Alan Tsai

CS 430 – Computer Graphics
Fall 2016
Assignment 6 – 3D Wireframe Drawing

In this assignment you will demonstrate your understanding of 3D graphics as it pertains to polygon mesh drawing and SMF file reading.

In this assignment you should be able to:

1. Read an SMF file containing specifications for a polygonal 3D mesh.
2. Store 3D mesh data in homogenous coordinates
3. Apply the graphics pipeline for drawing 3D specified polygons. In particular:
    a. Project points to a canonical view.
    b. Trivially test polygons against a canonical view volume.
    c. Project points to an orthogonal canonical view.
    d. Go from an orthogonal canonical view to 2D view plane/window
    e. Transform view plane to viewport
    f. Draw viewport into software framebuffer.

Make sure you give yourself adequate time. The programming components in particular can be quite time consuming.

As a reminder you may use the programming language of your choice, though I recommend C/C++ and also make sure that your program can run on the Drexel tux cluster to insure its system independence.

Submission Guidelines

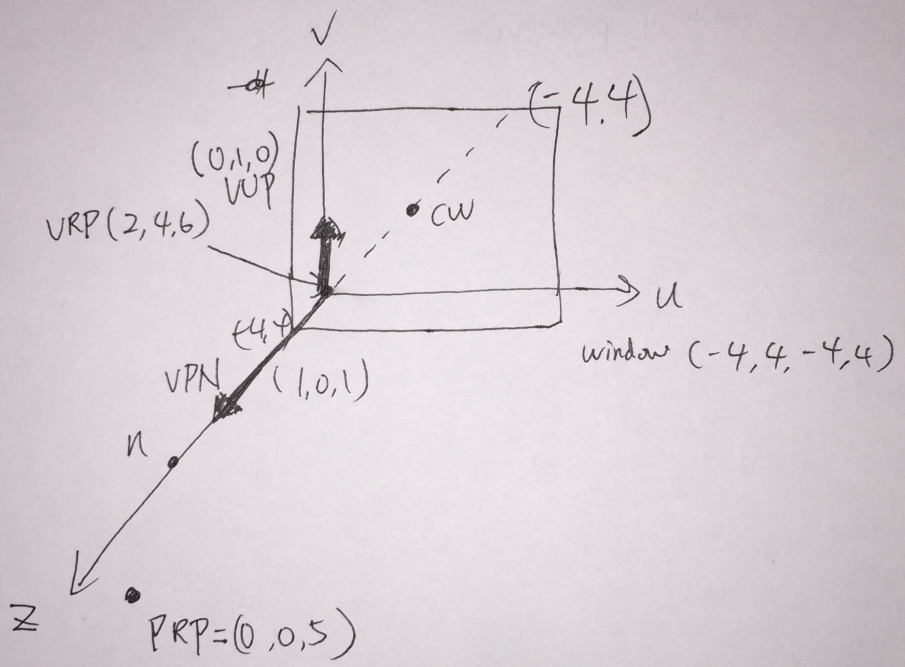
1. Assignments must be submitted via Bd Learn
2. Submit a single compressed file (zip, tar, etc..) containing:
    a. A PDF file with your solutions to the theory questions.
    b. A README text file (**not** Word or PDF) that explains
        i. Features of your program
        ii. Language and OS used
        iii. Compiler or interpreter used
        iv. Name of file containing main()
        v. How to compile/link your program
    c. Your source files and any necessary makefiles, scripts files, etc… to compile and run your program.

Theory Question(s)

1. Given the following camera parameters, draw the camera set up as related to the world coordinate system. (4pts)

```
VRP = (2, 4, 6)
VRN = (1, 0 1)
VUP = (0, 1, 0)
PRP = (0, 0, 5)
Window = (-4, 4, -4, 4)
```

VRP = (2, 4, 6)
VRN = (1, 0, 1)
VUP = (0, 1, 0)
PRP = (0, 0, 5)
window = (-4, 4, -4, 4)



Simple Model Format (*.smf)

For the last two assignments we will reading from a different file type, one that supports specifications for polygonal meshes. The format we will us is called the *Simple Model Format* (SFM).
SMF is a standard file format for specifying meshes. You do not have to implement a complete specification in SMF, just the and rules.

Make sure that you ignore any other commands in the input file. A *v* denotes a vertex, and an *f* denotes a face. Three floating point numbers follow a *v* that define the *x*, *y*, and *z* positions of the vertex. Integer references to the vertices follow an *f*. The references assume the order of the vertices specified in the file. The vertices for a face are defined in counter-clockwise order. The numbering of the vertices begins at 1.

Here's a small example of the format:

```
v -1.0 -1.0 -1.0
v -2.0 -3.0 -4.0
…….
f 1 2 3
f 3 4 1
```

Assignment Details

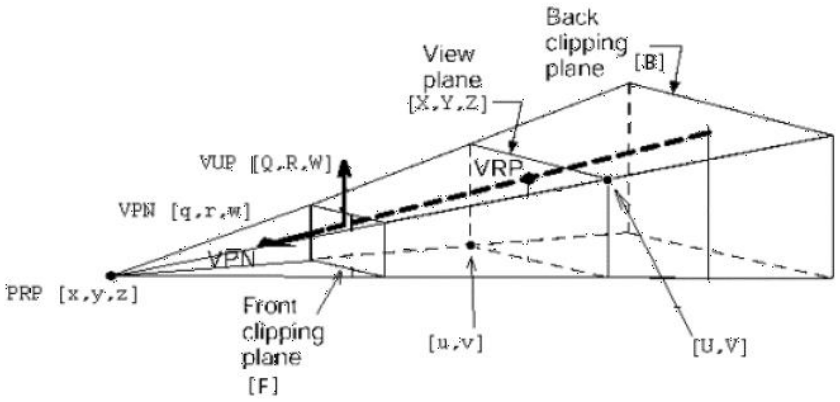
Write a program that accepts the following command arguments. Defaults are in parenthesis. You should be able to process any subset of the options in any arbitrary order.

- [-f] The next argument is the input SMF file
- [-j] The next argument is an integer lower bound in the x dimension of the viewport (0)
- [-k] The next argument is an integer lower bound in the y dimension of the viewport (0)
- [-o] The next argument is an integer upper bound in the x dimension of the viewport window (499)
- [-p] The next argument is an integer upper bound in the y dimension of the viewport window (499)
- [-x] The next argument is a floating point value for the x of the Projection Reference Point (PRP) in VRC coordinates. (0.0)
- [-y] The next argument is a floating point value for the y of the Projection Reference Point (PRP) in VRC coordinates. (0.0)
- [-z] The next argument is a floating point value for the z of the Projection Reference Point (PRP) in VRC coordinates. (1.0)
- [-X] The next argument is a floating point value for the x of the View Reference Point (VRP) in VRC coordinates. (0.0)
- [-Y] The next argument is a floating point value for the y of the View Reference Point (VRP) in VRC coordinates. (0.0)
- [-Z] The next argument is a floating point value for the z of the View Reference Point (VRP) in VRC coordinates. (0.0)
- [-q] The next argument is a floating point value for the x of the View Plane Normal (VPN) in VRC coordinates. (0.0)
- [-r] The next argument is a floating point value for the y of the View Plane Normal (VPN) in VRC coordinates. (0.0)
- [-w] The next argument is a floating point value for the z of the View Plane Normal (VPN) in VRC coordinates. (-1.0)
- [-Q] The next argument is a floating point value for the x of the View Up Vector (VUP) in VRC coordinates. (0.0)
- [-R] The next argument is a floating point value for the y of the View Up Vector (VUP) in VRC coordinates. (1.0)
- [-W] The next argument is a floating point value for the z of the View Up Vector (VUP) in VRC coordinates. (0.0)
- [-u] The next argument is a floating point value for the u min of the VRC window in VRC coordinates. (-0.7)
- [-v] The next argument is a floating point value for the v min of the VRC window in VRC coordinates. (-0.7)
- [-U] The next argument is a floating point value for the u max of the VRC window in VRC coordinates. (0.7)
- [-V] The next argument is a floating point value for the v max of the VRC window in VRC coordinates. (0.7)
- [-P] Is a flag to indicate to use parallel projection. If not present, use perspective projection

Additional things to note:

- For this assignment you may fix the front and back clipping planes to be 0.6 and -0.6, respectively.
- Your frame buffer size should be 500x500

The command line options correspond to the value in the image below.



The general program flow should be:

1. Read an SMF file containing specifications for a polygonal 3D mesh.
2. Compute the necessary view and projection matrices based on command line options.
3. Perform clipping in 3D. **Note for this assignment you only need to detect if a 3D polygon is completely within the view volume**. If it is not (because it's either fully outside or partially outside) then you may simply discard it (extra credit if you actually do 3D clipping).
4. Project the accepted 3D polygons (currently in an canonical view) to 2D default view plane
5. Map the vertices from default view plane coordinates to viewport coordinates.
6. Rasterize the lines that define the 2D polygons to your framebuffer and output framebuffer as an XPM image.

Extra Credit

For ten extra credit points, if you detect that the polygon actually clips against the canonical view volume, then do 3D Sutherland-Hodgman Polygon Clipping.

