An investigation of the usability of software

for producing origami instructions

A dissertation submitted in partial fulfilment of the requirements for

the

Open University's Master of Science Degree

in

Computing for Commerce and Industry

by

Tung Ken Lam

R4879389

13 September 2005

Word Count: 15,713

Preface

I would like to acknowledge the help and advice from my supervisor, Tony Roberts. I would also like to thank anyone who has helped make this project possible, either directly or indirectly. I especially wish to thank Sue Pope for her unstinting enthusiasm and encouragement throughout this project.

PostScript is a registered trademark of Adobe Systems Incorporated.

Windows is a registered trademark of Microsoft Corporation.

Contents

List of tables

List of figures

Abstract

The task of making origami diagrams is both difficult and time-consuming. It is now common for diagrammers to use computers but few authors use programs that are specifically written for origami diagramming.

This study evaluates different types of software for making origami diagrams. It draws on the field of Human-Computer Interaction (namely the Cognitive Dimensions framework) to determine the usability strengths and weaknesses of existing software. The study also seeks the opinions of both readers and authors of origami diagrams via a questionnaire, partly based on Lang's ten principles for diagramming and QUIS (Questionnaire for User Interaction Satisfaction.)

These usability findings inform the design of an improved interface for making origami diagrams. A prototype interface based on S. Miyazaki's origami simulation is evaluated by the author and by nine participants in a usability study. This evaluation shows that whilst the subjects rated the prototype as "wonderful", they criticised it for its inadequate power and being rigid. Respondents were mixed in their feelings of satisfaction, the prototype's ease of use and the visual attractiveness of its diagrams.

The main improvement that the prototype needs is to extend its repertoire of fold types. It should be relatively easy to implement *outside reverse folds*, but other folds such as *rabbit's ears* may cause problems in both specifying the user interface and the implementation.

1 Introduction

This chapter introduces the task of making origami diagrams. It then follows with a review of the existing literature on computers and origami, none of which specifically focuses on the usability of software for making origami diagrams. It concludes with an outline of the objectives of this project and describes the structure of this report.

1.1 Origami Diagrams

Origami is the "Japanese art of paperfolding...If you can think of an object either natural or man-made, someone, somewhere, has probably folded an origami version." (Lang, 2003a., p. 1-2).

Origami designs have traditionally been passed on by people teaching a folding method to other people. The earliest known written origami instructions date from the early 18th century in Japan. (Lister, 2003). The instructions combine images and text into a sequence that takes the reader through the steps needed to fold a figure from paper. Figure 1 shows a Japanese example from the mid-nineteenth century.

Kanomado, c. 1850
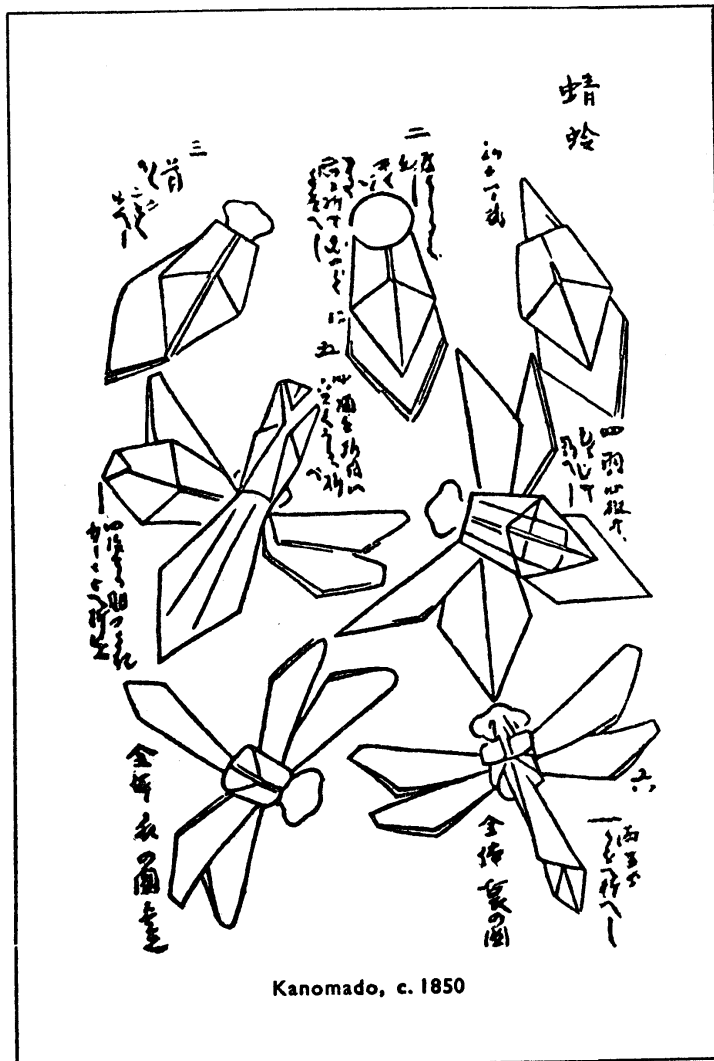
Figure 1 Mid-nineteenth century Japanese diagrams for folding a Dragonfly (Harbin, 1956, p. 8)

Later, authors of books

on paperfolding struggled in different ways to draw diagrams illustrative of their text with varying degrees of success, but they were not, in fact, very successful at all. Some supplemented their outline step diagrams with perspective drawings. Some added letters at the corners. A few used photographs." (Lister, n.d.).

Akira Yoshizawa revolutionised origami instructions by "adopting different dotted lines for mountain and valley folds and by using arrows to show the moves in the paper, he at once transformed static diagrams into dynamic pictures" (Lister, n.d.). Yoshizawa pioneered his

system in the 1950s and it was soon adopted and adapted by Samuel Randlett and Robert

Harbin. Figure 2 shows Harbin's version of the standard and Figure 3 shows the standard in

use. As Koshak (2003, p. 7) notes, "The vast majority of all origami books and publications

use [Yoshizawa-Harbin-Randlett] diagramming and most origami practitioners can read and

understand diagrams." (In fact, most readers can understand diagrams published in a

language that the reader does not understand. Figure 40, p. 209, shows a newer set of symbols

in both Japanese and English)

THE SYMBOLS – These should be memorised now

mountain fold

valley fold

cut   or   cut

creases ————— ————   X-ray view ·····················

hold here   watch this spot X or Y

fold in front   fold under or into

fold behind   push in sink or squash

open out   pull out flap from underneath

turn over

fold over and over

repeat a fold one or more times   once   3 times   fold and unfold to crease only
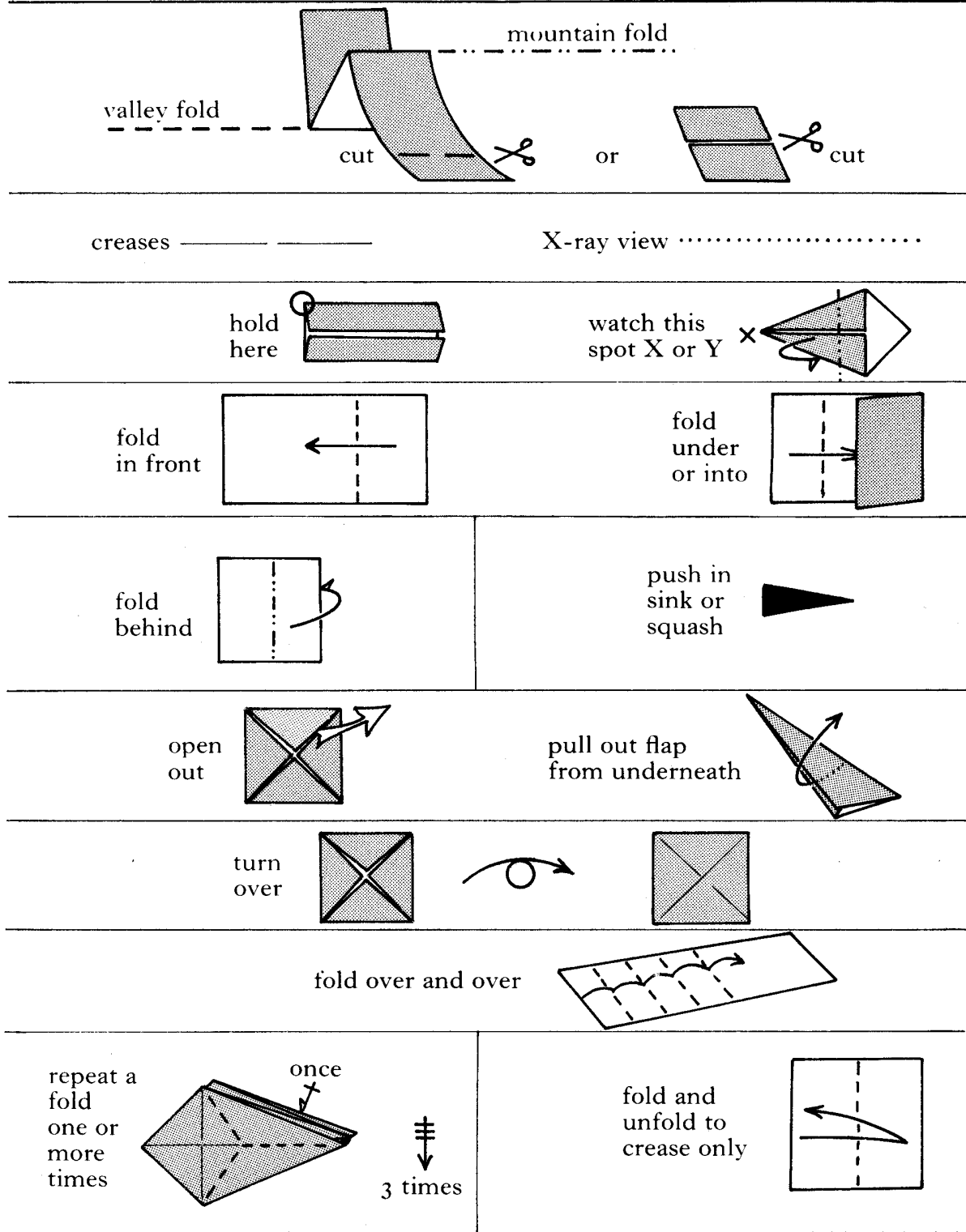
Figure 2 Standard folding symbols (Harbin, 1974, p. 8)

Lover's Knot – Traditional *An exercise*

crease diagonals
first

this is a BLINTZ
FOLD

Precrease

open out
4 flaps

push middle
of SALT CELLAR
in–

– so,
press flat

– so

insert thumbs
either side
of X

grip firmly
open flap

– so

– and so
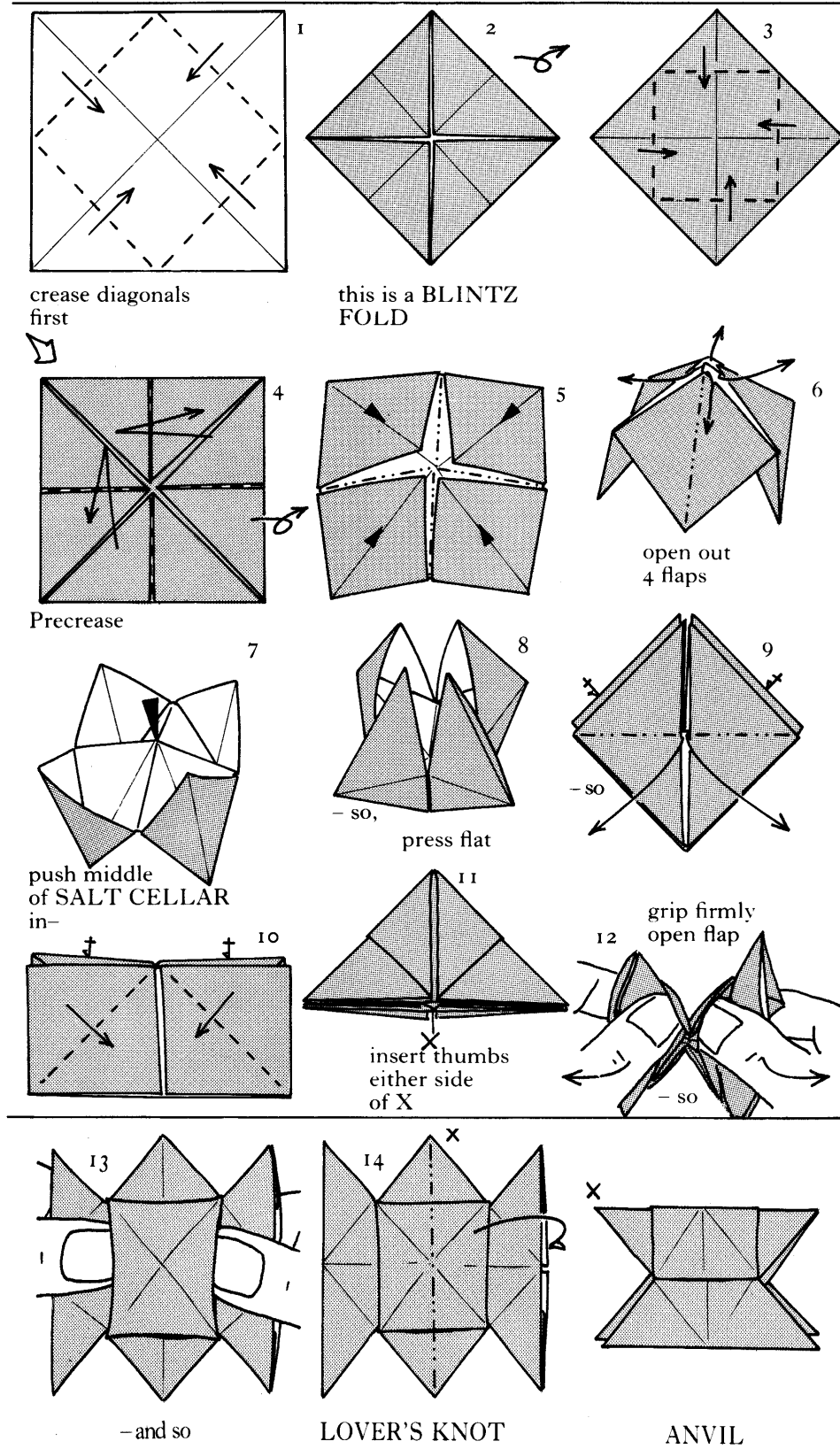
LOVER'S KNOT

ANVIL

Figure 3 Modern diagrams for folding a traditional salt cellar, lover's knot and anvil (Harbin, 1974, p. 15-16)

One special characteristic of origami diagrams is that they need to distort reality in order to clearly show the reader how the paper is arranged. (Petty, n.d; Szinger, 2001, 2002; Robinson, 2004). In step 2 of Figure 3, notice how the corners of the square do not meet at the centre, and that there is a thin gap between the original edges of the square. The instruction in step 1 *is* to fold the corners exactly to the centre, but the result is partially opened up so that the reader can perceive the three dimensionality of the drawing.

1.2 Making Origami Diagrams without a computer

Diagramming has been described as "boring, tedious work" (Lang, 1989a, p.16). Many origami designers prefer to design models rather than document them. There are several reasons why diagramming is considered difficult and time-consuming (Cunliffe, 1988, 1989a, 1989b, 1989c):

- There is a need to work out a folding sequence that is
    - enjoyable and understandable by others
    - can be drawn relatively easily.
- Some perceive a lack of their own artistic ability.
- It is difficult to change once substantial work has started on a set of diagrams.
- The work of drawing similar step folds is repetitive (steps are usually only slightly different from each other).
- Drawing flat representations of three-dimensional objects can be challenging.

1.3 Making Origami Diagrams with Computers

Origami instructions have been drawn by hand for many years. Traditional diagramming is time-consuming and hence many authors have sought to use computer power to speed up the

process (e.g. Gout, 2001; Lang, 1989a, 1989b; Petty, n.d). Another reason is that computerised diagrams are convenient to distribute electronically (Petty, n.d).

For these reasons a number of approaches have been used

- Use CAD programs (Glassner, 1996)

- Use programs to construct 3-dimensional models e.g. Mathematica has been used in this way (Hull, 1995)

- Use text languages, e.g.

  - Oridraw (van Gelder, 2002)

  - Doodle (Gout, 2001)

  - Fisher (1994)

- Use computer drawing packages intended for artists and illustrators, e.g.

  - CorelDRAW!

  - Adobe Illustrator

  - Macromedia Freehand (Petty, n.d; Lang, 1996)

- Or general graphics module in general purpose software e.g.

  - the drawing module in Ami Pro, a word processor (Petty, n.d.)

- Or graphics programs intended for specific niches e.g.

  - Visio, which was originally designed for producing flow charts, organisation charts, etc.

- Simulate folding using virtual paper (Szinger, 2001, 2002)

There are two methods that are often used on a computer, but can be done without using a computer:

- Use mathematical/textual descriptions e.g. OIL, Origami Instruction Language
  (Smith, 1975)

- Use partial, abbreviated diagrams e.g. a "crease pattern" shows the fold lines when a
  folded model is unfolded back to the original piece of paper (Koshak, 2003). A
  relatively uncommon extension of this is the annotated crease pattern: the reader
  prints out and folds the paper containing the instructions (Nordal, 2001; Ward, 1976).

One approach not widely discussed is the use of mathematical drawing programs (e.g.
Geometers' Sketch Pad, FXDraw, Cabri or Cabri3D).

1.4 Literature Review of Origami and Computers

The approaches listed in the previous section focus on using existing software, or developing
new software, to *document folding methods*. Other researchers have been using *origami as a
context* for novel research in other areas. These researchers are not specifically interested in
improving methods for documenting origami (or their qualities) *per se*, but do touch on it in
the course of their main research:

- Simulation of folding and Virtual Reality (Miyazaki *et al.*, 1996)

- Constraint functional logic programming (Ida *et al.*, 2003)

- Mathematical analysis and proof e.g. Lavoie (n.d).; Ida *et al.* (2004); Ida and Buchberger
  (2004)

- Simulation for "studying geometric and graphic primitives in a picturesque and eye-
  catching context... [and] appreciating both creative and educational sides of Origami"
  (Zamiatina, 1994)

- Usability of computer representations for teaching and learning origami (Kishi and Fujii, 1998; Leventhal, 2001; Zimmerman *et al,* 2003; Ilsley, 2003)

- Automatic diagram capture (Kato *et al.*, 1997; Kato *et al.*, 1998; Shimanuki *et al.,* 2003; Suzuki *et al.*, 2002) and scene capture (Wilkes and Tsotsos, 1992)

- Use of novel multi-modal technologies for teaching origami (Ju *et al.*, 2002)

- Teaching origami (Ariel, 1998, reports how Kittyhawk Software, Inc., commissioned bespoke software for modelling and animating origami. It seems that the software cannot simulate folding – the user is tasked with "plugging in coordinates for the different facets and then telling how to rotate the facet(s) etc...")

- Usability of different types of origami instructions e.g. Novick and Morse (2000) investigated the effectiveness of three types of instructions: text only; final diagram with text instructions; step-by-step with final diagram.

- Design of origami using computer tools (Lang, 2003a; Shimanuki *et al.,* 2004)

- Tools to assist origami design (Lang, 2003b; Bateman, 2005)

- Application of SVG (Scalable Vector Graphics) to a visualisation and modelling problem using CSS, SMIL animation and ECMAScript (Teng and Mansfield, 2003)

Many of the authors have created 3-dimesional animations of folding methods (e.g. Miyazaki et al., 1996; Ida et al., 2003; Zamiatina, 1994, etc.) Some have based this on the assumption that printed diagrams are difficult to follow (e.g. Leventhal, 2001; Shimanuki *et al.*, 2003).

1.4.1 Computer Origami Simulation

Some early work on origami simulation indicated the difficulty of both the implementation of an algorithm and the specification of the user interface (Lang, 1991). Lang (2005b) reported that by the early 1990s his simulator (Figure 4) allowed users to

[make] basic mountain and valley folds, turn the paper over, rotate it in the plane of the paper, and so forth: what is now called "Pureland Origami" (a name and concept coined by John Smith ... in the 1970s).

Several authors reported successful implementations (Miyazaki *et al.*, 1992; Miyazaki *et al.*, 1996; Zamiatina, 1994; Fisher, 1994). Two recent examples are Szinger, (2001, 2002) and Nimoy (2002). The latter based his work on source code provided by Miyazaki.
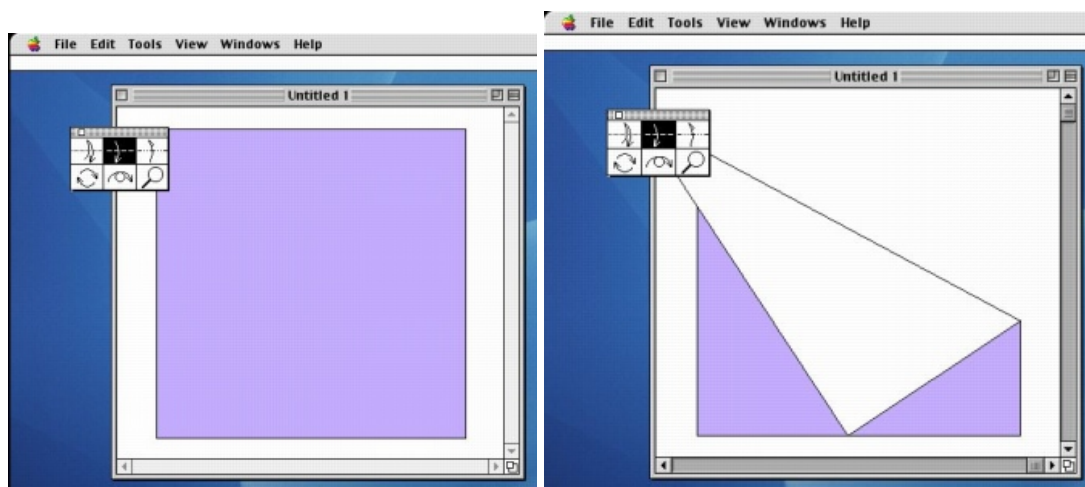
Figure 4 Lang's (2005b) Origami Simulation showing the result of dragging a corner of a square to the bottom edge

Of the work mentioned above, Ida (n.d.), Nimoy (2002), Miyazaki *et al.* (1996) and Zamiatina (1994) provided working versions of their simulations. Their work took the form of, respectively, an interactive version on a web site, a Java applet, C++ source code and Mathematica notebooks.

1.4.2 Origami-Oriented Software

Only Gout (2001), van Gelder (2002), Szinger (2001, 2002) and Nimoy (2002) have focused on documenting origami models using internationally accepted standards i.e. those based on systems by Yoshizawa, Randlett and Harbin. Szinger's Foldinator project (Figure 5) and

Nimoy's Java Origami (Figure 6) both use origami symbols to manipulate the on-screen

virtual paper i.e. they use fold lines and arrows to specify folds, rather than using the mouse

as an on-screen virtual "hand". The latter approach was taken by Miyazaki *et al.* (1996) and

Lang (1991). Gout's Doodle (Figure 7) and van Gelder's OriDraw are effectively graphics

programming languages with special features for origami diagrams.
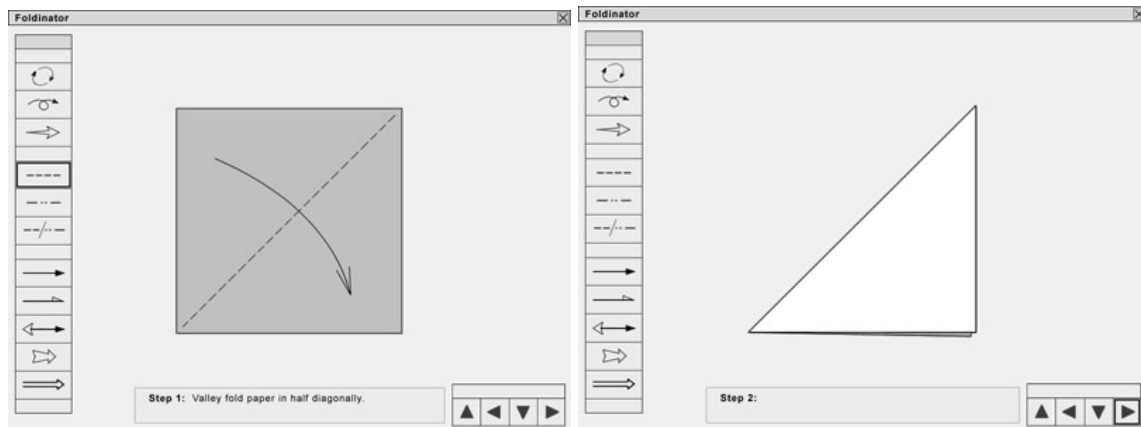
Figure 5 Foldinator screenshot showing use of symbols for defining folds (Szinger, 2001)
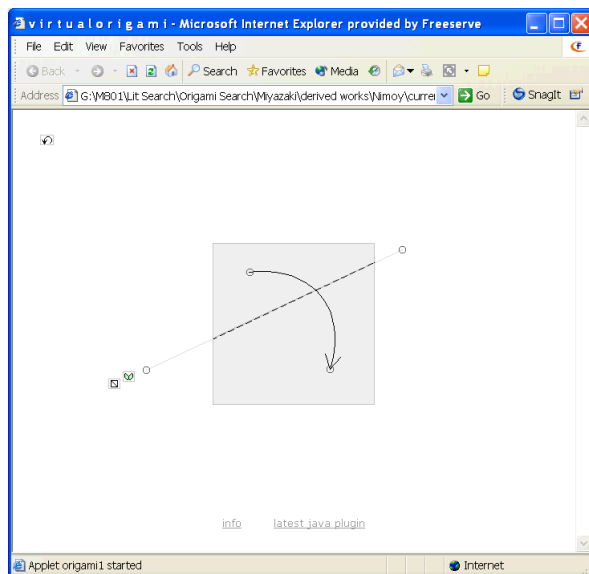
Figure 6 Nimoy's Java Origami showing use of fold line and arrow for defining a fold

Test

Design: TKL

Level : test
Paper : test

1- \square(a, b, c, d);

2- v1 = \line_to_line(b, a, d, [a, d]);

3- \cut([a, d], v1);
% fold cut the orignal edge

4- \move(a, [v1, b]);
% effective paper move

5- \border(v1, b);
% fold line becomes border

6- \fill(back, v1, b, a);
% we see now a part of the back side
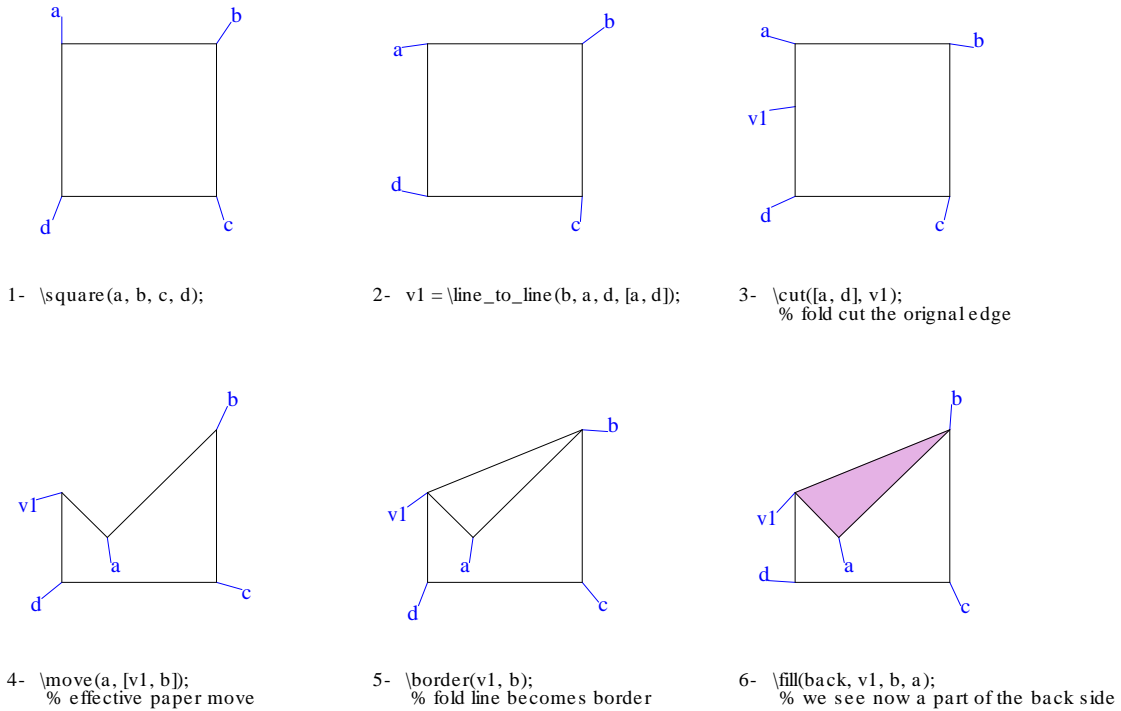
1

Figure 7 Doodle output for a test file illustrating selected Doodle commands

Doodle and OriDraw are unlikely to be used by many people due to the programming skills required. Command languages need "substantial training and memorisation" (Shneiderman and Plaisant, 2005, p. 72). They note, however, that command languages appeal to "power users".

