model from any direction.

The user may start a new model with a square piece of paper and change the back colour of the paper at any time (the front is always white). The faces, edges and vertices are displayed, but each group may be toggled individually to be invisible. To create creases on the paper, the user may add new vertices at the midpoint of any edge, and join two existing coplanar vertices with a new edge.

Simple folds, including both mountain and valley folds, are possible in this origami system. Each simple fold is animated during folding and automatically folds 89.5° each time the fold button is pressed. Simple folds are not permitted past 180° (since this would cause the paper to pass through itself).

One type of complex fold, the inside reverse fold, has been implemented in the system. Reverse folds are not animated, since doing so would involve using curved surfaces or unfolding parts of the paper. Consequently, reverse folds appear to be folded "all at once", i.e. the faces are moved immediately to their correct final resting places in the fold without animation.

The rest of the complex folds are not explicitly implemented with their own buttons, but they are possible to achieve in the current system. Many complex folds can be

performed as a combination of mountain, valley and reverse folds as described in §5.3.3.

The following figures show several views of a paper airplane modelled in the origami system.

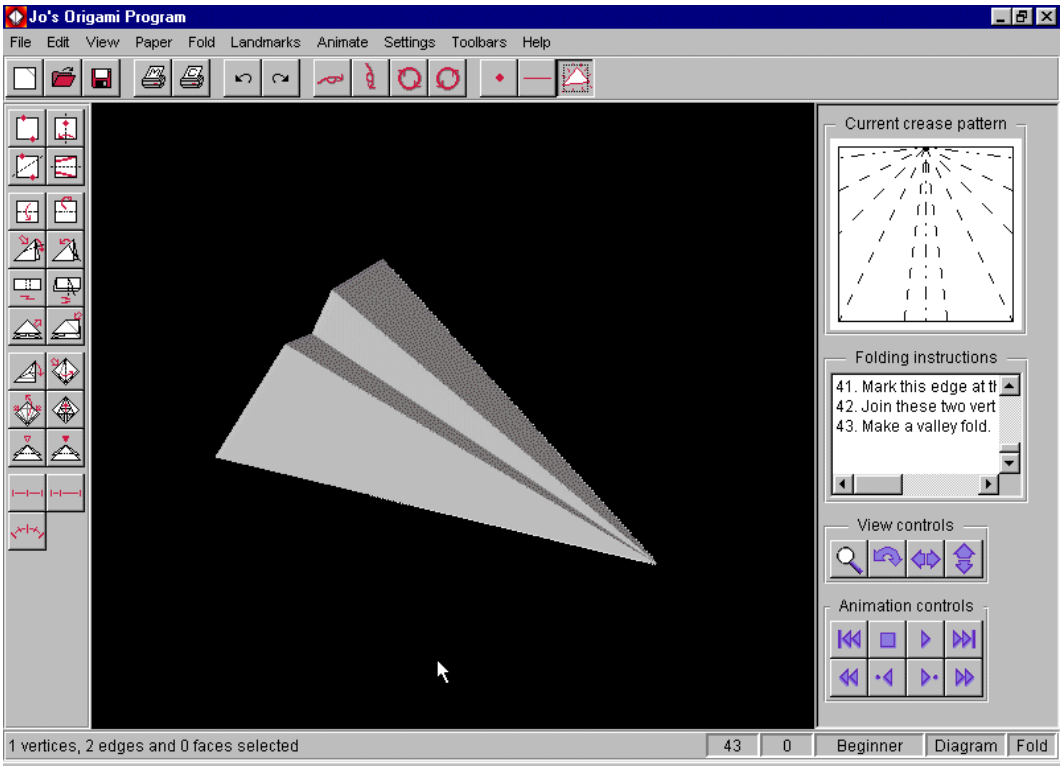


Figure 65. A full screen capture of a completed origami model.

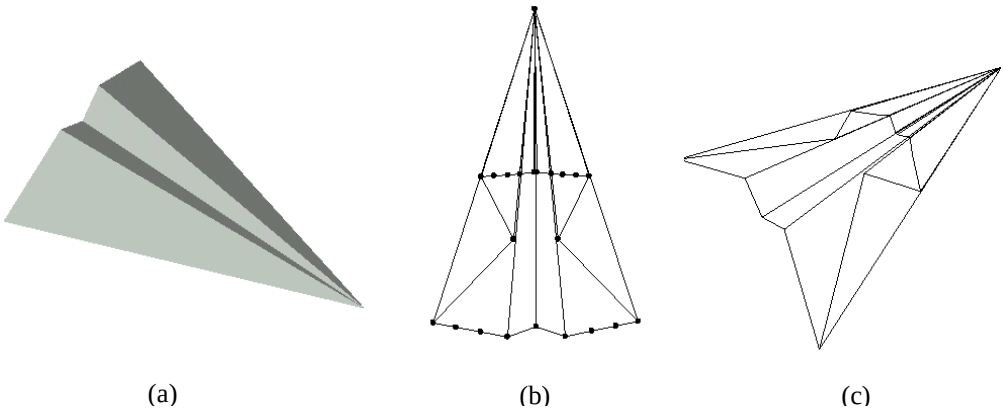


Figure 66. Three views of the paper airplane.
Normal shaded paper airplane in (a), shown with
vertices and edges in (b), and wireframe only in (c).

