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 Using Glut

1. Request a depth buffer
   ```
   glutInitDisplayMode( GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH );
   ```
2. Enable depth buffering
   ```
   glEnable( GL_DEPTH_TEST );
   ```
3. Clear color and depth buffers
   ```
   glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );
   ```
4. Render scene
5. Swap color buffers

Other Buffers
Other 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.

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

- 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.

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)
- 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\_ADD}, \text{GL\_MULT}
  - from read buffer: \text{GL\_ACCUM}, \text{GL\_LOAD}
  - transfer back to write buffer: \text{GL\_RETURN}
- \text{glAccum}(\text{GL\_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\_MULTISAMPLE}

Depth of Focus: Keeping a Plane in Focus

- Jitter the viewer to keep one plane unchanged
- \text{Back Plane}\text{Focal Plane}\text{Front Plane}

Depth of Field

Motion Blur
**Motion Blur w/o Accum.Buffer**

Details:
- Scene dynamically rendered to texture
- Modulate with a polygon (1,1,1,a)

```c
void RenderScene(void) {
    // Clear dark gray window
    glClearColor(0.2f, 0.2f, 1.2f, 0.0f);
    glClear(GL_COLOR_BUFFER_BIT);
    // Now set scissor to smaller gray sub region
    glClearColor(0.5f, 0.5f, 0.5f, 0.0f);
    glScissor(100, 100, 600, 400);
    glEnable(GL_SCISSOR_TEST);
    glClear(GL_COLOR_BUFFER_BIT);
    // Finally, an even smaller gray rectangle
    glClearColor(0.75f, 0.75f, 0.75f, 0.0f);
    glScissor(200, 200, 400, 200);
    glClear(GL_COLOR_BUFFER_BIT);
    // Turn scissor back off for next render
    glDisable(GL_SCISSOR_TEST);
    glutSwapBuffers();
}
```

**Fragment Operations**

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

**Scissor Box**

- **Additional Clipping Test**
  - `glScissor(x, y, w, h)`
  - Reject pixels based on their alpha value
  - `glClipPlane( func, value )`
  - `glEnable( GL_ALPHA_TEST )`

**Alpha Test** (deprecated)

- **Alpha Test**
  - `glAlphaFunc( func, value )`
  - Use alpha as a mask in textures
  - Alpha Test:
    - Accept pixels based on their alpha value
    - Implement transparency
    - Use depth test to filter opaque objects
    - Use depth of alpha equation to reject transparent fragments (from ruins the depth buffer)
An Interactive Introduction to OpenGL
Programming

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

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)`

`s` `t` `u` `v` `w` `x` `y` `z`

Pass Stencil?

- Update stencil buff w/ `fail` op
- discard fragment

Pass Depth?

- Update stencil buff w/ `zpass` op
- fragment → blending

- Keep
- Zero
- Replace
- Instr (_WRAP)
- Decr (_WRAP)
- Invert
An Interactive Introduction to OpenGL Programming

How to set the stencil?

- glInitDisplayMode(GLUT_STENCIL);
- glEnable(GL_STENCIL_TEST);
- glClearStencil(0x0);
- glStencilFunc(GL_ALWAYS, 0x1, 0x1);
- glStencilOp(GL_REPLACE, GL_REPLACE, GL_REPLACE);

• draw mask

Creating a Mask

Creating a Mask

- glInitDisplayMode(GLUT_STENCIL);
- glEnable(GL_STENCIL_TEST);
- glClearStencil(0x0);
- glStencilFunc(GL_ALWAYS, 0x1, 0x1);
- glStencilOp(GL_REPLACE, GL_REPLACE, GL_REPLACE);

• draw mask

Using Stencil Mask

- 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. 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

1. glColorMask(0.0, 0.0, 0.0, 0.0)
2. glDepthMask(0)
3. glEnable(stencil-test)
4. glStencilFunc(A, 0x01, 0x01)
5. Draw the window
6. glStencilFunc(=, 0x01)
7. glColorMask(1.0, 1.0, 1.0, 1.0)
8. glDepthMask(1)
9. Draw background
10. glStencilFunc(!=, 0x01, 0x01)
11. Draw wall
12. glDisable(stencil-test)
13. Draw anything else
Decal

How 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/ stencil buffer

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

// Draw |GREEN| rectangle
// Begin stencil write to stencil buffer
// So that all pixels in stencil buffer are 0
// except for area filled with 1
glStencilFunc(GL_ALWAYS, 1, 0); // Set stencil function
// Doesn't modify the color buffer
// Assign stencil value
// So that all pixels in stencil buffer are 0
// except for area filled with 1
// This is the stencil buffer
// Begin stencil write
// Don't change the color buffer
// Don't modify depth buffer
// Use stencil function
// Assign stencil value
// So that all pixels in stencil buffer are 0
// except for area filled with 1
// This is the color buffer
// Use color buffer
// Assign color value
// Don't change the stencil buffer
// Don't modify depth buffer
// Use color function
// Assign color value
// So that all pixels in stencil buffer are 0
// except for area filled with 1
// This is the depth buffer
// Use depth buffer
// Assign depth value
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

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

How to resolve z-fighting

Decal

Drawn in different order!

What is wrong with this?

What is wrong with this?
Hidden Lines

- Page 274 (294) – polygon offset, draw twice
- Polygon Offset (depth-buffer biasing)
- Page 622-623 (659) - draw on per object basis with stencilling
- Correct method

P. 623 (659)

- Outline polygon (FG) setting the stencil
  - glStencilFunc(GL_ALWAYS, 0, 0x1)
  - GLStencilOp(GL_INVERT, GL_INVERT, GL_INVERT)
  - Set color to foreground
  - Draw the polygon outline
- Fill polygon (BG) where stencil is not set
  - glStencilFunc(GL_EQUAL, 0, 0x1)
  - glStencilOp(GL_KEEP, GL_KEEP, GL_KEEP)
  - Fill the polygon (BG)
- Outline polygon (FG) resetting stencil
  - glStencilFunc(GL_ALWAYS, 0, 0x1)
  - GLStencilOp(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

Slide credits
Dave Shreiner, Ed Angel, Vicki Shreiner
Siggraph 2000