Both Nimoy and Szinger both appear to be no longer actively developing their projects. The author of Foldinator acknowledges that its development is "proceeding at a snail's pace" given his current personal circumstances (Szinger, 2005)

Teng and Mansfield (2003) claim that they have developed a method to make "origami instructions ... more understandable for the user and easier for the author". Their method does allow the author to use high-level terms like the paper front and back, valley and mountain fold lines and arrows. Figure 8 shows the result of an SVG file (excerpt in Figure 9) and its style sheet, Figure 10. However, Teng and Mansfield do not present other methods for simplifying the task of authoring origami instructions. For example, the author must define coordinates of vertices to define polygons, and hence paper flaps. Changing the appearance of a style may be easy, but altering the position of flaps is still difficult.
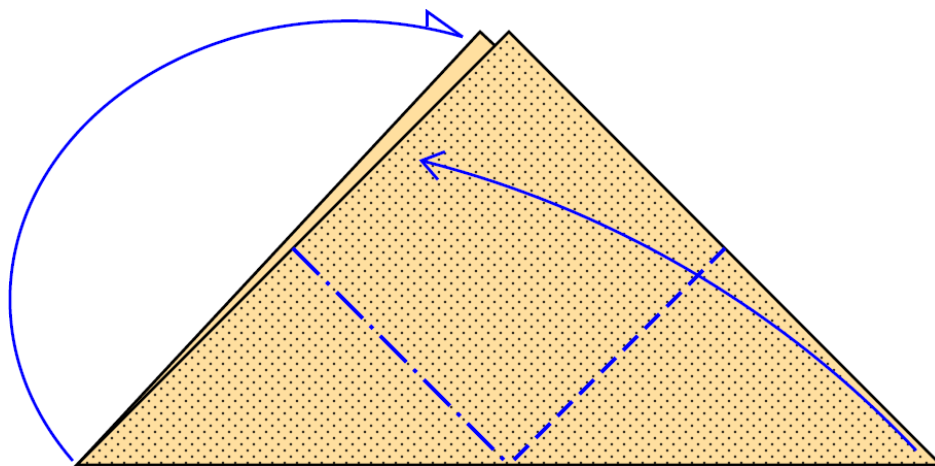
Figure 8 bunny02.svg (Teng and Mansfield, 2003) as displayed by Adobe SVG Viewer version 3.0

```
  <g class="paper">

    <polygon class="back" points="246,50 470,260 50,260"/>

    <polygon class="front" points="260,50 470,260 50,260"/>

  </g>

  <g class="fold">

    <line class="mountain" x1="155" y1="155" x2="260" y2="260"/>

    <line class="valley" x1="365" y1="155" x2="260" y2="260"/>

  </g>

  <g class="arrow">

    <path class="behind" d="M48,258 A165,135 0 0,1 237.5,52.5"/>

    <path class="forward" d="M458,253 A500,500 0 0,0 217.5,112.5"/>

  </g>
```

Figure 9 Excerpt of bunny02.svg (Teng and Mansfield, 2003)

```
.paper {fill:rgb(255,223,159); stroke-width:1.5; stroke:black}

.paper .front {fill:url(#stipple)}

pattern .paper {stroke-width:0; stroke:none}

.fold {fill:none; stroke-width:2; stroke:blue}

.fold .mountain {stroke-dasharray:18,5,2,5}

.fold .valley {stroke-dasharray:10,5}

.arrow {fill:none; stroke-width:1.5; stroke:blue}

.arrow.closed {fill:blue; stroke-width:0; stroke:none}

.arrow.thin {stroke-width:1.2}

.arrow .forward {marker-end:url(#pointer)}

.arrow .behind {marker-end:url(#halfTaper)}

.arrow .tuck {marker-end:url(#solidTaper)}

.arrow .repeat {marker-start:url(#doubleSlash);

    marker-mid:url(#loop); marker-end:url(#pointer)}

.arrow .flip {marker-mid:url(#loop); marker-end:url(#pointer)}

.arrow .blow {stroke-opacity:0; marker-end:url(#whoosh)}
```

Figure 10 origami.css (Teng and Mansfield, 2003)

1.5 Motivation

As mentioned in Section 1.2, it is commonly acknowledged that diagramming can be difficult and time-consuming. Lang (1989a) wrote that he "hates" diagramming because it is "boring, tedious work". Koshak (2003, p.8) claimed:

> The major disadvantage to diagramming is that generating them is a tedious, laborious and error prone process. ... Even though diagrams are the most common way to document origami, the labor involved keeps many model designers from documenting their models.

Even with the use of computers, diagrammers still find diagramming difficult and time-consuming. There seems to be little research on the *usability of software* for producing origami instructions.

1.6 Objectives

The main research question of this project is "How can the task of origami diagramming using a computer be improved using Human-Computer Interaction (HCI) theory, principles and methods?"

This project aims to investigate

   a) Which software is used?

   b) What qualities should "good diagrams" possess?

   c) How well is the software used?

   d) Which approaches are most fruitful?

   e) What other approaches could be used?

   f) How could such approaches be refined?

1.7 Report Structure

This chapter introduced some of the existing software and approaches used. It stated that Yoshizawa-Randlett-Harbin's diagramming system is a *de facto* standard, and hence "good diagrams" should use it. However, this system has evolved over time – individual diagrammers have their own interpretations of the standard. Some adopt, adapt or define new symbols for their own purposes. Therefore guidance on interpretation is needed and this is given by Lang (2000).

Chapter 3 describes a questionnaire to gather approaches and how well software is used. This is based on the HCI work of others described in Chapter 2. In addition to the questionnaire, I will evaluate the software myself. Chapter 4 contains this evaluation and describes the benchmark software and tasks in more detail.

The usability findings in Chapters 3 and 4 are the basis for proposing an improved interface in Chapter 5. Chapter 6 describes the implementation and Chapter 7 its evaluation.

Chapter 8 presents the conclusions of the project and suggests further work.

This work will be of interest to others who wish to find systematic research on the use of computers for origami diagramming. The work could be of use to authors in other specialist fields e.g. people who produce mathematical diagrams. The sequential nature of origami instructions is similar to that needed for certain kinds of illustrated instructions e.g. assembly and operating instructions for bicycles, doll's houses, electrical circuits, jigsaws, exercise machines, etc. (Novick and Morse, 2000)

2 Human-Computer Interaction

2.1 Introduction

This chapter outlines the field of Human-Computer Interaction (HCI). It then describes the most relevant aspects of HCI, namely the definition of usability and methods for evaluating it.

Different types of evaluation are described and the most suitable methods are chosen: expert evaluation using the Cognitive Dimensions framework and asking for users' opinions of usability via a questionnaire. Reasons for choosing the Cognitive Dimensions framework are given and the framework is compared with other principles and guidelines for design and usability.

2.2 HCI as a field

Carroll (2003, p1-9) states that HCI is a large, diverse and multi-disciplinary field – indeed, the success and strength of HCI is founded on this diversity. Over the past two decades, HCI has grown to take in fields as diverse as anthropology, sociology, computer science, cognitive science and Marxism. However, HCI as a discipline has fragmented and this means that it can be difficult for both researchers and practitioners to select appropriate theories and methods.

Shneiderman (2003, p. xv) argues that HCI has been "accepted in academic departments and corporate boardrooms." HCI could

become a basic science like physics and psychology; it could remain an eclectic

interdiscipline like biophysics and sociolinguistics; or it could mature into a

professional discipline with scientific foundations like architecture and medicine.

(Shneiderman, 2003, p. xv)

Thus many equally valid methods exist for improving the usability of software for a given task. The next section considers definitions of usability.

2.3 Usability

Preece *et al.* (2003, p. 14) suggest that usability has the following components:

- Effective to use (effectiveness)

- Efficient to use (efficiency)

- Safe to use (safety)

- Good utility (utility)

- Easy to learn (learnability)

- Easy to remember how to use (memorability)

Dix *et al.* (2004, p260-261) present three broad aspects of usability

- Learnability – the ease with which new users can begin effective interaction and achieve maximal performance

- Flexibility – the multiplicity of ways in which the user and system exchange information

- Robustness – the level of support provided to the user in determining successful achievement and assessment of goals

For each aspect, they present abstract principles affecting usability

- Learnability

    - predictability

- o synthesisability

- o familiarity

- o generalisability

- o consistency

- Flexibility

  - o dialog initiative

  - o multi-threading

  - o substitutivity

  - o customisability

- Robustness

  - o observability

  - o recoverability

  - o responsiveness

  - o task conformance

2.4 Evaluation

Dix *et al.* (2004, p319-320) identify three goals for evaluating a system:

1. To assess the extent and accessibility of the functionality: e.g. ease of use, meeting users' expectations, etc

2. To assess users' experience of interacting with a system: e.g. ease of learning, usability, satisfaction, etc.

3. To identify specific problems in order to address them at a later stage

This project needs to select an evaluation method that allows all three goals to be fulfilled.

Different types of evaluation are now considered.

2.5 Types of evaluation

Evaluation is a significant aspect of HCI. As there are so many individual evaluation methods, authors classify them into broad categories. For example, Dix *et al.* (2004, p360-362) use the following categories:

- Analytical
  - o cognitive walkthrough
  - o heuristic evaluation
  - o review based
  - o model based
- Experimental and query
  - o experiment
  - o interviews
  - o questionnaire
- Observational
  - o think aloud
  - o protocol analysis
  - o post-task walkthrough
- Monitoring
  - o eye tracking
  - o physiological measurement

Ivory and Hearst (2001) identified more evaluation methods (39 are listed in Table 1) and organised them into five broad categories:

Testing: an evaluator observes users interacting with an interface (i.e., completing tasks) to determine usability problems.

Inspection: an evaluator uses a set of criteria or heuristics to identify potential usability problems in an interface.

Inquiry: users provide feedback on an interface via interviews, surveys, and the like.

Analytical Modeling: an evaluator employs user and interface models to generate usability predictions.

Simulation: an evaluator employs user and interface models to mimic a user interacting with an interface and report the results of this interaction (e.g., simulated activities, errors, and other quantitative measures).

The authors state that the first three categories (testing, inspection, and inquiry) are "appropriate for formative (i.e., identifying specific usability problems) and summative (i.e., obtaining general assessments of usability) purposes." (*ibid.*, p. 473)

Method Class	Method Type	Description
Testing		
	Thinking-Aloud Protocol	user talks during test
	Question-Asking Protocol	tester asks user questions
	Shadowing Method	expert explains user actions to tester
	Coaching Method	user can ask an expert questions
	Teaching Method	expert user teaches novice user
	Codiscovery Learning	two users collaborate
	Performance Measurement	tester records usage data during test
	Log File Analysis	tester analyses usage data
	Retrospective Testing	tester reviews videotape with user
	Remote Testing	tester and user are not colocated during test

Inspection

Guideline Review	expert checks guideline conformance
Cognitive Walkthrough	expert simulates user's problem solving
Pluralistic Walkthrough	multiple people conduct cognitive walkthrough
Heuristic Evaluation	expert identifies violations of heuristics
Perspective-Based Inspection	expert conducts narrowly focused heuristic evaluation
Feature Inspection	expert evaluates product features
Formal Usability Inspection	expert conducts formal heuristic evaluation
Consistency Inspection	expert checks consistency across products
Standards Inspection	expert checks for standards compliance

Inquiry

Contextual Inquiry	interviewer questions users in their environment
Field Observation	interviewer observes system use in user's environment
Focus Groups	multiple users participate in a discussion session
Interviews	one user participates in a discussion session
Surveys	interviewer asks user specific questions
Questionnaires	user provides answers to specific questions
Self-Reporting Logs	user records UI (User Interface) operations
Screen Snapshots	user captures UI (User Interface) screens
User Feedback	user submits comments

Analytical Modeling

GOMS (Goals, Operators, Method and Selection) Analysis	predict execution and learning time
UIDE (User Interface Development Environment) Analysis	conduct GOMS analysis within a UIDE
Cognitive Task Analysis	predict usability problems
Task-Environment Analysis	assess mapping of user's goals into UI (User Interface) tasks

Knowledge Analysis	predict learnability
Design Analysis	assess design complexity
Programmable User Models	write program that acts like a user

Simulation

Information Processing Modeling	mimic user interaction
Petri Net Modeling	mimic user interaction from usage data
Genetic Algorithm Modeling	mimic novice user interaction
Information Scent Modeling	mimic Web site navigation

Table 1 39 usability evaluation methods (Ivory and Hearst, 2001, p. 476)

These are useful classifications, but Preece *et al.* (2003, p.3 44) provide a better classification because they separate paradigms of evaluation from the techniques of evaluation. For example, Dix *et al.*'s monitoring examples may be considered as being either experimental or observational. In Preece *et al.*'s classification, monitoring is a technique for "observing users" which can be applied to the paradigm of "usability testing" (although eye tracking and physiological measurement may be more invasive than the usual techniques of video or interaction logging).

Preece *et al.* (2003, p. 344) identified four paradigms

- "Quick and dirty"

- Usability testing

- field studies

- Predictive

which are summarised in Table 2.

	"Quick and dirty"	Usability testing	Field studies	Predictive
Role of users	Natural behavior.	To carry out set tasks.	Natural behavior.	Users generally not involved.
Who controls	Evaluators take minimum control.	Evaluators strongly in control.	Evaluators try to develop relationships with users.	Expert evaluators.
Location	Natural environment or laboratory.	Laboratory.	Natural environment.	Laboratory-oriented but often happens on customer's premises.
When used	Any time you want to get feedback about a design quickly. Techniques from other evaluation paradigms can be used - e.g. experts review software.	With a prototype or product.	Most often used early in design to check that users' needs are being met or to assess problems or design opportunities.	Expert reviews (often done by consultants) with a prototype, but can occur at any time. Models are used to assess specific aspects of a potential design.
Type of data	Usually qualitative, informal descriptions.	Quantitative. Sometimes statistically validated. Users' opinions collected by questionnaire or interview.	Qualitative descriptions often accompanied with sketches, scenarios, quotes, other artifacts.	List of problems from expert reviews. Quantitative figures from model, e.g., how long it takes to perform a task using two designs.
Fed back into design by . ..	Sketches, quotes, descriptive report.	Report of performance measures, errors etc. Findings provide a benchmark for future versions.	Descriptions that include quotes, sketches, anecdotes, and sometimes time logs.	Reviewers provide a list of problems, often with suggested solutions. Times calculated from models are given to designers.
Philosophy	User-centered, highly practical approach.	Applied approach based on experimentation, i.e. usability engineering.	May be objective observation or ethnographic.	Practical heuristics and practitioner expertise underpin expert reviews. Theory underpins models.

Table 2 Characteristics of different evaluation paradigms (Preece *et al.*, 2003, p. 344)

The authors then list five techniques (p.347)

- Observing users

- Asking users for their opinions

- Asking experts for their opinions

- Testing users' performance

- Modeling users' performance to predict the efficacy of a user interface

Table 3, p.39, shows their matrix of their paradigms and techniques. Each combination of technique and paradigm could be selected as a possible evaluation method. These methods are similar to the three evaluation methods identified by Sears (2003, p. 1091-1092): namely, user-, inspection- and model-based evaluation methods:

- User-based methods map to the technique of "asking users"

- Inspection-based methods map to the technique of "asking users" in "predictive" paradigm

- Model-based methods map to the technique of "modelling users' task performance" in "predictive" paradigm

I have added colour to the original source table for Table 3. Combinations of techniques and paradigms in dark red are not feasible and/or not suitable for this project. Field studies require more time, and access to, subjects than I would reasonably expect from users. Additionally, in order to gather meaningful data, there is a need for sustained access to subjects and for me to develop sufficient expertise to correctly apply the methods.

Techniques	Evaluation paradigms			
	"Quick and dirty"	**Usability testing**	**Field studies**	**Predictive**
Observing users	Important for seeing how users behave in their natural environments.	Video and interaction logging, which can be analysed to identify errors, investigate routes through the software, or calculate performance time.	Observation is the central part of any field study. In ethnographic studies evaluators immerse themselves in the environment. In other types of studies the evaluator looks on objectively.	N/A
Asking users	Discussions with users and potential users individually, in groups or focus groups.	User satisfaction questionnaires are administered to collect users' opinions. Interviews may also be used to get more details	The evaluator may interview or discuss what she sees with participants. Ethnographic interviews are used in ethnographic studies.	N/A
Asking experts	To provide critiques (called "crit reports") of the usability of a prototype.	N/A	N/A	Experts use heuristics early in design to predict the efficacy of an interface.
User testing	N/A	Testing typical users on typical tasks in a controlled laboratory-like setting is the cornerstone of usability testing.	N/A	N/A
Modelling users' task performance	N/A	N/A	N/A	Models are used to predict the efficacy of an interface or compare performance times between versions.

Table 3 The relationship between evaluation paradigms and techniques (Preece *et al.,* 2003, p. 347)

The aim of evaluation for this project is to inform the design process with the usability strengths and weaknesses of existing software. Some methods give usability information at the wrong level of detail. For example, whilst GOMS (Goals, Operators, Method and Selection) has had some remarkable success[1], it is not suitable because it models low-level data entry. Furthermore, GOMS has "highly limited scope: it can only really model computer-based tasks that involve a small set of highly routine-data entry type tasks." (Preece *et al.*, 2003, p. 454)

Combinations in white in Table 3 do not exist (not applicable) or are not suitable for M801: all of the "Quick and Dirty" methods may be useful, but must be supplemented with more substantial evaluation methods ("Quick and Dirty" methods lack the rigour needed for M801.)

The two remaining combinations in green are both possible and suitable:

a) Technique of "asking users" in "usability testing" paradigm: "User satisfaction questionnaires are administered to collect users' opinions. Interviews may also be used to get more details"

b) Technique of "asking experts" in "predictive" paradigm: "Experts use heuristics early in design to predict the efficacy of an interface."

These methods are now examined, selected and adapted in the next two sections.

---

[1] Project Enerstine showed that a new computer system to support telephone operators would be slower than the system it was designed to replace. By not adopting the new system NYNEX, a US telephone company, saved millions of dollars (Dix *et al.*, 2004, p. 424)

2.5.1 "Asking Users" in a "Usability Testing" Paradigm

In order to decrease possible bias of asking a limited number of "experts" in a predictive paradigm, opinions about usability will be sought from users via a questionnaire. This is based in part on QUIS, the Questionnaire for User Interaction Satisfaction. Shneiderman and Plaisant (2005, p.153-161) quote the full version of this questionnaire. The results will provide information about what people feel about current systems' usability and capabilities.

There are six aspects of usability in the QUIS questionnaire. They are not defined in the QUIS questionnaire, so each respondent will interpret them in his or her own way. However, they will be taken to mean the following[2]:

QUIS aspect	Description
Terrible / wonderful	Wonderful programs evoke feelings of positive surprise and admiration. Terrible programs are unpleasant to learn and to use.
Frustrating / Satisfying	Satisfying programs allow users fulfil their goals. Frustrating programs prevent users fulfilling their goals and/or discourage users in their attempts.
Dull / Stimulating	Dull programs are boring. Stimulating programs inspire users with new ideas or ways of doing things.
Difficult / Easy	Easy programs take little effort to learn and to use. Difficult programs and hard to learn and hard to use.

---

[2] Based, in part, on word definitions from Cambridge Dictionary Online (Cambridge Advanced Learner's Dictionary) URL http://dictionary.cambridge.org (13 Sep 2005)

Inadequate / Adequate Power	Programs with adequate power offer relevant features that users need to accomplish their goals. Note, however, that it is possible for a program to have adequate power but be frustrating (user cannot use the features) or dull (user does not enjoy using the features).
Rigid / Flexible	Flexible programs can adapt to the way users want to do things. Rigid programs lack flexibility: they offer a limited number of ways of doing things.

Dix *et al.*'s (2004, p260-261) three broad aspects of usability are covered by some of the QUIS aspects:

- Learnability – Difficult / Easy

- Flexibility – Rigid / Flexible

- Robustness – Frustrating / Satisfying and Inadequate / Adequate Power

Of Preece *et al.*'s (2003, p. 14) usability components, all expect one are covered:

- Effective to use (effectiveness) – Frustrating / Satisfying

- Efficient to use (efficiency) – Frustrating / Satisfying and Difficult / Easy

- Safe to use (safety) – not specifically addressed

- Good utility (utility) – Inadequate / Adequate Power

- Easy to learn (learnability) – Difficult / Easy

- Easy to remember how to use (memorability) – Difficult / Easy

2.5.2 "Asking Experts" in a "Predictive" Paradigm

Typical usability inspection methods are Cognitive Walkthrough and Nielsen's Heuristic Evaluation.

Preece *et al.* (2003, p.422) observe that "cognitive walkthrough is a useful technique for evaluating a small part of a system in detail, whereas heuristic evaluation is a useful for examining whole or parts of systems." This project evaluates a number of systems, so Cognitive Walkthrough is not feasible.

Nielsen's Heuristic Evaluation therefore appears more promising. It has become popular since its introduction (Nielsen, 1994) but has suffered from serious criticism (Gray and Salzman, 1998). Researchers have subsequently attempted to improve Heuristic Evaluation: Cockton *et al.* (2004) found that a questionnaire prompting for reflection from evaluators improved their abilities at finding relevant usability problems.

Law and Hvannberg (2004) examined five research questions regarding Heuristic Evaluation. They determined the success of Heuristic Evaluation (HE) by comparing the usability problems found by HE with actual user testing . They found that Nielsen's ten heuristics were superior to Gerhardt-Powal's Cognitive Engineering Principles (listed in Appendix A – Gerhart-Powals' Cognitive Engineering Principles, p. 133), even though the latter had a better theoretical grounding. Some of the reasons for this (*ibid.,* p. 248) are that Gerhardt-Powal's principles:

- Are hard for novice evaluators to understand
- Are narrow and deep ("confined to principles of perception") whereas Nielsen's heuristics are broad

- "Were initially developed for evaluating a highly dynamic military system with many numeric data presentations."

They note that

> there exist hundreds of guidelines, principles and criteria for design and evaluation of interactive interfaces. While it is not so challenging to add one more item into this an already (too) large knowledge pool, what is more challenging for researchers as well as practitioners is to select a right assortment and to validate it with highest possible experimental rigor. (*ibid.,* p. 248)

Therefore it is appropriate to select existing guidelines rather than attempt to create new ones. The next section describes suitable guidelines.

2.6 Principles, Guidelines and Standards

Section 2.3, presented some definitions of usability. Dix *et al.* (2004, p282) observes that abstract principles of usability need effort from a designer, "either to track down ... or to interpret". Designers need to be able to "determine the usability consequences of their design decisions ... *design rules* ...[allow a designer to]... increase the usability of the eventual software product." (*ibid.*, p. 259). Both Shneiderman and Plaisant (2005, p. 60-76) and Dix *et al.* (2004, p. 259-260) classify design rules into three levels (although they use the same words to mean slightly different things):

- High-level principles, models and theories are widely applicable. They are general and abstract and therefore require careful interpretation.

- Guidelines are more specific but less general. These include heuristics and "golden rules" like Nielsen's ten heuristics for Heuristic Evaluation, Norman's "Seven Principles for Transforming Difficult Tasks into Simple Ones" and Shneiderman's

"Eight Golden Rules of Interface Design" (Dix *et al.,* 2004, p. 282-284). Another set of guidelines is the Cognitive Dimensions framework.

- Standards are specific and practical. They have a narrow focus and usually must be followed. Typical examples are style guidelines like Apple's Macintosh *Human Interface Guidelines* (Apple Computer Inc., 1995) and Microsoft's *The Microsoft Windows User Experience* (Microsoft Corporation, 2004a).

Guidelines are "broad-brush" design rules (Dix *et al.,* 2004, p. 282) that "need validation and tuning for specific design domains" (Shneiderman and Plaisant (2005, p. 74). Dix *et al.* (2004, p. 282) state that "it is clear than any designer following even these simple rules will produce a better system than one who ignores them."

Guidelines are pitched at an appropriate level of generality: neither too wide nor too narrowly focused. Each of the four sets of guidelines mentioned above covers similar ground: this is to be expected because there is at least some kind of consensus about what "good/usable design" is. However, each has a different emphasis. The next sections examine each set of guidelines. Section 2.7 compares these guidelines and choose the Cognitive Dimensions framework as the most appropriate.

2.6.1 Nielsen's ten heuristics for Heuristic Evaluation

Dix *et al.* (2004, p282-284) lists Nielsen's ten heuristics (see Appendix B – Nielsen's ten heuristics for Heuristic Evaluation, p. 134, for a full description)

NH1.   Visibility of system status

NH2.   Match between system and the real world

NH3.   User control and freedom

NH4.   Consistency and standards

NH5.   Error prevention

NH6.   Recognition rather than recall

NH7.   Flexibility and efficiency of use

NH8.   Aesthetic and minimalist design

NH9.   Help users recognise errors, diagnose and recover from them

NH10.   Help and documentation

2.6.2 Norman's "Seven Principles for Transforming Difficult Tasks into Simple Ones"

Norman's (1988, p. 188-189) seven principles are:

N1. Use both knowledge in the world and knowledge in the head.

N2. Simplify the structure of tasks.

N3. Make things visible: bridge the gulfs of execution and evaluation.

N4. Get the mappings right.

N5. Exploit the power of constraints, both natural and artificial.

N6. Design for error.

N7. When all else fails, standardise.

See Appendix C – Norman's "Seven Principles for Transforming Difficult Tasks into Simple Ones", p. 136, for a fuller description of each principle.

2.6.3 Shneiderman's "Eight Golden Rules of Interface Design"

Shneiderman's eight golden rules (Shneiderman and Plaisant, 2004, p. 74-75) are:

S1. Strive for consistency

S2. Enable frequent users to use shortcuts

S3. Offer informative feedback

S4. Design dialogs to yield closure

S5. Offer error prevention and simple error handling

S6. Permit easy reversal of actions

S7. Support internal locus of control

S8. Reduce short-term memory load

See Appendix D – Shneiderman's "Eight Golden Rules of Interface Design", p. 138, for a fuller description.

2.6.4 The Cognitive Dimensions Framework

Green and Blackwell (1998) provide a tutorial on the Cognitive Dimension framework. They explore in detail the dimensions that are "less psychological" in character and mention the others (p. 11):

Less psychological dimensions treated in the tutorial

Abstraction   types and availability of abstraction mechanisms

Hidden dependencies   important links between entities are not visible

Premature commitment   constraints on the order of doing things

Secondary notation   extra information in means other than formal syntax

Viscosity   resistance to change

Visibility   ability to view components easily

Psychological dimensions not treated in the tutorial

Closeness of mapping	closeness of representation to domain
Consistency	similar semantics are expressed in similar syntactic forms
Diffuseness	verbosity of language
Error-proneness	notation invites mistakes
Hard mental operations	high demand on cognitive resources
Progressive evaluation	work-to-date can be checked at any time
Provisionality	degree of commitment to actions or marks
Role-expressiveness	the purpose of a component is readily inferred

The next section chooses compares the four sets of guidelines and chooses the Cognitive Dimensions framework as the most appropriate.

2.7 A Comparison of Cognitive Dimensions, Nielsen's Heuristics, Norman's Seven Principles and Shneiderman's Eight Golden Rules

As mentioned in section 2.6, p. 44, there is a degree of consensus between the four guidelines. The next sections describe each Cognitive Dimension (CD) in more detail (Blackwell and Green, 2003, p. 115-118; Green and Blackwell, 1998, p. 12-41) and compares each CD with the other guidelines. Table 5, p. 66, summarises these findings.

2.7.1 Abstraction

Nielsen	Shneiderman	Norman	Cognitive Dimension
NH7	S2	N1	**Abstraction**
Flexibility and efficiency of use	Enable frequent users to use shortcuts	Use both knowledge in the world and knowledge in the head.	types and availability of abstraction mechanisms

Figure 11 Comparison of guidelines by Nielsen, Shneiderman and Norman with the Cognitive Dimension

Abstraction

Figure 11 shows a thumbnail description of this Cognitive Dimension and its related guidelines. Green and Blackwell (1998, p. 24) state that

> An abstraction is a class of entities, or a grouping of elements to be treated as one entity, either to lower the viscosity or to make the notation more like the user's conceptual structure.

An example of abstraction is the use of styles in word processors: Styles allow the user to define a new term to redefine a longer series of definitions into one. For example setting all level 1 headings to 24-point bold would require the user to find the next heading, select it, set the size to 24, and set the style to bold. This can be defined as select "Heading 1" and set size and style. The word processor is *abstraction tolerance* because styles are not necessary for its use (*ibid.*)

Abstraction hungry notations require users to learn a large number of existing abstractions before they can get to work. Some programming languages could be considered as being abstraction hungry. (*ibid.*)

Although using abstractions may save the user work in the future (by making the notation less viscous), abstractions can be costly due to

- abstractions being hard to learn and to use
- the overhead of creating, editing and maintaining the abstraction (*ibid.*)

Allowing the user to create abstractions satisfies NH7 and S2. Abstractions may address N1: "experts ... need to be able to internalise regular tasks to increase their efficiency."

2.7.2 Hidden dependencies

Nielsen	Shneiderman	Norman	Cognitive Dimension
			Hidden dependencies important links between entities are not visible

Figure 12 Comparison of guidelines by Nielsen, Shneiderman and Norman with the Cognitive Dimension

Hidden dependencies

Figure 12 shows a thumbnail description of this Cognitive Dimension and the fact that it has

no related guidelines. Green and Blackwell (1998, p. 17) define a hidden dependency as

> a relationship between two components such that one of them is dependent on the
>
> other, but that the dependency is not fully visible.

Classic examples are (*ibid.*)

> HTML links: if your page is linked to someone else's .... how will you know if and
>
> when that page is moved, changed, or deleted?
>
> Many links are fossils – out of date pointers to pages that have been deleted or moved.
>
> Because checking links is slow, the *search cost* for testing integrity of a site is quite
>
> high, so fossils are likely to increase over the years. (*my emphasis*)

Hidden dependencies may be

- one-way (shows only the target) or symmetric (shows both source and target of a
  dependency)

- local (only points to immediate target) or distant (leads to more deeply nested target)

- hidden (not normally visible) or explicit (always shown in the notation's normal
  viewing state)

Abstractions may impose hidden dependencies (*ibid.,* p. 19). Hidden dependencies may be a

side-effect of low viscosity (*ibid.,* p. 20)

2.7.3 Premature commitment

Nielsen	Shneiderman	Norman	Cognitive Dimension
	S7 Support internal locus of control		**Premature commitment** constraints on the order of doing things

Figure 13 Comparison of guidelines by Nielsen, Shneiderman and Norman with the Cognitive Dimension

Premature commitment

Figure 13 shows a thumbnail description of this Cognitive Dimension and its related

guideline. Green and Blackwell (1998, p. 21) define premature commitment:

> Constraints on the order of doing things force the user to make a decision before the
>
> proper information is available.

The authors distinguish *enforced lookahead:* this happens when the user must "look ahead in

a way that is cognitively expensive" (*ibid.*)

Blackwell and Green (1998, p. 116) give the examples of "being forced to declare identifiers

too soon; choosing a search path down a decision tree; having to select your cutlery before

you choose your food"

2.7.4 Secondary notation

Nielsen	Shneiderman	Norman	Cognitive Dimension
			Secondary notation extra information in means other than formal syntax

Figure 14 Comparison of guidelines by Nielsen, Shneiderman and Norman with the Cognitive Dimension

Secondary notation

Figure 14 shows a thumbnail description of this Cognitive Dimension and the fact that it has no related guidelines. Green and Blackwell (1998, p. 29) define secondary notation as

> Extra information carried by other means than the official syntax. *Redundant recoding* gives a separate and easier channel for information that is already present in the official syntax [e.g. indentation in programs, grouping controls by function]. *Escape from formalism* allows extra information to be added, not present in the official syntax. [e.g. comments in programs, colour and formatting in spreadsheets]

Extensive use of secondary notation may increase viscosity unless tools exist to support it e.g. automatic indentation facilities (*ibid,* p. 33).