6.1.2 Limitations of the Implementation

In this section, aspects of the original design described in Chapter 4 that have not yet been implemented due to time constraints are discussed. The current implementation of the origami model also has some more difficult problems, most of which are due to the paper's lack of thickness. These issues are discussed in detail in §6.2.

The data structure has been designed to allow for infinite undo and redo, as well as animation of the final model. However, no undo, redo, animation or replay of the steps used in creating the origami model has been implemented yet.

The camera controls, other than the virtual trackball, are not implemented. The user cannot "flip the paper over" or use the "rotate the model" buttons. This a minor

inconvenience since the virtual trackball is sufficient for most viewing requirements. The implementation of these controls would be reasonably trivial.

Currently, the user is not allowed to provide any parameters to the creasing, folding, and landmark buttons. For example, the angle of a fold is always incremented by exactly 89.5° when the valley or mountain button is pressed. In a future version, the user should be able to specify any angle to fold. However, the method of using a default value for functions is sufficient to show that creasing and folding work correctly.

The only method currently available to add new edges to the model is to join two existing vertices. Ideally, it should be possible to add new creases using any of Huzita's six origami axioms (see §3.2.2). Inclusion of these powerful folding tools is essential to fully simulate the power of geometrical and mathematical paper folding.

6.1.3 Programming Language and Toolkits

The program has been implemented under Windows 95 on a PC using Java and various Java libraries. This makes the final product available to more people that would actually like to use it, such as children in schools. The computational requirements are appropriate for implementation on a personal computer. The Java language was chosen because it is platform-independent and object-oriented, and I wanted to get some experience using this popular programming language.

The user interface is programmed using the Java Abstract Windowing Toolkit (AWT) and Java Foundation Classes (Swing). They are freely available on the Internet. For details please see Appendix A.

All 2D and 3D graphics utilities such as lighting, perspective projection and the virtual trackball are implemented using OpenGL. OpenGL is also a platform-independent API, and was chosen over Java3D because at the time of this writing, Java3D was still in Beta testing. The Magician OpenGL bindings have been indispensable in the development of this program. For more information about Magician, please see Appendix A.5.

6.2 Difficulties Encountered

This section discusses some of the difficulties encountered during implementation and how they are being handled.

6.2.1 Folding Flat

The representation of paper in this system is an abstraction of the way real paper might behave. Unlike a real origami model, the virtual paper has no thickness, and faces remain perfectly flat. Thus in the origami system, as originally conceived, it is possible to fold a flap of the model flat against another face, so that the dihedral angle is exactly 180°. Unfortunately, this does not model real paper very well, and folding flat introduces user interface problems. If a piece of virtual paper is folded to exactly 180°,

a face of the paper becomes (partially) superimposed onto another face. Subsequently, when the user tries to select one of the vertices that has become overlapped, it is more difficult to determine which vertex is actually chosen. There are several interaction techniques that can be used to deal with this, but they are not discussed here.

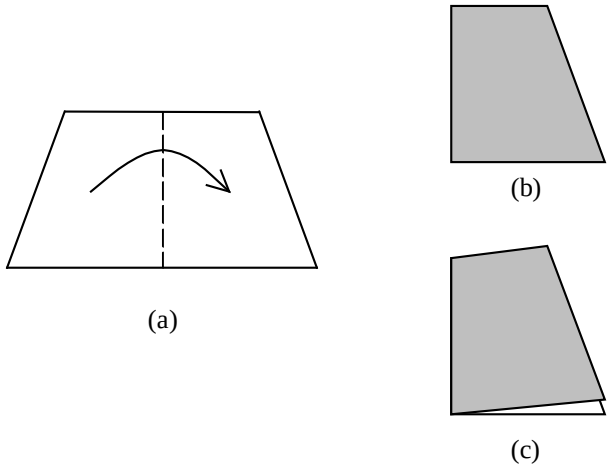


Figure 67. The original paper (a) is folded to exactly 180° in (b), but only 179° in (c).

To temporarily avoid this problem, the user is currently allowed to fold the paper only exactly 89.5° in a simple fold[4]. The user may fold twice in the same direction in order to achieve a dihedral angle of 179°. At this angle, the paper appears to be folded flat, but there is enough space to differentiate closely aligned vertices and edges. This makes the model visually acceptable, and simplifies the selection of edges and vertices.

However, this solution is weak since a few carefully chosen consecutive folds can cause the virtual paper model to become a physical impossibility. A better solution is

to fold the virtual paper to exactly 180° so that closely parallel faces are actually parallel, and then use a modified rendering technique suggested in §6.3.1.

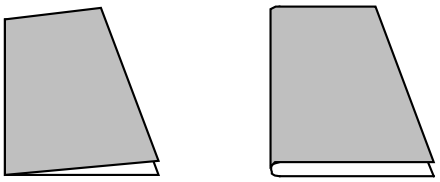


Figure 68. Paper (a) folded to exactly 179° compared to 180° with simulated thickness in (b).

6.2.2 Rendering of Faces

Since OpenGL has built-in capabilities for rendering, the current origami system depends on OpenGL to remove hidden surfaces. Unfortunately, polygons that lie very close together are not always rendered properly by OpenGL. If two polygons are exactly coplanar and different colours, OpenGL is inconsistent in its choice of which polygon to render at any given time. This problem might be alleviated by manually storing the layering of faces in the origami model, relative to the viewer's location. However, the implementation of a good hidden surface algorithm is beyond the scope of this project.

