# CSCI3260
# Principles of Computer Graphics

---------Tutorial 4

XU Jiaqi

# OUTLINE

- Problems in your assignment 1
  - How to represent a model and copy data to GPU?
  - How to program if object is drawn with indexing?
  - How to represent multiple models and copy data to GPU?
  - How to use vertex shader & fragment shader to transfer model attributes?

# Problem 1

Problem 1 –
How to represent a model and copy data to GPU?

# Problem 1

## Problem 1 –
## How to represent a model and copy data to GPU?

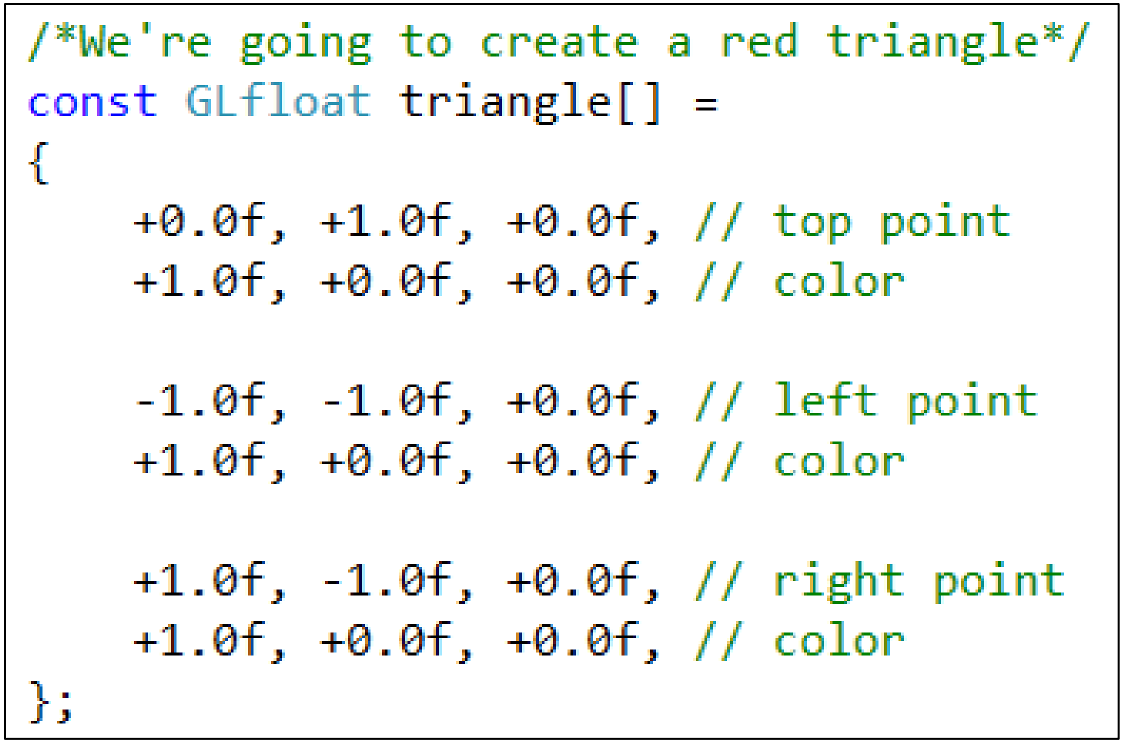
$$\begin{matrix}(one\ model\ with\ one\ array)\\ \begin{bmatrix} Position\ Data\\ Color\ Data\\ Normal\ Data\\ \dots\dots \end{bmatrix} \Rightarrow one\ Vertex\ Array\ Object \Rightarrow one\ Vertex\ Buffer\ Object\ \#1\end{matrix}$$

$$\begin{matrix}(one\ model\ with\ multiple\ arrays)\\ [Position\ Data]\\ [Color\ Data]\\ [Normal\ Data]\\ [\dots\dots]\end{matrix} \Rightarrow \text{one VAO} \Rightarrow \begin{cases} multiple\ VBOs \begin{cases} Buffer\ Data\ of\ Position\\ Buffer\ Data\ of\ Color\\ Buffer\ Data\ of\ Normal\\ \dots\dots \end{cases} \#2 \\ \\ one\ VBO \begin{cases} BufferSubData\ of\ Position\\ BufferSubData\ of\ Color\\ BufferSubData\ of\ Normal\\ \dots\dots \end{cases} \#3 \end{cases}$$

# Problem 1- #1

$(one\ model\ with\ one\ array)$

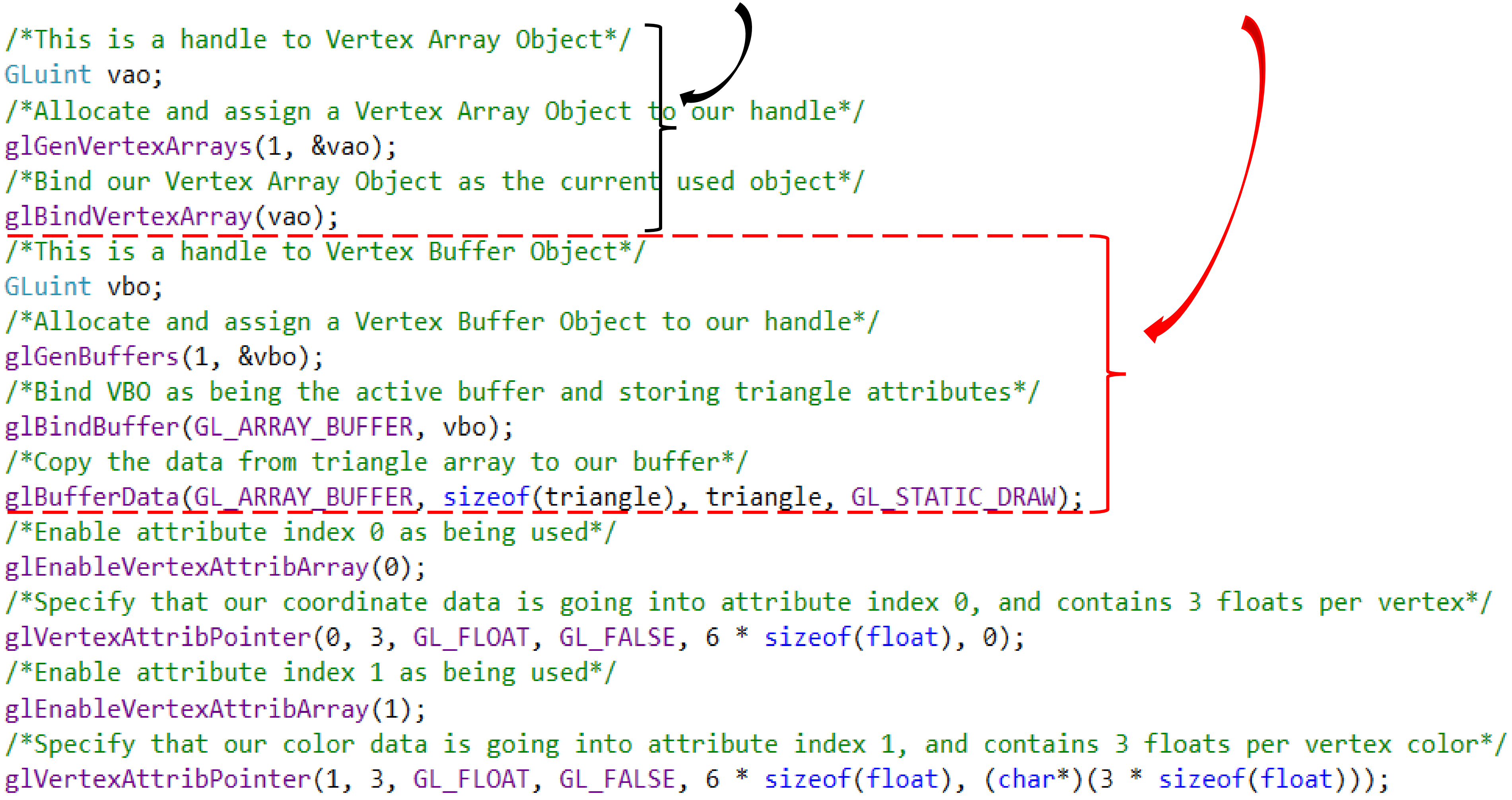
$$\begin{bmatrix} Position\ Data \\ Color\ Data \\ Normal\ Data \\ \dots\dots \end{bmatrix} \Rightarrow one\ Vertex\ Array\ Object \Rightarrow one\ Vertex\ Buffer\ Object\ \#1$$



```
/*We're going to create a red triangle*/
const GLfloat triangle[] =
{
    +0.0f, +1.0f, +0.0f, // top point
    +1.0f, +0.0f, +0.0f, // color

    -1.0f, -1.0f, +0.0f, // left point
    +1.0f, +0.0f, +0.0f, // color

    +1.0f, -1.0f, +0.0f, // right point
    +1.0f, +0.0f, +0.0f, // color
};
```

