



1


## Using the Stencil Buffer



Oregon State University

Mike Bailey  
mjb@cs.oregonstate.edu

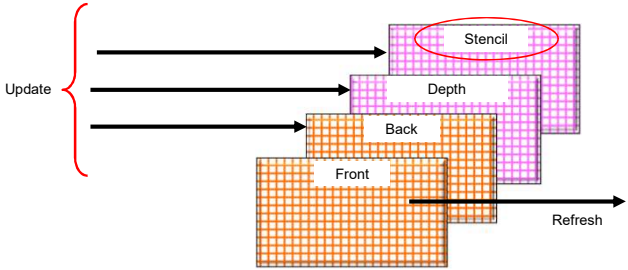
 This work is licensed under a [Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License](https://creativecommons.org/licenses/by-nc-nd/4.0/)

 Oregon State University Computer Graphics

stencilbuffer.pptx mjb - August 30, 2022


2

## The Framebuffers



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

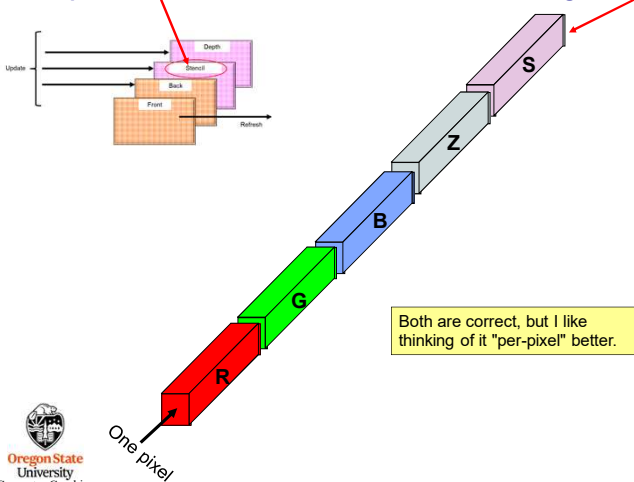
1. While drawing into the Back Buffer, you can write values into the Stencil Buffer at the same time.
2. While drawing into the Back Buffer, you can do arithmetic on values in the Stencil Buffer at the same time.
3. The Stencil Buffer can be used to write-protect certain parts of the Back Buffer.

 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



Both are correct, but I like thinking of it "per-pixel" better.

 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> Sexisting 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> Sexisting 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;
}
```



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 );

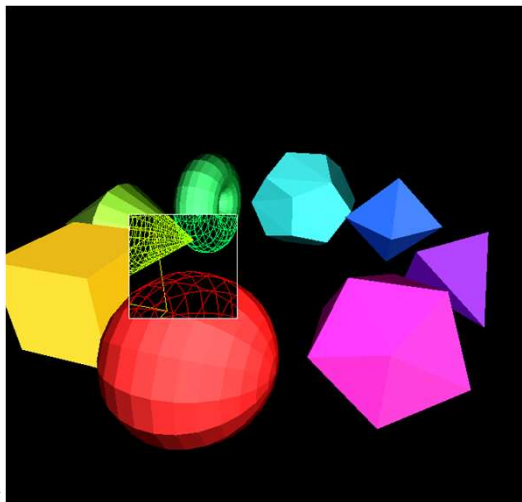
glClearColor( BACKGROUND_COLOR );
glClearStencil( DEFAULT_STENCIL );

// in Display( ):
...
glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT | GL_STENCIL_BUFFER_BIT );
...
glEnable( GL_STENCIL_TEST );
...
University
Computer Graphics
```

6

mjb - August 30, 2022

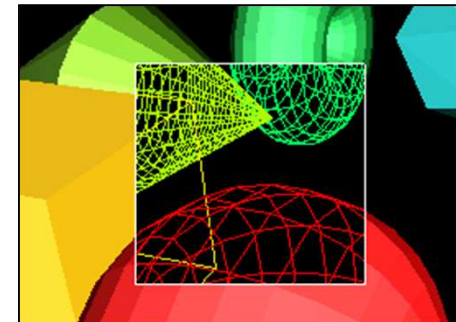
## 7 Using the Stencil Buffer to Create a Magic Lens



