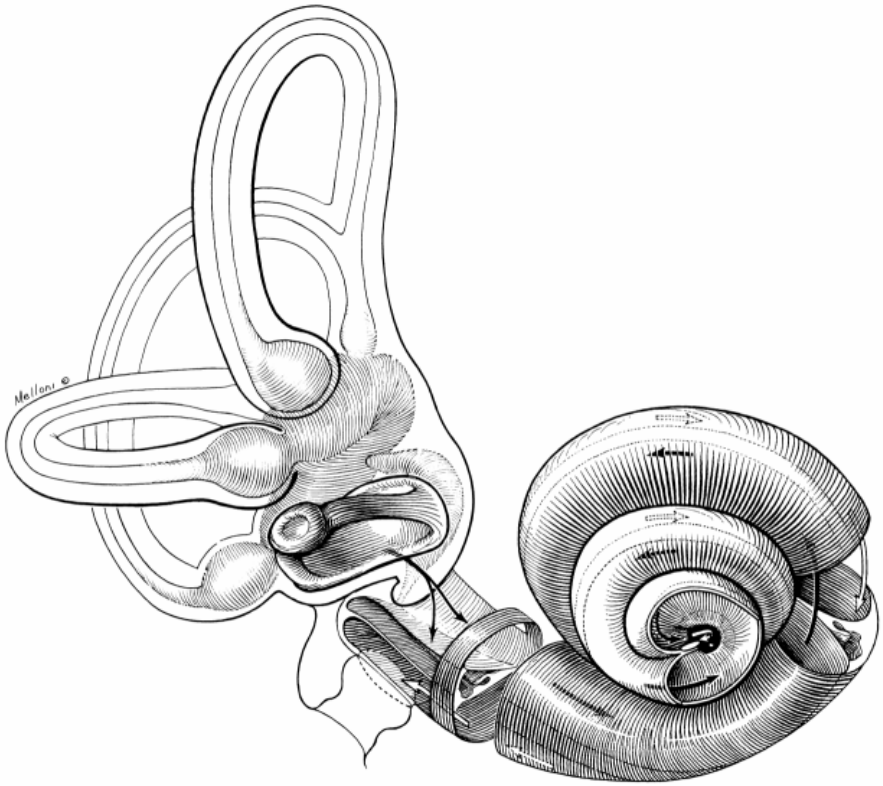
High Quality Hatching

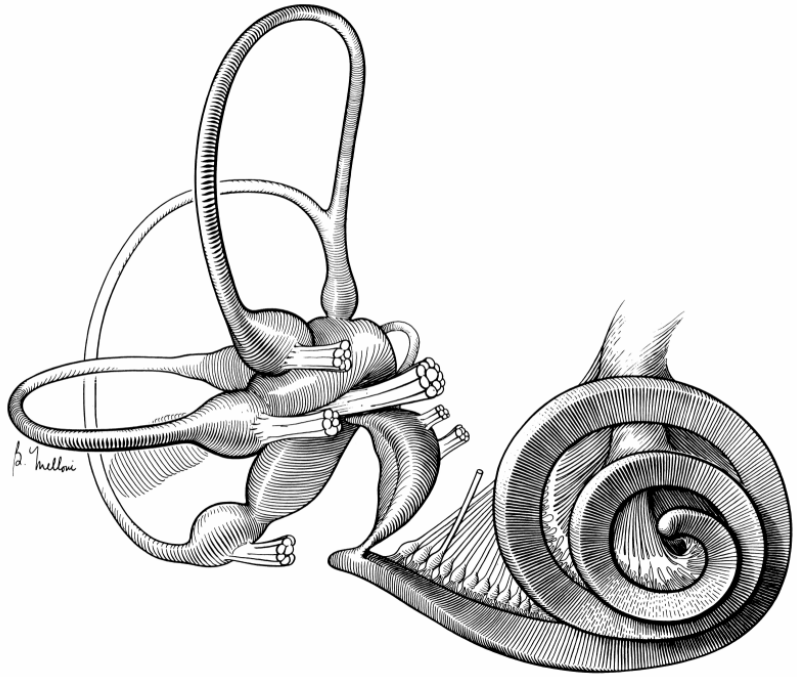
Johannes Zander, Tobias Isenberg,
Stefan Schlechtweg, Thomas Strothotte

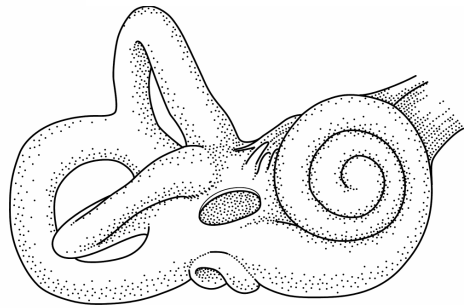
University of Magdeburg
Department of Simulation and Graphics
Magdeburg, Germany

Motivation

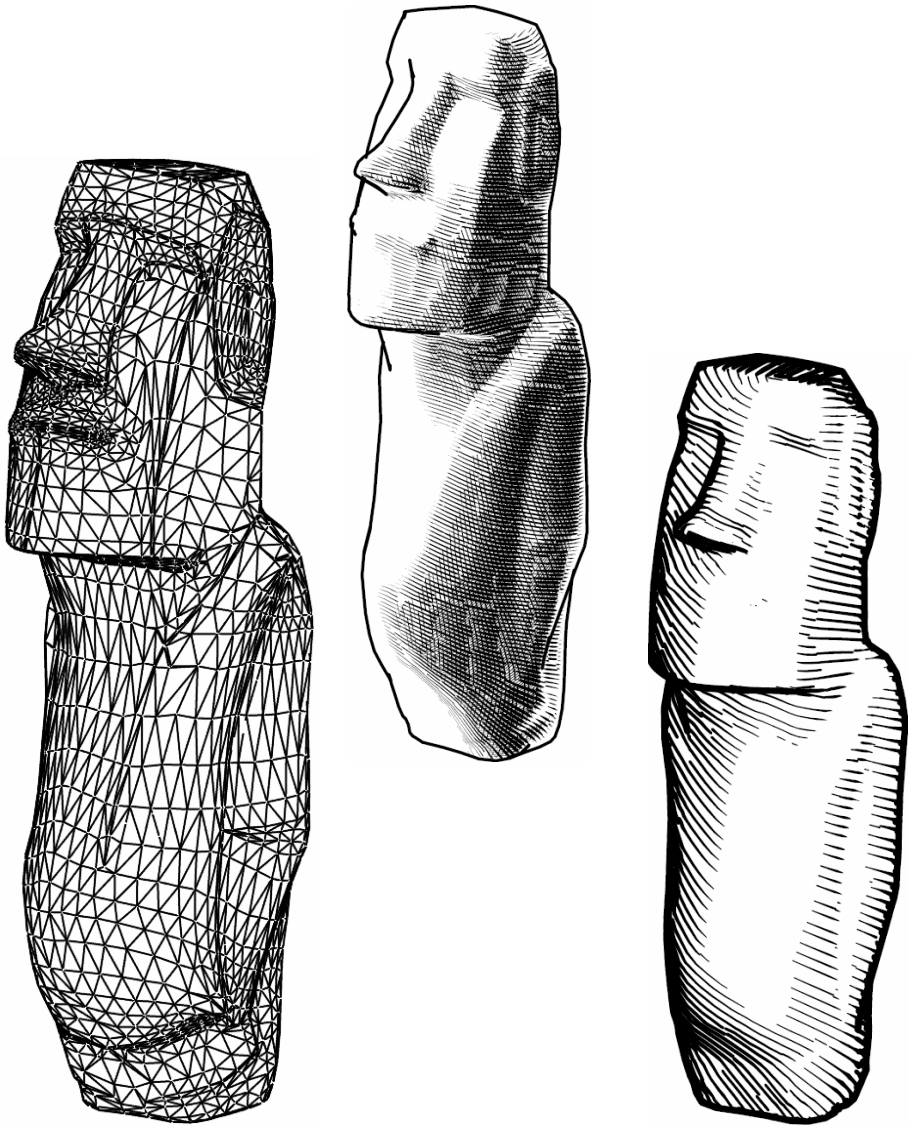
Hatching as a key technique for scientific illustration







