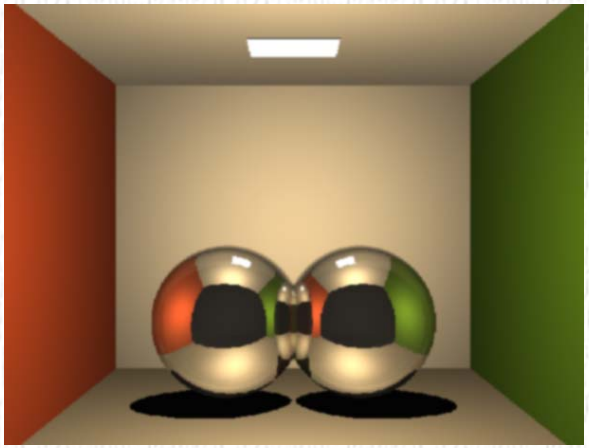
Lab 3

Kostas Vardis

http://graphics.cs.aueb.gr/graphics/people.html

Athens University of Economics and Business

Computer Graphics BSc



Introduction

- Vertex Attributes
- Vertex Buffer Objects
- Vertex Array Objects
- Primitive Types

Vertex Attributes

A shader expects a list of vertex data

Order of vertices is important

Example (triangle from lab1):

Vertex1: -1.0f, -1.0f, 0.0f

Vertex2: 1.0f, -1.0f, 0.0f

Vertex3: 0.0f, 1.0f, 0.0f

The vertex shader processes these in the order given (first v1, then v2, then v3)

Create a float buffer (v1, v2, v3)

Vertex Data

```
// counter-clockwise order
glm::vec3 v1(-1.0f, -1.0f, 0.0f); // bottom left
glm::vec3 v2(1.0f, -1.0f, 0.0f);  // bottom right
glm::vec3 v3(0.0f,  1.0f, 0.0f);  // top
std::vector<glm::vec3> triangle_data;
triangle_data.push_back(v1);
triangle_data.push_back(v2);
triangle_data.push_back(v3);
```

Vertex Data

Or create a struct:

```
struct VertexStruct {
glm::vec3 position;
glm::vec4 material_color
glm::vec3 normal;
glm::vec2 texcoords;
// etc.
};
```

Vertex Array Objects

- All that is needed to pass vertex data to GPU
- Only point to the buffers that store vertex attributes

Creating VAO example:

```
// Generate a VAO to point to buffer objects
glGenVertexArrays(1, &vao_triangle);
// Set the VAO active
glBindVertexArray(vao_triangle);
// store data in buffer objects (VBO) (shown below)
.
.
// enable each vertex attribute in the VAO
glEnableVertexAttribArray(0);
glEnableVertexAttribArray(1);
// etc.
```

Vertex Array Objects

Using VAO example:

```
// Set the shader active
glUseProgram(bgs_program_id);
// Pass uniform parameters
glUniformMatrix4fv(bgs_uniform_mvp, 1, false,
&mvp_matrix[0][0]);
// Bind the vertex array objects
glBindVertexArray(vao_quads);
// Draw!!
glDrawArrays(GL_POINTS, 0,
vao_quads_num_of_indices);
```

Vertex Buffer Objects

- VBOs store the vertex attribute data

Create VBO example:

```
GLuint vbo = 0;
// Generate a buffer object which holds the vertex data
glGenBuffers(1, &vbo);
// Bind the newly generated buffer object
glBindBuffer(GL_ARRAY_BUFFER, vbo);
// Pass the data
glBufferData(GL_ARRAY_BUFFER, total_size, &data[0], GL_STATIC_DRAW);
// Tell the VAO how the vertex data will be accessed
glVertexAttribPointer((GLuint)0, 3, GL_FLOAT, GL_FALSE,
sizeof(VertexData), (GLvoid*)(offset));
glVertexAttribPointer((GLuint)1, 4, GL_FLOAT, GL_FALSE,
sizeof(VertexData), (GLvoid*)(3 * sizeof(GLfloat)));
```

