Attributeless rendering

CGT 520, Fall 2019

Attributeless rendering

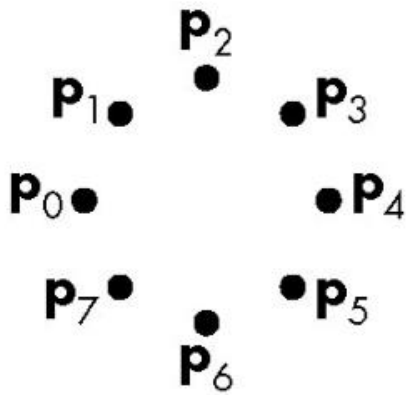
- This is an easy way to render simple objects and to learn about drawing geometry in OpenGL
- There are a few practical applications
  - This technique will not replace the use of Vertex Buffer Objects (VBOs)

Drawing vertices

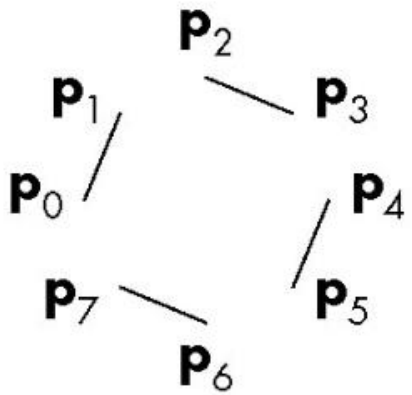
```
void glDrawArrays(GLenum mode, GLint first, GLsizei count);
```

- **mode**: Specifies what kind of primitives to render. Symbolic constants include GL_POINTS, GL_LINE_STRIP, GL_LINE_LOOP, GL_LINES GL_TRIANGLE_STRIP, GL_TRIANGLE_FAN, GL_TRIANGLES
- **first**: Specifies the starting index in the enabled arrays.
- **count**: Specifies the number of vertices to be rendered.

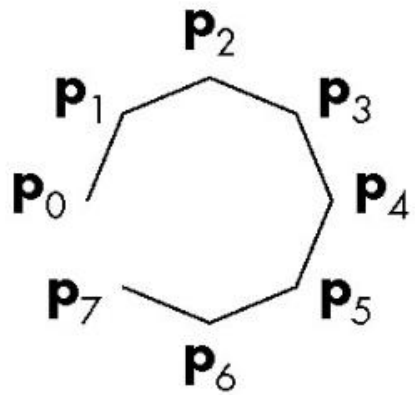
Primitive type (mode)



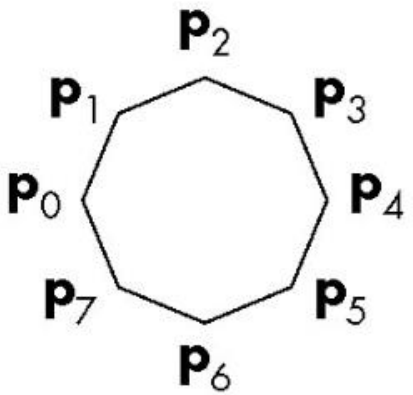
GL_POINTS



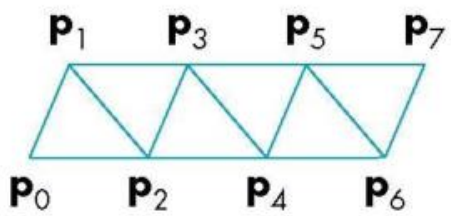
GL_LINES



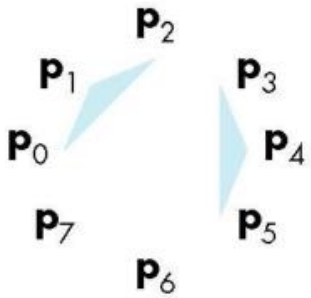
GL_LINE_STRIP



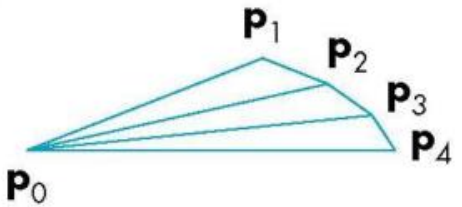
GL_LINE_LOOP



GL_TRIANGLE_STRIP



GL_TRIANGLES



GL_TRIANGLE_FAN

Steps to using

- Declare a global variable
  - `GLuint vao = -1;`
- In initialization function: Create a vertex array object
  - `glGenVertexArrays( 1, &vao );`
- In display function: Bind and draw
  - `glBindVertexArray( vao );`
  - `glDrawArrays(…);`

Example: full-screen quadrilateral

- Uses:
  - Full screen effects (blur, vignette, fade in/out, etc.)
  - Deferred shading
  - Clear screen to gradient or image instead of solid color
  - Prerendered HUD

Example: full-screen quadrilateral

- In display function: Bind and draw
  - `glBindVertexArray( vao );`
  - `glDrawArrays(GL_TRIANGLE_STRIP, 0, 4); //draw 4 vertex triangle strip`

- In vertex shader:

```
const vec4 quad[4] = vec4[]( vec4(-1.0,  1.0, 0.0, 1.0),
                             vec4(-1.0, -1.0, 0.0, 1.0),
                             vec4( 1.0,  1.0, 0.0, 1.0),
                             vec4( 1.0, -1.0, 0.0, 1.0) );

void main()
{
    gl_Position = quad[ gl_VertexID ];
}
```

Example: full-screen quadrilateral

```
const vec4 quad[4] = vec4[] ( vec4(-1.0,  1.0, 0.0, 1.0),
                              vec4(-1.0, -1.0, 0.0, 1.0),
                              vec4(  1.0,  1.0, 0.0, 1.0),
                              vec4( 1.0, -1.0, 0.0, 1.0) );

void main()
{
   gl_Position = quad[ gl_VertexID ];
}
```

- Things to notice about this code:
  - glsl syntax for array initialization is unlike C/C++
  - There are no ‘in’ variables
    - That’s why it is called “attributeless rendering”
  - **gl_VertexID** is built-in variable that automatically increments for each vertex drawn
  - No transformation matrix needed for full-screen quad: vertices are specified directly in clip coords so that they cover the view volume

Uses for attributeless rendering

- Fullscreen quad
- Completely procedural geometry generated in shaders
  - Particle systems
  - Terrains
  - Other things you can generate vertex coords based on gl_VertexID
- Skybox

Rendering with attributes

- Attributeless rendering is a neat trick with a few convenient uses
- Most rendering will happen with vertices fetched from Vertex Buffer Objects (VBOs)
- VBO memory is on the GPU and is fast to access
- It can be tricky to get your mesh data from main memory into the VBO in the right format