The basic OpenGL implementation is also not capable of rendering concave polygons. In real paper folding, concave faces are almost never created. However, it is possible

[4] Note that in complex folds, any value for the dihedral angle is possible.

to construct concave faces using the virtual origami system. Despite the fact that no concave faces ever appeared during testing of the origami system, support for rendering concave faces should be added for robustness.

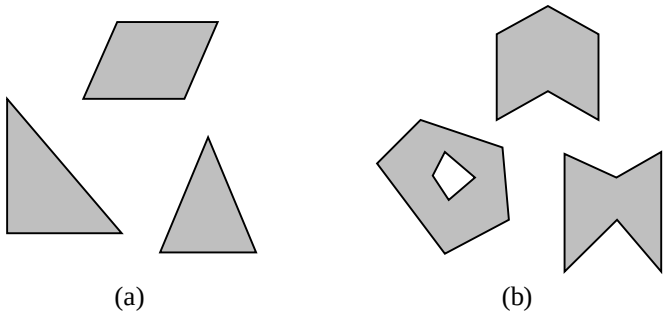


Figure 69. Convex polygons (a) can be rendered using OpenGL, but concave polygons (b) are not guaranteed to be rendered correctly.

When a face of the origami model is first created, its vertices are coplanar. Folding operations, when implemented correctly, should not cause vertices to vary from their original shared plane. However, numerical drift in computations, especially rotations, might cause the vertices that define a particular face to cease being coplanar while the model is manipulated. Moreover, OpenGL would have difficulty rendering these non-planar polygons. This problem has not yet arisen in the current implementation, so the inefficiency of continual testing for planar faces seems unwarranted. See Figure 70.

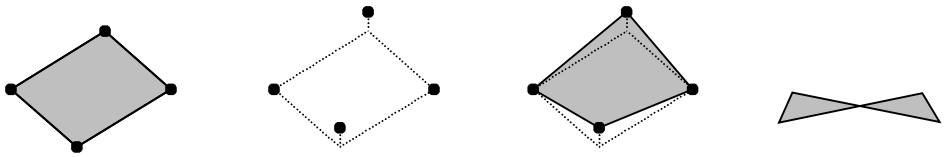


Figure 70. Vertices of a convex polygon that are coplanar (a) can drift from the plane as in (b). If the resulting polygon (c) is viewed from the side, a concave shape may be formed (d).

A well-known solution to this rendering problem is to break each face invisibly into triangles, which are guaranteed to be planar. However, nothing guarantees adjacent invisible triangles within a face to be coplanar, so more checking would be needed.

6.2.3 Interpenetrating Faces

In the current virtual origami model, every pair of faces has a strong and efficient geometric relationship, but their physical relationship is difficult to determine. It is possible that two faces will be moved into positions that would cause them to interpenetrate in the real world. For example, two non-adjacent faces in the model that overlap in three-space will affect one another as they are folded (see Figure 71(a)). Also, if a reverse fold is performed on one part of a double-layer flap, the inner layer should also be folded (see Figure 71(b)). Finally, two completely independent flaps may be folded so that one passes through the other (see Figure 71(c)).

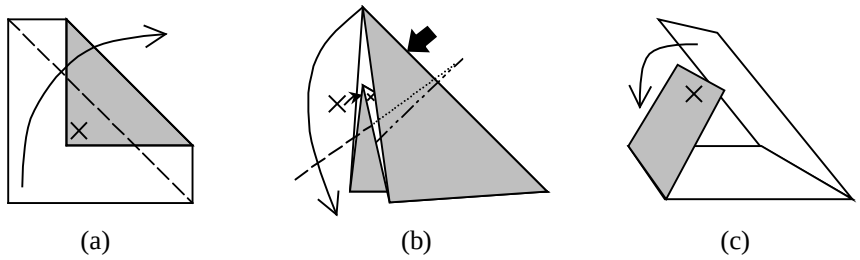


Figure 71. Different cases illustrating possible penetration of non-contiguous faces. The faces marked "x" will be affected by the moving faces.

Detecting faces that will interpenetrate during folding is difficult. The problem in the context of a piece-wise planar model such as this is more straightforward than collision detection between other types of surfaces. However, detecting collisions in the origami model is computationally time-consuming, and its implementation is beyond the scope of this project. Ignoring collisions does not seriously affect the folding process. Interpenetration is visible to the user, and this allows the user to create some beautiful, though physically impossible, virtual sculptures with the origami system.

6.2.4 Stretching of Faces

All faces in the origami model are polygons with constant edge length and constant internal angles. To manipulate the model, the user is allowed to either add a vertex (which does not change the size or shape of any of the faces) or perform a fold. A fold is essentially a combination of rotations and reflections of the faces in the model; rotations and reflections also do not change the size or shape of the faces. Therefore, the paper does not appear to stretch in this system.

6.3 Future Work

There are many unanswered questions about the applicability of using computers for origami. The first priority in future work would be to complete the implementation described in this document. The addition of curved surfaces is quite important. The current system is readily extended to allow multiple sheets of paper as in modular origami or cutting of the paper (known as *kirigami*). Animation of the process of folding a model is difficult, but essential to make the origami system useful in the way it was intended. A physically-based model of the origami object would likely best demonstrate the behaviour of physical paper. The existing implementation could be extended to take advantage of three-dimensional interactive displays by using stereo images and virtual reality input devices.

Several specific improvements to the current origami system have been considered and are presented in the following sections.

