

Auxiliar N°2

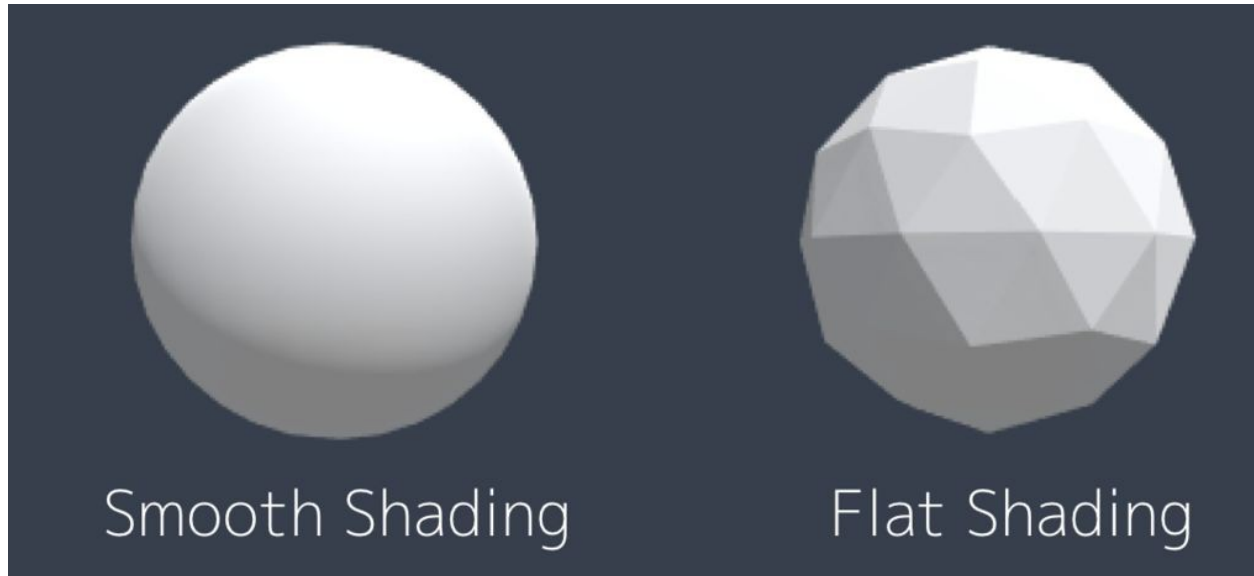
Shaders

Auxiliar: Pablo Pizarro R. [@ppizarror](#)
Estudiante MSc. Ingeniería Estructural
Universidad de Chile