# Problem 1- #1

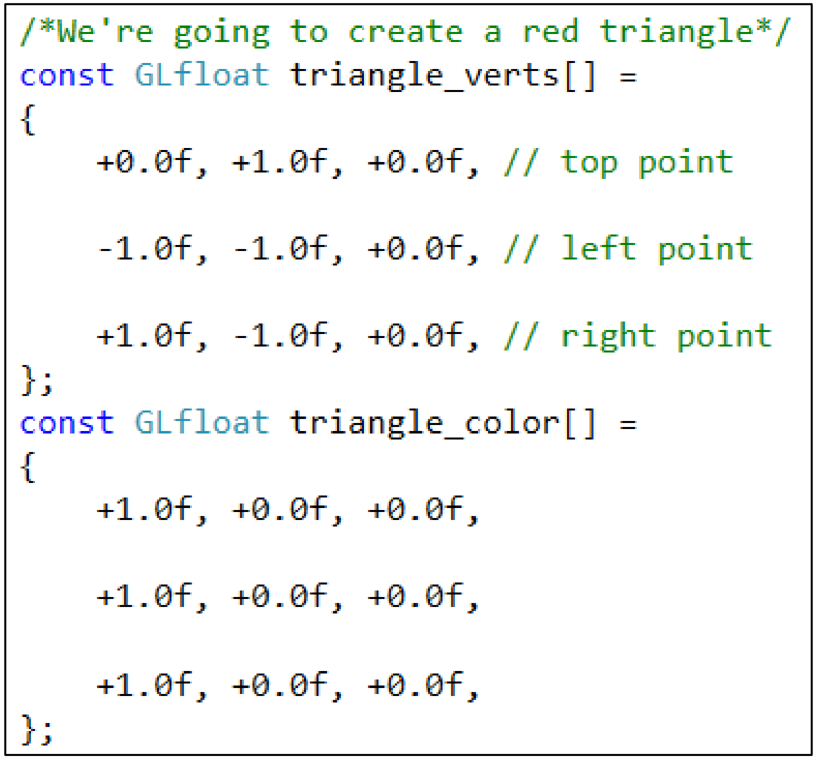
$$one\ model\ array \Rightarrow \underline{one\ Vertex\ Array\ Object} \Rightarrow \underline{one\ Vertex\ Buffer\ Object}\ \#1$$



```
/*This is a handle to Vertex Array Object*/
GLuint vao;
/*Allocate and assign a Vertex Array Object to our handle*/
glGenVertexArrays(1, &vao);
/*Bind our Vertex Array Object as the current used object*/
glBindVertexArray(vao);
/*This is a handle to Vertex Buffer Object*/
GLuint vbo;
/*Allocate and assign a Vertex Buffer Object to our handle*/
glGenBuffers(1, &vbo);
/*Bind VBO as being the active buffer and storing triangle attributes*/
glBindBuffer(GL_ARRAY_BUFFER, vbo);
/*Copy the data from triangle array to our buffer*/
glBufferData(GL_ARRAY_BUFFER, sizeof(triangle), triangle, GL_STATIC_DRAW);
/*Enable attribute index 0 as being used*/
glEnableVertexAttribArray(0);
/*Specify that our coordinate data is going into attribute index 0, and contains 3 floats per vertex*/
glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 6 * sizeof(float), 0);
/*Enable attribute index 1 as being used*/
glEnableVertexAttribArray(1);
/*Specify that our color data is going into attribute index 1, and contains 3 floats per vertex color*/
glVertexAttribPointer(1, 3, GL_FLOAT, GL_FALSE, 6 * sizeof(float), (char*)(3 * sizeof(float)));
```

