Interactive design of complex objects from thin shells

Linus Wigger

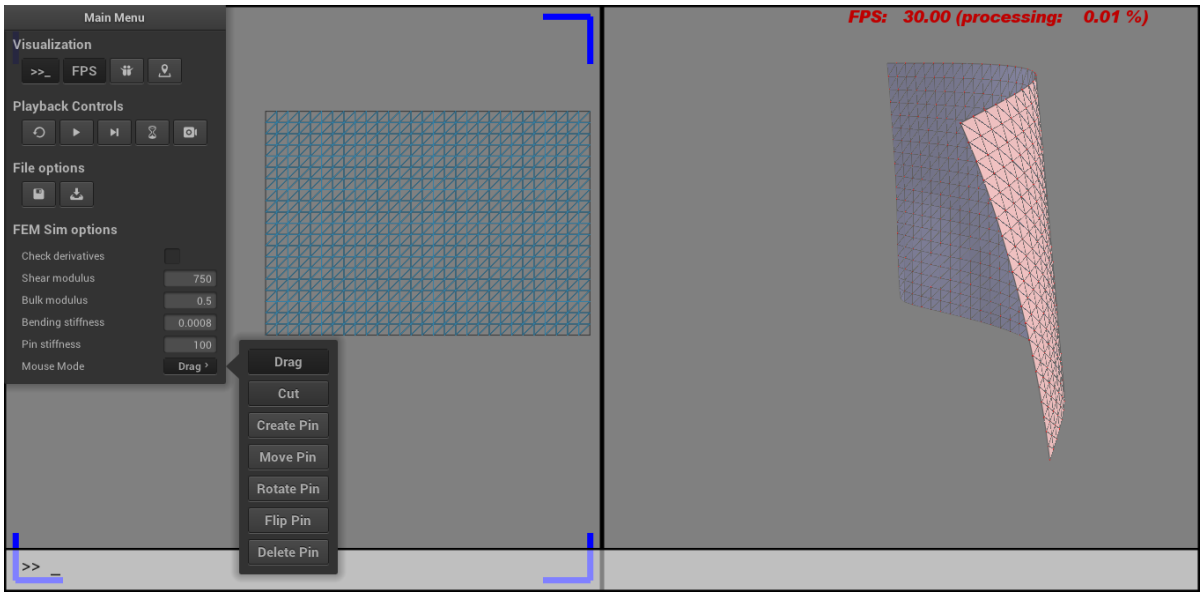

Figure 1: 2D template and 3D simulation side by side

Bachelor Thesis ETH Zurich
Supervisor: Prof. Dr. Stelian Coros
July 2018

1 Abstract

The goal of this project was the creation of an application for designing 3D objects from a sheet of paper, similar to [1] in terms of interface. Design is done using basic operations like dragging and cutting while the screen displays both the two-dimensional rest shape of the paper as well as a simulated three-dimensional deformed state. For the simulation, the paper is modelled as a thin shell structure as seen in [2]. In most cases, the resulting shape is a good approximation for the same object crafted from a physical paper.

2 Paper model

The deformed shape of the paper is obtained using the finite element method. More specifically, an energy function defined on a mesh representing the paper and additional constraints must be minimized. The optimization algorithm uses Newton's Method, which requires knowledge of the gradient and Hessian of the energy function. The mesh consists of nodes (vertices) and elements of different types connecting the nodes. The current configuration of the mesh is defined by the node positions. The mesh energy function is the sum of all values of energy functions defined for each element. There are three main element types: Triangles, edges, and pins. There are also simple springs used to move nodes to specific positions while dragging.

2.1 Triangle elements

Triangles are used as the basic building block for the paper surface. Each triangle is a constant strain triangle element embedded in 3D space. The strain is computed using the Saint Venant-Kirchhoff material model. Triangle elements in 2D were already implemented in the code provided to me and the change from two to three dimensions is small, therefore only the most important differences are highlighted below.
In the following equations, $\overline{A}$ is the rest area of the triangle, $\mathbf{x}$ contains the current 3D positions of the triangle corners, and $\overline{\mathbf{x}}$ contains the 2D positions of the triangle corners in the rest configuration.

$$\mathbf{x} = \begin{bmatrix} x_0 & y_0 & z_0 & x_1 & y_1 & z_1 & x_2 & y_2 & z_2 \end{bmatrix}^T$$

$$\overline{\mathbf{x}} = \begin{bmatrix} \overline{x}_0 & \overline{y}_0 & \overline{x}_1 & \overline{y}_1 & \overline{x}_2 & \overline{y}_2 \end{bmatrix}^T$$