mjb - August 30, 2022

## 8 Using the Stencil Buffer to Create a Magic Lens

1. Clear the SB = 0
2. Enable the SB
3. Write protect the color buffer and depth buffer
4. Draw a filled square, while setting SB = 1
5. Write-enable the color buffer and depth buffer
6. Draw the solids wherever SB == 0
7. Draw the wireframes wherever SB == 1
8. Disable the SB



mjb - August 30, 2022

## Moving the Magic Lens with the Middle Mouse Button

9

```
// in MouseMotion( ):
if( ActiveButton & MIDDLE )
{
    if( Stencil == LENS )
    {
        int w = glutGet( GLUT_WINDOW_WIDTH );
        int h = glutGet( GLUT_WINDOW_HEIGHT );
        Xlens = 2.*(float)x/(float)w - 1.;
        Ylens = -2.*(float)y/(float)h + 1.;
    }
    else
    {
        Scale += SCLFACT * (float) ( dx - dy );
    }
}
```

x/w ranges from 0. to 1.  
y/h ranges from 1. to 0  
Xlens and Ylens range from -1. to 1. (NDC)



mjb - August 30, 2022

## Using the Stencil Buffer to Create a Magic Lens

10

```
glMatrixMode( GL_PROJECTION );
glLoadIdentity( );

glMatrixMode( GL_MODELVIEW );
glLoadIdentity( );

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

glStencilFunc( GL_ALWAYS, 1, STENCILBIT );
glStencilOp( GL_REPLACE, GL_REPLACE, GL_REPLACE );

glBegin( GL_QUADS );
glVertex2f( Xlens-Box/2., Ylens-Box/2. );
glVertex2f( Xlens+Box/2., Ylens-Box/2. );
glVertex2f( Xlens+Box/2., Ylens+Box/2. );
glVertex2f( Xlens-Box/2., Ylens+Box/2. );
glEnd( );

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

These two identity transformation matrices cause the drawing to take place in NDC (-1 to 1.), which is what Xlens, Ylens, and Box are defined in

Write protect the depth and color buffers

Everywhere we draw, always replace the stencil value with a 1

Draw a filled-in box

Write-enable the depth and color buffers



mjb - August 30, 2022

## Using the Stencil Buffer to Create a Magic Lens

11

```
<< set the GL_PROJECTION and GL_MODELVIEW matrices as normal >>

glEnable( GL_LIGHTING );
glStencilFunc( GL_EQUAL, 0, STENCILBIT );
glStencilOp( GL_KEEP, GL_KEEP, GL_KEEP );
glShadeModel( GL_SMOOTH );
for( int i = 0; i < 8; i++ )
{
    glCallList( SolidLists[ i ] );
}

glDisable( GL_LIGHTING );
glStencilFunc( GL_EQUAL, 1, STENCILBIT );
glStencilOp( GL_KEEP, GL_KEEP, GL_KEEP );
glShadeModel( GL_FLAT );
for( int i = 0; i < 8; i++ )
{
    glCallList( WireLists[ i ] );
}