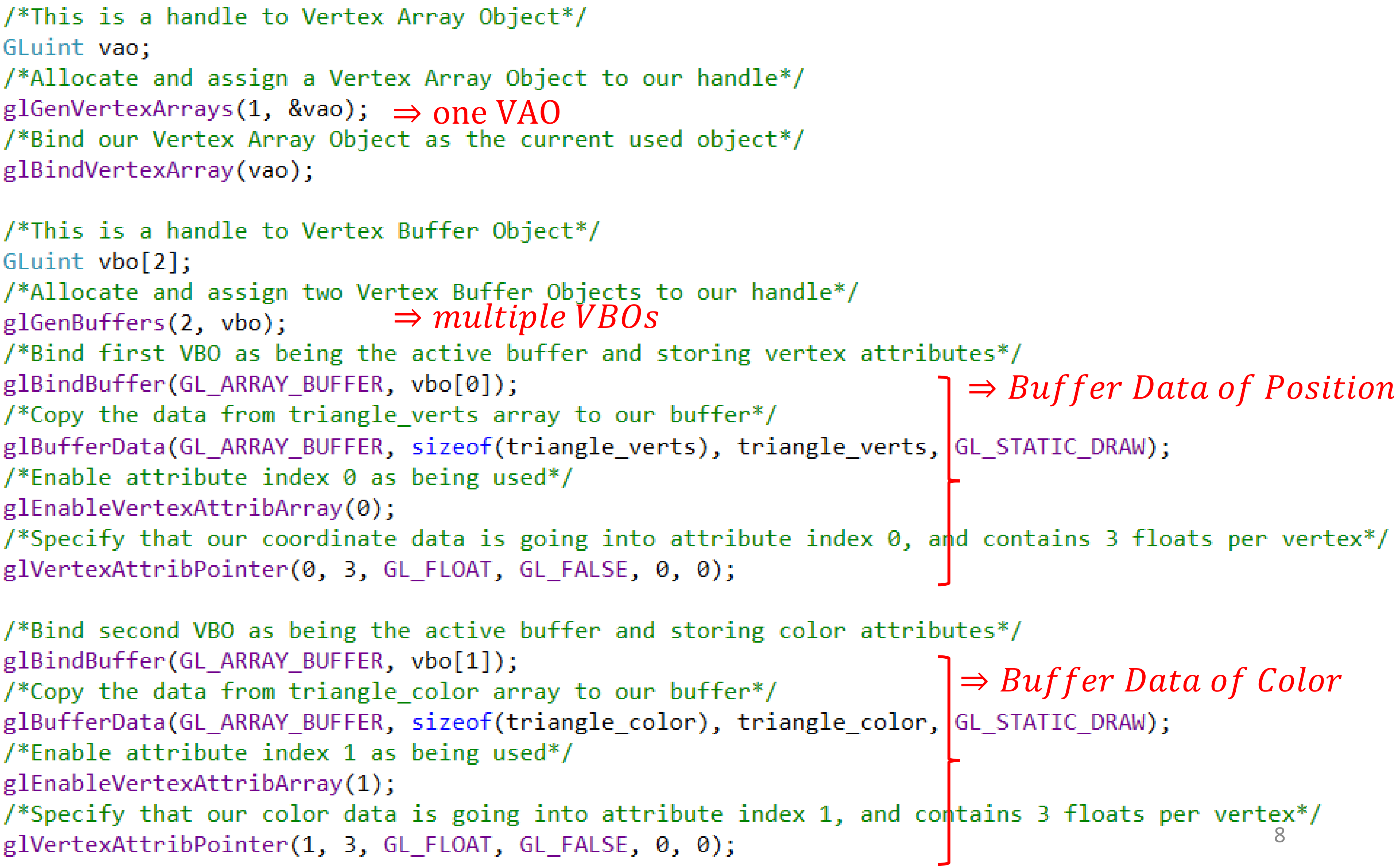
# Problem 1- #2

$$\begin{matrix}(one\ model\ with\ multiple\ arrays)\\ [Position\ Data]\\ [Color\ Data]\\ [Normal\ Data]\\ [\dots\dots]\end{matrix} \Rightarrow \text{one VAO} \Rightarrow \begin{cases} multiple\ VBOs \begin{cases} Buffer\ Data\ of\ Position\\ Buffer\ Data\ of\ Color\\ Buffer\ Data\ of\ Normal\\ \dots\dots \end{cases} \quad \#2 \\ one\ VBO \begin{cases} BufferSubData\ of\ Position\\ BufferSubData\ of\ Color\\ BufferSubData\ of\ Normal\\ \dots\dots \end{cases} \quad \#3 \end{cases}$$



```
/*We're going to create a red triangle*/
const GLfloat triangle_verts[] =
{
    +0.0f, +1.0f, +0.0f, // top point

    -1.0f, -1.0f, +0.0f, // left point

    +1.0f, -1.0f, +0.0f, // right point
};
const GLfloat triangle_color[] =
{
    +1.0f, +0.0f, +0.0f,

    +1.0f, +0.0f, +0.0f,

    +1.0f, +0.0f, +0.0f,
};
```

# Problem 1- #2



```
/*This is a handle to Vertex Array Object*/
GLuint vao;
/*Allocate and assign a Vertex Array Object to our handle*/
glGenVertexArrays(1, &vao);
```

⇒ one VAO

```
/*Bind our Vertex Array Object as the current used object*/
glBindVertexArray(vao);

/*This is a handle to Vertex Buffer Object*/
GLuint vbo[2];
/*Allocate and assign two Vertex Buffer Objects to our handle*/
glGenBuffers(2, vbo);
```

⇒ *multiple VBOs*

```
/*Bind first VBO as being the active buffer and storing vertex attributes*/
glBindBuffer(GL_ARRAY_BUFFER, vbo[0]);
/*Copy the data from triangle_verts array to our buffer*/
glBufferData(GL_ARRAY_BUFFER, sizeof(triangle_verts), triangle_verts, GL_STATIC_DRAW);
/*Enable attribute index 0 as being used*/
glEnableVertexAttribArray(0);
/*Specify that our coordinate data is going into attribute index 0, and contains 3 floats per vertex*/
glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 0, 0);
```

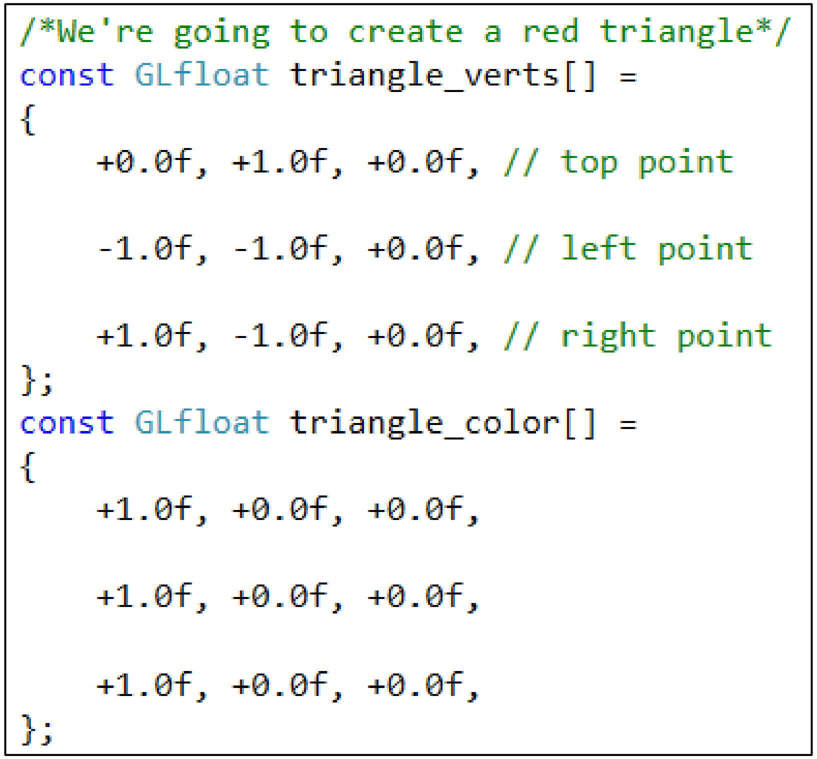
⇒ *Buffer Data of Position*

```
/*Bind second VBO as being the active buffer and storing color attributes*/
glBindBuffer(GL_ARRAY_BUFFER, vbo[1]);
/*Copy the data from triangle_color array to our buffer*/
glBufferData(GL_ARRAY_BUFFER, sizeof(triangle_color), triangle_color, GL_STATIC_DRAW);
/*Enable attribute index 1 as being used*/
glEnableVertexAttribArray(1);
/*Specify that our color data is going into attribute index 1, and contains 3 floats per vertex*/
glVertexAttribPointer(1, 3, GL_FLOAT, GL_FALSE, 0, 0);
```

