Fillets: Cues for Connections in Focus+Context Views of Graph-like Diagrams

Tuomas J. Lukka and Janne V. Kujala
Hyperstructure Group
Dept. of Mathematical Information Technology
University of Jyväskylä, PO. Box 35
FIN-40351 Jyväskylä
Finland
lukka@iki.fi, jvk@iki.fi

Marketta Niemelä
Dept. of Computer Science and Information Systems
University of Jyväskylä, PO. Box 35
FIN-40351 Jyväskylä
Finland

Abstract

We apply fillets — smoothing of sharp angles at the joints — between the connections and nodes of graph-like diagrams.

In situations where the graph layout is constrained, e.g. Focus+Context views or views where the coordinates of the nodes are informative, fillets can clarify the relationships considerably without altering the layout.

A visual search experiment supports our hypothesis that with fillets it is considerably easier to perceive node-connection structures.

We discuss algorithms with different tradeoffs for flexibility and performance for rendering these connections in a single pass using OpenGL.

1. Introduction

Graph-like diagrams are usually drawn using nodes (generally boxes or circles) and lines between them for connections[**?**]. If the graph is simple (e.g. a tree), or the layout is good, the way of drawing the nodes and edges does not matter much.

However, sometimes connections must cross other connections or nodes frequently. This happens, for example, in Focus+Context[**?**] views of multiply connected (non-planar) graphs, or when the connections have to leave the nodes in specified directions, or when there are other constraints to the layout. The Vanishing View of Gzz[**?**] to the ZigZag structure[**?**] is one example: here, the layout of the nodes (cells) and the directions the connectors leave the cells are determined by the underlying structure: a horizontal connector is different from a vertical one.

In such situations, the pure node-line diagrams are inherently ambiguous, as shown in Fig. 1 a). In order to be able to understand such diagrams if the layout cannot be changed (or if that would lead to other complications), a rendering method which allows the viewer to distinguish between the different possibilities is needed.

One possibility is a technique used in conventional ink drawing techniques where the lines that go behind an object do not actually touch the object, as in Fig. 2.

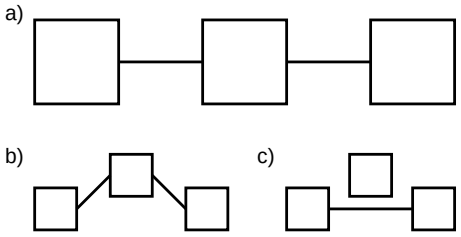

Figure 1. An extreme example of the visual ambiguity inherent in complex boxes-and-lines diagrams when the layout is restricted: a) the ambiguous diagram, b), c) possible meanings. If the diagram was designed by a human, one can be certain that b) is the correct interpretation.

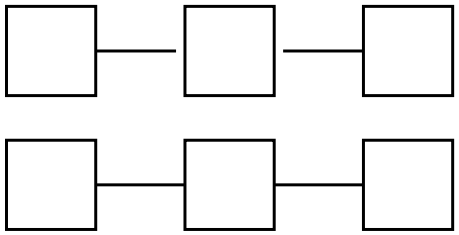

Figure 2. An approach in traditional drawing for showing one element going behind another by erasing the line.

Even though this approach helps, it only allows a human to distinguish between the situations in Fig. 1b) and Fig. 1c) by actually directing his/her focus of attention to the crossing. An explanation can be found in Fig. 5.5 on p.166 of [**?**]: pre-attentive processing cannot distinguish between juncture/non-juncture. The other examples of pre-attentive/non-pre-attentive features in the same book clearly demonstrate that having a pre-attentive cue for the connection/nonconnection could significantly help the viewer to perceive the structure.

In this article, we explore fillets, a novel approach to resolving the ambiguity. The following sections describe fillets and discuss an experiment demonstrating the advantage of fillets in simple graphs. The implementation of fillets us-

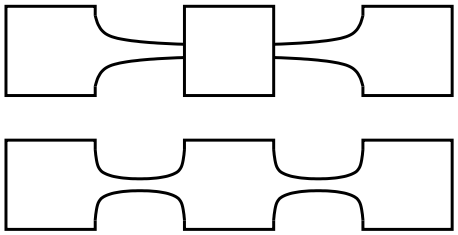

Figure 3. Fillets resolve the ambiguity of Fig. 1 clearly.

ing OpenGL is discussed in the Appendix.