Grading Scheme:

In order to get any credit at all you must be able to generate at least read in a PS file and write out a valid XPM file

1. Theory Questions (4pts)
2. Are able to compute view and projection matrices given parameters (30pts).
3. Can to trivial accept/deny clipping tests (20pts).
4. Can do both parallel and perspective projection (26pts).
5. Can perform viewport transformation correctly (20pts)
6. Extra Credit (10pts)

Common deductions:

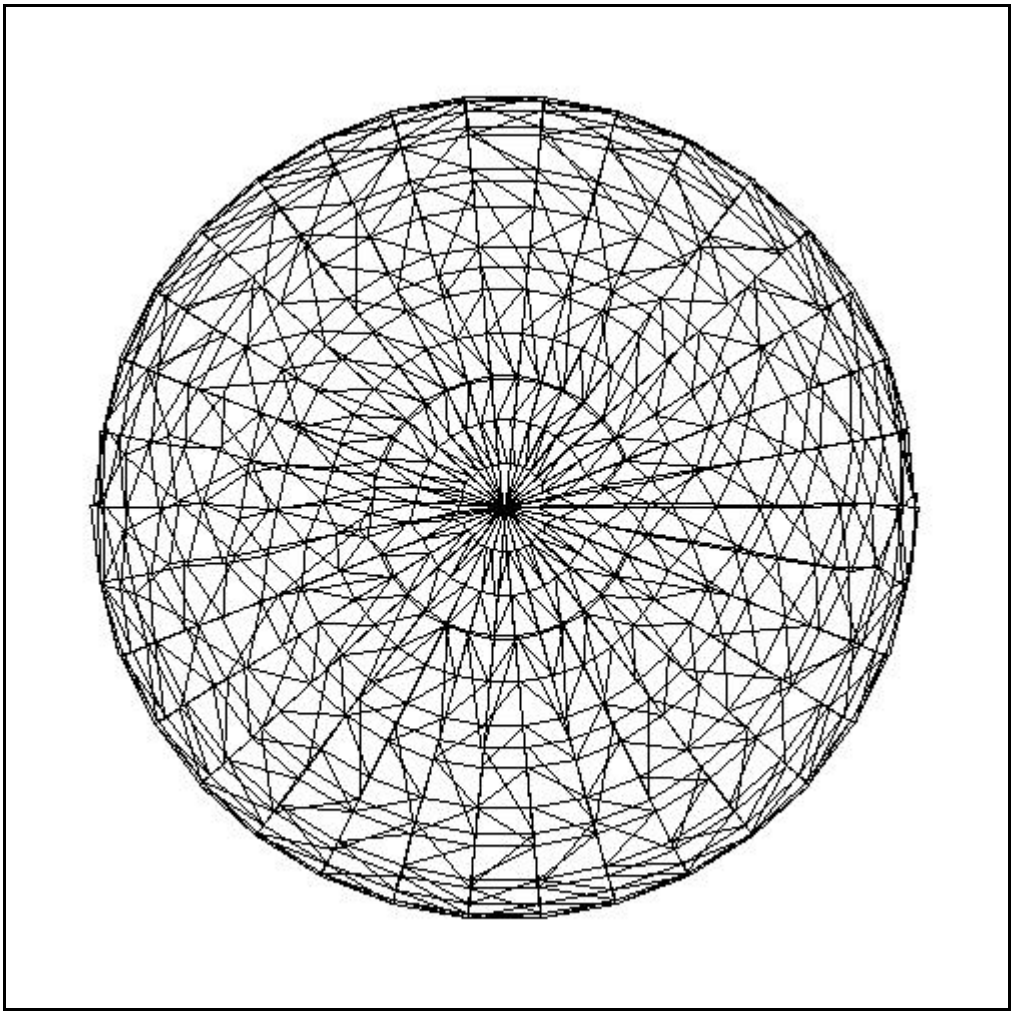
1. Nothing drawn (-100pts)
2. Implementation not done in homogenous coordinates (-40pts)
3. Missing files (including readme) (-5pts each)
4. Cannot compile/run out-of-the-box on TUX (-10pts)

Provided Tests

As usual we will provide several tests for you and may perform addition ones when grading you. Here are some tests to help you gauge the correctness of your implementation:

Default settings: ./A6 -f bound-lo-sphere.smf > out1.xpm

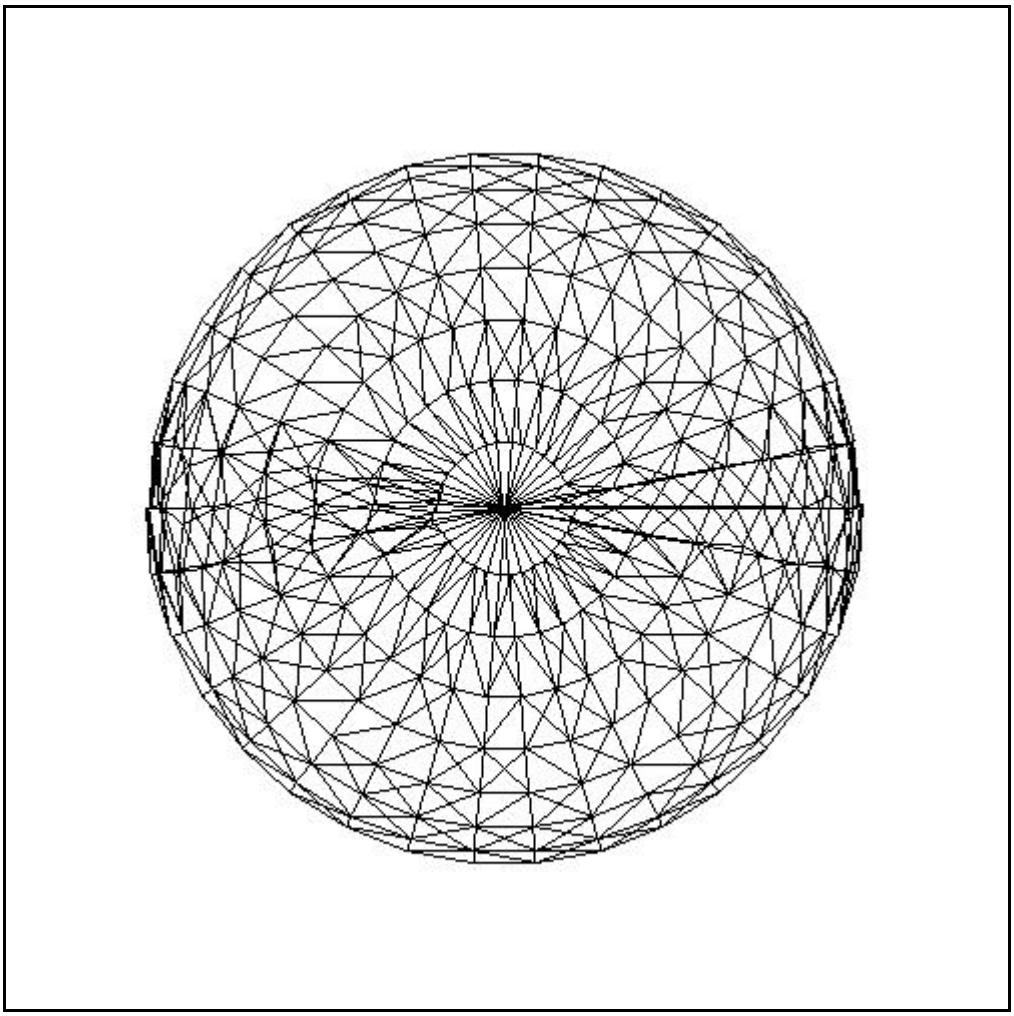
$$\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.89 & 0 & 0 & 0 \\ 0 & 0.89 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



$$\begin{matrix} -1 & 0 & 0 & 0 \end{matrix} \qquad \begin{matrix} 1.43 & 0 & 0 & 0 \end{matrix}$$

Default settings with Parallel: ./A6 -f bound-lo-sphere.smf -P > out2.xpm

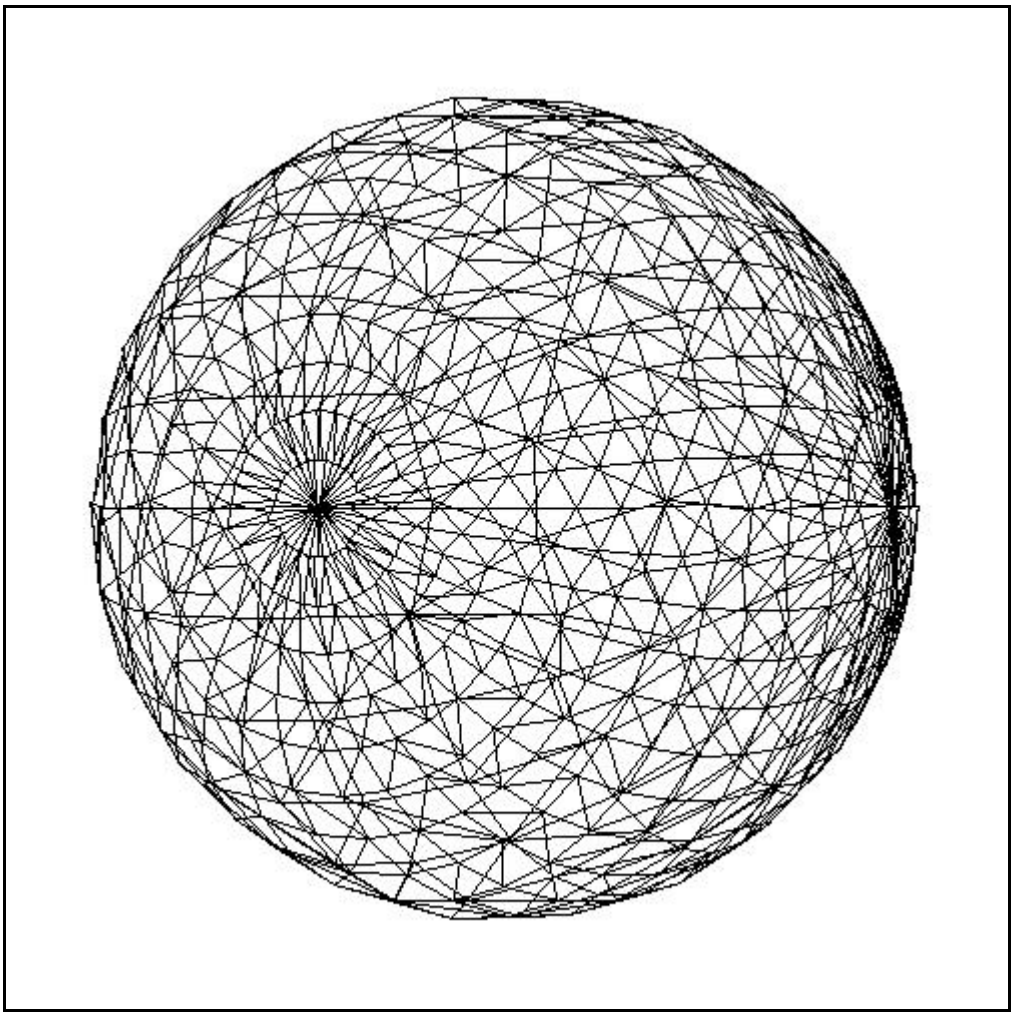
$$= \begin{matrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}, \qquad = \begin{matrix} 0 & 1.43 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$



Let’s rotate the camera so that it’s pointing in the direction$(-1, 0\ 1)$. Therefore let’s set=

$$(1,0,-1) \quad \begin{matrix} -0.71 & 0 & -0.71 & 0 \\ 0 & 1 & 0 & 0 \\ \end{matrix} = \begin{matrix} 0.89 & 0 & 0 & 0 \\ 0 & 0.89 & 0 & 0 \\ \end{matrix}$$

./A6 -f bound-lo-sphere.smf 0 -q 1.0 0 >out3.xpm0 1 0 0 0 1



$= (1,0, -0.5)$ $=$ $(0, 0, 1)$

.

