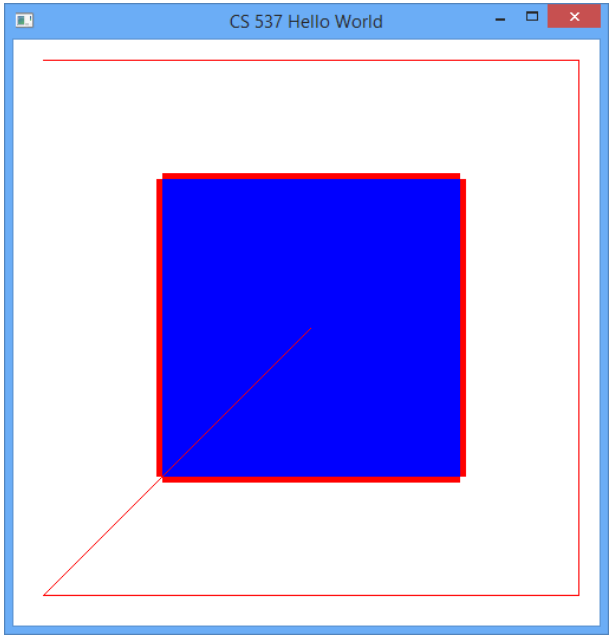
CS 432 – Interactive Computer Graphics

Lecture 2 – Part 3

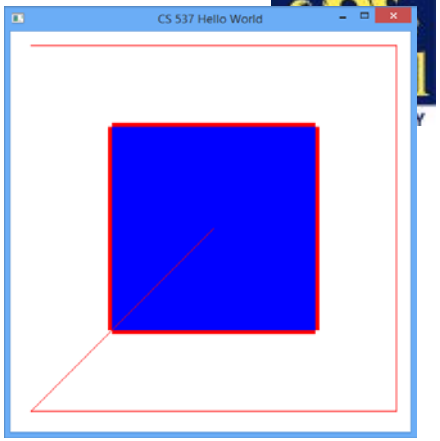
Another Example

Example: Multiple Objects, Multiple Colors

- Let's write a program that has:
  - A red line strip consisting of 5 vertices
  - A square with a blue interior and red edge



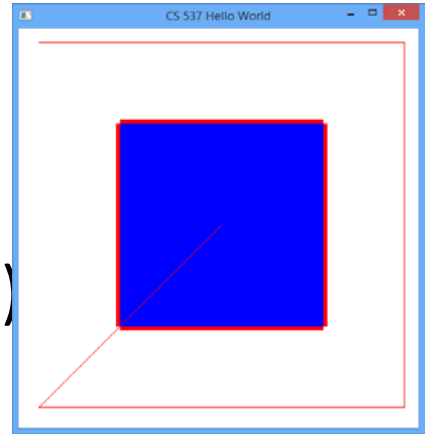
Example: Multiple Objects, Multiple Colors



- Need 4 vertices for our square
- Need 5 vertices for our line strip
- Note: Default “windows coordinate” have (-1,-1) as the bottom left, (1,1) as the top right

```
//Mesh 0
const int NumVertices = 4;

// Vertices of a unit cube centered at origin, sides aligned with axes
vec2 points[4] = {
    vec2( -0.5, 0.5),
    vec2( 0.5, 0.5),
    vec2( 0.5, -0.5),
    vec2( -0.5, -0.5)
};

//Line 1
const int NumVertices2 = 5;
vec2 points2[5] = {
    vec2( -0.9, 0.9),
    vec2( 0.9, 0.9),
    vec2( 0.9, -0.9),
    vec2( -0.9, -0.9),
    vec2( 0, 0)
};
```

Example: Multiple Objects, Multiple Colors



- Need 2 colors and 2 buffers
  (and 2 VAOs, each of which will have a VBO))

```
// RGBA colors
vec4 blue_opaque = vec4( 0.0, 0.0, 1.0, 1.0 );
vec4 red_opaque = vec4(1.0, 0.0, 0.0, 1.0);
//----------------------------------------------

GLuint VAOs[2];//VAO
GLuint VBOs[2];//VBO
GLuint color_loc;
GLuint program; //shader ID
```

- Both objects could use the same shader since they both
  - Have vertex location information in VBO
  - Will set all vertex colors to some uniform color.

