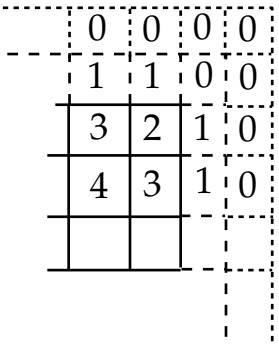
Stenciling

Alg 1:

1. Draw the object 4 times, offsetting in X & Y (screen−space) each time
2. increment the stencil buffer each time

stencil values	meaning
0	no object
1	outline of object
2	silhouette
3	silhouette
4	interior

3. test for cases 2 & 3, draw only these in line mode



Alg 2.

1. offset the polygon
2. disable color buffer
3. draw the objects (fills the depth buffer)
4. disable offset
5. make no further changes to the depth buffer (mask out)
6. toggle stencil bits
7. cull the back faces
8. draw objects in line mode
9. set stencil test = 1, zeroing out the stencil buffer
10. enable color buffers
11. disable the Z buffer
12. draw objects in line mode with stencil test
13. reset state

```
/*
** render silhouette edges using the first algorithm
*/
else if (mode == SILHOUETTE_ALG1) {
        /*
        ** render image translated in 4 directions
        ** counting each time a pixel pass the depth test
        */
        glEnable(GL_CULL_FACE);

        glEnable(GL_STENCIL_TEST);
        glClear(GL_STENCIL_BUFFER_BIT);

        glColorMask(GL_FALSE, GL_FALSE, GL_FALSE, GL_FALSE);
        glDepthMask(GL_FALSE);

        glStencilFunc(GL_ALWAYS, 0, 0xff);
        glStencilOp(GL_KEEP, GL_KEEP, GL_INCR);

        /* (x+1,y) */
        glViewport(1,0,winWidth,winHeight);
        DrawObjects();

        /* (x,y+1) */
        glViewport(0,1,winWidth,winHeight);
        DrawObjects();

        /* (x−1,y) */
        glViewport(−1,0,winWidth,winHeight);
        DrawObjects();

        /* (x−1,y+1) */
        glViewport(0,−1,winWidth,winHeight);
        DrawObjects();

        /* (x,y) */
        glViewport(0,0,winWidth,winHeight);

        glColorMask(GL_TRUE, GL_TRUE, GL_TRUE, GL_TRUE);
        glDepthMask(GL_TRUE);

        glDisable(GL_CULL_FACE);

        /*
        ** color all pixels in the framebuffer with stencil value 1
        */
        glStencilFunc(GL_EQUAL, 2, 0xff);
        glStencilOp(GL_KEEP, GL_KEEP, GL_KEEP);
        glColor3f(1,1,1);
        DrawObjects();

        glStencilFunc(GL_EQUAL, 3, 0xff);
        DrawObjects();

        /*
        ** return state to default values
        */
        glDisable(GL_STENCIL_TEST);

}
```

```
/*
** render silhouette edges using the second algorithm
*/
else if (mode == SILHOUETTE_ALG2) {
        /*
        ** render the offset depth image
        */
        glEnable(GL_POLYGON_OFFSET_FILL);
        glPolygonOffset(1,1);
        glColorMask(GL_FALSE, GL_FALSE, GL_FALSE, GL_FALSE);
        DrawObjects();
        glDisable(GL_POLYGON_OFFSET_FILL);

        /*
        ** make no further changes to the depth image
        */
        glDepthMask(0);

        /*
        ** cull all facets of one (arbitrary) orientation.  render the
        **   remaining facets in outline mode, toggling the stencil bit
        **   at each pixel.
        */
        glEnable(GL_STENCIL_TEST);
        glStencilFunc(GL_ALWAYS, 0, 1);
        glStencilOp(GL_KEEP, GL_KEEP, GL_INVERT);
        glEnable(GL_CULL_FACE);
        glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
        DrawObjects();


        /*
        ** color all pixels in the framebuffer with stencil value 1
        */
        glStencilFunc(GL_EQUAL, 1, 1);
        glStencilOp(GL_ZERO, GL_ZERO, GL_ZERO);
        glColorMask(GL_TRUE, GL_TRUE, GL_TRUE, GL_TRUE);
        glColor3f(1,1,1);
        glDisable(GL_DEPTH_TEST);
        DrawObjects();

        /*
        ** return state to default values
        */
        glDisable(GL_STENCIL_TEST);
        glDisable(GL_CULL_FACE);
        glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
        glDepthMask(GL_TRUE);
        glEnable(GL_DEPTH_TEST); /* XXX */
}
```