Goals

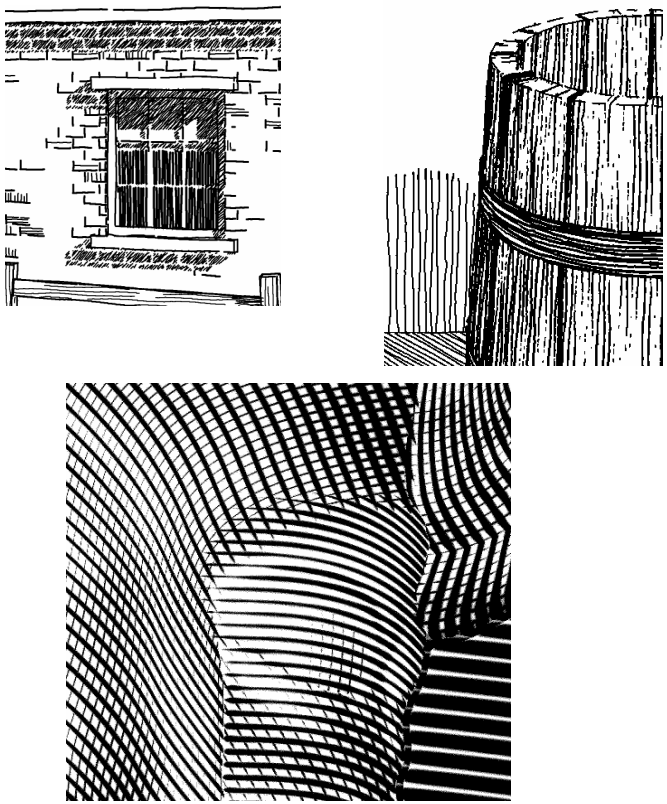


- hatching of polygonal meshes to represent surface features
- adapt placement and style of lines to visibility and lighting
- high-quality vector based output
- interactive output for design and tuning

Related Work

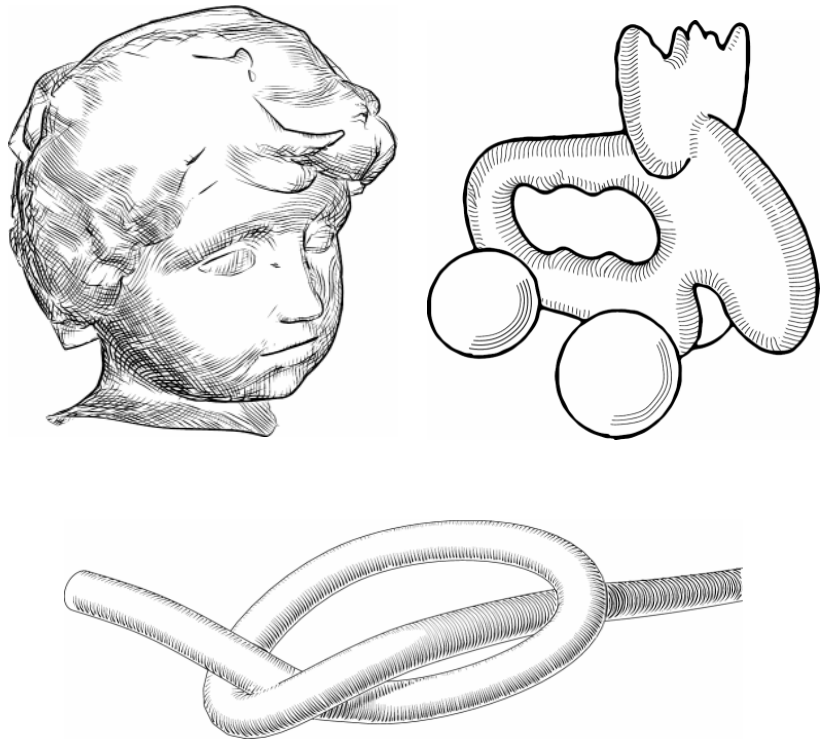
hatching

(Salisbury et al., 1994) (Leister, 1994)
(Winkenbach & Salesin, 1994)
(Ostromoukhov, 1999)



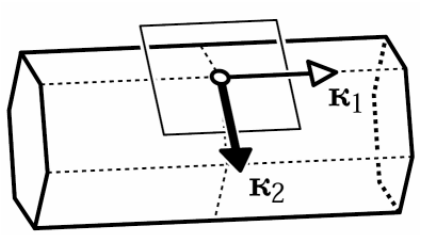
model-based hatching

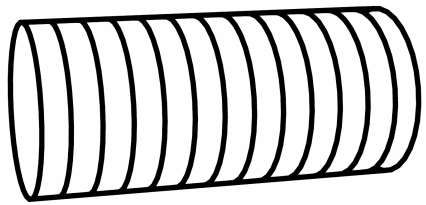
(Deussen et al., 1999)
(Rössl & Kobbelt, 2001)
(Hertzmann & Zorin, 2000)



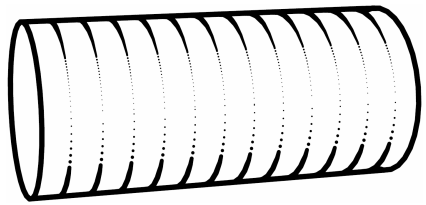
General Procedure

- preprocessing: generation of candidate lines
  - compute curvature values and directions at mesh vertices and construct a vector field
  - ***vector field optimization***
  - ***integration of streamlines***



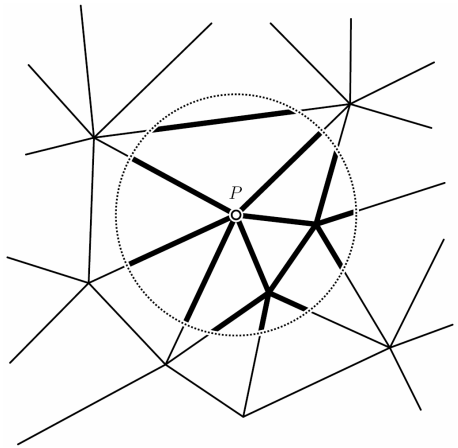


- rendering: adaptive line visualization
  - ***line shading***
  - ***media-specific output***