2.7.5 Viscosity

Nielsen	Shneiderman	Norman	Cognitive Dimension
			Viscosity resistance to change

Figure 15 Comparison of guidelines by Nielsen, Shneiderman and Norman with the Cognitive Dimension *Viscosity*

Figure 15 shows a thumbnail description of this Cognitive Dimension and the fact that it has no related guidelines. Viscosity is (Green and Blackwell, 1998, p. 12)

> resistance to change: the cost of making small changes.
>
> *Repetition viscosity* [is where] a single goal-related operation on the information structure (one change 'in the head') requires an undue number of individual actions [e.g. manually changing US spelling to UK spelling in a long document]
>
> *Knock-on viscosity* [is where] one change 'in the head' entails further actions to restore consistency [e.g. inserting a new figure into a document requires updating all subsequent figure numbers, cross-references within the text and also the list of figures and index]

Interestingly, the authors (*ibid.*, p. 13) note that

> Editing a drawing usually requires much tedious work, and frequently many similar
>
> alterations need to be made to different parts of the picture; automation tools,
>
> desirable as they might be, are not yet commercially available.

2.7.6 Visibility

Nielsen	Shneiderman	Norman	Cognitive Dimension
NH1	S3	N3	**Visibility**
Visibility of system status	Offer informative feedback	Make things visible: bridge the gulfs of execution and evaluation.	ability to view components easily

Figure 16 Comparison of guidelines by Nielsen, Shneiderman and Norman with the Cognitive Dimension
Visibility

Figure 16 shows a thumbnail description of this Cognitive Dimension and its related

guidelines. Visibility (Green and Blackwell, 1998, p. 34) is the

> ability to view components easily
>
> *Juxtaposability* [is the] ability to place any two components side by side.

High visibility is required for exploratory activities and may be useful for some transcription

activities (Green and Blackwell, 2003, p. 116)

Poor visibility can affect direct manipulation interfaces: if a user cannot see something, the

user cannot manipulate it.

Juxtaposability is needed when a user is transcribing data that needs to be consistent: the user

must be able to place components next to each other in order to check and/or copy data

and/or formats. (Green and Blackwell, 1998, p. 36)

The three other guidelines all mention visibility. Norman states that systems should allow the user to see what commands are possible and check the effect of any commands executed (feedback). Nielsen and Shneiderman emphasise system responsiveness and feedback.

2.7.7 Closeness of mapping

Nielsen	Shneiderman	Norman	Cognitive Dimension
NH2 Match between system and the real world		N1 Use both knowledge in the world and knowledge in the head.	**Closeness of mapping** closeness of representation to domain

Figure 17 Comparison of guidelines by Nielsen, Shneiderman and Norman with the Cognitive Dimension
Closeness of mapping

Figure 17 shows a thumbnail description of this Cognitive Dimension and its related guidelines. Green and Blackwell (1998, p. 39) define *closeness of mapping* as "closeness of representation to domain". They cite the visual programming language, LabVIEW, as having good closeness of mapping. LabVIEW uses the idea of a circuit diagram to minimise the number of new concepts that electronic engineers need to learn.

However, there are some well-known pitfalls of using a *metaphor* (Dix et al., 2004, p. 170) for user interfaces: the analogy can be inadequate or misleading, and might not have the same meaning across cultures. Pirhonen (2005) states that frequently cited examples of *metaphor* are in fact *simulations*, e.g. push buttons on a screen *imitate (simulate)* the action of real-life physical buttons.

Nielsen's second heuristic refers to the need for a good match between the system and the user's world. (NH2: Match between system and the real world. "The system should speak the

user's language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions")

Norman's first principle recommends that systems be designed to enable users to build an appropriate mental model. (N1: Use both knowledge in the world and knowledge in the head. "...systems should provide the necessary knowledge within the environment and their operation should be transparent to support the user in building an appropriate mental model of what is going on.") Norman (1988, p. 70-72) explains that mental models allow users to understand a system and predict its behaviour, even if that understanding is inadequate or wrong (ibid., 68-69). Good mental models use knowledge that users already have, and are hence easier to learn. (Dix *et al.*, 2004, p 261)

2.7.8 Consistency

Nielsen	Shneiderman	Norman	Cognitive Dimension
NH4 Consistency and standards	S1 Strive for consistency in action sequences, layout, terminology, command use and so on.	N7 When all else fails, standardise.	**Consistency** similar semantics are expressed in similar syntactic forms

Figure 18 Comparison of guidelines by Nielsen, Shneiderman and Norman with the Cognitive Dimension *Consistency*

Figure 18 shows a thumbnail description of this Cognitive Dimension and its related guidelines. Consistent notations express "similar semantics ... in similar syntactic forms" (Green and Blackwell, 1998, p. 39).

All guidelines recognise the importance of consistency for ease of learning and memorability. Nielsen observes that inconsistent interfaces can confuse users (NH4). Shneiderman (S1) notes that consistency should apply not only to terminology and actions, but also to layout, command use and other system properties. Norman (N7) notes that if a mapping needs to be arbitrary, it should follow any existing standards: there is no natural or logical layout for car controls, so it makes sense to follow existing standards for the accelerator, brake, clutch and steering.

2.7.9 Diffuseness

Nielsen	Shneiderman	Norman	Cognitive Dimension
NH8 Aesthetic and minimalist design	S8 Reduce short-term memory load: Displays are simple.	N2 Simplify the structure of tasks	**Diffuseness** verbosity of language

Figure 19 Comparison of guidelines by Nielsen, Shneiderman and Norman with the Cognitive Dimension *Diffuseness*

Figure 19 shows a thumbnail description of this Cognitive Dimension and its related guidelines. Diffuse languages are verbose. Long-winded names are usually harmless, but can be bad for exploratory activities as the can interfere with working memory. However, terseness can increase error-proneness (Green and Blackwell, 1998, p. 39-40).

Nielsen (NH8) recognises that irrelevant or rarely needed information competes with the relevant information and diminishes the relative visibility of relevant information. Shneiderman (S8) also recommends that display be kept simple – this it to reduce the load on short-term memory. Norman also acknowledges the limitations of short-term memory: he recommends that unnecessarily complex tasks be restructured (Norman, 1988, p. 191)

2.7.10 Error-proneness

Nielsen	Shneiderman	Norman	Cognitive Dimension
NH3 User control and freedom NH5 Error prevention NH9 Help users recognise errors, diagnose and recover from them	S5 Offer error prevention and simple error handling S6 Permit easy reversal of actions	N5 Exploit the power of constraints, both natural and artificial. N6 Design for error	**Error-proneness** notation invites mistakes

Figure 20 Comparison of guidelines by Nielsen, Shneiderman and Norman with the Cognitive Dimension *Error-proneness*

Figure 20 shows a thumbnail description of this Cognitive Dimension and its related guidelines. Error-prone notations invite mistakes. These can either be *slips* or *errors* (Blackwell and Green, 2003, p. 116). Presumably *error* in this context has the same meaning as Norman's (1988, p. 114) *mistake*. Norman (1988, p. 106) defines a *slip* as an error resulting from a lack of attention or concentration. A typical slip is a user forgetting which

mode they are in (*ibid.*, p 110), for example a user of a spreadsheet attempting to save when it is, in fact, in "edit formula" mode. A *mistake* is a result of a user choosing inappropriate goals (*ibid.*, p. 114). For example, a naïve BASIC programmer may mistake the PRINT statement as sending output to a printer, when in fact it displays output on a screen (although, in fact, the programmer may well have been right if using a very old BASIC system with a teletype device.)

All of the other guidelines mention error: error messages with recovery information are good, but preventing errors happening in the first place is even better. This can be expressed in slightly different ways: e.g. Norman implicitly addresses this with mappings and constraints (N5) that prevent *slips* and *mistakes*.

2.7.11 Hard mental operations

Nielsen	Shneiderman	Norman	Cognitive Dimension
NH6 Recognition rather than recall	S8 Reduce short-term memory load	N2 Simplify the structure of tasks	**Hard mental operations** high demand on cognitive resources

Figure 21 Comparison of guidelines by Nielsen, Shneiderman and Norman with the Cognitive Dimension *Hard mental operations*

Figure 21 shows a thumbnail description of this Cognitive Dimension and its related guidelines. Hard mental operations place a

> high demand on cognitive resources. A notation can make things complex or difficult
> to work out in your head, by making inordinate demands on working memory or by
> requiring deeply nested goal structures (Blackwell and Green, 2003, p. 117)

The other guidelines recognise that relying on recognition reduces demands on memory (NH6). Irrelevant information can burden the user's short-term memory (S8). Norman suggests that unnecessarily complex tasks be restructured to simplify them (N2).

2.7.12 Progressive evaluation

Nielsen	Shneiderman	Norman	Cognitive Dimension
	S3 Offer informative feedback for every user action, at a level appropriate to the magnitude of the action. S4 Design dialogs to yield closure		**Progressive evaluation** work-to-date can be checked at any time

Figure 22 Comparison of guidelines by Nielsen, Shneiderman and Norman with the Cognitive Dimension *Progressive Evaluation*

Green and Blackwell (1998, p. 40) define progressive evaluation as the ability to check work-to-date at any time. Blackwell and Green (1998, p. 40) state that evaluation is important for design and notational systems can help by allowing users to stop in the middle to check work so far, keep track of progress, or check what stage they are at. The authors cite BASIC as in interpreted programming language that lets users try out partially-completed programs.

Shneiderman addresses this (S3) as feedback for every user action. However, some notations required a goal to be broken into a number of user actions before the user can evaluate the success of the actions. Shneiderman's fourth golden rule addresses this to an extent, but is focused on dialogs. The CD framework attempts to take into account more general versions of *delayed gratification.*

2.7.13   Provisionality

Nielsen	Shneiderman	Norman	Cognitive Dimension
	S6 Permit easy reversal of actions		**Provisionality** degree of commitment to actions or marks

Figure 23 Comparison of guidelines by Nielsen, Shneiderman and Norman with the Cognitive Dimension

Provisionality

Figure 23 shows a thumbnail description of this Cognitive Dimension and its related

guidelines. Provisionality is the "degree of commitment to actions or marks". Even if there

are constraint on the order of doing things, some notations allow the user to record sketchy

ideas, design options and play *what-if* games. (Blackwell and Green, 2003, p. 117-118)

Shneiderman (S6) mentions easy reversal of action: this can interpreted as allowing

exploration and taking provisional actions. However, some provisional actions do not need to

be reversed, merely allowed to be sketchy and incomplete.

2.7.14   Role-expressiveness

Nielsen	Shneiderman	Norman	Cognitive Dimension
NH4 Consistency and standards	S1 Strive for consistency	N3 Make things visible: bridge the gulfs of execution and evaluation. N4 Get the mappings right.	**Role-expressiveness** the purpose of a component is readily inferred

Figure 24 Comparison of guidelines by Nielsen, Shneiderman and Norman with the Cognitive Dimension

Role-expressiveness

Figure 24 shows a thumbnail description of this Cognitive Dimension and its related

guidelines. Role-expressive notations make it easy to discover why an author has built a

structure in a particular way (Blackwell and Green, 2003, p. 116). When a user is reading a

role-expressive notation, it is easy to break the notation into its component parts, and pick out

relationships between those parts (Green and Blackwell, 1998, p. 41). It may be easy or

difficult to perceive how a set of origami diagram may been built: it depends on the

notational system being used.

Norman (N4) states that systems should make it "clear what does what and by how much."

Consistency may help (NH4).

2.7.15   Guidelines not present in the Cognitive Dimensions framework

Nielsen	Shneiderman	Norman	Cognitive Dimension
NH10 Help and documentation			

Figure 25 Guidelines by Nielsen, Shneiderman and Norman that are not present in the Cognitive Dimension framework

Only Nielsen specifically mentions documentation (NH10).

2.8 Unique features of the Cognitive Dimensions framework

The Cognitive Dimensions framework uses a number of concepts that do not explicitly

appear in other guidelines (Hidden dependencies, Secondary notation and Viscosity).

Furthermore, the Cognitive Dimensions framework classifies the importance of each

dimension according to activity (Table 4):

incrementation	adding cards to a cardfile; formulas to a spreadsheet, or statements to a program
transcription	copying book details to an index card; converting a formula into spreadsheet or code terms
modification	changing the index terms in a library catalogue; changing the layout of a spreadsheet; modifying a spreadsheet or program to compute a different problem
exploratory design	sketching; design of typography, software, etc; other cases where the final product cannot be envisaged and has to be "discovered"
searching	hunting for a known target, such as where a function is called
exploratory understanding.	discovering structure or algorithm, or discovering the basis of classification

Table 4 The six types of activity in the Cognitive Dimensions framework (Green and Blackwell, 2003, p. 113, table 5.1)

For example, "secondary notation" may be useful for transcription, but is considered very harmful for exploration. (Green and Blackwell, 1998, p. 42)

Another practical feature is that the CD framework makes explicit workarounds, remedies and design trade-offs.

The main drawbacks of Heuristic Evaluation are false positives and the fact that genuine usability problems are sometimes missed (Law and Hvannberg, 2004, p. 243). These drawbacks are important but are sidestepped in this project. The drawbacks happen when

evaluators are novices who are applying the heuristics to a program that do not know. However, for this project, evaluators are rating the usability of a program that they *do* know. They are not using heuristics: they are using a checklist of usability criteria that have been customised for the domain that they know (the domain of making origami instructions.)

2.9 Evaluation using the Cognitive Dimensions framework

Blackwell and Green (2003, p104-106) claim that the CD framework

- Offers a comprehensible, broad-brush evaluation (no "death by detail")

- Uses terms that are readily comprehended by non-specialists

- Is applicable not just to interactive devices, but also to paper-based notations and other non-interactive information systems e.g. timetables

- Is theoretically coherent

- Distinguishes between different types of user needs (such as the difference between dictation tasks and design tasks)

- Frequently reveals a variety of interesting design choices

- Describes trade-offs between design choices, showing how solving one type of user difficulty may create a different type

Roast *et al.* (2004) reported that the use of Cognitive Dimensions found two significant usability problems that were missed by Contextual Analysis and Ontological Sketch Mapping.

Blackwell and Green (2000) describe the methodology behind the use of a questionnaire to evaluate systems using the Cognitive Dimensions framework. They trialled the questionnaire with programmers ranging from novice to (mostly) expert. Systems evaluated included

programming languages (C++/Emacs, LaTeX, Oracle procedural SQL and Mathematica) and music typesetting languages such as PMS, Calliope and Finale. (*ibid.*, p. vii)

Unfortunately the original CD questionnaire would have taken too long with my target population, whether used in an interview or unsupervised (The former took 35 – 60 minutes, the latter days or several weeks to return, *ibid.*, p viii). Moreover, it is unlikely that the target population would have had the motivation to complete such an open-ended questionnaire, nor the ability to interpret the terminology and questions. This is despite claims that CD "terms are readily comprehended by non-specialists" (Blackwell and Green, 2003, p104-106) Therefore each question in the CD questionnaire has been replaced with a statement that a subject could agree or disagree with (Table 6, p. 68)

Nielsen	Shneiderman	Norman	Cognitive Dimension
NH7 Flexibility and efficiency of use	S2 Enable frequent users to use shortcuts	N1 Use both knowledge in the world and knowledge in the head.	**Abstraction** types and availability of abstraction mechanisms
			Hidden dependencies important links between entities are not visible
	S7 Support internal locus of control		**Premature commitment** constraints on the order of doing things
			Secondary notation extra information in means other than formal syntax
			Viscosity resistance to change
NH1 Visibility of system status	S3 Offer informative feedback	N3 Make things visible: bridge the gulfs of execution and evaluation.	**Visibility** ability to view components easily
NH2 Match between system and the real world		N1 Use both knowledge in the world and knowledge in the head.	**Closeness of mapping** closeness of representation to domain
NH4 Consistency and standards	S1 Strive for consistency in action sequences, layout, terminology, command use and so on.	N7 When all else fails, standardise.	**Consistency** similar semantics are expressed in similar syntactic forms
NH8 Aesthetic and minimalist design	S8 Reduce short-term memory load: Displays are simple.	N2 Simplify the structure of tasks	**Diffuseness** verbosity of language

Nielsen	Shneiderman	Norman	Cognitive Dimension
NH3 User control and freedom NH5 Error prevention NH9 Help users recognise errors, diagnose and recover from them	S5 Offer error prevention and simple error handling S6 Permit easy reversal of actions	N5 Exploit the power of constraints, both natural and artificial. N6 Design for error	**Error-proneness** notation invites mistakes
NH6 Recognition rather than recall	S8 Reduce short-term memory load	N2 Simplify the structure of tasks	**Hard mental operations** high demand on cognitive resources
	S3 Offer informative feedback for every user action, at a level appropriate to the magnitude of the action. S4 Design dialogs to yield closure		**Progressive evaluation** work-to-date can be checked at any time
	S6 Permit easy reversal of actions		**Provisionality** degree of commitment to actions or marks
NH4 Consistency and standards	S1 Strive for consistency	N3 Make things visible: bridge the gulfs of execution and evaluation. N4 Get the mappings right.	**Role-expressiveness** the purpose of a component is readily inferred
NH10 Help and documentation			

Table 5 : Common features of Nielsen's Heuristics, Shneiderman's Golden Rules, Norman's Seven

Principles and Cognitive Dimensions

Cognitive Dimension	Statements based on the CD questionnaire's open question(s)
Abstraction types and availability of abstraction mechanisms	1. I can define new terms (e.g. definition of arrow heads, named styles and colours) which allows me to express my ideas more clearly.
	2. I have to define new terms before I can do anything else (e.g. names of variables)
Hidden dependencies important links between entities are not visible	3. Some parts of the program are related to another: changing one part may affect others. I can usually see these kinds of dependencies (e.g. effect of changing a named style or colour; effect of inserting a new step on step numbering)
	4. As the document gets larger, problems with dependency get bigger.
Premature commitment constraints on the order of doing things	5. I can order the diagramming tasks in any way I like (e.g. start with final drawing; add/edit labels and captions to steps at any time)
	6. I need to plan and think ahead before starting to work.
Secondary notation extra information in means other than formal syntax	7. I can make notes to myself that are separate from the origami instructions e.g. use comments, colours, formatting, etc
Viscosity resistance to change	8. I can easily make changes to previous work.
	9. Some kinds of changes that are important are more difficult to make than they should be.
Visibility ability to view components easily	10. I can easily find the parts of the diagram that I am interested in whilst it is being created or changed.
	11. When I need to compare/combine different parts of the diagrams, I can see them at the same time.

Cognitive Dimension	Statements based on the CD questionnaire's open question(s)
Closeness of mapping closeness of representation to domain	12. The program works in a way that closely maps to how diagrams work.
	13. There are parts of the program which seem particularly strange for origami diagramming.
Consistency similar semantics are expressed in similar syntactic forms	14. Things that are similar are presented in similar ways (e.g. squares, rectangles and polygons can all be edited in similar ways; program asks for input in similar ways)
Diffuseness verbosity of language	15. The program lets me make diagrams reasonably briefly (not long-winded)
Error-proneness notation invites mistakes	16. It is easy to make mistakes.
	17. I often find myself making small slips that irritate me/make me feel stupid.
Hard mental operations high demand on cognitive resources	18. I sometimes need to work things out in my head that are complex or difficult.
	19. There are some tasks that make inordinate demands on my memory or are long-winded.
Progressive evaluation work-to-date can be checked at any time	20. It is easy to stop and check the diagrams in the middle of completion.
	21. I can check the work at any time.
	22. I can try out partially-completed versions of instructions.
Provisionality degree of commitment to actions or marks	23. I can sketch out things when playing with ideas, or when I'm not sure how to proceed.
Role-expressiveness the purpose of a component is readily inferred	24. I can easily tell what each function/feature of the program is for.

Table 6 Questionnaire statements for Cognitive Dimensions

2.10 Summary

This chapter introduced HCI as a field. It defined usability and described different types of

evaluation methods. Two methods were selected as appropriate to this project: expert

evaluation using the Cognitive Dimensions framework and a usability questionnaire.

The next chapter describes the design of the usability questionnaire and presents the results.

3   Initial Questionnaire

Two evaluation methods were selected in the previous chapter: expert evaluation using the Cognitive Dimensions framework (described in Chapter 4) and a usability questionnaire. This chapter discusses the design and result of the initial questionnaire to gather opinions about existing practice and the usability of software. (Chapter 7, p. 119, presents the results of a *second* usability questionnaire that focuses on the prototype implementation of a redesigned interface.)

3.1 *Aims of the questionnaire*

The general aim of the questionnaire is to gather users' opinions on the usability of the *software* that they use, not the abilities of the user. However, the user is an important variable, as are the methods that the user chooses to accomplish tasks in the software. For example, when using a vector drawing program a user may choose to either draw by eye only, or trace photographs in the software, or construct diagrams using tools such as scaling, rotation, perspective grids, etc. – some methods are inherently more usable than others.

The expertise of the user may affect the result of evaluation: a novice is likely to find an origami-oriented programming language like Doodle or Oridraw hard to use and learn, but an expert may be skilled and extremely productive. Alternatively, a novice may find a vector drawing program initially easy to use, but may "plateau" in productivity until he or she gains sufficient experience to use more advanced tools and techniques (e.g. a user may find it difficult to accurately position and align objects unless they use a "Snap to Point" feature).

3.2 The Design Rationale

The questionnaire was designed to gather data on the first four objectives listed in Chapter 1.

These were:

- Which software is used?

- What qualities should "good diagrams" possess?

- How well is the software used?

- Which approaches are most fruitful?

Preece *et al.* (2003, p. 171) point out that there are many interpretations of the "users" of a

system. Besides those who use the system directly, there those who

- Manage direct users

- receive products from the system

- Test the system

- Make purchasing decisions

- Use competitive products

I initially aimed my questionnaire at people who make diagrams. However, given Preece *et

al.'s* insight, I added a section for those who "receive products from the system". Jackson

(1989) criticised computer diagrams as being cold and lifeless. Lang (1989c) defended his

position: "The advantage of the computer is not that it makes it easier for the diagrammer to

draw diagrams, but that it makes it easier to draw good ones". He then mentions the shortage

of diagrammers and that the "mechanical approach has the virtue that less artistic ability is

required to produce acceptable drawings."

Later, Lang (2000, p. 30) wrote

> Of course, the greatest change [since 1990] has been the widespread adoption of computer-aided origami diagramming (CAOD). Hard to believe, but in 1990, CAOD was highly controversial: now it is the standard. It has brought the ability to draw reasonable-quality diagrams to virtually anyone with the desire to diagram.

Therefore it seemed sensible to make sure that computer diagrams did not repel potential readers of diagrams.

Comparing conventional and computer methods for the same respondent will show which activities are made easier by using a computer, if any – equally, there may be tasks that are made more difficult. This will highlight the activities where usability could be improved.

Preece *et al.* (2003) and Dix *et al.* (2004) both provide similar advice on using questionnaires to gather user opinion.

- Prefer closed questions to open-ended ones, both to ease burden on users and to ease analysis
- Design is critical to getting a good response
- Structure the questionnaire for logical order and ease of use
- Try to get a sample that is representative of the population – however, "in practice questionnaire respondents are self-selecting, anyway" (Dix et al. 2004, p350)
- Pilot with 4 or 5 users to check comprehensibility, results are as expected and can be used as intended

3.3 Contents

The questionnaire (Appendix G – Initial Questionnaire, p. 157) was divided into three

sections:

- General background, experience and origami questions, including opinions about the

  qualities of good diagrams (based on Lang's (2000, p2-5) ten guiding principles for

  diagramming)

- Conventional diagramming: questions drawn from Cunliffe (1988; 1989a, b)

- Computer diagramming: questions drawn from QUIS, the Questionnaire for User

  Interaction Satisfaction (Shneiderman and Plaisant, 2005, p.153-161)

Appendix F – Redesign of Initial Questionnaire based on Pilot Study Questionnaire, p. 152,

details the redesign based on the results of the pilot version.

3.4 Distribution

Paper versions of the questionnaire were distributed at the BOS (British Origami Society)

Convention, Nottingham, on Sunday, 10 April. I received around a dozen completed

questionnaires on the day. A couple of subjects kindly sent their responses by post.

I then posted announcements to three email discussion groups devoted to origami: origami-l,

BOSmail and paperwonders. I posted Microsoft Word and RTF (rich text format) versions to

a web site. This generated about twenty more responses over the first week. More responses

came when I made HTML and text-only versions available.

3.5 Results

My electronic invitation to mailing lists generated about 18 further responses. In total there

were 35 responses: 22 are conventional diagrammers and 22 are computer diagrammers. 16

gave information on both methods. See Appendix K – Results of Initial Questionnaire, p.

190, for results.

Subjects have a good range of experience:

- Years of interest in origami ranged from 2 to 50 years (mean 23 years, mode and median 20 years)

- Years of conventional diagramming ranged from 1 to 36 years (mean 15 years, mode and median 10 years)

- Years of computer diagramming ranged includes 1 to 15 years (mean 6 years, mode and median 3 years)

- Output ranged from personal and limited to self-publishers (including internet) and professional authors

However,

- The majority of subjects are male (78%) and the percentage is even higher for diagrammers (between 80 to 100% male, depending on the method). This may reflect the relative dearth of female diagrammers: in a recent BOS publication, just 13% of contributors were female (British Origami Society, 2005).

- There are very few subjects under 18 – again this may reflect reality (see Figure 26)
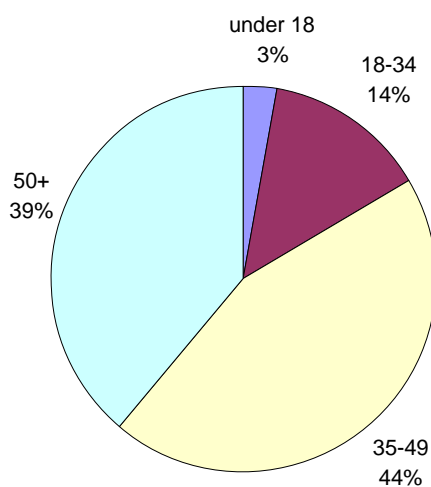
Figure 26 Pie chart of respondents by age category

3.5.1 Qualities of good diagrams
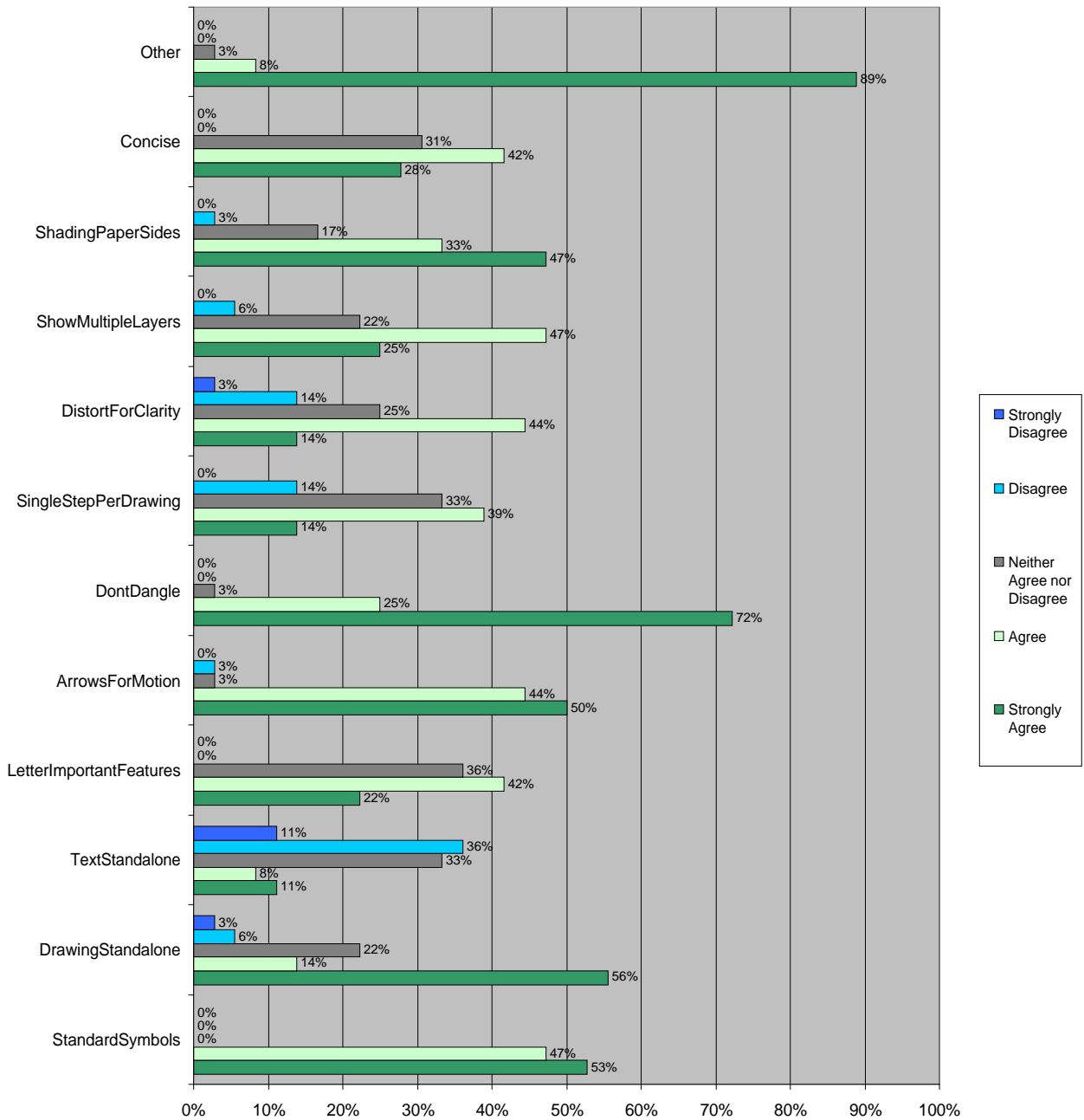
Figure 27 Bar chart of responses for qualities of diagrams

Figure 27 shows which of the modified set of Lang's guiding principles for diagramming

respondents considered important or unimportant.

