Technology
Arts Sciences
TH Köln

Origami Diagramming

Evaluating and Improving the Origami Diagramming Tool Origrammer

Bachelor Thesis

by

Julian Hardtung

at

TH Köln University of Applied Sciences
Campus Gummersbach
Institute of Informatics and Engineering

Course of Studies

Media Informatics

First supervisor: Prof. Dr. Martin Eisemann
TH Köln University of Applied Sciences

Second supervisor: Matthias Groß
TH Köln University of Applied Sciences

Gummersbach, July 15, 2020

Adresses: Julian Hardtung
Lachtstraße 12
51645 Gummersbach
ju.hardtung@gmx.de

Prof. Dr. Martin Eisemann
TH Köln University of Applied Sciences
Institute of Informatics and Engineering
Steinmüllerallee 1
51643 Gummersbach
martin.eisemann@th-koeln.de

Matthias Groß
TH Köln University of Applied Sciences
Institute of Informatics and Engineering
Steinmüllerallee 1
51643 Gummersbach
matthias.gross2@th-koeln.de

Contents

1 Abstract

2 Introduction

Origami is the Japanese art of folding paper into models of animals, people or other objects. Creating instructions for folding these Origami models is a tedious and time consuming task. These so called origami diagrams (see Figure 2.1) have to be accurate representations of the paper for every folding step, in order to unambiguously explain how to fold the model. Every flap, crease and edge has to be drawn (see Section **??** for exceptions), which makes the process slow and especially error-prone for complex models. While for example the crane model in Figure 2.1 can be folded in 17 steps, more complex models can take hundreds of steps to complete.

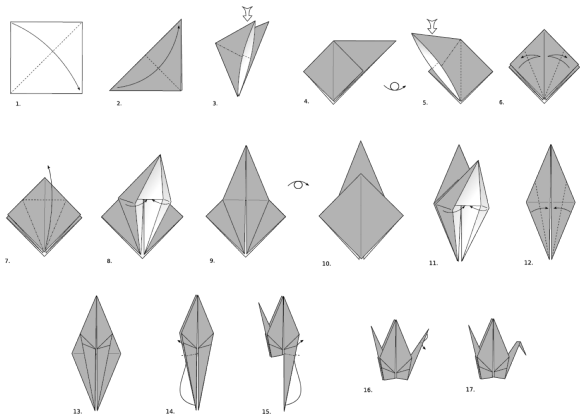


Figure 2.1: Crane Diagram - [Andrew Hudson 2011 [3]]

Historically, diagrams got either drawn by hand or created with the help of digital vector programs like Inkscape[1] or Adobe Illustrator[2]. Even though the digital diagramming improved the accuracy of the finished instructions, the task itself was still quite time consuming especially for models with hundreds of steps.

In contrast to diagramming programs there have been several attempts at creating programs that virtually fold the paper in order to create models. These include amongst others eGami by Jack Fastag [1], Origami Simulation by Robert J. Lang [4] and the Foldinator Project by John Szinger [6]. However none of these projects ever got outside the prototype phase and they haven't been updated in years.

[1] https://inkscape.org/
[2] https://www.adobe.com/de/products/illustrator.html

Up until the point of this publication there is no publicly available program that offers specific features for the origami diagramming process. The previously mentioned vector programs can be used for diagramming, however as they weren't developed with origami diagramming in mind, there are a lot of shortcomings in their feature set. These shortcomings of current programs will be further elaborated on in Section **??**.

The goal for this project is to develop a desktop application that implements features specifically for the origami diagramming process. The standardized symbols and overall notations have to be included and the program has to offer functions that increase the efficiency of creating diagrams.

This work starts with an overview and subsequently a categorization of current origami diagramming notations. Based on these findings, a requirements analysis can be carried out in order to define the required features of the planned program. Any problems or other findings during the development process will be documented so that they may be beneficial to others in the future.

3 Selecting Evaluation Methods

3.1 10 Usability Heuristics by Jakob Nielsen and Rolf Molich

"A disadvantage of the method is that it sometimes identifies usability problems without providing direct suggestions for how to solve them."[5]

The Origrammer should mostly be tailored for professionals that are already well versed in the area of Origami. There is a lot of initial knowledge required when developing new Origami models and subseuently diagrams. This is the main reason why novices most likely won't be using the Origrammer. But that still means that knowledgeable Origami artists should be assisted where possible through Origrammers featureset.