Vector Field Optimization

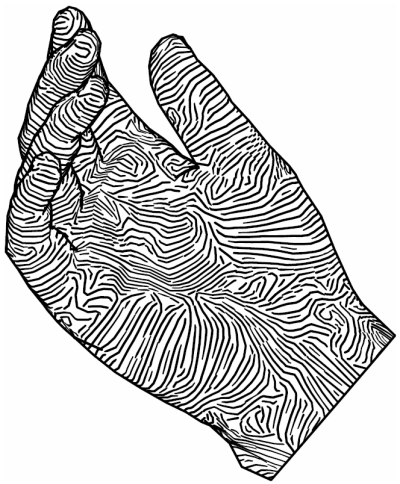
- initial vector field:
  - curvature not well defined everywhere
  - high frequency noise
  - mesh smoothing does not solve the problem
- two homogeneity criteria
  - tension between neighbors
  - tension within local neighborhoods

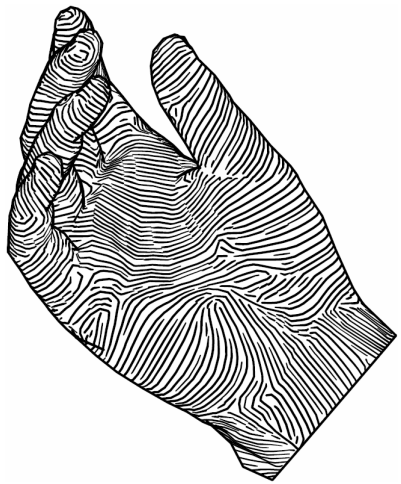


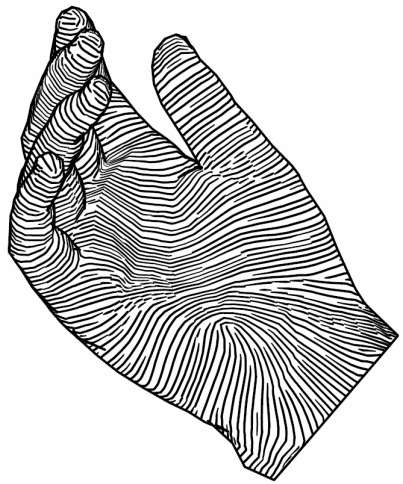
- optimization: minimization of this energy term

Vector Field Optimization

- minimization of energy term
  - reduction of cases with non-uniform directions
  - local relaxation of direction field
  - global optimization using L-BFGS-B technique (Zhu and Byrd, 1997)

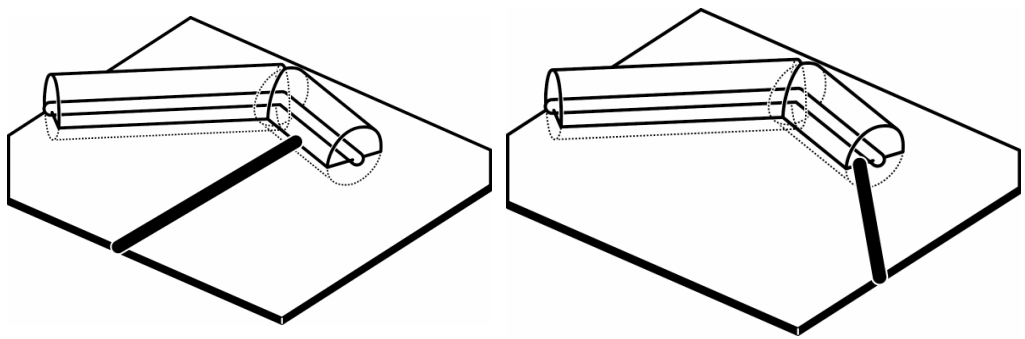






3D Streamline Generation

- integration of the vector field on the surface of the model adapted from (Jobard & Lefer, 1997)
- streamlines must not intersect the surface but always lie on the surface
- several streamline termination criteria
  - change in surface orientation
  - change in direction
  - streamline length
  - line crossings and proximity

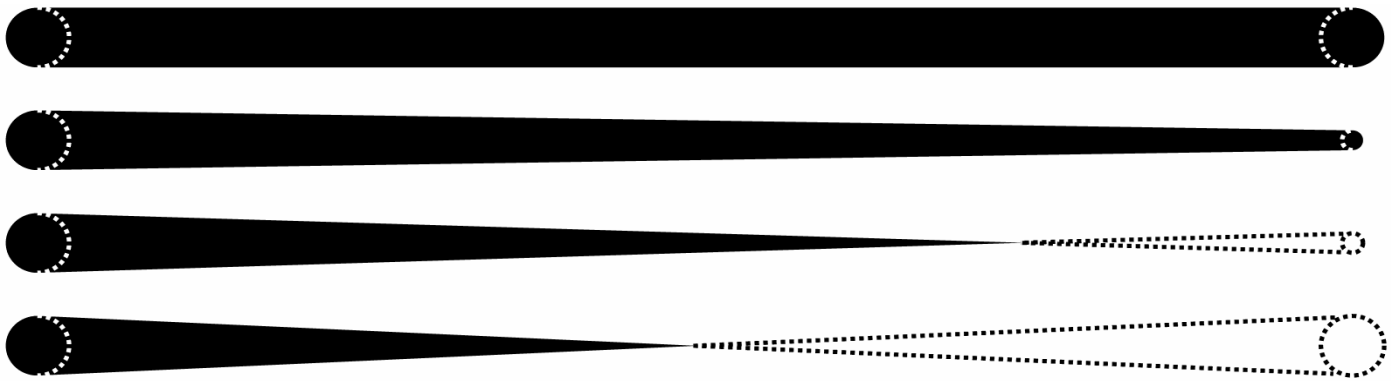


General Procedure

- preprocessing: generation of candidate lines
  - compute curvature values and directions at mesh vertices and construct a vector field
  - ***vector field optimization***
  - ***integration of streamlines***
- rendering: adaptive line visualization
  - ***line shading***
  - ***media-specific output***

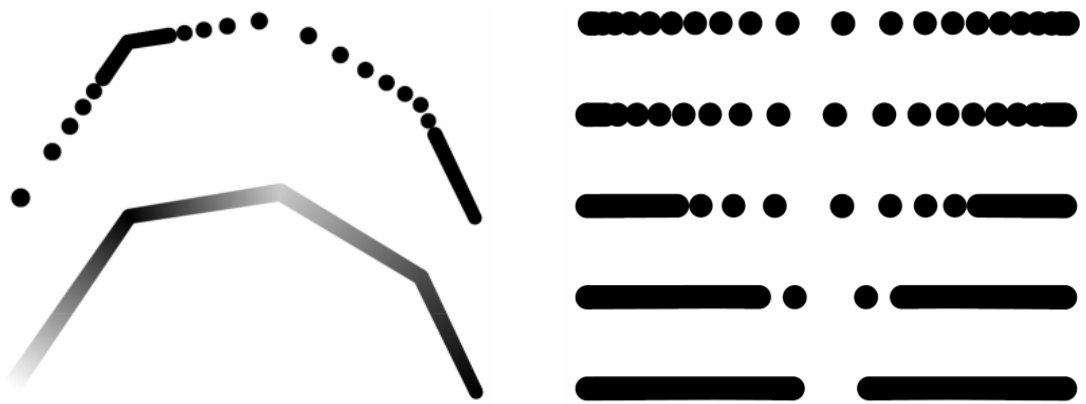
Line Shading

- adapt line width to the lighting situation
  - possibility to end lines at arbitrary positions
  - use of negative line widths
    - lines ends now at zero-crossings of line width
    - line ends not restricted to stroke vertices anymore
    - easy parameterization with only few vertices