2. Fillets

Filleting, or rounding corners of surfaces, is used in mechanical engineering to improve the properties of cast objects. Sharp corners are fragile and can also cause defects during molding. Filleting is an instance of a more general technique known as *blending* - creating surfaces that meet several existing surfaces smoothly[**?**].

Figure 3 shows how fillets can be used to avoid the ambiguity in Fig. 1. The connection which goes under the middle node seems to do so almost three-dimensionally.

There are two interacting graphical elements in fillets. First, the connection is blended smoothly onto the node, without a derivative discontinuity. Second, there is no black edge on the connection between the two nodes (unless there is something on top of the connection). Together, these two features make it easy to distinguish between the cases where a connection enters a node or goes under it.

A filleted connection conforms to the Gestalt principle of good continuation. Smoothly changing contours enable more efficient perceptual grouping of visual elements [**?**], in this case, grouping of the node and the connection.

This use of fillets is entertainingly analogous to the use in mechanical engineering: fillets ensure that the human perception system doesn't break an object and a connection starting from it into two distinct objects.

One might object that fillets might take more space than standard box-line diagrams. However, the actual space increase is marginal and is compensated by the clarity of the view: nodes can overlap more freely without making it difficult to understand the diagram.

Also, it might be said that the blending makes the background more solid and so the cells and connections might be perceived as the outside of the figure (figure-ground ambiguity). However, cues such as color, text on the nodes, animation, shading, and fog can be used to prevent this.

3. Experiment

A filleted connection consists of two visual features, widening and borderlessness of the connection. This experiment was designed to explore the effect of widening, borderlessness, and gaps (as in Fig. 2) of the connections on search efficiency in node-link graphs. The paradigm of visual search, in which the target search time is measured as a function of the number of distractor elements, was applied (e.g., [**?**], [**?**], [**?**]). Our main hypothesis was that visual segregation between connections entering a node or going behind a node is more effortless with filleted connections than connections without widening, borderlessness (i.e., with borders), or both. Gaps should facilitate search when filleted connections are not implemented. In addition, the effect of the visual width of connections as itself (without fillets) was tested. We expected that the width of the connection would not make a difference on search time.

Table 1. Visual features in the eight conditions. Conditions with filleted connections are indicated in bold text

Condition	Widening	Borderlessness	Gap	Thick
1	**yes**	**yes**	**yes**	**yes**
2	no	yes	yes	yes
3	yes	no	yes	yes
4	no	no	yes	yes
5	no	no	yes	no
6	**yes**	**yes**	**no**	**yes**
7	no	yes	no	yes
8	yes	no	no	yes

A. Method

Participants. Ten participants, seven females and three males, perfomed the experiment for a small monetary reward. Their ages ranged from 19 to 40 years. All of them participated in all conditions.

Design and materials. In the experiment, there were seven conditions with different combinations of the three main features. One variation (no widening, no borderlessness, and no gap) was not implemented since there would be no way to distinguish between a connection entering a node or going behind a node. Instead of this, one condition with narrow line-type connection was implemented. The eight conditions are listed in Table 1 and Fig. 4. Gap width was 1.2 times the width of node border.

On the test figures, nodes were in a regular grid in which all nodes except nodes on the edges and the target node were connected to four neighbouring nodes. At the target node, a connection went behind the node either in vertical or horizontal direction (Fig. 5). Grids of sizes 4x4, 6x6, or 8x8 were tested to find out whether the search time was independent of the set-size, which would indicate that the target node was perceived preattentively. Node size was equal across all conditions and grid sizes.

Procedure. The participants were tested individually. They performed all conditions in a random order. One condition consisted of 24 trials, among which the three different

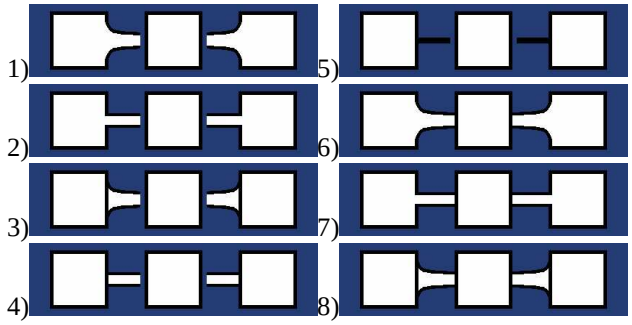

Figure 4. Simple examples of the eight conditions showing a situation in which a connection goes behind a node.

