Buffers

Define a buffer by its spatial resolution \((n \times m)\) and its depth (or precision) \(k\), the number of bits/pixel.

Depth Buffering and Hidden Surface Removal

Depth Buffering (already using it)

1. Hint for depth buffer resolution
   \[
   \text{void}\ \text{glfwWindowHint}(\text{GLFW_DEPTH_BITS, 16});
   \]
2. Enable depth buffering
   \[
   \text{glEnable( GL_DEPTH_TEST );}
   \]
3. Clear color and depth buffers
   \[
   \text{glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );}
   \]
4. Render scene
5. Swap color buffers

Other Buffers
Other Buffers

Using Framebuffers

- clearing buffers
  - clearing individual buffer is expensive
  - Use glClear with bitwise-ORed masks to clear multiple buffers
- selecting color buffers for writing/clearing
  - glBindFramebuffer: useful in FBO (framebuffer object)

Masking Buffers

- Before OpenGL writes data into the enabled color, depth, or stencil buffers, a masking operation is applied to the data, as specified with one of the following commands.
  - A bitwise logical AND is performed with each mask and the corresponding data to be written

Masking Buffers (cont)

- void glColorMask(GLboolean red, GLboolean green, GLboolean blue, GLboolean alpha);
- void glDepthMask(GLboolean flag);
- void glStencilMask(GLuint mask);
  - If a 1 appears in mask, the corresponding bit in the stencil buffer is written; where a 0 appears, the bit is not written.
  - The default values of all the GLboolean masks are GL_TRUE, and the default values for the two GLuint masks are all 1's

Accumulation Buffer

- Gone after OpenGL 3.1 (deprecated)
- Can use FBO for multi-pass rendering with an appropriate fragment program
- Useful for several effects
  - Basically, same functions can be done with multi-pass rendering.
  - Initially, it was the floating-point buffer but now all buffers can be floating-point!
Accessing Accumulation Buffer

\[ \text{glAccum}( \text{op}, \text{value} ) \]
- operations
  - within the accumulation buffer: \text{GL}_\text{ADD}, \text{GL}_\text{MULT}
  - from read buffer: \text{GL}_\text{ACCUM}, \text{GL}_\text{LOAD}
  - transfer back to write buffer: \text{GL}_\text{RETURN}
- \text{glAccum}(\text{GL}_\text{ACCUM}, 0.5) multiplies each value in write buffer by 0.5 and adds to accumulation buffer

Accumulation Buffer Applications
- Compositing
- Full Scene Antialiasing
- Depth of Field
- Filtering
- Motion Blur

Full Scene Antialiasing: Jittering the View
- Each time we move the viewer, the image shifts
  - Different aliasing artifacts in each image
  - Averaging images using accumulation buffer averages out these artifacts
- Replaced with
  - \text{GL}_\text{MULTISAMPLE}

Depth of Focus: Keeping a Plane in Focus
- Jitter the viewer to keep one plane unchanged

Depth of Field

Motion Blur

Full Scene Antialiasing:
- Jittering the View
  - Each time we move the viewer, the image shifts
    - Different aliasing artifacts in each image
    - Averaging images using accumulation buffer averages out these artifacts
  - Replaced with
    - \text{GL}_\text{MULTISAMPLE}

Depth of Focus: Keeping a Plane in Focus
- Jitter the viewer to keep one plane unchanged
Motion Blur w/o Accum.Buffer
Details:
- Render previous frame as background with an
  render current scene
- Save result as next background
  [thus image combining previous frame]

Fragment Operations
- Scissor Test
- Alpha Test
- Stencil Test
- Depth Test
- Blending
- Dithering
- Logical Operations

Fragment Operations
Scissor Box
- Additional Clipping Test
  `glScissor( x, y, w, h )`
  - any fragments outside of box are clipped
  - useful for updating a small section of a viewport
  - affects `glClear()` operations

Scissor test
- Alpha Test (deprecated)
  `glAlphaFunc( func, value )`
  `glEnable( GL_ALPHA_TEST )`
  - use alpha as a mask in textures

