

1

Using the Stencil Buffer




Oregon State University

Mike Bailey  
mjb@cs.oregonstate.edu



This work is licensed under a [Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License](#).



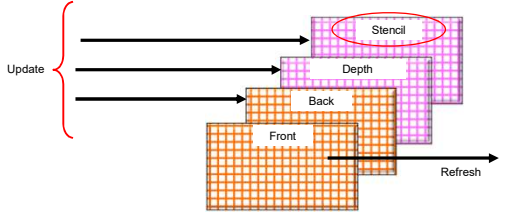
Oregon State University  
Computer Graphics

stencilbuffer.pdf

mjb - August 30, 2022


2

The Framebuffers



Here's what the Stencil Buffer can do for you:

- While drawing into the Back Buffer, you can write values into the Stencil Buffer at the same time.
- While drawing into the Back Buffer, you can do arithmetic on values in the Stencil Buffer at the same time.
- The Stencil Buffer can be used to write-protect certain parts of the Back Buffer.

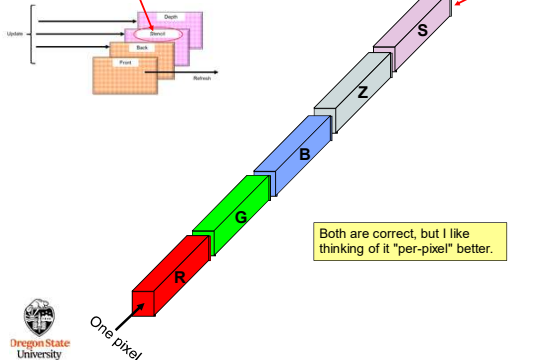



Oregon State University  
Computer Graphics

mjb - August 30, 2022

3

You Can Think of the Stencil Buffer as a Separate Framebuffer, or, You Can Think of it as being Per-Pixel





Oregon State University  
Computer Graphics

mjb - August 30, 2022

4

The Stencil Buffer is Tested Per-Pixel, Very Much Like the Depth Buffer

```
glStencilFunc( func, ref, mask )
```

This specifies the comparison test that is to be done per-pixel.


**func** can be any of GL\_NEVER, GL\_ALWAYS, GL\_EQUAL, GL\_NOTEQUAL, GL\_LESS, GL\_LEQUAL, GL\_GREATER, GL\_GEQUAL

**ref** is an integer reference value that is used to test the pixel's existing stencil value against using the chosen **func**

**mask** is set to 1 in all these examples

The stencil test produces a *true* or *false* value at each pixel where drawing is to be done.

```
if ( ref <func> S_existing is true )
{
    Allow the color write to the existing pixel to take place;
    Modify the pixel's existing stencil value depending on what the glStencilOp says to do;
}
```



Oregon State University  
Computer Graphics

mjb - August 30, 2022

5

This Tells You What to Do with the *true* or *false* Value from the Stencil Test


```
glStencilOp( sfail, zfail, zpass )
```

This specifies how a pixel's stencil value is modified when a fragment passes or fails the stencil test depending on what combinations of *true* and *false* the stencil test and the depth buffer test produce. If the stencil test fails, then *sfail* happens. If the stencil test succeeds, then either *zfail* or *zpass* happen depending on if the depth-buffer test failed or succeeded.

The three values can be any of:

GL_KEEP	Retain the existing stencil value
GL_ZERO	Set the stencil value to zero
GL_REPLACE	Replace the stencil value with <b>ref</b> from the Stencil Func
GL_INCR	Increment the stencil value, with clamping
GL_INCR_WRAP	Increment the stencil value, without clamping
GL_DECR	Decrement the stencil value, with clamping
GL_DECR_WRAP	Decrement the stencil value, without clamping
GL_INVERT	Bitwise toggle the stencil bits: 0's → 1's, 1's → 0's

```
if ( ref <func> S_existing is true )
{
    Allow the color write to the existing pixel to take place;
    Modify the pixel's existing stencil value depending on what the glStencilOp says to do;
}
```