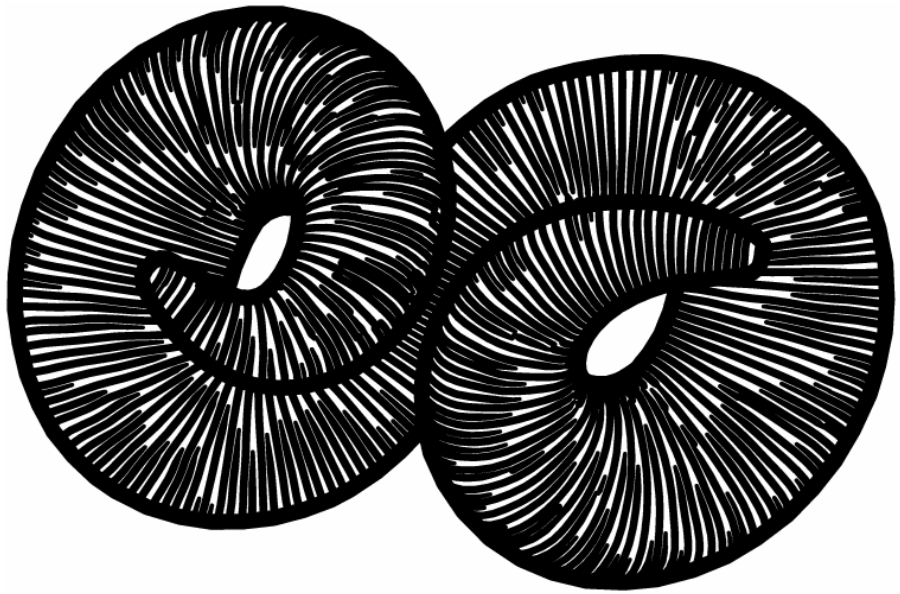
Line Shading

- line intensity halftoning through line stippling
- for print media no “real” gray values
- stipples placed on line path
  - stipple distance simulates local gray value
  - stipples must not overlap
  - dissolve lines into dots with distance dependent on brightness



Line Shading

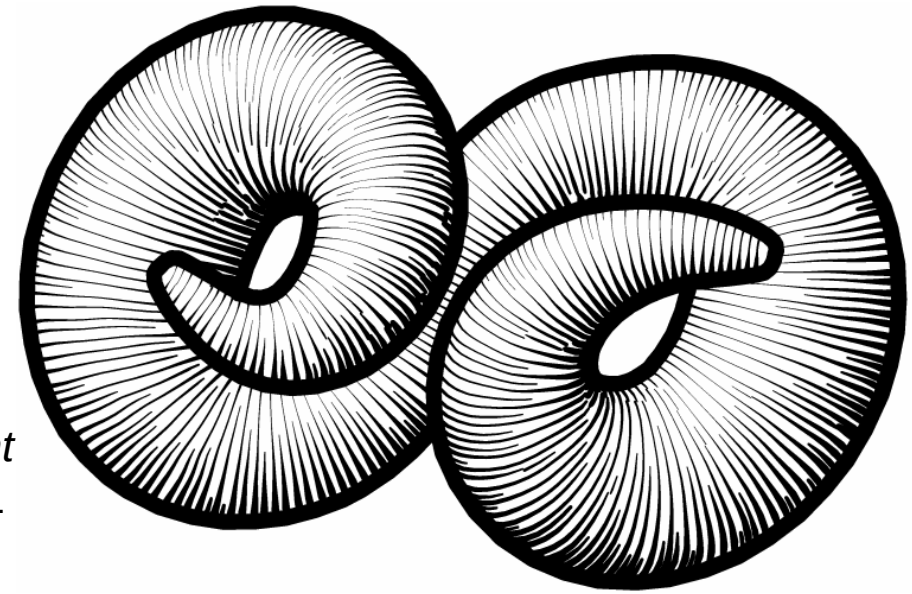
- implemented using a virtual machine
- shading equation can be changed at runtime for experiments
- line width and intensity (stipples) depend on illumination variables



line width = 1
line density = 1

Line Shading

- implemented using a virtual machine
- shading equation can be changed at runtime for experiments
- line width and intensity (stipples) depend on illumination variables

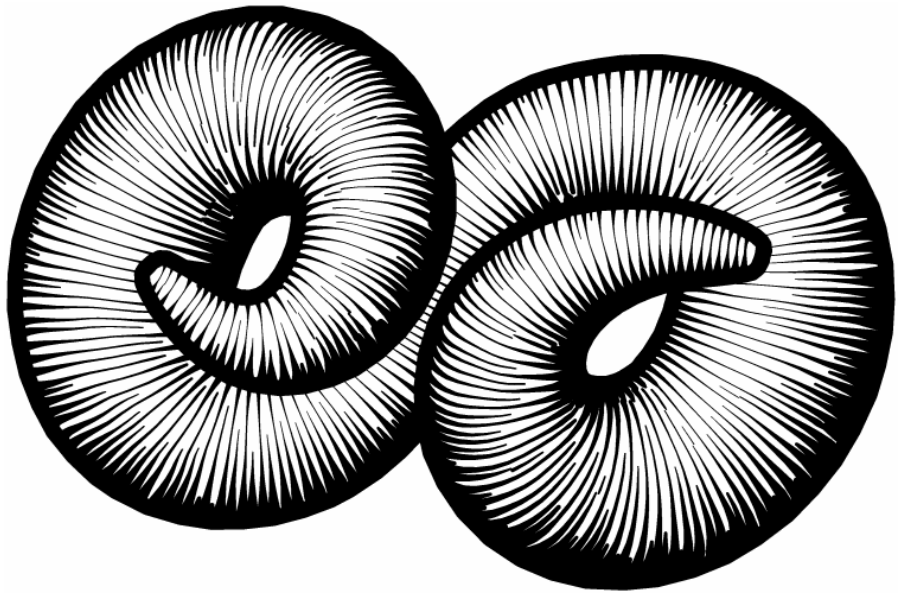


line width = *light*
line density = 1

Line Shading

- implemented using a virtual machine
- shading equation can be changed at runtime for experiments
- line width and intensity (stipples) depend on illumination variables

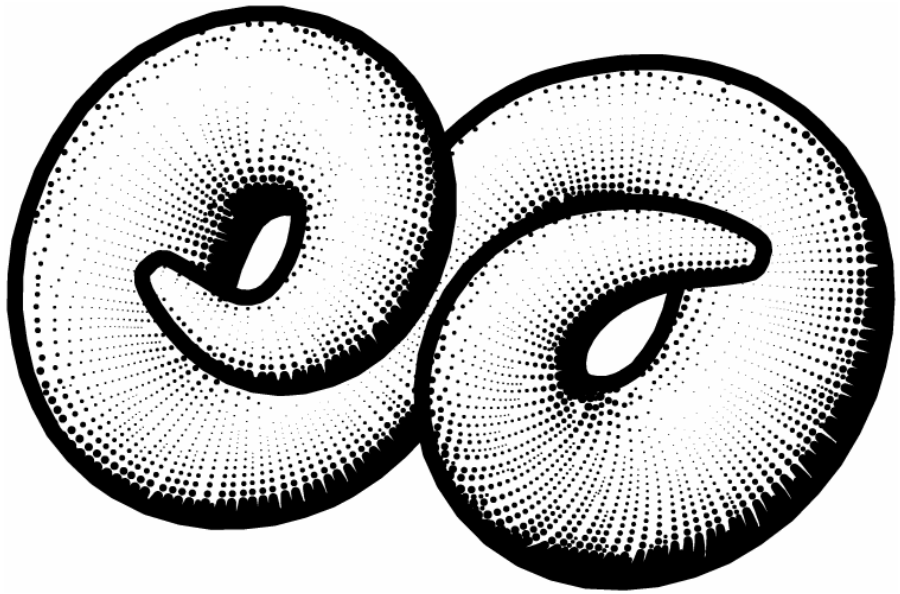
line width = *light* + *rim*
line density = 1