The most important principles which virtually all subjects agreed with were

- Don't Dangle (L7)

- Standard Symbols (L1)

- Arrow for Motion (L6)

The next most important principles are those where most subjects agreed but a few were

neutral (none disagreed):

- Shading Paper Sides (question 15)

- Show Multiple Layers (L10)

- Drawing Standalone (L2)

- Concise (question 16)

The least important principles were

- Letter Important Features (L4)

- Single Step Per Drawing (L8)

- Distort For Clarity (L9)

Subjects only disagreed about one principle:

- Text Standalone (L3)

3.5.2 Overall Ease of making origami instructions

Questions 30 to 38 asked respondents to rate different types of programs for ease of making

origami instructions. Respondents answered for both their current main program and any

others that they have used. Therefore there are likely to be more negative ratings as people

may have tried a number of unsatisfactory programs before finding one that they feel is best

for them.

Table 7 tabulates individual responses where each letter represents an individual respondent. Vector program received the best ratings by far. General programs like Microsoft Word or PowerPoint received the worst ratings. There were few, if any, users of the other program types, which means conclusive analysis is impossible. However, none fared particularly well. One user rated a graphics tablet as Easy in the "Other" program type.

Ease of making origami instruction		Vector Drawing	General	CAD	Other Specialist Graphics	Origami Programming	3D	Dynamic Geometry	Origami Simulator	Other	Bitmap
Easy	1	jklqrz			i					l	
	2	cst	g	d							
Neither	3	mnopu	a	gs		H				bce	ab)
	4		cempq	i	s		t	f			
Difficult	5		bfisuv	q	c	S	s				

Table 7 Individual responses for ease of making origami instructions for different types of programs

3.5.3 Vector drawing programs

Of the 22 computer diagrammers, the majority chose a vector drawing program (e.g.

Freehand, Illustrator, CorelDRAW! and Serif Draw Plus 4.0) as their main program (Table 8

and Figure 28)

Number of subjects	Program type	% of computer diagrammers using as main program
13	Vector	58%
3	Bitmap	13%
3	General	14%
1	CAD	5%
1	Programming	5%
1	Specialist	5%

Table 8 Popularity of different program types for diagramming with a computer
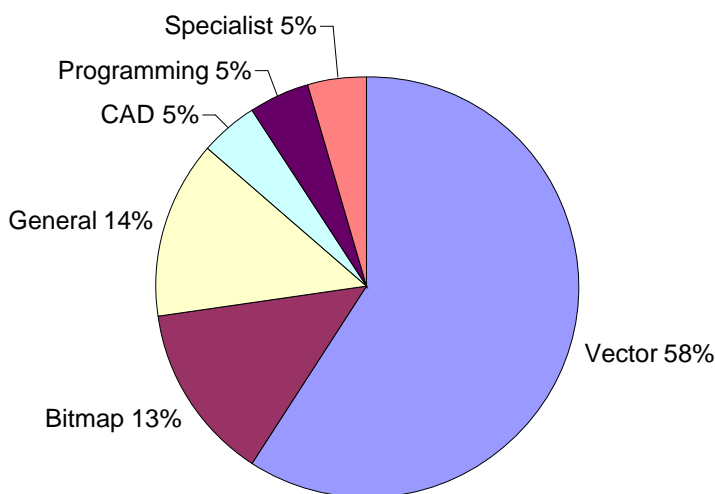
Figure 28 Pie chart showing the popularity of different program types for diagramming with a computer

	Terrible	Frustrating	Dull	Difficult	Inadequate Power	Rigid
1						
2		k	t	k		r
3				c	kn	
4	cno	not	cnoqsu	lmqs	pu	kp
5	kps	msu	kmr	ou	os	cmosu
6	jrt	cjpr	l	jnprt	cjmr	jnt
7	lmqu	lq	jp		lqt	lq
	Wonderful	Satisfying	Stimulating	Easy	Adequate Power	Flexible

Table 9 Individual QUIS responses for vector drawing program users

Table 9 shows each subject's responses in the grid – each letter represents an individual respondent. None of the individual QUIS questions received a negative rating from more than two of the vector program users. Of the four subjects who gave negative responses, only subject "k" gave more than one negative response. Giving appropriate weight to the proportion of positive, neutral and negative ratings shows that subjects felt vector drawing programs are

- Wonderful

- Have adequate power (one subject disagreed).

- Satisfying (two subjects disagreed)

The users felt that the programs allowed them to fulfil their goals in a pleasant manner. Three users specifically mentioned the ability to define named styles as a valued feature. Several praised their program's printing quality and flexibility of layout.

Vector drawing programs were rated, to a slightly lesser extent than the three aspects mentioned above, as:

- Flexible (less strong agreement)

- Not particularly dull nor stimulating (six neutral)

- Fairly easy (less strong agreement)

Supporting comments mentioned some difficulty in customising line styles and difficulty in editing graphic objects. This may explain the lack of stimulation and range of responses for ease of learning and ease of use.

Comparing Ease of Diagramming for vector users with their experience of conventional diagramming shows that whilst one subject felt all stages became easier (subject 26), some felt that some stages became harder.

For vector users, there was ratio of about 10:1 between the fastest and slowest self-reported estimates of diagramming the traditional crane and cup. Those with 5 or more years experience claimed times between 0.5 and 1 hour. Those with less than 5 years' experience estimated between 1 – 4 hours (mean 2.4 hours)

3.5.4 General programs

Only three users used this as their main diagramming program. Two used Microsoft Word and one Microsoft PowerPoint.

	Terrible	Frustrating	Dull	Difficult	Inadequate Power	Rigid
1						
2	fg	fg		f	g	g
3		e	efg	g		
4	e			e	e	f
5						
6						e
7					f	
	Wonderful	Satisfying	Stimulating	Easy	Adequate Power	Flexible

Table 10 Individual QUIS responses for general program users

Table 10 shows that there was little positive feeling about any aspect. Most agreed that their program was terrible, frustrating, dull and difficult. There were mixed opinions about adequacy of power and flexibility. These are reflected in comments about lack of power (line styles and curves) and control (e.g. editing angles).

These three users' not particularly positive ratings are matched by the generally negative ratings for General programs in Table 7. However, given the small number of users, any conclusive analysis is not possible.

3.5.5 CAD program

Only one user used this as their main diagramming program. The subject was neutral about

the level of power but felt the program was fairly rigid. However, he did praise two features:

- very accurate - can be used to check angles

- scale, move and copy facilities are easy

	Terrible	Frustrating	Dull	Difficult	Inadequate Power	Rigid
1						
2						d
3						
4					d	
5						
6						
7						
	Wonderful	Satisfying	Stimulating	Easy	Adequate Power	Flexible

Table 11 Individual QUIS responses for CAD user

Table 7 shows that five respondents have used CAD programs. These responses were mixed:

one slightly easy, two were neutral, one slightly difficult and one difficult. However, given

the small number of users, any conclusive analysis is not possible.

3.5.6 Bitmap painting programs

Only three users used this as their main diagramming program. The two that used Microsoft Paint gave the QUIS ratings summarised in Table 12:

	Terrible	Frustrating	Dull	Difficult	Inadequate Power	Rigid
1					b	
2	b					
3			B		a	a
4	a	a		b		b
5		b		a		
6						
7						
	Wonderful	Satisfying	Stimulating	Easy	Adequate Power	Flexible

Table 12 Individual QUIS responses for bitmap painting program users

Table 12 shows that there was no strong feeling about any particular aspect, only that Paint had inadequate power. The inadequate power is reflected in comments about the effort required to achieve basic goals e.g.

- There are no labour-saving tools, apart from being able to copy the previous step to act as the basis of the next step

- I have to draw every damned dot, dash and arrowhead myself.

This respondent also mentioned the problems of bitmapped graphics needing to be large in order to get the required level of detail. He mentions the knock-on problem of either using large uncompressed disk files or smaller JPEG files that can be blurry.

Table 7 shows that there was one other respondent who had used bitmap painting programs This respondent, and the two who responded in Table 12, were neutral about ease of making diagrams. However, given the small number of users, any conclusive analysis is not possible.

3.5.7 Specialist program

Only one user used this as their main diagramming program. Table 13 shows that this user was mildly positive about Microsoft Visio.

	Terrible	Frustrating	Dull	Difficult	Inadequate Power	Rigid
1						
2						
3						
4		i	i			i
5						
6	i			I	i	
7						
	Wonderful	Satisfying	Stimulating	Easy	Adequate Power	Flexible

Table 13 Individual QUIS responses for specialist program user

Table 7 shows that there are two other respondent who have used specialist programs: both were neutral. The respondent who used Visio as his main program rated it as easy to make origami diagrams. However, given the small number of users, any conclusive analysis is not possible

3.5.8 Programming

Only one user used this as their main diagramming program. Table 14 shows that the subject was neutral about terrible/wonderful, dull/stimulating and difficult/easy. This may be due to the lack of direct manipulation ("no freehand drawing" was a negative aspect).

He was very positive about the power and flexibility to produce diagrams in a satisfying way, commenting that the "special origami features" were the most positive aspect of his program.

	Terrible	Frustrating	Dull	Difficult	Inadequate Power	Rigid
1						
2						
3						
4	h		h	H		
5						
6						
7		h			h	h
	Wonderful	Satisfying	Stimulating	Easy	Adequate Power	Flexible

Table 14 Individual QUIS responses for programming user

3.6 Summary

This chapter described the design of the usability questionnaire and presented its results. The next chapter compares these results with my evaluation using the Cognitive Dimensions framework.

4   Evaluation using the Cognitive Dimensions Framework

Two evaluation methods were selected in Chapter 2, namely expert evaluation using the Cognitive Dimensions framework and a usability questionnaire. The previous chapter discussed the design and results of the usability questionnaire. This chapter describes the expert evaluation using the Cognitive Dimensions Framework which was introduced in section 2.6.4, p. 47.

I start by selecting benchmark tasks and software. I then describe the results of the expert evaluation, including a comparison with the outcomes of the usability questionnaire to see if the respondents agree with the expert evaluation.

4.1 Benchmark Tasks

A number of benchmark tasks were selected for evaluating the programs:

- Pureland origami (simple valley folds only): traditional cup

- Origami with reverse folds: traditional crane (a common benchmark task in the literature e.g. Fisher (1994) and Miyazaki *et al.* (1996))

- Huzita axioms: the 6 fundamental types of fold. There are 7 axioms if Huzita-Hatori axioms are used: Hatori (n.d); Lang (2005a).

- Individual folding procedures e.g. squash fold, rabbit ear fold, swivel, crimp, sink, etc

4.2 Benchmark Software

The following software was selected for evaluation:

- Macromedia Freehand

- Microsoft Word

- Cabri

- Doodle

- Miyazaki's origami simulator

- Nimoy's Java Origami

These are reasonably easily available programs. The initial usability questionnaire gathered opinions on this and other software.

The following sections describe each program. The programs are rated on each Cognitive Dimension described in section 4.3, p. 95.

4.2.1 Freehand – vector drawing program
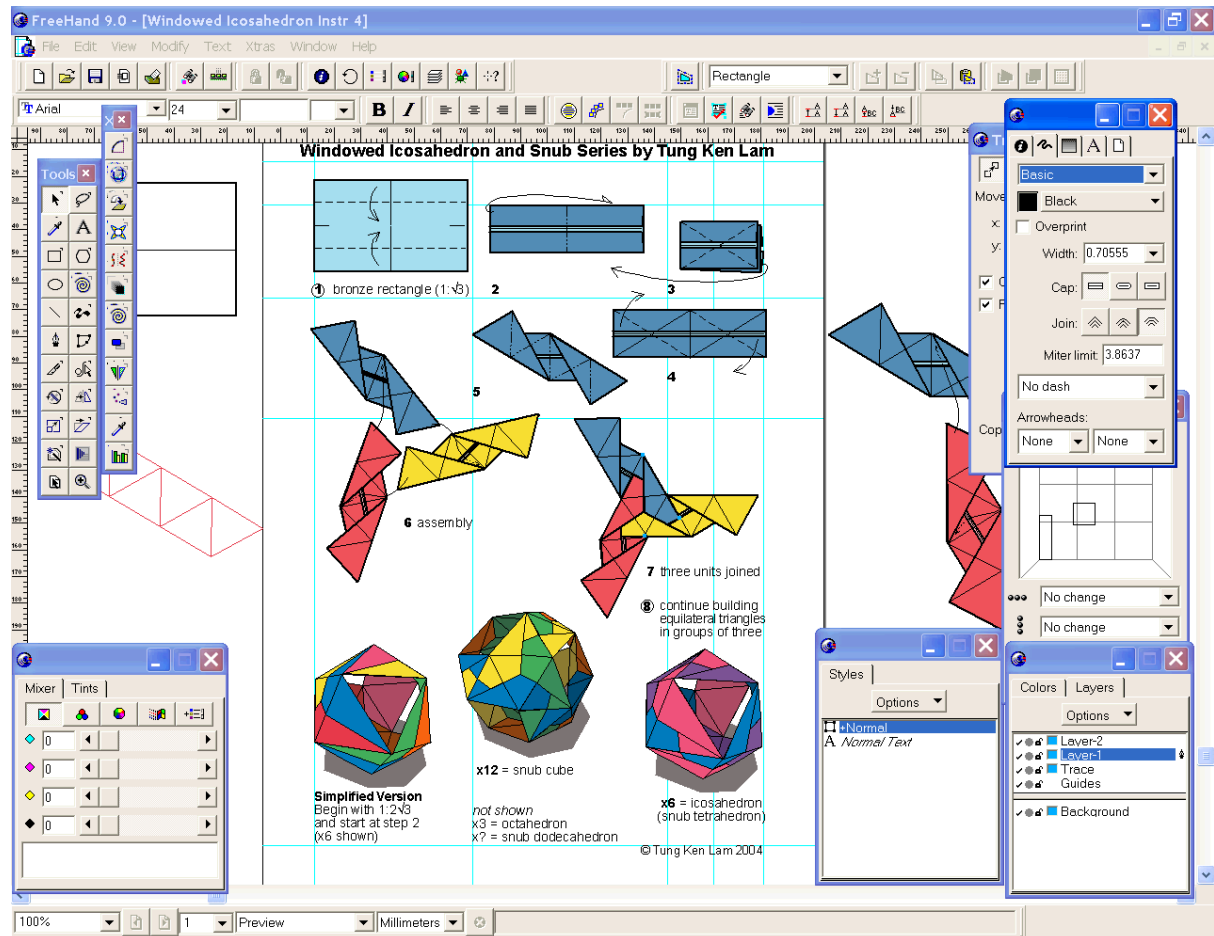
Figure 29 Screenshot of Macromedia Freehand 9.0 showing diagrams in preview mode

Macromedia Freehand is a vector drawing program for illustrators. Figure 29 shows a typical arrangement of windows, toolbars and palettes. Note the non-printing blue guidelines on the page for aligning objects and use of the off-page "pasteboard" for temporary storage of objects.

4.2.2 Word – general-purpose program
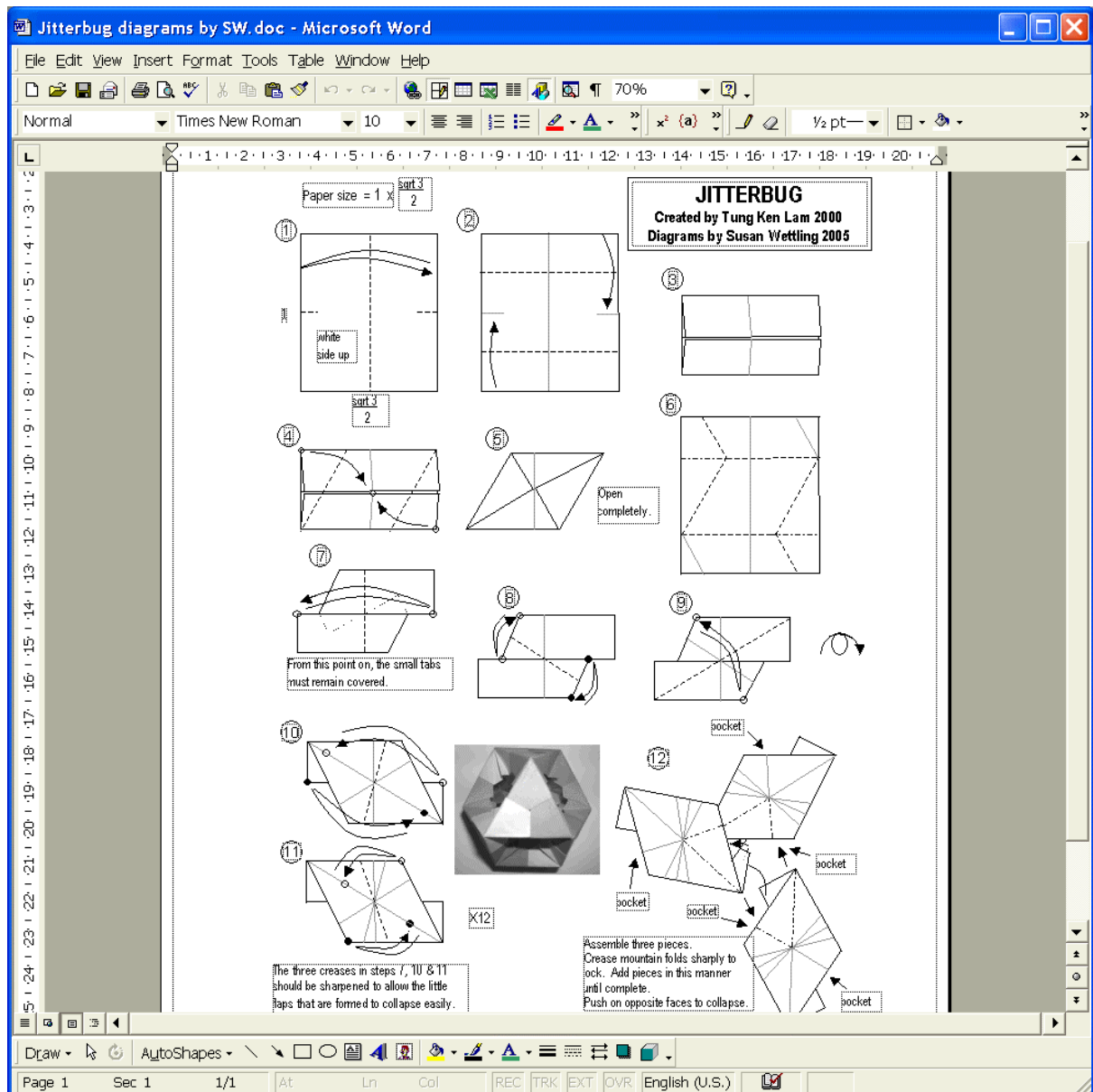
Figure 30 Screenshot of Microsoft Word 2000 showing diagrams created by Susan Wettling

Microsoft Word is a popular word processing program. It has evolved to offer features typically found in desktop-publishing software, including drawing tools. Figure 30 shows a set of diagrams created using combination of lines, text boxes and scanned images.

4.2.3 Miyazaki origami simulator
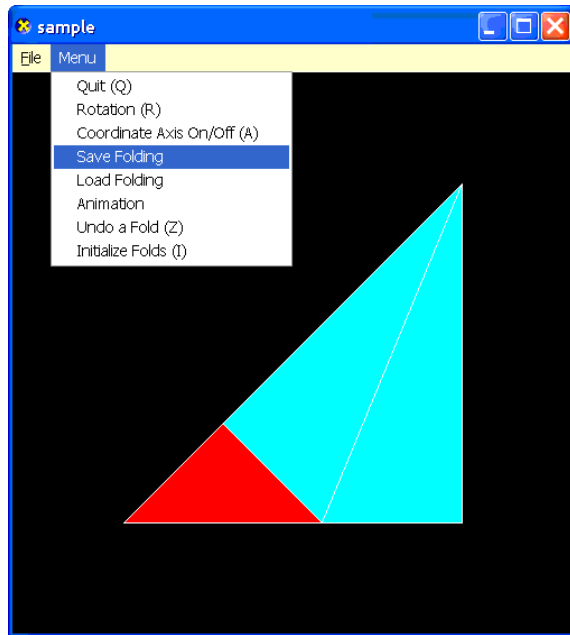
Figure 31 Screenshot of Miyazaki's origami simulator showing second step of traditional cup

Miyazaki has made his origami simulator available for Linux, OpenGL and Windows

DirectX platforms. Figure 31 shows the DirectX 9 version of Miyazaki's simulator.

4.2.4 Nimoy origami simulator
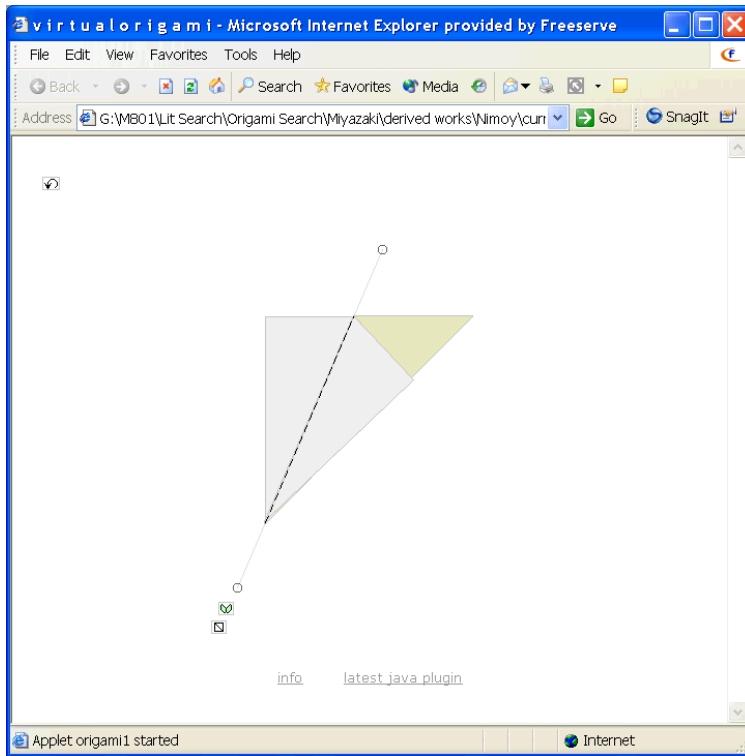
Figure 32 Nimoy's Java origami simulator showing second step of traditional cup

Figure 32 shows Nimoy's applet running in a web browser. As described in section 1.4.2, the user makes folds by dragging the fold line, orientating the fold arrow (thus setting the fold direction) and executing the fold. Figure 6, p. 24, shows another screenshot of Nimoy's simulator.

4.2.5 Cabri dynamic geometry software – mathematical
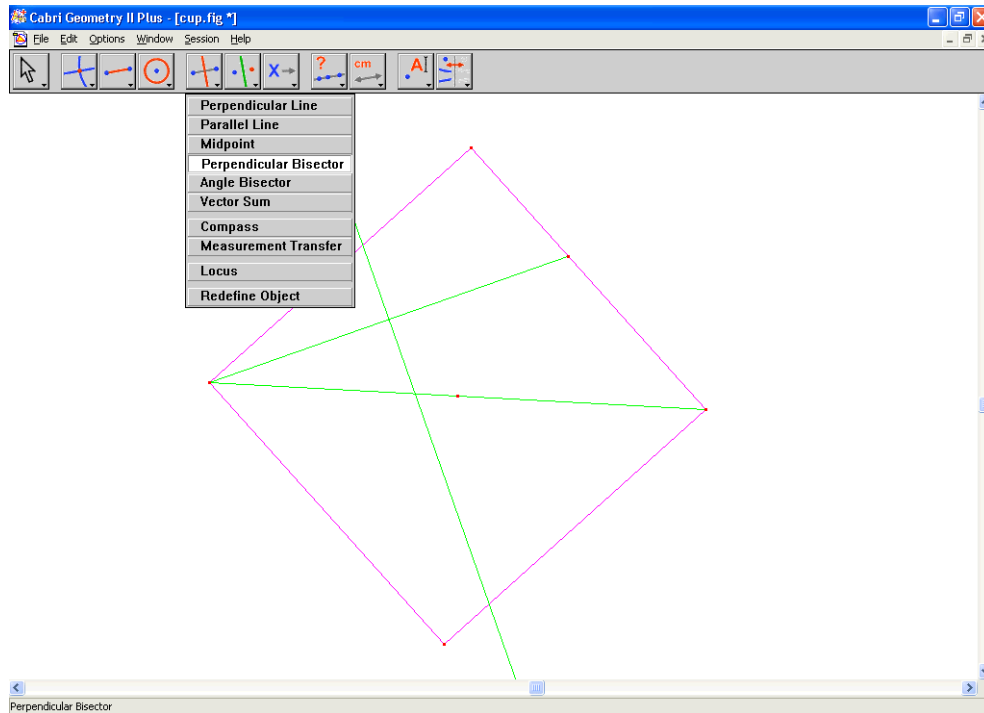
Figure 33 Screenshot of Cabri II Plus showing partial construction of traditional cup

Cabri is a dynamic geometry software package originally intended for teaching Euclidean geometry. Figure 33 shows the partial construction of the traditional cup in Cabri.

4.2.6 Doodle – origami-oriented programming language
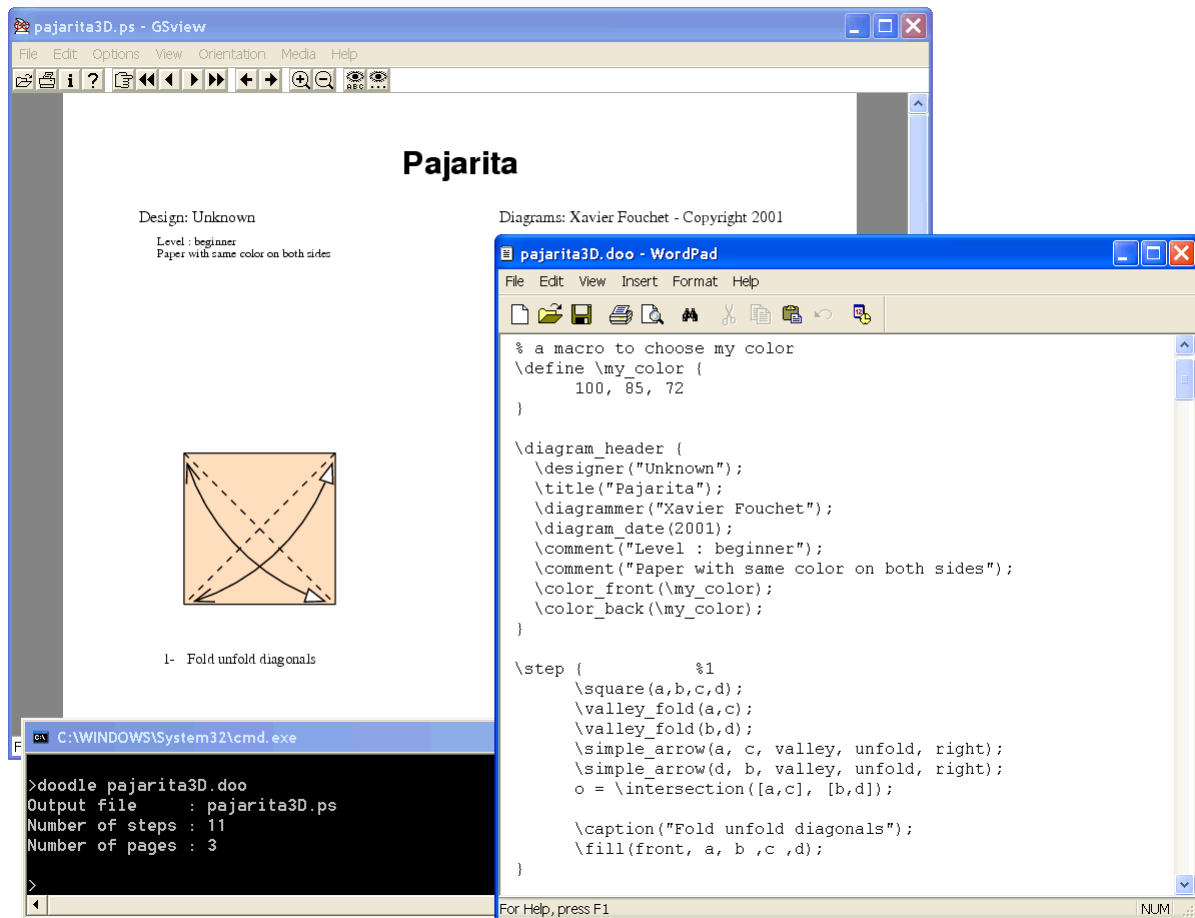
Figure 34 Typical arrangement of windows for working with a Doodle file (Xavier Fouchet's Pajarita)

Doodle is a programming language designed for producing origami diagrams. It compiles

source code into PostScript output. Figure 34 shows: top left – GSView showing Doodle

PostScript output; bottom left – command line prompt for compiling Doodle source; bottom

right – text editor for editing Doodle source. Note that the source does not specify coordinates

numerically. Figure 7, p. 25, shows another example of Doodle output which uses the

debug_point command to annotate vertices.

4.3 Evaluation using the Cognitive Dimension framework

			Freehand	Word	Miyazaki	Nimoy	Cabri	Doodle
Abstraction	1.	ABSCanDefine	1	3	5	5	2	3
	2.	ABSMustDefine	5	5	4	5	3	2
Hidden dependencies	3.	HIDDVis	4	4	5	5	4	3
	4.	HIDDSize	3	4	3	4	2	1
Premature commitment	5.	PREMFreeOrder	1	2	5	5	4	4
	6.	PREMMustPlan	4	3	2	4	4	1
Secondary notation	7.	SECN	1	1	4	5	4	2
Viscosity	8.	VISCEasyChange	1	1	4	4	3	4
	9.	VISCImportantDifficult	3	2	2	4	4	1
Visibility	10.	VISEasyFind	2	1	NA	NA	2	4
	11.	VISJuxtaposable	1	1	NA	NA	5	2
Closeness of mapping	12.	CLOSEMap	2	3	4	2	4	2
	13.	CLOSEStrange	4	2	3	4	NA	2
Consistency	14.	CONS	1	2	2	2	3	2
Diffuseness	15.	DIFF	3	3	4	3	3	3
Error-proneness	16.	ERREasyMistakes	4	4	3	2	3	1
	17.	ERROftenSlips	2	4	2	2	3	1
Hard mental operations	18.	HARDNeed	4	3	3	4	2	1
	19.	HARDEffort	2	3	4	4	2	3
Progressive evaluation	20.	PROGEasy	1	1	NA	NA	NA	4
	21.	PROGAnyTime	1	1	NA	NA	NA	1
	22.	PROGPartial	1	1	NA	NA	NA	1
Provisionality	23.	PROV	1	2	2	2	2	4
Role-expressiveness	24.	ROLE	1	2	3	3	3	4

NA = Not Applicable; 1 = Strongly Agree; 3 = Neither Agree nor Disagree; 5 = Strongly Disagree

Table 15 Summary of evaluation of benchmark program using the Cognitive Dimensions framework

(positive aspect; negative aspect)

NA = Not Applicable; 1 = Strongly Agree; 3 = Neither Agree nor Disagree; 5 = Strongly Disagree

Table 15 summarises the results of the expert evaluation using the Cognitive Dimensions framework. The next sections discuss the evaluation of each Cognitive Dimension.