The internal energy E_s of a triangle element can be computed as follows.

$$E_s(\mathbf{x}) = \overline{A}(\frac{\lambda}{2}\, Tr(E)^2 + \mu\, Tr(E^T E))$$

Where λ and μ are the bulk modulus and shear modulus of the material respectively. The Green strain E depends on the deformation gradient F. F changes between 2D and 3D.

$$E = \frac{1}{2}(F^T F - I)$$

$$F_{2D} = \begin{bmatrix} x_1 - x_0 & x_2 - x_0 \\ y_1 - y_0 & y_2 - y_0 \end{bmatrix} \begin{bmatrix} \overline{x}_1 - \overline{x}_0 & \overline{x}_2 - \overline{x}_0 \\ \overline{y}_1 - \overline{y}_0 & \overline{y}_2 - \overline{y}_0 \end{bmatrix}^{-1}$$

$$F_{3D} = \begin{bmatrix} x_1 - x_0 & x_2 - x_0 \\ y_1 - y_0 & y_2 - y_0 \\ z_1 - z_0 & z_2 - z_0 \end{bmatrix} \begin{bmatrix} \overline{x}_1 - \overline{x}_0 & \overline{x}_2 - \overline{x}_0 \\ \overline{y}_1 - \overline{y}_0 & \overline{y}_2 - \overline{y}_0 \end{bmatrix}^{-1}$$

It should be noted that the right part of F_{3D} is still a 2×2 matrix. This is because of a special property that the simulated paper has: The sheet is completely flat in the rest state and therefore it lies in a plane. Choosing the xy-plane for that plane means that the values of $\overline{z}_0$, $\overline{z}_1$ and $\overline{z}_2$ are 0 and can be ignored (because of this, they are not in $\overline{\mathbf{x}}$). That way, the inverse matrix from the 2D case can be reused, avoiding any calculations needed for the general case.

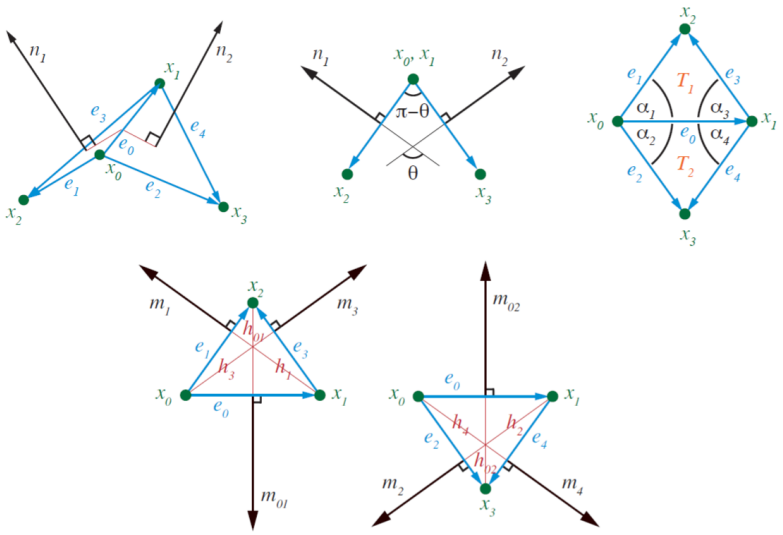

Figure 2: Structure of an edge element (taken from [3])

2.2 Edge elements

In order to compute the bending energy E_b of the deformed paper sheet, the approach from [3] is used. [3] is an older version of [4] and uses a slightly different notation, in particular, the node indices are different. In [3], one version of the bending energy for one mesh edge is

$$E_b(\mathbf{x}) = k_b \frac{3||\overline{\mathbf{e}}_0||^2}{\overline{A}}(\theta - \overline{\theta})^2$$

where k_b is the bending stiffness, $||\overline{\mathbf{e}}_0||$ is the length of the edge in the rest state, $\overline{A}$ the sum of the rest areas of the two triangles adjacent to the edge, θ is the current bend angle and $\overline{\theta}$ is the bend angle in the rest configuration. In this case, $\overline{\theta}$ is always zero because the rest shape is flat. The vector $\mathbf{x}$ contains the coordinates of the four nodes defining the edge element (x_0 and x_1 on the actual edge, x_2 and x_3 being additional triangle corners).