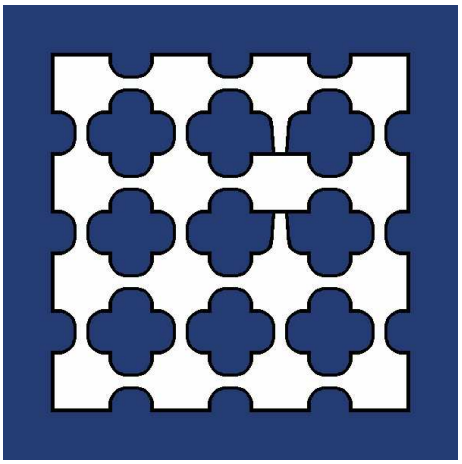

Figure 5. Condition 6 with 4x4 grid size.

grid sizes were presented randomly, eight trials of each size. A trial started with a fixation point appearing on the computer display for 1.5 sec. Then a node grid was shown and it remained on until the participant responded. The location of the target in the grid was randomized (target was never a corner node). The task of the participants was to find the target node and to choose whether the target was on the left or on the right side of the grid. The participant would indicate this by pressing either the left or right control key in the keyboard. The participants were requested to do this task as fast as possible. The feedback was a plus or minus sign on the display, which was shown for 1 sec, thus the pause between two successive trials was 2.5 seconds. Each condition began with a practice block of three trials, one of each size. The participant was allowed to perform the practice block twice if desired. Search time and correctness were collected for each test trial.

B. Results

The search times and errors (Tab. 2) were analysed with the repeated measures analysis of variance.

Search times. The mean correct search times in the

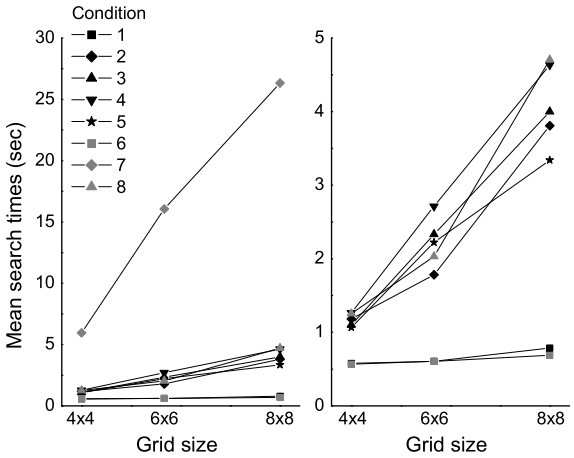

Figure 6. Mean search times in conditions as a function of grid size. The two flat lines are conditions 1 and 6 (both filleted), the steepest slope is in condition 7. The right side is a scaled version of the left side (condition 7 is not visible).

Table 2. Correct search times (msec) and error rates (%) by condition and grid size

	Grid size			
Cond	16	36	64	Mean
1	577 (0.00)	608 (0.00)	786 (0.00)	657 (0.00)
2	1181 (0.00)	1782 (0.00)	3809 (0.00)	2257 (0.00)
3	1100 (1.25)	2332 (1.25)	4000 (2.50)	2477 (1.67)
4	1265 (0.00)	2714 (1.25)	4634 (0.00)	2871 (0.42)
5	1067 (1.25)	2223 (1.25)	3339 (1.25)	2210 (1.25)
6	568 (0.00)	608 (0.00)	687 (0.00)	621 (0.00)
7	5964 (6.25)	16064 (0.00)	26326 (3.75)	16118 (3.33)
8	1254 (0.00)	2032 (1.25)	4705 (0.00)	2663 (0.42)
Mean	1622 (1.09)	3545 (0.63)	6036 (0.94)	3734 (0.89)

eight conditions differed from each other [$F(1.1, 10.2) = 48.0$, $p < .001$]. The search times were clearly fastest in conditions 1 and 6, in which connections were filleted (i.e. both widening and borderlessness) [$F(1, 9) = 93.4$, $p < .001$]. Excluding the obviously different condition 7 did not change this.

The effects of the four visual features - widening, borderlessness, gap, and connection width - were analysed by comparing the conditions in a pairwise fashion. The results of the analyses are collected in Table 3. Widening facilitated search only when combined with borderlessness. Borderlessness was helpful in any condition, except when connection gap was not implemented (again condition 7; see discussion below). Gaps did not affect search time when used with widening, either with or without borderlessness. Width of the connection did not make a difference on search

Table 3. ANOVA table of pairwise comparisons of the conditions. The fixed features column shows which other features were implemented in the compared conditions (W - widening, B - borderlessness, G - gap).