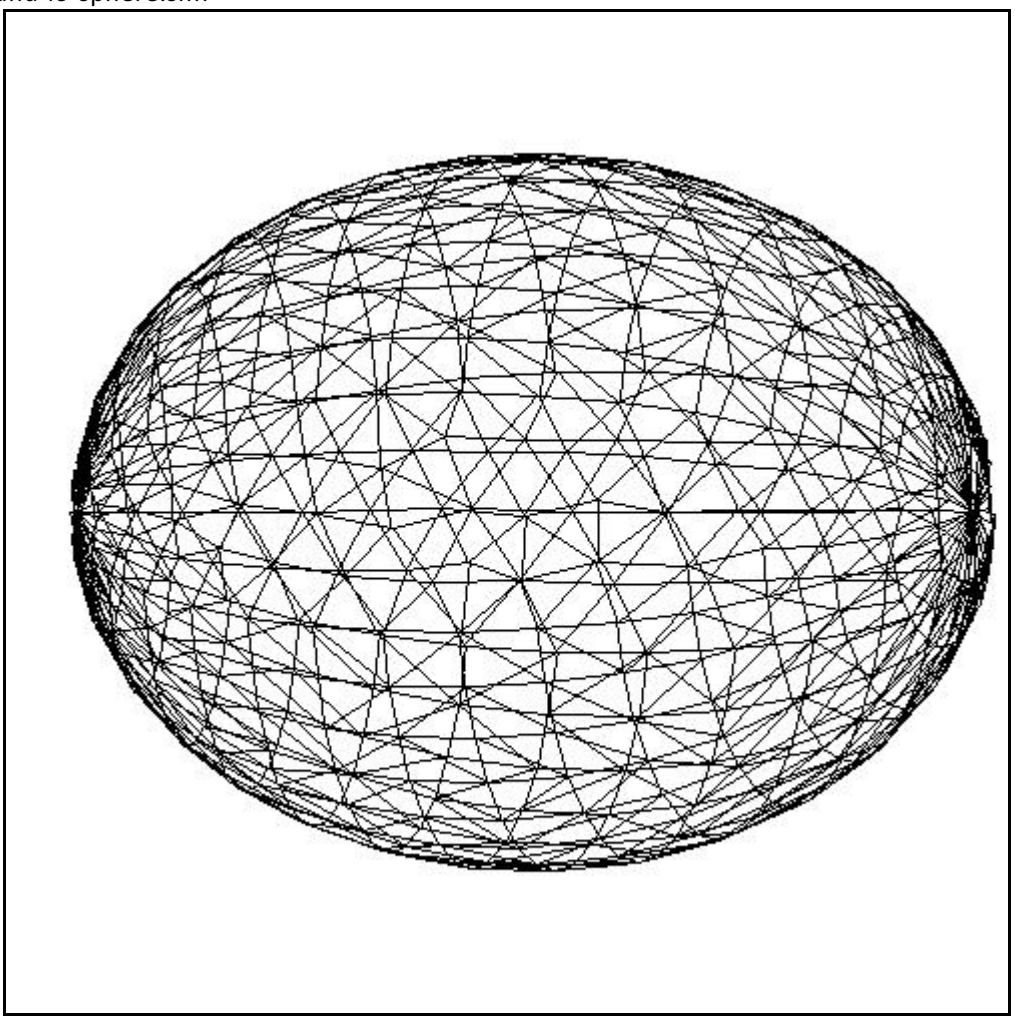
Next let's set the and move the projection reference point from to

The former will result in pointing the camera in the direction of , the latter will result in longer focal length with shearing (since it's off-axis).

$$= \begin{matrix} -0.45 & 0 & -0.89 & 0 \\ 0 & 1 & 0 & 0 \\ \cdots & \cdot & \cdots & \cdot \end{matrix} \begin{matrix} 1.28 & 0 & 1.0 & 0 \\ 0 & 1.28 & 0 & 0 \\ \cdot & \cdot & \cdots & \cdots \end{matrix}$$

^{0}-x -4.0^{0}-z 5.0 ^{0}-q 1.0 ^{1}-w -0.5 >out4.xpm^{0} 0 0 1

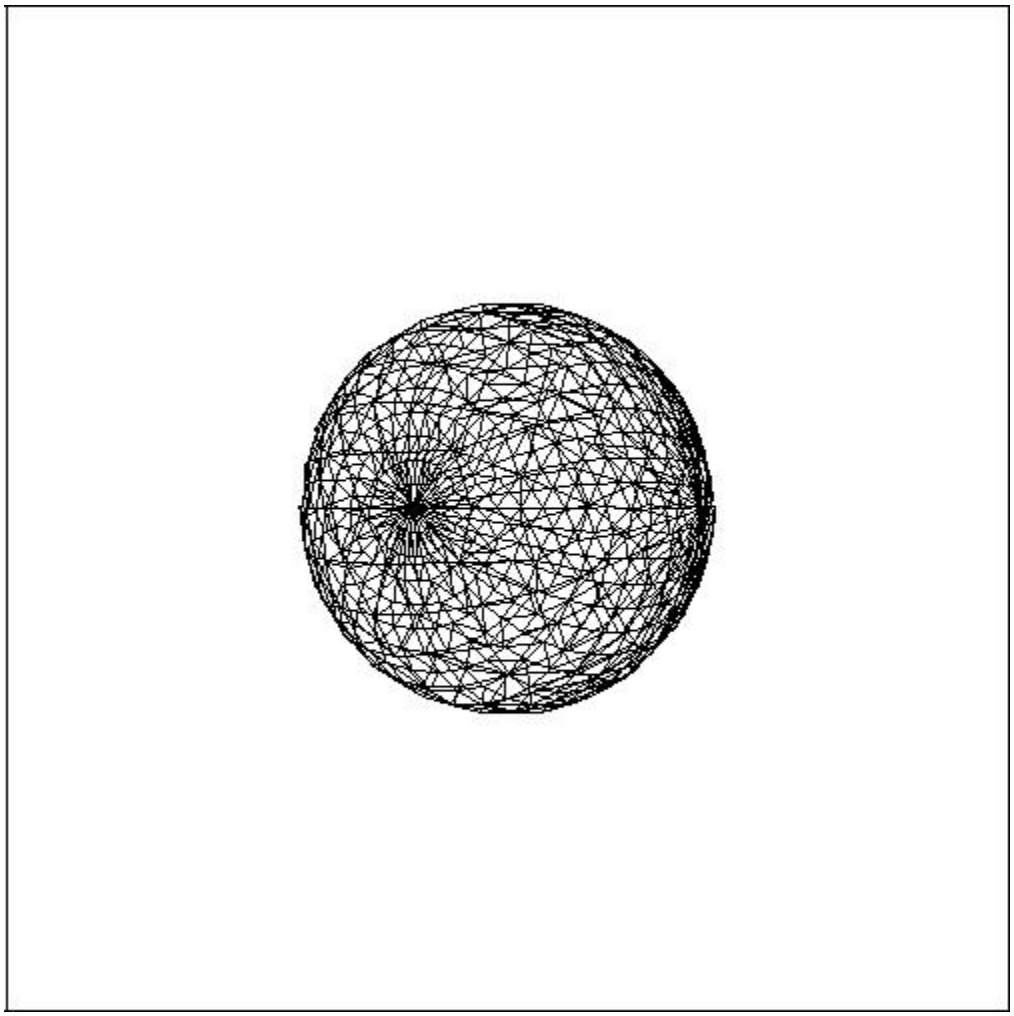
./A6 -f bound-lo-sphere.smf



$(1,0,-1)$ $=$, let's set the bounds of the VRC window to be (-1.4, -0.7) to (1.4, 0.7) and set Next let's play with the viewport and VRC window. First let's rotate our camera so that

$^{(0,125)}$ the viewport to go from to (499,375)

