Interactive differential growth simulation for design

Emilie Yu
s181757@student.dtu.dk
Danmarks Tekniske Universitet

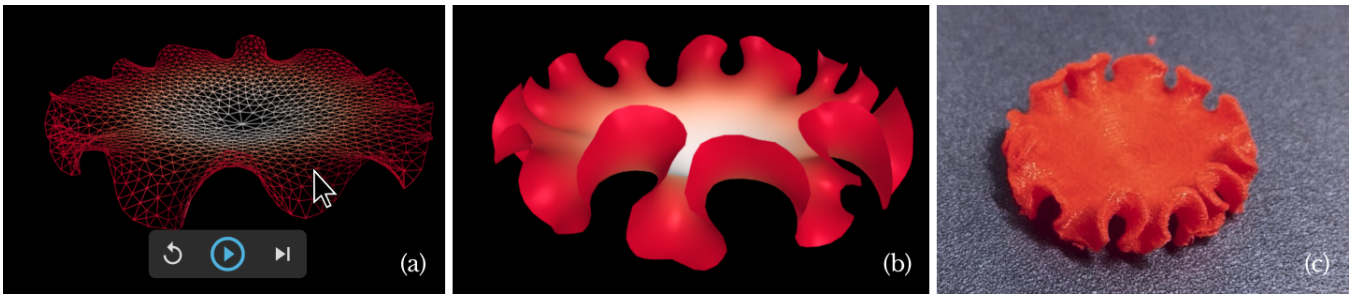

Figure 1: (a) Interface to run and control the simulation. (b) A shape generated by the simulation. (b) Fabricated model

ABSTRACT

Designing curved and wrinkled organic looking 3D shapes is hard with traditional modeling software. We propose a web-based application capable of generating such shapes based on differential growth simulation. We then create a user interface such that novice users can explore a large space of possible shapes. A user study confirms that all users can generate a diversity of shapes. However further work is needed to guide users while they learn how to use the application, and to bring more value to them through extended customisation and fabrication possibilities.

Figure 2: Ruffled kale leaves and flower petals. Photos by Erda Estremera on Unsplash.

1 INTRODUCTION

Designing 3D shapes can have many applications, from fabrication to video game asset creation. Many design softwares exist to cater to the needs of content creators, and with those, a skilled artist can succeed in creating any complex shape, given enough time. However, some shapes such as trees, plants, and objects with a repetitive pattern or structure such as buildings, still remain especially time-consuming to create manually, and procedural modeling techniques have been developed to generate them while still giving control to the artist [10, 11].

In this project, we create an online interactive tool to generate thin surfaces with curved and wrinkled edges, reminiscent of what can be observed in nature on flower petals, leaves, or some animals (Fig. 2).

We propose to generate these surfaces by simulating differential growth. Differential growth is observed in plant leaves as growth is faster near the edge. To respect bio-mechanical constraints and this spatially varying growth strain, they become curved and rippled on the edge as they grow [6]. Our growth simulation is based on a 3D triangular mesh on which we simulate cell creation, repulsion between cells and bending/stretching resistance.

With our solution, the user can control the generation of the shape through a number of exposed parameters, thus exploring a large diversity of results. A user study validates that our interface is accessible to anyone, even users with no prior modeling experience.

2 RELATED WORK

Here we present work related to the effect of differential growth on shape formation in plants. Then we present other simulation tools that generate shapes with a growth simulation and explain how our solution differentiates from these.

2.1 Differential growth and shape formation

It has been found that differential growth on leaves and petals drives the formation of a variety of complex 3D shapes.

The wavy shapes of the edge of a leaf was compared to that of a torn plastic sheet and it was explained by Marder et al. that this pattern was the result of elastic relaxation after a permanent deformation of the sheet leading to a different equilibrium distance between points [9].

Liang and Mahadevan [6] studied more specifically the effects of this phenomenon when taking into account the boundary conditions of a long leaf. They found through observation of live leaves and numerical simulation that differential growth could cause the leaves to become saddle-shaped and/or to present edge ripples, depending on the scale of the growth strain at the edge.

Lastly, Huang et al. [4] showed that this effect could cause leaves to adopt a variety of 3D shapes, such as being twisted, helix shaped or wavy, depending on growth strain scale and the steepness of growth strain fade.