4.3.1 Abstraction

Abstraction-tolerant programs typically allow users to define styles for frequently used sets of attributes and macros for common repetitive tasks. *Abstract-hungry* programs require users to learn a significant number of abstractions before they can get started. Abstractions are not inherently good or bad: they can make some tasks easier, but at a cost (potentially more hidden dependencies, viscosity and increased need for lookahead).

Freehand allows the user to define graphic styles, but does not require them to do so (it is *abstract-tolerant*). Furthermore, users can define their own colours, arrowheads and line styles. Word is also *abstract-tolerant*, but only for text styles. However, Word does allow users to both record and edit macros. Cabri allows users to define macros by simply telling it the inputs objects for a macro and the required output objects. Cabri does not have styles for graphic objects i.e. the user cannot define a "mountain fold" style, say 1 point line thickness with a dash-dot-dot line dashing attribute.

Since Doodle is a programming language, it lets users choose names for edges and vertices. It is possible for users to define their own macros. The user must learn a moderately-sized set of built-in commands and syntax in order to be productive in Doodle: Doodle is relatively *abstract-hungry*.

The other programs are not particularly *abstract-hungry* except Cabri, which requires the user to construct all objects mathematically (unless macros exist). Nimoy allows no user

customisation at all. Miyazaki has little user customisation except for allowing different paper shapes and colours. However, this is a little convoluted as the user must edit a text file to set up the paper when the user wants non-default values (e.g. coordinates of vertices for defining paper shapes; *rgb* values for paper colours).

4.3.2 Hidden dependencies

None of the programs suffer particularly from hidden dependencies. Programs with few abstractions have correspondingly few hidden dependencies (Nimoy and Miyazaki). Programs with more abstractions have more potential for hidden dependencies (Word, Cabri and Freehand).

Doodle has the most potential for hidden dependencies: for example, Doodle allows vertices to be displayed at different coordinates to their actual coordinates with the *shift* command. This is intended to help users separate layers in a diagram step. However, moving a vertex by redefining its location, may, or may not, have the intended consequence. Also, it is difficult to find out which vertices have had a shift applied. Consequently, hidden dependencies become worse as the document grows larger. This problem applies to Freehand to a lesser extent (e.g. if similar colours have been defined, what is the effect of changing a particular defined colour?)

4.3.3 Premature commitment

Freehand and Word are the most flexible programs as they impose few constraints on the order of doing things. The worst are Nimoy and Miyazaki, because the user must always start at the beginning and fold on screen in the same order as the real paper folding. However, these constraints are no worse than that of real paper-folding. Cabri and Doodle also require

the user to start from the beginning and develop the diagrams in order. However, the user can choose to start at an intermediate step, if needed.

Doodle requires careful planning in order for the user to be successful. For example, a good choice of vertex names helps define edge names. Poor choices make later steps harder, and if they need to be changed, are difficult to propagate throughout the Doodle source (high *viscosity*). To a lesser extent, Miyazaki needs careful planning because certain folds need to be re-sequenced for its limited simulation capabilities. For example, a petal fold cannot be made in the traditional manner – it must be made as two reverse folds and one valley fold.

4.3.4 Secondary notation

Word allows the user to insert non-printing text and comments. Freehand allows comments and temporary objects to be stored away from the page on the paste-board.

Since Doodle is relatively advanced programming language, it allows users to indent, add whitespace and comments in flexible manner. If the Doodle source is edited in a syntax-highlighting text editor, further support for secondary notation is easily achieved.

Cabri and the origami simulators have little support for secondary notation.

4.3.5 Viscosity

Both Word and Freehand allow users to easily make changes to previous work. This is partly due to their direct manipulation interfaces and lack of premature commitment. Graphic elements can easily be repositioned and reformatted. However, Word is slightly more viscous than Freehand because Word lacks the ability to define and change graphic styles. Also, Word lacks a search and replace facility for graphic styles.

Cabri suffers from *premature commitment* more than Word and Freehand and therefore is slightly more *viscous*. It also can be long-winded to make certain changes in Cabri: it depends on how the construction was made.

Both of the origami simulators and Doodle require users to start from the first step and make each step in sequence. Therefore it is difficult to take an existing model and adapt it into a new one. The origami simulators suffer from the fact that some fold cannot be unfolded, even though it should be possible. Furthermore, if part of the paper is hidden, it is not possible to manipulate it at all. This means users need to re-sequence folds in order to prevent hiding the paper of interest. Sometimes no amount re-sequencing helps: it is impossible to make the required fold. Nimoy suffers from another problem in that it lacks the ability to save folding and restore previous work.

4.3.6 Visibility

Word and Freehand both allow the users to easily scroll and zoom around their documents. They also allow users to open multiple views of the same document which are in separate windows. Word has the convenient extra feature of allowing the user to split a view within the same window.

Cabri has fewer features for zooming but has the usual facilities scrolling. Cabri only keeps one view of the document, so juxtaposability is difficult.

Doodle is a compiled programming language so visibility and juxtaposability depend on how the user works on his or her computer. Figure 34, p. 94, shows a typical arrangement of programs. This shows that juxtaposability can be good, but it can be difficult to find the parts

of the diagram that are of interest due to hidden dependencies and the fact that visible parts are indirectly manipulated via the source code. Doodle provides a *debug_point* command which helps by labelling parts in the PostScript output.

The origami simulators only work with a single document and a single view. Miyazaki provides a number of commands for manipulating the camera viewpoint, but Nimoy has none.

4.3.7 Closeness of mapping

If the domain is taken to be drawing, then Freehand and, to a lesser extent, Word, map closely to the domain of origami diagramming. If the domain is taken to be origami (i.e. the folding of paper) then Nimoy and Doodle also map closely to the domain of origami diagramming. Nimoy supports valley and mountain fold lines and arrows, but has no support for multiple steps. It can only make a fold through two points directly –other fold types must be judged by eye. Doodle has a number of features intended to support origami diagramming – e.g. hidden coordinate shifts to show layers; line shortening; automatic, built-in step numbering and layout.

Cabri and Miyazaki do not map well to the domain of origami diagramming. Cabri maps well to the domain of geometrical construction (some represent folding operations e.g. angle bisectors and mirror lines). However, Cabri does not map well to the domain of origami diagramming because it does not support multiple steps in the same document.

Like Cabri, Miyazaki cannot support multiple steps visible at the same time. Furthermore, Miyazaki does not use any conventional diagramming symbols at all. Miyazaki may map closely to the domain of folding paper, but not this is not the same as diagramming.

Programs that lack specific origami support tend to fare badly when considering the statement: "There are parts of the program which seem particularly strange for origami diagramming." Origami support can be divided into two kinds: support for folding and support for diagramming. Thus Doodle supports diagramming, and to a lesser extent, folding, and rates quite well on this aspect. On the other hand, Miyazaki supports folding, but has no support for diagramming.

4.3.8 Consistency

All programs are consistent, to a greater or lesser extent. Those using direct manipulation interfaces are generally consistent. For example, all folds are performed in the same way in Miyazaki – pick up the paper, move it and drop it to make a fold. Cabri suffers slightly because the order of parameters for different commands does not always seem to be logical nor consistent. (However, some geometers might not find this a problem because Cabri follows the conventions of Euclidean geometry.)

4.3.9 Diffuseness

No program is particular diffuse, nor particularly terse. Miyazaki can be sometimes be long-winded because every fold must be made – there are no repeat operations. However, this verbosity is no worse than real paper-folding. Nimoy and Miyazaki both need augmenting with separate software in order to layout steps on a page with numbering and captions. Doodle is a relatively terse language, but Doodle source can become large even for a small project.

4.3.10   Error-proneness

Doodle can be confusing to use due to its abstractions and hidden dependencies. A dedicated Doodle development environment could reduce mistakes. For example, modern integrated development environments recognise command names as they are typed and can automatically complete them and show the command's required parameters – this would reduce demands on memory.

Nimoy, unlike Miyazaki, has no support for snapping the cursor to important locations, and hence accurate folding is difficult. However, both origami simulators can demand that folding sequences are restructured and this can be error-prone.

Programs that need intensive mouse control can invite *slips*. For example, in the direct manipulation programs: Word, Cabri and Freehand, mouse control is sometimes difficult when selecting objects that overlap or are very close together. For example, it can be troublesome in Freehand to select a Bézier control point because it is too small to click (although a user can customise control point sizes). In Cabri, failing to select the correct object can force the user to re-enter a command. For example, the command for bisecting an angle requires three points: if the third point is not successfully selected, the user must enter the first two points again before trying to enter the third point again.

Doodle, like other compiled programming languages, can penalise the user for mistyping a statement, or forgetting syntax elements like */* or *;* , or failing to match braces *{ }*. Doodle only informs the user if these are wrong when it compiles the source. As mentioned above, this could be prevented, to an extent, by using a text editor that checks syntax as it is typed.

4.3.11   Hard mental operations

Since Doodle is a compiled programming language, it uses high-level abstractions and this can place heavy demands on cognitive and memory resources. Figure 7, p. 25, shows that folding one corner over requires five commands, each of which requires three or more parameters.

Cabri can make demands on cognitive resources – this may be deliberate because of its original intention as a tool for learning Euclidean geometry!

4.3.12   Progressive evaluation

Word and Freehand offer good opportunities to check the work in progress at any time. However, Doodle requires source to be compiled before checking. Thus, the Doodle user must switch to the command line, enter the command for compilation, and then switch to the PostScript viewer to check the output. (GSView can ease this process slightly as it automatically opens a PostScript file if it has been updated by another program. It even uses the same zoom settings and location when the file is reopened.)

Programs with no built-in support for multiple steps allow progressive evaluation only for the current step – they have no support for checking the final layout of steps on a page because this must be done in a separate program.

4.3.13   Provisionality

In general, all programs have good provisionality except Doodle due to its high cognitive demands.

4.3.14 Role-expressiveness

Freehand rates the best in role-expressiveness, closely followed by Word.

Nimoy does not score highly here because the symbols for mountain/valley and "fold it" have no explanation. However, once learned, they are not hard to remember. More positively, the symbols for fold line and arrow have the same meaning as for standard origami diagram notation.

Cabri requires mathematical and geometrical knowledge in order to understand it.

Miyazaki uses a direct manipulation interface which gives almost instant feedback. The use of the "T" key for inside reverse fold (tuck folds) shows an effect on-screen. However, the use of the "M" and "N" keys for peeling and unpeeling paper does not seem to be a natural mapping.

Doodle has some commands that are misnomers. For example */fold* does not fold a flap, it draws a fold line that is a line of reflection. Some command names are hard to remember – the order and meaning of the parameters of some functions are not easily memorised either. Perhaps these issues only affect novice Doodle users.

4.4 Results

In general, the questionnaire outcomes matched the expert evaluation. Vector drawing programs were by far the best type of program. General programs such as Microsoft Word

were close behind. Cabri was neutral but Nimoy's origami and Miyazaki's simulator were the worst.

Some subjects indicated that part of the difficulty of diagramming is not necessarily the software but the task of structuring a logical, clear and enjoyable folding sequence. These difficulties are beyond the scope of this project, but some could be dealt with: for example, one respondent requested a "fold checker" that operated like a "spell checker".

4.5 Summary

This chapter described the benchmark software and tasks selected. Expert evaluation using the Cognitive Dimensions framework showed that vector drawing programs were by far the best type of program. Nimoy's origami and Miyazaki's simulator were the worst, mainly because they were not designed to produce instructions for the printed page.

These evaluation findings inform the design of an improved interface that is proposed in the next chapter.

5 The Proposed Improved Interface

This chapter describes a redesigned interface and the rationale for its redesign, based on an existing computer origami simulation. The previous chapter found that origami simulations were the worst type of software for making printed origami instructions. Despite this unpromising starting point, adapting an existing origami simulation will address the main reason why origami simulations fared badly: they lacked features for producing diagrams laid out on a page. I consider the importance of each design change together with the feasibility of implementation.

This chapter starts by considering the potential pitfalls of redesigning interfaces.

5.1 Radical Solutions?

Whittaker *et al.* (2002, p. 79) warn against proposing "radical solutions to things that users do not consider to be major problems and [radical solutions] can neglect major problems that users do experience." The authors believe that the use of reference tasks help prevent this. They cite the example of the DARPA (Defense Advanced Research Projects Agency) "bakeoffs" that did create genuine progress in speech recognition. They give the following criteria for selecting reference tasks (p. 87):

- frequent
- critical
- real (research should establish real tasks from real users)
- not likely to become obsolete

Therefore the next section looks at evidence from real users.

5.2 Problems indicated by the evaluation

Although the use of computers has made some stages of diagramming easier, not all stages have been made easier. Comparing Table 16 and Table 17 shows that arranging a final layout is much easier, but the ease of the first three diagramming stages has not substantially improved.

Ease of diagram-ming stage		Sketch / plan a draft sequence	Draw step outlines	Add fold lines, arrows, number and text	Arrange a final layout	Other
Easy	1	bfjns	l	gijlt	gls	
	2	adgilo	dij	cdopsw	djp	
Neither	3	epwx	abmosuw$	abkmx	kw$	
	4	cmu$	efgptxy	euy$	abemx	j
Difficult	5	kt	k			
Total responses		21	20	15	15	1

Table 16 Individual responses for ease of diagramming stage for conventional methods

Ease of diagram-ming stage		Sketch / plan a draft sequence	Draw step outlines	Add fold lines, arrows, number and text	Arrange a final layout	Other
Easy	1	bl	ln	ghilz	bghlopt	
	2	gjmrz	jz	jmopt	ijmz	
Neither	3	eikps	himoru	br	cn	
	4	u	bcegst	cenu	er	o
Difficult	5	cnt	p			
Total responses		16	17	16	15	1

Table 17 Individual responses for ease of diagramming stage for computer methods

When respondents have estimated the percentage of time they spend on the diagramming stages, there is little difference between conventional and computer methods. The most time-consuming stage is drawing step outlines: approximately 50% of time is spent on this stage (mean average).

Therefore any attempt to reduce the time taken to draw step outlines will have the most beneficial impact.

5.3 Problems with Miyazaki's origami simulator

5.3.1 Menu

As mentioned in section 2.6, Microsoft Windows has a recommended style. For example, users should expect most applications to have a File and Edit menu. (Petzold, 1992, p. 355). Microsoft's *The Windows User Experience* (Microsoft Corporation, 2004b) suggests that the File menu "provides an interface for the primary operations applied to a file... [it] should include commands such as New, Open, Save, Send To, and Print." Figure 35 shows that Miyazaki's simulator does have a File menu, but all other menu items have been placed into the "Menu" menu. This is likely to be for reasons of portability between the Windows DirectX, OpenGL and Linux versions. Figure 36, p. 109, shows part of the redesigned menu. Note that

- Menu items have a letter underlined to enable Alt key combinations e.g. pressing the sequence Alt, F, A invokes <u>F</u>ile – Save <u>A</u>s...

- Menu accelerators are defined where appropriate and are displayed right-justified in the menu item e.g. Ctrl+S invokes the File – Save menu item.
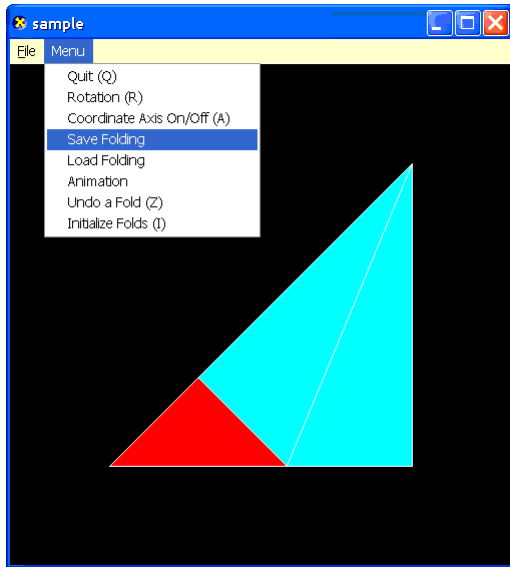
Figure 35 Screenshot of Miyazaki's origami simulator showing the "Menu" menu
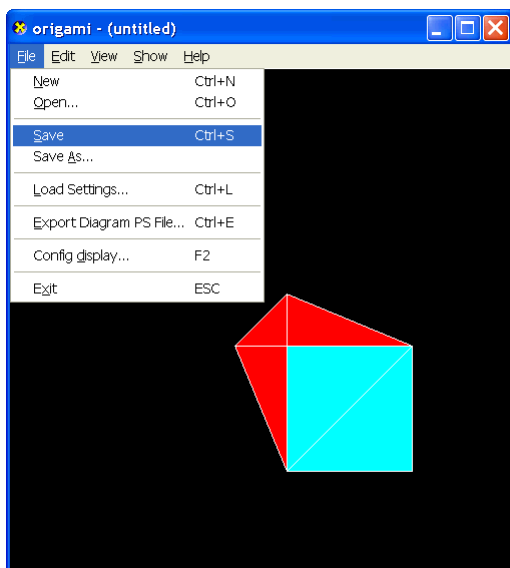
Figure 36 Screenshot of redesigned origami simulator showing the new "File" menu

Table 21, p. 180-182, defines the full menu structure. Items in the original "Menu" menu have been reorganised into the File, Edit, View and Show menus. The Edit menu follows Microsoft Corporation's (2004b) recommendation that it includes an Undo command. The

View menu has commands "that change the user's view of data in the window. .... [the] commands on this menu that affect the view and not the data itself." (*ibid.*)

The new Help menu has "commands that provide user access to Help information. ... [the] Help Topics command .... provides access to the HTML Help Viewer, which displays topics included in your application's Help file" (*ibid.*)

A number of keyboard commands did not have menu items e.g. commands for manipulating the camera and peeling/unpeeling paper. New menu items have been defined for these commands in order to decrease the user's need to memorise these commands.

By following Microsoft's suggestions for menus, the redesigned menu addresses Nielsen's fourth heuristic (NH4: "Consistency and standards ... Follow platform conventions and accepted standards") and Norman's seventh principle (N7: When all else fails, standardise) and Shneiderman's first golden rule (S1: Strive for consistency).

By reducing the user's need to memorise keyboard commands, the menus address Norman's first principle (N1: "Use both knowledge in the world and knowledge in the head.") and Nielsen's sixth heuristic (NH6: Recognition rather than recall.)

5.3.2 Document-centric operation

The original simulator always saved to, and loaded folding from, a file called "default.ori". If a user wished to keep more than one folding sequence, he or she needed to copy and/or rename files outside of the simulator. Therefore standard File menu items for opening and saving files have been added and the functionality implemented.

The original simulator always loaded settings from the "origami.txt" file. If a user wished to work with a number of different settings, e.g. paper shapes or colours, he or she needed to copy and/or rename files outside of the simulator. The File – Load Settings menu allows the user to choose a settings file without having to rename it and restart the program.

Both of the above changes address Norman's second principle (N2: "Simplify the structure of tasks. Tasks need to be simple in order to avoid complex problem solving and excessive memory load ....[one] approach to simplification is to change the nature of the task so that it becomes something more simple.")

5.3.3 Exporting PostScript diagrams

This functionality was implemented to a basic level. The prototype uses a simplified version of Doodle's (Gout, 2001) technique for automatically laying out steps on a page. It is simplified because page and margin definitions are hard-coded and each step is the same size. Step numbers start at one and increment by one for each subsequent step.

The orientation of the folding is used for making each diagram step: this is achieved by defining the PostScript Current Transformation Matrix to be the same as the DirectX Model-View matrix. However, this lacks perspective projection, and hence the user's zoom setting is ignored.

The prototype keeps track of camera view for each step both before and after a fold is made. This allows the program to draw extra steps if the orientation changes between folds. This is to support Lang's seventh principle of always showing the result of a fold (L7: Don't leave the reader dangling).

Each fold is represented by drawing an arrow from the picked point to the moved point. As the program constructs each step dynamically, the exporting processing hooks into the animation process. Unfortunately fold lines were too difficult to extract.

Reverse folds are indicated by the caption "Reverse fold." The ideal solution is to draw the standard symbols for an inside reverse fold (first diagram in second row of Figure 40, p. 209).

Colours for paper regions are taken from the program settings. This supports the principle of distinguishing each side of the paper using shading (question 15, initial questionnaire)

5.3.4 Documentation

Nielsen's tenth heuristic (see Appendix B – Nielsen's ten heuristics for Heuristic Evaluation, p.134, NH10) states that most systems need instructions for use: they should "not be too large". See Appendix H – Help file for redesigned origami simulator, p. 169, for the help file that was developed. The "How do I...?" section follows the recommendations of Nielsen's tenth heuristic of being "focused on the user's task … [and listing]… concrete steps to be carried out."

6 Implementation of Improved Interface

This chapter describes how the improved interface was implemented. It lists the software used, details the changes made and concludes with a list of changes attempted but not achieved.

6.1 Tools

The following software was used to implement the prototype

- Bloodshed Dev-C++ 4.9.9.0[3]

- Microsoft DirectX 9 SDK[4]

- MinGW tools[5]

- Open Watcom 1.3[6]

- S. Miyazaki Virtual Origami source code for DirectX version (Miyazaki, 2004)

Programming utilities and tools used included:

- AFPL Ghostscript 8.14 (2004-02-20) (File – Convert... allows conversion of PS to PDF)[7]

- GSview 4.6 (2004-01-11)[8]

- Seapine Surround SCM[9]

- Textpad[10]

---

[3] http://www.bloodshed.net/devcpp.html

[4] http://msdn.microsoft.com/directx/sdk/

[5] www.mingw.org/

[6] http://www.openwatcom.org/

[7] www.ghostscript.com

[8] http://www.cs.wisc.edu/~ghost/gsview/

[9] www.seapine.com/surroundscm.html

[10] www.textpad.com/

6.2 Existing implementation

The original source code for the origami simulator was relatively clean. Although there were only a few comments, most of the variables and functions had meaningful names and overall the structure was good with sensible modularisation.

Furthermore, Miyazaki *et al.* (1996) gives a good overview of the data structures for face groups, faces, edges and vertices. Mapping these to the source code is relatively straightforward, for example COrigami contains an array of CStage objects. Each CStage represents one step of the folding sequence and stores a CFaceGroup object. This CFaceGroup object maps to the "Facecell tree" (*ibid.*, p. 32).

6.2.1 Structure

Miyazaki *et al.* (1996) originally wrote their code for Silicon Graphics IRIS Crimson with Reality Engine. Miyazaki (2004) then ported OpenGL versions to Linux and Windows and created a DirectX version (Windows only). The DirectX version appears to have been initially created by the Visual C++ AppWizard.

The multi-platform nature of the code explains its slightly unconventional arrangement of source code files. For example, Figure 37 shows that user.cpp includes "mainx.cpp" which then includes further "header files", namely, vertex.h, edge.h, etc. However, the latter "header files" have no associated ".cpp" files that implement the classes and function prototypes defined by the header file. These header files in fact contain both the definition and the implementation of classes and functions. The makefile for Dev-C++ must be altered to reflect this arrangement, otherwise files might not be compiled when they are edited.

this file	includes these files...	which then includes these files...
user.cpp	DXUtil.h D3DApp.h D3Dutil.h resource.h user.h	
	og.h	geometry.h bitmap.h time.h render.h
	mainx.cpp	vertex.h edge.h face.h facegroup.h stage.h origami.h

Figure 37 File structure of Miyaki's DirectX origami simulation

6.3 Changes made

The implementation consisted of the following steps:

1. Extract vertex, edge and face information and create a set of 2D overlapping polygons

    for exporting to a 2D format, namely Postscript

2. Layout the steps on pages with numbering and captions

The program saves each step (maximum 255 steps). Each step holds the source and destination point of a fold, and the type of fold e.g. "fold up" or "tuck in". (The program dynamically generates the required edges, vertices and faces from this information). Therefore each step can be drawn and an arrow and fold line placed to show the fold instruction.

1. When the folding is finished, the user exports the diagrams using the menu. The output file consists of each step laid out on a page. The layout is automatically computed in a similar way to Doodle. Each step is labelled with a number and automatically generated text, if any (e.g. "reverse fold").

2. The user can view the output, convert it to another format or import and edit it (using external applications like GhostScript and GSView).

The user can alter their viewpoint and orientation. This needs to be taken into account when generating the diagram of a step so that they have the same viewpoint (e.g. user rotates model, turns over or zooms in/out). The camera viewpoint at the time the user completes a fold will be used i.e. at the time when the user releases the mouse button.

Full source code is provided on the CD-ROM (see Appendix M – Contents of CD-ROM, p. 192).

6.4 Postscript

I have decided to use PostScript, a page description language (Foley *et al.*, 1997, p 999), for writing diagrams.

PostScript is essentially like assembly programming – it is powerful, but

programming in it is not too pleasant. On the other hand, defining procedures is

simple, which makes PostScript far easier to use. (Foley *et al.*, 1997, p 1003)

Although PostScript is a low-level language, and doesn't support high level naming and

structuring of objects, it is well established, well documented (Adobe Systems Incorporated,

1992, 1999), can be imported by many applications, and easily convertible to PDF[11].

6.5 Conclusions

A number of ideas for improvement were abandoned due to implementation difficulties:

- Showing multiple layers. Potential algorithms may involve displacing vertices that are
  free to move like Doodle's *\shift* command. The displacement could either be in three-
  dimensions or two dimensions.

- Drawing mountain and fold lines. This information is not straightforward to extract
  from data structures. A simpler solution could be to draw the perpendicular bisector of
  the fold arrows' start and end points.

- Drawing standard symbols for reverse folds. The difficulty of this is related to the
  difficulty of drawing fold lines.

- Drawing fold steps with perspective projection. The diagrams use the orientation at
  the time each fold is made by applying the Model-View matrix to the PostScript
  Current Transformation Matrix (CTM). However, the diagrams can be different to the
  screen folding if the zoom is different.

---

[11] e.g. www.ps2pdf.com

Only PostScript output was implemented due to time constraints. Other formats such as WMF or 3D model formats like VRML, X3D or AutoCAD were not implemented.

Despite these problems, will the developments that were implemented sufficiently improve the task of diagramming with a computer?

7 Evaluation of Improved Interface

The previous two chapters outlined the design and implementation of an improved interface for making origami diagrams. This prototype implementation is evaluated in this chapter by subjects using a usability questionnaire drawn from relevant sections of the initial questionnaire. (It is possible to compare ratings for the same person if they have responded to both questionnaires.)

Subjects' ratings for the Cognitive Dimensions are compared with the expert evaluation: has the prototype improved the intended dimensions?

7.1 Usability Questionnaire

7.1.1 Distribution

An announcement to potential subjects was made by email to the same mailing lists as the initial questionnaire (namely origami-l, BOSmail and paperwonders). The announcement directed subjects to a website where they could download the software, documentation and questionnaire in a number of formats. The website is reproduced on the CD-ROM (see Appendix M – Contents of CD-ROM, p. 192, for details). Appendix I – Usability of Prototype Questionnaire, p. 184, shows the usability questionnaire held on the web site.

In the week following the announcement, several users reported a problem with the program that made it crash. Thanks to the report of one user, I managed to fix this problem in a subsequent version released to the web site. However, as some of the subjects were evaluating earlier versions some results below refer to this problem i.e. the program did not remember the settings for vertex processing ("software vp").

7.1.2 QUIS Results

See Appendix L – Results of Usability of Prototype Questionnaire, p. 191 for results. Of the nine subjects that responded, only three completed the questionnaire in full. Only two of the subjects had responded to the earlier initial questionnaire.

The small sample size means that statistical analysis is neither feasible nor meaningful – any conclusions drawn are relevant only to the context and are not necessarily generalisable. Nevertheless, the data can be regarded as a qualitative evaluation of the prototype. Table 18 shows the QUIS for individual subjects.

- All subjects agreed that the prototype was "stimulating" (not "dull"). The majority gave the rating "wonderful" but some gave the rating "terrible".

- There were mixed responses for "Difficult/Easy" and "Frustrating/Satisfying".

- The worst ratings were given for power and flexibility.

	Terrible	Frustrating	Dull	Difficult	Inadequate Power	Rigid
1						
2		h		gh	ah	b
3	bh	ab		bd	cd	cdgh
4		cd			bei	ai
5	cdegi	e	bcdi	f	f	ef
6	A	fi	egh	cei		
7	F		af	a		
	Wonderful	Satisfying	Stimulating	Easy	Adequate Power	Flexible

Table 18 Individual QUIS responses for the prototype

The wonderful rating is echoed by subjects' comments for positive aspects:

- Animation (four subjects)

- Snapping / accuracy of folding (two subjects)

"Wow" factor of simulation (three subjects) e.g. "great and amazing that it's possible to program stuff like this!" (subject c)

The mixed response for ease of use and the extent to which the prototype was satisfying is echoed by the conflicting comments from subjects. The positive comments:

- Easy to use

- Accuracy of folding

- Seems obvious what to do

- Difficult/Easy: especially when the program recognises the line you're folding to

were contradicted by negative comments:

- The snapping to a point makes it impossible to get to certain other points of paper;   The "unfold" procedure is difficult

- Difficult to undo things

- Difficult to manipulate with mouse

Comments supporting "Inadequate Power" include:

- Inability to do complex folds

- Folding virtual paper is restrictive [may be talking about inability to make some 3d folds e.g. in traditional masu]

- Does not cope with more than basic folds (as detailed in email correspondence.)

- No mountain fold possible.

- No signs for "turn over" on .ps output.

