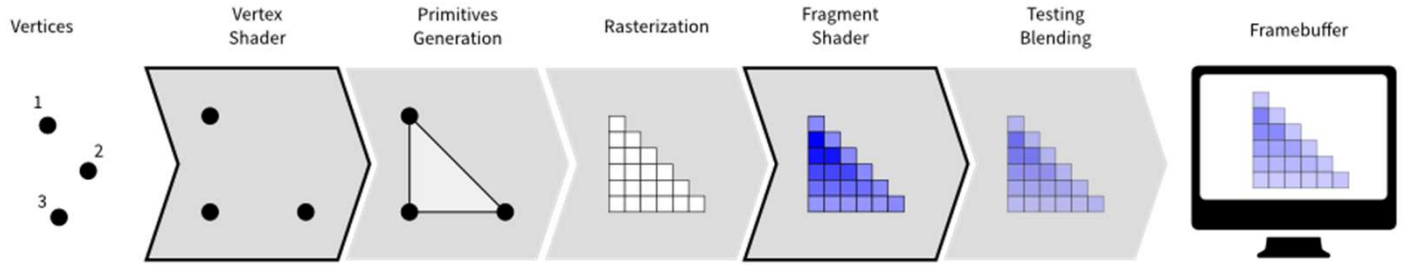
ΔΗΜΟΚΡΙΤΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΡΑΚΗΣ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Εισαγωγή στα Γραφικά Υπολογιστών

Εργαστήριο 2

1ο Εργαστήριο



1) Γεωμετρία κορυφών → Vertex Buffer Object
2) Κώδικας Vertex Shader → Compile
3) Κώδικας Fragment Shader → Compile
4) Shader Program → Linking

5) Αντιστοίχιση των μεταβλητών του application με τις μεταβλητές των Shaders
6) Rendering

Visual Studio

- Συντομεύσεις:

  Ctrl+K, Ctrl+F → Σωστή στοίχιση του επιλεγμένου κώδικα

  Ctrl+A → Επιλογή όλων

  Ctrl+K, Ctrl+C → Comment τις επιλεγμένες γραμμές

  Ctrl+K, Ctrl+U → Uncomment τις επιλεγμένες γραμμές

User Input

- Καταστάσεις ενός πλήκτρου: GLFW_PRESS, GLFW_RELEASE
- Συνάρτηση: glfwGetKey()

```
if(glfwGetKey(window, GLFW_KEY_ESCAPE) == GLFW_PRESS)
    glfwSetWindowShouldClose(window, true);
```

- Ονόματα των πλήκτρων στο http://www.glfw.org/docs/latest/group__keys.html

- Ενιαία συνάρτηση για όλα τα πλήκτρα:
  void processInput(GLFWwindow *window)

Wireframe Mode

- Για να δούμε καλύτερα τη δομή του πλέγματος, μπορούμε να αλλάξουμε το state, ώστε τα τρίγωνα να μην γεμίζουν με χρώμα:

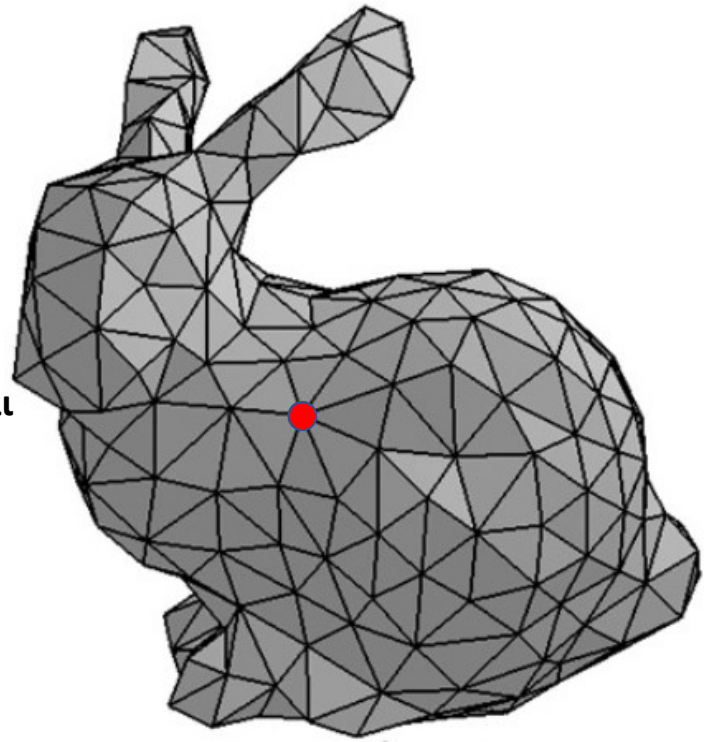
```
//wireframe mode
glPolygonMode(GL_FRONT_AND_BACK, GL_LINE)

//default
glPolygonMode(GL_FRONT_AND_BACK, GL_FILL)
```