Line Shading

- implemented using a virtual machine
- shading equation can be changed at runtime for experiments
- line width and intensity (stipples) depend on illumination variables

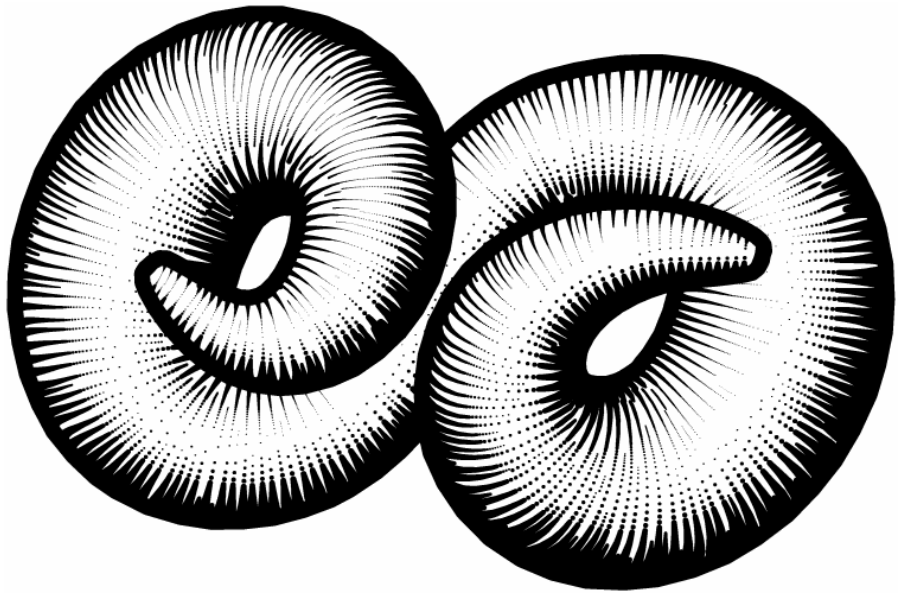
line width = *light* + *rim*
line density = *light*



Line Shading

- implemented using a virtual machine
- shading equation can be changed at runtime for experiments
- line width and intensity (stipples) depend on illumination variables

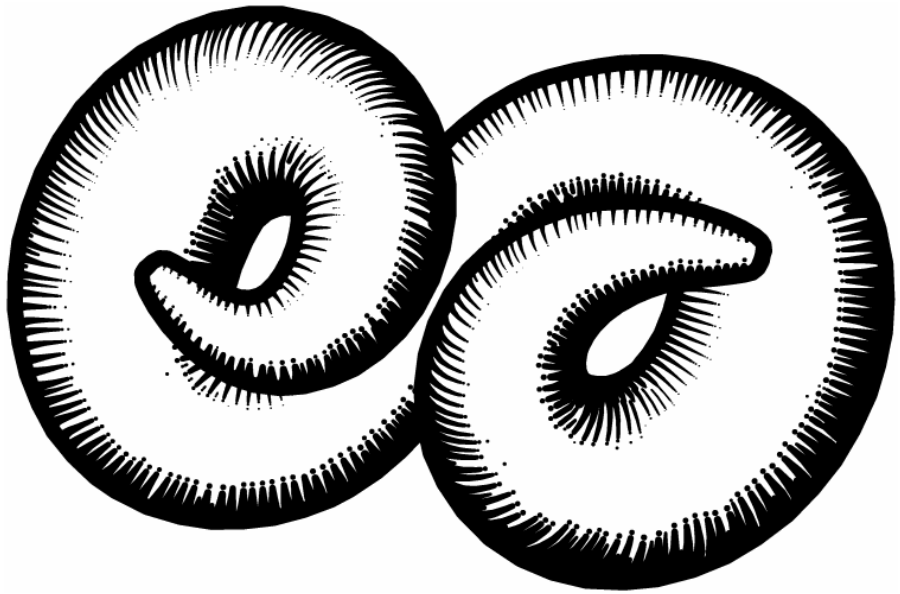
line width = *light* + *rim*
line density = 5 *light*



Line Shading

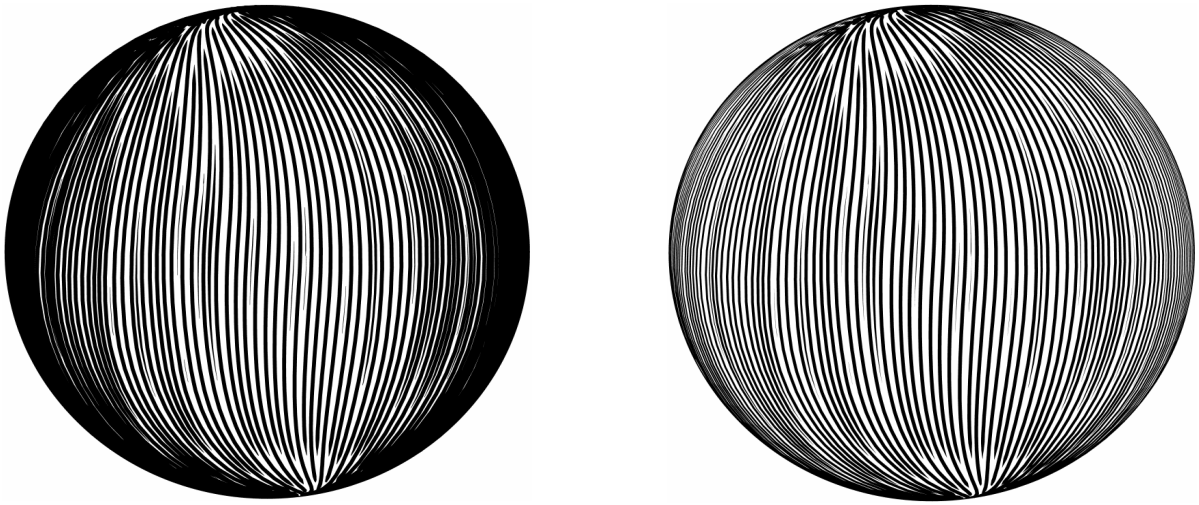
- implemented using a virtual machine
- shading equation can be changed at runtime for experiments
- line width and intensity (stipples) depend on illumination variables

line width = *light* + *rim*
line density = (5 *rim*)^{50}



Line Shading

- perceived brightness of 3D lines
  - brighter in approximately front-facing regions
  - darker in approximately perpendicular regions
- disturbing in 3D hatching applications
- scale line width accordingly using a correction factor