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_0^T & \mathbf{x}_1^T & \mathbf{x}_2^T & \mathbf{x}_3^T \end{bmatrix}^T$$

As the bending energy depends on the angle between two triangles connected by the edge, edges on the boundary of the mesh (which are only part of one triangle) do not contribute to the total bending energy of the system. The gradient and the Hessian of the bending energy are derived in [3] and are repeated here. The meaning of the vectors $\mathbf{e}_i$, $\mathbf{n}_i$, $\mathbf{m}_i$ and angles α_i used in these formulas can be seen in figure 2, note that any vector $\hat{\mathbf{v}}$ is the normalized version of $\mathbf{v}$. The derivatives of θ are needed to calculate the derivatives of E_b.

$$k = -k_b \frac{6||\overline{\mathbf{e}}_0||^2}{\overline{A}}$$

$$\nabla_{\mathbf{x}_0}\theta = \frac{\cos\alpha_3}{h_3}\hat{\mathbf{n}}_1^T + \frac{\cos\alpha_4}{h_4}\hat{\mathbf{n}}_2^T$$

$$\nabla_{\mathbf{x}_1}\theta = \frac{\cos\alpha_1}{h_1}\hat{\mathbf{n}}_1^T + \frac{\cos\alpha_2}{h_2}\hat{\mathbf{n}}_2^T$$

$$\nabla_{\mathbf{x}_2}\theta = -\frac{1}{h_{01}}\hat{\mathbf{n}}_1^T$$

$$\nabla_{\mathbf{x}_3}\theta = -\frac{1}{h_{02}}\hat{\mathbf{n}}_2^T$$

The following definitions are used for parts of the Hessian of θ.

$$H_{ij} = \nabla_{\mathbf{x}_j}(\nabla_{\mathbf{x}_i})^T$$

$$M_{ijk} = \frac{\cos\alpha_i}{h_i h_j}\hat{\mathbf{m}}_j\hat{\mathbf{n}}_k^T$$

$$N_{ij} = \frac{1}{h_{0i}h_j}\hat{\mathbf{n}}_i\hat{\mathbf{m}}_j^T$$

$$S(A) = A + A^T$$

$$B_i = \frac{1}{||\mathbf{e}_0||^2}\hat{\mathbf{n}}_i\hat{\mathbf{m}}_{0i}^T$$

The Hessian of θ is then computed as follows.

$$H_{00} = S(M_{331}) - B_1 + S(M_{442}) - B_2$$

$$H_{01} = H_{10}^T = M_{331} + M_{131}^T + B_1 + M_{442} + M_{242}^T + B_2$$

$$H_{02} = H_{20}^T = M_{3.01.1} - N_{13}$$

$$H_{03} = H_{30}^T = M_{4.02.2} - N_{24}$$

$$H_{11} = S(M_{111}) - B_1 + S(M_{222}) - B_2$$

$$H_{12} = H_{21}^T = M_{1.01.1} - N_{11}$$

$$H_{13} = H_{31}^T = M_{2.02.2} - N_{22}$$

$$H_{22} = -S(N_{1.01})$$

$$H_{23} = H_{32}^T = 0$$

$$H_{33} = -S(N_{2.02})$$

$$H(\theta) = \begin{bmatrix} H_{00} & H_{01} & H_{02} & H_{03} \\ H_{10} & H_{11} & H_{12} & H_{13} \\ H_{20} & H_{21} & H_{22} & H_{23} \\ H_{30} & H_{31} & H_{32} & H_{33} \end{bmatrix}$$

Finally, the gradient and Hessian of E_b are

$$-\nabla E_b = k(\theta - \overline{\theta})\nabla\theta$$

$$-H(E_b) = k(\theta - \overline{\theta})H(\theta) + k\nabla\theta^T\nabla\theta$$

This energy is not the most accurate for simulating real paper. As an example, the simulated paper is completely elastic and will always try to go back to its original shape. Real paper will deform permanently if bent too much but in this case, total accuracy is actually undesired: If the user bends the simulated paper and then decides that it isn't what they wanted, going back to a previous step is much easier if the paper doesn't deform permanently. This bending energy is also rather easy to implement and the results are close to the real behaviour, making it a fine choice.