./A6 -f bound-lo-sphere.smf -k 125 -p 375 -q 1.0 -u -1.4 -U 1.4 >out5.xpm



, set the bounds of the view plane to be from (-0.35,-0.35) to (0.7,0.7), $=$

$(0.35, -0.3, 0.3)$

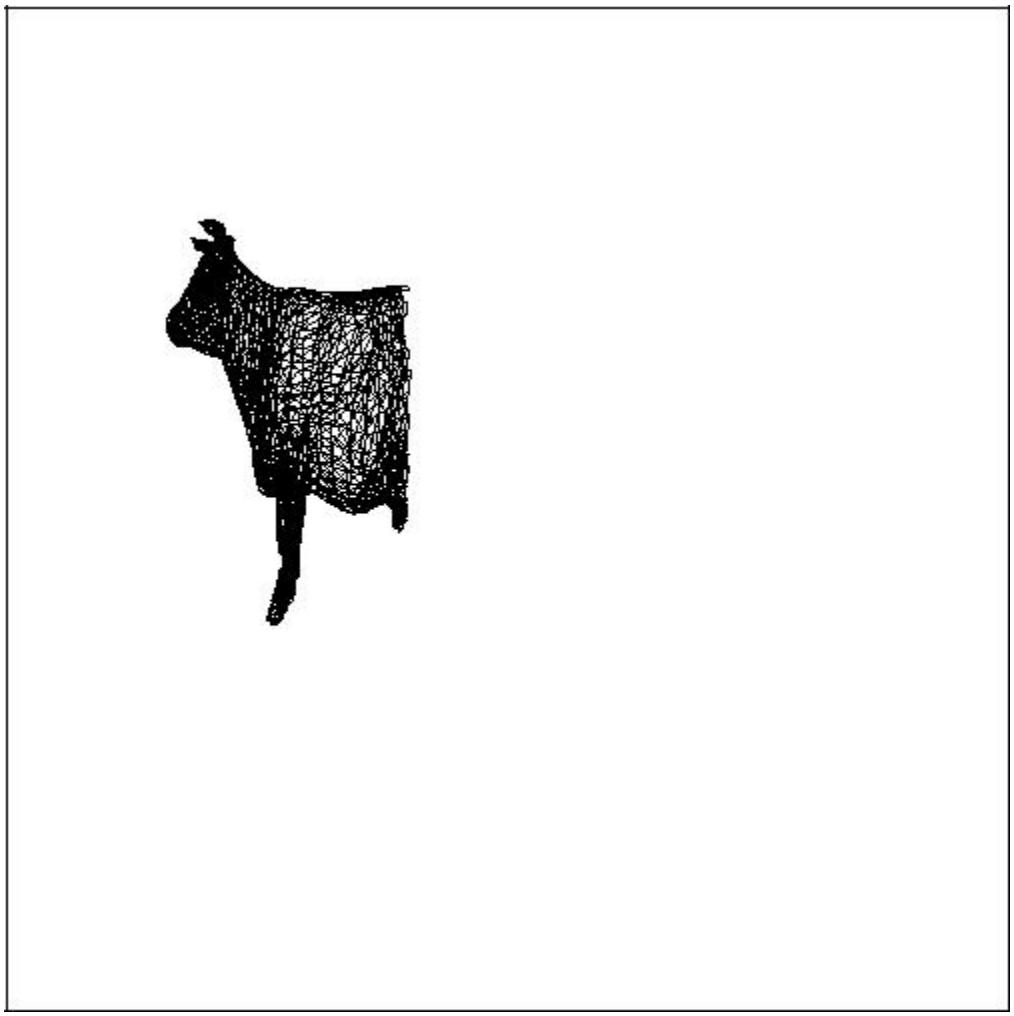
As one last example, let's play with a different mesh. Let's move the camera so that the

set the viewport

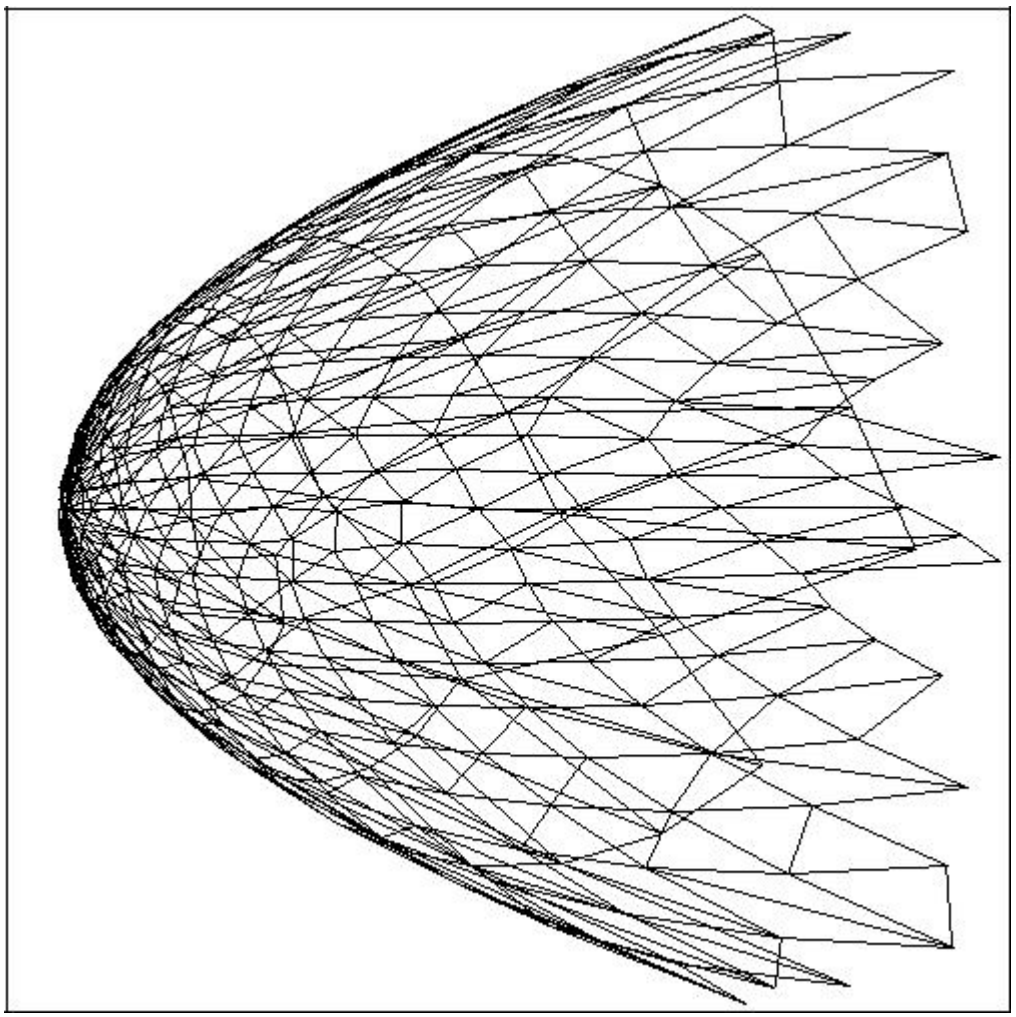
to extend from (43, 71) to (201, 402), and do parallel projection

$$\begin{bmatrix} -1 & 0 & 0 & 0.35 \\ 0 & 1 & 0 & 0.3 \\ \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1.90 & 0 & 0.33 & -0.33 \\ 0 & 1.90 & 0.33 & -0.33 \\ \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & 1 \end{bmatrix} \blacklozenge$$

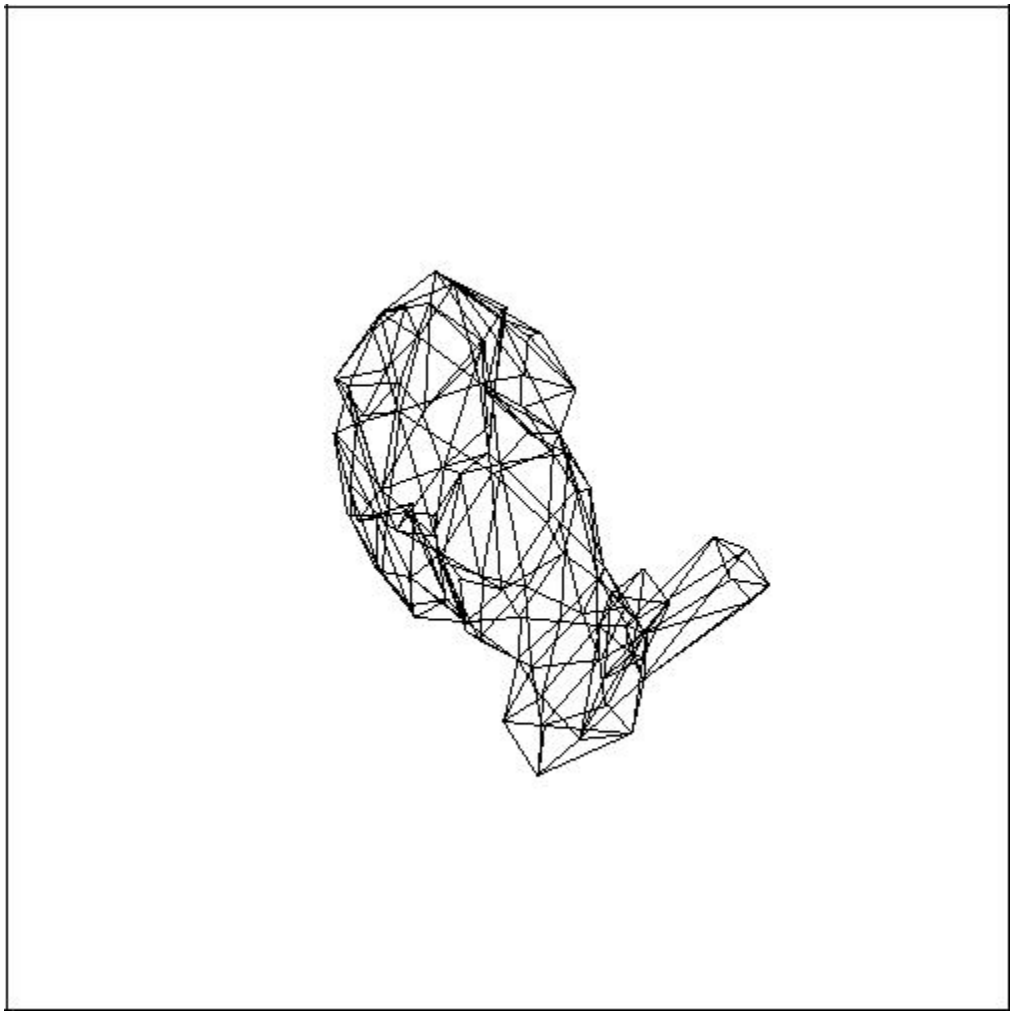
./A6 -f bound-cow.smf -X 0.35 -Y -0.3 -Z 0.3 -u -0.35 -v -0.35 -j 43 -k 71 -o 201 -p 402 -P >out6.xpm