Example: Multiple Objects, Multiple Colors

- In our `init` function we must generate the buffers and bind the appropriate data to each

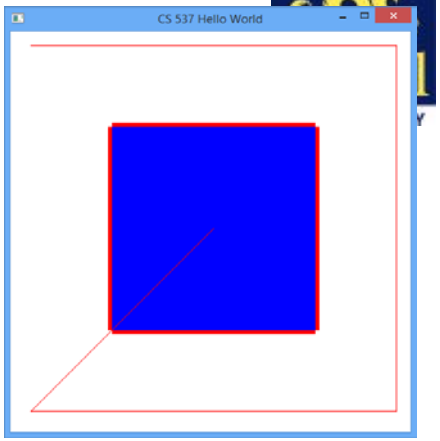
```
// OpenGL initialization
void init()
{
    //Get IDs for the VAOs and VBOs
    glGenVertexArrays(2, VAOs);
    glGenBuffers(2, VBOs);

    //Setup first object (square)
    glBindVertexArray(VAOs[0]);
    glBindBuffer( GL_ARRAY_BUFFER, VBOs[0] );
    glBufferData( GL_ARRAY_BUFFER, sizeof(points), points, GL_STATIC_DRAW );

    // Load shaders and use the resulting shader program
    program = InitShader( "vshader00_v150.glsl", "fshader00_v150.glsl" );
    glUseProgram( program );

    // set up vertex arrays
    GLuint vPosition = glGetAttribLocation( program, "vPosition" );

    glEnableVertexAttribArray( vPosition );
    glVertexAttribPointer( vPosition, 2, GL_FLOAT, GL_FALSE, 0, BUFFER_OFFSET(0) );
    color_loc = glGetUniformLocation(program, "color");
```

Example: Multiple Objects, Multiple Colors

- In our `init` function we must generate the buffers and bind the appropriate data to each

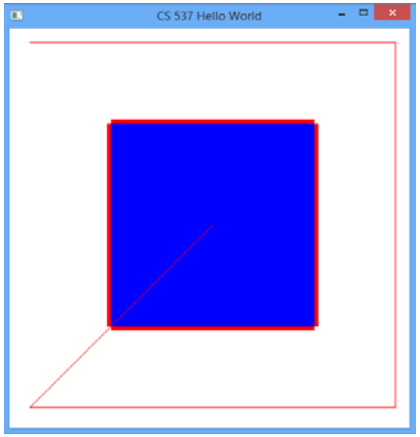
```
    //Setup second object (lines)
    glBindVertexArray(VAOs[1]);
    glBindBuffer(GL_ARRAY_BUFFER, VBOs[1]);
    glBufferData(GL_ARRAY_BUFFER, sizeof(points2), points2, GL_STATIC_DRAW);
    //since this will use the same shader program as object1 we don't need to re-read it

    glUseProgram(program);  //this was already loaded, but we'll do it again
    //we already know the location so no need to re-find it
    glEnableVertexAttribArray(vPosition);  //but lets make sure this VAO knows this attribute is enabled
    glVertexAttribPointer(vPosition, 2, GL_FLOAT, GL_FALSE, 0, BUFFER_OFFSET(0));
    //we already have the location of this shader's color variable

    glClearColor( 1.0, 1.0, 1.0, 1.0 );
}//end init
```

Example: Multiple Objects, Multiple Colors



- Finally we need to draw the objects
- For each of the objects
  - Make active the desired shader program and VAO
  - Set the shader's `color_loc` value
  - Maybe set some other state variables
  - Call `glDrawArrays` specifying the way to draw
- Do this three times
  - Once for the line loop
  - Once for the solid blue rectangle
  - Once for the red outline of the rectangle

Example: Multiple Objects, Multiple Colors




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

    //draw the filled square
    glBindVertexArray(VAOs[0]);
    glUseProgram(program);
    glUniform4fv(color_loc, 1, blue_opaque);
    glDrawArrays(GL_TRIANGLE_FAN, 0, 4);

    //draw the red outline of the square
    glBindVertexArray(VAOs[0]);  //already active so we don't really need this
    glUseProgram(program);  //already active so we don't really need this
    glLineWidth(10.0);
    glUniform4fv(color_loc, 1, red_opaque);
    glDrawArrays(GL_LINE_LOOP, 0, 4);

    //draw the line strip
    glBindVertexArray(VAOs[1]);
    glUseProgram(program); //already active so we don't really need this
    glLineWidth(1.0);
    glUniform4fv(color_loc, 1, red_opaque);
    glDrawArrays(GL_LINE_STRIP, 0, 5);

    glFlush();
}
```

Things to Think About

- Moving forward we're going to want to have several objects
  - Not to mention many instances of objects.
- It might be a good time to start thinking about object oriented programming….
- Each *object* should have…
  - VAO
  - VBO
  - Initialize method
  - Update method
  - Draw method
- Also maybe a good chance to use inheritance/abstract classes?