Compared conditions		Fixed features	$F(1,9)$	Statistical significance
no W	W			
2	1	B,G	58.5,	$p < .001$
4	3	G	2.9,	$p > .05$
7	6	B	57.0,	$p < .001$
no B	B			
3	1	G,W	60.4,	$p < .001$
4	2	G	18.2,	$p < .01$
8	6	W	27.0,	$p < .01$
no G	G			
6	1	B,W	1.1,	$p > .05$
7	2	B	46.5,	$p < .001$
8	3	W	0.2,	$p > .05$
narrow	thick			
5	4	G	4.2,	$p > .05$

time.

Grid size had a main effect on search time [$F(2, 18) = 33.5$, $p < .001$], and there was interaction between condition and grid size [$F(1.5, 13.4) = 11.8$, $p < .01$]. Search times as a function of grid size are shown in Fig. 6. Search times were linearly dependent on grid size also in conditions 1 and 6 [$F(2, 18) > 11.2$, $p < .001$ for both conditions]. Thus it seems that although graphs with fillets are undoubtedly most efficient to search, a connection going behind a node is not found during preattentive visual processing but requires attentive search.

Errors. Error rates depend on condition [$F(2.5, 22) = 4.2$, $p < .01$] but not grid size [$F(2, 18) = .6$, $p > .05$]. There is no interaction effect between condition and grid size [$F(4.5, 40.4) = 1.8$, $p > .05$]. Condition 7 was most error-prone (3.3%).

To summarize, the results of the experiment support our hypothesis that in a node-link graph with filleted connections, it is easier to distinguish between connections going either behind a node or into a node. Search in fillet graphs was quite unsensitive to the addition of non-target nodes, even close to parallel (preattentively processed) when compared to searches in the other graphs in the experiment.

We expected a gap between a connection and a node to facilitate search when fillets were not implemented but it did not. Probably the gap width used in the experiment was too small to be effortlessly found. The width of the connection had no effect on search efficiency, as expected.

In condition 7, search times and error rates were especially long when compared to the other conditions. One possible explanation is that a Hermann Grid illusion[**?**] is created in the junction of a connection and a node. This means that if not directly in the focus of the eye, the junction area appears darker than the node or the connection, which makes it hard to distinguish between a target and non-target junctions.

4. Conclusion

There has been much research on graph visualization recently. However, the research seems to have been mostly concentrated on graph and tree layout and Focus + Context methods.

The one significant exception we have found in the literature is the work of Irani, Tingley and Ware on geon diagrams[**?**], [**?**]. Geon diagrams use basic 3D geometric shapes as building blocks[**?**] for diagrams, aiming at making the parts of the diagram more easily recognizable.

However, the geon diagrams do not necessarily help much in the ambiguity of Fig. 1; their point is in making the *types* of connections recognizable, not in helping to understand the *connectivity* itself.

In this article, we have introduced fillets as a way of improving perception of node-line structures. With filleted connections, it is easier to discriminate whether a connection enters a node or goes behind it. Fillets are useful in constrained layouts where the number of crossing edges cannot be minimized for some reason.

Experimentally it was seen that neither of the two graphical subfeatures of fillets, widening and borderlessness, helps much by itself. We suggest that this is because fillets allow the search to be based on the shape of the node, which can be very efficient[**?**]. Fillets are not visual primitive features, but for example Wolfe[**?**] has shown that visual search can be highly efficient even with complex-shaped stimuli. In our experiment, visual search with fillets was very close to parallel (preattentively processed). In other words, fillets are effortless to perceive in node graphs, independent of the number of nodes.

5. Acknowledgments

We would like to thank Benja Fallenstein and Prof. John Canny for comments on this manuscript.

References

[1] Paul Haeberli and Mark Segal. Texture mapping as a fundamental drawing primitive. In Michael F. Cohen, Claude Puech, and Francois Sillion, editors, *Fourth Eurographics Workshop on Rendering*, pages 259–266, 1993.

[2] Ivan Herman, Guy Melancon, and M. Scott Marshall. Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):24–43, 2000.

[3] Glyn W. Humphreys and Vicki Bruce. *Visual cognition: Computational, Experimental, and Neuropsychological Perspectives*. Lawrence Erlbaum Associates Ltd., 1989.

[4] Biederman I. Recognition–by–components: A theory of human image understanding. *Psychological Review*, 94(2):115–147, 1987.