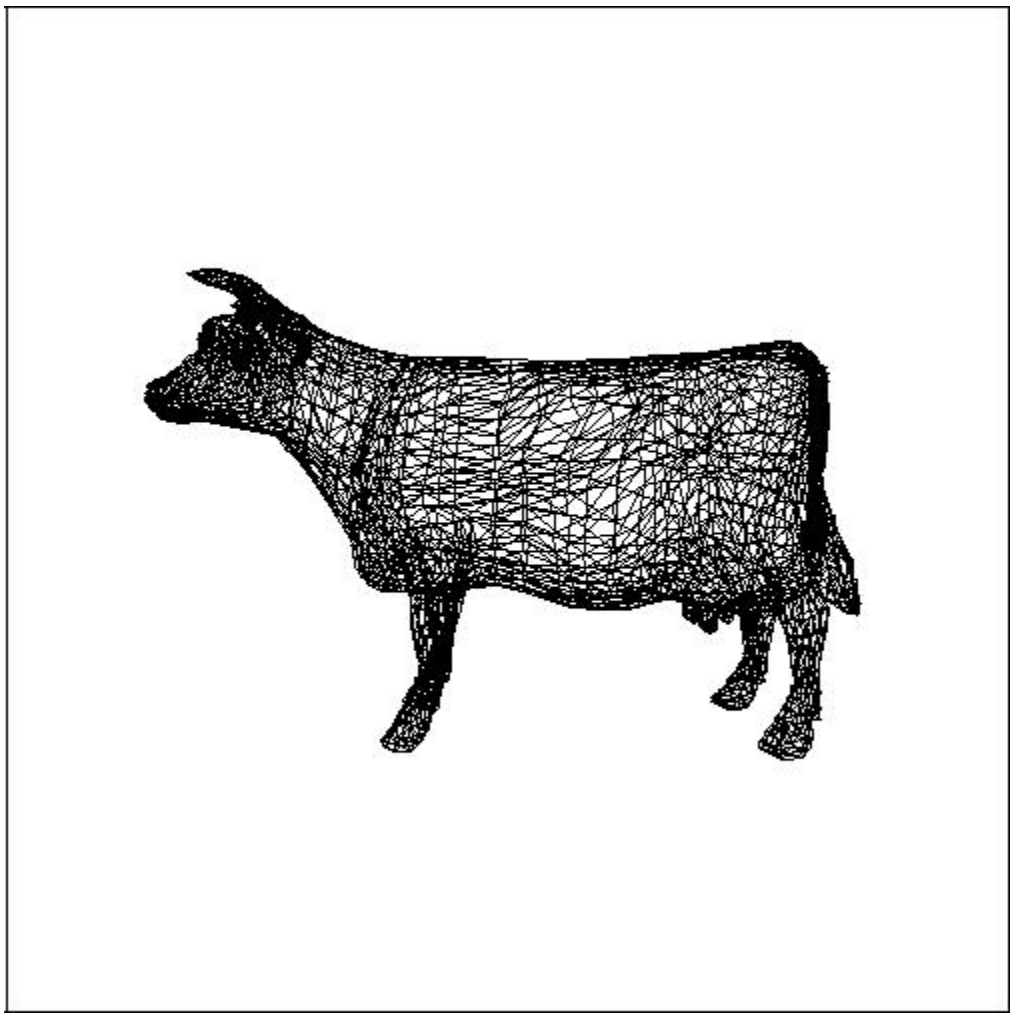
And here's some more tests….
./A6 -f bound-lo-sphere.smf -x -1.0 -z 0.5 -q 1.0 -w -0.5 >out7.xpm



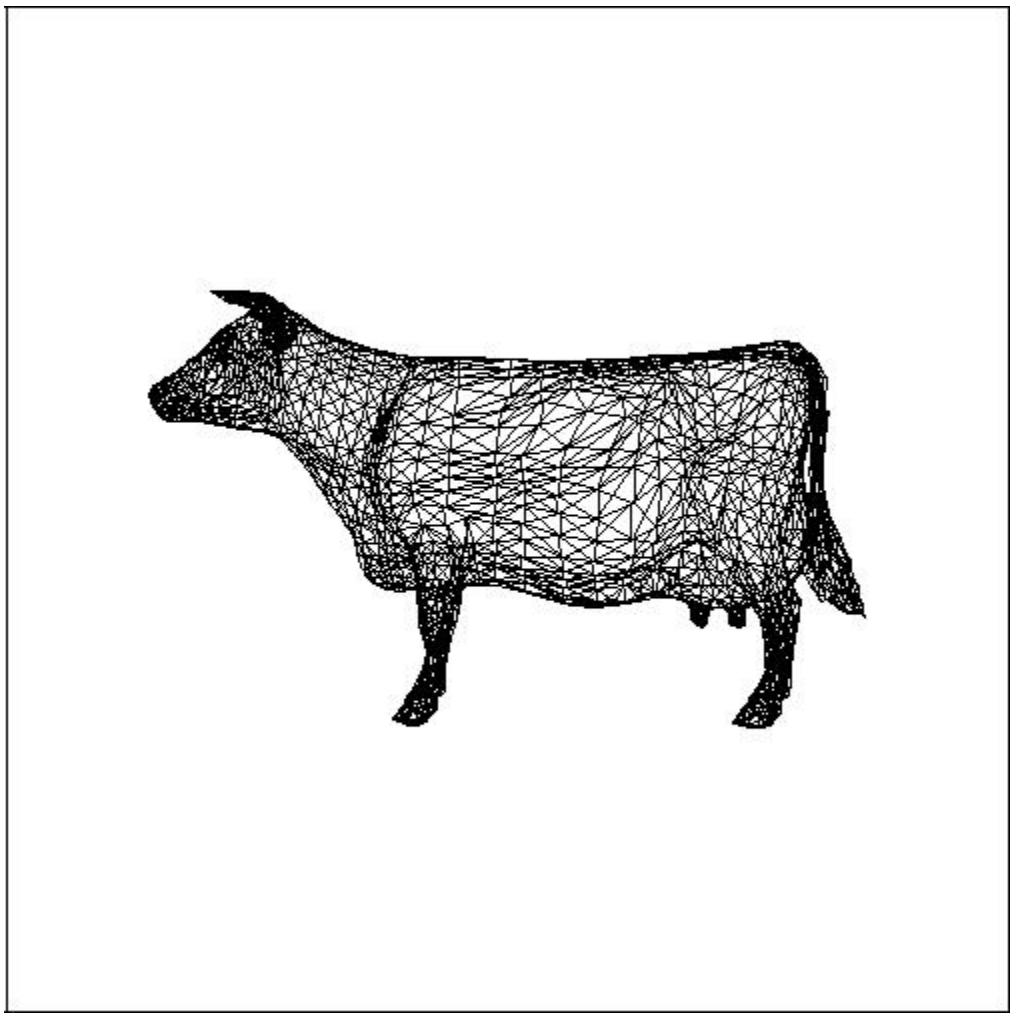
./A6 -f bound-bunny_200.smf -j 100 -k 50 -o 400 -p 450 -x 0.5 -y 0.2 -z 1.0 -X 0.2 -Y -0.2 -Z 0.3 -q -3.0 -r - 2.0 -w 1.0 -Q 3.0 -R -2.0 -W -4.0 -u -0.5 -U 1.2 -V 0.8 -P >out8.xpm