4 Evaluating Origrammer

To begin the evaluation process the 10 usability heuristics by Jakob Nielsen [5] will be used in order to facilitate a base, on which can be build upon with other evaluation methods if required. As established in 3, the Usabilty Heuristics sometimes only identify problems without providing direct solutions. This is why this chapter will focus on finding problems and shortcomings of the Origrammer first. Afterwards, solutions can be developed that optimally fix most, or all, discovered issues without contradicting or counteracting other measures.

As the current (at the time of writing this thesis) Covid-19 pandemic hinders user involvement for the evaluation process, other measures have to be taken to ensure maximum efficiency and thoroughness. This is why there is an exhaustive list of all parts and features of the Origrammer below. Figure 4.1 roughly shows what features are located where on the Origrammer.

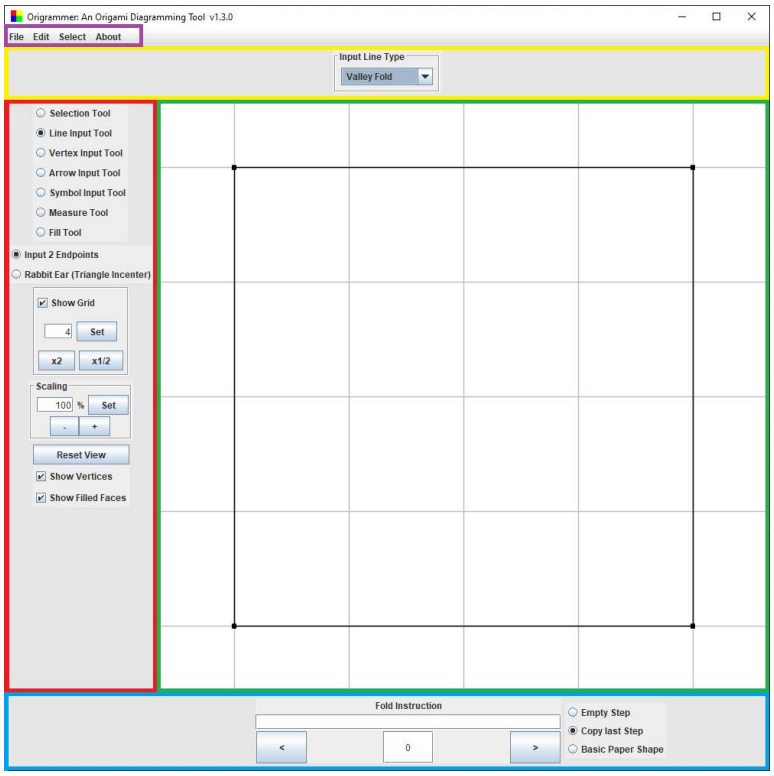


Figure 4.1: Menu Bar (Purple), Top Panel (Yellow), Side Panel (Red), Editing Panel (Green), Navigation Panel (Blue)

4.1 Origrammer Feature List

1. **Menu Bar (Purple)**
   (a) New File (b) Open File (c) Save File
   (d) Export File (e) Model Preferences (f) Origrammer Preferences
2. **Side Bar (Red)**
   (a) Selection Tool
      1. Click to Select 2. Hover over Object 3. Rectangular Selection
   (b) Line Input
      1. Two Point Input 2. Triangle Incenter
   (c) Vertex Input
      1. Absolute Position 2. Fraction of a Line
   (d) Arrow Input
      1. Valley Fold 2. Mountain Fold 3. Turn over
      4. Push Here 5. Pull out 6. Inflate here
   (e) Symbol Input
      1. Leader 2. Repetition Box 3. Next View Here
      4. Rotations 5. Hold Here 6. Hold Here and Pull
      7. X-Ray Circle *8. Fold Over & Over* 9. Equal Distances
      10. Equal Angles 11. Crimps 12. Pleats
      13. Closed Sinks
   (f) Measure Tool
      1. Measure Length 2. Measure Angle
   (g) Fill Tool
   (h) Grid Settings
   (i) Scaling Settings
3. **Navigation Panel (Blue)**
   1. Fold Instructions 2. Step Navigation 3. New Step Options
4. **Top Panel (Yellow)**
   See Side Panel for related features, as options for the selected Side Panel Tools appear on the Top Panel.
5. **Editing Panel (Green)**
   See Side Panel for related features, as the selected Side Panel Tools are being used on the Editing Panel.

4.2 10 Usability Heuristics

With the Origrammer Feature List as a basis, the evalutation using the 10 Usability Heuristics can be carried out. Every feature from the list will be checked against all 10 heuristics in order to try and maximise the completeness of the result. The found usability issues will then lead to the planning of potential usability improvements.