[5] P. Irani and C. Ware. Diagrams based on structured object perception. In *Proceedings of Advanced Visual Interfaces, AVI'2000, Palermo, Italy*, pages 61–67, 2000.

[6] Pourang Irani, Maureen Tingley, and Colin Ware. Using perceptual syntax to enhance semantic content in diagrams. *IEEE Computer Graphics and Applications*, 21(5):76–85, 2001.

[7] Y. K. Leung and M. D. Apperley. A review and taxonomy of distortion-oriented presentation techniques. *ACM Transactions on Computer-Human Interaction*, 1(2):126–160, 1994.

[8] Tuomas J. Lukka et al. Gzz. http://gzz.info/, http://savannah.gnu.org/projects/gzz.

[9] Ted Nelson. Confidential presentation of Ted Nelson's ZigZag™ and Ted Nelson's Dimensia™. http://www.xanadu.net/zigzag/manual/, March 1994.

[10] Alyn Rockwood. Chapter 6: Blending. In Jules Bloomenthal, editor, *Introduction to Implicit Surfaces*. Morgan Kaufmann, 1997.

[11] M. Segal and K. Akeley. The design of the OpenGL graphics interface. Technical report, Silicon Graphics Computer Systems, 1994.

[12] Lothar Spillmann. The hermann grid illusion: A tool for studying human perceptive field organization. *Perception*, 23(6):691–708, 1994.

[13] Anne Treisman and Garry Gelade. A feature-integration theory of attention. *Cognitive Psychology*, 12:97–136, 1980.

[14] Anne Treisman and Stephen Gormican. Feature analysis in early vision: evidence from search asymmetries. *Psychological Review*, 95:15–48, 1988.

[15] C. Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann, 2000.

[16] Jeremy M. Wolfe. Guided search 2.0. a revised model of visual search. *Psychonomic Bulletin and Review*, 1:202–238, 1994.

[17] Jeremy M. Wolfe. Visual search in continuous, naturalistic stimuli. *Vision Research*, 34(9):1187–1195, 1994.

Appendix

1. Implementing fillets using OpenGL

OpenGL, while often associated with 3D graphics, is designed from ground up to be a general *rendering* library[**?**], offering a selection of mutually orthogonal features such as Z-buffering, texture mapping, curved surfaces (through evaluators) and lighting.

It is possible to draw the filleted graph in one pass, using the Z buffer to make the resulting image independent of the order of rendering the nodes and connections.

A. Pre-rendered connections

If all connectors start and end in a few specific ways and the connectors' beginnings and ends do not overlap, it is possible to pre-render all connectors' ends into 2D textures[**?**] with an alpha (transparency) channel, as in Fig. 7.

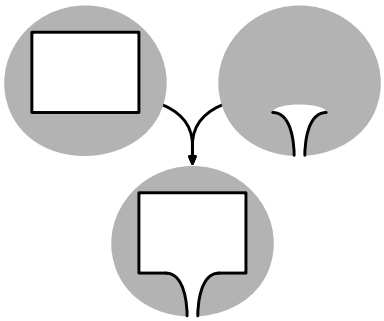

Figure 7. The pre-rendered textures used in alpha compositing in the first algorithm. The connector can be stretched outside the node as desired using e.g. a NURBS surface.

The texture of the connector is designed so that it smoothly joins the edge of the node.

Rendering a connection then is a simple matter of rendering the connector slightly closer to the eye than the node is, stretched using e.g. NURBS to meet the connector coming from the other node.

This is the algorithm used in our experiment.

B. Blending at runtime

In the previous algorithm the relationships of the connectors and nodes are fixed beforehand by prerendering the connections into textures. In this section, we discuss an algorithm which uses polygons to shape the fillets, trading some performance for generality.

First of all, we use bevels and 1D textures for drawing figures with edges. The bevels make the 1D border textures of two figures drawn at the same depth meet appropriately by using the Z-buffer.

In the literature, there is a wealth of blending algorithms for curves and surfaces[**?**]. There is nothing conceptually new about the algorithm presented here; it is tailored for efficiently rendering fillets.

When direction of the connector is rotated around the node, the resulting animation should be smooth; an artifact such as popping when the connector goes over a corner is distracting and undesirable.