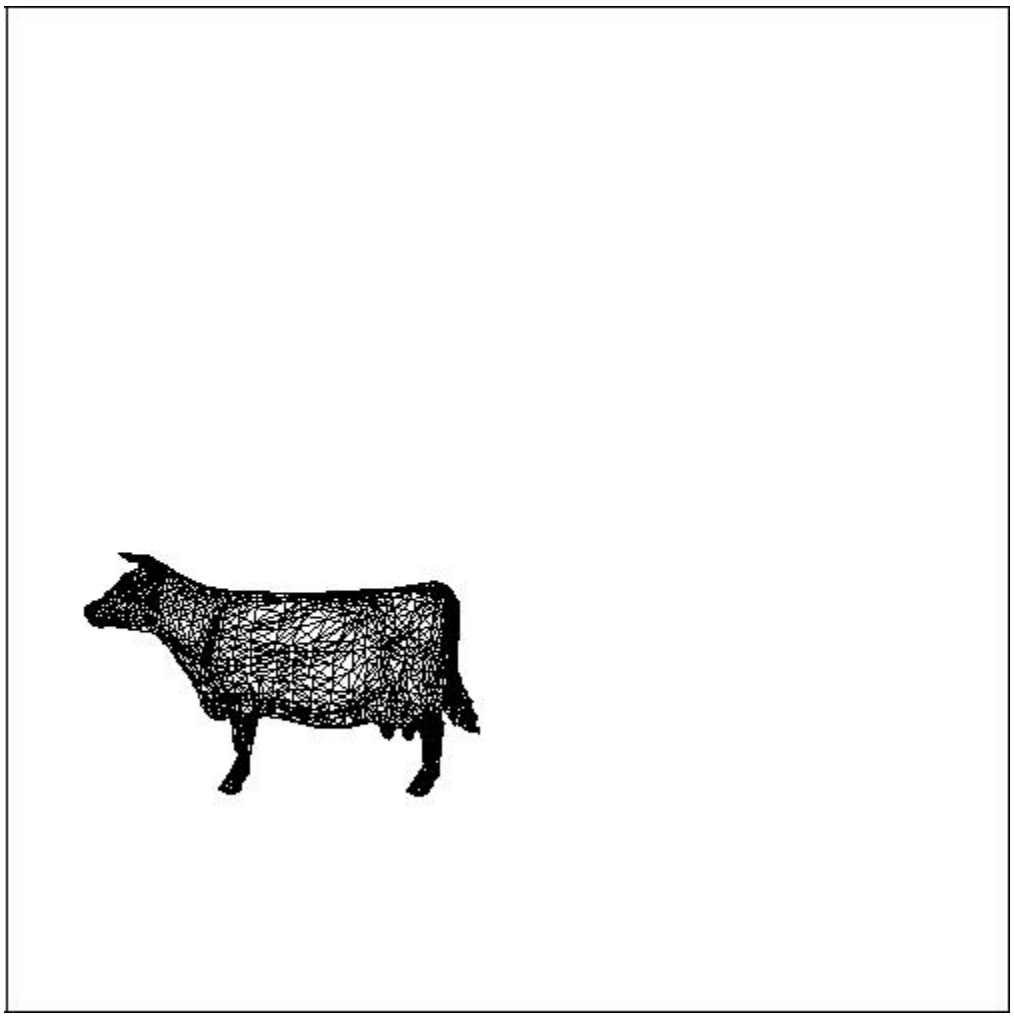
./A6 -f bound-cow.smf >out9.xpm



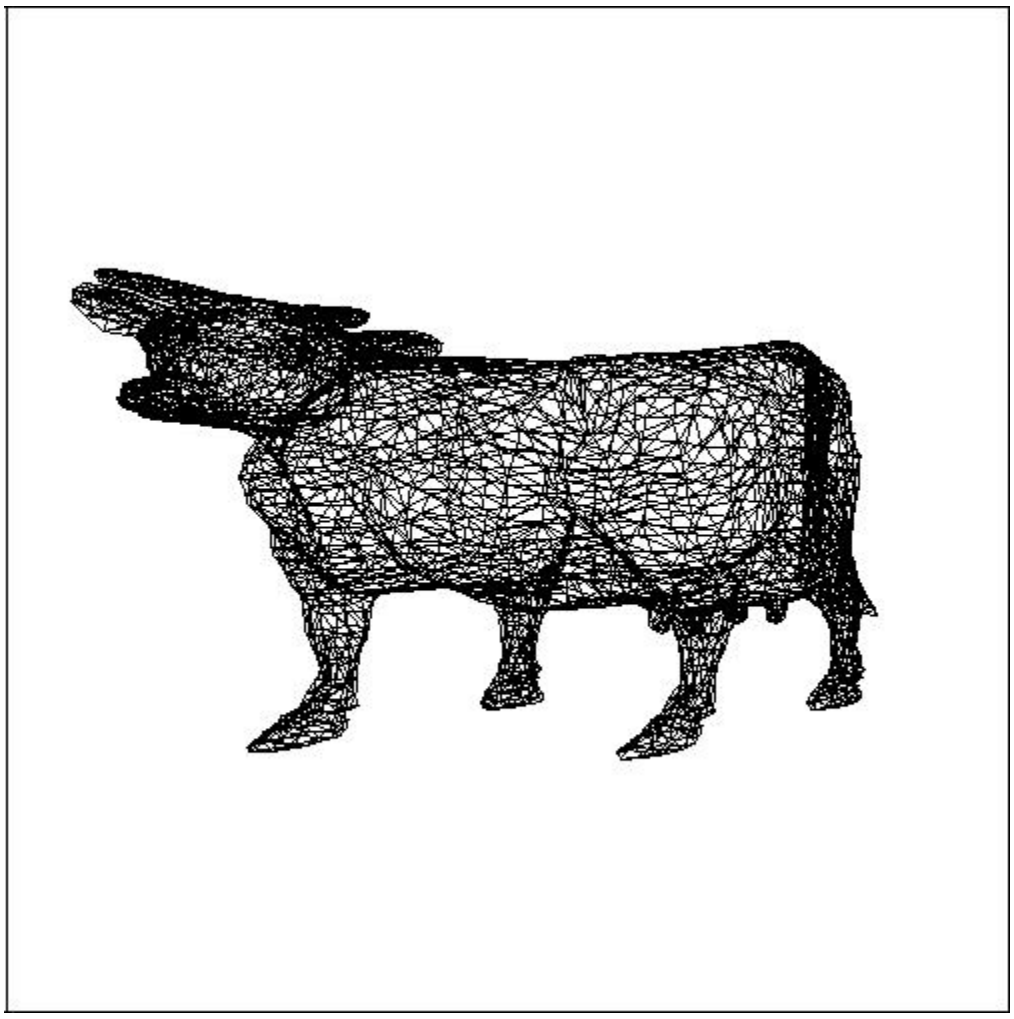
./A6 -f bound-cow.smf -P >out10.xpm



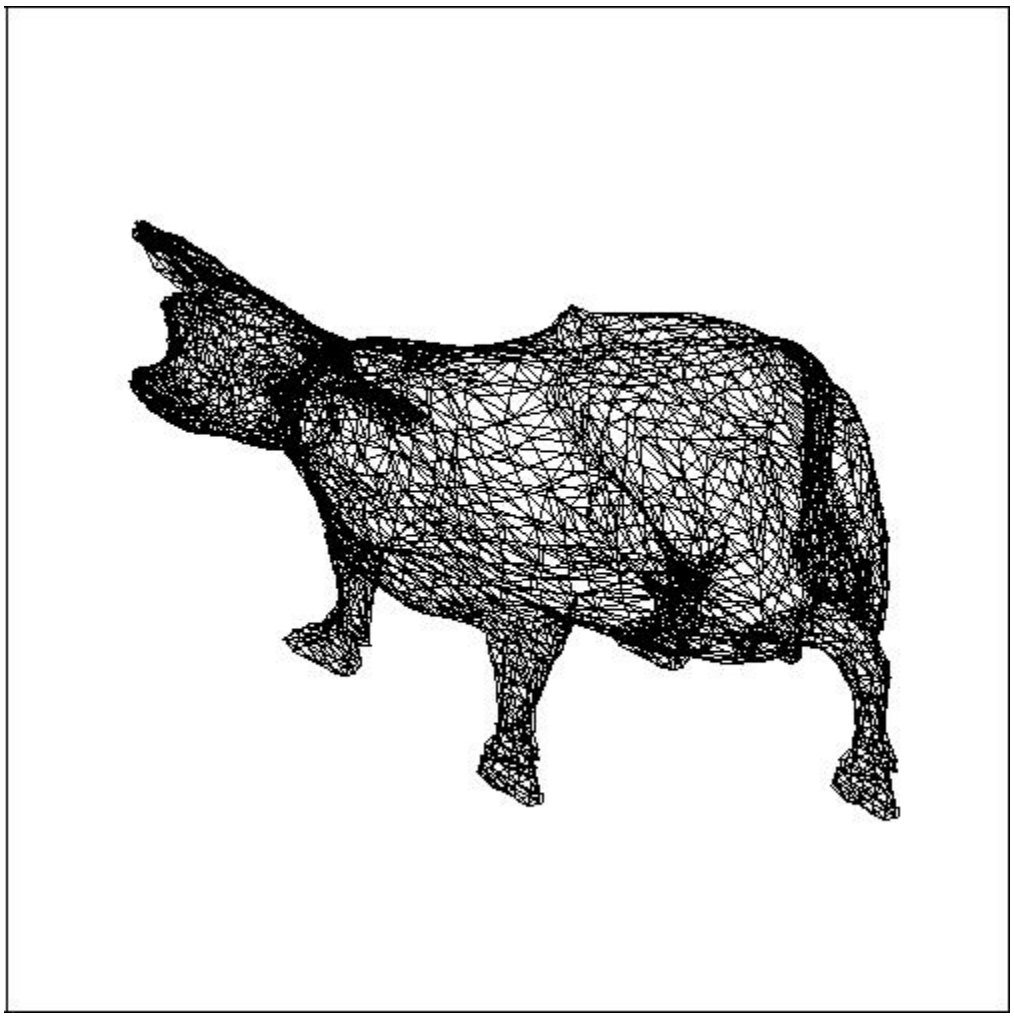
./A6 -f bound-cow.smf -j 0 -k 30 -o 275 -p 305 -P >out11.xpm