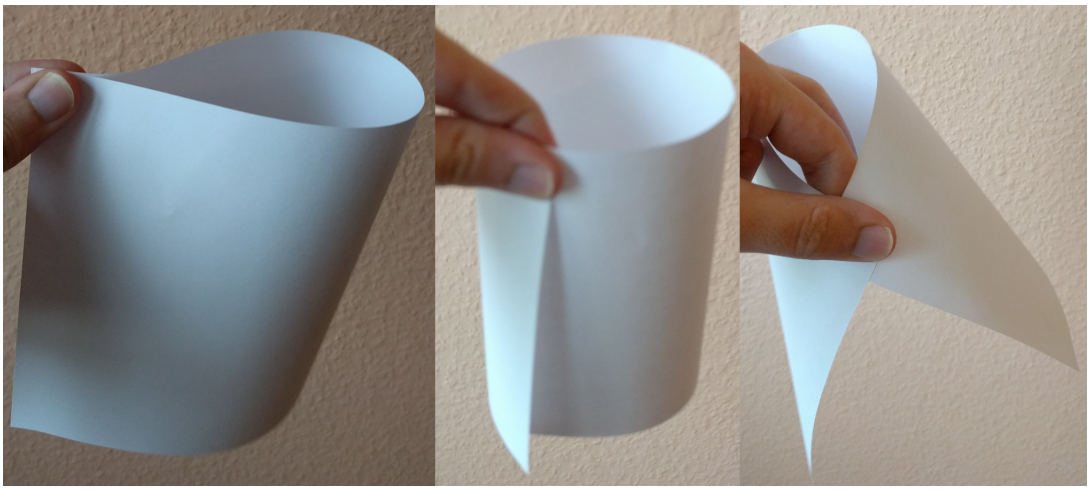

Figure 3: Paper held together in different ways at the same points

2.3 Pins

The user can connect two spots on the paper to each other. This can be thought of as a literal pin, a paperclip, glue or something else. As there are many ways to connect two points on a paper (see figure 3), it is not enough to simply pin two nodes together. It must also be possible to connect arbitrary points on the mesh, not just nodes. The first problem is solved by connecting two sets of three points, defined as the corners of two small but equally sized regular triangles somewhere on the mesh surface. For the second problem, it is possible to map each point to a triangle in the mesh, then represent it using barycentric coordinates. This way, the position of an arbitrary point is obtained as a linear combination of the positions of three nodes. Finally, a pin is modelled as three springs of rest length zero connecting the corners of two triangles on the mesh surface. Given the energy of a zero-length spring as

$$E_p(\mathbf{x}) = \frac{1}{2} k_p ||\mathbf{p}_1 - \mathbf{p}_0||^2$$

Where k is the stiffness and the points $\mathbf{p}_0$ and $\mathbf{p}_1$ are a weighted sum of the node positions in $\mathbf{x}$:

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_{00}^T & \mathbf{x}_{01}^T & \mathbf{x}_{02}^T & \mathbf{x}_{10}^T & \mathbf{x}_{11}^T & \mathbf{x}_{12}^T \end{bmatrix}^T$$

$$\mathbf{p}_0 = w_{00}\mathbf{x}_{00} + w_{01}\mathbf{x}_{01} + w_{02}\mathbf{x}_{02}$$

$$\mathbf{p}_1 = w_{10}\mathbf{x}_{10} + w_{11}\mathbf{x}_{11} + w_{12}\mathbf{x}_{12}$$

The formula for the gradient and Hessian of the energy is

$$\nabla_{\mathbf{0i}} E_p(\mathbf{x}) = w_{0i} k_p (\mathbf{p}_1 - \mathbf{p}_0)$$

$$\nabla_{\mathbf{1i}} E_p(\mathbf{x}) = -w_{1i} k_p (\mathbf{p}_1 - \mathbf{p}_0)$$

$$H_{ij} = k_p w_i w_j I$$

$$H(E_p(\mathbf{x})) = \begin{bmatrix} H_{00.00} & H_{00.01} & H_{00.02} & -H_{00.10} & -H_{00.11} & -H_{00.12} \\ H_{01.00} & H_{01.01} & H_{01.02} & -H_{01.10} & -H_{01.11} & -H_{01.12} \\ H_{02.00} & H_{02.01} & H_{02.02} & -H_{02.10} & -H_{02.11} & -H_{02.12} \\ -H_{10.00} & -H_{10.01} & -H_{10.02} & H_{10.10} & H_{10.11} & H_{10.12} \\ -H_{11.00} & -H_{11.01} & -H_{11.02} & H_{11.10} & H_{11.11} & H_{11.12} \\ -H_{12.00} & -H_{12.01} & -H_{12.02} & H_{12.10} & H_{12.11} & H_{12.12} \end{bmatrix}$$

It is technically possible for two nodes n_{0i} and n_{1j} to be the same, in which case the corresponding gradient and Hessian entries are summed up. In practice however, pins between such close points should never exist.