Oregon State University  
Computer Graphics

mjb - August 30, 2022

6


Setting Up the Stencil Buffer

```
// at the top of the program:
const int STENCILBIT = 1;
const int DEFAULT_STENCIL = 0;
const float BIGX = 2;
const float BIGY = BIGX;
const float CLOSEZ = -1;
float Xlens, Ylens;
float Box = 0.40f;

// in InitGraphics():
glutInitDisplayMode( GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH | GLUT_STENCIL );

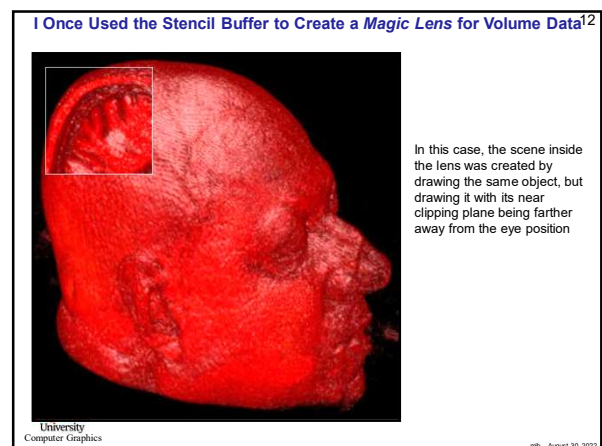
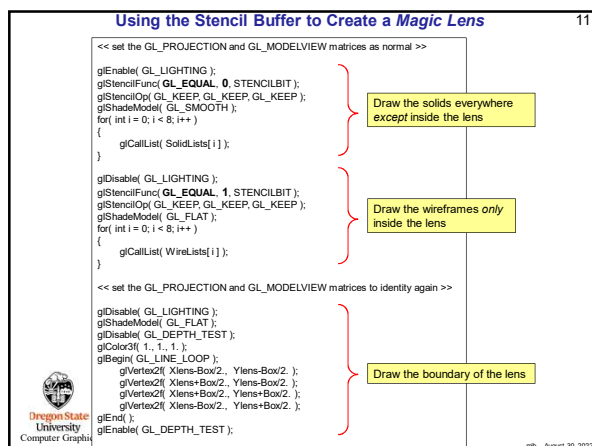
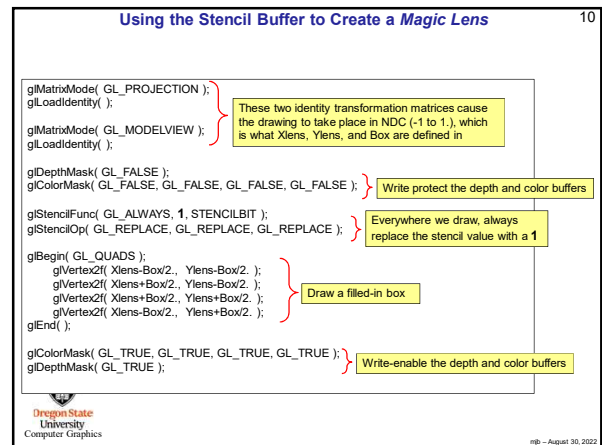
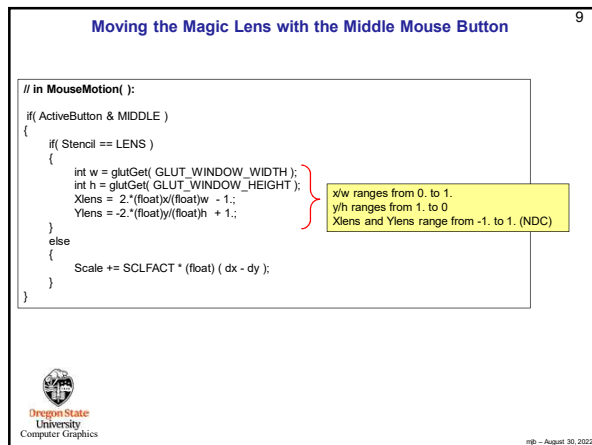
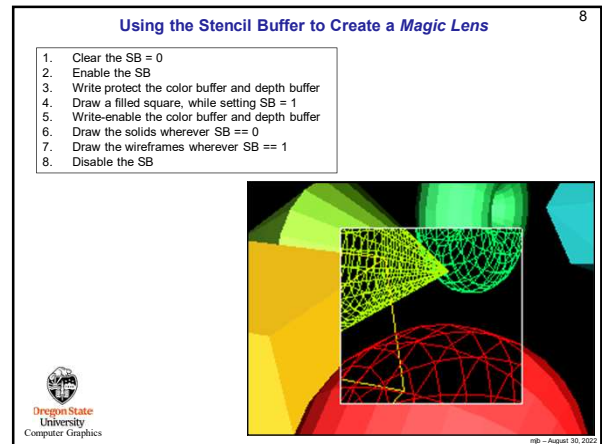
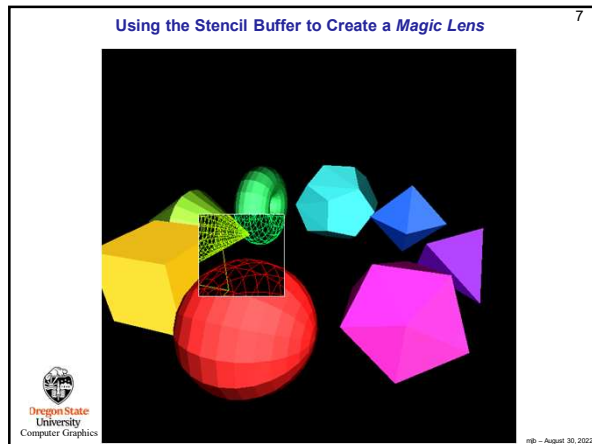
glClearColor( BACKGROUND_COLOR );
glClearStencil( DEFAULT_STENCIL );