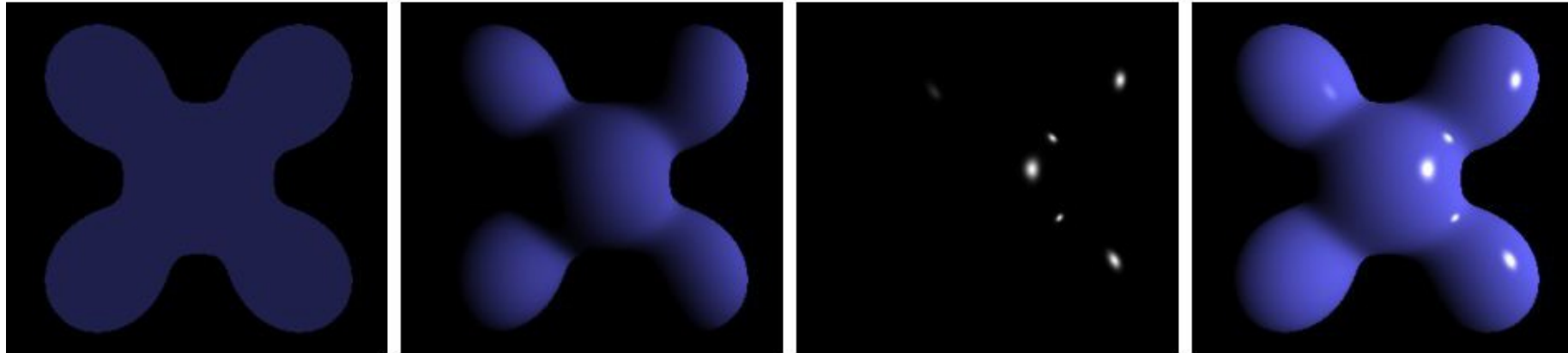
Introducción

- Concepto de shaders
- Tipos de shaders
- Pipeline de OpenGL
- Código en OpenGL

Shaders



Phong shading



Ambient

+

Diffuse

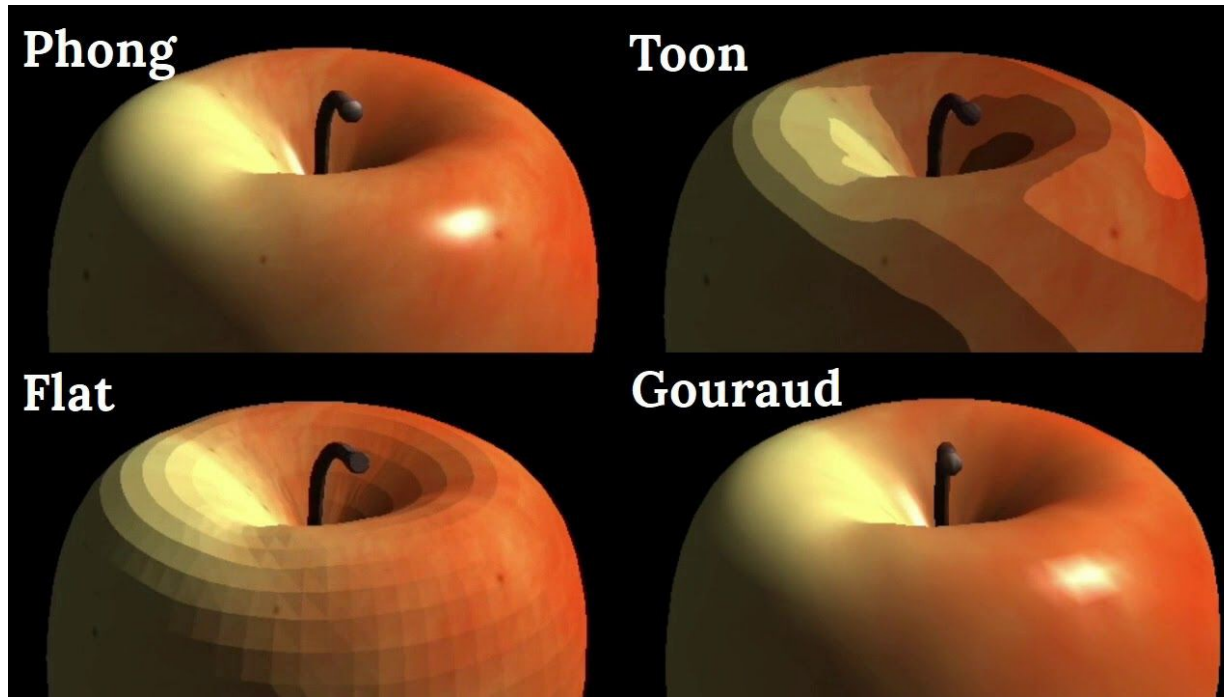
+

Specular

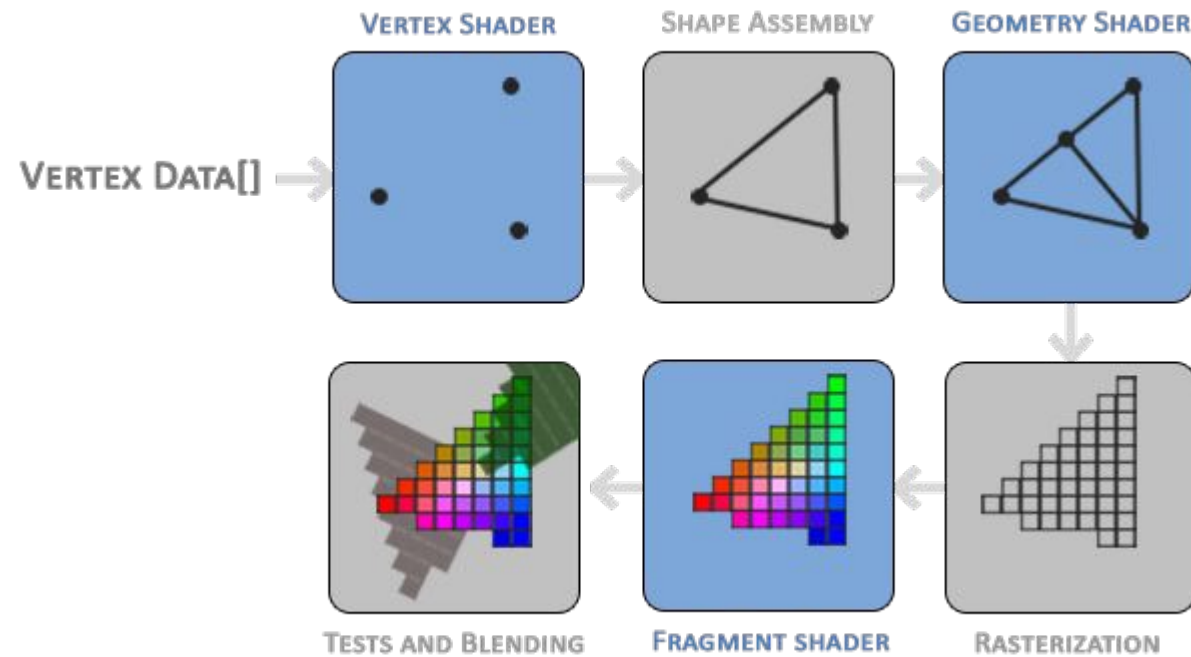
=

Phong Reflection

Gouraud shading



Rendering pipeline



Rendering pipeline



Vertex shader runs
once for every vertex

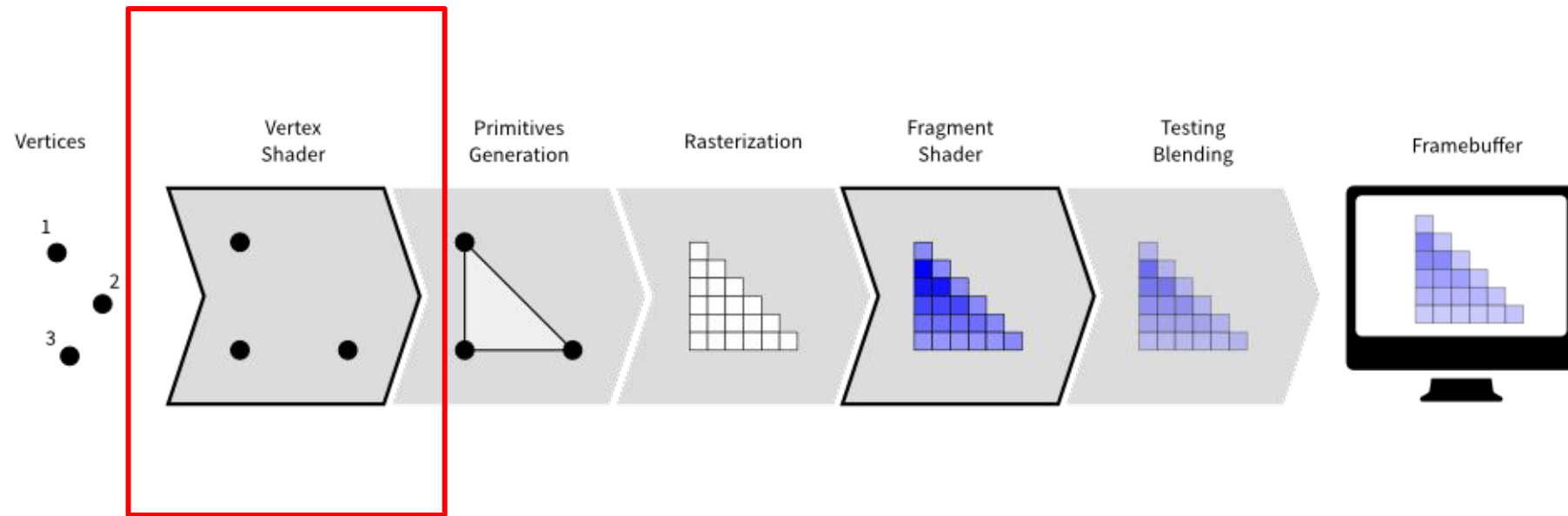


Fragment shader runs
once for every pixel
rendered in the scene,
with vertex data
interpolated

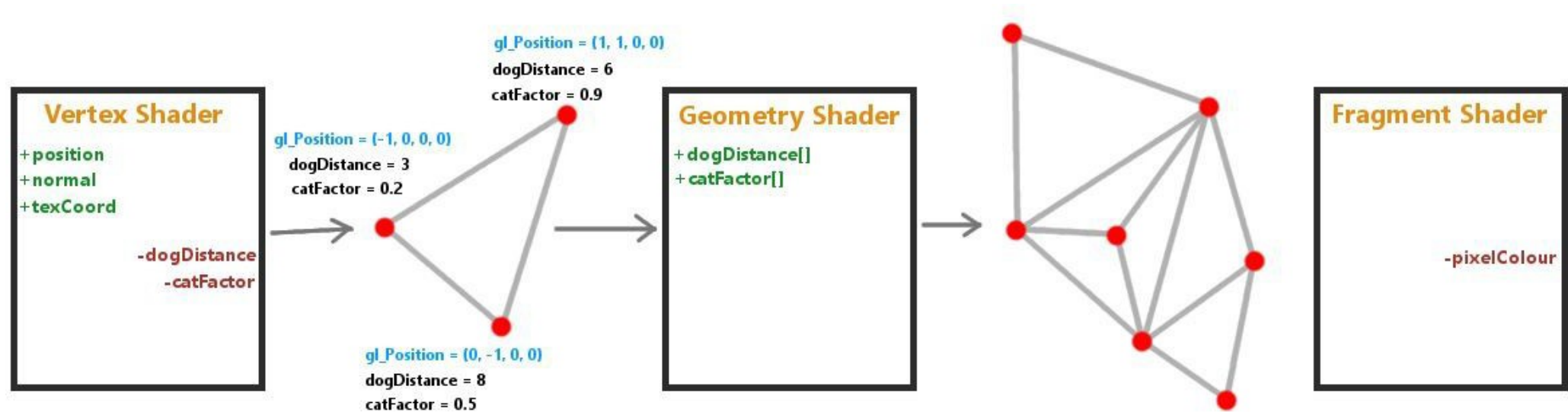
Tipos de shaders

- 2D
 - Fragment/Pixel Shader
- 3D
 - Vertex Shader
 - Geometry Shader
 - Tessellation Shader