<< set the GL_PROJECTION and GL_MODELVIEW matrices to identity again >>

glDisable( GL_LIGHTING );
glShadeModel( GL_FLAT );
glDisable( GL_DEPTH_TEST );
glColor3f( 1., 1., 1. );
glBegin( GL_LINE_LOOP );
glVertex2f( Xlens-Box/2., Ylens-Box/2. );
glVertex2f( Xlens+Box/2., Ylens-Box/2. );
glVertex2f( Xlens+Box/2., Ylens+Box/2. );
glVertex2f( Xlens-Box/2., Ylens+Box/2. );
glEnd( );
glEnable( GL_DEPTH_TEST );
```

Draw the solids everywhere except inside the lens

Draw the wireframes only inside the lens

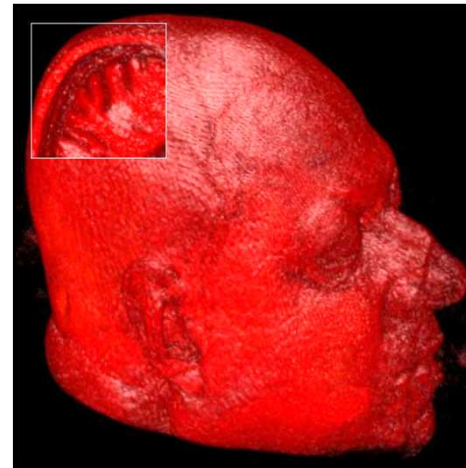
Draw the boundary of the lens



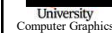
mjb - August 30, 2022

## I Once Used the Stencil Buffer to Create a Magic Lens for Volume Data

12



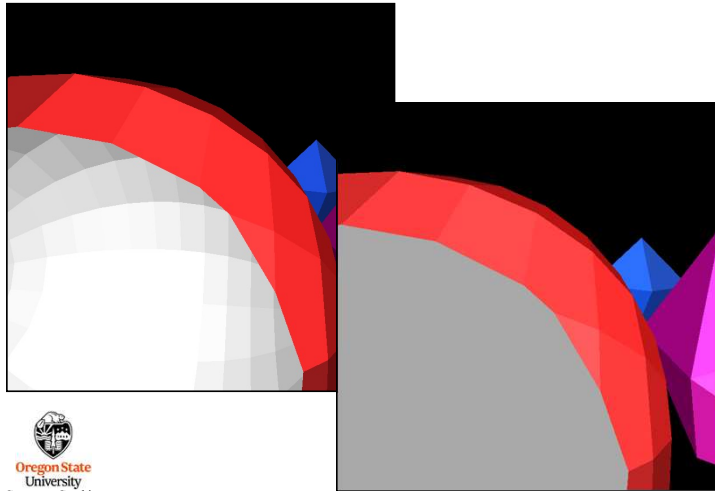
In this case, the scene inside the lens was created by drawing the same object, but drawing it with its near clipping plane being farther away from the eye position



mjb - August 30, 2022

### Using the Stencil Buffer to Perform *Polygon Capping*

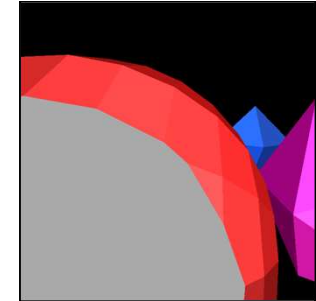
13



### Using the Stencil Buffer to Perform *Polygon Capping*

14

1. Clear the SB = 0
2. Enable the SB
3. Draw the polygons, setting SB = ~ SB: 0's → 1's, 1's → 0's
4. Draw a large gray polygon in front of the entire scene wherever SB != 0
5. Disable the SB



### Using the Stencil Buffer to Perform *Polygon Capping*

15

```
glStencilFunc( GL_ALWAYS, 0, STENCILBIT );
glStencilOp( GL_INVERT, GL_INVERT, GL_INVERT );
<< draw all objects >>
```

As we draw the **solid** objects,  
always invert the stencil bits:  
0's → 1's  
1's → 0's

Because these were all **solid** objects, they had a front face and a back face drawn. Thus, most of the time, the SB values got inverted back to 0. If they didn't, that means that the solid object penetrated the near clipping plane and now needs to be capped.

```
glMatrixMode( GL_PROJECTION );
glLoadIdentity( );
```

```
glMatrixMode( GL_MODELVIEW );
glLoadIdentity( );
```

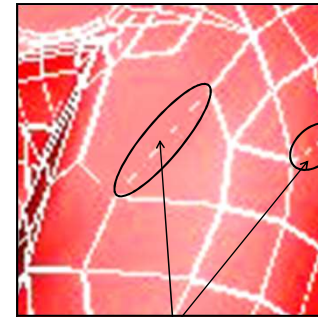
```
glDisable( GL_LIGHTING );
glDisable( GL_LIGHT0 );
glStencilFunc( GL_NOTEQUAL, 0, STENCILBIT );
glStencilOp( GL_KEEP, GL_KEEP, GL_KEEP );
glShadeModel( GL_FLAT );
glColor3f( .5f, .5f, .5f );
glBegin( GL_QUADS );
glVertex3f( -BIGX, -BIGY, CLOSEZ );
glVertex3f( BIGX, -BIGY, CLOSEZ );
glVertex3f( BIGX, BIGY, CLOSEZ );
glVertex3f( -BIGX, BIGY, CLOSEZ );
glEnd( );
```

Only draw the large gray plane  
in front where the SB != 0

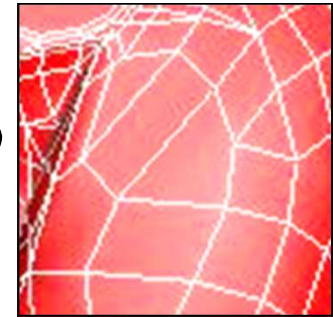
### Using the Stencil Buffer to Better Outline Polygons

16

Before



After



Z-fighting

## Using the Stencil Buffer to Better Outline Polygons

17