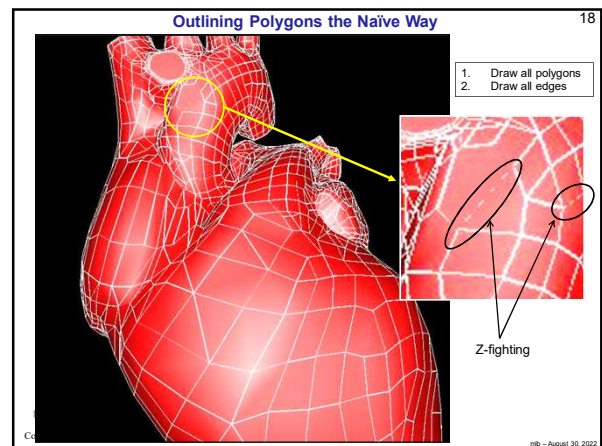
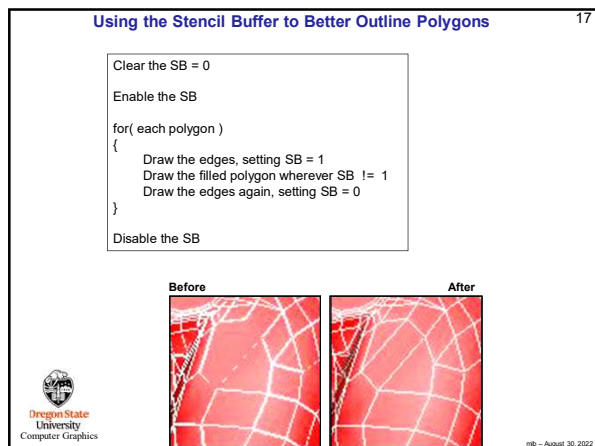
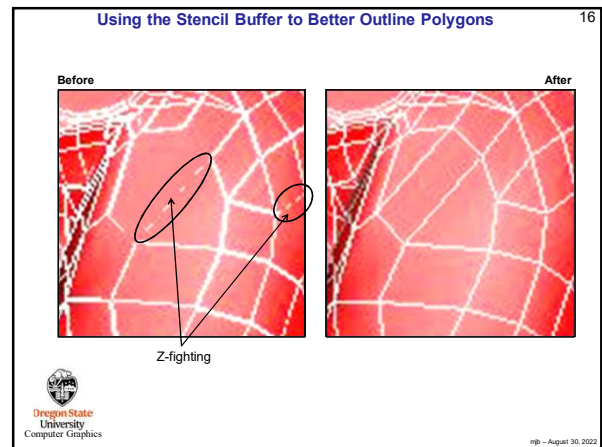
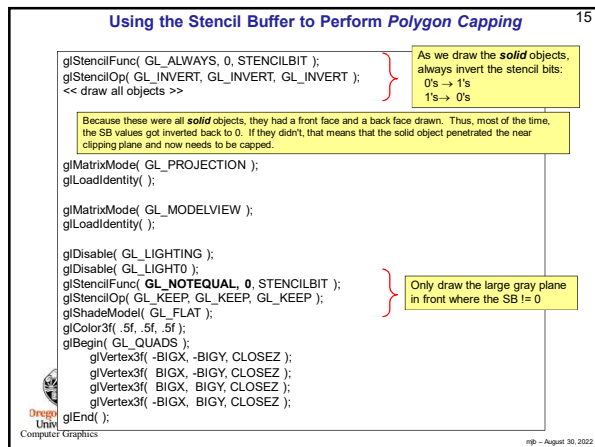
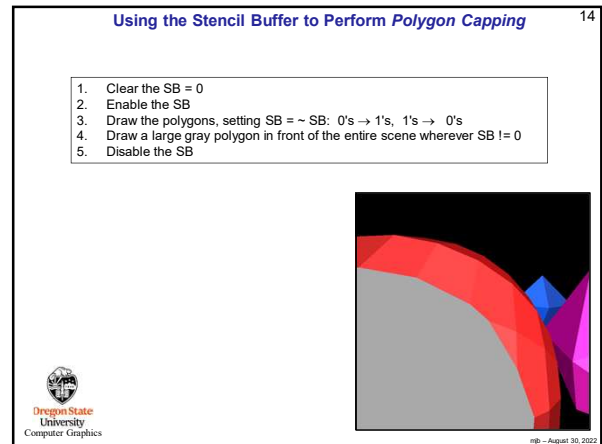
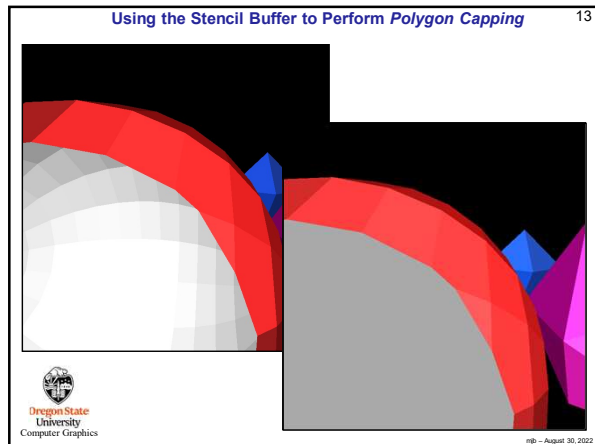
// in Display():
glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT | GL_STENCIL_BUFFER_BIT );
glEnable( GL_STENCIL_TEST );
```