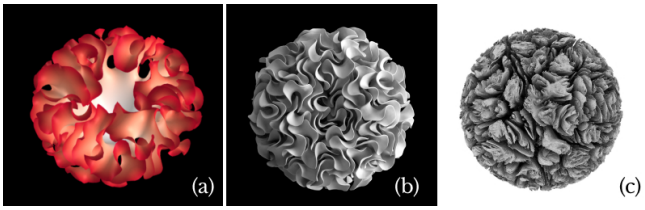

Figure 3: (a) Ours. (b) Edge based growth by Nervous System. (c) Cellular Form 17_0007_0025 by Andy Lomas.

This theoretical work on morphogenesis is at the foundation of our growth simulation. The effect of growth fade on the resulting shape is one aspect that we will explore to provide user control.

2.2 Growth simulations for design

Using simulation of growth to design shapes is not a novel idea. Here we present 2 interesting takes on this problem.

Nervous System explored quite extensively the idea of using differential growth to drive the creation of 3D surfaces reminiscent of flowers and other ruffled organisms [8]. They model the surface by a triangle mesh that resists bending and stretching with the discrete shells model by Grinspun et al. [3] and resolve by implicit integration. Our solution is directly inspired by this work. We use a triangular mesh to model the surface and use the same method as them to compute geodesic distance [2]. However, we propose a web-based interactive solution, which puts more emphasis on limiting computationally expensive methods. Therefore we have to do without the discrete shells model. Our results are less intricate than theirs (see Fig. 3), as we face slow convergence for very detailed models (see Appendix B for more details about performance).

Andy Lomas goes with a different approach in Cellular Forms [7]. In this project, growth is simulated on cells represented by a particle system. The simulation is implemented on the GPU and this enables simulating growth for over 50 million particles. Differential growth is also simulated, though it is not edge based but comes from reaction-diffusion over the surface from a source. Because of this, the results are very different from Nervous System's work and ours (see comparison Figure 3).

None of the previous work feature an interactive simulation available to the public.

3 METHOD

In this section, we present the algorithm devised to simulate growth on a surface embedded in a 3D space, and introduce the parameters that are exposed to users in the web application.

The simulation runs in small time steps, with the result of each time step being displayed in the application. Users can see immediately how the simulation is evolving and change parameters to fine tune their result.

Each step of the simulation is composed of 4 mechanisms:

- **Differential growth**: add vertices
- **Repulsion**: integrate repulsive influence of vertices on their neighbors
- **Environment influence**: integrate influence of gravity
- **Surface cohesion**: smooth the mesh

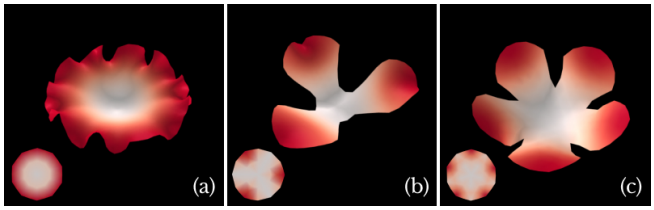

Figure 4: Results for different growth sources configurations (70 growth steps). Inset: initial state. (a) All edge. (b) 3 sources. (c) 5 sources.

3.1 Differential growth

To model differential growth, we first introduce the concept of growth source, and we use those to compute the growth factor on each vertex of the mesh. Growth is then driven by splitting edges where the local growth factor is above a threshold. We can metaphorically think of edge splitting as "adding cells" to the living organism.

Growth source. A growth source is a point on the surface where the growth factor will be at its maximum. It was observed that the wavy shapes of leaves is driven by differential growth with growth sources along the edge [4, 6].

To allow for a wider range of generated shapes and inspired by the work of Nervous System [8], we give users the option to choose between the whole edge being covered in growth sources, or a small number of growth sources equally distributed on the edge (see Figure 4).

Growth factor. To determine whether a new cell should be added in a given zone of the mesh, we need to compute the local growth factor. The growth factor at a point is a function of the geodesic distance from this point to the nearest growth source. We use the heat method by Crane et al. [2] to compute the geodesic distance between 2 vertices of the mesh.

The normalized growth factor ϕ_i at vertex v_i is computed from the geodesic distance to the closest source d_i and the maximum geodesic distance computed on the mesh $D = \max_k d_k$:

$$\phi_i = \text{sigmoid}\left(\frac{D - d_i}{D}, 1 - growthZone\right) \tag{1}$$

The sigmoid function [1] is applied to give users control on the scale of the growth zone. They can control this through the exposed parameter *growthZone* (see Figure 5).