Vertex Buffer Objects

- VBOs also store index buffers

Example:

```
GLuint vbo_quad_index = 0;
// Generate a buffer object which holds the vertex data
glGenBuffers(1, &vbo_quad_index);
// Bind the newly generated buffer object
glBindBuffer(GL_ELEMENT_ARRAY_BUFFER,
vbo_quad_index);
// pass the data of the index buffer to the GPU as a
sequence of bytes
glBufferData(GL_ELEMENT_ARRAY_BUFFER,
total_index_size, &quad_index[0], GL_STATIC_DRAW);
```

Primitives

OpenGL assembles vertices into basic primitives

Three basic types:

- Points
- Lines
- Triangles

The most basic draw functions are glDrawArrays and glDrawElements

Primitives

Point Primitives

Example:

```
glDrawArrays(GL_POINTS, 0, num_indices);
```

Primitives

Line Primitives (lines and line strips)

Example:

```
glDrawArrays(GL_LINES, 0, num_indices);
glDrawArrays(GL_LINE_STRIP, 0, num_indices);
```

Primitives

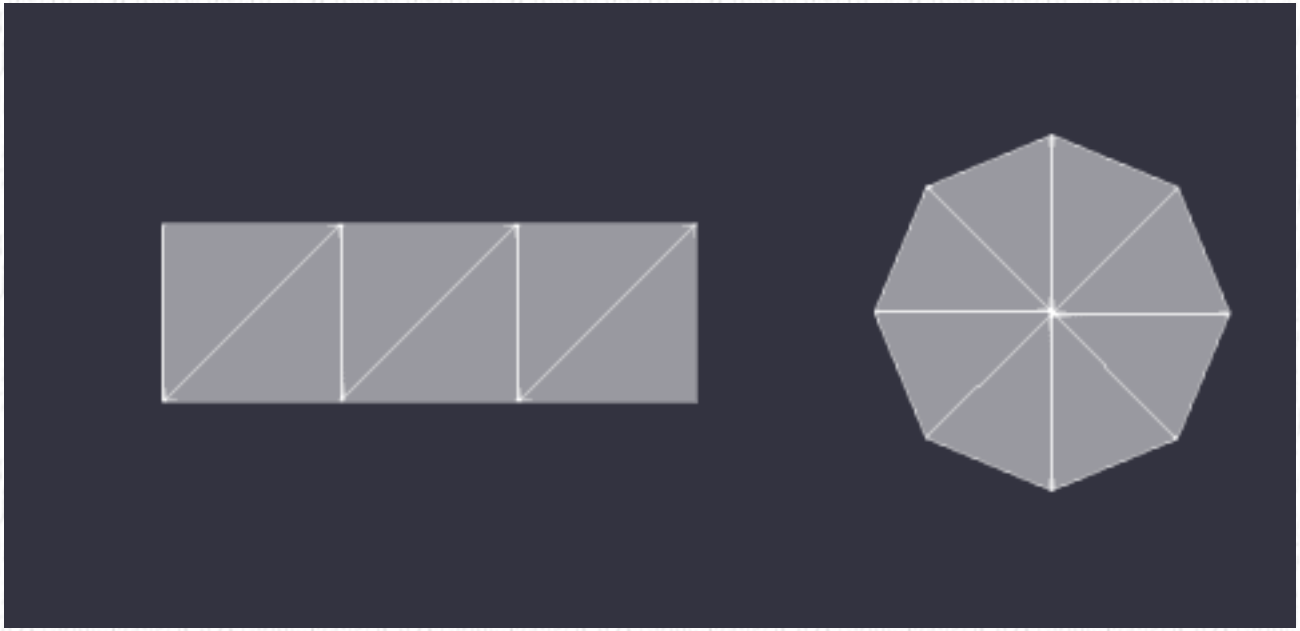
Triangle Primitives (triangles, strips and fans)