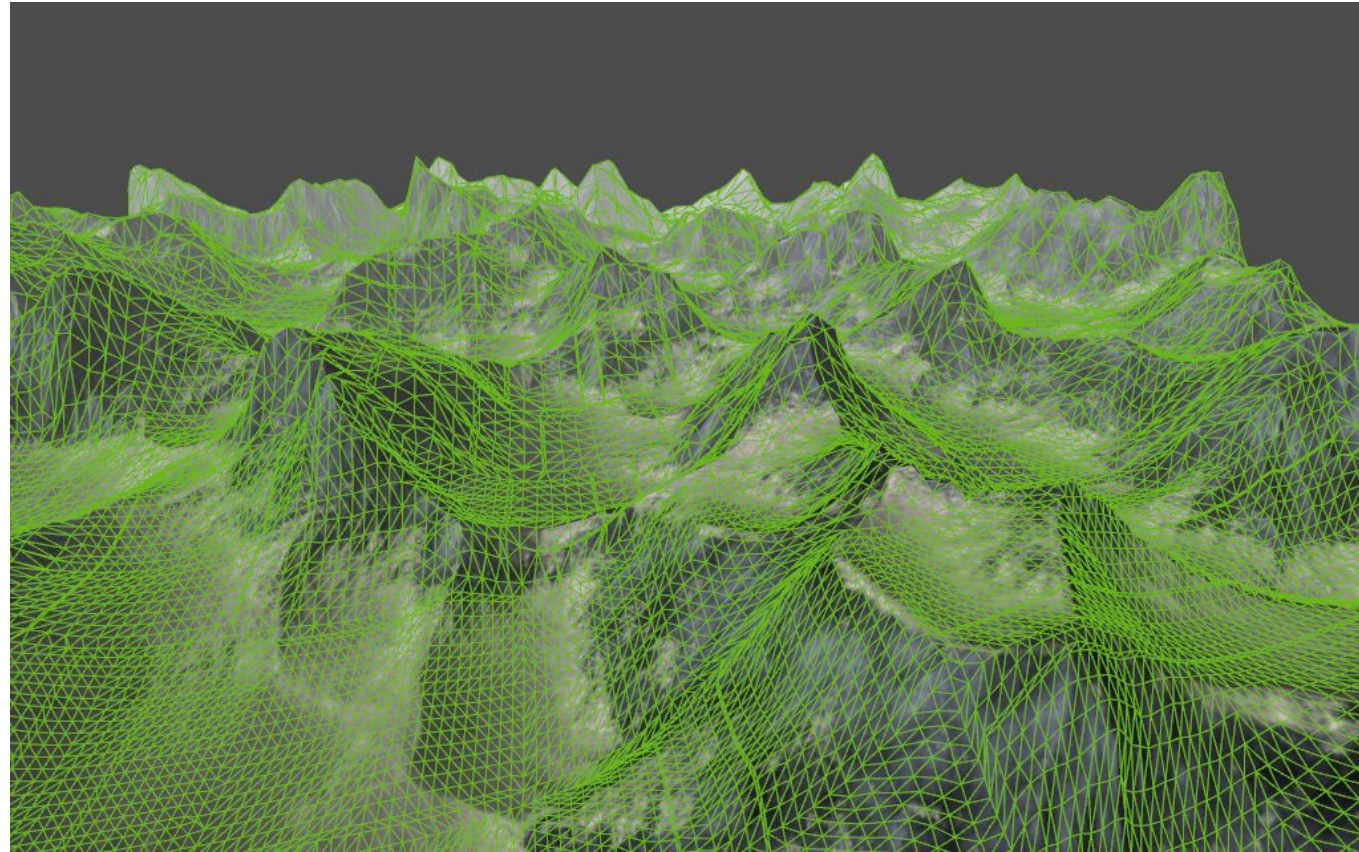
Vertex shader



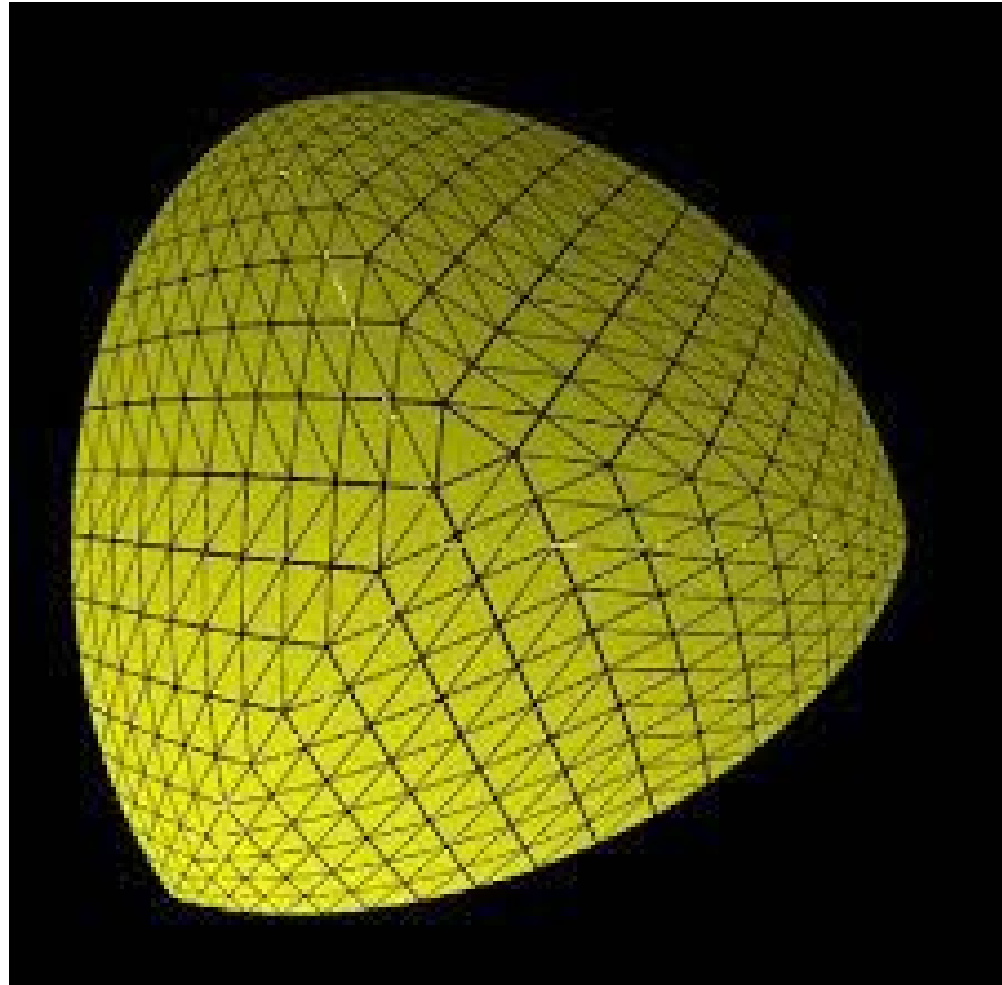
Geometry shader



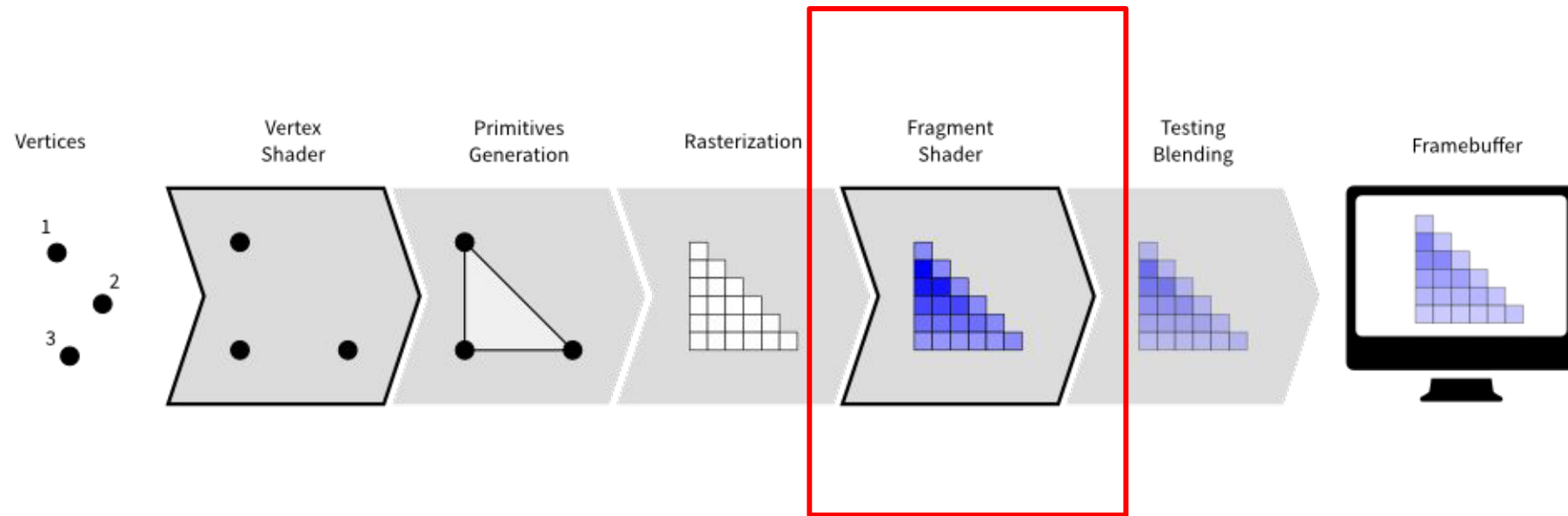
Tessellation Shader



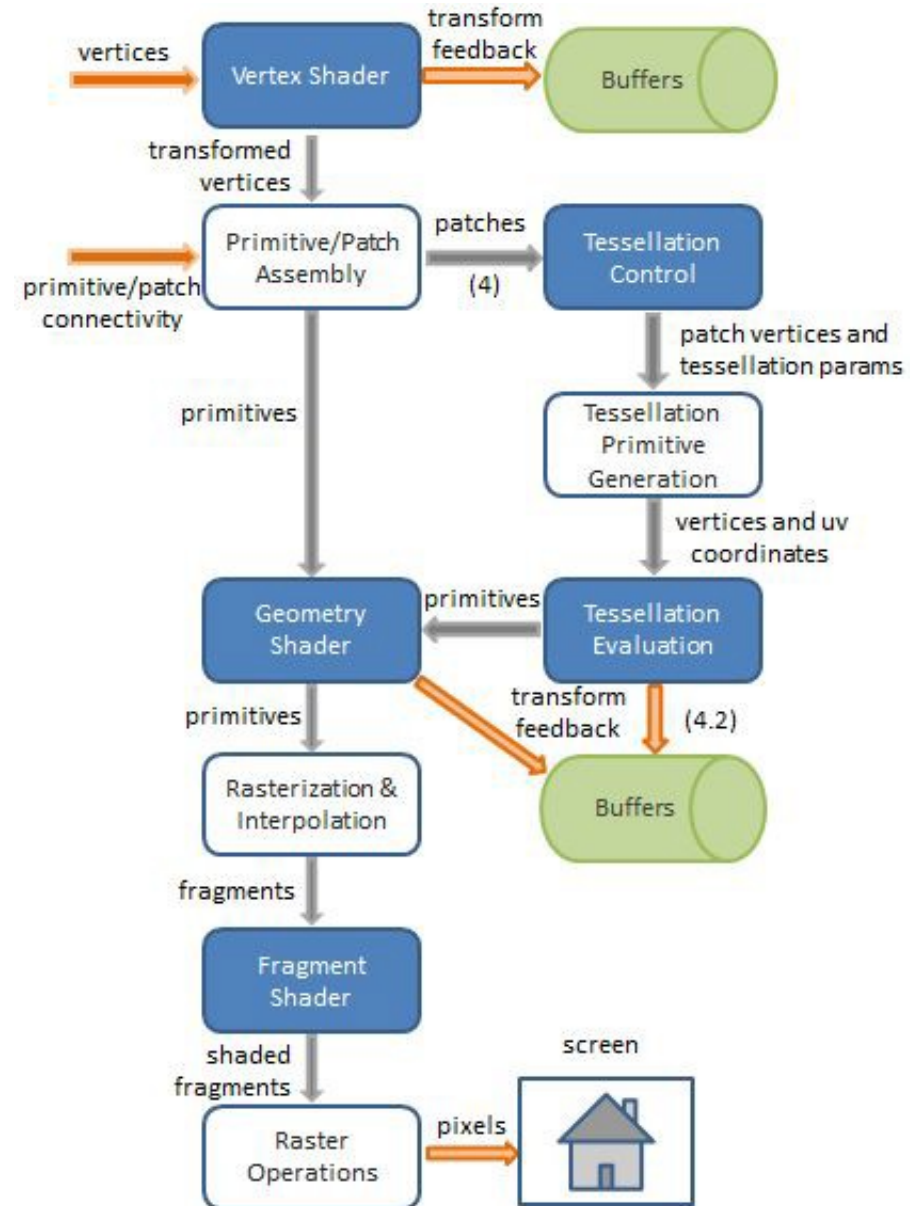
Tessellation Shader



Fragment shader

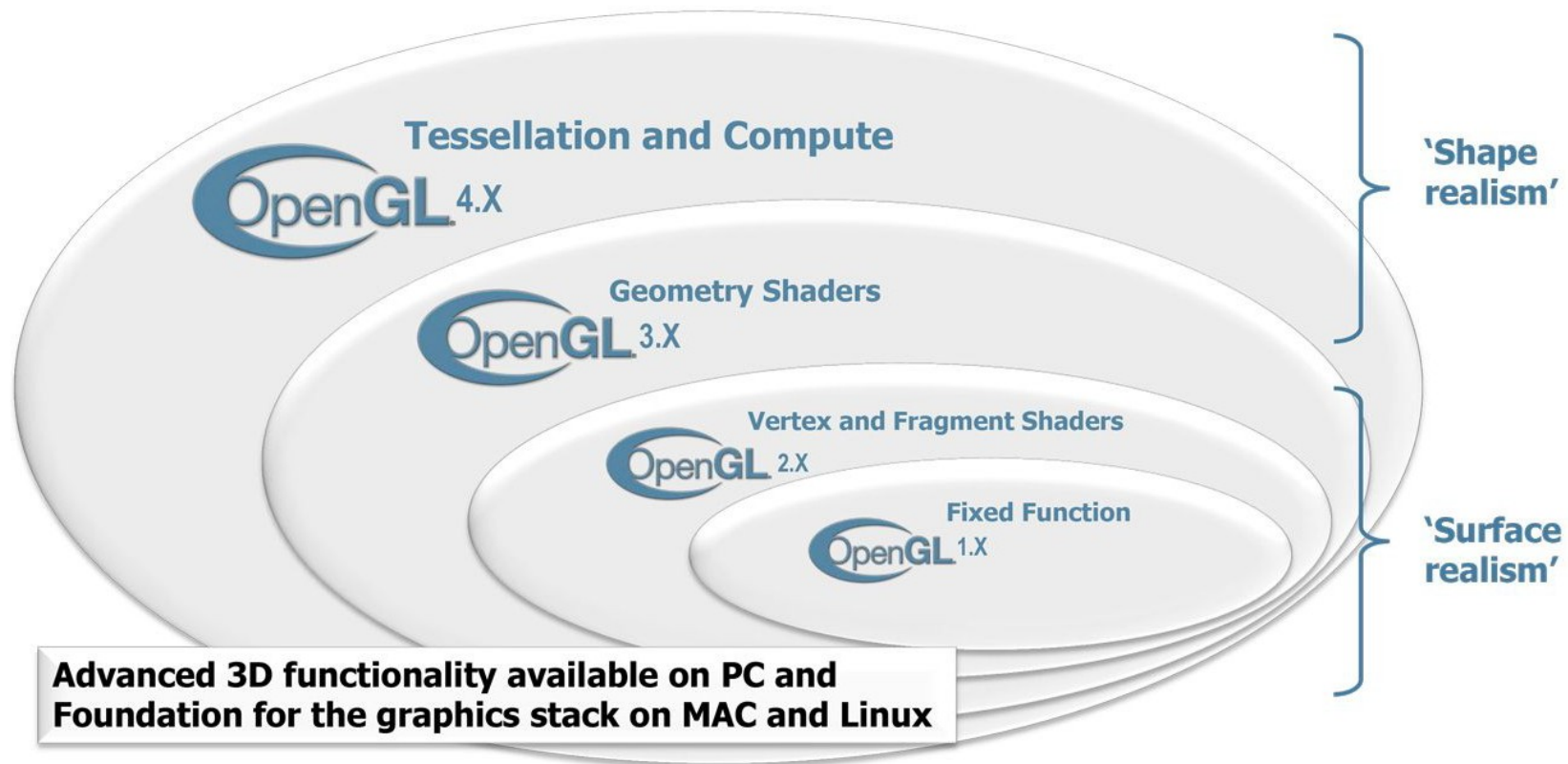


Nuevo pipeline

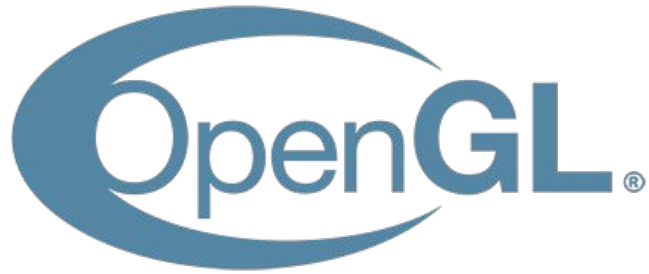


Evolución de OpenGL

OpenGL for Each Hardware Generation



Alternativas a OpenGL



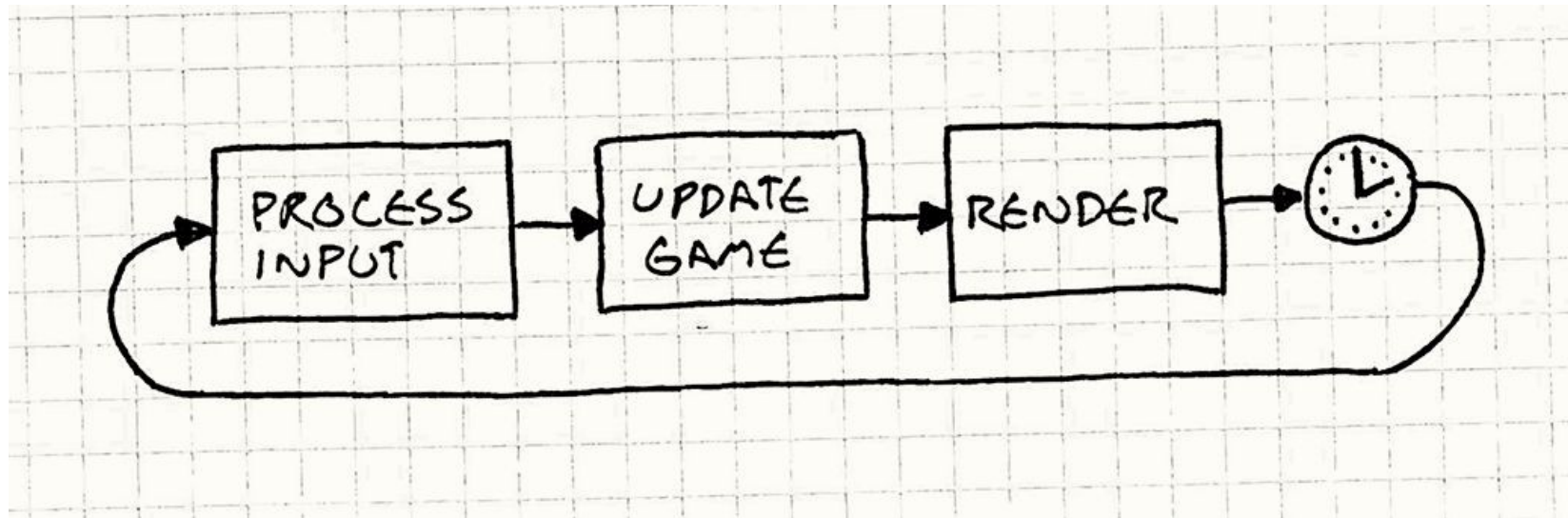
PyOpenGL 3.1.0



OpenGL



OpenGL – En Juegos



Programar en OpenGL

Inicialización

```
glfwWindowHint(GLFW_SAMPLES, 4);           // 4x antialiasing
glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, 3); // Define la versión de OpenGL
glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, 3);
glfwWindowHint(GLFW_OPENGL_FORWARD_COMPAT, GL_TRUE); // Compatibilidad MacOS
glfwWindowHint(GLFW_OPENGL_PROFILE, GLFW_OPENGL_CORE_PROFILE); // No queremos el viejo OpenGL
```