Some subjects noted problems not directly related to the simulation and diagramming aspects of the prototype:

- Because of the bugs and software vp problem which I know is now fixed

- Settings don't stay

- Crashes a lot

- It seemed to draw a lot of processing power

Two users found manipulating the model in three dimensions difficult:

- I was not able to move to whole model in the screen

- Difficult to turn objects to three dimensions

This echoes the statement by Dix *et al.* (2005, p.144):

> We are good at moving objects around with our hands in three dimensions, rotating, turning them on their side. However, we walk around in two dimensions and do not fly. Not surprisingly, people find it hard to visualise and control movements in three dimensions.

Intriguingly, one subject reported the program produced

- Neat diagrams

but two other subjects stated that

- Diagrams are ugly

- It is visually not attractive

Clearly, visual attractiveness is subjective, but this shows that no single style of diagramming can be considered "perfect".

7.1.3 Cognitive dimensions results

Only three subjects gave ratings for the cognitive dimensions. This small sample size means that conclusive analysis is not possible.

The prototype was criticised on these dimensions

- premature commitment (PREMMustPlan)

- the need for hard mental operations (HARDNeed)

and to a lesser extent

- viscosity - difficult of changing folding and/or diagrams (VISCEasyChange)

- hidden dependencies worsening with larger documents (HIDDSize)

These echo the mediocre QUIS ratings for adequacy of power, flexibility and ease of use and ease of learning.

However, the prototype was praised on these dimensions:

- visibility (HIDDVis)

- ability to use secondary notation (SECN)

- ability to evaluate work at any time (PROGAnyTime)

and to a lesser extent

- ability to make diagrams reasonably briefly (DIFF)

- ability to evaluate partially-completed work (PROGPartial)

- ability to sketch things out and play with ideas (PROV)

7.1.4 Comparison of QUIS ratings between initial and usability questionnaires

Only subjects b and c responded to both questionnaires. They both only completed the QUIS questions in the usability questionnaire, so unfortunately it is not possible to compare ratings of Cognitive Dimensions.

Subject b used Serif Draw Plus 4.0, a vector drawing program. Subject b rated the prototype worse in all but one QUIS aspects, all by two or three points difference. Only Inadequate/Adequate Power received the same rating, 4 (neutral).

Subject c, a CorelDraw 9 (vector drawing program) user, rated the prototype better on three QUIS aspects (easy, dull/stimulating and wonderful) but worse on the other three (inadequate power, rigid and frustrating).

Subject c may have been slightly more positive about the prototype than subject b because although both gave the same time estimates for diagramming the cup and crane, subject c found three of the stages of diagramming difficult whilst subject b found three of the stages neutral or easy (initial questionnaire questions 40 – 43). Another possible reason is that subject b uses their vector drawing program more intensively than subject c, and hence had higher expectations of any new software. (Subject b used Serif Draw Plus 4.0 for four to less than ten hours a week, whereas subject b only used CorelDraw 9 for one to less than four hours a week.)

7.2 Summary

This chapter described the usability evaluation by subjects using a usability questionnaire. A total of nine subjects responded and gave the best QUIS rating for "wonderful". The worst ratings were for power and flexibility. "Difficult/Easy" and "Frustrating/Satisfying" received mixed responses.

Only three subjects answered the questions on Cognitive Dimensions. They praised the prototype for visibility, secondary notation, ability to evaluate at any time (HIDDVis, SECN and PROGAnyTime) but criticised the need to plan and use hard mental operations (PREMMustPlan and HARDNeed.)

Two subjects answered both the initial questionnaire and the usability questionnaire. Both used vector drawing programs. One subject gave worse ratings for the prototype; the other gave a mix of better and worse ratings.

8 Conclusions and Further Work

8.1 Meeting of Objectives

Section 1.6, p. 28, listed these objectives

a) Which software is used?

b) What qualities should "good diagrams" possess?

c) How well is the software used?

d) Which approaches are most fruitful?

e) What other approaches could be used?

f) How could such approaches be refined?

Section 3.5, p. 73, listed the types of programs used by initial questionnaire respondents. It also described the responses to the qualities of good diagrams. Considering objectives c, d and e: the most fruitful approach was using vector drawing programs. However, respondents gave information about a number of other methods. None used origami simulators, and although they appeared to be the least promising type of program in the expert evaluation using the Cognitive Dimensions framework (section 4.3, p. 96), their use could address the most time-consuming stage of diagramming, namely drawing step outlines.

The next section considers how successful the prototype developed as a result of preliminary research has been in improving the task of making origami diagrams with a computer.

8.1.1 QUIS ratings for prototype and vector drawing program

	Terrible	Frustrating	Dull	Difficult	Inadequate Power	Rigid
1						
2		k	t	k		r
3				c	kn	
4	cno	not	cnoqsu	lmqs	pu	kp
5	kps	msu	kmr	ou	os	cmosu
6	jrt	cjpr	l	jnprt	cjmr	jnt
7	lmqu	lq	jp		lqt	lq
	Wonderful	Satisfying	Stimulating	Easy	Adequate Power	Flexible

Table 19 Individual QUIS responses for vector drawing program users

	Terrible	Frustrating	Dull	Difficult	Inadequate Power	Rigid
1						
2		h		gh	ah	b
3	bh	ab		bd	cd	cdgh
4		cd			bei	ai
5	cdegi	e	bcdi	f	f	ef
6	a	fi	egh	cei		
7	f		af	a		
	Wonderful	Satisfying	Stimulating	Easy	Adequate Power	Flexible

Table 20 Individual QUIS responses for prototype

Table 19 reproduces Table 9, p. 79, and Table 20 reproduces Table 18, p. 120.These show

that the prototype was more stimulating than vector programs. However, vector programs

were still better for power and flexibility. Frustrating/Satisfying ratings were largely the

same. Terrible/Wonderful and Difficult/Easy ratings for the prototype were slightly worse because no subject was neutral about these.

Two issues arose in this study:

- the effect of the prototype's *fidelity* on the usability evaluation

- the choice of questionnaire content and analysis

Firstly, what was the effect of the prototype's *fidelity* on the usability evaluation? Could the time spent improving the fidelity of the prototype have been better spent elsewhere? Virzi *et al.* (1996) reported that the number of usability problems found in their two experiments was largely unaffected by the fidelity of product being tested. They characterise the fidelity of prototypes along four dimensions:

- **Breadth of features** – the number of features the prototype supports

- **Degree of functionality** – the extent to which the details of its operation are complete

- **Similarity of interaction** – how one communicates with the product (whether by pressing buttons, clicking a mouse, touching a screen, speaking, etc.)

- **Aesthetic refinement** – aspects of the product that do not directly influence its functionality, such as choice of colours and graphic design

The prototype developed in this project can be regarded as a medium-fidelity prototype. It is not low-fidelity because these typically are paper or on-screen mock-ups where interaction is usually simulated by an actor. It is not high-fidelity because the breadth of features and degree of functionality has been compromised e.g. lack of fold lines and turn over symbol in the PostScript diagrams.

Virzi *et al.* (1996, p. 237 warn of one of the possible pitfalls of low-fidelity prototyping:

> Because [researchers] did not accurately represent the slow response times for some aspects of the actual device's performance, estimates of usability for all the prototypes were greater than that of the actual device.

The opposite may be true for the prototype origami diagrammer and simulator: respondents' opinions of usability may be worse than that of the finished program. However, this would be hard to confirm without actually building the finished program. Some evidence for this is in the comments in the usability survey about the lack of performance, stability and robustness:

- Because of the bugs and software vp problem which I know is now fixed
- Settings don't stay
- Crashes a lot
- It seemed to draw a lot of processing power

However, the prototype was criticised for features (or lack thereof) that were intended to be faithful to the finished program (namely the fact the user can only make valley, mountain and inside reverse folds only):

- Inability to do complex folds
- Does not cope with more than basic folds (as detailed in email correspondence.)
- Folding virtual paper is restrictive [may be talking about inability to make some 3d folds e.g. in traditional masu]

Virzi *et al.* (1996, p. 241) acknowledge that there is a belief that low-fidelity prototypes are not suitable for "direct manipulation interfaces, virtual reality or other immersive systems, or

to systems that are extremely response-time sensitive." They are "not so sure", but provide little evidence to support their belief.

The second issue of the project was the effectiveness of the questionnaires used. Would shorter questionnaires have stimulated a larger response rate? Did the questions measure what they intend to measure? There was generally a positive correlation between ratings for QUIS aspects and the Cognitive Dimensions – however, could a different set of questions have been more useful in identifying specific usability strengths and weaknesses? If there were more respondents then more sophisticated analysis of the data would have been possible.

Although the numbers of subjects is too small to allow statistical analysis, their QUIS ratings showed that there is some promise in the use of origami simulation for diagramming. In particular, the results show improved stimulation. Increasing power and flexibility may improve users' level of satisfaction, ease of use and enjoyment.

8.2 Further work

The main criticisms from the usability questionnaire were the prototype's inadequate power and lack of flexibility. The implementation so far only allows certain kinds of folding, namely valley and inside reverse folds. There are many other types of fold that cannot be done e.g. all types of rabbit's ear, certain kinds of multiple, overlapping reverse folds, inflation, stretching[12], etc. If a user needs to make such a fold then they cannot proceed any further. The only option is to continue the diagramming by editing the output in a separate

---

[12] Steps 11-13 of Figure 3, p. 18, show a *stretching* move combined with a squash fold. This kind of move requires more sophisticated simulation than currently available in the prototype.

application. Some believe that this is fatal flaw (Lang, 1996). Therefore the usability of the prototype would be substantially increased by extending the repertoire of folds.

It should be relatively straightforward to implement *outside reverse folds*, given their close similarity to *inside reverse folds*. This would enable users to make a fish base (requested by one usability questionnaire subject). It should also be straightforward to implement *mountain folds* (note that users can already make mountain folds, indirectly, by turning the turn the paper over before making a valley fold).

However, there are two obstacles to implementing other folds: firstly, how to specify the fold in the user interface and secondly, how to implement the change in the data structure. The current user interface forces the user into folding with one hand only, whereas most folders use two hands to manoeuvre and manipulate the paper – they do this without even thinking about it. Nimoy (2002) suggest using a constraint symbol to overcome ambiguity in certain kinds of book folds: "The draggable "Hold Here" object acts as a paper weight. Two option buttons will send the circle up and down between layers of paper." However, these types of abstractions may make a user interface harder to learn and to use.

Jackson (1989) criticised the visual appeal of computer-generated diagrams. One subject in the initial questionnaire wrote that his diagrams were "functional but lifeless". It is possible to produce more appealing diagrams, but this requires more effort, skill and time from diagrammers than they may have. The field of Non-Photorealistic Rendering has the potential to improve the aesthetic appeal of diagrams. Strothotte (2002) describes some methods:

- changing uniform lines into less regular lines using stroke-based techniques
- rendering 3D scenes using non-photorealistic methods

Stroke-based methods are commercially available but are still considered as a niche product e.g. Creature House Expression 3, soon to be Microsoft Acrylic[13].

Future development could use diagram formats in addition to PostScript. SVG (Adobe Systems Incorporated, n.d.) allows styles to be applied to objects. This may make the output easier to edit.

---

[13] http://graphicssoft.about.com/cs/illustration/gr/expression.htm

9 Appendices

9.1 Appendix A – Gerhart-Powals' Cognitive Engineering Principles

Taken from Law and Hvannberg, 2004, p. 250.

P1: Automate unwanted workload

P2: Reduce uncertainty

P3: Fuse data - reduce cognitive load by bringing together lower level data into a higher level summation

P4: Present new information with meaningful aids to interpretation

P5: Use names that are conceptually related to function

P6: Group data in consistently meaningfully ways to decrease search time

P7: Limit data-driven tasks

P8: Include in the displays only that information needed by the users at a given time

P9: Provide multiple coding of data

P10: Practice judicious redundancy

9.2 Appendix B – Nielsen's ten heuristics for Heuristic Evaluation

Taken from Dix *et al.*, 2004, p325-326.

NH1.   **Visibility of system status** Always keep users informed about what is going on, through appropriate feedback within reasonable time. For example, if a system operation will take some time, give an indication of how long and how much is complete.

NH2.   **Match between system and the real world** The system should speak the user's language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions; making information appear in natural and logical order.

NH3.   **User control and freedom** Users often choose system functions by mistake and need a clearly marked 'emergency exit' to leave the unwanted state without having to go through an extended dialog. Support undo and redo.

NH4.   **Consistency and standards** Users should not have to wonder whether words, situations or actions mean the same thing in different contexts. Follow platform conventions and accepted standards.

NH5.   **Error prevention** Make it difficult to make errors. Even better than good error messages is a careful design that prevents a problem from occurring in the first place.

NH6.   **Recognition rather than recall** Make objects, actions and options visible. The user should not have to remember information from one part of the dialog to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

NH7.   **Flexibility and efficiency of use** Allow users to tailor frequent actions. Accelerators - unseen by the novice user - may often speed up the interaction for the expert user to such an extent that the system can cater to both inexperienced and experienced users.

NH8.   **Aesthetic and minimalist design** Dialogs should not contain information that is irrelevant or rarely needed. Every extra unit of information in a dialog competes with the relevant units of information and diminishes their relative visibility.

NH9.   **Help users recognise errors, diagnose and recover from them** Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

NH10.   **Help and documentation** Few systems can be used with no instruction so it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

9.3 Appendix C – Norman's "Seven Principles for Transforming Difficult Tasks into Simple Ones"

Taken from Dix *et al.*, 2004, p283-284.

N1. **Use both knowledge in the world and knowledge in the head.** People work better when the knowledge they need to do a task is available externally - either explicitly or through the constraints imposed by the environment. But experts also need to be able to internalise regular tasks to increase their efficiency. So systems should provide the necessary knowledge within the environment and their operation should be transparent to support the user in building an appropriate mental model of what is going on.

N2. **Simplify the structure of tasks.** Tasks need to be simple in order to avoid complex problem solving and excessive memory load. There are a number of ways to simplify the structure of tasks. One is to provide mental aids to help the user keep track of stages in a more complex task. Another is to use technology to provide the user with more information about the task and better feedback. A third approach is to automate the task or part of it, as long as this does not detract from the user's experience. The final approach to simplification is to change the nature of the task so that it becomes something more simple. In all of this, it is important not to take control away from the user.

N3. **Make things visible: bridge the gulfs of execution and evaluation.** The interface should make clear what the system can do and how this is achieved, and should enable the user to see clearly the effect of their actions on the system.

N4. **Get the mappings right.** User intentions should map clearly onto system controls. User actions should map clearly onto system events. So it should be clear what does

what and by how much. Controls, sliders and dials should reflect the task so a small movement has a small effect and a large movement a large effect.

N5. **Exploit the power of constraints, both natural and artificial.** Constraints are things in the world that make it impossible to do anything but the correct action in the correct way. A simple example is a jigsaw puzzle, where the pieces only fit together in one way. Here the physical constraints of the design guide the user to complete the task.

N6. **Design for error.** To err is human, so anticipate the errors the user could make and design recovery into the system.

N7. **When all else fails, standardise.** If there are no natural mappings then arbitrary mappings should be standardised so that users only have to learn them once. It is this standardisation principle that enables drivers to get into a new car and drive it with very little difficulty - key controls are standardised. Occasionally one might switch on the indicator lights instead of the windscreen wipers, but the critical controls (accelerator, brake, clutch, steering) are always the same.

9.4 Appendix D – Shneiderman's "Eight Golden Rules of Interface Design"

Taken from Dix *et al.,* 2004, p282-283.

S1. **Strive for consistency** in action sequences, layout, terminology, command use and so on.

S2. **Enable frequent users to use shortcuts***,* such as abbreviations, special key sequences and macros, to perform regular, familiar actions more quickly.

S3. **Offer informative feedback** for every user action, at a level appropriate to the magnitude of the action.

S4. **Design dialogs to yield closure** so that the user knows when they have completed a task.

S5. **Offer error prevention and simple error handling** so that, ideally, users are prevented from making mistakes and, if they do, they are offered clear and informative instructions to enable them to recover.

S6. **Permit easy reversal of actions** in order to relieve anxiety and encourage exploration, since the user knows that he can always return to the previous state.

S7. **Support internal locus of control** so that the user is in control of the system, which responds to his actions.

S8. **Reduce short-term memory load** by keeping displays simple, consolidating multiple page displays and providing time for learning action sequences.

9.5 Appendix E – Pilot Version of Initial Questionnaire

Dear participant,

Thank you for agreeing to take part in this study.

This project aims to investigate the use of computers for making origami instructions.

Even if you don't use a computer to make origami instructions yourself, you can still give

useful information in this questionnaire.

This questionnaire will gather data on

- your knowledge and previous experiences....

    o ...as a reader following existing diagrams (Sections 2)

    o ...as a diagrammer (Sections 3 and 4)

- the activities you carry out for making origami instructions

    o without a computer (Section 3)

    o with a computer (Section 4)

Additionally, there are questions in Section 5 that ask for your feedback on this pilot study.

These will be used to improve the questionnaire before being distributed to a wider audience.

Yours,

Tung Ken Lam

tklorigami@yahoo.co.uk

Making Origami instructions with and without computers

The term "origami instructions" is taken to mean any kind of communication designed to

explain how to fold a particular model. This includes diagrams, annotated photographs,

verbal instructions, video, etc.

Please skip a question if you do not wish to, or do not feel able, to answer it.

Your details

1. Name and/or Email *(optional)*:...

2. Age: 1 18 or under   2 19-34   3 35-49   4 50 or over

Your experience of origami

I have been

   3. interested in origami for .... years and .... months.

   4. following origami instructions for .... years and .... months.

   5. making origami instructions without a computer for .... years and .... months.

   6. making origami instructions with a computer for .... years and .... months.

7. Do you think that present-day diagrams are

   1 (a) the best method of conveying folding instructions?

   2 (b) not the best and another method such as O.I.L. (Origami Instruction Language)

   needs to be devised?

      if (b), do you have an alternative in mind?

      (please state)..

Please circle one number below for each statement to indicate how strongly you agree or disagree with the following statements:

	Strongly Agree	Agree	Neither Agree nor Disagree	Disagree	Strongly Disagree
8. I can usually tell when a computer has been used to make origami instructions.	1	2	3	4	5
9. I think that diagrams drawn by hand are usually superior to those drawn with a computer.	1	2	3	4	5
10. I feel that diagrams drawn with a computer are cold and lifeless.	1	2	3	4	5
11. Diagrams drawn with a computer are clearer and easier to follow than those drawn without a computer.	1	2	3	4	5

Conventional diagramming

"Conventional" methods are those that do not rely on using a computer (but may be

supported by using a computer, e.g. using a word processor to print out captions)

I have used the following in making diagrams *(tick all that apply)*:

12. Drawing

   1 pen   2 pencil   3 tone transfers   4 other *(please state)*.............................

13. Drawing tools

   1 ruler   2 set square   3 T square   4 French curve   4 protractor

   5 drawing board   6 other *(please state)*...

14. Output

   1 plain paper   2 photocopier   3 tracing paper   4 transparent film   5 squared

paper

   6 other *(please state)*...

15. Other tools

   1 eraser   2 paste   3 scissors   4 scalpel   5 razor blade

   6 other *(please state)*..

16. Lettering

   1 freehand   2 transfer lettering   3 typewritten   4 stencils

   5 other *(please state)*..

17. Please estimate the amount of time you use (0% = never, 100%=always):

   .... % draughting equipment such as T- and set-squares

   .... % drawing freehand

.... % using a computer (as support)

.... % other *(please specify)*...

18. Please estimate the amount of time you would typically spend using the following

diagramming methods 0% = never, 100%=always):

.... % marking the corners of a folded model on paper

and joining the dots *(sometimes called the "blob" method)*

.... % using a scale and drawing to the required proportion,

say 1/4 of full size

.... % drawing entirely by eye

.... % tracing a photograph or scan

.... % other *(please specify)*...

There are typically four steps for diagramming a model from start to finish. How easy or

difficult do you find each step?

	Easy		Neither Easy nor Difficult		Difficult
19. sketch/plan a draft sequence	1	2	3	4	5
20. draw step outlines	1	2	3	4	5
21. add fold lines, arrows, numbering & text	1	2	3	4	5
22. arrange a final layout.	1	2	3	4	5

23. Imagine you're advising a novice diagrammer. What lessons have you learned from your

experiences that would help a beginner? What advice would have helped you when you first

started to make diagrams? *(please continue on reverse, if needed)*:

....

....

Computer-aided diagramming

"Computer-aided" methods are those that cannot be done without a computer.

Hardware

24. Computer manufacturer .... Model *(if known)* ....

Processor name and speed .... Memory (RAM) .... Hard disk size ....

I have used the following in making diagrams *(tick all that apply)*:

25. Input devices

1 keyboard   2 mouse   3 track ball   4 graphics tablet

5 other *(please state)*..

26. Other input devices

1 scanner   2 digital camera   3 video camera   4 other *(please state)*............

27. Output

1 inkjet   2 laser printer   3 electronic file *(please state format)*......................

4 other *(please state)*..

Software

You may have used a number of software programs to document origami models. Please list

the program that you have used. Include programs that you currently use (an example is

given), and also ones that you have used in the past, but perhaps no longer use:

	Program Name	version(s) used	dates used	Use
Example	*Macromedia Freehand*	*7-9*	*1998-*	*Main diagramming*

	Freehand		*present*	*program.*
Example	*CorelDRAW!*	*-*	*1992-* *1997*	*My first vector drawing* *program. No longer use.*
28.				
29.				
30.				
31.				

32. Please estimate the amount of time you would typically spend using the following

diagramming methods 0% = never, 100%=always)::

.... % marking the corners of a folded model on paper

and joining the dots *(sometimes called the "blob" method)*

.... % using a scale and drawing to the required proportion,

say 1/4 of full size

.... % drawing entirely by eye

.... % tracing a photograph or scan

.... % other *(please specify)*...

There are typically four steps for diagramming a model from start to finish. How easy or difficult do you find each step?

	Easy		Neither Easy nor Difficult		Difficult
33. sketch/plan a draft sequence	1	2	3	4	5
34. draw step outlines	1	2	3	4	5
35. add fold lines, arrows, numbering & text	1	2	3	4	5
36. arrange a final layout.	1	2	3	4	5

37. Please describe any special techniques you use for any of these steps e.g. "Sometimes I trace a photograph of the final model when it complex or highly three-dimensional." *(please continue on reverse, if needed)*

....

....

38. What lessons have you learned from your experiences that would help a beginner using a computer for diagramming? What advice would have helped you when you first started to make diagrams with a computer? *(please continue on reverse, if needed)*:

....

....

Usability of a specific program

Please select the main program that you use for diagramming:

Name of program ....

39. On average, how much time do you spend per week using this program?

  ₁ less than 1 hour   ₂ 1 to less than 4 hours   ₃ 4 to less than 10 hours   ₄ 10 hours or more

Overall reactions

Please circle the numbers which most appropriately reflect your impressions about the

software. If not applicable, select NA:

40. terrible	1	2	3	4	5	6	7	wonderful	NA
41. frustrating	1	2	3	4	5	6	7	satisfying	NA
42. dull	1	2	3	4	5	6	7	stimulating	NA
43. difficult	1	2	3	4	5	6	7	easy	NA
44. inadequate power	1	2	3	4	5	6	7	adequate power	NA
45. rigid	1	2	3	4	5	6	7	flexible	NA

Please estimate the time you would need using this program to produce diagrams for:

46. Traditional cup   basic quality............ hours   high quality............ hours

47. Traditional Crane   basic quality............ hours   high quality............ hours

List the most **positive** aspect(s):

48. …………………………………………………………………………………….

49. …………………………………………………………………………………….

50. …………………………………………………………………………………….

List the most **negative** aspect(s):

51. …………………………………………………………………………………….

52. …………………………………………………………………………………….

53. …………………………………………………………………………………….

54. …………………………………………………………………………………….

55. ……………………………………………………………………………………

It would be helpful if could you answer the following sections on Screen, Learning, System Capabilities, Technical Manuals and On-line Help. **However, if you wish, you may skip these sections for now and return to them later.**

Screen

56. Screen layouts are helpful

   never   1   2   3   4   5   6   7   always   NA

57. Sequence of screens

   confusing   1   2   3   4   5   6   7   clear   NA

Please write your comments about the screens here *(please continue on reverse, if needed)*::

....

....

Learning

58. Learning to operate the system

   difficult   1   2   3   4   5   6   7   easy   NA

59. Exploration of features by trial and error

 disencouraging   1   2   3   4   5   6   7   encouraging   NA

60. Remembering names and use of commands

difficult   1   2   3   4   5   6   7   easy   NA

61. Tasks can be performed in a straightforward manner

 never   1   2   3   4   5   6   7   always   NA

Please write your comments about learning here *(please continue on reverse, if needed)*:

....

....

System capabilities

62. System speed

   too slow   1   2   3   4   5   6   7   fast enough   NA

63. The system is reliable

   never   1   2   3   4   5   6   7   always   NA

64. Correcting your mistakes

   difficult   1   2   3   4   5   6   7   easy   NA

65. Ease of operation depends on your level of experience

   never   1   2   3   4   5   6   7   always   NA

Please write your comments about system capabilities here *(please continue on reverse, if needed)*::

....

....

Technical manuals and on-line help

66. Technical manuals are

   confusing   1   2   3   4   5   6   7   clear   NA

67. Amount of help given

   inadequate   1   2   3   4   5   6   7   adequate   NA

68. On-line help

   useless   1   2   3   4   5   6   7   helpful   NA

Please write your comments about technical manuals and on-line help here *(please continue on reverse, if needed)*::

....

....

Feedback on this questionnaire

	Strongly Agree	Agree	Neither Agree nor Disagree	Disagree	Strongly Disagree
69. I found it easy to follow.	1	2	3	4	5
70. There were no ambiguous questions.	1	2	3	4	5
71. I felt all relevant areas were covered.	1	2	3	4	5
72. All of the questions were relevant to the proposed topic.	1	2	3	4	5
73. The number of questions was about right.	1	2	3	4	5

Please explain any problems here:

....

....

....

....

....

....

....

....

74. I would find the following data collection methods acceptable *(tick all that apply):*

   1 paper form by post   2 paper form in person

   3 electronic document by email   4 text only format   5 rich text format

₆ on-line web form   ₇ with ability to save mid-way and complete later

Please write any further comments here:

....

....

....

....

Thank you for taking the time to complete this questionnaire.

Your help is appreciated.

9.6 Appendix F – Redesign of Initial Questionnaire based on Pilot Study Questionnaire

Design of Pilot Questionnaire

The pilot questionnaire (Appendix E – Pilot Version of Initial Questionnaire, p. 139) was divided into four sections:

- General background, experience and origami questions, including reactions to computer diagrams

- Conventional diagramming: questions drawn from Cunliffe (1988; 1989a, b)

- Computer diagramming: questions drawn from QUIS, the Questionnaire for User Interaction Satisfaction (Shneiderman, 2005)

- Questions soliciting feedback on the design of the questionnaire

Results of Pilot Questionnaire

Four respondents completed the pilot questionnaire. (See Appendix J – Results of Pilot Initial Questionnaire, p. 189 for the results) The longest time taken was 15 minutes. One questionnaire was not completed in full because of a lack of time, another because the respondent only used conventional methods.

One respondent gave negative feedback to three of the five questions asking for feedback on the questionnaire. Two did not agree that the number of question was about right. One was moderately positive to all of the feedback questions.

One written comment was "you've assumed that I generate origami diagrams regularly - which I don't so I found some questions difficult to answer". A verbal comment by another

respondent mirrored this: he said that he makes different kinds of instructions for different purposes – instructions for only personal use might involve gluing step folds on paper, whereas diagrams for publishing would be made on a computer.

Two respondents completed the QUIS overall reactions; only one respondent completed all of the detailed QUIS questions.

Redesign

Based on the feedback from subjects, I decided to make the following changes for the main study (Appendix G – Initial Questionnaire, p. 157):

- Reduce the number of questions by removing the detailed QUIS questions.

- Add questions asking about purpose of diagramming and asking for estimates of number of pages/number of designs.

- Add computer technique of "using transformation tools such as rotation, reflection, scaling, etc" (because this type of technique was mentioned twice: once as a positive aspect "I use the program for producing mathematics diagrams and it includes transformations which is helpful", once as a repetition of the analogous conventional method).

- Make it easier to compare conventional diagramming with the computer methods.

- Add questions to confirm the validity of Lang's (2000) guidelines.

On reading about viscosity, it occurred to me that the stage of "plan/sketch a sequence" should be considered outside the diagramming activity, because it is way of overcoming the viscosity of a program. Therefore a "Not Applicable", N/A, category should be added.

The questions asking for years and months of origami and following origami instructions were intended to identify subjects who have only recently started to use origami instructions. However all pilot responses used the same number for both, hence these two questions were replaced by a single question.

No meaningful data was given in response to questions asking about computer hardware. These were therefore simplified to a single question asking for the operating system used.

I decided that questions to confirm Lang's (2000) ten guiding principles for diagramming (p2-5) would be more useful than finding out if readers tolerated computer diagrams. The motive for this change was that more detail on the important qualities of good diagrams would provide specific aspects to concentrate on. For example, it is easier to determine how well a program supports the showing of multiple layers than if its output is "clear". "Clear" is a very general term that is open to widely different interpretations: what it is that makes the output "clear"?

In some way Lang's principles resemble design guidelines like Norman's (1988) e.g. the first guideline is effectively the same as Norman's 7th principle (N7: When all else fails, standardise). Lang's principles are:

> L1. Be consistent with the past
>
> L2. Make the drawings stand alone
>
> L3. Make the text stand alone, if possible
>
> L4. Use letters to indicate important features
>
> L5. Be grammatically correct (i.e. use consistent verbs and nouns)

      L6.   Use arrows to indicate motion

      L7.   Don't leave the reader dangling (i.e. does not show the result of a fold)

      L8.   Show one step per drawing

      L9.   Distort the model for clarity

      L10.   Show multiple layers whenever possible

I wanted to devote an appropriately small amount of space to this section of the questionnaire because of its minor importance. I removed any principle that depends entirely on the author: e.g. the quality of grammatical correctness depends on the author, not the software. I therefore removed the fifth principle.

I added two questions on extra qualities that have an impact on diagrammers' work and may or may not be important to readers:

      15. Distinguish each side of the paper by shading

      16. Good diagrams are compact and use concise explanations, when possible.

The quality of text being able to stand alone *can* depend on the software and was therefore retained (e.g. because some programs handle text better than others). A simple five-point Likert scale was adequate for these statements. A scale with more points would be difficult to analyse, but a simple Yes/No answer would mask nuances and subjects' priorities.