Growth by edge splitting. Once the per-vertex growth factors have been computed, we look at each edge of the mesh and decide whether it should be split. When an edge is split, a new vertex is added at its center and the neighboring faces are split in 2.

An edge of length l_e should be split if it were to become longer than the rest length l_0 (mean initial length of edges) after being virtually scaled up by the growth factor ϕ_e.

$$l_e * (1 + \phi_e) \geq l_0 \quad \Rightarrow \quad \text{split } e$$

[1] See Appendix A for more details

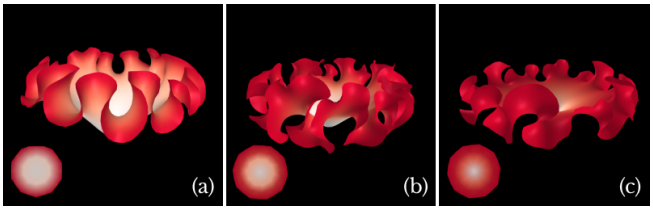

Figure 5: Results for different growth zone values (155 growth steps). Inset: initial state. (a) *growthZone* = 0.1 (b) *growthZone* = 0.5 (c) *growthZone* = 0.8

Mesh quality. After all necessary edge splits are performed, mesh quality is improved by flipping edges if it makes the minimum angle of the adjacent triangles bigger [1]. Edges with a dihedral angle $\geq$ 0.3rad are prevented from flipping to preserve geometry.

3.2 Repulsion

The second mechanism driving the formation of the 3D shape is repulsion between vertices of the mesh. If we think again of the vertices as cells of an organism, each cell has a volume around it which cannot be interpenetrated by other cells.

At each time step, we compute the elastic force $\boldsymbol{f_e}$ applied on a vertex v_i by other vertices in its neighborhood $\mathcal{N}_i$.

$$\boldsymbol{f_e} = k_e \sum_{j, v_j \in \mathcal{N}_i} (\|\boldsymbol{x_j} - \boldsymbol{x_i}\| - l_e) * \frac{\boldsymbol{x_j} - \boldsymbol{x_i}}{\|\boldsymbol{x_j} - \boldsymbol{x_i}\|} \tag{2}$$

This formulation is Hooke's law, where we model a spring of stiffness k_e and of equilibrium length l_e between the vertex v_i and each of its neighbors v_j. The position of a vertex v_i is noted $\boldsymbol{x_i}$.

To make it computationally viable to compute this force, we use a 3D grid structure of resolution l_e implemented as a hash map.

3.3 Environment influence: gravity

To add user control to the simulation, a directional force is added to represent gravity. The users can control direction and norm of this vector through the interface.

The force corresponding to gravity $\boldsymbol{g}$ on vertex v_i is:

$$\boldsymbol{f_g} = \Phi_i \, m \, \boldsymbol{g} \tag{3}$$

All vertices are considered to be of unit mass for simplicity. Weighing $\boldsymbol{f_g}$ by the growth factor Φ_i is a way to constrain vertices that are not in a growth zone to be fixed, which prevents the whole model from falling.

Once $\boldsymbol{f_e}$ and $\boldsymbol{f_g}$ are computed for each vertex using (2) and (3), explicit Euler integration is used to update the positions.

3.4 Surface cohesion

With only the 2 previous rules applied to the surface each time step, the result will become very wrinkled and jagged. This is because some properties from the surface are not taken into account: it should resist bending and stretching.

As the surface is a triangle mesh, the problem can be approached as a discrete smoothing problem. We want to minimize the membrane energy and the thin plate energy. We do this by iteratively updating vertex positions using a discrete approximation of the Laplacian (as proposed by Kobbelt et al. [5]).