Programar en OpenGL

Inicialización

```
// Abre una ventana y crea el contexto de OpenGL
GLFWwindow *window;
// Variable global, para simplicidad
window = glfwCreateWindow(1024, 768, "Tutorial 01", NULL, NULL);
if (window == NULL)
{
    fprintf(stderr, "Failed to open GLFW window.");
    glfwTerminate();
    return -1;
}
```

Programar en OpenGL

Inicialización

```
glfwMakeContextCurrent(window); // Inicializa GLEW
glewExperimental = true;        // Se requiere en el core
if (glewInit() != GLEW_OK)
{
    fprintf(stderr, "Failed to initialize GLEW\n");
    return -1;
}
```

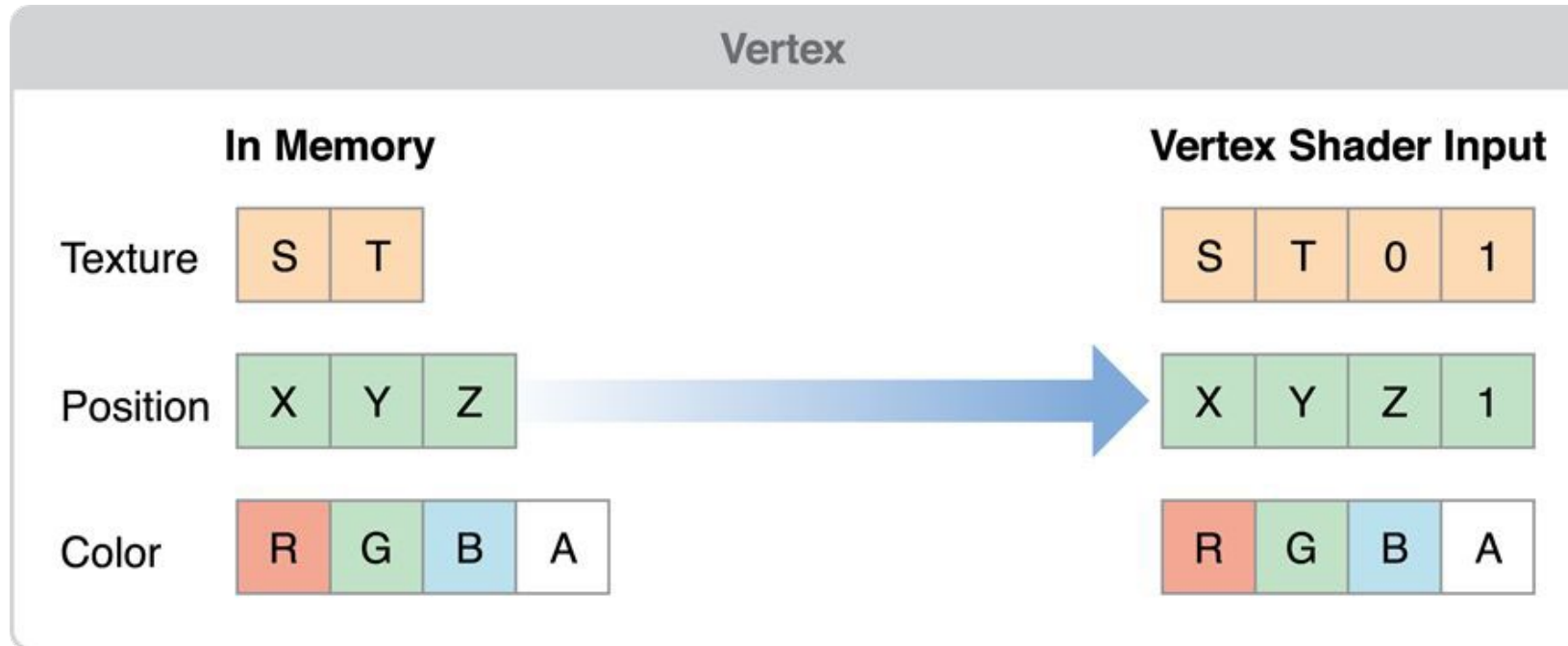
Programar en OpenGL

Window loop

```
// Aseguramos poder capturar los eventos del teclado presionados
glfwSetInputMode(window, GLFW_STICKY_KEYS, GL_TRUE);
do
{
    // Borra la pantalla
    glClear(GL_COLOR_BUFFER_BIT);
    // Swap buffers
    glfwSwapBuffers(window);
    glfwPollEvents();
} // Chequea que la tecla ESC haya sido presionada, si pasa eso se cierra la ventana
while (glfwGetKey(window, GLFW_KEY_ESCAPE) != GLFW_PRESS &&
        glfwWindowShouldClose(window) == 0);
```

Programar en OpenGL

VAO (Vertex Array Object)



Programar en OpenGL

VAO (Vertex Array Object)

```
// Identifica el vertex buffer
GLuint vertexbuffer;
// Genera 1 buffer, pone el identificador resultante en vertexbuffer
glGenBuffers(1, &vertexbuffer);
// Los siguientes comandos se comunicarán con 'vertexbuffer'
glBindBuffer(GL_ARRAY_BUFFER, vertexbuffer);
// Da los vértices a OpenGL
glBufferData(GL_ARRAY_BUFFER, sizeof(g_vertex_buffer_data),
             g_vertex_buffer_data, GL_STATIC_DRAW);
```

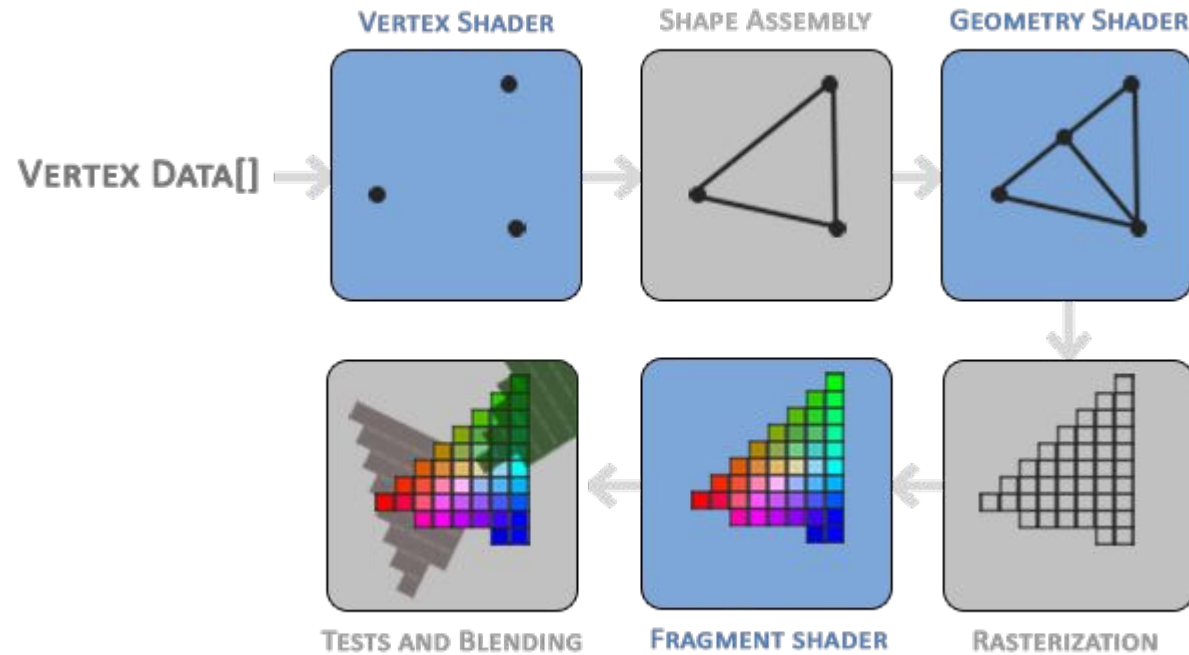

Programar en OpenGL

VAO (Vertex Array Object)

```
// 1º atributo del buffer: vértices
glEnableVertexAttribArray(0);
glBindBuffer(GL_ARRAY_BUFFER, vertexbuffer);
glVertexAttribPointer(
    0, // Atributo N°0
    3, // Tamaño
    GL_FLOAT, // Tipo
    GL_FALSE, // Normalizado?
    0, // Stride
    (void*)0 // Array buffer offset
);
// Dibuja el triángulo
glDrawArrays(GL_TRIANGLES, 0, 3);
// Partiendo desde el vértice 0, 3 vértices totales -> 1 triángulo
glDisableVertexAttribArray(0);
```