To avoid discontinuities when rotating the connection, we distort the boundary curve continuously as shown in Fig. 8. The displacement r in the Figure is defined by

$$r = f(t) \cdot s. \tag{1}$$

where $f(t)$ is the blending function, chosen on aesthetic grounds. In Fig. 8 a quarter arc of a circle was used for f.

Then, the distorted curve is drawn with the same bevel, texture and color as the original node. Different connections to the same node will join appropriately, as seen in Fig. 9.

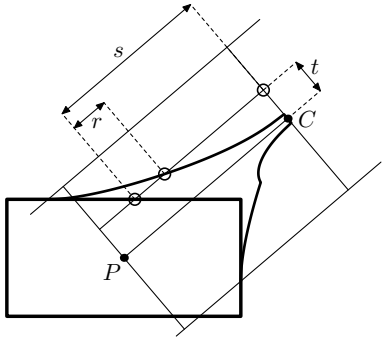

Figure 8. The general algorithm geometrically. The rectangle is to be connected, with the middle of the connector at C and the starting location inside the rectangle at P. The edge of the rectangle is displaced in the direction of PC according to Eq. (1). The angle of the rectangle is visible on the displaced curve.

This algorithm can also use OpenGL lighting to achieve a true 3D look by adjusting the normals at the bevels, as seen in Fig. 10. Note that in OpenGL, the surface normals are specified separately from the surface geometry; the actual varying Z depths of the different cells do not affect the lighting.

In Fig. 11, the same graph is rendered more conventionally for comparison.

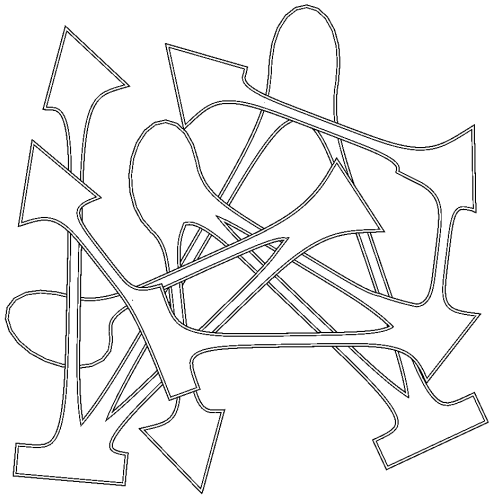

Figure 9. A random, badly laid out graph rendered using the general algorithm. The image has been rendered with narrow, double lines at the edge to show that a general 1D texture can be used at the edges without problems, due to the beveling and Z-buffering. Unfortunately this degrades the quality of the image on paper somewhat.

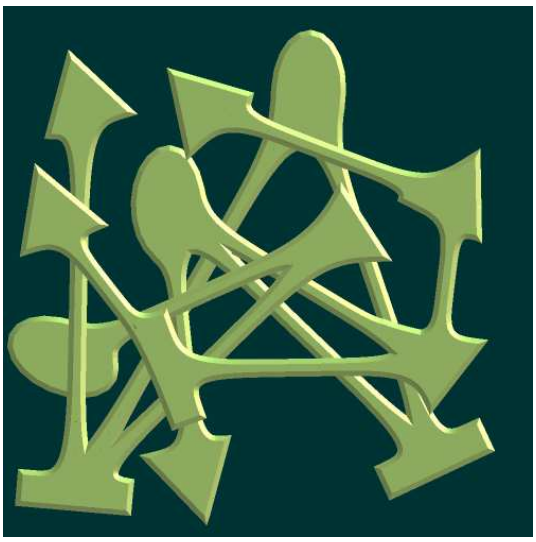

Figure 10. The same scene rendered using the general algorithm, with beveling using OpenGL lighting.

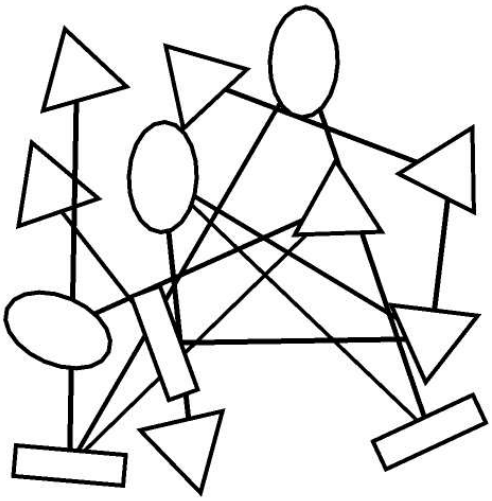

Figure 11. The same scene rendered in the conventional way with lines.