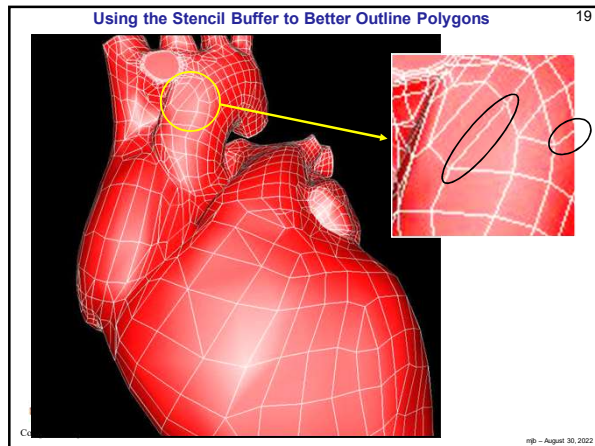


Oregon State University  
Computer Graphics

mjb - August 30, 2022







### Using the Stencil Buffer to Better Outline Polygons

```

for (int f = 0; f < NumFaces; f++)
{
    glStencilFunc(GL_ALWAYS, 1, STENCILBIT);
    glStencilOp(GL_REPLACE, GL_REPLACE, GL_REPLACE);
    glDisable(GL_LIGHTING);
    glStencilMask(GL_FLAT);
    glColor3f(1, 1, 1);
    glBegin(GL_LINE_LOOP);
    for (int v = FirstVertex[f]; v < FirstVertex[f+1]; v++)
    {
        glVertex3f(Vertices[v].x, Vertices[v].y, Vertices[v].z);
    }
    glEnd();

    glStencilFunc(GL_EQUAL, 0, STENCILBIT);
    glStencilOp(GL_KEEP, GL_KEEP, GL_KEEP);
    glEnable(GL_LIGHTING);
    glStencilMask(GL_SMOOTH);
    glMaterialfv(...);
    glBegin(GL_POLYGON);
    for (int v = FirstVertex[f]; v < FirstVertex[f+1]; v++)
    {
        glNormal3f(Normals[v].x, Normals[v].y, Normals[v].z);
        glVertex3f(Vertices[v].x, Vertices[v].y, Vertices[v].z);
    }
    glEnd();

    glStencilFunc(GL_ALWAYS, 0, STENCILBIT);
    glStencilOp(GL_REPLACE, GL_REPLACE, GL_REPLACE);
    glDisable(GL_LIGHTING);
    glStencilMask(GL_FLAT);
    glColor3f(1, 1, 1);
    glBegin(GL_LINE_LOOP);
    for (int v = FirstVertex[f]; v < FirstVertex[f+1]; v++)
    {
        glVertex3f(Vertices[v].x, Vertices[v].y, Vertices[v].z);
    }
    glEnd();
}

```

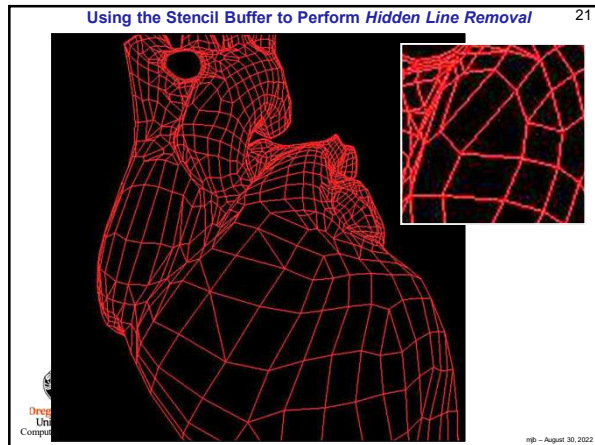
Put "masking tape" down on the polygon edges

Paint the polygon, which also paints the edges

Pull the "masking tape" up and paint just the polygon edges

Oregon State University Computer Graphics

mpb - August 30, 2022



### Using the Stencil Buffer to Perform Hidden Line Removal

```

Clear the SB = 0
Enable the SB

for( each polygon )
{
    Draw the edges, setting SB = 1
    Draw the polygon, unlit and flat shaded, in the background color wherever SB != 1
    Draw the edges again, setting SB = 0
}

Disable the SB

```

Oregon State University Computer Graphics

mpb - August 30, 2022