Programar en OpenGL

Volvamos al pipeline

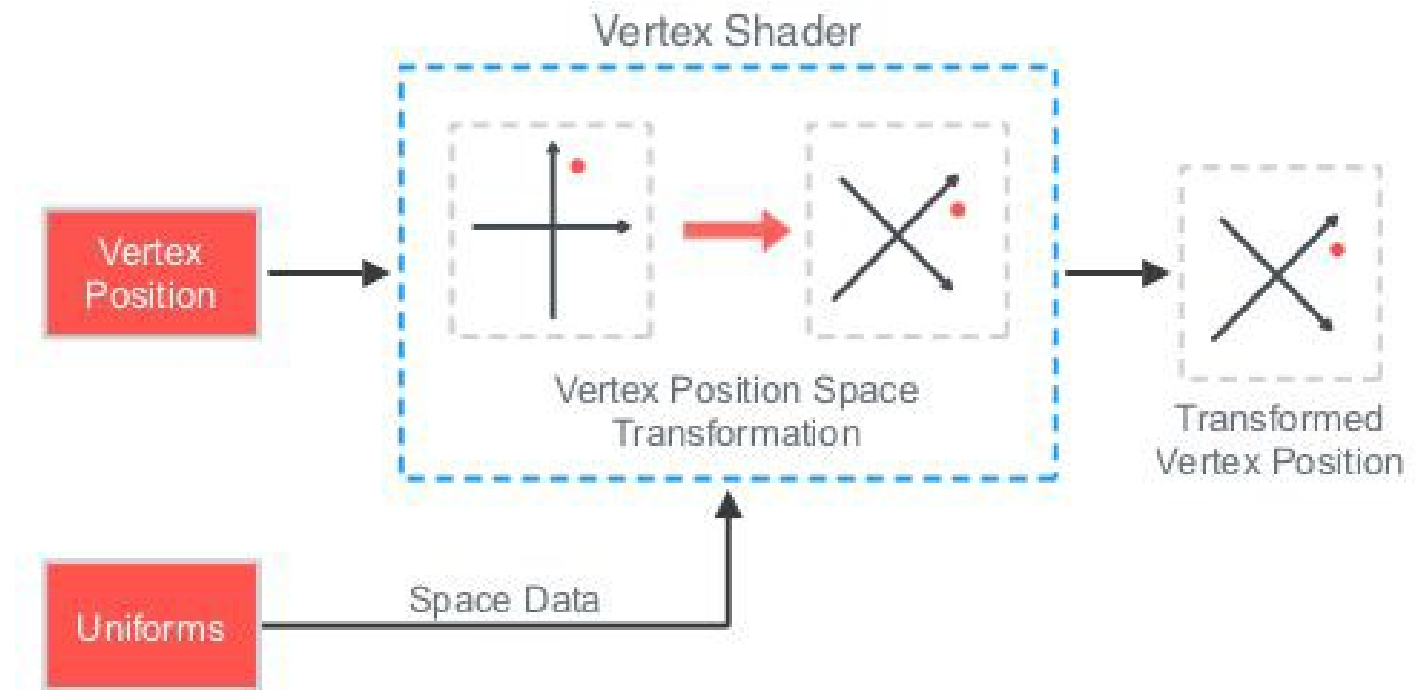


Programar en OpenGL

Volvamos al pipeline

El shader se ejecutará por cada vértice.

Los shaders se programan con el lenguaje GLSL.



Programar en OpenGL

Vertex Shader – Código del Shader

```
#version 330 core
// Input vertex data, diferente por cada ejecución de este shader
layout(location = 0) in vec3 vertexPosition_modelspace;
void main() {
    gl_Position.xyz = vertexPosition_modelspace;
    gl_Position.w = 1.0;
}
```

Programar en OpenGL

Vertex Shader – Carga

```
// Crea el shader
GLuint VertexShaderID = glCreateShader(GL_VERTEX_SHADER);
// Lee el código desde un archivo
std::string VertexShaderCode;
std::ifstream VertexShaderStream(vertex_file_path, std::ios::in);
if (VertexShaderStream.is_open()) {
    std::stringstream sstr;
    sstr << VertexShaderStream.rdbuf();
    VertexShaderCode = sstr.str();
    VertexShaderStream.close();
}
```

Programar en OpenGL

Vertex Shader - Compilación

```
// Compila el shader
printf("Compiling shader : %s\n", vertex_file_path);
char const * VertexSourcePointer = VertexShaderCode.c_str();
glShaderSource(VertexShaderID, 1, &VertexSourcePointer, NULL);
glCompileShader(VertexShaderID);
```

Programar en OpenGL

Vertex Shader - Chequeo

```
// Chequeo del shader
glGetShaderiv(VertexShaderID, GL_COMPILE_STATUS, &Result);
glGetShaderiv(VertexShaderID, GL_INFO_LOG_LENGTH, &InfoLogLength);
if (InfoLogLength > 0) {
    std::vector<char> VertexShaderErrorMessage(InfoLogLength + 1);
    glGetShaderInfoLog(
        VertexShaderID,
        InfoLogLength,
        NULL,
        &VertexShaderErrorMessage[0]
    );
    printf("%s\n", &VertexShaderErrorMessage[0]);
}
```

Programar en OpenGL

Vertex Shader - Link

```
// Link the program
printf("Linking program\n");
GLuint ProgramID = glCreateProgram();
glAttachShader(ProgramID, VertexShaderID);
glLinkProgram(ProgramID);
```

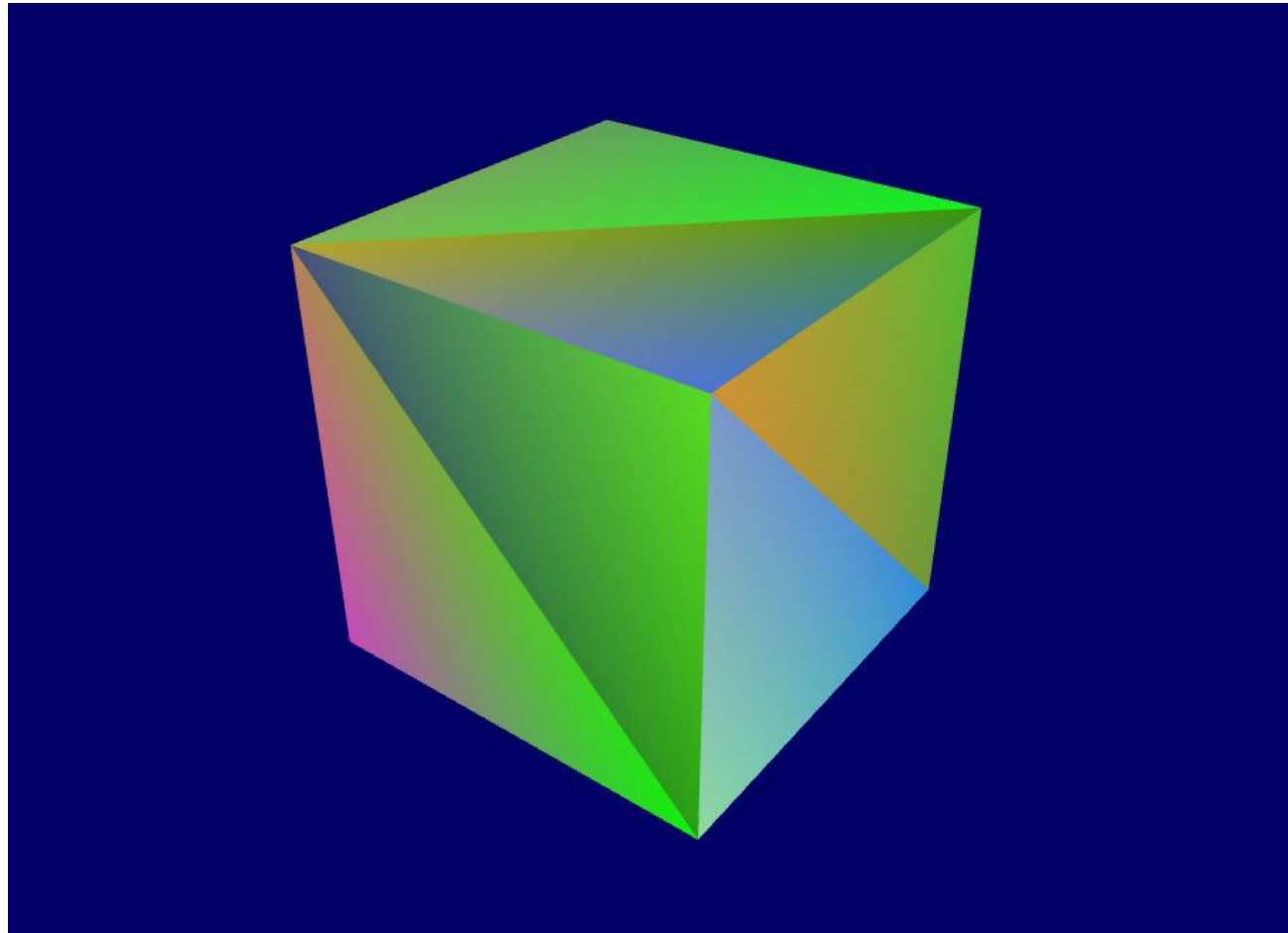

Programar en OpenGL

Fragment shader

```
#version 330 core
out vec3 color;
void main() {
    color = vec3(1, 0, 0);
}
```