1. Visibility of System Status

Nr:	Affects	Impact	Description
01	2.b; 2.d-2.g	3	When an action requires multiple input points (e.g. placing a Fold Line by two endpoints), the user doesn't know where he is in the process.
15	2.d; 2.e	7	Always show a preview of the `OriArrow` or `OriSymbol` before final placing

2. Aesthetic and Minimalist Design

Nr:	Affects	Impact	Description
02	2.a-2.i	4	The text in the Tool Selection on the Side Panel should be replaced by icons
21	4	3	The text in the TopPanel for input and editing options should be replaced by icons
22	2.f; 3	6	The User Interface should always fit properly (is currently not fitting in the SidePanel for MeasureTool & for some settings in the TopPanel)

3. User Control and Freedom

Nr:	Affects	Impact	Description
03	2.a; 2.b; 2.d; 2.e; 2.g	5	User should always be able to cancel an action (e.g. when inputting an object with multiple input points))

4. Consistency and Standards

Nr:	Affects	Impact	Description
04	2.a; 2.b; 2.d; 2.e; 2.g	2	Editing options on the TopPanel should be labeled correctly (consistent naming scheme needed)
05	2.a; 4; 5	6	When selecting and editing Symbols/Arrows make it consistent (e.g. when selecting different OriObject types like OriArrows and OriSymbols at the same time, the TopPanel overfills)
23	1; 2; 3; 4	6	UI elements should have consistent sizes (e.g. "Set" Buttons, `JTextFields`)
24	1; 2; 3; 4	4	The sequential placement of UI parts should be consistent (e.g. for Input and Editing options on the TopPanel, have `JTextFields` first and then `JCheckBoxes` to the right

5. Error Prevention

Nr:	Affects	Impact	Description
06	2.b.2	7	Selecting the same point multiple times breaks the RabbitEar lines
07	2.e.10	8	Selecting points in the wrong order breaks inputs for the EqualAngle symbol
08	2.g	7	Selecting the same point multiple times breaks the FillTool
14	1.a; 1.e;	6	Restrict the input of every JTextBox (e.g. only allow numbers for number inputs)

6. Recognition Rather than Recall

Nr:	Affects	Impact	Description
09	2.f	5	User should not be forced to remember the measured values (currently the measured values are being hidden, once the user selects a different tool from the SideBar)

7. Flexibility and Efficiency of Use

Nr:	Affects	Impact	Description
10	2.h	3	Give shortcuts for the Grid halve/ double buttons
11	2.i	3	Give shortcuts for the ResetView button
12	2.i	3	Give shortcuts for the step by step zoom-in/zoom-out
20	2.b	8	Give more Line Input options to avoid tedious work with grid adjustments
25	2.d; 3; 4	8	Give more flexibility when editing or placing arrows
26	2.e; 3; 4	8	Give more flexibility when editing or placing symbols

8. Recognition, Diagnosis and Recovery from Errors

Nr:	Affects	Impact	Description
13	2.b-2.	5	Error messages for wrong user inputs should be self-explanatory
19	2.b-2.g	4	Error messages for wrong user inputs should explain how to fix the error

9. Help and Documentation

Nr:	Affects	Impact	Description
16	2.a-2.i; 3; 4	6	Show tooltips for all icons or non self-explanatory parts
17	2.a-2.g	7	Show short explanation on how inputs work for every possible input feature (could be combined with Nr.01 –> 4.2 Visibility

10. Match between System and Real World

Nr:	Affects	Impact	Description
18	2.b-2.e	5	Use “Origami terminology” everywhere (e.g. Rabbit Ear instead of Triangle Incenter; Next View Here Symbol instead of Eye Symbol)

5 Planning Solutions

This section will build on the discovered usability issues found during the initial evaluation process (see 4). On this basis, first solutions can be planned that fix the most impactful usability issues. Furthermore, new features can be developed that increase productivity (by automating processes and part of the workflow), while also fixing the current problems.

5.1 Graphical User Interface related changes

These changes include everything that is related to the graphical user interface that the user interacts with.

5.1.1 Overall UI Changes

Every feature area (e.g. grid options, scaling options, arrow/symbol inputs etc.) should be seperated through a `EtchedBorder` with its name as the title. Within these borders, elements should follow a consistent order. This is why this hierarchy is being set for the placement within these areas: `JRadioButton, JTextField, JButton, JCheckBox`.

Though additional `JLabels` can be used to explain or to give indicators what the elements do specifically. Whenever possible, icons should replace text and tooltips should be used to give further explanation. Explanatory `JLabels` should only be used when icons and tooltips are not enough.