2.4 Material parameters

As paper is not a well-defined material with widely known material properties, all physical constants used in the simulation were decided by trial and error. The parameter values used are (ignoring units) $\lambda = 0.5$, $\mu = 750$, $k_b = 0.0008$ and $k_p = 100$. μ must be high to prevent the paper from stretching, k_p must be high to make pins matter and k_b is important for the bending behaviour. λ turns out to be irrelevant in most cases (e.g. changing it's value by a factor of 1000 makes almost no visible difference).
The paper is considered to have an area density of 1. The starting mesh is shaped like a rectangle with dimensions 2.1×3.0, approximating an A4 sheet ($210\,\text{mm} \times 297\,\text{mm}$).

3 Operations

This section details the tools available to the user and their effects on the system.

3.1 Dragging

Dragging can be done by grabbing a node with the mouse and moving it around. When doing this, a zero-length spring is created between the selected node and the mouse position. The spring gets deleted again after the node is released, unless the CTRL key is being pressed to keep the node at that position.

3.2 Pin operations

The user can create, edit and delete pins on the paper. The pin is created from two handles defining the position and orientation on the paper sheet. The supported operations for pin handles are "Move", "Rotate" and "Flip". "Move" and "Rotate" are self-explanatory. "Flip" changes the order of the triangle vertices, flipping a triangle ABC leads to a triangle ACB and vice versa. When thinking of a pin as glue, the "Flip" operation changes the side of the paper where the glue is applied.

3.3 Cutting

[h] Making arbitrary cuts on the mesh can become complicated very quickly as mesh elements need to be split, reshaped or replaced. For the sake of simplicity, only straight cuts along mesh edges are allowed here. The user may select a path $[n_0, n_1, ..., n_{n-1}]$ of nodes in the mesh. In a first step, all edge elements on that path are deleted. In the next step, the nodes on the path have to be analyzed. Some of them need to be copied so each side of the cut has enough nodes. It is then necessary to update the local structure of the mesh such that all elements connect to the right nodes. To facilitate this step, there is a data structure (which also gets updated during the cut) storing an ordered list of each node's adjacent nodes.
The number of copies of a node n_i depends on the position of n_i in the path (the endpoints of the path are treated differently) and where on the mesh the nodes n_{i-1}, n_i, and n_{i+1} lie (on the boundary or interior). Figure 4 shows six different cuts. The red node is the current node n_i and the yellow line shows the cutting path.
The full procedure is as follows: If n_i is not on the mesh boundary, then a copy of n_i needs to be created unless n_i is an endpoint of the cutting path (in which case there will be no copy, see the upper left and upper middle cases in figure 4). For n_i on the boundary there can be between zero and two copies. For each of the nodes n_{i-1} and n_{i+1} that exist, a copy of n_i must be created if that node is not on the mesh boundary or if it lies on the boundary, but can't be reached by moving one step along the boundary from n_i (Figure 5 shows this second case). For the update of the local mesh structure, the region around n_i can be partitioned into up to three regions. Each of these regions has