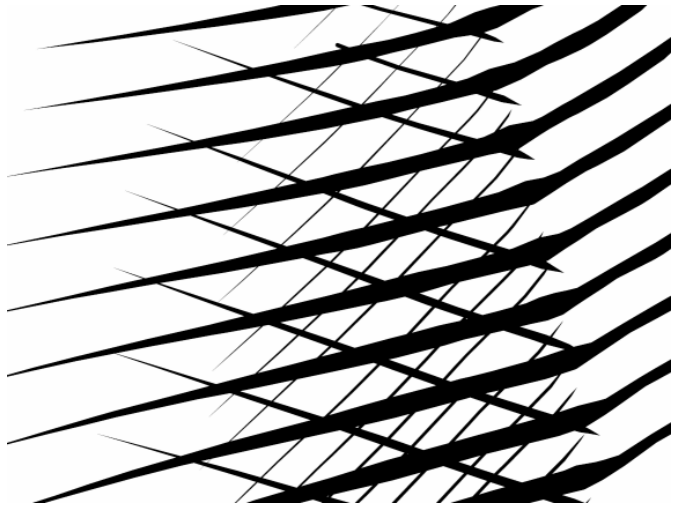


Line Output

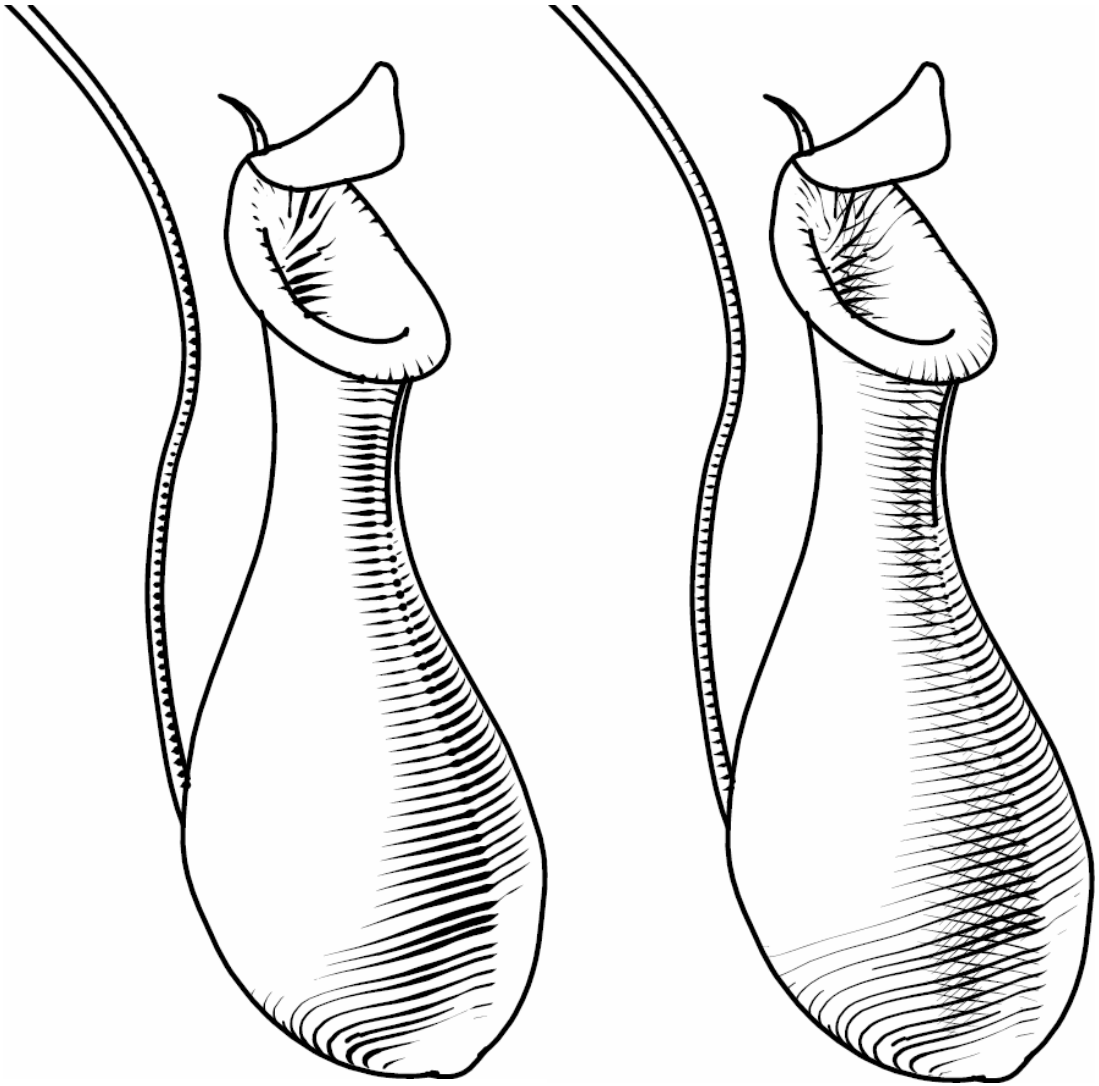
- media specific rendering of lines
- high resolution, high quality, vector-oriented for print media (using PDF)
  - direct native coding of vector data in PDF primitives
  - no aliasing problems
  - further processing easily possible
- interactive, OpenGL-based for WYSIWYG design changes and exploration
  - representing lines as triangle strips, arcs as triangle fans
  - user defined resolution
  - antialiasing