6.3.1 A Better Approach to Handling Paper Thickness

In this system, as in all previous known origami implementations, thickness of the paper is ignored. This has both advantages and disadvantages, many of which have already been discussed here. A better approach to handling the thickness of the paper was suggested by Alain Fournier (Fournier, 1998a) in personal discussions with him.

This fundamental approach to simulating thickness in the origami modelling system does not require the current origami data structure to be changed in any way. Instead, only the rendering method is modified. If we solve the rendering problems mentioned in §6.2.2, this eliminates the resulting work-around of folding to 179° as described in §6.2.1. So the user is actually allowed to fold to exactly 180°, which makes the rest of the manipulation of the paper much simpler and more accurate.

When a sheet of paper is folded in half, the two halves of the paper lie very close to one another. In the virtual origami model, they would be superimposed exactly. In Fournier's proposed method the two faces would be rendered parallel but slightly separated. The fold line, instead of being an abrupt change in direction between the faces, would be rendered as a number of narrow strips parallel to the fold line. This would make the fold line appear to be more realistic. Although no volume is actually added to the paper model, thickness is simulated by the separation of layers. Note that we maintain a limit to the radius of curvature.

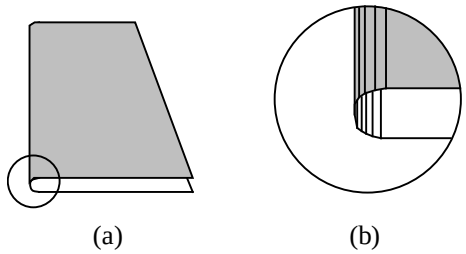


Figure 72. Paper (a) folded to 180° with simulated thickness, and a close-up view (b).

This method will probably work well for simple folds. However, more work needs to be done when two folds occur. For example, when the paper is folded in one direction, and then in the perpendicular direction, two creases i.e. four edges, meet at one vertex. Two of the edges will have tiny curved surfaces in one direction, while the other two might be in another direction. The general case of how to merge several folded edges meeting at a point has to be considered.

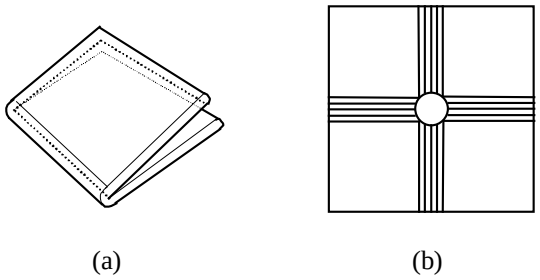


Figure 73. Paper folded in half twice (a) causes many intersections of strips (b) that have to be merged.

The best approach we have found so far is to use a tiny virtual sphere at the intersection point. Adjoining edges can be distributed around this sphere so that they are carefully blended. Finally, a portion of the tiny sphere can be drawn to complete the rendering.

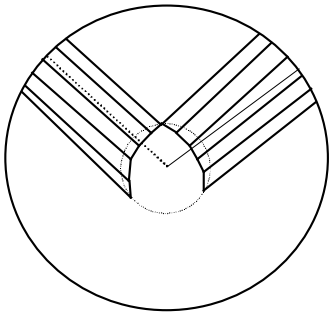


Figure 74. A close up of the corner of the folded paper shown in Figure 73(a).

This method has several obstacles that have not yet been resolved. Consider a sheet of paper that is folded in half, then in half again, then in half again, and so on. Real paper can only be folded in this manner seven or eight times before it becomes too thick to fold any more. Modelling this phenomenon with any of the current systems is impossible. Modelling it with Fournier's suggestion is possible, but difficult. As more layers are added, more complex relationships between the faces involved in the layering are formed. Also the number and configuration of narrow strips needed to render the curvature at the fold line becomes more difficult to determine. The diameter of curvature would grow as layers are added. Simulating thickness of paper in this way is intuitively a good method, but its practical application will require more thought.

Figure 75. Two-dimensional plan view of paper folded in half three times.

6.3.2 Animation Using Constrained Geometry

In many complex folds, including reverse folds, it is often possible to maintain the planarity of all the faces during the fold. However, doing so will unavoidably force other faces to move if the planarity constraint is to be maintained. Also, these affected faces will usually end up in a predictable final location after the fold. The current origami data structure could therefore be used to animate the complete folding operation if a set of constraints on the model could be set up and maintained.

The animations shown in "Paper Animal Workshop" (see Kittyhawk, 1995) are a good example of achieving complex folds using only planar faces. Their modelling and animation methods are not available for inspection, however. It is assumed that they have simply hard-coded the locations of all vertices through all frames of their animations.

Baraff and Witkin describe a differential method of maintaining constraints that may be applied directly to the existing origami model (Baraff and Witkin, 1998). During a

discussion with Alain Fournier, he suggested that there might exist a simple set of constraints on an origami model (Fournier, 1998b). This would include maintaining all edge lengths (which should remain constant), maintaining internal angles within each face (also constant), and then updating the dihedral angle between each pair of adjacent faces. If one dihedral angle is adjusted, for example when the paper is being folded, then the rest of the dihedral angles are probably constrained by the internal angles around the vertices. Preliminary work suggests that these constraints are not sufficient to yield a unique result without ambiguities. However, the idea has not yet been explored fully, and its successful implementation would make an excellent addition to the current system.