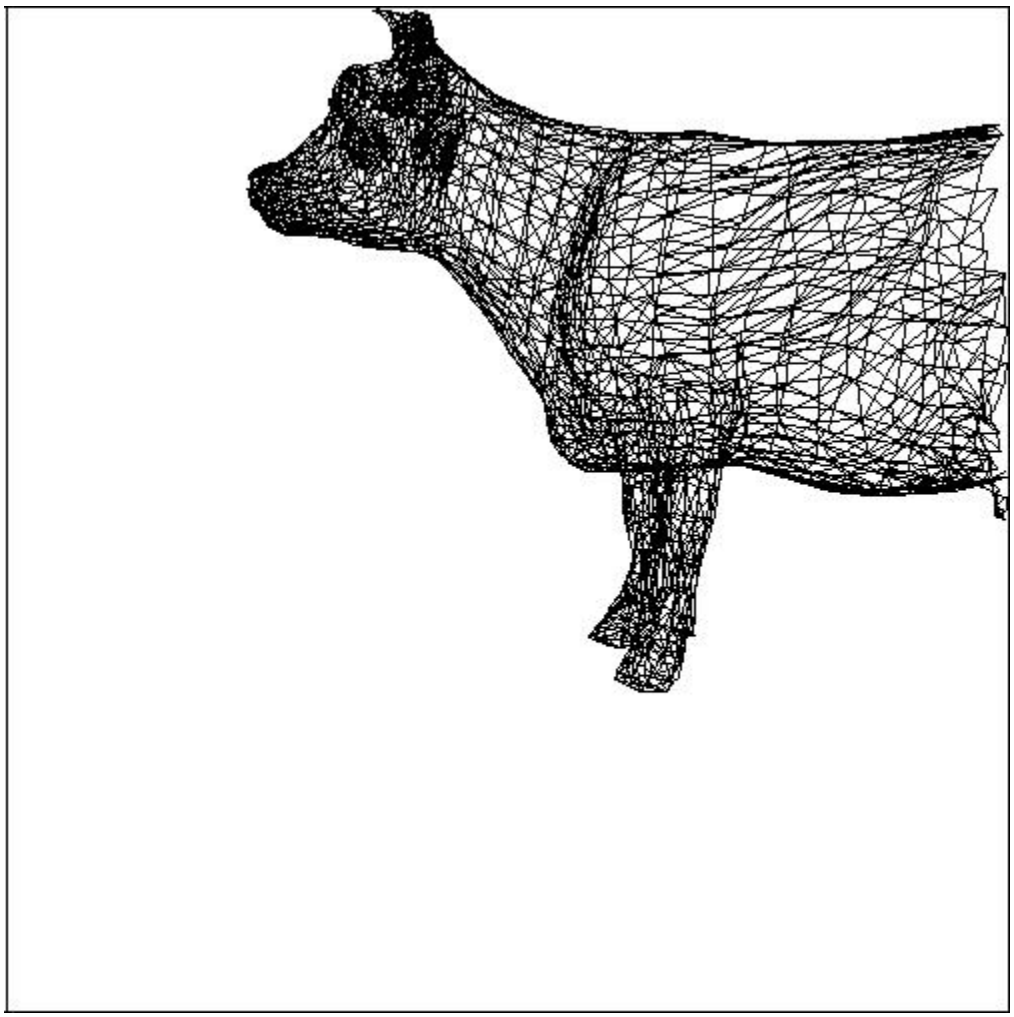
./A6 -f bound-cow.smf -x 1.5
>out12.xpm



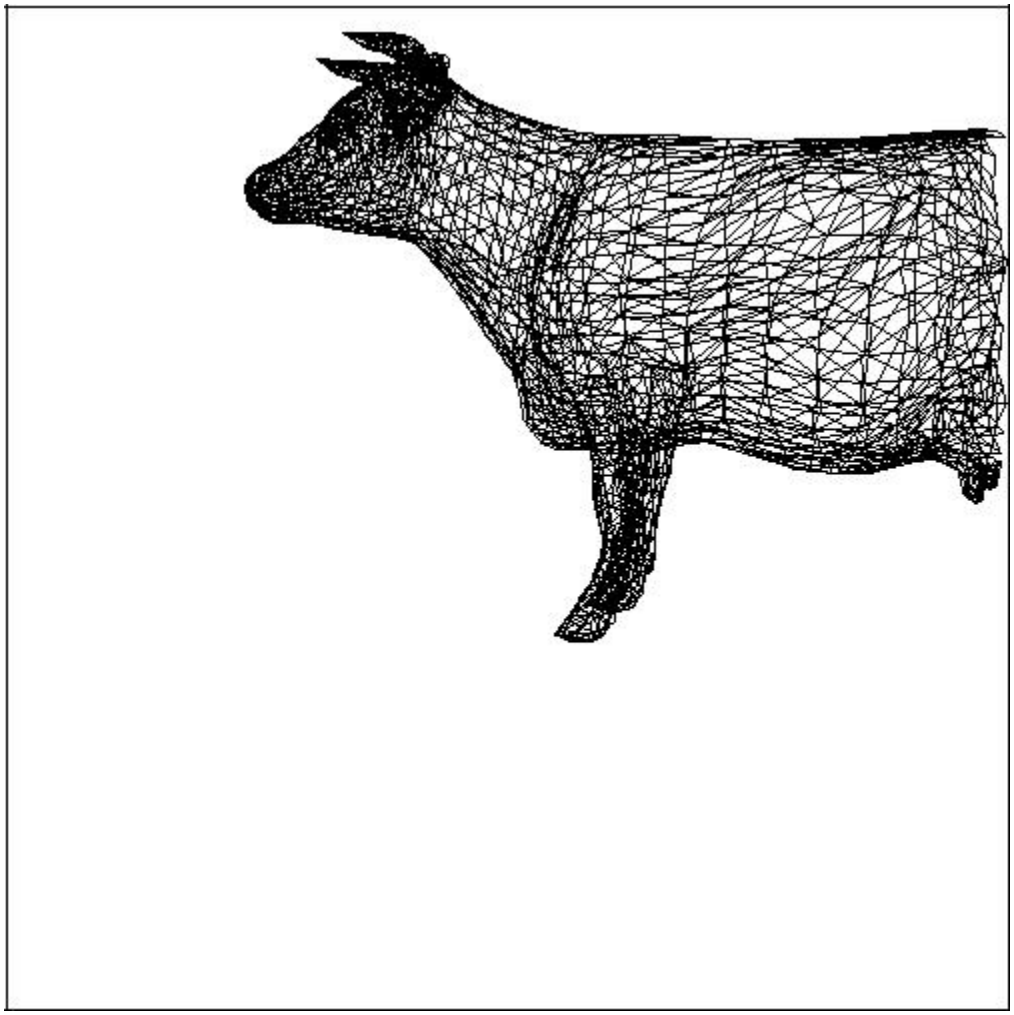
./A6 -f bound-cow.smf -x 4.75 -y -3.25 -z 3.3 -P >out13.xpm



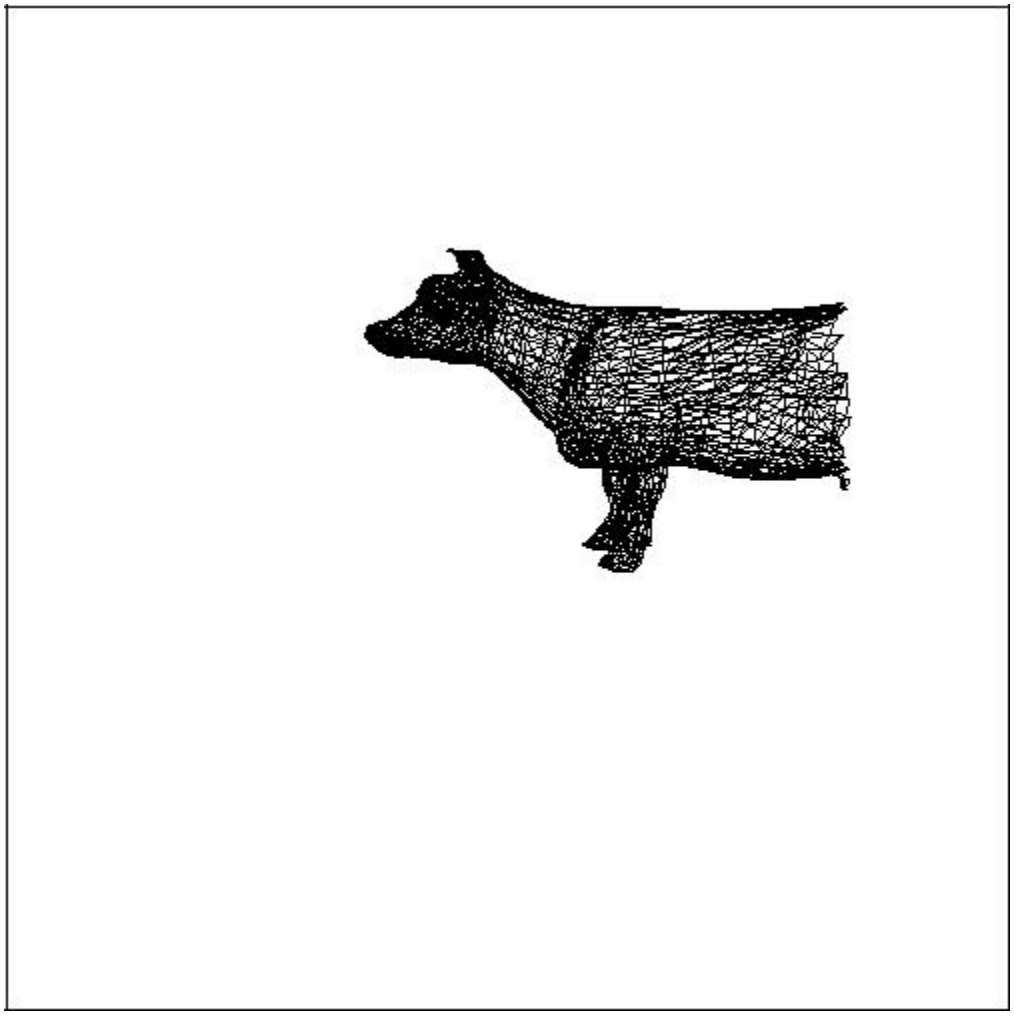
./A6 -f bound-cow.smf -X 0.25 -Y -0.15 -Z 0.3 >out14.xpm