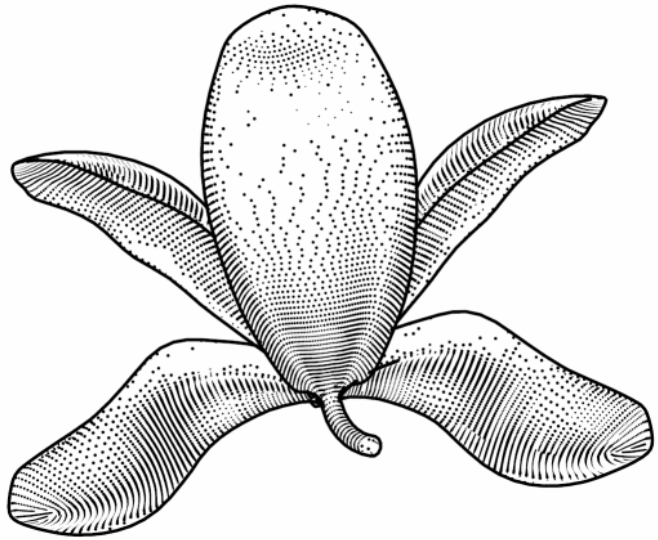
Line Output

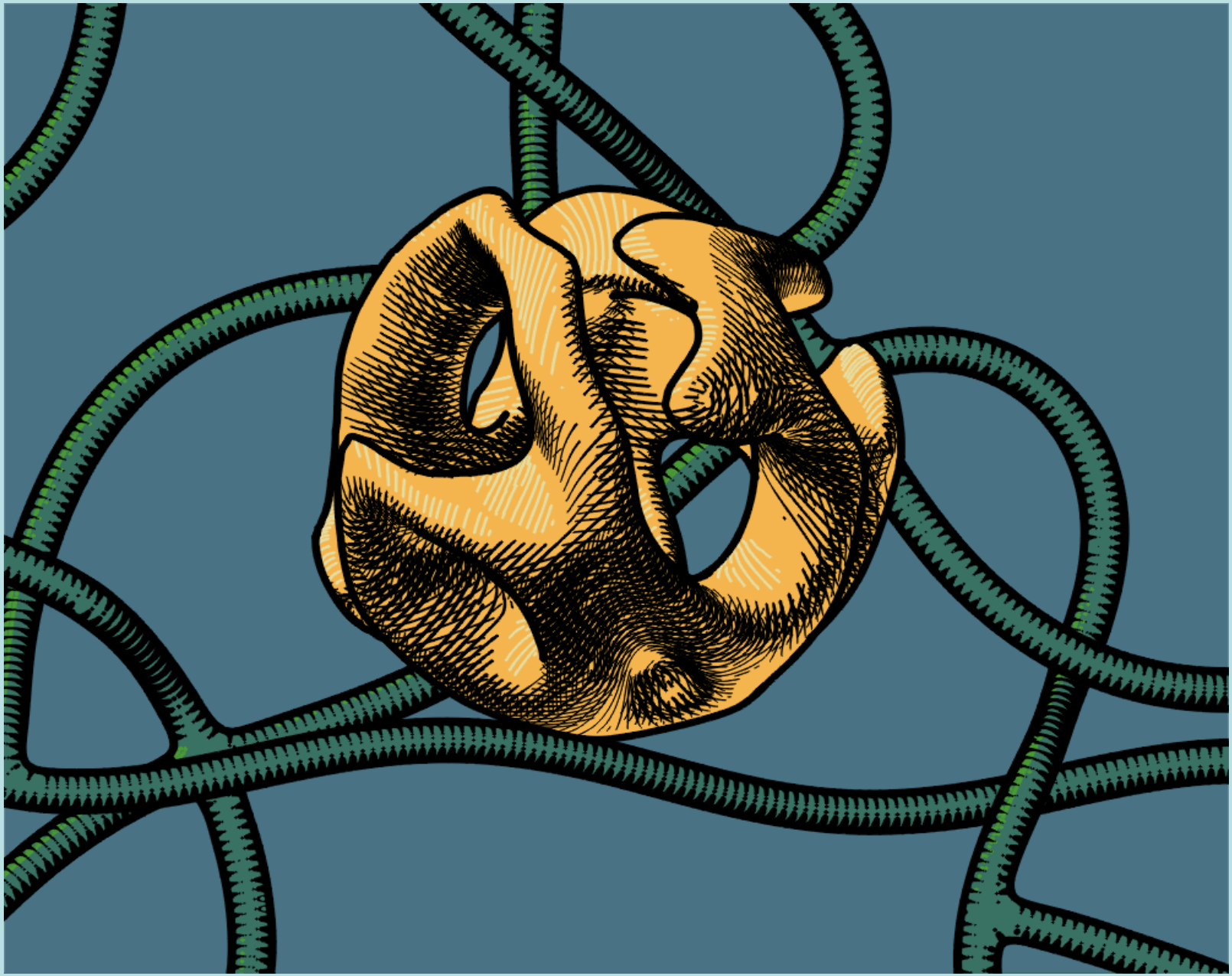
- cross-hatching achieved by combining multiple rendering passes and overlaying the results
- for each pass: rotating the direction field around arbitrary angles
- compose several layers of differently shaded strokes
- use spot colors for color prints
- combine with silhouettes and colored areas