Programar en OpenGL

Fragment shader



Programar en OpenGL

Nuevo buffer con colores

```
GLuint colorbuffer;
glGenBuffers(1, &colorbuffer);
glBindBuffer(GL_ARRAY_BUFFER, colorbuffer);
glBufferData(GL_ARRAY_BUFFER, sizeof(g_color_buffer_data), g_color_buffer_data, GL_STATIC_DRAW);
// 2nd attribute buffer : colores
glEnableVertexAttribArray(1);
glBindBuffer(GL_ARRAY_BUFFER, colorbuffer);
glVertexAttribPointer(
    1, // Atributo N°1
    shader.
    3, // size
    GL_FLOAT, // type
    GL_FALSE, // normalized?
    0, // stride
    (void*)0 // array buffer offset
);
```

Programar en OpenGL

Usar Z-buffer

```
// Activamos depth test  
glEnable(GL_DEPTH_TEST);
```

```
// Activa profundidad entre la cámara y las superficies dibujadas con el fragment  
glDepthFunc(GL_LESS);
```

```
// Borra la pantalla  
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
```

Programar en OpenGL

Vertex Shader (Cubo)

```
#version 330 core
// Input del vertex, recordar las ubicaciones (0:posición, 1:colores)
layout(location = 0) in vec3 vertexPosition_modelspace;
layout(location = 1) in vec3 vertexColor;
// Output, lo que recibirá el fragment shader
out vec3 fragmentColor;
// Valores constantes para el mesh
uniform mat4 MVP;

void main() {
    // Posición del vértice en la ventana de clipping
    gl_Position = MVP * vec4(vertexPosition_modelspace, 1);

    // El color de cada vértice será interpolado para producir el color en el fragment
    fragmentColor = vertexColor;
}
```

Programar en OpenGL

Fragment Shader (Cubo)

```
#version 330 core
// Color interpolado desde el vertex shader
in vec3 fragmentColor;
// Output data
out vec3 color;
void main() {
    // Color (output), se interpolará por los
    // tres vértices de cada cara
    color = fragmentColor;
}
```

Programar en OpenGL

Cómo cargar un modelo - OBJ



Programar en OpenGL

Cómo cargar un modelo - OBJ

List of geometric vertices, with (x, y, z [,w]) coordinates, w is optional and defaults to 1.0.

v 0.123 0.234 0.345 1.0

v ...

...

List of texture coordinates, in (u, v [,w]) coordinates, these will vary between 0 and 1, w is optional and defaults to 0.

vt 0.500 1 [0]

vt ...

...

List of vertex normals in (x,y,z) form; normals might not be unit vectors.

vn 0.707 0.000 0.707

vn ...

...

Parameter space vertices in (u [,v] [,w]) form; free form geometry statement (see below)

vp 0.310000 3.210000 2.100000

vp ...

...

Polygonal face element (see below)

f 1 2 3

f 3/1 4/2 5/3

f 6/4/1 3/5/3 7/6/5

f 7//1 8//2 9//3

f ...

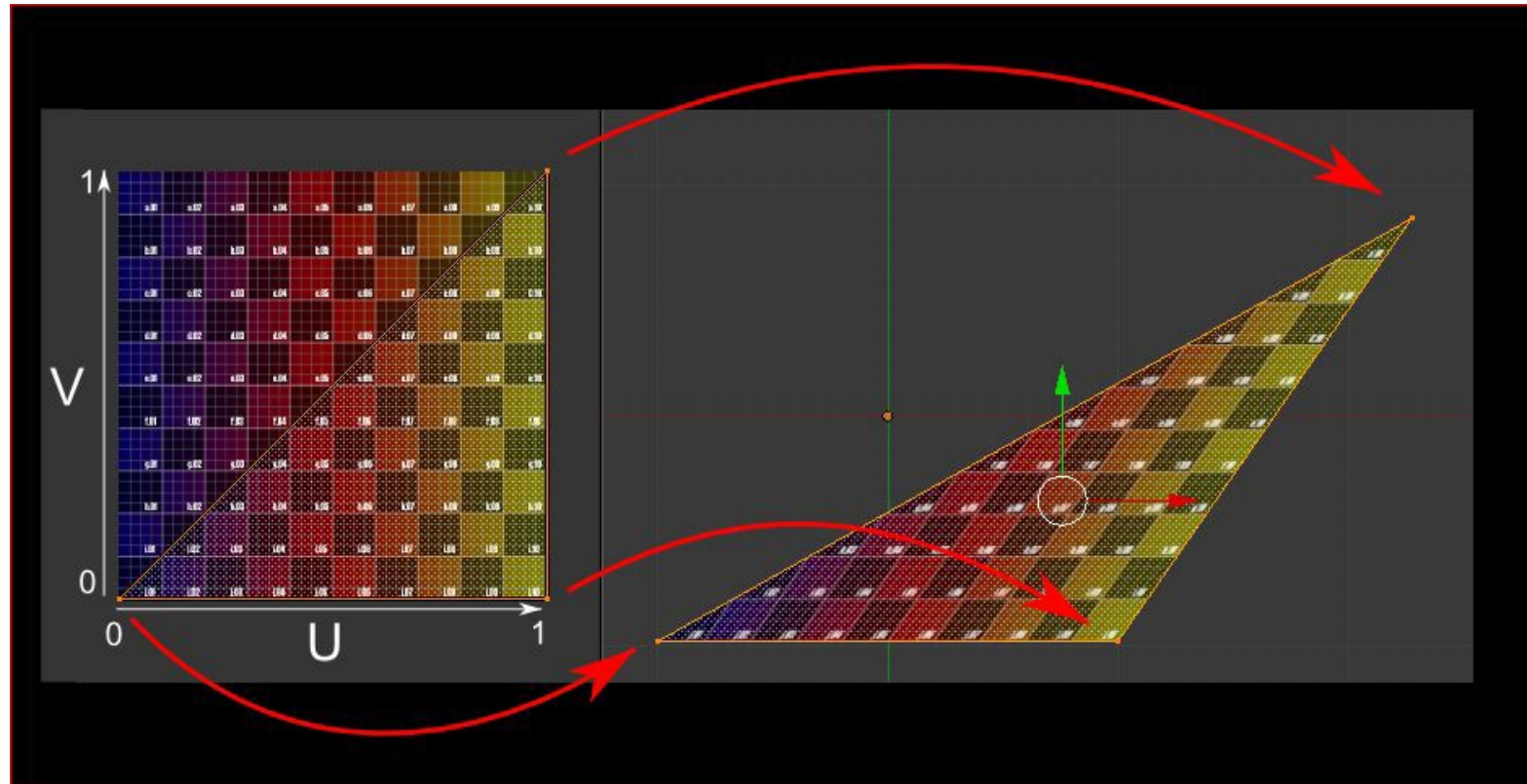
...

Line element (see below)