Additionally to the more consistent placement of elements within the UI, their sizes should be standardized as well. `JTextFields` for example should only be large enough to encapsulate the largest possible entry. This means that the `height` should always be set to 25 pixels (and the `width` depending on what is being entered (e.g. entering an angle between 0-360° will be shorter than entering the textual explanation of a folding step).
Fixes: 02; 04; 21; 22; 23

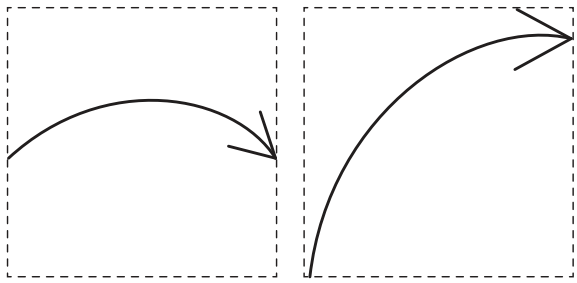
At a later point a complete redesign and rework of the UI will have to be carried out, in order to facilitate the goal of a unified and obstructionless design. At the current stage of development though the focus is being set to increase the efficiency and speed of creating origami diagrams, as this is the main reason why the Origrammer exists.

5.1.2 Arrows & Symbols

A large part of the Origrammer is made up of the different arrows and symbols that can be used in diagrams. So far, these objects were simple vector graphics that were being placed on the diagram as `ImageIcons` of `JLabels`. This approach did initially work, but brought restrictions and problems with it.

As vector graphics do not have explicit width or height values, the library Batik[2], which was being used to load the .svg files, presented wrong values to the `JLabel.setBounds(width, height)`-method. As a result of this limitation, the arrows & symbols got partially cut off at the original bounds of the `JLabel` when rotating them. To fix this issue, a new, pre-rotated vector graphic was being loaded whenever an arrow or symbol got rotated. Though this in turn facilitated itself in a wrong, always square border around the arrows and symbols. Additionally this made interactions with arrows and symbols far slower and unflexible.

Another sideeffect was a change in scale when rotating a non square object. The `JLabel` tries to display the biggest possible object that can fit within the border bounds. As seen on Figure 5.1 the size of the arrow changes after rotating it by 45°.



(a) Straight Arrow (b) Angled Arrow

Figure 5.1: Unwanted Scaling when Rotating

As a result of all these problems and limitations, the decision was made to completely rework how arrows & symbols function in the Origrammer. The new approach was to rebuild all objects with simple `java.awt.Shape`-objects. This gave total control over rotations, scaling, and on-the-fly-editing of all arrows/symbols. Another advantage was the removed reliance on `JLabels` and associated with that, there were no longer issues with wrong

hitboxes or different scaling while rotating. The only disadvantage was the work-intensive nature of remodeling all arrows and symbols with `Shape`-objects by hand.
Fixes: 15; 25; 26

5.1.3 Side Panel

The text of the `JRadioButtons` at the tool bar should be replaced by self-explanatory icons in order to improve overall clarity. But the textual explanation of all the tools should still be present to help especially novizes. This can be done through tooltips that appear when hovering over the buttons.

Fixes: 02; *22*

5.2 User Input related changes

5.3 New Origrammer features

These new features should increase the productivity and efficiency of the Origrammer. The focus should be set on maximising the work that can be done in the shortest amount of time. This will be achieved by implementing features that either automate parts of the workflow, or that give new, faster possibilities of achieving the goal.

6 Evaluating Solutions

7 Problems & other Findings

This section discusses decisions and problems during the development process.

8 Prospect

Glossary

Origami (jpn: *ori* = *folding* and *kami* = *paper*) is the art of folding paper into models of animals, people or other objects. 5

List of Figures

References

[1] Jack Fastag. egami: Virtual paperfolding and diagramming software. In Robert J. Lang, editor, *Origami 4*, pages 273–284. A K Peters, Ltd., 2006.

[2] The Apache Software Foundation. Apache™ batik svg toolkit, May 2020. https://xmlgraphics.apache.org/batik/ [Online; accessed July 07, 2020].

[3] Anrew Hudson. Crane, 2011. https://ahudsonorigami.files.wordpress.com/2011/11/tsuru.pdf [Online; accessed March 21, 2020].

[4] Robert J. Lang. Origami simulation, 2015. http://www.langorigami.com/article/origami-simulation [Online; accessed March 21, 2020].

[5] Jakob Nielsen. 10 usability heuristics for user interface design, April 1994. https://www.nngroup.com/articles/ten-usability-heuristics/ [Online; accessed June 10, 2020].

[6] John Szinger. The foldinator project, 2009. http://zingman.com/origami/foldinator.php [Online; accessed March 21, 2020].