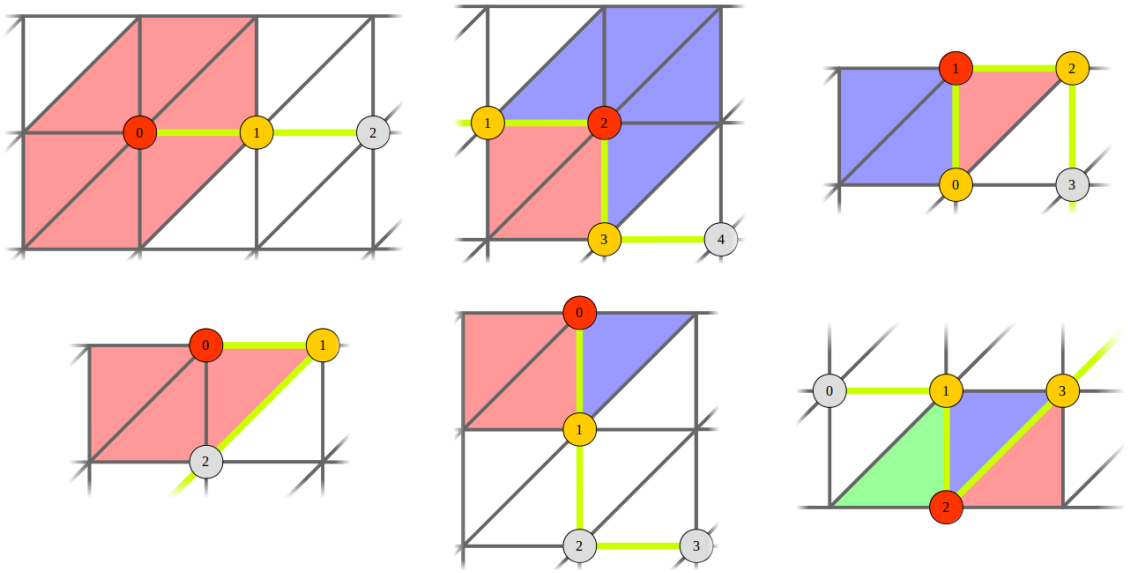

Figure 4: Possible cutting cases

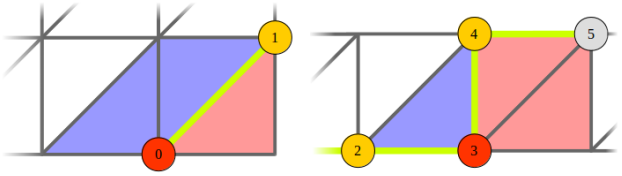

Figure 5: Special cases with n_i and n_{i+1} on the boundary

one instance of n_i associated with it (either the orignal or a copy). The regions are shown in different colours (red, blue, green) in figures 4 and 5. Everything (triangle elements, edge elements, adjacency lists) previously connected to n_i is part of one region and must be connected to the node assigned to that region. The new ordered adjacent node lists of n_i and its copies can be obtained by splitting the old list of n_i, though special care is needed for the region boundaries containing n_{i-1} or n_{i+1}: If those nodes have their own copies, the correct connections can difficult to figure out. In most cases, the rule is to connect copies to copies and originals to originals.
The implementation of this step is done piecewise. Each piece consists of n_{i-1}, n_i, n_{i+1} and an additional input/output node c. As an input, c is either a copy of n_{i-1} or nothing. If the next piece needs to know of a copy of n_i, c will become that copy as an output of the current piece and input to the next. The case where n_i gets copied but c is set to nothing can be seen in the upper right picture of figure 4 (assuming the copy of n_i belongs to the blue region). The adjacency lists of n_{i+1} are not altered (that will happen in the next piece), but those of n_i, n_{i-1} and it's copy c are updated according to the new local structure.

4 Application description

The final application displays the original 2D mesh on the left and the simulated 3D paper on the right, as seen in figure 1. A menu on the left is used for selecting the editing tools and also allows changes in the material constants used for the simulation. Not all tools can be used in both views: The "Drag" option is used to position nodes in the right window and only moves the camera on the left side. All pin operations ("Create", "Move", "Rotate", "Flip", "Delete") only work with the 2D

view because of how the pins are handled internally. Cutting can be done in either window. Some hints on how to avoid commonly occurring problems:

- Hold CTRL while dragging to permanently move a node to some position. To free the node again, drag and release it without pressing CTRL.
- As the cutting process ignores pins, pins may try to connect incorrect parts of the paper if a cut is made nearby. It is recommended to only add pins after cutting. Cuts near any pin endpoints should be avoided unless the pin is manually updated afterwards (by changing it in any way).
- The initial 3D configuration is slightly curved. Letting the simulation run before making any changes may lead to a degenerate case where the entire paper sheet lies in one plane and the existence of the third dimension is ignored.
- Situations where a triangle becomes flipped can usually be resolved by dragging the ends of the problematic region apart. Additionally, these situations are easier to avoid if there are no paper strips with a width of only one triangle.

