Rasterization: Images

Outline

- Pixmaps
- Blending
- Anti-aliasing
Pixmaps

• Pixmap is a chunk of memory representing an image
  – \( w \times h \) array of pixel values
  - Not necessarily the image, and not just color information
• Common formats
  - **Bitmap**: 1 bit per pixel—e.g., true/false, white/black
  - **Grayscale**: 1 byte per pixel = 256 values spanning black -> gray -> white
  - **True color (RGB)**: 3 color channels red, green, and blue (1 byte each)
  - **True color with alpha (RGBA)**: Transparency channel + color channels = 4 bytes per pixel

Pixmaps in OpenGL

• Categories
  - Off-screen (OS): What we can allocate & manipulate directly
  - Frame buffers (FB): Multiple kinds, only OpenGL can directly access
• Copy operations
  - `glReadPixels()`: Region of FB -> somewhere OS
  - `glCopyPixels()`: Region of FB -> elsewhere in FB
  - `glDrawPixels()`: Region of OS -> elsewhere in FB
FB Setup in GLUT

• `glutInitDisplayMode()` tells OpenGL what types of buffers the program will need
  • Types
    - **Color** (GLUT_RGBA, GLUT_DOUBLE, GLUT_STEREO)
    - **Depth** (GLUT_DEPTH)

Select an FB to Operate Pixels

• `glClear()`: Buffers are arguments
  - GL_COLOR_BUFFER_BIT
  - GL_DEPTH_BUFFER_BIT
• `glReadBuffer()` selects a color buffer to read (i.e., the source for subsequent pixel reads or copies)
  - GL_FRONT (default), GL_BACK (in setup of double buffer)
  - GL_LEFT, GL_RIGHT (in setup of stereoscopic)
• `glDrawBuffer()` selects a color buffer to draw (i.e., the destination for subsequent pixel draws or copies)
  - Similar to arguments of `glReadBuffer()`
Raster Position

- OpenGL state variable specifying lower-left corner \((x, y)\) (in 2-D) of destination for copies to frame buffers
  - `glCopyPixels()`, `glDrawPixels()`
- Set for subsequent copies with `glRasterPos()`

OpenGL: Drawing step-by-step

1. Make an off-screen image buffer
2. Set bits
3. Put that image in the OpenGL buffer

```c
// Step 1 (RGBA float pixmap)
GLfloat *im = (GLfloat*) malloc(4 * im_w * im_h, sizeof(GLfloat));
...
// Step 2 (\(r, g, b,\) and \(a\) are GLfloats)
im[4 * (x + im_w * y)] = r;
im[1 + 4 * (x + im_w * y)] = g;
im[2 + 4 * (x + im_w * y)] = b;
im[3 + 4 * (x + im_w * y)] = a;
...
// Step 3
glRasterPos2i(0, 0);
glDrawPixels(im_w, im_h, GL_RGBA, GL_FLOAT, im);
```
Compositing

• When a pixel is drawn to a buffer, what happens to what’s already there?
• Normally, we just overwrite... but there are more options
• Operations
  - Blending: Use alpha channel to control transparency vs. opacity
    - alpha = 1 => Perfect opacity (default)
    - alpha = 0 => Perfect transparency
    - In between, pixel is a mix of source and destination colors
  - Logical

Blending Details

• In general, new destination pixel values \( \mathbf{d}^0 = (r_{d0}, g_{d0}, b_{d0}, a_{d0}) \) are per-channel blend of source \( \mathbf{s} \) and old destination \( \mathbf{d} \) pixel values:
  \[
  \begin{align*}
  r_{d0} &= r_s s_r + r_d d_r \\
  g_{d0} &= g_s s_g + g_d d_g \\
  b_{d0} &= b_s s_b + b_d d_b \\
  a_{d0} &= a_s s_a + a_d d_a
  \end{align*}
  \]
  Destination blending factor
  Source blending factor

• More typically, \( \mathbf{d}^0 \) is just a linear combination of \( \mathbf{s} \) and \( \mathbf{d} \):
  \[
  \mathbf{d}^0 = \mathbf{s} \mathbf{s} + \mathbf{d} \mathbf{d}
  \]
Blending Details

- **over** operator: Intuitive use of alpha for color opacity/transparency

\[ d^0 = a_s \mathbf{s} + (1 - a_s) \mathbf{d} \]

Compositing in OpenGL

- **Blending**
  - Enable with `glEnable(GL_BLEND)`
  - `glBlendFunc()` sets blending parameters
    - Takes source, destination coefficients
    - E.g., `glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA)` implements **over** operation
Blending: Applications

- Chromakey: Make border region transparent
- Billboard: Place such images in the scene instead of more complex 3-D geometry

Aliasing (aka “jaggies,” “staircasing”)

- Problem is one of sampling: Pixels are samples of underlying continuous signal
Approaches to anti-aliasing

1. Filtering methods
   - **Prefiltering**: based on objects’ coverage
   - **Supersampling**: Generate higher-resolution virtual image

Prefiltering for anti-aliasing

- Standard approach of rounding when rasterizing only fills in primitive color pixels that are mostly covered and ignores all others
- Idea: Color each pixel with intensity proportional to fraction of area covered by primitive

![Prefiltering Example](image)
Prefiltering for anti-aliasing: Example

Ideal shape

Prefiltered rasterization

Supersampling

• Rasterize at higher resolution
Anti-aliasing in OpenGL

• Prefiltering
  - Must enable blending
  - Primitive types
    • Points: glEnable(GL_POINT_SMOOTH)
    • Lines: glEnable(GL_LINE_SMOOTH)
    • Polygons: glEnable(GL_POLYGON_SMOOTH)

• Supersampling
  - Scene anti-aliasing: Blend together multiple fractionally shifted images
  - See section on accumulation buffer in Red book, Chap. 10 (“The Frame Buffer”)