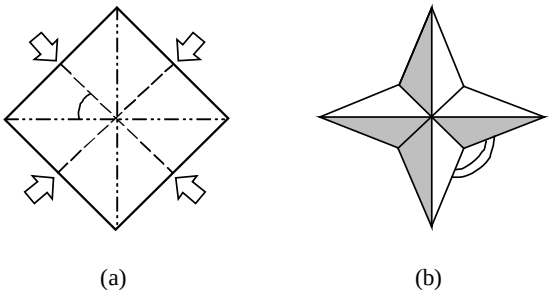


Figure 76. An internal angles around the central vertex is marked with a single line in (a). A dihedral angle is marked with a double line in (b).

6.3.3 Animation Using Curved Surfaces

As can be seen in any real origami sculpture, curved surfaces are essential in a good piece of modern origami. Especially in recent designs, origami has become very

sculptural, requiring the surface of the paper to have a curved shape in the final model. Moreover, almost all processes require that parts of the paper be curved or even forced into strange configurations[5] during folding, though in the final configuration the fold may lie flat. A method must be found to simulate these processes in the virtual origami model.

The current origami system is not capable of handling curved surfaces. Miyazaki et al. present their method to fold a sheet with curved surfaces, using a physically-based model (Miyazaki and others). They divide the curved portion of the paper evenly into long strips parallel to the fold, and use these strips to approximate the curve. This method works well for curving of a single layer of paper, but they do not extend it to handle two curved faces that meet, for example. They demonstrate that their curved surface implementation works correctly when adding details during the final stages of folding an origami model. However, the method is very limited and has no application to the curved surfaces that arise during folding.

It is very difficult to produce intermediate frames for the animation of folds where curved surfaces appear. These surfaces are non-planar, but they are ruled surfaces, meaning that the curvature must be zero in at least one direction. The surface function itself contains discontinuities as a fold in the paper is turned inside out, wrapped around itself, or otherwise forced into place. For an example, try folding an

[5] Forcing and inadvertently crunching parts of the paper into place is fondly known as "crumple-gami".

open or closed sink (see page 20) with the raw edges at the bottom of the model held together tightly. This can be difficult to do, even with real paper. Accurately modelling these manoeuvres would require a very detailed physical model. The model would have to maintain the volume of the paper during the fold while allowing tiny facets to pass arbitrarily close to one another without passing through one another. Small creases appear and disappear as the fold is completed. Producing a model that does this in a geometrically correct and visually satisfying way would make a good research project in itself.

Chapter 7

CONCLUSION

Modelling folded paper using the computer is a much harder problem than might first be assumed. Any origami model made out of real paper appears to be comprised of merely some polygonal facets, and a few curved parts. However, translating this seemingly simple object into a mathematical, computationally-tractable, accurate, manipulable and consistent computer model is quite difficult. If we simplify the problem too much, by ignoring curved surfaces and thickness of the paper, too many essential details are lost.

This thesis has attempted to give a detailed and interesting history of the development and study of traditional and technical origami. The origami interface design presented here is an example of how virtual reality and computer modelling can be used to represent, store and teach origami models. Although the current implementation is not complete, the remaining problems have been identified and solutions suggested. If we augment the design and implementation with the elements discussed in §6.3, Future Work, a very powerful virtual origami system seems possible.

REFERENCES

-. "Ask Dr. Math." Internet. Web page http://forum.swarthmore.edu/dr.math/dr-math.html; accessed 28 September, 1998.

-. "Paper animal workshop." CD-ROM. Kittyhawk Software Inc., 1995.

Agui, Takeshi, M. Takeda, M. Nakajima. "Note: Animating planar folds by computer." *Computer Vision, Graphics and Image Processing* 24 (1983): 244-254.

Allan, J.B., B. Wyvill and I.H. Witten. "A methodology for direct manipulation of polygon meshes." *New advances in computer graphics* (June 1989): 451-469.

Auckly, David and John Cleveland. "Totally real origami and impossible paper folding." *The American mathematical monthly: the official journal of the Mathematical Association of America* 102, no. 3 (March 1995): 215-226.

Baraff, David and Andrew Witkin. "Physically-based modelling: principles and practice." SIGGRAPH 97 course notes, course 19 (1997).

Baraff, David and Andrew Witkin. "Physically based modelling." SIGGRAPH 98 course notes, course 13 (1998).

Baraff, David and Andrew Witkin. "Large steps in cloth simulation." *Computer Graphics* Proceedings, Annual Conference Series (1998) 43-54.

Bern, Marshall and Barry Hayes. "The complexity of flat origami." *Proceedings of the ACM SIAM Symposium on Discrete Algorithms* (1996): 175-183.

Bill, James R. and Suresh K. Lodha. "Sculpting polygonal models using virtual tools." *Graphics Interface '95* (1995): 272-279.

Breen, David E., D. H. House and M. J. Wozny. "A particle-based model for simulating the draping behaviour of woven cloth." *Textile Research Journal* 64, No. 11 (1994): 663-685.

British Origami Society. *Diagram-free zone*. (Phone folding). Internet web document. http://www.rpmrecords.co.uk/bos/phone.html; accessed 18 September 1998.

Cooper, Alan. *About Face: The Essentials of User Interface Design*. Foster City, CA: IDG Books Worldwide, 1995.

Descartes, Alligator. *Magician Programmer's Guide for Java*. Internet. Available from http://www.arcana.co.uk/products/magician/docs/index.html; accessed 21 September, 1998.