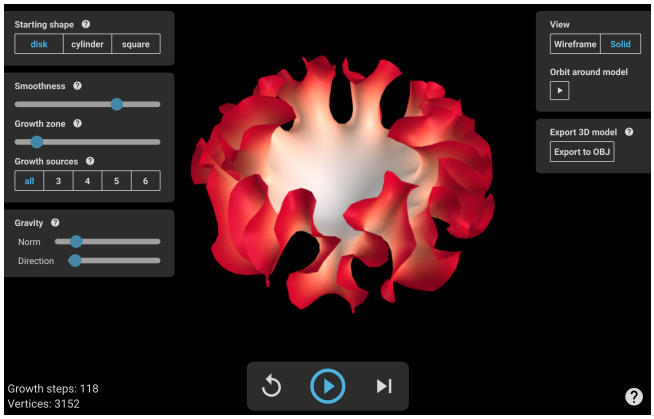

Figure 6: Screenshot of the user interface

After implementing minimization of both membrane and thin plate energies, we observed that we could get a satisfying result using only the less expensive membrane energy optimization update. Therefore we use only this update rule for vertex v_i of position $\boldsymbol{x_i}$ and with N_i neigbhors in $\mathcal{N}_i$:

$$\boldsymbol{x_i} \leftarrow \boldsymbol{x_i} + smoothness * \left(\frac{1}{N_i} \sum_{v_j \in \mathcal{N}_i} \boldsymbol{x_j} \quad - \quad \boldsymbol{x_i} \right) \tag{4}$$

We expose parameter *smoothness* in the user interface.

The resulting update is simply to move each vertex towards the barycenter of its neighbors. We can think of this update as cells keeping cohesion between each other.

4 RESULTS AND DISCUSSION

The tangible output of this project is a web application [2] consisting of a user interface to run the simulation and control its parameters (Fig. 6). The application runs at interactive speed for a low number of vertices but becomes slow for a big mesh (see Appendix B). In this section we will discuss how well the initial goals are fulfilled by analysing results from a user study and present potential applications.

4.1 User study

Through a qualitative user study, we validate that the web application satisfies the criteria of being:

(1) Accessible to anyone, without requiring previous modeling experience.
(2) Controllable, such that the user can explore the space of possible results and create shapes from parameters in a predictable way.

The user study was conducted with 7 participants. 3 of the participants were supervised and were tasked with reproducing 4 target shapes under a limited time while thinking aloud. The other participants were asked to create a shape without any constraints. All

[2] The application is accessible online at: differential-growth.surge.sh

Figure 7: 3D models generated by users after a short exploration of the web application

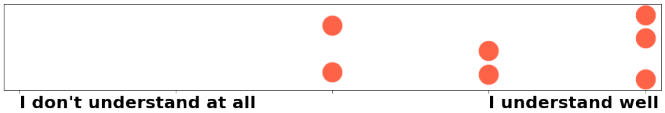

Figure 8: How well do you understand the cause/effect relationship between changing these parameters and the generated shape? (7 participants)

participants answered a survey and uploaded a shape that they generated (Fig. 7).

Accessible interface. 6 of the 7 participants are not familiar with any modeling software. However, all participants succeeded in creating a 3D model with our application, with no or little outside help other than the embedded tutorial in the application (see Fig. 7).

Controllable interface. All participants understand the cause/effect relationship between changing the parameters in the interface and the generated shape relatively well (see Fig. 8). During the supervised thinking aloud tests, all 3 participants succeeded in reproducing the target shapes under 5 minutes.

However, this success is open to discussion. First, some parameters and their effect are confusing to users. Half of the participants stated that the *smoothness* parameter was unclear to them.

Secondly, thinking aloud tests showed that participants discover the effect of parameters mostly through trial and error. There is no apriori intuition about the parameters and their effect. This learning phase can feel overwhelming to participants, as they try to tweak each parameter and discover its effect.

Lastly, all 3 supervised participants failed to discover that the number of *growth sources* could be changed during the simulation to create more varied results. This is interesting, as it shows that more guidance is required from the interface to encourage users to try interactions that they would think impossible.

A quick fix to these limitations would be to expose less parameters, we think of hiding *smoothness* which was more confusing than useful. Another feature that would help the learning process is an interactive tutorial, revealing parameters one at a time on the interface, although further work is needed to create it.

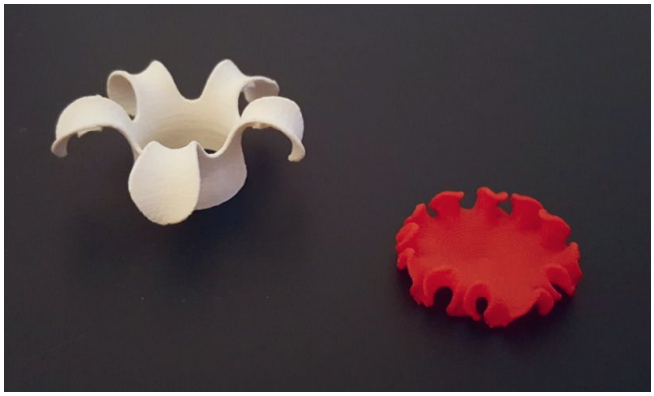

Figure 9: Fabricated models

4.2 Potential applications

All participants found their exploration of the simulation very enjoyable. However, without a particular intent in mind, they didn't feel compelled to continue using it. Therefore it seems like the next step for this project would be to look at potential applications for the generated shapes, and think of how the simulation should be extended to permit those.