Only one pilot subject completed the QUIS detailed questions in full – perhaps they were considered too difficult or off-putting by the other subjects (the use of contrasting adjectives and a seven-point Likert scale may have caused cognitive overload). I retained the QUIS overall reaction question because they are only six questions. These questions have been used

many times in other studies (p. 402, Preece *et al.*, 2003) and have good reliability and validity (Chin *et al.*, 1988, found that the ratings of 150 subjects correlated with whether a subject liked or disliked a program). I also retained the use of a seven-point scale because this did not seem to cause a problem with pilot subjects. As Perlman (2001) writes, "seven-point rating scales ... allow three levels of either positive or negative ratings; two levels seems too few. I prefer to range from 1-7 (bad-good) instead of the more techno -3...+3."

I retained Perlman's questions asking for negative/positive aspects because these did attract useful comments. The number of aspects was right: Perlman (2001) writes "I like to ask users about the N most negative / positive points about a system (first negative, then positive). I usually get N/2 answers, so 3 or 5 is a good number to ask for."

9.7 Appendix G – Initial Questionnaire

Dear participant,

Thank you for agreeing to take part in this study.

This project aims to investigate the use of computers for making origami instructions. It aims to summarise current approaches and to make most productive approaches more widely known. If appropriate, new software will be developed to improve the usability of computers for the task of making origami instructions.

Even if you don't use a computer to make origami instructions yourself, you can still give useful information in this questionnaire.

This questionnaire will gather data on

- your knowledge and previous experiences....

    o ...as a reader following existing instructions (Section 2)

    o ...as an author of origami instructions (Sections 3)

- the activities you carry out for making origami instructions

    o without a computer (Section 3A)

    o with a computer (Section 3B)

Yours,

Tung Ken Lam

tklorigami@yahoo.co.uk

Making Origami instructions with and without computers

The term "origami instructions" is taken to mean any kind of communication designed to explain how to fold a particular model. This includes diagrams, annotated photographs, verbal instructions, video, etc.

Please skip a question if you do not wish to, or do not feel able, to answer it.

Your details

1. Name and/or Email *(optional)*:...

2. Age: 1 18 or under   2 19-34   3 35-49   4 50 or over   3. Gender: 1 Male   2 Female

4. I am interested in the results of this project, including the opportunity to try out any software developed as a result.

   1 Yes   2 No

      If yes – I prefer to be contacted by

         1 email   2 telephone   3 conventional mail   4 other

      My preferred contact details are ....

         ....

Your experience of origami

5. I have been interested in origami for .... years.

Please CIRCLE one number below for each statement to indicate how strongly you agree or disagree with the following statements:

Good instructions...	Strongly Agree	Agree	Neither Agree nor Disagree	Disagree	Strongly Disagree
6. use the internationally-accepted standard symbols and conventions derived from Yoshizawa, Randlett and Harbin	1	2	3	4	5
7. make each drawing stand alone (i.e. can be understood without reading the text)	1	2	3	4	5
8. make the text for a drawing stand alone, if possible	1	2	3	4	5
9. use letters to indicate important features	1	2	3	4	5
10. use arrows to indicate motion	1	2	3	4	5
11. don't leave you dangling (by failing to show the result of a fold)	1	2	3	4	5
12. show a single step (i.e. fold or procedure) per drawing	1	2	3	4	5
13. distort the model for clarity	1	2	3	4	5
14. show multiple layers whenever possible	1	2	3	4	5
15. distinguish each side of the paper by shading	1	2	3	4	5
16. are compact and use concise explanations, when possible	1	2	3	4	5
17. other (please state)..................................	1	2	3	4	5

Your experience of making origami instructions

18. I have made origami instructions.   ₁ Yes   ₂ No

If no – there is no need to continue, thank you.

   If yes –

      19. without a computer for .... years.

      20. with a computer for .... years.

Purpose	Without a computer Approx no. per year of		With a computer Approx no. per year of	
	models	pages	models	pages
Personal use only (not for distribution)				
Limited distribution (<50 copies)				
Wider distribution e.g. magazine, internet, etc				
To publish in a book				

Conventional diagramming

"Conventional" methods are those that do not rely on using a computer (but may be

supported by using a computer, e.g. using a word processor to print out captions)

21. Please estimate the amount of time you (0% = never, 100%=always):

.... % use draughting equipment such as T- and set-squares

.... % draw freehand

.... % use computer (as support)

.... % other *(please specify)*...

22. Please estimate the amount of time you would typically spend using the following

diagramming methods 0% = never, 100%=always):

.... % marking the corners of a folded model on paper

and joining the dots *(sometimes called the "blob" method)*

.... % using a scale and drawing to the required proportion, say 1/4 of full size

.... % drawing entirely by eye

.... % tracing a photograph or scan

.... % other *(please specify)*...

There are typically four steps for diagramming a model from start to finish. How easy or

difficult do you find each step? How much time do spend on each step?

	Easy		Neither Easy nor Difficult		Difficult	Aprrox Time spent
23. sketch/plan a draft sequence	1	2	3	4	5	.........%
24. draw step outlines	1	2	3	4	5	.........%
25. add fold lines, arrows, numbering & text	1	2	3	4	5	.........%
26. arrange a final layout.	1	2	3	4	5	.........%
27. other *(please specify)*...........................	1	2	3	4	5	.........%

Computer-aided diagramming

"Computer-aided" methods *rely* on using a computer.

The Computer

28. Operating system

1 Windows ......   2 Macintosh System ......   3 other *(please state)*...................

Software

29. If you have used any of the following types of program for making origami instructions,

please CIRCLE how easy or difficult they were to use for making origami instructions and

write the program name

	Easy to make origami instructions		Neither Easy nor Difficult		Difficult to make origami instructions	Name & version of program
30. Vector drawing packages e.g. Freehand, Illustrator	1	2	3	4	5	....
31. General graphics module e.g. Drawing tools Microsoft Word XP	1	2	3	4	5	....
32. CAD e.g. AutoCAD, TurboCAD	1	2	3	4	5	....
33. Other specialist graphics program e.g. Visio, Quark XPress *(please state type)*	1	2	3	4	5	....
34. Origami-oriented programming	1	2	3	4	5	....

languages e.g. Doodle, ORIDRAW

35. 3-dimensional   1   2   3   4   5   ....

mathematical/modeling software e.g.

Mathematica, Cabri 3D

36. Dynamic Geometry software e.g.   1   2   3   4   5   ....

Cabri, Geometer's sketchpad

37. Origami simulator   1   2   3   4   5   ....

38. Other *(please state type)*   1   2   3   4   5

....

39. Please estimate the amount of time you would typically spend using the following

diagramming methods 0% = never, 100%=always):

.... % tracing the outline of folded model on screen

.... % using transformation tools such as rotation, reflection, scaling, etc

.... % drawing entirely by eye

.... % automatically tracing a photograph or scan with a program

.... % manually tracing a photograph or scan

.... % other *(please specify)*...

There are typically four steps for diagramming a model from start to finish. How easy or difficult do you find each step? How much time do spend on each step?

	Easy		Neither Easy nor Difficult		Difficult	Aprrox Time spent
40. sketch/plan a draft sequence	1	2	3	4	5	.........%
41. draw step outlines	1	2	3	4	5	.........%
42. add fold lines, arrows, numbering & text	1	2	3	4	5	.........%
43. arrange a final layout.	1	2	3	4	5	.........%
44. other *(please specify)*............................	1	2	3	4	5	.........%

....

Usability of a specific program

Please select the main program that you use for diagramming:

45. Name of program ....

46. On average, how much time do you spend per week using this program?

　₁ less than 1 hour　　₂ 1 to less than 4 hours　　₃ 4 to less than 10 hours　　₄ 10 hours or more

Overall reactions

Please circle the numbers which most appropriately reflect your impressions about the software. If not applicable, select NA:

47. terrible	1	2	3	4	5	6	7	wonderful	NA
48. frustrating	1	2	3	4	5	6	7	satisfying	NA
49. dull	1	2	3	4	5	6	7	stimulating	NA
50. difficult	1	2	3	4	5	6	7	easy	NA
51. inadequate power	1	2	3	4	5	6	7	adequate power	NA
52. rigid	1	2	3	4	5	6	7	flexible	NA

Please estimate the time you would need using this program to produce diagrams for:

53. Traditional cup   basic quality............ hours   high quality............ hours

54. Traditional Crane   basic quality............ hours   high quality............ hours

List the most **positive** aspect(s):

55. ....

56. ....

57. ....

List the most **negative** aspect(s):

58. ....

59. ....

60. ....

61. ....

62. ....

Detailed usability questions (Optional)

	Strongly Agree	Agree	Neither Agree nor Disagree	Disagree	Strongly Disagree
63. I can define new terms (e.g. definition of arrow heads, named styles and colours) which allows me to express my ideas more clearly.	1	2	3	4	5
64. I have to define new terms before I can do anything else (e.g. names of styles)	1	2	3	4	5
65. Some parts of the program are related to another: changing one part may affect others. I can usually see these kinds of dependencies (e.g. effect of changing a named style or colour)	1	2	3	4	5
66. As the document gets larger, problems with dependency get bigger.	1	2	3	4	5
67. I can order the diagramming tasks in any way I like (e.g. start with final drawing first; add/edit labels and captions to steps at any time)	1	2	3	4	5
68. I need to plan and think ahead before starting to work.	1	2	3	4	5
69. I can make notes to myself that are separate from the origami instructions e.g. use	1	2	3	4	5

comments, colours, formatting, etc

70. I can easily make changes to previous work.	1	2	3	4	5
71. Some kinds of changes that are important are more difficult to make than they should be.	1	2	3	4	5
72. I can easily find the parts of the diagram that I am interested in whilst it is being created or changed.	1	2	3	4	5
73. When I need to compare/combine different parts of the diagrams, I can see them at the same time.	1	2	3	4	5
74. The program works in a way that closely maps to how diagrams work.	1	2	3	4	5
75. Parts of the program seem particularly strange for origami diagramming.	1	2	3	4	5
76. Things that are similar are presented in similar ways (e.g. squares, rectangles and polygons can all be edited in similar ways)	1	2	3	4	5
77. The program lets me make diagrams reasonably briefly (not long-winded)	1	2	3	4	5
78. It is easy to make mistakes.	1	2	3	4	5
79. I often find myself making small slips that irritate me/make me feel stupid.	1	2	3	4	5

80. I sometimes need to work things out that   1   2   3   4   5

are complex or difficult outside of the

program

81. There are some tasks that make inordinate   1   2   3   4   5

demands on my memory or are long-winded.

82. It is easy to stop and check the diagrams in   1   2   3   4   5

the middle of completion.

83. I can check the work at any time.   1   2   3   4   5

84. I can try out partially-completed versions   1   2   3   4   5

of instructions.

85. I can sketch out things when playing with   1   2   3   4   5

ideas, or when I'm not sure how to proceed.

86. I can easily tell what each function/feature   1   2   3   4   5

of the program is for.

Thank you for taking the time to complete this questionnaire.

Your help is appreciated.

9.8 Appendix H – Help file for redesigned origami simulator

Origami Simulator and Diagrammer

This program is based on the work by Miyazaki *et al.* (1996) and the source code that he has

made available at http://www.om.sccs.chukyo-u.ac.jp/main/research/origami/index.html

What does this program do?

Why have you made this program?

What can I do to help?

Requirements

Computer

Program files

Other software

Caveats

Getting Started

How do I...?

Fold the paper

Use other paper shapes

Change the paper colours

Make a reverse fold

Make a squash fold

Make a petal fold

Make a three-dimensional fold that lifts the paper away from the starting plane

Move the paper

Rotate the paper

Turn the model over

Change my view

Undo a fold

Export diagrams

Menu command reference

Contact

References

What does this program do?

The program simulates origami on screen. You can make valley folds and inside reverse folds. Combining these folds allows certain kinds of squash folds, petal folds, sinks and rabbit's ears.

The program works in 3 dimensions so that you can rotate, spin and zoom in and out. You can make folds that are not parallel to the plane original paper e.g. you can fold the wings of a crane so that they "stick out" from the main body.
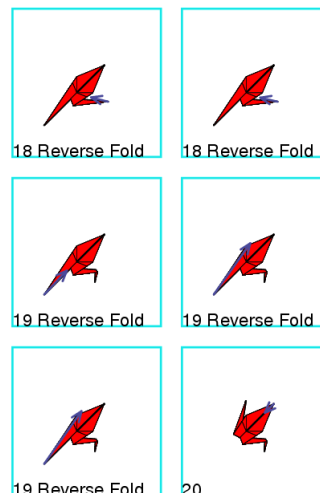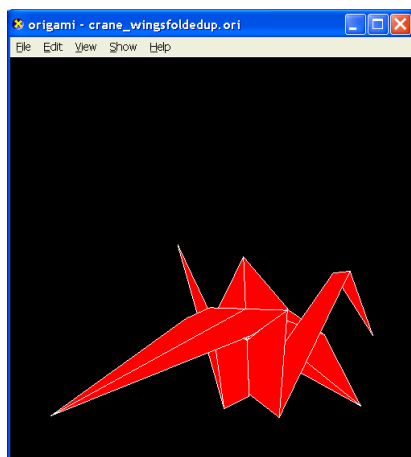
This program has been extended to make diagrams in PostScript format.

Why have you made this program?

I have chosen origami diagramming as the subject of a dissertation project.

What can I do to help?

I would be grateful if you could complete the usability survey at

http://www.angelfire.com/or3/tklorigami0

Requirements

Computer

PC with Windows 98 or later.

DirectX version needs to be DirectX 9

(http://www.microsoft.com/downloads/details.aspx?familyid=9226A611-62FE-4F61-ABA1-914185249413&displaylang=en) and a graphics card with a display driver that supports hardware-accelerated Direct3D.

If the DirectX version does not work on your PC, please contact me. I may be able to supply an OpenGL version. The OpenGL version has less functionality than the DirectX version, so please try the DirectX version first.

You will need a mouse to operate the software.

Program files

Filename	Description	Comments
Miyazaki.exe	executable program file	

origami.txt	Settings file	File is created if it does not exist. Program will fail to run if it does not exist and cannot create it.
Origami.chm	HTML Help File	
d3dx9d.dll	DirectX 9 dll	place in same directory as the executable file
log.txt	Log file	Created each time the program is run: contains information for debugging purposes
	Usability questionnaire	
fold.tmp	Animation stage	Created when exporting or animating
*.ori	Saved folding (Original or new file format)	Program always saves version 2 file format, but can read original or new file format.
*.ps	diagrams in PostScript format	

Other software

In order to view the PostScript diagrams, you need a PostScript viewer. I recommend the freely downloadable GhostScript 8.51

http://prdownloads.sourceforge.net/ghostscript/gs851w32.exe?download and GSview 4.7

ftp://mirror.cs.wisc.edu/pub/mirrors/ghost/ghostgum/gsv47w32.exe

GhostScript can convert PS files to PDF. Several websites can do this free e.g.

www.ps2pdf.com. Alternatively, try the "Online viewer for PDF, PostScript and Word" at

http://view.samurajdata.se/

Another option is to import the PostScript file into a graphics application.

Caveats

The simulation is not perfect – the program can sometimes make mistakes such as cutting the

paper or incorrectly moving paper layers. It does not perform collision detection and can

sometimes make impossible folds. When this happens, press 'Z' to undo the fold. The total

number of fold steps cannot exceed 255 folds.

There are some features that could be better implemented, but have been postponed due to

time constraints.

Getting Started

When run for the first time, the program displays a blue square.

How do I...?

Fold the paper

1. Click the paper with the

left mouse button and keep

the button pressed.
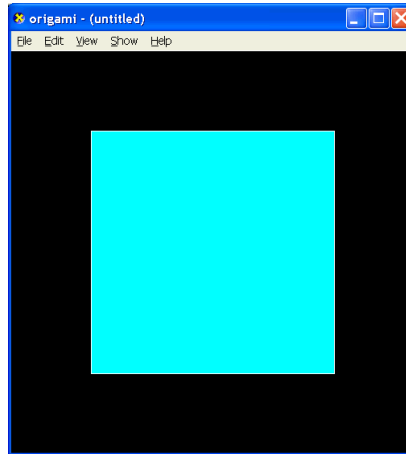
2. Keep the button pressed

and drag the mouse to adjust

the fold.

As you drag the mouse,

important locations are

highlighted in yellow, which

allows you to make accurate

folds e.g. diagonal, matching

edge, vertex.
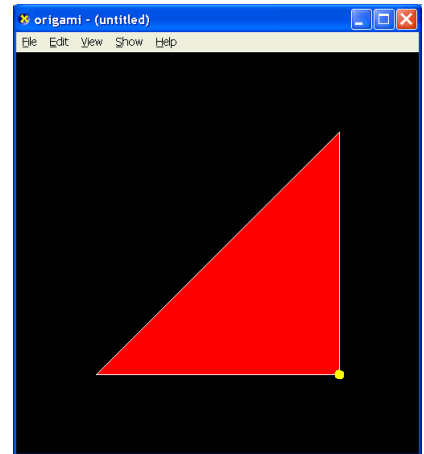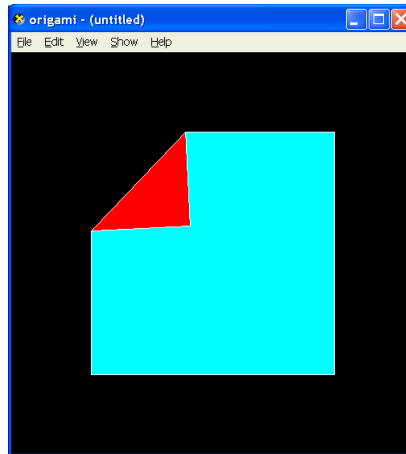
3. Release the mouse button

to make the fold.
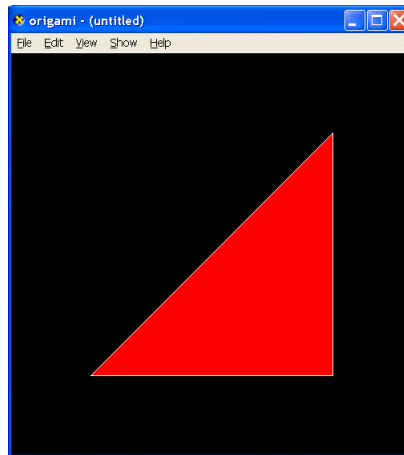
Use other paper shapes

The default paper shape is a square. Use menu File – Load Settings... to open settings files

with other shapes defined.

To create your own shape, edit "origami.txt" e.g. use the following text for an equilateral

triangle

```
VertexSize 3
0.0000    20.0000
-17.3205  -10.0000
17.3205   -10.0000
```

"paper shapes.xls" contains worksheets that show how to determining the coordinates of 1:n

rectangles (halving width and height) and regular polygons (converts polar to Cartesian

coordinates).

NB: the program may crash if you load folding defined for a different paper shape. For

example, you load a 2:1 rectangle, fold and save. The program may crash if you load this

folding when starting with a square.

Change the paper colours

When run for the first time, the program creates a settings file "origami.txt"

```
VertexSize  4
  10.0  10.0
-10.0  10.0
-10.0 -10.0
  10.0 -10.0
FrontColor  255   0   0
BackColor     0 255 255
BorderColor 255 255 255
/FrontTexture   chiyo1.bmp
/BackTexture    chiyo3.bmp
```

Change the FrontColor and BackColor settings which are in the form r g b.

Make a reverse fold

Whilst dragging the paper, press 'T' once to make a "tuck" fold. You will see that the arrangement of paper layers is now different. (Press 'T' once again to change back to a simple valley fold.) Release the mouse button to make the fold.
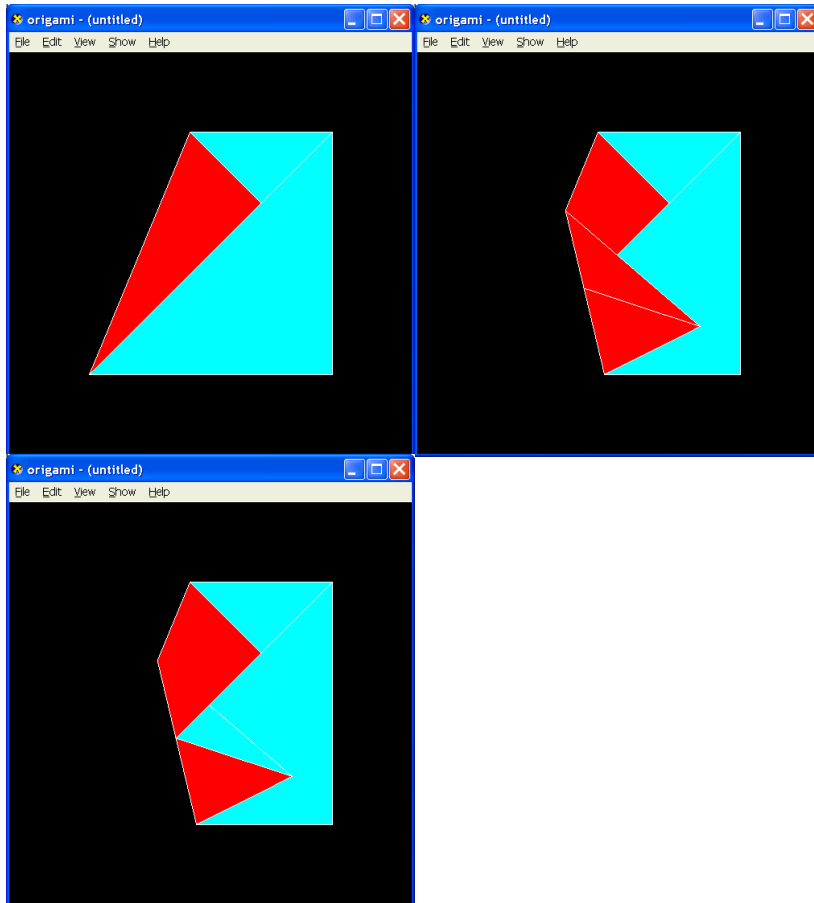
Make a squash fold

You cannot do this directly. Instead, first make a reverse fold and then fold one of the two

new flaps across.
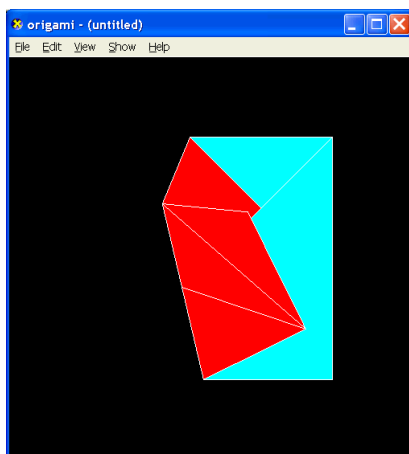
Make a petal fold

You can make certain petal folds using a series of reverse folds and valley folds.

Make a three-dimensional fold that lifts the paper away from the starting plane

Click on the paper. Keep the mouse button down and press 'M' to "peel" the paper. The longer you keep 'M' pressed, the more paper that is "peeled".

If you peel too much, press 'N' to "unpeel" the paper.

Use the view controls to see the results.

Move the paper

Use the cursor keys and the number pad to change your viewpoint.

7	8	9
4	5	6
1	2	3

	^ |	
<-	| v	->

cursor keys   number pad

	Zoom in	
Rotate CW	Zoom out	Rotate CCW
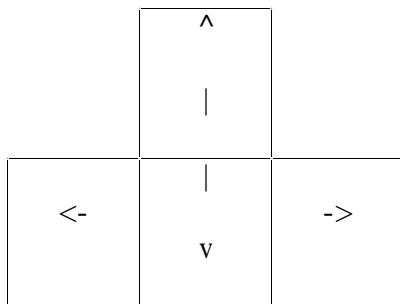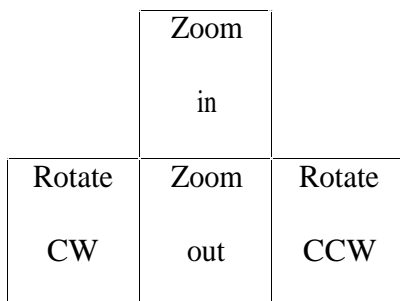
cursor keys

	Tilt Up	
Tilt Left	Reset	Tilt Right
Slide Left	Tilt Down	Slide Right

number pad

Rotate the paper

Use cursor key left arrow and right arrow to rotate the paper in the plane.

Turn the model over

Press 'O' to turn the model over.

Change my view

In addition to rotating the model in the plane, you can tilt the model and slide it left and right.

Undo a fold

Press 'Z' to undo the most recent fold. Note that this is different to unfolding i.e. crease a fold line, make the fold and then pick and move the paper back to its original position.

Export diagrams

Use menu item File – Export... (Ctrl-E). You will need a PostScript viewer to read the diagrams.

Menu command reference

Top level menu	Sub-menu item	Keyboard Shortcut	Description and comments
File			
	New	Ctrl-N or I	Abandon current folding and start with a new sheet of paper (uses current settings).
	Open...	Ctrl-O	Open a saved folding.
	-		
	Save	Ctrl-S	Save the current folding. You will asked for a filename if the current folding has not been saved before.
	Save As...		Save the current folding.
	-		
	Load Settings....	Ctrl-L	Use this to set different colours and paper shapes.
	-		
	Export Diagram PS File...	Ctrl-E	Export the current folding as diagrams in PostScript format. The folding will be animated on screen as part of this process. If current folding is called saved.ori, diagram file will be named saved.ps, otherwise default.ps.
	-		

	Config Display...	F2	Configure Direct3D settings.
	-		
	Exit	ESC	Quit the program.
Edit			
	Undo	Ctrl-Z or Z	Undo the most recent fold. (Note that you cannot redo the fold.)
	-		
	Reverse fold	T	You cannot easily access this menu
	Valley fold	T	whilst folding. This menu item is here as reminder of the keyboard command.
	-		
	Peel paper	M	You cannot easily access this menu
	Unpeel paper	N	whilst folding. This menu item is here as reminder of the keyboard command.
	-		
	Turn Over	O	Rotate the model 180° horizontally.
	-		
	Debug	Ctrl-D	Print out current folding state for debugging purposes.
View			Note that all keyboard shortcuts only work from the number keypad. They number keys across the top of the conventional part of the keyboard do not work.
	Rotate Clockwise	left arrow	
	Rotate Anticlockwise	right arrow	
	-		
	Tilt Left	4	

	Tilt Right	6	
	Tilt Up	8	
	Tilt Down	2	
	-		
	Slide Left	1	
	Slide Right	3	
	-		
	Zoom in	up arrow	
	Zoom out	down arrow	
	-		
	Reset	5	
Show			
	Rotation	R	Toggle mode of continuous horizontal rotation.
	Coordinate Axis	A	Show/hide the X, Y and Z axes.
	Animation		Animate the folding from start to current state.
Help			
	Help	F1	Open the help file.
	-		
	About...		Shows program version and links to further resources.

Table 21 Menu command reference

Contact

If you have any questions about this software, please contact me tklorigami@yahoo.co.uk

References

Miyazaki, S.Y., Yasuda, T., Yokoi, S. and Toriwaki, J. I. (1996) 'An origami playing simulator in the virtual space', *Journal of Visualization and Computer Animation,* 7 (1): 25-42 Jan-Mar 1996, URL http://www.om.sccs.chukyo-u.ac.jp/main/research/origami/indexj.html (1 Jul 2005)

9.9 Appendix I – Usability of Prototype Questionnaire

Dear participant,

Thank you for evaluating this software.

This short questionnaire gathers your views of the usability of the Origami Simulator and Diagrammer software.

Please note that the software is only a prototype: it is not a fully-developed professional program. It has been created for a research project to test the usability of origami simulation for the making origami diagrams.

This questionnaire will gather data on

- your overall reactions
- specific usability issues

Yours,

Tung Ken Lam

tklorigami@yahoo.co.uk

Usability of the Origami Simulator and Diagrammer

Please skip a question if you do not wish to, or do not feel able, to answer it.

Your details (optional)

Please skip this section if you have already completed the earlier "Diagrams Survey".

1. Name and/or Email *(optional):* ....

2. Age: ₁ 18 or under ₂ 19-34 ₃ 35-49 ₄ 50 or over   3. Gender: ₁ Male ₂ Female

4. I am interested in the results of this project.

   ₁ Yes ₂ No

   If yes – I prefer to be contacted by

   ₁ email ₂ telephone ₃ conventional mail ₄ other

   My preferred contact details are ....

Your experience of origami

Please skip this section if you have already completed the earlier "Diagrams Survey".

5. I have been interested in origami for .... years.

6. I have made origami instructions. ₁ Yes ₂ No

   If yes –

   7. without a computer for .... years.

   8. with a computer for .... years.

Overall reactions

Please CIRCLE the numbers which most appropriately reflect your impressions about the software. If not applicable, select NA (On an electronic version, please indicate your choice by formatting e.g. highlight, colour, **bold** format, border style, etc):

9. terrible	1	2	3	4	5	6	7	wonderful	NA
10. frustrating	1	2	3	4	5	6	7	satisfying	NA
11. dull	1	2	3	4	5	6	7	stimulating	NA
12. difficult	1	2	3	4	5	6	7	easy	NA
13. inadequate power	1	2	3	4	5	6	7	adequate power	NA
14. rigid	1	2	3	4	5	6	7	flexible	NA

Please estimate the time you would need using this program to produce diagrams for:

15. Traditional cup   basic quality .... hours   high quality .... hours

16. Traditional Crane   basic quality .... hours   high quality .... hours

List the most **positive** aspect(s):

17. ....

18. ....

19. ....

List the most **negative** aspect(s):

20. ....

21. ....

22. ....

23. ....

24. ....

Specific usability questions (Optional)

On an electronic version, please your indicate choice by formatting e.g. ==highlight==, colour, **bold** format, border style, etc.

	Strongly Agree	Agree	Neither Agree nor Disagree	Disagree	Strongly Disagree
25. I can define new terms (e.g. definition of arrow heads, named styles and colours) which allows me to express my ideas more clearly.	1	2	3	4	5
26. I have to define new terms before I can do anything else (e.g. names of styles)	1	2	3	4	5
27. Some parts of the program are related to another: changing one part may affect others. I can usually see these kinds of dependencies (e.g. effect of changing a named style or colour)	1	2	3	4	5
28. As the document gets larger, problems with dependency get bigger.	1	2	3	4	5
29. I can order the diagramming tasks in any way I like (e.g. start with final drawing first; add/edit labels and captions to steps at any time)	1	2	3	4	5
30. I need to plan and think ahead before starting to work.	1	2	3	4	5
31. I can make notes to myself that are separate from the origami instructions e.g. use comments, colours, formatting, etc	1	2	3	4	5
32. I can easily make changes to previous work.	1	2	3	4	5
33. Some kinds of changes that are important are more difficult to make than they should be.	1	2	3	4	5
34. I can easily find the parts of the diagram that I am interested in whilst it is being created	1	2	3	4	5

or changed.