⇒ *Buffer Data of Color*

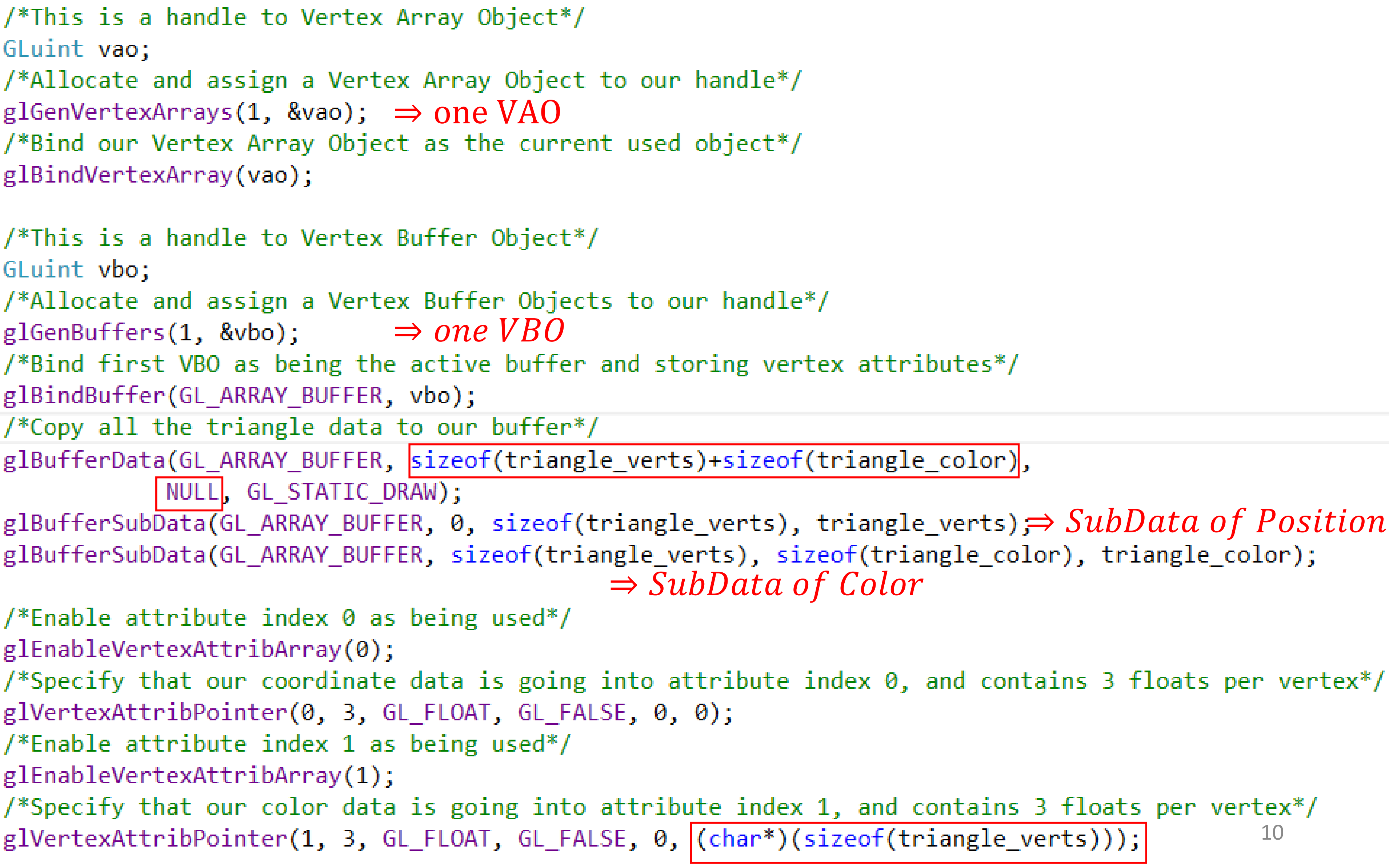
# Problem 1- #3

$$\begin{matrix} (\textit{one model with multiple arrays}) \\ [\textit{Position Data}] \\ [\textit{Color Data}] \\ [\textit{Normal Data}] \\ [\dots\dots] \end{matrix} \Rightarrow \text{one VAO} \Rightarrow \begin{cases} \textit{multiple VBOs} \begin{cases} \textit{Buffer Data of Position} \\ \textit{Buffer Data of Color} \\ \textit{Buffer Data of Normal} \\ \dots\dots \end{cases} \#2 \\ \textit{one VBO} \begin{cases} \textit{BufferSubData of Position} \\ \textit{BufferSubData of Color} \\ \textit{BufferSubData of Normal} \\ \dots\dots \end{cases} \#3 \end{cases}$$



```
/*We're going to create a red triangle*/
const GLfloat triangle_verts[] =
{
    +0.0f, +1.0f, +0.0f, // top point

    -1.0f, -1.0f, +0.0f, // left point

    +1.0f, -1.0f, +0.0f, // right point
};
const GLfloat triangle_color[] =
{
    +1.0f, +0.0f, +0.0f,

    +1.0f, +0.0f, +0.0f,

    +1.0f, +0.0f, +0.0f,
};
```

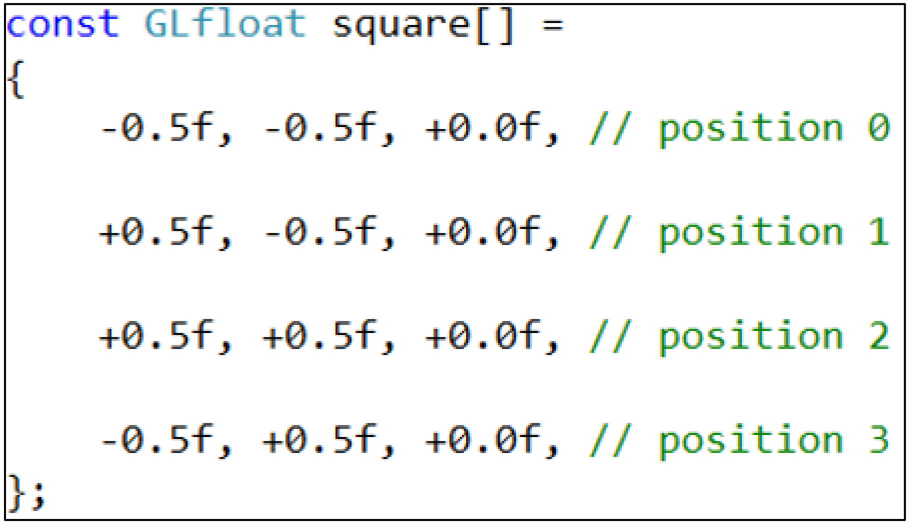
# Problem 1- #3



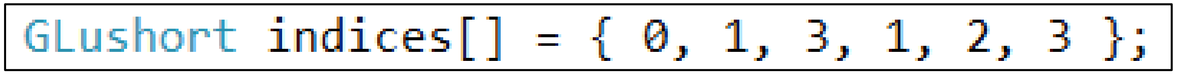
```
/*This is a handle to Vertex Array Object*/
GLuint vao;
/*Allocate and assign a Vertex Array Object to our handle*/
glGenVertexArrays(1, &vao);  ⇒ one VAO
/*Bind our Vertex Array Object as the current used object*/
glBindVertexArray(vao);

/*This is a handle to Vertex Buffer Object*/
GLuint vbo;
/*Allocate and assign a Vertex Buffer Objects to our handle*/
glGenBuffers(1, &vbo);       ⇒ one VBO
/*Bind first VBO as being the active buffer and storing vertex attributes*/
glBindBuffer(GL_ARRAY_BUFFER, vbo);
/*Copy all the triangle data to our buffer*/
glBufferData(GL_ARRAY_BUFFER, sizeof(triangle_verts)+sizeof(triangle_color),
            NULL, GL_STATIC_DRAW);
glBufferSubData(GL_ARRAY_BUFFER, 0, sizeof(triangle_verts), triangle_verts);⇒ SubData of Position
glBufferSubData(GL_ARRAY_BUFFER, sizeof(triangle_verts), sizeof(triangle_color), triangle_color);
                                              ⇒ SubData of Color
/*Enable attribute index 0 as being used*/
glEnableVertexAttribArray(0);
/*Specify that our coordinate data is going into attribute index 0, and contains 3 floats per vertex*/
glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 0, 0);
/*Enable attribute index 1 as being used*/
glEnableVertexAttribArray(1);
/*Specify that our color data is going into attribute index 1, and contains 3 floats per vertex*/
glVertexAttribPointer(1, 3, GL_FLOAT, GL_FALSE, 0, (char*)(sizeof(triangle_verts)));
```