Alpha Test (deprecated)
  - Reject pixels based on their alpha value
  - `glAlphaFunc( func, value )`
  - `glEnable( GL_ALPHA_TEST )`
  Just use a fragment program!
Stencil Buffer

- Used to control drawing based on values in the stencil buffer
  - Fragments that fail the stencil test are not drawn
  - Example: create a mask in stencil buffer and draw only objects not in mask area

Stenciling

- Fragments that fail the stencil test are not drawn
- Example: create a mask in stencil buffer and draw only objects not in mask area

Mimicking Stencil

- Compose stencil template
- Control template then render
- Multi-pass rendering

Controlling Stencil Buffer

- glStencilFunc(func, ref, mask)
  - Compare value in buffer with ref using func
  - Only applied for bits in mask which are 1
  - Func is one of standard comparison functions
- glStencilOp(fail, zfail, zpass)
  - Allows changes in stencil buffer based on passing or failing stencil and depth tests: GL_KEEP, GL_INCR
- glStencilFuncSeparate(face, ref, mask)
- glStencilOpSeparate(face, fail, zfail, zpass)

### glStencilFunc(func, ref, mask)

- `never` always
- `always` never
- `<=` always
- `>=` never
- `=` never
- `!=` always

- Compare value in stencil buffer with ref using func
- Bit-wise mask for comparison

### glStencilOp(fail, zfail, zpass)

- Keep
- Zero
- Replace
- Incr (_WRAP)
- Decr (_WRAP)
- Invert

- Update stencil buffer with 'fail' op
- Update stencil buffer with 'zfail' op
- Update stencil buffer with 'zpass' op
- Update stencil buffer with 'fail' op and discard fragment
- Update stencil buffer with 'zfail' op and discard fragment
- Update stencil buffer with 'zpass' op and discard fragment
- Fragment → blending
How to set the stencil?

Creating a Mask

```gl
glfwWindowHint(GLFW_STENCIL_BITS, 16);
glEnable( GL_STENCIL_TEST );
glClearStencil( 0x0 );

glStencilFunc( GL_ALWAYS, 0x1, 0x1 );
glStencilOp( GL_REPLACE, GL_REPLACE, GL_REPLACE );
```

• draw mask

Using Stencil Mask

```gl
glStencilFunc( GL_EQUAL, 0x1, 0x1 )

• draw objects where stencil = 1

glStencilFunc( GL_NOT_EQUAL, 0x1, 0x1 );
glStencilOp( GL_KEEP, GL_KEEP, GL_KEEP );

• draw objects where stencil != 1
```

Example: Room w/ Window

Room with a view

1. Turn off color buffer
2. Turn off depth buffer updates
3. Turn on stencil buffer
4. Setup the stencil test
5. Draw the window
6. Sets up the stencil test for background
7. Turn on the color buffer
8. Turn on the depth buffer
9. Draw the background
10. Setup test for the wall
11. Draw the wall
12. Reset state
13. Draw any interior

Room with a view

1. glColorMask(F,F,F,F)
2. glDepthMask(F)
3. glEnable(stencil-test)
4. glStencilFunc(A,0x01,0x01) glStencilOp(K,K,R)
5. Draw the window
6. glStencilFunc(=,0x01,0x01) glStencilOp(k,k,k)
7. glColorMask(T,T,T,T)
8. glDepthMask(T)
9. Draw background
10. glStencilFunc(!=,0x01,0x01)
11. Draw wall
12. glDisable(stencil-test)
13. Draw anything else
Decal

Bad way to resolve z-fighting

Decaling w/ Depth Buffer (Painter's Alg)
1. Disable depth buffer updates
2. Draw the base polygon
3. Draw the decal polygons
4. Disable color buffer updates
5. Enable depth buffer updates
6. Draw base polygon
7. Reset state (enable color buffers)