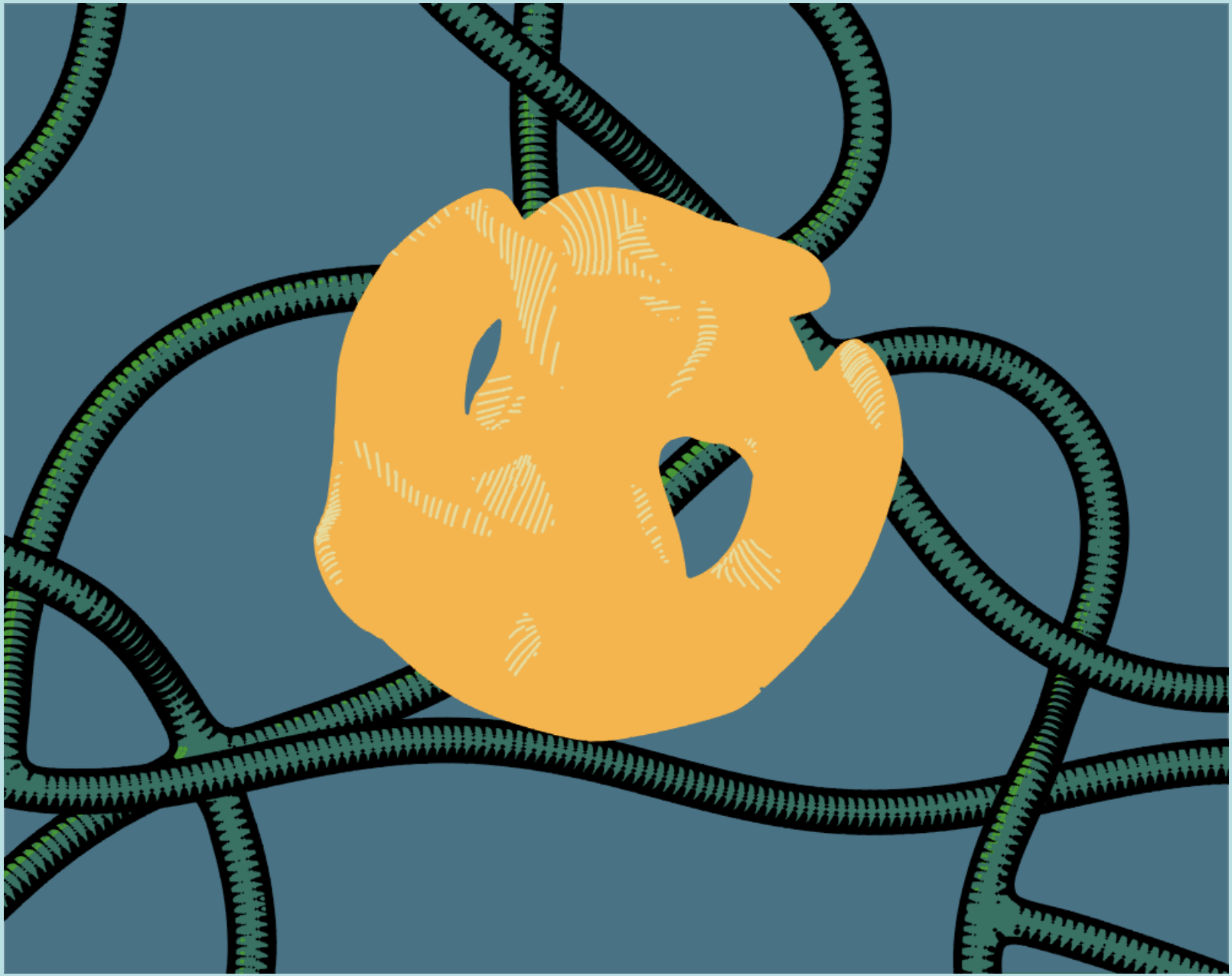
Examples

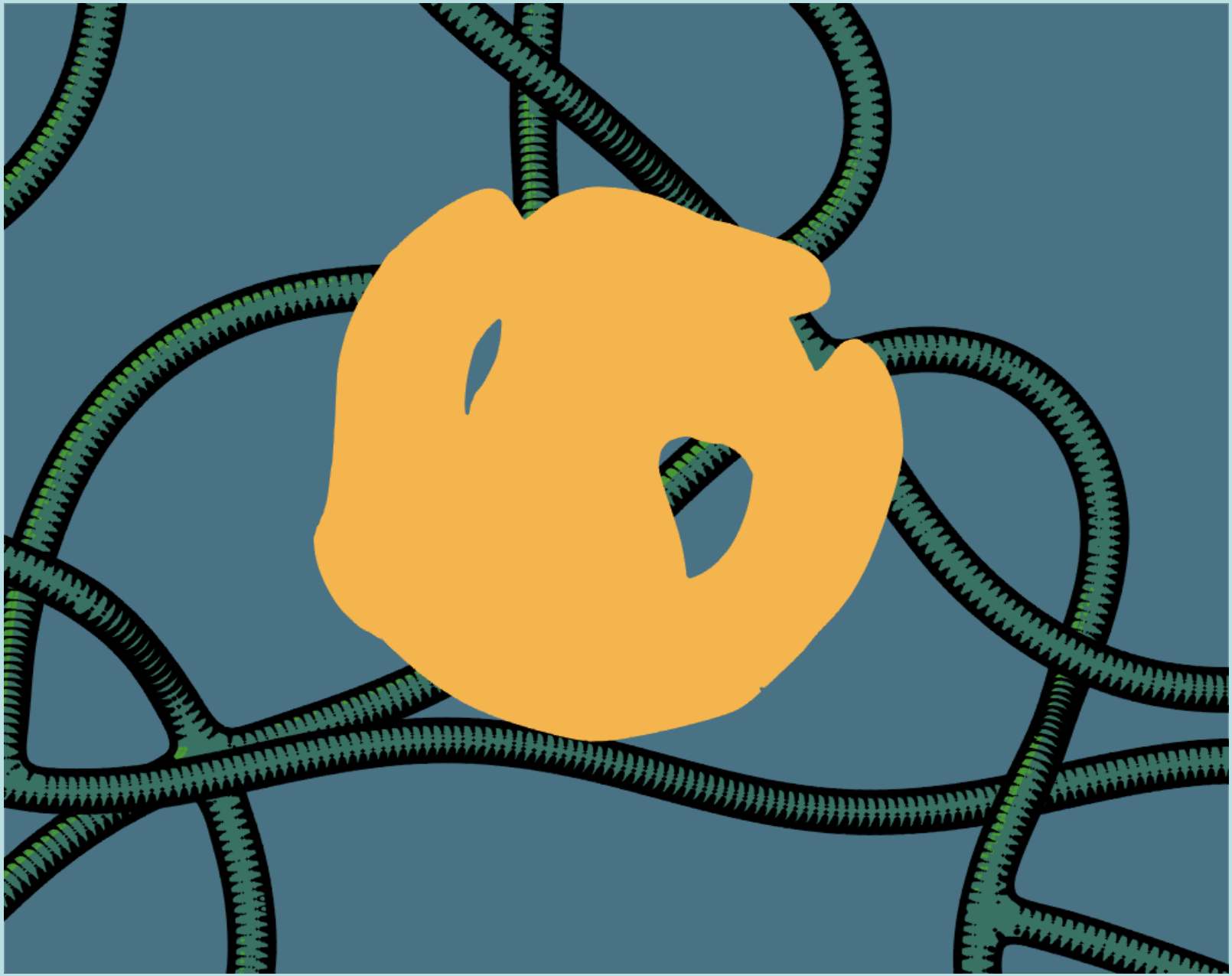


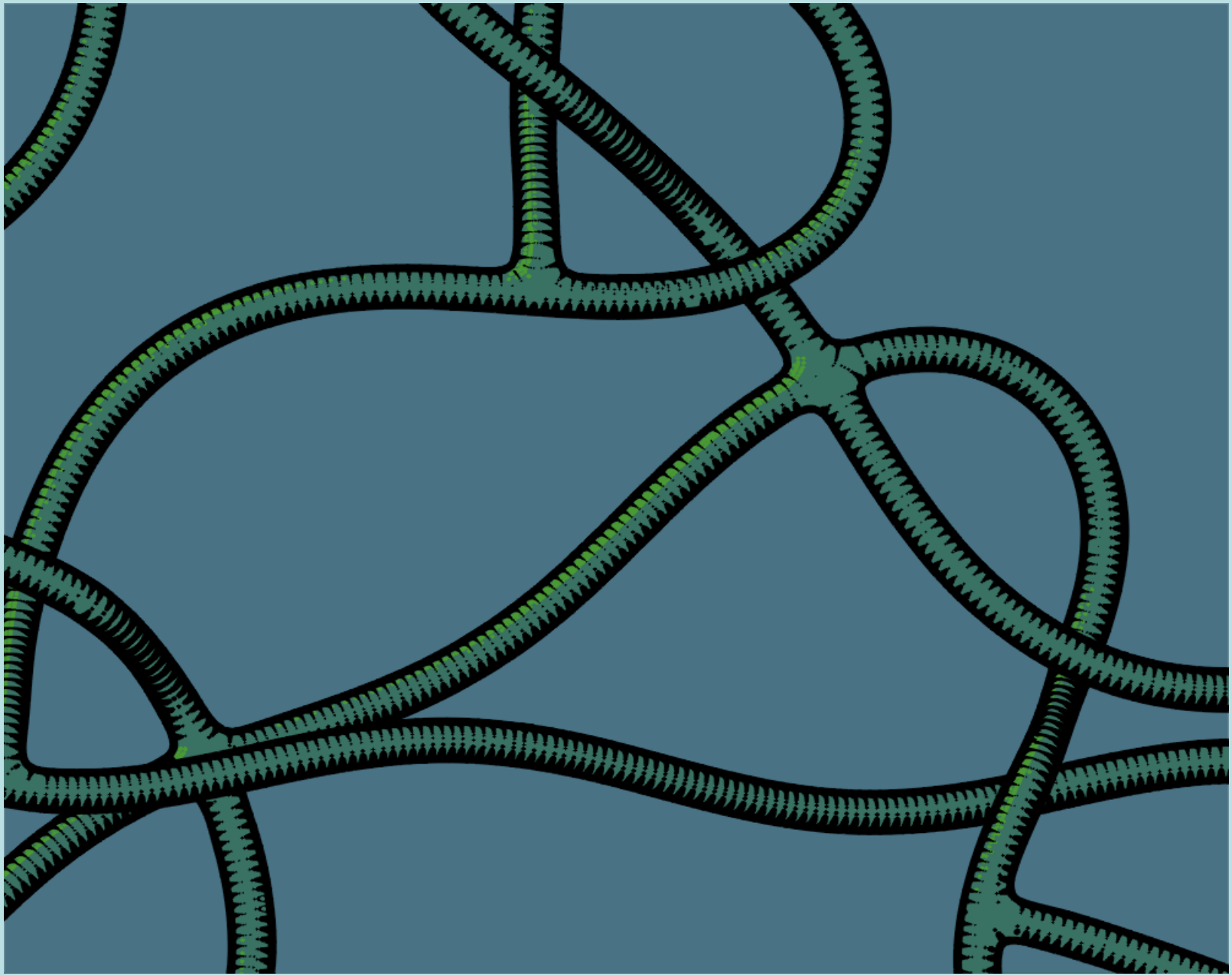




Zander, Isenberg, Schlechtweg, Strothotte
High Quality Hatching – EG'04







Conclusion

- contributions
  - object-space streamline generation as preprocessing
  - optimization of streamlines generated from curvature directions for a polygonal model
  - run-time line shading (line width and stippling) using a virtual machine
  - high quality line rendering techniques tailored to vector-based print output
- future work
  - improved streamline computation
  - more support for cross-hatching

Thank you!