# Problem 2

## Problem 2 –
## How to program if object is drawn with indexing?



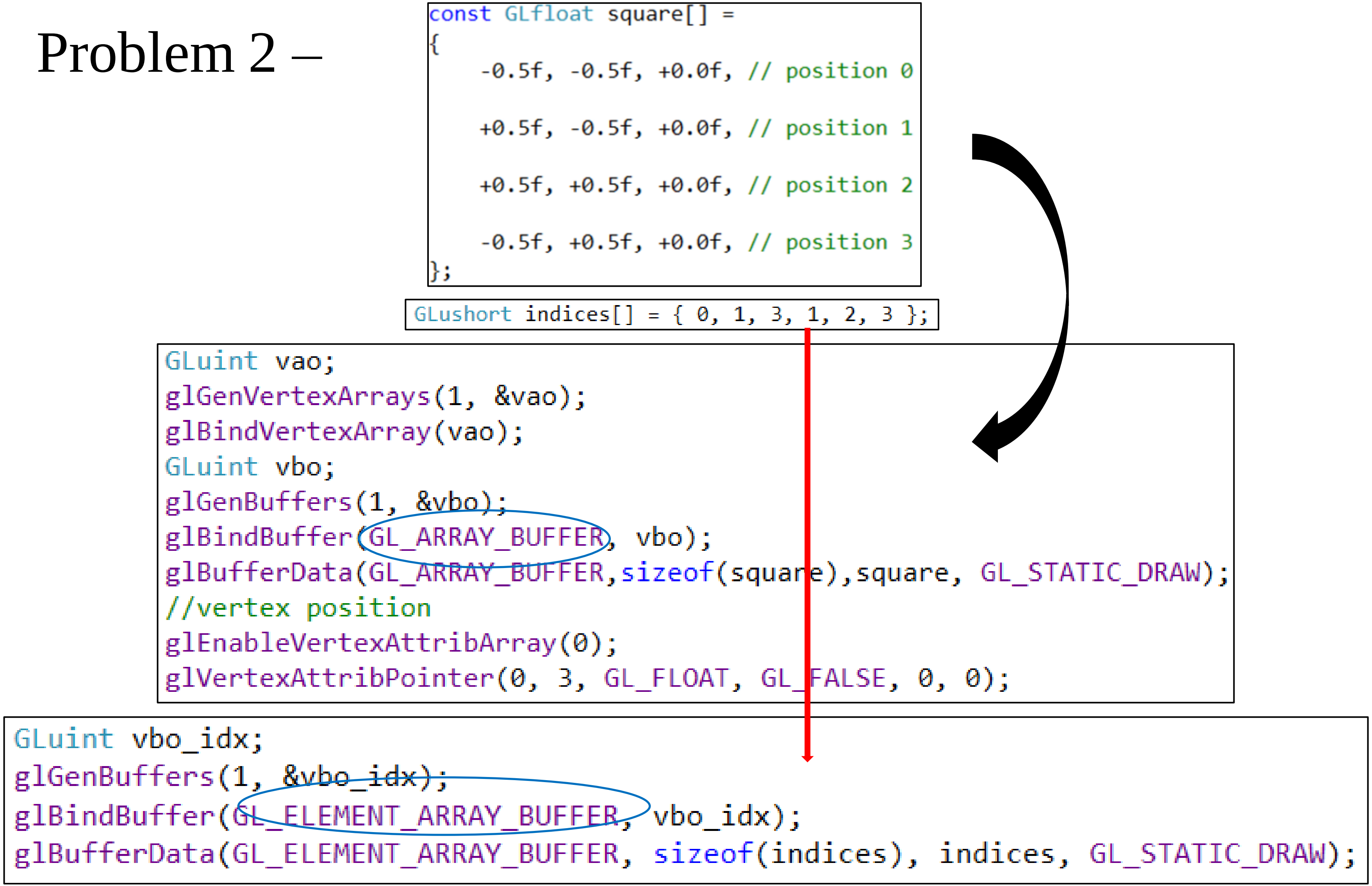
```
const GLfloat square[] =
{
    -0.5f, -0.5f, +0.0f, // position 0

    +0.5f, -0.5f, +0.0f, // position 1

    +0.5f, +0.5f, +0.0f, // position 2

    -0.5f, +0.5f, +0.0f, // position 3
};
```



```
GLushort indices[] = { 0, 1, 3, 1, 2, 3 };
```

# Problem 2

## Problem 2 –



```
const GLfloat square[] =
{
    -0.5f, -0.5f, +0.0f, // position 0

    +0.5f, -0.5f, +0.0f, // position 1

    +0.5f, +0.5f, +0.0f, // position 2

    -0.5f, +0.5f, +0.0f, // position 3
};
```

```
GLushort indices[] = { 0, 1, 3, 1, 2, 3 };
```

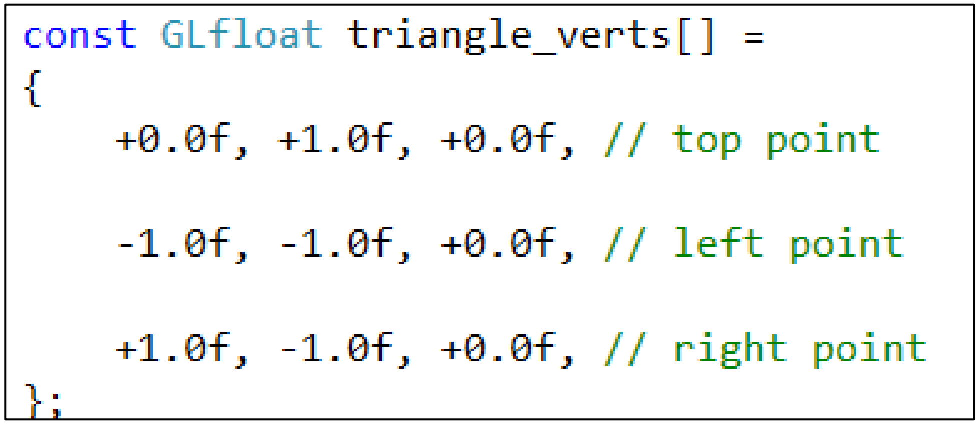
```
GLuint vao;
glGenVertexArrays(1, &vao);
glBindVertexArray(vao);
GLuint vbo;
glGenBuffers(1, &vbo);
glBindBuffer(GL_ARRAY_BUFFER, vbo);
glBufferData(GL_ARRAY_BUFFER,sizeof(square),square, GL_STATIC_DRAW);
//vertex position
glEnableVertexAttribArray(0);
glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 0, 0);
```

```
GLuint vbo_idx;
glGenBuffers(1, &vbo_idx);
glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, vbo_idx);
glBufferData(GL_ELEMENT_ARRAY_BUFFER, sizeof(indices), indices, GL_STATIC_DRAW);
```

Use *glDrawElements* to replace *glDrawArrays*

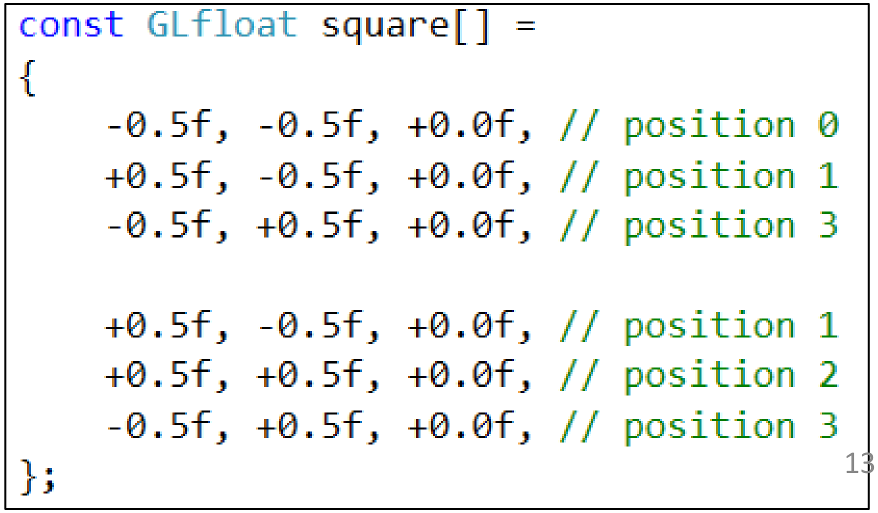
# Problem 3

## Problem 3 –
## How to represent multiple models and copy data to GPU?



```
const GLfloat triangle_verts[] =
{
    +0.0f, +1.0f, +0.0f, // top point

    -1.0f, -1.0f, +0.0f, // left point

    +1.0f, -1.0f, +0.0f, // right point
};
```