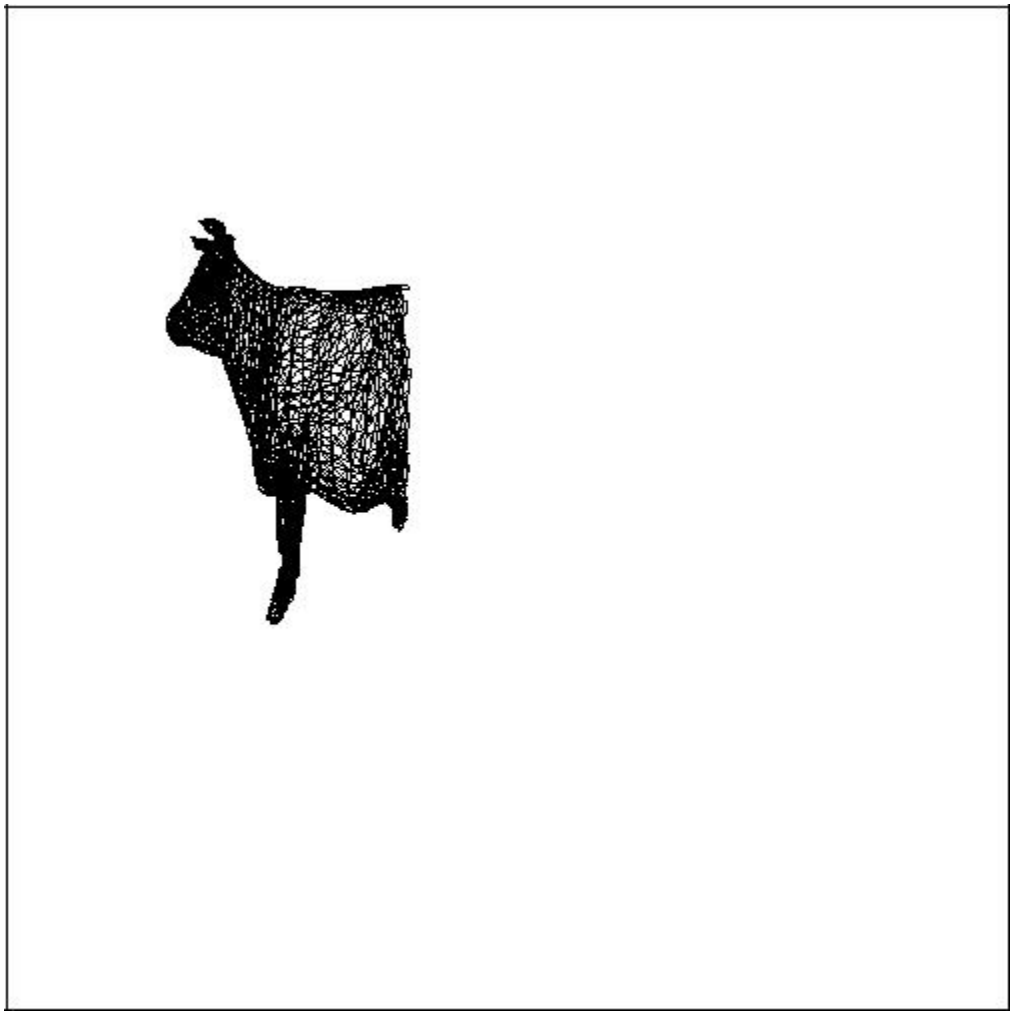
./A6 -f bound-cow.smf -X 0.35 -Y -0.3 -Z 0.3 -u -0.35 -v -0.35 -P >out15.xpm



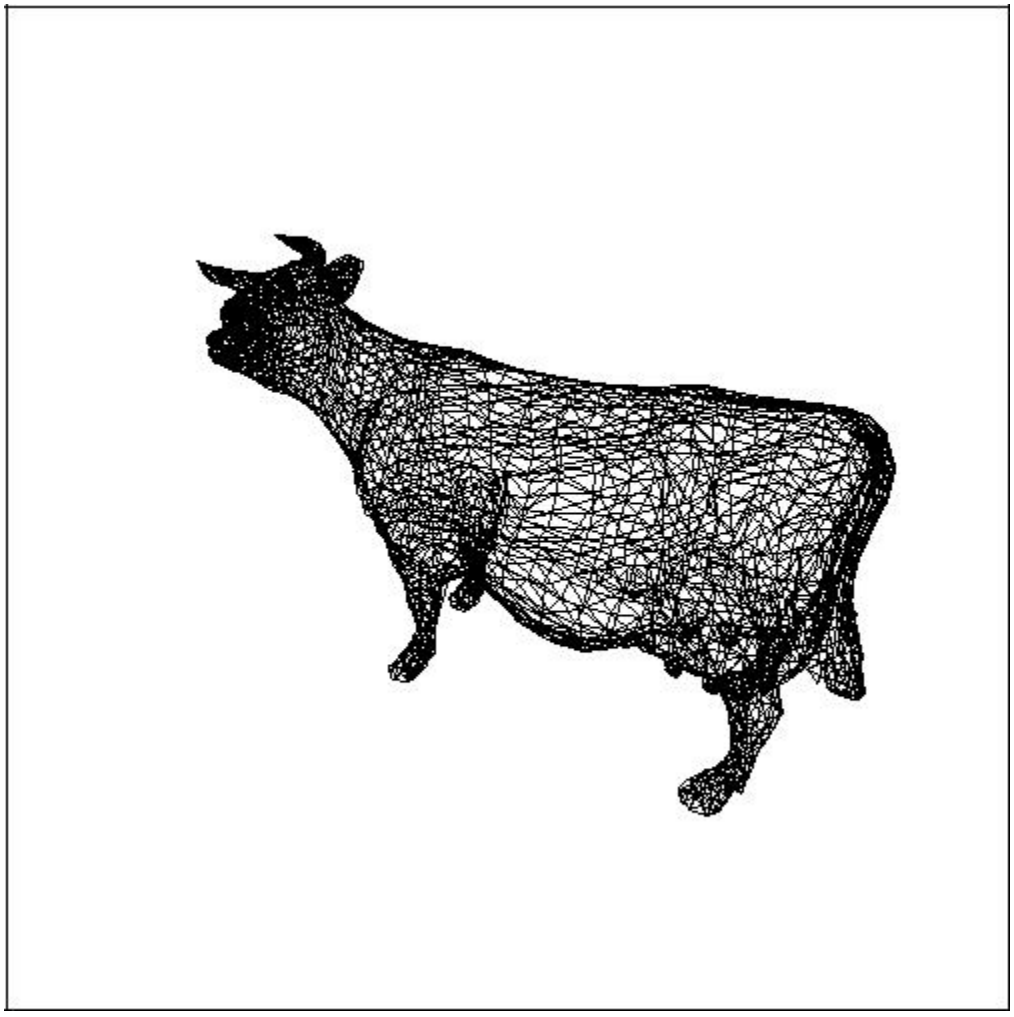
./A6 -f bound-cow.smf -X 0.25 -Y -0.15 -Z 0.3 -j 103 -k 143 -o 421 -p 379 >out16.xpm



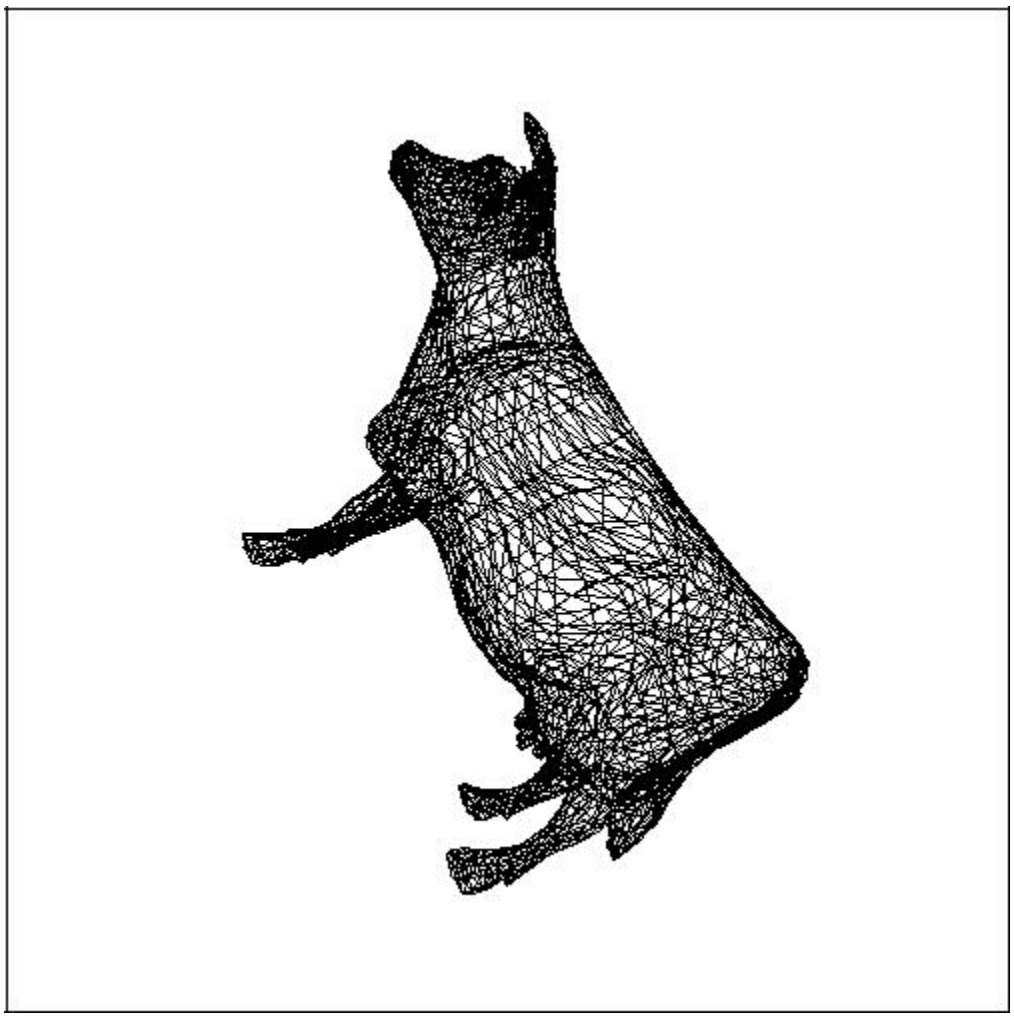
./A6 -f bound-cow.smf -X 0.35 -Y -0.3 -Z 0.3 -u -0.35 -v -0.35 -j 43 -k 71 -o 201 -p 402 -P >out17.xpm



./A6 -f bound-cow.smf -q -1 -r 1.5 -w -2.0 >out18.xpm



./A6 -f bound-cow.smf -Q 1.5 -R 1 -W 0.4 >out19.xpm



./A6 -f bound-cow.smf -u -1.5 -v -0.9 -U 1.2 -V 0.7 >out20.xpm