Eberhardt, Bernhard, A. Weber and W. Strasser. "A fast, flexible, particle-system model for draping cloth." *IEEE Computer Graphics and Applications* (Sept. 1996): 52-59.

Eischen, Jeffery W., S. Deng and T. G. Clapp. "Finite-element modelling and control of flexible fabric parts." *IEEE Computer Graphics and Applications* (Sept. 1996): 71-81.

Eisenberg, Mike and Ann Nishoika. "HyperGami: a cad system for paper sculpture." Internet. Web page http://www.cs.colorado.edu/~eisenbea/hypergami/; accessed 26 September, 1998.

Engel, Peter. "A paper folder's finding." *The Sciences* 24 (May/June 1984): 16-20.

Engel, Peter. "Origami: the mathematician's art." *Discover* 9 (June 1988): 54-61.

Engel, Peter. *Origami from Angelfish to Zen.* New York: Dover Publications, 1994.

Ewins, Brian. "Proof of a conjecture of technical origami". Internet. Available from ftp://ftp.rug.nl/origami/articles; accessed 18 September, 1998.

Fehr, Howard. "On teaching dihedral angle and steradian." *Mathematics Teacher* 51 (1958): 272-275.

Fisher, David. "Origami on Computer." Internet. Available from ftp://ftp.rug.nl/origami/articles; accessed 18 September, 1998.

Flanagan, David. *Java in a Nutshell.* 2nd ed. Sebastopol, CA: O'Reilly and Associates, Inc., 1997.

Flanagan, David. *Java Examples in a Nutshell.* Sebastopol, CA: O'Reilly and Associates, Inc., 1997.

Fournier, Alain. A method to render folded paper that simulates thickness. Personal communication. August 1998.

Fournier, Alain. Suggestions for setting and maintaining geometric constraints in folded paper. Personal communication. September 1998.

Fuse, Tomoko. *Unit Origami. New York:* Japan Publications, 1990.

Fuse, Tomoko. *Joyful Origami Boxes.* Tokyo: Japan Publications Trading Co., 1997.

Fuse, Tomoko. *Fabulous Origami Boxes.* Tokyo: Japan Publications Trading Co., 1998.

Gardner, Martin. "Mathematical Games." *Scientific American* 201 (1959): 138-143.

Gardner, Martin. *The 2nd Scientific American Book of Mathematical Puzzles and Diversions.* New York: Simon and Schuster, 1961.

Geretschlager, Robert. "Euclidean constructions and the geometry of origami." *Mathematics Magazine* 68, no. 5 (1995): 357.

Glassner, Andrew. "Maintaining winged-edge models." In *Graphics Gems II*, ed. James Arvo. Boston: Academic Press, 1991.

Glassner, Andrew. "Origami platonic solids." *IEEE Computer Graphics and Applications* 16, no. 4 (1996): 85-91.

Glassner, Andrew. "More origami solids." *IEEE Computer Graphics and Applications* 16, no. 5 (1996): 81-85.

Gleicher, Michael and Andrew Witkin. "Drawing with constraints." *The Visual Computer* 11 (1994): 39-51.

Gurkewitz, Rona and Bennett Arnstein. *3-D Geometric Origami: Modular Polyhedra.* New York: Dover Publications, 1995.

Harbin, Robert. *Origami: the art of paper-folding.* Norwich: The English Universities Press Ltd., 1970.

Harbin, Robert. *Origami 2: the art of paper-folding.* Kent: Coronet Books, 1978.

Hayes, Brian. "Pleasures of plication." *American Scientist* 83 (Nov./Dec. 1995): 504-509.

Huffman, A. "Curvatures and creases: a primer on paper." *IEEE Transactions on Computers* C-25 (1976): 1010-1019.

Hull, Thomas. "On the mathematics of flat origamis." *Congressus Numerantium* 100 (1994): 215-224.

Hull, Thomas. "A note on "impossible" paper folding." *The American mathematical monthly: the official journal of the Mathematical Association of America* 103, no. 3 (March 1996): 240-241.

Hull, Thomas. "Planar graphs and modular origami." Internet. Available from ftp://ftp.rug.nl/origami/articles; accessed 18 September, 1998.

Hull, Thomas. "Origami and geometric constructions." Internet. Web page http://chasm.merrimack.edu/~thull/geoconst.html; accessed 18 September, 1998.

Hull, Thomas. "Origami mathematics." Internet. Web page http://chasm.merrimack.edu/~thull/OrigamiMath.html; accessed 28 September, 1998.

Huzita, Humiaki. "Understanding geometry through origami axioms." *Proceedings of the First International Conference on Origami in Education and Therapy (COET91)*, J. Smith ed. British Origami Society (1992): 37-70.

Kanade, T. "A theory of Origami world." *Artificial Intelligence* 13 (1980): 279-311.

Kasahara, Kunihiko and Toshie Takahama. *Origami for the Connoisseur.* New York: Japan Publications, 1987.

Kasahara, Kunihiko. *Creative Origami.* New York: Japan Publications, 1967.

Kasahara, Kunihiko. *Origami Omnibus.* New York: Japan Publications, 1988.

Kergosien, Yannick L., H. Gotoda, T. L. Kunii. "Bending and creasing virtual paper." *IEEE Computer Graphics and Applications* 14, no. 1 (1994): 40-48.

Lang, Robert J. "Albert joins the fold." *New Scientist* 23 (1988): 38-47.

Lang, Robert J. *The Complete Book of Origami.* New York: Dover Publications, 1988.