$\Rightarrow model\ 1$



```
const GLfloat square[] =
{
    -0.5f, -0.5f, +0.0f, // position 0
    +0.5f, -0.5f, +0.0f, // position 1
    -0.5f, +0.5f, +0.0f, // position 3

    +0.5f, -0.5f, +0.0f, // position 1
    +0.5f, +0.5f, +0.0f, // position 2
    -0.5f, +0.5f, +0.0f, // position 3
};
```

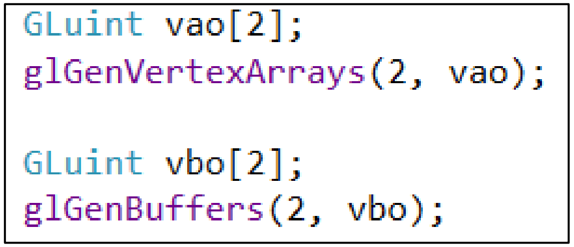
$model\ 2 \Leftarrow$

# Problem 3

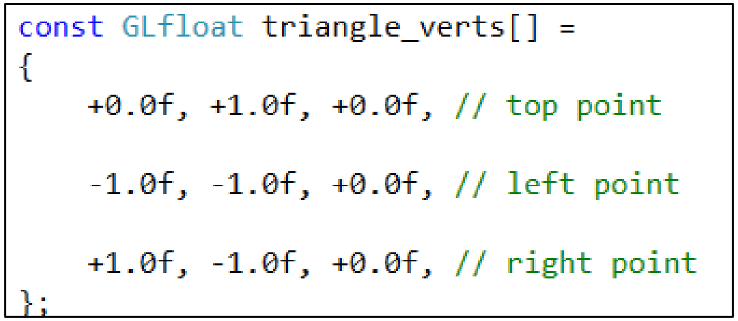
Problem 3 –

$[model\ 1] \Rightarrow vao1 \Rightarrow vbo1$

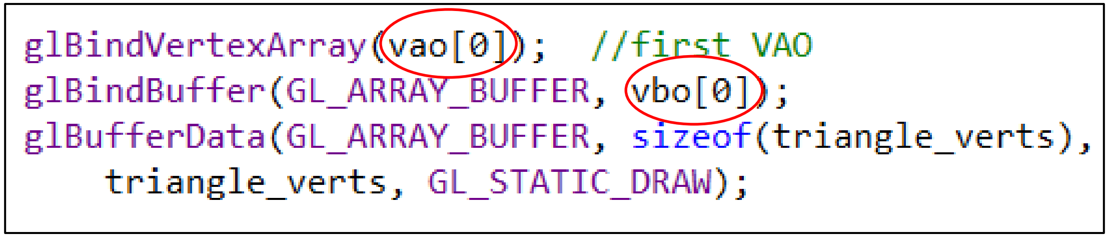
$[model\ 2] \Rightarrow vao2 \Rightarrow vbo2$



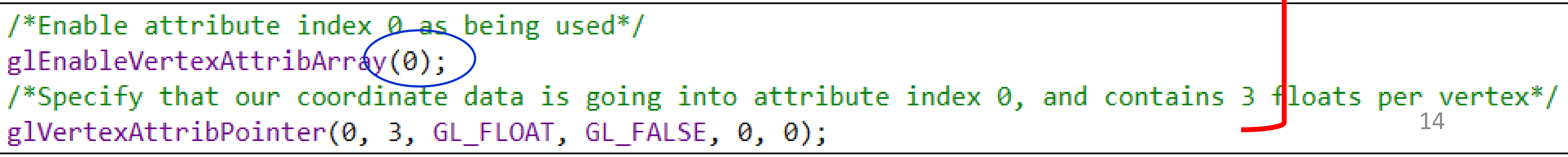
```
GLuint vao[2];
glGenVertexArrays(2, vao);

GLuint vbo[2];
glGenBuffers(2, vbo);
```



```
const GLfloat triangle_verts[] =
{
    +0.0f, +1.0f, +0.0f, // top point

    -1.0f, -1.0f, +0.0f, // left point

    +1.0f, -1.0f, +0.0f, // right point
};
```



```
glBindVertexArray(vao[0]);  //first VAO
glBindBuffer(GL_ARRAY_BUFFER, vbo[0]);
glBufferData(GL_ARRAY_BUFFER, sizeof(triangle_verts),
    triangle_verts, GL_STATIC_DRAW);
```



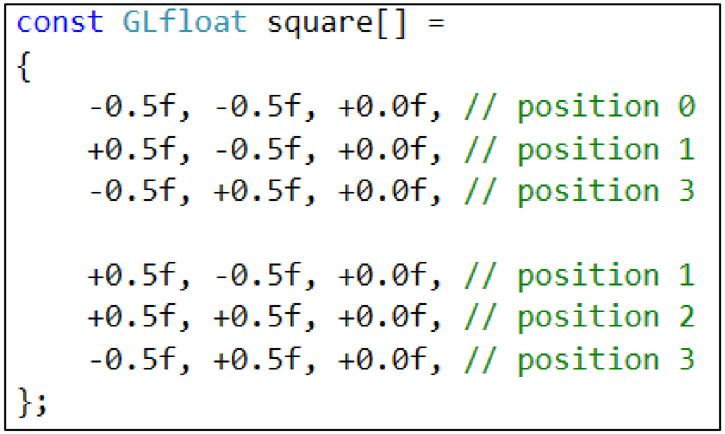
```
/*Enable attribute index 0 as being used*/
glEnableVertexAttribArray(0);
/*Specify that our coordinate data is going into attribute index 0, and contains 3 floats per vertex*/
glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 0, 0);
```

# Problem 3

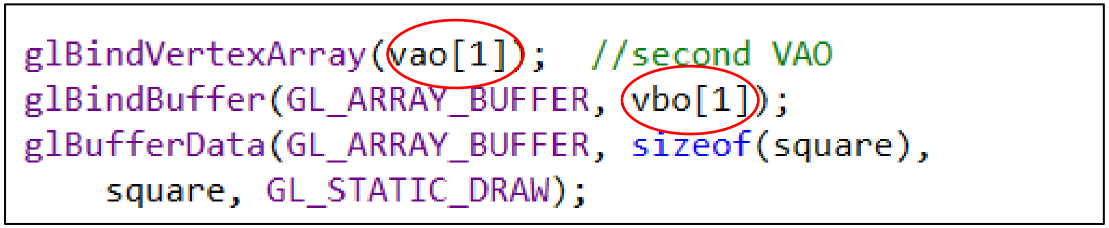
Problem 3 –

$$[model\ 1] \Rightarrow vao1 \Rightarrow vbo1$$

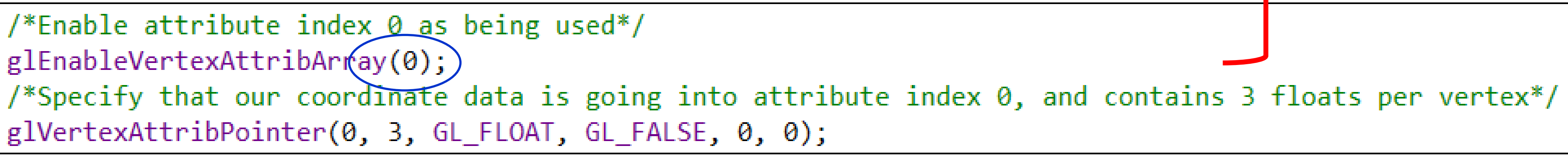
$$[model\ 2] \Rightarrow vao2 \Rightarrow vbo2$$



```
const GLfloat square[] =
{
    -0.5f, -0.5f, +0.0f, // position 0
    +0.5f, -0.5f, +0.0f, // position 1
    -0.5f, +0.5f, +0.0f, // position 3

    +0.5f, -0.5f, +0.0f, // position 1
    +0.5f, +0.5f, +0.0f, // position 2
    -0.5f, +0.5f, +0.0f, // position 3
};
```



