CSCD396

Beginning Graphics

Today’s topic

- Drawing polygon with **glDrawElements()**;
- Updating an object;
- Drawing multiple objects

OpenGL Drawing Commands (glDrawElements)

- void **glDrawElements**(GLenum *mode*, GLsizei *count*, GLenum *type*, const GLvoid **indices*);

- Parameters:
  - **mode**:
  - specifies what kind of primitives to render. Symbolic constants GL_POINTS, GL_LINE_STRIP, GL_LINE_LOOP, GL_LINES, GL_LINE_STRIP_ADJACENCY, GL_LINES_ADJACENCY, GL_TRIANGLE_STRIP, GL_TRIANGLE etc.

  - **count** specifies the number of elements to be rendered.

  - **type** specifies the type of the values in indices. Must be one of GL_UNSIGNED_BYTE, GL_UNSIGNED_SHORT, or GL_UNSIGNED_INT.

  - **indices** specifies a pointer to the location where the indices are stored.

OpenGL Drawing Commands (glDrawElements)

OpenGL Drawing Commands (glDrawElements)

```
// Color for each vertex
GLfloat colors[] =
{
1.0f, 1.0f, 1.0f, 1.0f,
1.0f, 1.0f, 0.0f, 1.0f,
1.0f, 0.0f, 1.0f, 1.0f,
0.0f, 1.0f, 1.0f, 1.0f
};

// Indices for the triangle strips
GLushort indices[] =
{
0, 1, 2, 2, 1, 3
};
```

OpenGL Drawing Commands (glDrawElements)

- glGenBuffers(1, &ebo);
- glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, ebo);
- glBufferData(GL_ELEMENT_ARRAY_BUFFER, sizeof(indices), indices, GL_STATIC_DRAW);

Updating Data

- **glBufferData** creates storage for the object.

- **glBufferSubData()** update the contents of the object's storage.

Updating data

- void glBufferData(GLenum target, GLsizeiptr size, const GLvoid * data, GLenum usage);
  - glBufferData actually *allocates* the memory for the buffer, as well as setting the contents; it may be filled with null data; this call uploads your data to the GPU.

```
glBufferData(GL_ARRAY_BUFFER, sizeof(vertex_positions) + sizeof(vertex_colors), NULL, GL_STATIC_DRAW);
glBufferSubData(GL_ARRAY_BUFFER, 0, sizeof(vertex_positions), vertex_positions);
glBufferSubData(GL_ARRAY_BUFFER, sizeof(vertex_positions), sizeof(vertex_colors), vertex_colors);
```

glBufferSubData

- void **glBufferSubData**(GLenum *target*, GLintptr *offset*, GLsizeiptr *size*, const GLvoid **data*);

- Replaces a subset of a buffer object’s data store with new data. The section of the buffer object bound to target starting at offset bytes is updated with the size bytes of data addressed by data.

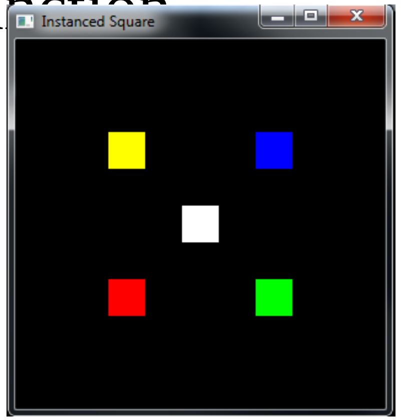
- Example : Deforming a square;

glBufferSubData

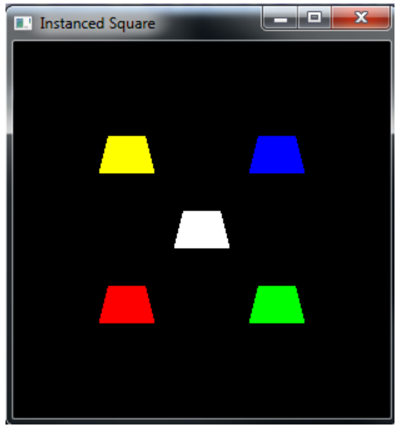
- glGenBuffers(1, vbo);
- glBindBuffer(GL_ARRAY_BUFFER, vbo[0]);
- glBufferData(GL_ARRAY_BUFFER, sizeof(vertex_positions) + sizeof(vertex_colors), NULL, GL_STATIC_DRAW);
- glBufferSubData(GL_ARRAY_BUFFER, 0, sizeof(vertex_positions), vertex_positions);
- glBufferSubData(GL_ARRAY_BUFFER, sizeof(vertex_positions), sizeof(vertex_colors), vertex_colors);