Lang, Robert J. "Origami: complexity increasing." *Engineering & Science* 52, no. 2 (Winter 1989): 16-23.

Lang, Robert J. *Origami Zoo.* New York: Dover Publications, 1990.

Lang, Robert J. "Mathematical algorithms for origami design." *Symmetry: Culture and Science* 5, no. 2 (1994): 115-152.

Lang, Robert J. *Origami Insects and Their Kin.* New York: Dover Publications, 1995.

Lang, Robert J. "A computational algorithm for origami design." *Proceedings of ACM Symposium on Computational Geometry* (1996): 98-105.

Lepowsky, William L. "The total angular deficiency of polyhedra." *Mathematics Teacher* 66 (1973): 748-752.

Maekawa, Jun and Kunihiko Kasahara. *Viva Origami!* Tokyo: Sanrio Publications, 1983.

Maekawa, Jun. "Evolution of origami models." *Symmetry: Culture and Science* 3, no. 2 (1992): 160-161.

Miura, Koryo. "Folds - the basis of origami." *Symmetry: Culture and Science* 5, no. 1 (1994): 13-22.

Miyazaki, Shin-Ya, T. Yasuda, S. Yokoi and J-I. Toriwaki. "An origami playing simulator in the virtual space." *The Journal of Visualization and Computer Animation* 7 (1996): 25-42.

Montroll, John and Robert J. Lang. *Origami Sea Life.* New York: Dover Publications, 1990.

Neider, Jackie, T. Davis and M. Woo. *OpenGL Programming Guide, The Official Guide to Learning OpenGL, Release 1.* Don Mills, Ontario: Addison-Wesley Publishing Company, 1993.

Nishoika, Ann and Mike Eisenberg. "HyperGami crafts." Internet. Web page http://www.hypergami.com/; accessed 26 September, 1998.

Ng, Hing N. and Richard L. Grimsdale. "Computer graphics techniques for modelling cloth." *IEEE Computer Graphics and Applications* (Sept. 1996): 28-41.

OpenGL Architecture Review Board. *OpenGL Reference Manual, The Official Reference Document for OpenGL, Release 1.* Don Mills, Ontario: Addison-Wesley Publishing Company, 1992.

Parodi, Pietro. "The complexity of understanding line drawings of Origami scenes." *International Journal of Computer Vision* 18, no. 2 (May 1996): 139-170.

Peterson, Ivars. "Paper folds, creases and theorems." *Science News* 147, no. 3 (Jan. 1995): 44.

Randlett, Samuel. *The Art of Origami.* New York: E. P. Dutton, 1966.

Row, Sundara. *Geometric Exercises in Paper Folding*. New York: Dover Publications, 1966.

Saupe, Ethel. "A paper model for solid geometry." *Mathematics Teacher* 49 (1956): 185-186.

Scher, Daniel P. "Folded paper, dynamic geometry, and proof: A three-tier approach to the conics." *Mathematics Teacher* 89 (1996): 188-193.

Smith, John. "An origami instruction language." *British Origami Society Booklet* 4 (Dec. 1975).

Struyk, Adrian. "Three folding models of polyhedra." *Mathematics Teacher* 49 (1956): 286-288.

Turk, Greg. "Re-tiling polygonal surfaces." *Computer Graphics (Proceedings of SIGGRAPH '92)* 26, No. 2. (July 1992): 55-64.

Watt, Alan and Mark Watt. *Advanced Animation and Rendering Techniques: Theory and Practice*. New York: Addison-Wesley Publishing Company, 1992.

Wu, Joseph. "Joseph Wu's origami page." Internet. Web page http://www.origami.vancouver.bc.ca/; accessed 27 September, 1998.

Yoshizawa, Akira. *Origami Masterpieces*. Torrance: Heian International Publishing Inc., May 1997.

Yoshizawa, Akira. *Origami Museum I: Animals*. New York: Japan Publications (U.S.A) Inc., January 1987.

Appendix A

JAVA PROGRAMMING

A.1 Java

Java is a very new programming language that seems to be quickly developing a large group of users. Java is described by Sun as:

> "A simple, object-oriented, distributed, interpreted, robust, secure, architecture neutral, portable, high-performance, multithreaded and dynamic language."

Learning to program in Java while writing the origami system was interesting. The object-oriented nature of the language, as well as the elegance of various language constructs made it worthwhile. Among other advantages, Java does not allow pointer manipulation, and invisibly takes care of releasing dynamically-allocated memory when it is no longer being used.

A drawback of the Java environment is that it runs slowly. For example, if the window containing the origami interface is resized, you can watch the buttons being redrawn

on the toolbars. However, the OpenGL 3D display does not flicker when updated or accessed, and so the most important element of the origami system works well.

This implementation was done using Sun's JDK 1.1.5. The latest version of Java is available free on the Internet from http://java.sun.com/ or http://www.javasoft.com/. The API documentation and other useful documents are also available from these sites.

A.2 Java Abstract Windowing Toolkit (AWT)

The Abstract Windowing Toolkit (AWT) is the part of the Java API that defines all graphical user interface components. These include buttons, windowing, labels, frames, text boxes, window arrangements and layouts, and event generation and handling. The Java AWT was used in conjunction with Swing (see §A.3) and Magician (see §A.5) to create the origami system interface.