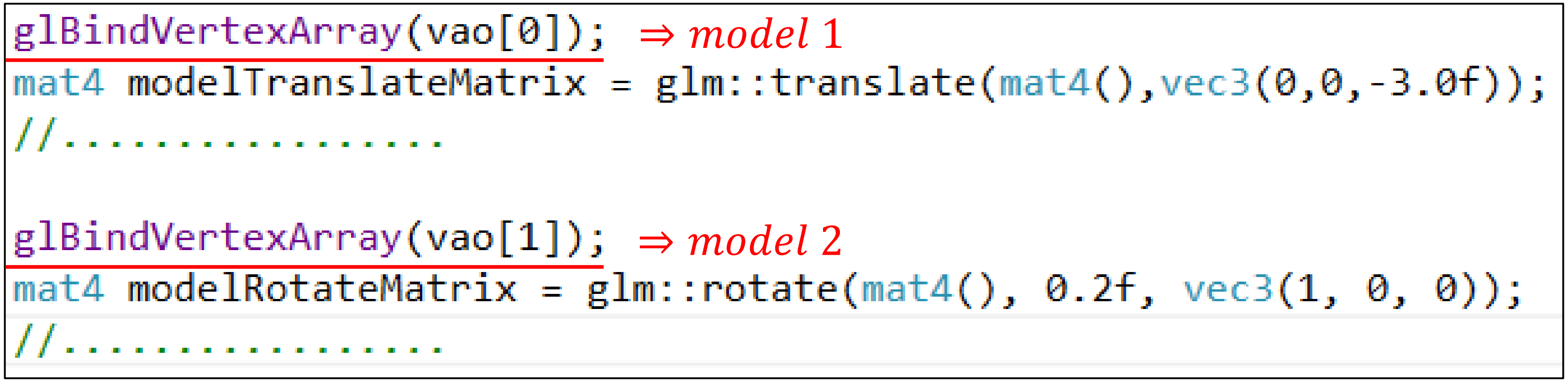
```
glBindVertexArray(vao[1]);  //second VAO
glBindBuffer(GL_ARRAY_BUFFER, vbo[1]);
glBufferData(GL_ARRAY_BUFFER, sizeof(square),
    square, GL_STATIC_DRAW);
```



```
/*Enable attribute index 0 as being used*/
glEnableVertexAttribArray(0);
/*Specify that our coordinate data is going into attribute index 0, and contains 3 floats per vertex*/
glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 0, 0);
```

# Problem 3

If you want to do different operations on different models, Then:



```
glBindVertexArray(vao[0]);
mat4 modelTranslateMatrix = glm::translate(mat4(),vec3(0,0,-3.0f));
//..................
```

$\Rightarrow model\ 1$

```
glBindVertexArray(vao[1]);
mat4 modelRotateMatrix = glm::rotate(mat4(), 0.2f, vec3(1, 0, 0));
//..................
```

$\Rightarrow model\ 2$

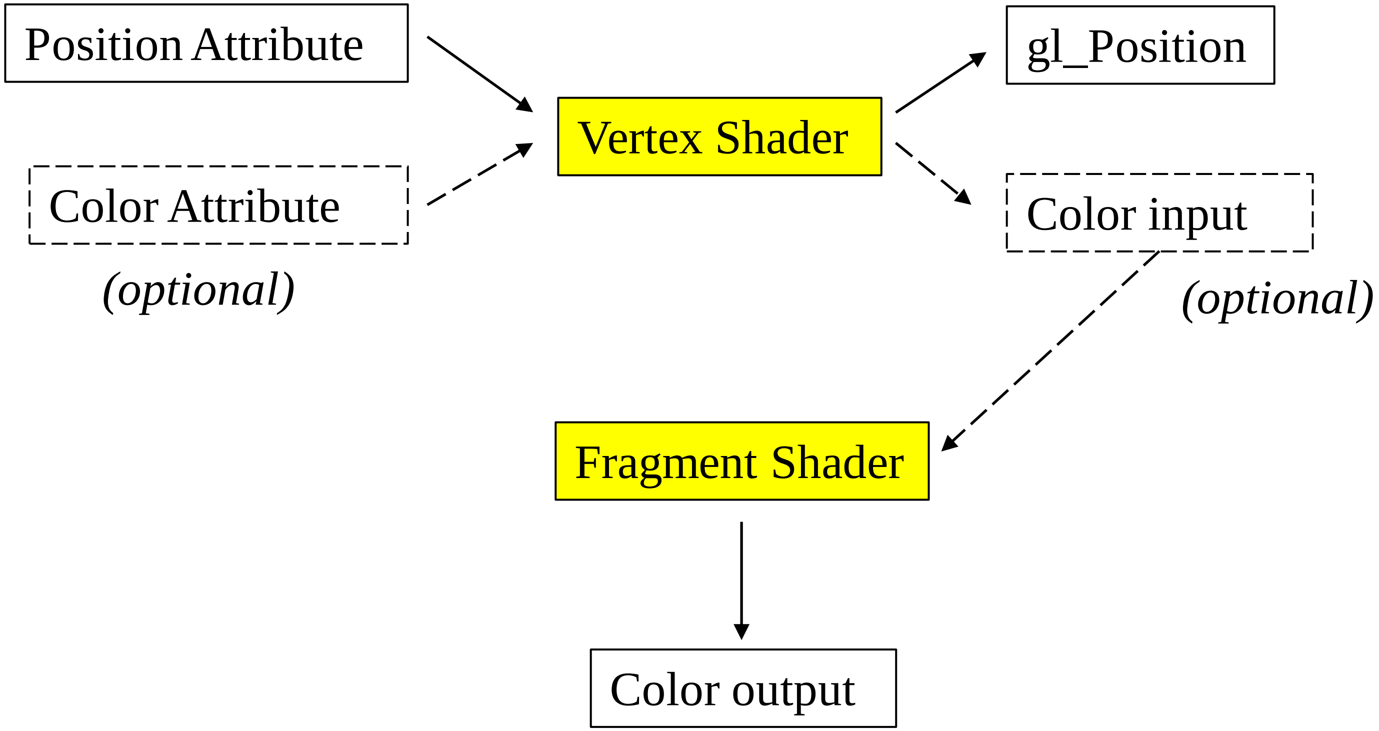
OpenGL is a state machine. If you don’t call *glBindVertexArray* to bind another model as the current used object, all the following commands will control the previous bound object.

# Problem 4

Problem 4 –
How to use vertex shader & fragment shader to transfer model attributes?