Fabrication. As the output from the simulation is a 3D mesh, it is possible to 3D print it to turn it into a tangible object. We experimented with this technology and printed 2 simple shapes (see Fig. 9). The result is satisfying but 3D printing more complex shapes such as those on Figure 7 is close to impossible with filament-based printers because they must generate support structures for overhanging parts.

To make 3D printing fabrication easier, some constraints could be added to the simulation. We prototyped adding repulsive surfaces such that the generated shape has one flat side that can become the first layer of the 3D print. More constraints could be added, such as penalizing formation of overhangs to prevent issues with support.

Custom jewellery. A potential application (see [8]) is to create custom jewellery from the generated shapes. To create viable jewellery pieces, the simulation needs to support imported initial meshes, and it would need to be more controllable. For example it could allow users to choose growth sources precisely by clicking on a zone of the mesh.

5 CONCLUSION

We present a differential growth simulation and an associated user interface, that enable novice users to create custom, varied shapes that would be very hard to create with a classic CAD or modeling software.

We validate our solution through a user study and find out that improvements need to be done regarding guiding users to explore the complete space of possible shapes.

Finally, further work on making the generated shapes more customizable and suitable for fabrication is the next step to take to bring more value to users.

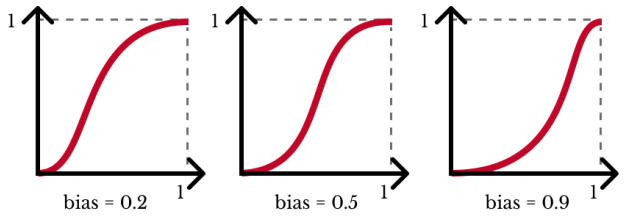

Figure 10: Sigmoid function for different bias values

ACKNOWLEDGMENTS

Thanks to Rohan Sawhney and Mark Gillespie of the **Geometry Collective** (Carnegie Mellon University) for building **geometry-processing-js** [3], the geometry and linear algebra library on top of which this work is built.

REFERENCES

[1] J Andreas Bærentzen, Jens Gravesen, François Anton, and Henrik Aanæs. 2012. *Guide to computational geometry processing: foundations, algorithms, and methods.* Springer Science & Business Media.

[2] Keenan Crane, Clarisse Weischedel, and Max Wardetzky. 2013. Geodesics in Heat: A New Approach to Computing Distance Based on Heat Flow. *ACM Trans. Graph.* 32, 5, Article 152 (Oct. 2013), 11 pages. https://doi.org/10.1145/2516971.2516977

[3] Eitan Grinspun, Anil N. Hirani, Mathieu Desbrun, and Peter Schröder. 2003. Discrete Shells. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '03)*. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 62–67. http://dl.acm.org/citation.cfm?id=846276.846284

[4] Changjin Huang, Zilu Wang, David Quinn, Subra Suresh, and K. Jimmy Hsia. 2018. Differential growth and shape formation in plant organs. *Proceedings of the National Academy of Sciences* 115, 49 (2018), 12359–12364. https://doi.org/10.1073/pnas.1811296115 arXiv:https://www.pnas.org/content/115/49/12359.full.pdf

[5] Leif Kobbelt, Swen Campagna, Jens Vorsatz, and Hans-Peter Seidel. 1998. Interactive multi-resolution modeling on arbitrary meshes. In *Siggraph*, Vol. 98. 105–114.