```

Clear the SB = 0
Enable the SB
for( each polygon )
{
    Draw the edges, setting SB = 1
    Draw the filled polygon wherever SB != 1
    Draw the edges again, setting SB = 0
}
Disable the SB
    
```

Before



After

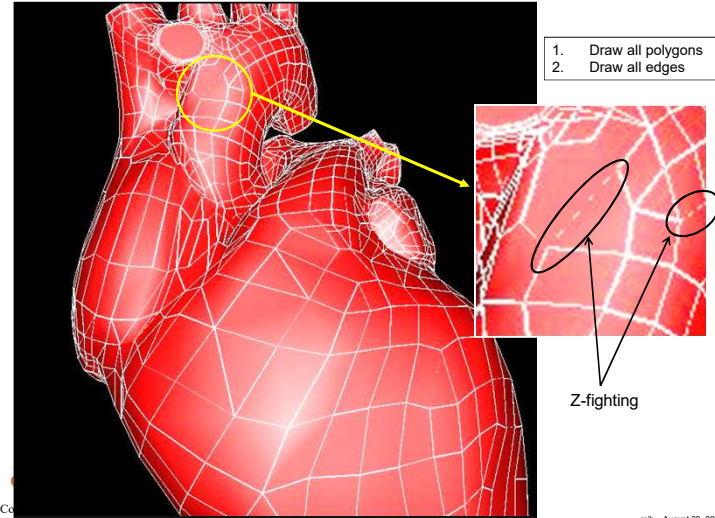


mjb - August 30, 2022

## Outlining Polygons the Naïve Way

18

1. Draw all polygons
2. Draw all edges



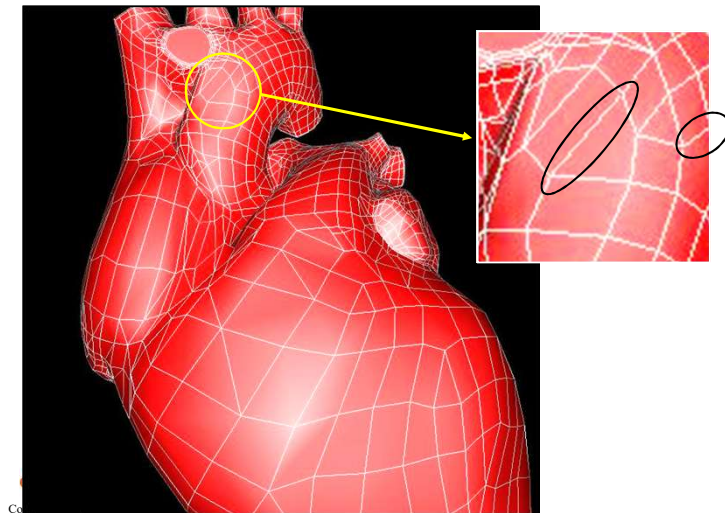
Z-fighting

Co

mjb - August 30, 2022

## Using the Stencil Buffer to Better Outline Polygons

19



Co

mjb - August 30, 2022

## Using the Stencil Buffer to Better Outline Polygons

20

```

for( int f = 0; f < NumFaces; f++ )
{
    glStencilFunc( GL_ALWAYS, 1, STENCILBIT );
    glStencilOp( GL_REPLACE, GL_REPLACE, GL_REPLACE );
    glDisable( GL_LIGHTING );
    glShadeModel( 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 );
    glShadeModel( 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 );
    glShadeModel( 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



mjb - August 30, 2022

mjb - August 30, 2022