5 Results

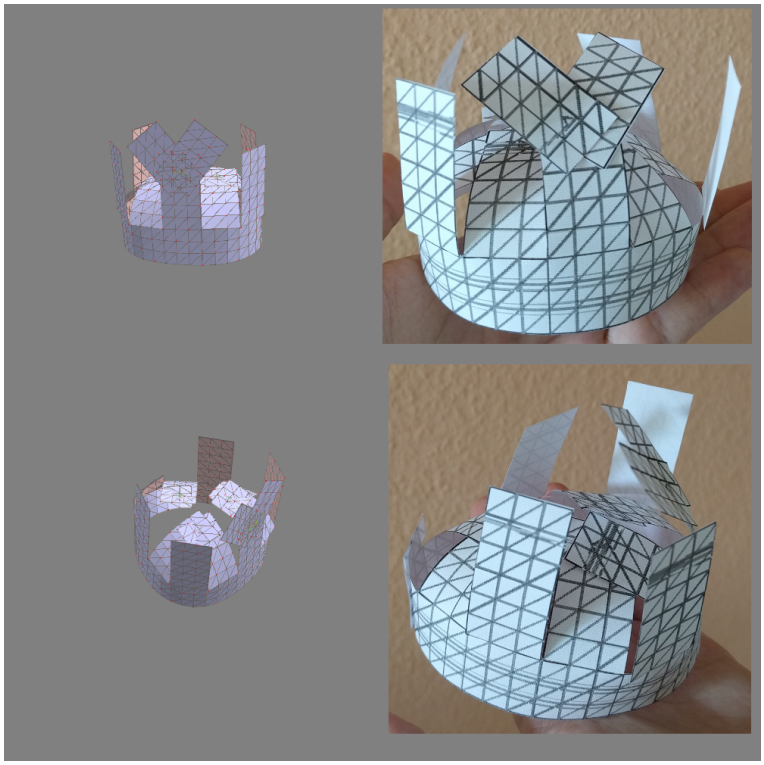

Figure 6: Paper crown

Using the tools described above, I designed multiple objects, then recreated three of them in real life using paper and glue. The simulated and real objects are shown in figures 6, 7 and 8. They are a crown, inspired by the ones that come with (Swiss) three kings cakes, the head of an animal, described as resembling a goat by various people, and a boat with a large sail. The crown was created from a printed screenshot of the 2D view, the other two from templates measured and drawn by hand.