[6] Haiyi Liang and L. Mahadevan. 2009. The shape of a long leaf. *Proceedings of the National Academy of Sciences* 106, 52 (2009), 22049–22054. https://doi.org/10.1073/pnas.0911954106 arXiv:https://www.pnas.org/content/106/52/22049.full.pdf

[7] Andy Lomas. 2014. Cellular Forms: an Artistic Exploration of Morphogenesis. *AISB* (August 2014). http://research.gold.ac.uk/24732/

[8] Jesse Louis-Rosenberg. 2015. *FLORAFORM – AN EXPLORATION OF DIFFERENTIAL GROWTH.* https://n-e-r-v-o-u-s.com/blog/?p=6721

[9] M Marder, E Sharon, S Smith, and B Roman. 2003. Theory of edges of leaves. *Europhysics Letters (EPL)* 62, 4 (may 2003), 498–504. https://doi.org/10.1209/epl/i2003-00334-5

[10] Pascal Müller, Peter Wonka, Simon Haegler, Andreas Ulmer, and Luc Van Gool. 2006. Procedural modeling of buildings. In *Acm Transactions On Graphics (Tog)*, Vol. 25. ACM, 614–623.

[11] Przemyslaw Prusinkiewicz, Mark Hammel, Jim Hanan, and Radomir Mech. 1996. L-systems: from the theory to visual models of plants. In *Proceedings of the 2nd CSIRO Symposium on Computational Challenges in Life Sciences*, Vol. 3. Citeseer, 1–32.

A SIGMOID FUNCTION WITH BIAS

The sigmoid function used to compute the growth factor in (1) is a polynomial function similar to a smoothstep between 0 and 1, but such that it is possible to bias it towards 0 or 1. See Figure 10.

$$\text{sigmoid}(x, \text{bias}) = \begin{cases} \frac{x^3}{\text{bias}^2} & 0 \le x \le \text{bias} \\ 1 - \frac{(1-x)^3}{(1-\text{bias})^2} & \text{bias} \le x \le 1 \end{cases}$$

[3] https://github.com/GeometryCollective/geometry-processing-js

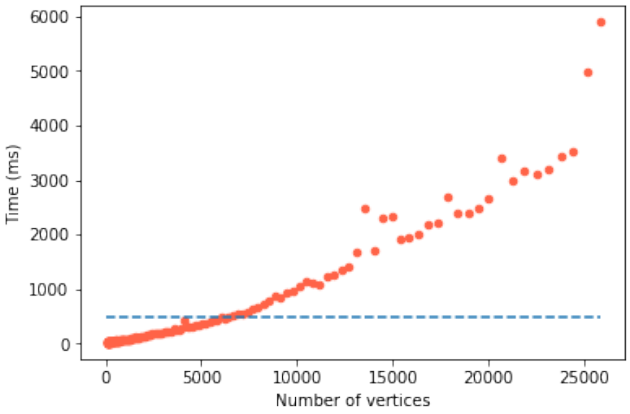

Figure 11: Time to perform one step of the simulation and render the frame as a function of the number of vertices in the mesh. Blue line corresponds to 0.5s (Measured on laptop with Intel© Core™ i7-6600U CPU @ 2.60GHz × 2)

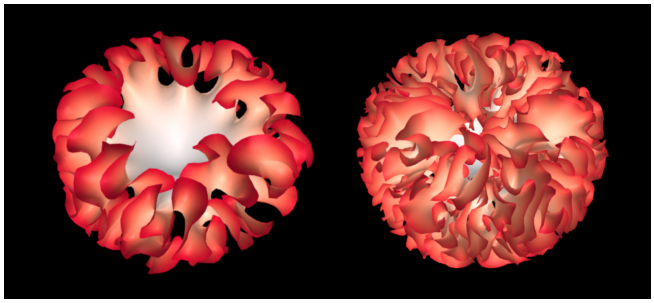

Figure 12: Shapes generated. Left: with a frame time below 0.5s (t = 14s). Right: at the end of the performance test (t = 100s)

B PERFORMANCE

Our solution maintains an acceptable framerate only for a rather small number of vertices in the mesh.

We evaluated performance by measuring the time necessary to perform one step of the simulation (and render the frame). On Figure 11 we see that the performance quickly deteriorates for meshes bigger than 10 000 vertices.

The result obtained with an acceptable framerate (time < 0.5s) seen on Figure 12 left is still quite interesting but it is disappointing that more intricate results such as the model generated at the end of the performance test (Figure 12 right). This result took 100s in total to obtain, which would be an acceptable time for an offline simulation, but is too slow for our purpose of an interactive simulation.

However, performance is something that we have not sought to optimize in our current implementation. Further work on this aspect could improve the current results. For example we could take advantage of web workers which are a browser functionality to run tasks in background threads. We could use different threads for the simulation and frame rendering.