Hello Rectangle

- Για να σχεδιάσουμε ένα ορθογώνιο, αρκεί να προσθέσουμε ένα ακόμα τρίγωνο:

```
GLfloat vertices[] =
{
        // first triangle
        0.5f, 0.5f, 0.0f,     // top right
        0.5f, -0.5f, 0.0f,    // bottom right
        -0.5f, 0.5f, 0.0f,    // top left
        // second triangle
        0.5f, -0.5f, 0.0f,    // bottom right
        -0.5f, -0.5f, 0.0f,   // bottom left
        -0.5f, 0.5f, 0.0f     // top left
};
```

επαναλαμβάνονται



Indexed Drawing

- Εναλλακτικός τρόπος rendering: **Indexed Drawing**

```
GLfloat  vertices[] =
{
0.5f, 0.5f, 0.0f,    // top right
0.5f, -0.5f, 0.0f,   // bottom right
-0.5f, -0.5f, 0.0f, // bottom left
-0.5f, 0.5f, 0.0f   // top left
};
```

```
GLuint indices[] =
{
0, 1, 3, // first triangle
1, 2, 3 // second triangle
};
```

Indexed Drawing

- Element Buffer Object:

```
//create the buffer ID, similar to VBOs
GLuint triangle_ebo;
glGenBuffers(1, &triangle_ebo);

//bind the EBO buffer to the context
glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, triangle_ebo);

// buffer the index data to the bound EBO
glBufferData(GL_ELEMENT_ARRAY_BUFFER, sizeof(indices), indices,
GL_STATIC_DRAW);
```

Indexed Drawing

- Όταν θέλουμε να σχεδιάσουμε το σχήμα στο rendering loop:

```
//bind the VBO with the unique vertices
glBindBuffer(GL_ARRAY_BUFFER, triangle_vbo);

//bind the corresponding EBO
glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, triangle_ebo);

//send the application data to the shaders (same code as before)

//glDrawElements() instead of glDrawArrays()
glDrawElements(GL_TRIANGLES, 6, GL_UNSIGNED_INT, 0);
```