35. When I need to compare/combine different parts of the diagrams, I can see them at the same time.	1	2	3	4	5
36. The program works in a way that closely maps to how diagrams work.	1	2	3	4	5
37. Parts of the program seem particularly strange for origami diagramming.	1	2	3	4	5
38. Things that are similar are presented in similar ways (e.g. squares, rectangles and polygons can all be edited in similar ways)	1	2	3	4	5
39. The program lets me make diagrams reasonably briefly (not long-winded)	1	2	3	4	5
40. It is easy to make mistakes.	1	2	3	4	5
41. I often find myself making small slips that irritate me/make me feel stupid.	1	2	3	4	5
42. I sometimes need to work things out that are complex or difficult outside of the program	1	2	3	4	5
43. There are some tasks that make inordinate demands on my memory or are long-winded.	1	2	3	4	5
44. It is easy to stop and check the diagrams in the middle of completion.	1	2	3	4	5
45. I can check the work at any time.	1	2	3	4	5
46. I can try out partially-completed versions of instructions.	1	2	3	4	5
47. I can sketch out things when playing with ideas, or when I'm not sure how to proceed.	1	2	3	4	5
48. I can easily tell what each function/feature of the program is for.	1	2	3	4	5

Thank you for taking the time to complete this questionnaire.

Your help is appreciated.

9.10 Appendix J – Results of Pilot Initial Questionnaire

Full results are on the CD-ROM in the Excel spreadsheet file "Anonymised Pilot Initial

Questionnaire Results.xls"

9.11 Appendix K – Results of Initial Questionnaire

Full results are on the CD-ROM in the Excel spreadsheet file "Anonymised Initial

Questionnaire Results.xls"

9.12 Appendix L – Results of Usability of Prototype Questionnaire

Full results are on the CD-ROM in the Excel spreadsheet file "Anonymised Usability of

Prototype Questionnaire Results.xls"

9.13 Appendix M – Contents of CD-ROM

The CD-ROM contains working software, source code and questionnaire raw data.

Directory: Data

This directory contains the raw data from questionnaires in Excel spreadsheets. If your PC

does not have Excel installed, please install the *Excel Viewer* in the *Viewers* directory.

Filename	Description
Anonymised Pilot Initial Questionnaire Results.xls	Results from the pilot version of initial questionnaire (four respondents).
Anonymised Initial Questionnaire Results.xls	Results from the initial questionnaire (36 respondents).
Anonymised Usability of Prototype Questionnaire Results.xls	Results from the questionnaire seeking opinions about the usability of the prototype origami simulator and diagrammer (nine respondents)

Directory: DirectX

Filename	Description
directx_9c_redist.exe	Installer for runtime version of DirectX 9c.

Directory: Prototype

This directory contains three subdirectories: Miyazaki, system32 and Website.

Miyazaki

Contains directory *origami-dx* which has all source code and project files for the prototype

origami simulator and diagrammer.

System32

Contains the prototype *OrigamiSD.exe* executable and associated files. It should be possible

to run the executable directly from the CD-ROM. If this does not work, please install the

DirectX 9c from root directory *DirectX*.

Some of the help file is printed in Appendix H – Help file for redesigned origami simulator,

p. 170.

Website

A website was created to host the prototype, help file, documentation and usability

questionnaire. All files have been reproduced in this directory. Open *index.html* to start

viewing the website.

Directory: Viewers

Filename	Description
gsv46w32.exe	Installation program for GSView, a PostScript viewer. Requires GhostScript
gs814w32.exe	Installation program for GhostScript, a PostScript viewer.
AdbeRdr70_enu_full.exe	Installation program for Adobe Acrobat, a PDF file viewer.
xlviewer.exe	Installation program for Microsoft Excel Viewer, an Excel spreadsheet viewer.

10 References

Adobe Systems Incorporated (1992) *PostScript Language Document Structuring Conventions (DSC) Specification Version 3.0*, URL http://partners.adobe.com/public/developer/en/ps/5001.DSC_Spec.pdf (22 Jun 2005)

Adobe Systems Incorporated (1999) PostScript Language Reference Manual, 3rd ed. Addison-Wesley, Reading, Mass., URL http://partners.adobe.com/public/developer/en/ps/PLRM.pdf (22 Jun 2005)

Adobe Systems Incorporated (n.d.) *SVG*, URL http://www.adobe.com/enterprise/svg.html (2 May 2005)

Apple Computer Inc. (1995) *Macintosh Human Interface Guidelines*, URL http://developer.apple.com/documentation/mac/pdf/HIGuidelines.pdf (8 Sep 2005)

Ariel (1998) *Re: other forms of showing folds, URL* http://origami.kvi.nl/archives/a0037x/arc00378.txt (15 Feb 2005)

Bateman, A. (2005) *Tess: origami tessellation software, URL* http://www.papermosaics.co.uk/software.html (6 Jun 2005)

Blackwell, A.F. and Green, T. R. G. (2000). 'A Cognitive Dimensions questionnaire optimised for users' in A. F. Blackwell and E. Bilotta (eds.) *Proceedings of the Twelfth Annual Meeting of the Psychology of Programming Interest Group*, p.137-152, URL http://ppig.org/papers/12th-blackwell.pdf (19 Jan 2005)

Blackwell, A. F. and Green, T. R. G. (2003) 'Notational Systems – The Cognitive Dimensions of Notations Framework' in J. M. Carroll (ed.) (2003) *HCI models, theories, and frameworks : toward a multidisciplinary science,* Morgan Kaufmann, Amsterdam, p. 103-133

British Origami Society (2005) *Model Collection: Nottingham Spring 2005,* British Origami Society, Lymm, Cheshire

Carroll, J. M. (ed.) (2003) *HCI models, theories, and frameworks: towards a multidisciplinary science,* Morgan Kaufmann, Amsterdam

Chin, J.P., Diehl, V. A. and Norman K. L. (1988) 'Development of a Tool Measuring User Satisfaction of the Human-Computer Interface' in *Proc. SigChi '88,* ACM, Washington, D.C. p. 213 - 218

Cockton, G., Woolrych, A., Hall, L. and Hindmarch, M. (2004) 'Changing Analysts' Tunes: the surprising impact of a new instrument for usability inspection method assessments' in E. O'Neill, P. Palanque, and P. Johnson (eds.) *People and Computers XVII – Designing for Society, Proc. HCI*, Springer-Verlag, London, p.145-161

Cunliffe, J. (1988) 'First Steps in Origami Diagrammatic Techniques', *British Origami*, No. 133, Dec 1988, p. 2-3

Cunliffe, J. (1989a) 'First Steps in Origami Diagrammatic Techniques – Part Two', *British Origami*, No. 134, Feb 1989, p.18-20

Cunliffe, J. (1989b) 'First Steps in Origami Diagrammatic Techniques – Part Three', *British Origami*, No. 135, Apr 1989, p.11-14

Cunliffe, J. (1989c) 'First Steps in Origami Diagrammatic Techniques – Part Four (Final)', *British Origami*, No. 136, Jun 1989, p.30-32

Dix, A., Finlay, J., Abowd, G. D. and Beale, R. (2004). *Human-Computer Interaction (3rd ed.),* Pearson/Prentice Hall, Harlow

Fisher, D. (1994) *Origami On Computer*, URL http://origami.kvi.nl/articles/thesis.ps (22 Jun 2004)

Foley, J., van Dam, A., Feiner, S. and Hughes, J. (1997) *Computer Graphics: Principles and Practice (2nd ed. in C),* Addison-Wesley, Reading, Massachusetts. [Reprinted with corrections 1997]

Glassner, A. (1996) 'More origami solids' *IEEE Computer Graphics and Applications* Vol. 16, Issue 5 (Sept 1996) p. 81-85

Gout, J. (2001) *Doodle*, URL http://doodle.sourceforge.net/resources.html (4 Feb 2005)

Gray, W.D. and Saltzman, M. C. (1998) 'Damaged merchandise? A review of experiments that compare usability evaluation methods', *Human Computer Interaction* 13, (1998), 203-261.

Green, T.R.G. and Blackwell, A.F. (1998) *Cognitive Dimensions of Information Artefacts: a tutorial Version 1.2 October 1998*, URL

 http://www.cl.cam.ac.uk/~afb21/CognitiveDimensions/CDtutorial.pdf (9 Feb 2005)

Harbin, R. (1956) *Paper Magic: The Art of Paper Folding,* Oldbourne Press, London

Harbin, R. (1963) *Secrets of Origami Old and New,* Oldbourne Book Company, London

Harbin, R. (1974) *Origami: A Step-by-Step Guide,* Hamlyn, London

Harbin, R. (1975) *Origami: An Illustrated Teach Yourself Book,* Knight Press, Leicester

Hatori, K. (n.d.) *Origami Construction*, URL
http://www.jade.dti.ne.jp/~hatori/library/conste.html (14 Feb 2005)

Hull, T. (1995) *Re: Mathematica, URL* http://origami.kvi.nl/archives/a0009x/arc00099.txt
(15 Feb 2005)

Ida, T., Marin, M and Takahashi, H. (2003) 'Constraint functional logic programming for
origami construction' in *Programming Languages And Systems, Proceedings, Lecture Notes
In Computer Science 2895* p. 73-88, URL http://www.risc.uni-
linz.ac.at/people/buchberg/papers/2003-10-00-B.pdf (27 Jan 2005)

Ida, T., Tepeneu, D., Buchberger, B. and Robu, J. (2004) *Proving and constraint solving in
computational origami,* URL
http://www.score.is.tsukuba.ac.jp/~ida/Ida2004/AISC2004.pdf (27 Jan 2005)

Ida. T and Buchberger, B. (2004) *Proving and Solving in Computational Origami*, URL
http://www.score.is.tsukuba.ac.jp/~ida/Ida2004/AISC2004.pdf (27 Jan 2005)

Ilsley, S. (2003) *Welcome to the Virtual Origami Experiment,* URL
http://web.archive.org/web/20030929093900/http://mmedia.chelt.ac.uk/students/mi451/s92
50399/ins.asp (21 Feb 2005)

Ivory, M. Y. and Hearst, M. A. (2001) 'The state of the art in automating usability evaluation
of user interfaces'*, ACM Computing Surveys* Vol. 33, Issue 4 (December 2001) p. 470 - 516

Jackson, P. (1989) 'Letters', *British Origami*, No. 137, Aug 1989, pp. 24-25.

Japan Origami Academic Society (2003) *Origami Tanteidan Convention Book vol. 9*, Japan Origami Academic Society, Tokyo

Ju, W., Bonanni, L., Fletcher, R., Hurwitz, R., Judd, T., Post, R., Reynolds, M. and Yoon, J. (2002) 'Exhibits: Origami Desk: integrating technological innovation and human-centric design' in *Proceedings of the conference on Designing interactive systems: processes, practices, methods, and techniques,* ACM, London, p. 399-405

Kato, J., Watanabe, T. and Nakayama, T. (1997) 'Recognition of essential folding operations: a step for interpreting illustrated books of origami' in *Document Analysis and Recognition, 1997. Proceedings of the Fourth International Conference on , Volume: 1 , 18-20 Aug. 1997* p.81 – 85

Kato, J., Watanabe, T., Nakayama, T., Guo, L. and Kato, H (1998) 'A model-based approach for recognizing folding process of origami' in *Proceedings of the Fourteenth International Conference on Pattern Recognition, 1998, Volume: 2 , 16-20 Aug. 1998,* p.1808 - 1811

Kishi, N. and Fujii, Y. (1998) **'**Origami, folding paper over the Web' in *Computer Human Interaction, 1998. Proceedings. 3rd Asia Pacific, 15-17 July 1998*, Kangawa, Japan p. 337 – 342

Koshak, R. (2003) *Automated Crease Pattern Generation: Applying Transdisciplinary Engineering to Software Design, A Master of Engineering Report*, URL http://www.mythsearch.com/Koshak-TTU_Master_of_Engineering_Report.pdf (4 Jan 2005)

Lang, R. (1989a) 'Because It's There: Computers and Folding: 1', *British Origami*, No. 135, Apr 1989, pp16-18.

Lang, R. (1989b) 'Because It's There: Computers and Folding: 2', *British Origami*, No. 136, Jun 1989, pp12-16.

Lang, R. (1989c) 'Letters', *British Origami*, No. 139, Dec 1989, p. 29.

Lang, R. (1991) 'Because It's There: Idiot Savant' *British Origami,* 148 (June 1991), URL

http://www.britishorigami.org.uk/practice/highlite/140-49.htm#148 (15 Feb 2005)

Lang, R. (1996) *Re: "Visual/Virtual" diagramming application,* URL

http://origami.kvi.nl/archives/a0017x/arc00172.txt (15 Feb 2005)

Lang, R. (2000) *Origami Diagramming Conventions: A Historical Perspective*, URL

http://origami.kvi.nl/articles/diagram.pdf (22 Jun 2004)

Lang, R. (2003a) *Origami Design Secrets: Mathematical Methods for an Ancient Art*, A.K. Peters, Natick, MA

Lang, R. (2003b) *ReferenceFinder - a program for finding efficient folding sequences for locating reference points and lines in a unit square,* URL

http://origami.kvi.nl/programs/reffind/referencefinder-3.0.zip (2 Jul 2004)

Lang, R. (2005a) *Huzita-Hatori axioms*, URL

http://www.langorigami.com./science/hha/hha.php4 (23 Mar 2005)

Lang, R. (2005b) *Origami Simulation*, URL

http://www.langorigami.com./science/origamisim/origamisim.php4 (23 Mar 2005)

Lavoie, C. (n.d.) *Axiomatic Origami -- or the Mathematical backbone of paper folding*, URL

http://cgm.cs.mcgill.ca/~athens/cs507/Projects/2002/ChristianLavoie/maths.html (14 Feb 2005)

Law, E. L. and Hvannberg, E. T. (2004) 'Analysis of strategies for improving and estimating the effectiveness of heuristic evaluation', *Proceedings of the third Nordic conference on Human-computer interaction*, ACM, Tampere, Finland, p. 241-250

Leventhal, L. (2001) 'Tools and techniques for interaction: Delivering instructions for inherently-3D construction tasks: lessons and questions for universal accessibility' in *Proceedings of the 2001 EC/NSF workshop on Universal accessibility of ubiquitous computing: providing for the elderly,* ACM, Alcácer do Sal, Portugal, p. 51-55

Lister, D. (2003) *Errors and misconceptions about the history of paperfolding*, URL
 http://www.britishorigami.org.uk/academic/lister/errors.htm (5 Apr 2005)

Lister, D. (n.d) *Origami Diagramming,* URL
 http://www.britishorigami.org.uk/theory/lister/diagrams.htm (31 Jan 2005)

Microsoft Corporation (2004a) *Official Guidelines for User Interface Developers and Designers,* URL http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwue/html/welcome.asp (2 Sep 2005)

Microsoft Corporation (2004b) *Menus,* http://msdn.microsoft.com/library/en-us/dnwue/html/ch08b.asp (2 Sep 2005)

Miyazaki, S., Yasuda, T., Yokoi, S. and Toriwaki, J. (1992) 'An interactive simulation system of origami based on virtual space manipulation' in
Robot and Human Communication, 1992. Proceedings., IEEE International Workshop on, 1-3 Sept. 1992 p. 210 – 215

Miyazaki, S.Y., Yasuda, T., Yokoi, S. and Toriwaki, J. I. (1996) 'An origami playing simulator in the virtual space', *Journal of Visualization and Computer Animation,* 7 (1): 25-42 Jan-Mar 1996, URL http://www.om.sccs.chukyo-u.ac.jp/main/research/origami/journal/jvca.html (22 June 2005)

Miyazaki, S. (2004) *Origami Simulation, URL* http://www.om.sccs.chukyo-u.ac.jp/main/index_e (7 Feb 2005)

Nielsen, J. (1994) *Usability Engineering*, AP Professional, Boston

Nimoy, J. (2002) *Making Origami Instructional Symbolics Interactive*, URL http://www.jtnimoy.com/itp/origami/ (7 Feb 2005)

Nordal, R. (2001) *Origami Surprise*, URL http://www.britishorigami.org.uk/fun/surprise.htm (31 Jan 2005)

Norman, D.A. (1988) *The Psychology of Everyday Things,* Basic Books, USA.

Novick, L. and Morse, D.L. (2000) 'Folding a fish, making a mushroom: The role of diagrams in executing assembly procedures', *Memory & Cognition*, 28(7), 1242-1256.

Perlman, G. (2001) *Web-Based User Interface Evaluation with Questionnaires, URL* http://www.acm.org/~perlman/ (28 Feb 2005)

Petty (n.d.) *Diagramming,* URL http://members.aol.com/ukpetd/origami_diagramming.htm (31 Jan 2005)

Petzold, C. (1992) *Programming Windows 3.1*, Microsoft Press, Redmond, Washington

Pirhonen, A. (2005) 'To simulate or to stimulate? In search of the power of metaphor in design' in A. Pirhonen, H. Isomäki, C. Roast and P Saariluoma (eds.) *Future Interactive Design*, Springer-Verlag, London

Preece, J., Rogers, Y. and Sharp, H. (2003) *Interaction Design: beyond human-computer interaction,* John Wiley, New York

Roast, C., Dearden, A. and Khazaei, B. (2004) 'Enhancing Contextual Analysis to Support the Design of Development Tootls' in Fincher, S., Markopoulos, P., Moore, D. and Ruddle, R. (eds.) *People and Computers XVIII – Design for Life, Proc. HCI*, Springer-Verlag, London, p298-313

Robinson, N. (2004) *Valley and Mountain folds*, URL
 http://www.britishorigami.org.uk/practice/tech/valley_mountain.htm (31 Jan 2005)

Sears, A. (2003) 'Testing and Evaluation' in J. A. Jacko and A. Sears (eds.) (2003)
The Human-Computer Interaction Handbook: fundamentals, evolving technologies and emerging applications, Lawrence Erlbaum Associates, Mahwah, N.J. p. 1091-1092

Shimanuki, H., Kato, J. and Watanabe, T. (2003) 'Recognition of folding process from origami drill books' in *Document Analysis and Recognition, 2003. Proceedings. Seventh International Conference on* , (3-6 Aug. 2003) vol.1 p. 550 - 554

Shimanuki, H., Kato, J. and Watanabe, T. (2004) 'Constituting origami models from sketches' in *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on* (23-26 Aug. 2004) Vol.1 p. 628 - 631

Shneiderman, B. (2003) 'Foreword' in J.A. Jacko and A. Sears (ed.) (2003) *The Human-Computer Interaction Handbook: fundamentals, evolving technologies and emerging applications,* Lawrence Erlbaum Associates, Mahwah, N.J. p. xv

Shneiderman, B. and Plaisant, C. (2005) *Designing the user interface: strategies for effective human-computer interaction*, Pearson Addison-Wesley, Boston, Mass.

Smith, J. S. (1975) *An Origami Instruction Language* British Origami Society, Lymm, Cheshire (republished 1998), URL http://www.users.waitrose.com/~pureland/oil.htm (7 Feb 2005)

Smith, J. S. (1980) *Pureland Origami* British Origami Society, Lymm, Cheshire (second printing 1985)

Suzuki, T., Kato, J. and Watanabe, T. (2002) 'Extraction of contextual information existing among component elements of origami books', *Graphics Recognition: Algorithms and Applications, Lecture Notes in Computer Science 2390*, p. 158-166

Strothotte, T. (2002) *Non-photorealistic Computer Graphics,* Morgan Kaufmann, San Francisco

Szinger, J. (2001) *The Foldinator Modeler and Document Generator*, URL http://www.zingman.com/foldinator3OSMEpaper.html (24 Jan 2005)

Szinger, J. (2002) 'The Foldinator Modeler and Document Generator' in T. Hull, ed. (2002) *Origami^3: Third International Meeting of Origami Science, Math and Education,* A.K. Peters, Natick, MA, p 129 – 135

Szinger, J. (2005) *Re: origami software*, URL

http://origami.kvi.nl/archives/a0138x/arc01387.txt (20 Apr 2005)

Teng, S. and Mansfield, P. A. (2003) *Virtual Origami*, URL

http://www.svgopen.org/2003/paperAbstracts/VirtualOrigami.html (15 Mar 2005)

van Gelder, M. (2002) *ORIDRAW 5.03*, URL

http://origami.kvi.nl/programs/oridraw/read.me (4 Feb 2005)

Virzi, R. A., Sokolov, J. L. and Karis, D. (1996) 'Usability problem identification using both low- and high-fidelity prototypes' in *Proceedings of the SIGCHI conference on Human factors in computing systems: common ground*, ACM, Vancouver, British Columbia, Canada, p. 236-243

Ward, T.R. (1976) *Origami Pad,* John Adams Toys, Wargrave, Berks.

Whittaker, S., Terveen, L. and Bardi, B.A. (2002) 'Let's stop pushing the envelope and start addressing it: a reference task agenda for HCI', *Human-Computer Interaction*, Vol 15, Nos. 2 & 3, 75-106

Wilkes, D. and Tsotsos, J.K. (1992) 'Active Object Recognition' in *Computer Vision and Pattern Recognition, 1992. Proceedings CVPR '92., 1992 IEEE Computer Society Conference on, 15-18 June 1992* p 136 - 141

Zamiatina, L.I. (1994) 'On Computer Simulation of Origami', *Mathematica in Education*, Vol. 3 No. 3, Summer 1994, URL http://library.wolfram.com/infocenter/Articles/1786/ (14 Feb 2005)

Zimmerman, G., Barnes, J. and Leventhal, L. (2003) 'A comparison of the usability and effectiveness of web-based delivery of instructions for inherently-3D construction tasks on handheld and desktop computers' in *Proceeding of the eighth international conference on 3D Web technology,* ACM, Saint Malo, France, p. 49-54

11 Glossary

BOS   British Origami Society

CD   Cognitive Dimension

CSS   Cascading Style Sheet (Teng and Mansfield, 2003)

CTM   Current Transformation Matrix (PostScript)

ECMAScript   European Computer Manufacturer's Association's standardization for

                       JavaScript and JScript. (Teng and Mansfield, 2003)

fish base   A base is a fold that is a common starting for many origami designs.

                       The fish base the result of making two rabbit's ear to a square (Figure

                       39 shows half a fish base). Figure 38 shows the fish base in two

                       configurations: the left hand diagram may be considered to be a the

                       result of making two outside reverse folds to a square folded along the

                       diagonal.
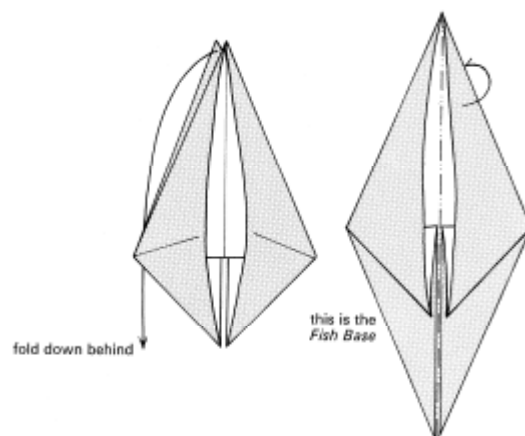
Figure 38 Two versions of the fish base (Harbin, 1975, p. 73)

GOMS   "Goals, Operators, Method and Selection", a predictive HCI

                       evaluation technique.

HCI   Human-Computer Interaction

HE	Heuristic Evaluation
inside reverse fold (procedure)	The first set of diagrams in the second row of Figure 40, p.209, show the standard symbols in step 1, the intermediate step in step 2 and the result in step 3.
mountain fold	A convex fold. Equivalent to a valley fold turned over (Dash-dot line in Figure 40, p.209. See also the first diagram in Figure 2, p. 17)
outside reverse fold (procedure)	The second set of diagrams in the second row of Figure 40, p.209, show the standard symbols in step 1, the intermediate step in step 2 and the result in step 3.
PostScript	A page description language developed by Adobe Systems Incorporated.
pureland origami	Smith (1980, p. 6) first used this term for "simplest folding using only a square" in 1978. He lists four rules:

1. Only a square may be folded
2. Only a single mountain or valley fold is allowed in each step. (unfolding or turning over is permissible)
3. "Tucking in or opening up to 3D is acceptable provided no [new] creases are made in the process."
4. "All folds should be exactly locatable."

QUIS	Questionnaire for User Interaction Satisfaction
rabbit's ear (procedure)	A procedure in which the corner is effectively "pinched" into a flap. Figure 39 shows the standard symbols for this procedure in step 1 and the result in step 8. Steps 2 to 7 show the full series of folds for this procedure.
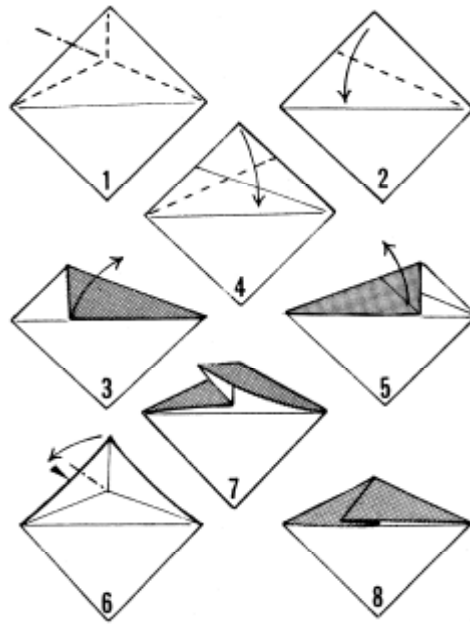
Figure 39 Standard folding symbols and full explanation of rabbit's ear procedure (Harbin, 1963, p. 16)

reverse fold (procedure)	Either an *inside reverse fold* or *outside reverse fold*. Usually taken to be an inside reverse fold if not specified.
SMIL	Synchronized Multimedia Integration Language (Teng and Mansfield, 2003)
SVG	Scalable Vector Graphics. A "W3C standard XML-based imaging model that enables ... users to ... create robust visual content and interactivity through a simple declarative programming model." (Adobe Systems Incorporated, n.d.)
UIDE	User Interface Development Environment
valley fold	A concave fold. Equivalent to turning a page in a book (Dashed line in Figure 40. See also the first diagram in Figure 2, p. 17)
XML	Extensible Markup Language.
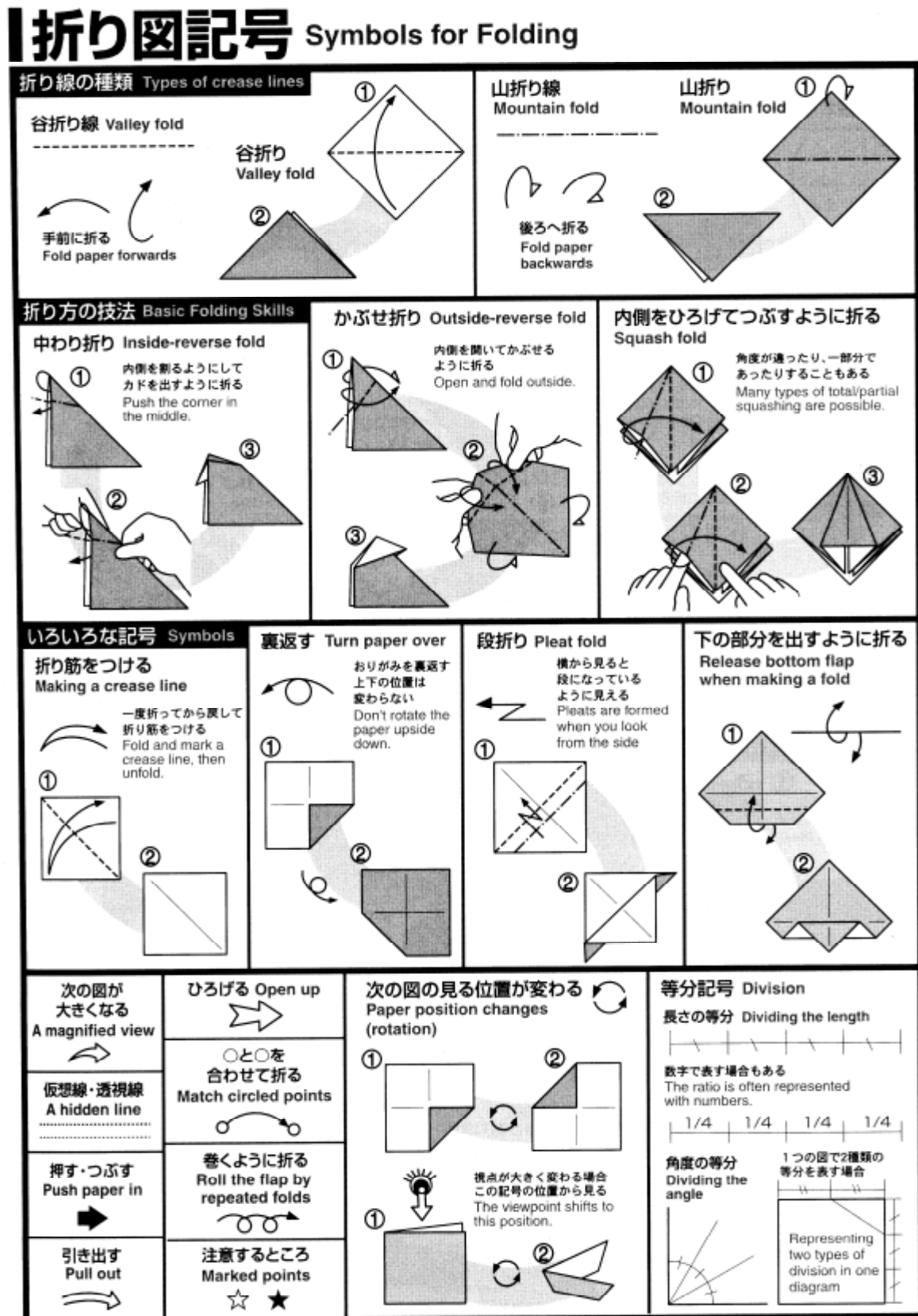
Figure 40 Standard folding symbols explaining reverse fold procedures in second row (Japan Origami Academic Society, 2003, p. 3)

12 Index