Decaling w/ Depth Buffer (Painter's Alg)

```c
glEnable(GL_DEPTH_TEST)
glDepthMask(GL_FALSE)
```

```c
1. Disable depth buffer updates
2. Draw the base polygon
3. Draw the decal polygons
4. Disable color buffer updates
5. Enable depth buffer updates
6. Draw base polygon
7. Reset state (enable color buffers)
```

Decaling w/ stencil buffer

A. Create a mask in the stencil buffer which defines the decal region
B. Use this mask in 2 passes:
   base polygon
deal polygon(s)

Stenciling

- Steps to draw 2 coplanar rectangles:
  1. Make the stencil for yellow one first (by drawing the green polygon)
  2. Draw the yellow one with the stencil
  3. Draw the green one

Stenciling (cont)

```c
gEnable(GL_STENCIL_TEST);
gStencilFunc(GL_EQUAL, GL_NOTEQUAL, 0)
gDepthMask(GL_FALSE, GL_TRUE);
```

```c
1. Disable depth buffer updates
2. Draw the base polygon
3. Draw the decal polygons
4. Disable color buffer updates
5. Enable depth buffer updates
6. Draw base polygon
7. Reset state (enable color buffers)
```

```c
// draw green rectangle, to the area of GREEN filled with 1
// ready to write to color buffer
// draw yellow rectangle, to the area of YELLOW filled with 1
```
Decaling w/ stencil buffer

1. Enable stenciling
2. Set test to always pass
   w/ref=1, mask=1
3. Set stencil op
   1: if depth passes
   0: if depth fails
4. Draw the base polygon
5. Set stencil function to pass
6. Disable writes to the stencil buf
7. Turn off depth buffering
8. Render the decal polygon
9. Reset state

Decaling w/ stencil buffer

gEnable(GL_Stencil_Test)
gStencilFunc(GL_ALWAYS,1,1)
gStencilOp(GL_KEEP, GL_ZERO, GL_REPLACE)
gStencilFunc(GL_EQUAL,1,1)
gStencilOp(GL_KEEP, GL_KEEP, GL_KEEP)
gDisable(GL_DEPTH_TEST)
gDisable(GL_STENCIL_TEST)
gEnable(GL_DEPTH_TEST)

Decal

How to resolve z-fighting

Decaling

[0] Base: (-5,0)→(5,6)
[1] R: (-3,1)→(2,4)
[2] G: (1,2)→(3,5)
[3] B: (0,0)→(4,3)

Drawn in different order!

What is wrong with this?
Hidden Lines
-polygon offset, draw twice

**Polygon Offset (depth-buffer biasing)**

```c
glEnable(GL_DEPTH_TEST);
glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
set_color(foreground);
draw_object_with_filled_polygons();
glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
glEnable(GL_POLYGON_OFFSET_FILL);
glPolygonOffset(1.0, 1.0);
set_color(background);
draw_object_with_filled_polygons();
glDisable(GL_POLYGON_OFFSET_FILL);
```

**Hidden Lines**
draw on per object basis with stencilling

```c
// Outline polygon (FG)
// setting the stencil
stencilFunc(GL_ALWAYS, 0, 0x1);
StencilOp(GL_INVERT, GL_INVERT, GL_INVERT);
Set color to foreground
Draw the polygon outline

// Fill polygon (BG)
where stencil is not set
stencilFunc(GL_EQUAL, 0, 0x1);
StencilOp(GL_KEEP, GL_KEEP, GL_KEEP);
Fill the polygon (BG)

// Outline polygon (FG)
// resetting stencil
stencilFunc(GL_ALWAYS, 0, 0x1);
StencilOp(GL_INVERT, GL_INVERT, GL_INVERT);
Set color to foreground
Draw the polygon outline
```

**Correct Version**
- Need to save/reset the depth-buffer for each object.
- See the web-page (Lectures notes) for the details

**Silhouettes**
- See web-page (lectures notes) solutions

Correct method – save the depth buffer
Slide credits
Dave Shreiner, Ed Angel, Vicki Shreiner
Siggraph 2000