glDrawElements

```
glDrawElements(GL_TRIANGLES, 6, GL_UNSIGNED_INT, 0);
```

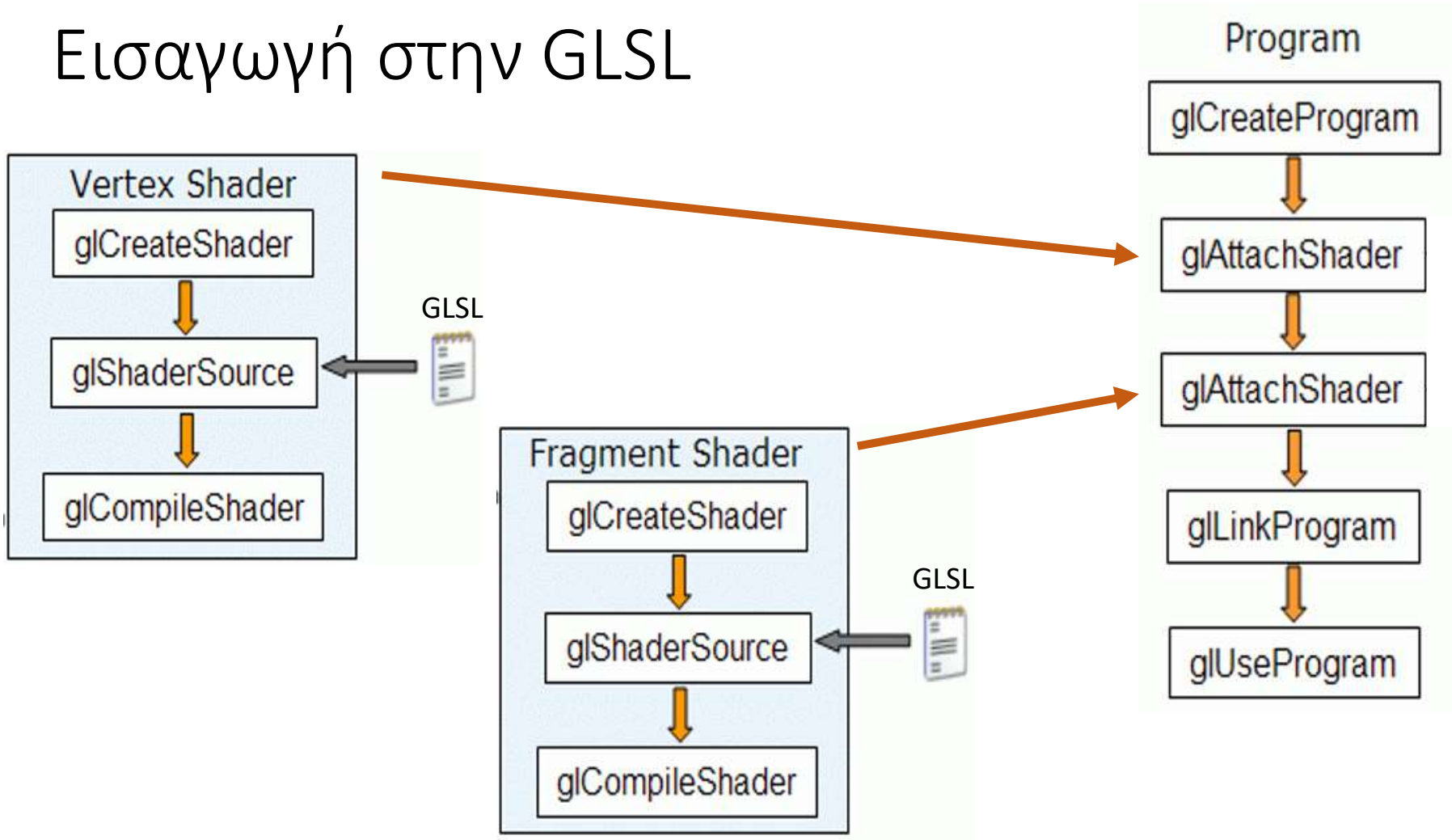
- 1ο: ο τύπος primitive
- 2ο: πόσα στοιχεία θέλουμε να σχεδιάσουμε
- 3ο: ο τύπος δεδομένων του πίνακα με τους δείκτες
- 4ο: ο πρώτος δείκτης του πίνακα με τους δείκτες (offset σε bytes)

glDrawArrays

glDrawArrays(GL_TRIANGLES, 0, 6);

- 1ο: ο τύπος primitive (πχ GL_TRIANGLES, GL_POINTS, GL_LINE_STRIP)
- 2ο: ο πρώτος δείκτης του πίνακα με τις κορυφές (offset ακέραιος)
- 3ο: πόσες κορυφές θέλουμε να σχεδιάσουμε

Εισαγωγή στην GLSL



Shader class

- Για να μην γράφουμε τους shaders στο πρόγραμμα, θα χρησιμοποιήσουμε μια κλάση που τους διαβάζει από αρχεία

- Η κλάση αυτή θα:
  - Διαβάζει τους shaders από αρχεία στο δίσκο,
  - Κάνει compile και link στο shader program,
  - Ελέγχει για λάθη

- Όλη η κλάση θα περιέχεται σε ένα αρχείο .h για λόγους ευκολίας και φορητότητας

Το InfoLog

- Δεν υπάρχουν ακόμα εργαλεία για debugging στους shaders, ούτε δυνατότητα εξόδου στο prompt με printf

- Τα αντικείμενα των shaders κατά το compile παράγουν ένα **info log** με τα λάθη που έχουν γίνει (αν υπάρχουν)

- Αντίστοιχα, το αντικείμενο του shader program παράγει **info log** για τα λάθη που έχουν γίνει κατά το linking

- Τα αποθηκεύουμε σε μεταβλητές και τα τυπώνουμε στο prompt

Εισαγωγή στην GLSL

- GLSL: OpenGL Shading Language
- Σύνταξη παρόμοια με τη C
- Ειδικοί τύποι δεδομένων για πράξεις με πίνακες και διανύσματα
- Σύνταξη:

```
#version …
Declaration of variables

void main()
{
    …
}
```

GLSL

Τύποι Δεδομένων:

- Όπως στην C: **float**, **bool**, **int**

- Διανύσματα με 2,3 ή 4 στοιχεία:

Παραδείγματα: **vec4** (4 floats), **ivec3** (3 integers)

- Τετράγωνοι πίνακες με 2x2, 3x3 ή 4x4 floats:

Παραδείγματα: **mat3**, **mat4**

Swizzling

- Τα στοιχεία των διανυσμάτων είναι διαθέσιμα σαν μεταβλητές αντικειμένου:

  vec4 vertexA → vertexA[0] → vertex.x

  vertexA[1] → vertex.y

  (κ.ο.κ. για τα vertex.z και vertex.w)

- Για ευκολία, μπορούμε να έχουμε πρόσβαση με τα γράμματα xyzw, ή rgba (για χρώματα), ή stpq (για υφές)

- Επίσης, μπορούμε να χρησιμοποιήσουμε πάνω από ένα γράμμα:

  vec3 vertexB = vertexA.xyz

- Και μπορούμε να αναμείξουμε τα γράμματα:

  vec2 vertexC = vertexB.yx και vec4 vertexD = vertexA.zxzz

Άσκηση

- Πώς μπορούμε να αναποδογυρίσουμε το τρίγωνο, αλλάζοντας μόνο τον κώδικα του vertex shader?

Qualifiers

in:

- Μεταβλητή εισόδου
- Υπάρχει και για τους vertex και για τους fragment shaders

out:

- Μεταβλητή εξόδου
- Υπάρχει και για τους vertex και για τους fragment shaders

uniform:

- Μεταβλητή που αλλάζει ανά rendering call
- Υπάρχει και για τους vertex και για τους fragment shaders

To qualifier ‘uniform’

- Fragment shader:

```
#version 330 core
uniform vec4 color;

void main()
{
    gl_FragColor = color;
}
```

Το qualifier ‘uniform’

- Στο rendering loop:

```
GLint colorLocation = glGetUniformLocation(shaderProgram, "color");
glUniform4f(colorLocation, 0.7f, 1.0f, 0.5f, 1.0f);
```

- Διαφορετικές περιπτώσεις της glUniform.., ανάλογα με τον τύπο της μεταβλητής:

https://www.khronos.org/registry/OpenGL-Refpages/gl4/html/glUniform.xhtml

Άσκηση

- Η μεταβλητή uniform είναι ίδια κατά τη διάρκεια μιας κλήσης, αλλά μπορεί να αλλάζει από κλήση σε κλήση:

```
float timeValue = glfwGetTime();
float colorValue = (sin(timeValue) / 2.0f) + 0.5f;

glUniform4f(colorLocation, 0.0f, colorValue, 0.0f, 1.0f);
```

Μετακίνηση Μεταβλητών από τον vertex στον fragment shader

- Vertex shader:

```
#version 330 core
layout (location = 0) in vec3 in_position;
out vec4 color;

void main()
{
   gl_Position = vec4(in_position.x, in_position.y, in_position.z, 1.0);
   color = vec4(0.5, 0.4, 0.7, 1.0);
}
```

Μετακίνηση Μεταβλητών από τον vertex στον fragment shader

- Fragment shader:

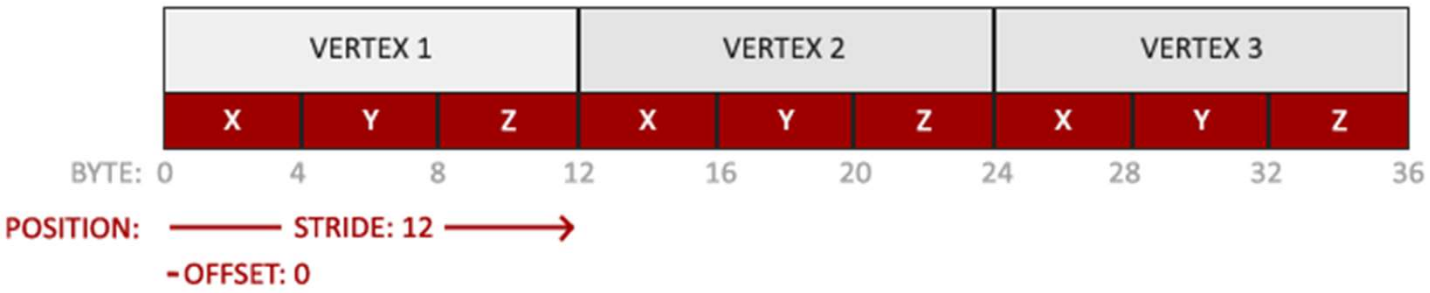
```
#version 330 core
in vec4 color;

void main()
{
    gl_FragColor = color;
}
```

Άσκηση

- Σχεδιάστε το σχήμα έτσι ώστε κάθε κορυφή να έχει σαν χρώμα το διάνυσμα που αντιστοιχεί στις συντεταγμένες της.