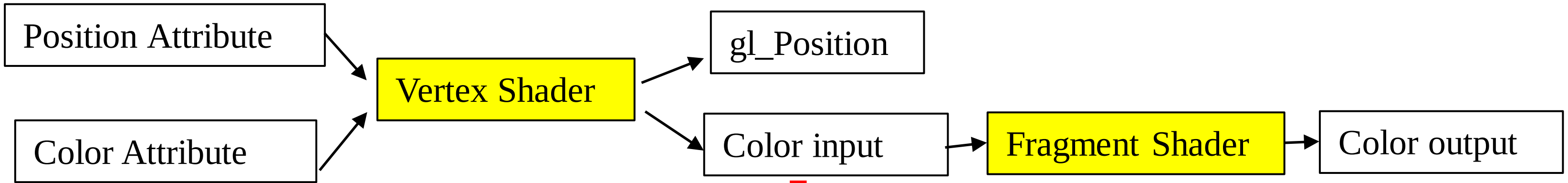
# Problem 4

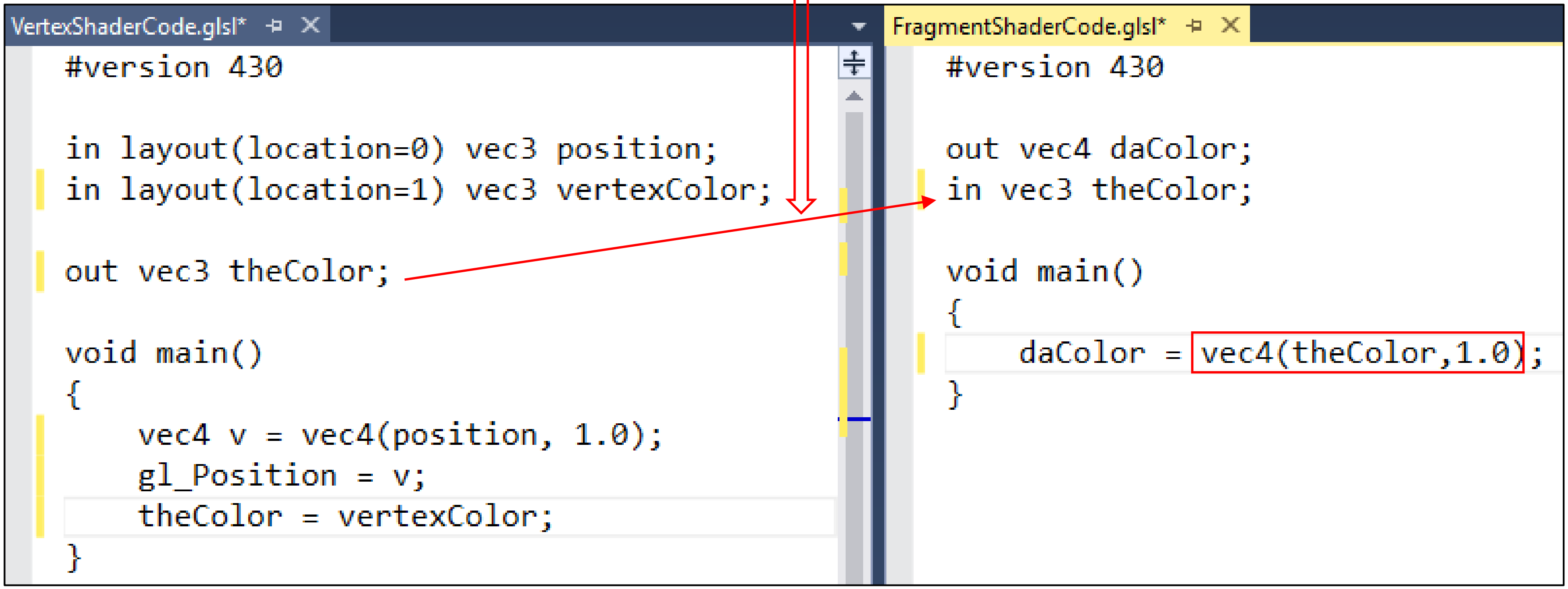
## Model attributes transfer in Shaders:



# Problem 4

## Code examples:



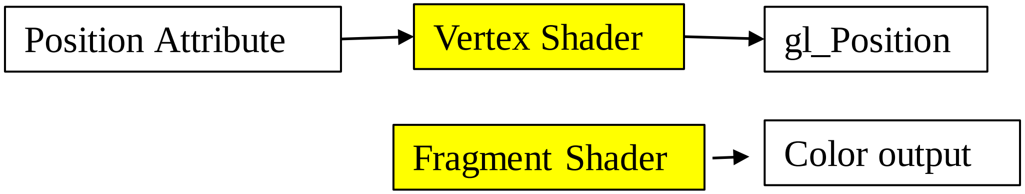


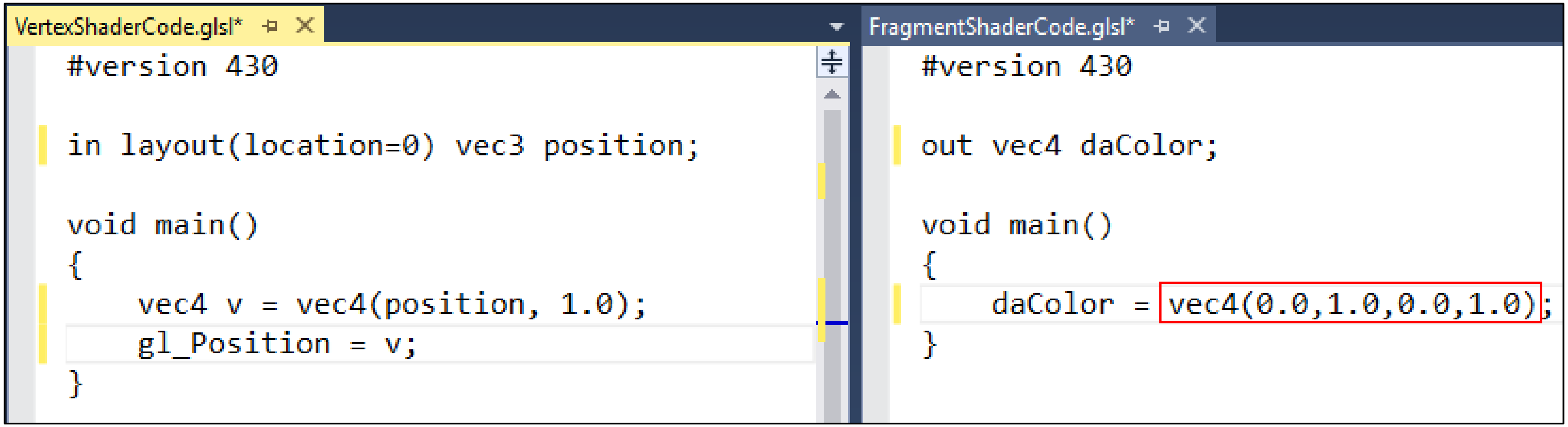
```
#version 430

in layout(location=0) vec3 position;
in layout(location=1) vec3 vertexColor;

out vec3 theColor;

void main()
{
    vec4 v = vec4(position, 1.0);
    gl_Position = v;
    theColor = vertexColor;
}
```

```
#version 430

out vec4 daColor;
in vec3 theColor;

void main()
{
    daColor = vec4(theColor,1.0);
}
```

# Problem 4

## Code examples:





```
#version 430

in layout(location=0) vec3 position;

void main()
{
    vec4 v = vec4(position, 1.0);
    gl_Position = v;
}
```

```
#version 430

out vec4 daColor;

void main()
{
    daColor = vec4(0.0,1.0,0.0,1.0);
}
```

```
sendDataToOpenGL()
{
    model array 01;

          ┌ glGenVertexArrays();
    VAO  ─┤
          └ glBindVertexArray();

          ┌ glGenBuffers();
    VBO  ─┤  glBindBuffer();
          └ glBufferData();

    glEnableVertexAttribArray();
    glVertexAttribPointer();

    model array 02;
    VAO;
    VBO;
    glEnableVertexAttribArray();
    glVertexAttribPointer();

    model array 03;
    ……..
    ……..
}
```

```
paintGL()
{
    glGetUniformLocation(“modelMatrix”);
    // also for view & projection

    glBindVertexArray(01);
    modelMatrix = translate*rotate*scaling*…;
    viewMatrix = glm:lookAt ();
    projectionMatrix = glm:perspective();
    glUniformMatrix4fv(&modelMatrix);
    // also for view & projection
    glDrawArrays(); / glDrawElements();

    glBindVertexArray(02);
    modelMatrix = translate*rotate*scaling*…;
    // optional for view & projection
    glUniformMatrix4fv(& modelMatrix);
    // optional for view & projection
    glDrawArrays(); / glDrawElements();

    glBindVertexArray(03);
    ……..
    ……..
}
```

```
[VertexShader]

#version ***
…….
…….

Uniform modelMatrix;
Uniform viewMatrix;
Uniform projectionMatrix;

void main()
{
    gl_Position = …..;
}
```

```
[FragmentShader]

#version ***
…….
…….

void main()
{
    Color = …..;
}
```