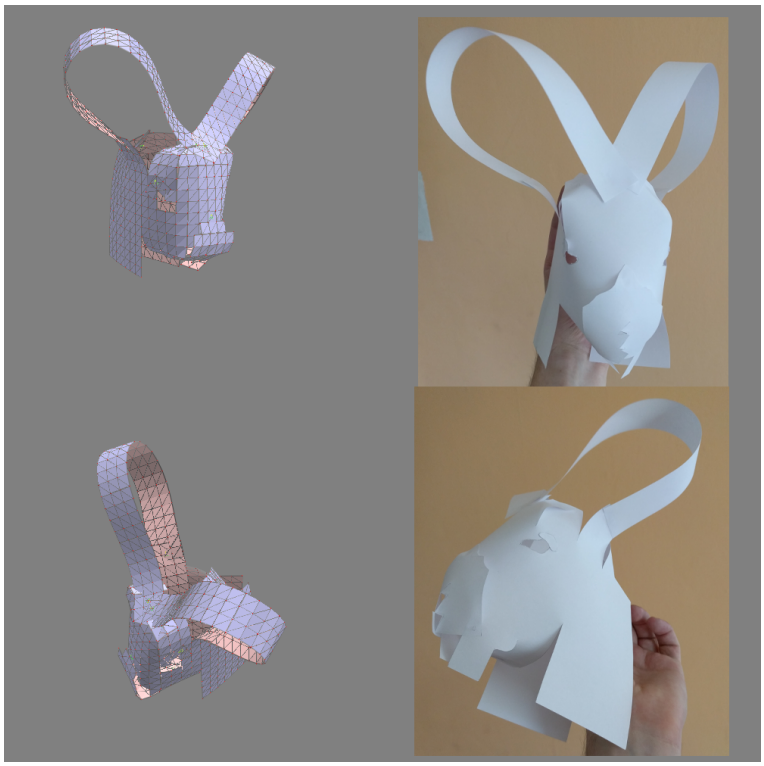

Figure 7: Paper animal head

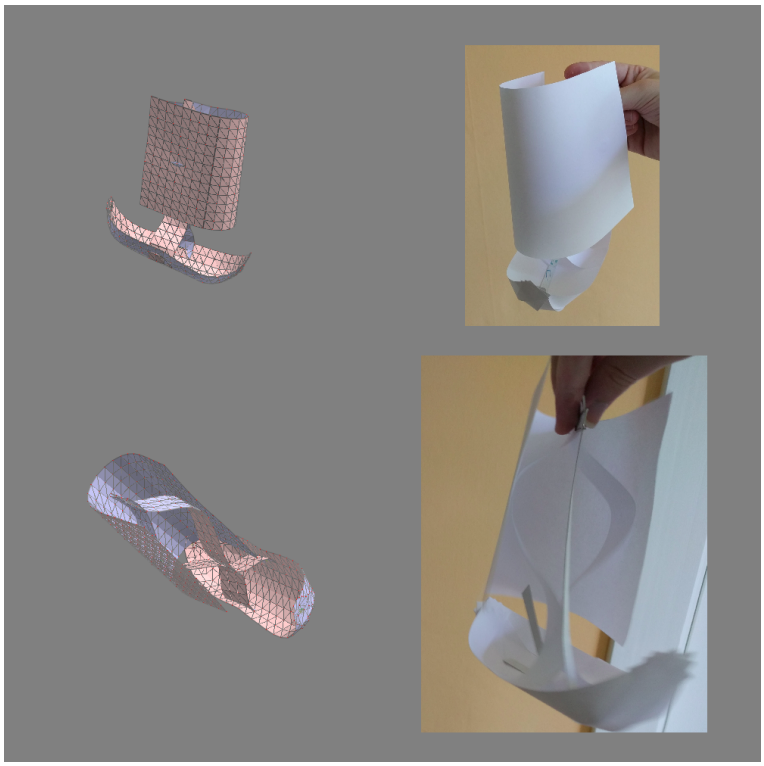

Figure 8: Paper ship

6 Discussion

Looking at the three objects, the most important features can be recognized in both versions, which is a good sign. Still, the physical versions highlighted multiple problems.
The ship was hit particularly badly. Its sail is by far the biggest part and as the mast is made from paper too, it always falls over. The reason this wasn't discovered sooner is because the simulation ignores gravitational forces.
Various small details that are different between the simulated and real objects can be attributed to different issues. The shape of the nose/mouth of the animal differs mostly because of the resolution of the mesh, which isn't high enough for that kind of detail. The horns/ears have a different curvature mostly due to gravity but it might also be a sign that the material parameters (mainly k_b) should be revised.
All three objects (but mostly the ship) have pieces of paper passing through each other in the simulation, something which obviously doesn't happen in real life. Another reason for differences is shoddy craftsmanship. There are at least two spots on the animal head that I didn't glue together in the same angle as in the original.
Fortunately, most of these inaccuracies are not severe or can be worked around, making the application suitable for at least a rough draft of complex designs.

7 Future work

Two obvious extensions of the project are arbitrary cuts and collision detection.
Currently, only cuts along mesh edges are possible. The algorithm for applying cuts would have to be rewritten almost completely for cuts in any shape. It would require rebuilding the mesh in a way that fits the shape and without increasing the number of elements too much.
Concerning collisions, there is currently nothing to avoid self-intersections directly. As long as the simulated object is made of paper, this still works reasonably well, as the intersections seen in figure 8 are negligible. But as soon as stiffer materials are used, those annoyances turn into problems that have to be solved efficiently.
There are more things that can be done: The code is not optimized and reaching real-time performance for larger meshes could be a future goal. The user interface could be improved in various ways like extending pin operations to the 3D view or adding an "Undo" button. A relatively important feature that is currently missing is a sort of "Export to print" command. When trying to reconstruct designs in real life, there should be a way to print a template without visible mesh edges and with usable markings for the pins (e.g. numbered arrows instead of triangles with lines between them).

References

[1] Nobuyuki Umetani, Danny M. Kaufmann, Takeo Igarashi, Eitan Grinspun, 2011, Sensitive Couture for Interactive Garment Modeling and Editing. *ACM Trans. Graph. (Proc. SIGGRAPH) 30*, 4.

[2] Eitan Grinspun, Anil N. Hirani, Mathieu Desbrun, Peter Schröder, 2003, Discrete Shells, Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation

[3] Rasmus Tamstorf, 2013, Derivation of discrete bending forces and their gradients, received per e-mail on April 27, 2018 from Roi Poranne (ETH Zurich)

[4] Rasmus Tamstorf, Eitan Grinspun, 2013, Discrete bending forces and their Jacobians, *Graphical Models 75*, 362-370.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Interactive design of complex objects from thin shells

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
Wigger	Linus

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
Aarau, 15.7.2018	Linus Wigger

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.