To qualifier ‘in’ vertex shader

- Μέχρι στιγμής, VBO με ένα attribute αποτελούμενο από 3 floats:



- Vertex position: μέγεθος→3 floats, stride→3 floats, offset→0

```
glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 3 * sizeof(GL_FLOAT), (void*)0);
glEnableVertexAttribArray(0);
```

To qualifier ‘in’ vertex shader

- Μπορούμε να έχουμε παραπάνω πληροφορία ανά κορυφή:

```
GLfloat vertices[] =
{
// positions        // colors
0.5f, -0.5f, 0.0f, 1.0f, 0.0f, 0.0f, // bottom right, red
-0.5f, -0.5f, 0.0f, 1.0f, 1.0f, 0.0f, // bottom left, yellow
0.0f, 0.5f, 0.0f, 0.0f, 0.0f, 1.0f // top, blue
};
```

To qualifier ‘in’ vertex shader

- Vertex shader:

```
#version 330 core
layout (location = 0) in vec3 in_position;
layout (location = 1) in vec3 in_color;
out vec3 color;

void main()
{
	gl_Position = vec4(in_position, 1.0);
	color = in_color;
}
```

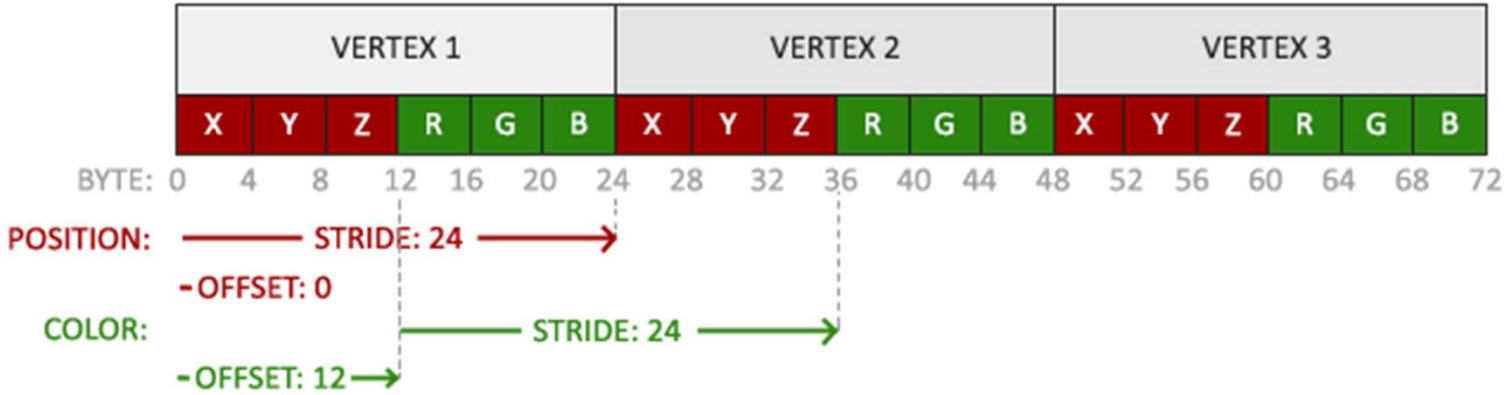
To qualifier ‘in’ fragment shader

- Fragment shader

```
#version 330 core
in vec3 color;
void main()
{
    gl_FragColor = vec4(color, 1.0f);
}
```

Το qualifier ‘in’ vertex shader

- Τα δύο attributes είναι αποθηκευμένα στο VBO με αυτόν τον τρόπο:



- Vertex position: μέγεθος→3 floats, stride→6 floats, offset→0
- Vertex color: μέγεθος→3 floats, stride→6 floats, offset→3 floats

To qualifier ‘in’ vertex shader

```
// position attribute
glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 6*sizeof(float), (void*)0);
glEnableVertexAttribArray(0);

// color attribute
glVertexAttribPointer(1, 3, GL_FLOAT, GL_FALSE, 6*sizeof(float), (void*)(3* sizeof(float)));
glEnableVertexAttribArray(1);
```

Άσκηση

- Αλλάξτε τον τρόπο που διαβάζεται το VBO ώστε οι τρεις πρώτες θέσεις να θέτουν το χρώμα του τριγώνου και οι επόμενες τρεις τη θέση του τριγώνου.