Example:

```
glDrawArrays(GL_TRIANGLES, 0, num_indices);
glDrawArrays(GL_TRIANGLE_STRIP, 0, num_indices);
glDrawArrays(GL_TRIANGLE_FAN, 0, num_indices);
```

Primitives

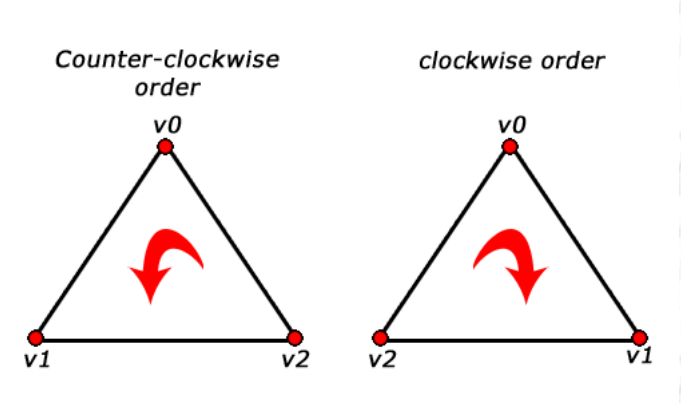
Triangle Primitives (triangles, strips and fans)

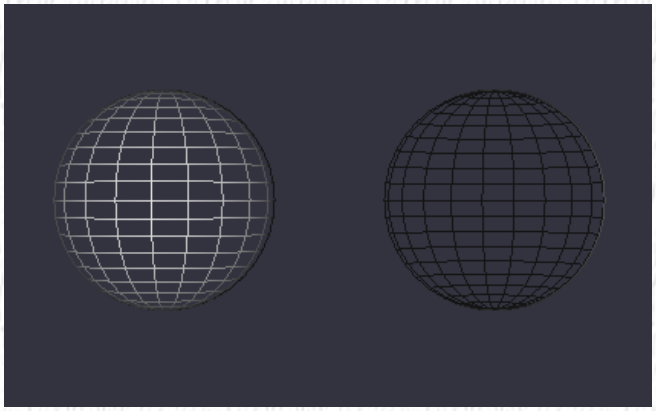


Face Culling

- Primitives can be culled depending on the order of the coordinates
- Use:

```
glEnable(GL_CULL_FACE);
glCullFace(GL_BACK);
glFrontFace(GL_CCW);
```

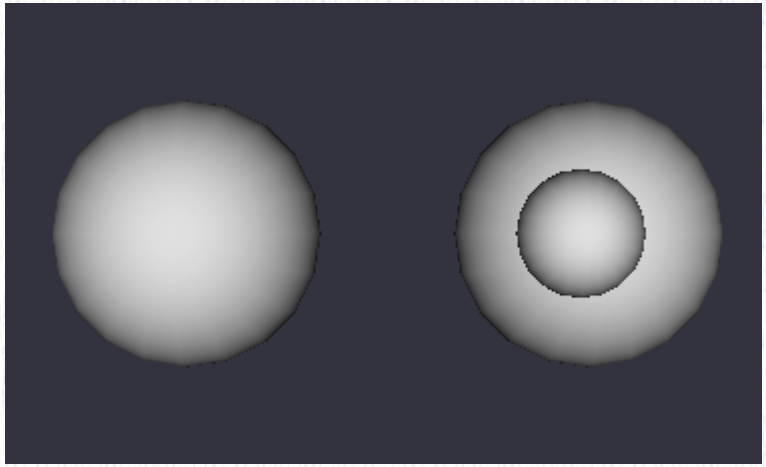




Depth Testing

- Need to make sure that objects that are nearer the camera are drawn first
- Use depth buffer and compare each fragment's depth value based on a function parameter

```
glEnable(GL_DEPTH_TEST);
glDepthFunc(GL_LEQUAL);
```



Done!

Check lab3 project