A.3 Java Foundation Classes (Swing)

The Java Foundation Classes, also known as Swing, provide a set of ready-to-use components that extend the standard Java AWT. The Swing components used in this implementation include toolbars, buttons with images on them, tooltips, rollover icons, and panels. Other available Swing components include scrollbars, sliders, tables, combo boxes and tabbed panes.

This implementation was done using Swing version Beta 0.7. The latest version of Swing and its documentation can be downloaded free from http://java.sun.com/products/jfc/.

A.4 OpenGL

OpenGL is a software interface for graphics hardware. The functions it provide to programmers include three-dimensional geometric modelling, controlling viewpoints and perspective projections, applying colour and texture to models, lighting scenes and manipulating pixels. OpenGL was used in this project to create the 3D and 2D displays in the graphical interface.

The latest version of OpenGL is available for free download on the web from http://www.opengl.org/. Some other resources for developers can be found at http://trant.sgi.com/opengl/.

A.5 Magician OpenGL Interface

The Magician interface is an implementation of OpenGL for Java. Magician integrates functionality from both Java and OpenGL to provide powerful rendering over a variety of platforms, using existing OpenGL libraries. Magician preserves the portability of both Java and OpenGL.

Java does not have built-in support for three-dimensional drawing, and as of this writing, a reliable version of Java3D is not available. Thus Magician was chosen to

implement the 2D and 3D modelling and drawing functions required by the origami interface. The virtual trackball functionality of the origami system was implemented entirely thanks to the trackball utility provided by Magician.

The latest version of Magician and its documentation can be found on the web at http://www.arcana.co.uk/products/magician/. It is currently available for free download for personal and non-commercial use, or for a nominal licensing fee for other uses.

INDEX

A

animated diagrams, 54
animation, 43, 76

B

base, *21*
bases
  bird, 23
  fish, 23
  frog, 23
  kite, 23
  preliminary fold, *15*
  waterbomb, *14*
bird base, 23
blintzed bird base, 25
blintzing, *25*
box pleating, *24*

C

capped polyhedron, 38
closed sink, *20*, 74
cloth modelling, 49
complex folds, *72*, 100
computational geometry, 51
crease, *21*
crease pattern, *22*, 30, 66, 84
crimp, *17*
crumple-gami, *120*
cut-away view, *12*

D

data structure, 85
design methods, 23
detail folds, *21*
diagram, 5
differential equations, 49
double-click, 65
drawing with constraints, *50*

E

edge, *6*
edge configurations, *9*
equal angles, *11*
equal distances, *10*

F

fish base, 23
flap, *6*
flat foldable, *34*
fold
  closed sink, 20
  crimp, 17
  fold and unfold, 7
  inside reverse, 16
  mountain, 7
  open sink, 19
  outside reverse, 17
  petal, 18, 19
  rabbit-ear, 15
  squash, 18
  valley, 6
  waterbomb, 14
fold and unfold, *7*
fold over and over, *11*
folding net, *44*
formal language, 2
Four Colour Theorem, 39
frog base, 23, 25, 26

G

grafting, *25*, *26*
graph theory, 51

H

hold here and pull, *14*
HyperGami, 44

I

inside reverse, *16*, 17
interface, 53

J

Java Foundation Classes, 130
Java3D, 108, 131

K

kirigami, *114*
kite base, 23

L

line drawings, 50

M

Magician, 108, 130, 131
main toolbar, 67
META key, *64*
modular origami, 28, 38, 114
mountain fold, *7*, 72
moving face, *98*
moving vertex, *98*

N

natural language, 2
next view larger, *13*
non-planar surfaces, 42

O

object-oriented, 129
one-button mouse, 63
open sink, *19*, 74
OpenGL, 108, 130, 131
origami, ii, *5*
origami geometry, 34
origami line graph, *35*
origami sekkei, *26*
Origami world, 50
outside reverse, *17*

P

paper folding, 31
petal fold, *18*, *19*, 73
phone folding, 2
physically-based, 49, 51
planar graph, 38
planar graph theory, 38
polygonal modelling, 46
polyhedra, 44
pre-creases, 24
pre-creasing, *3*, *71*, 73
preliminary fold, *15*
pull paper out, *12*
push here, *8*

R

rabbit-ear, *15*, 74
raw edges, *6*
regular polygon, 31
repeat, *8*
reverse fold, 73, 74, 100
rollover icon, *66*, 76
rotate, *10*

S

simple fold, *71*
simple folds, 100
squash, *18*, 73
squash fold, 102
status bar, 68, *82*
stellated polyhedron, 38
Swing, 130
symbol
  cut-away view, 12
  edge configurations, 9
  equal angles, 11
  equal distances, 10
  fold and unfold, 7
  fold over and over, 11
  hold here and pull, 14
  mountain, 7
  next view larger, 13
  preliminary fold, 15
  pull paper out, 12
  push here, 8
  repeat, 8
  rotate, 10
  turn the paper over, 12
  valley, 6
  watch this spot, 9
  x-ray line, 13

T

technical folding, *26*
thickness, 28, 30
three-button mouse, 64
toolbar, *66*
tooltip, *84*
traditional diagrams, 54
Treemaker, 41
trisecting an angle, 11
turn the paper over, *12*
two-button mouse, 64

U

unit, 28
user interface, 103, 108, 130

V

valley fold, *6*, 72
virtual trackball, 76

W

watch this spot, *9*
waterbomb base, *14*, 15

X

x-ray line, *13*

Y

Yoshizawa, Akira, 5, 24