glBufferSubData

- Deforming an object: call **glBufferSubData** in the **Display** fu



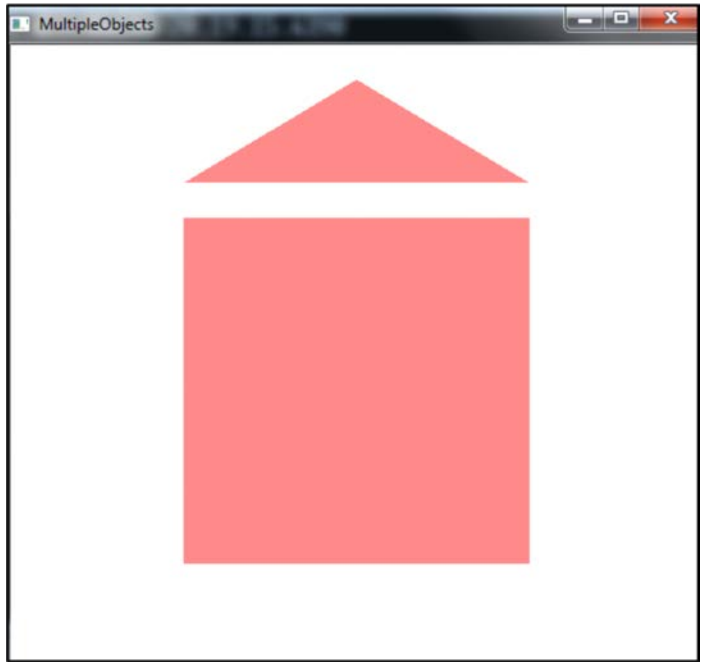
```
void Display(void)
{
glClear(GL_COLOR_BUFFER_BIT);

glBindVertexArray(square_vao);
if (update_vertices)
    glBufferSubData(GL_ARRAY_BUFFER, 0, sizeof(square_vertices_updated), square_vertices_updated);
else
   glBufferSubData(GL_ARRAY_BUFFER, 0, sizeof(square_vertices), square_vertices);

glDrawArraysInstanced(GL_TRIANGLE_FAN, 0, 4, 5);
glFlush();
}
```



Drawing multiple objects

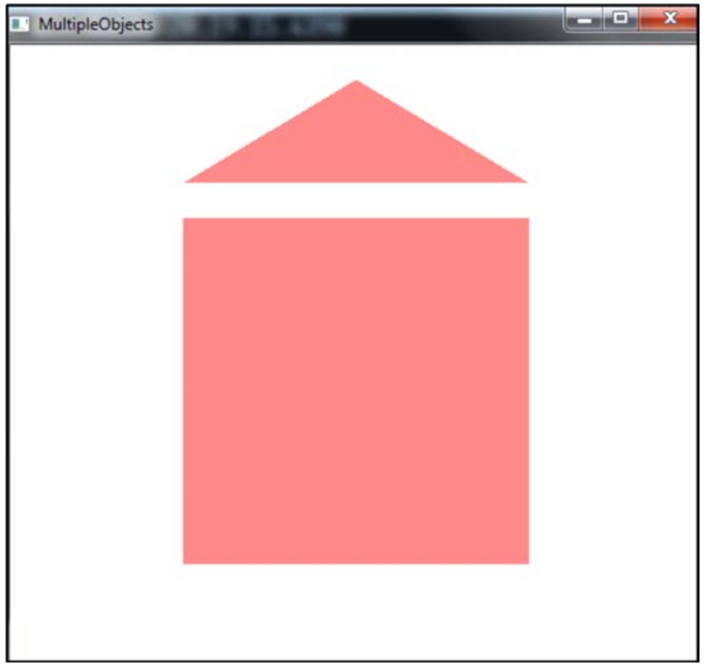
- There should be as many VAOs as the number of objects;

- There should be different sets of attributes for all objects:



Drawing multiple objects

```
GLuint  vao[2];
GLfloat square_vertices[] = { 0.5f, 0.5f, 0.0f, 1.0f,
                                          :
                                          :
                                        };

GLfloat tri_vertices[] = { -0.5f, 0.6f, 0.0f, 1.0f,
                                     :
                                     :
                                   };
```



Drawing multiple objects

- Initialize the objects:

```
glGenVertexArrays(2, vao);

glBindVertexArray(vao[0]);
glGenBuffers(1, &cube_vbo);
glBindBuffer(GL_ARRAY_BUFFER, cube_vbo);
:
glBindVertexArray(0);

glBindVertexArray(vao[1]);
glGenBuffers(1, &tri_vbo);
glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, tri_ebo);
:
glBindVertexArray(0);
```

Initialize the square

Initialize the triangle

Drawing multiple objects

- // draws a square

```
glBindVertexArray(vao[0]);
glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, cube_ebo);
glDrawElements(GL_TRIANGLES, 6, GL_UNSIGNED_SHORT, NULL);
glBindVertexArray(0);
```

draw the square

- // draw a triangle

```
glBindVertexArray(vao[1]);
glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, tri_ebo);
glDrawElements(GL_TRIANGLES, 3, GL_UNSIGNED_SHORT, NULL);
glBindVertexArray(0);
```

draw the triangle