l 5 8 1 2 4 9

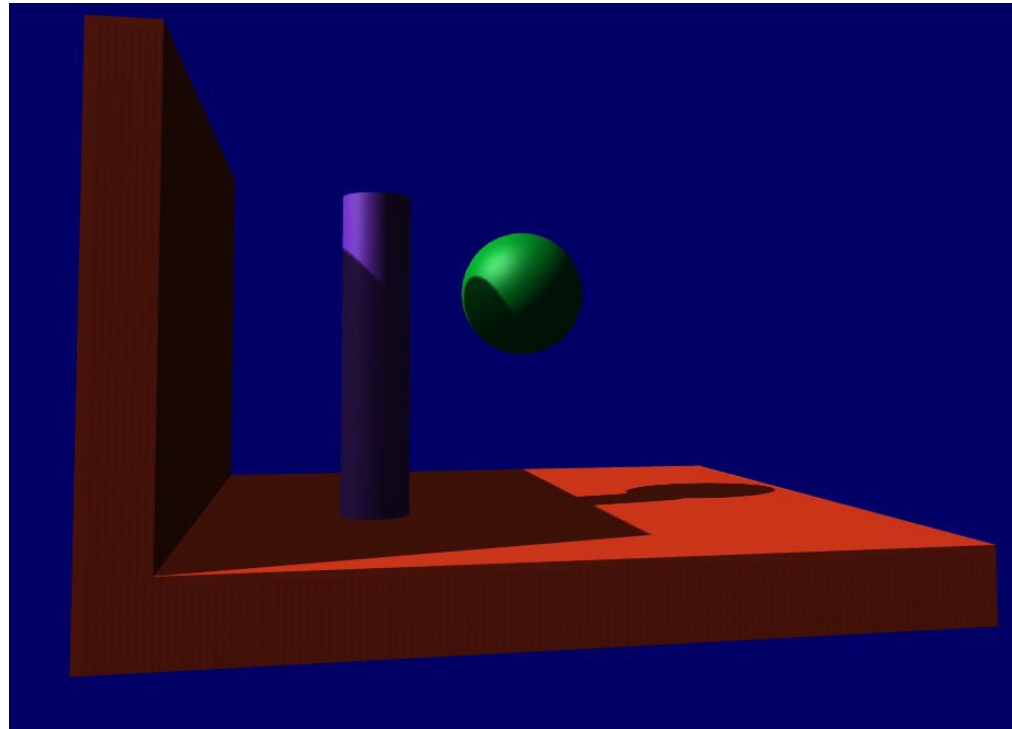
Programar en OpenGL

Otros conceptos – Coordenadas UV



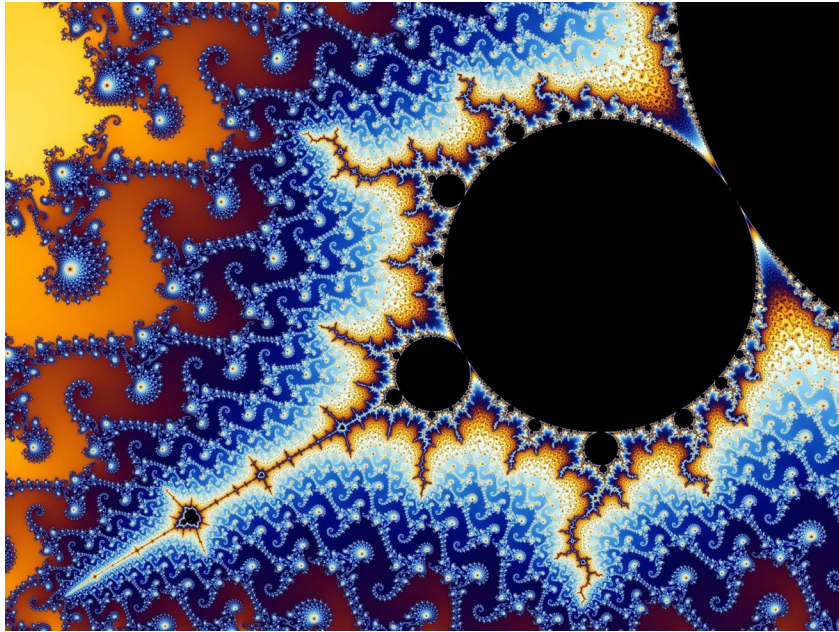
Programar en OpenGL

Ejemplos



Programar en OpenGL

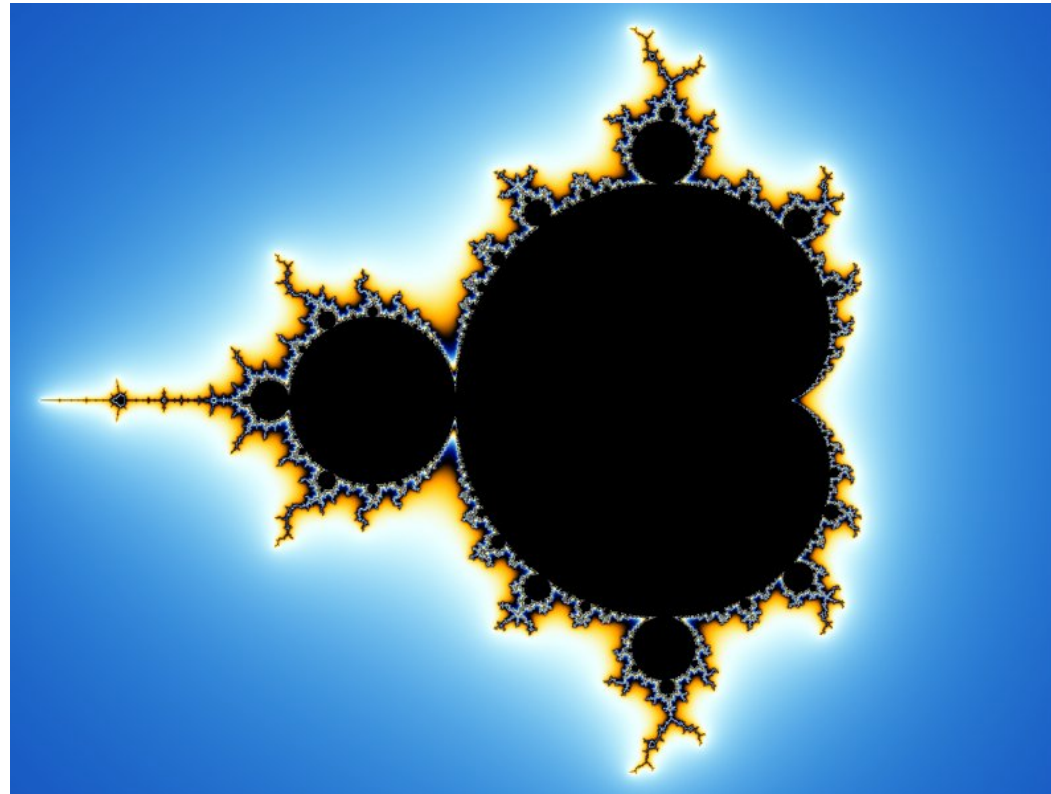
Ejemplos



Programar en OpenGL

Ejemplos – Tarea N°2 2018

Construir un fractal utilizando shaders



Programar en OpenGL

Ejemplos – Tarea N°2 2018

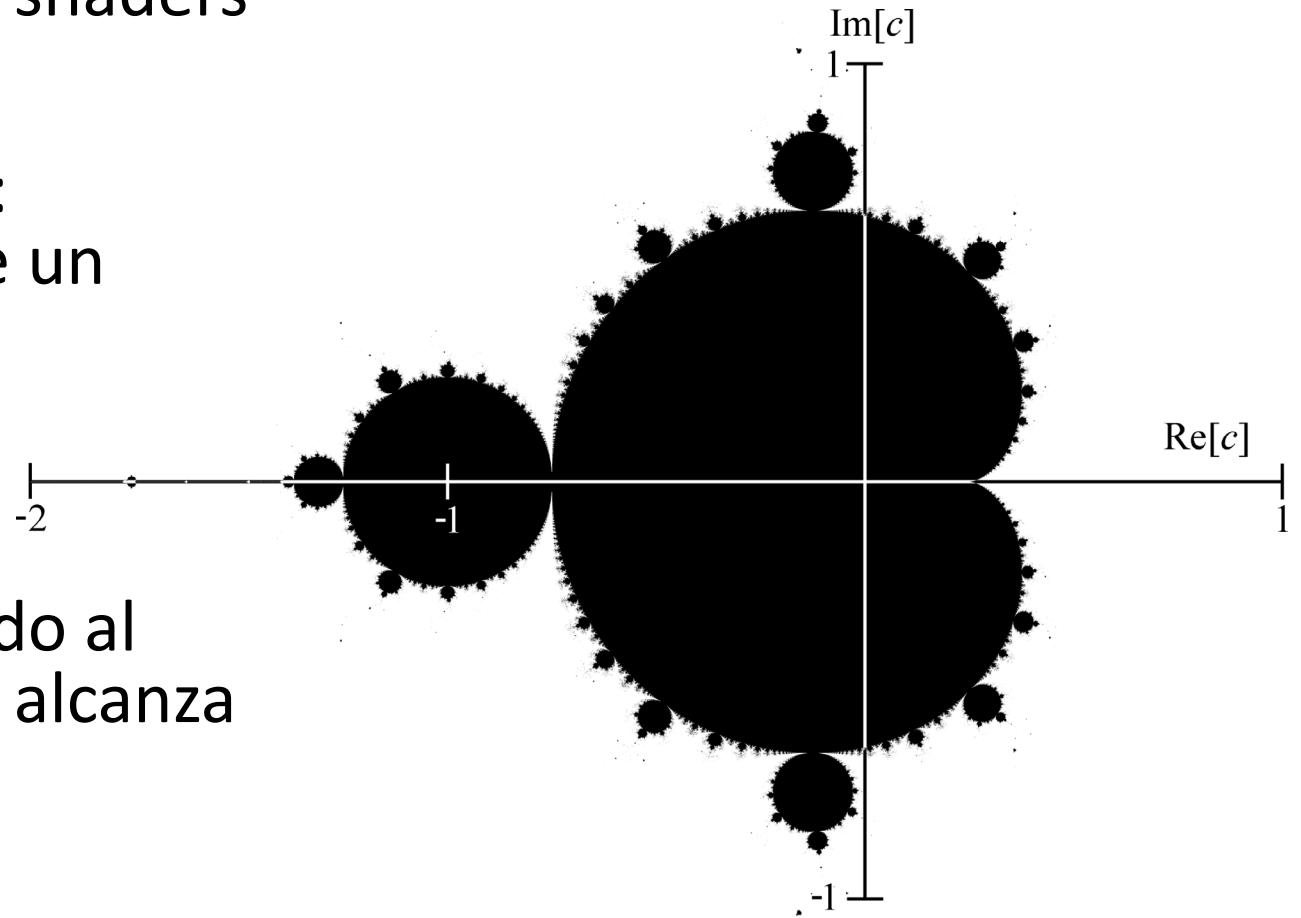
Construir un fractal utilizando shaders

¿Ideas?

Funcionamiento de un fractal:
Ecuación de recurrencia sobre un plano

$$\begin{cases} z_0 = 0 \in \mathbb{C} & (\text{término inicial}) \\ z_{n+1} = z_n^2 + c & (\text{sucesión recursiva}) \end{cases}$$

El color del fractal está asociado al número de iteraciones que se alcanza antes de diverger

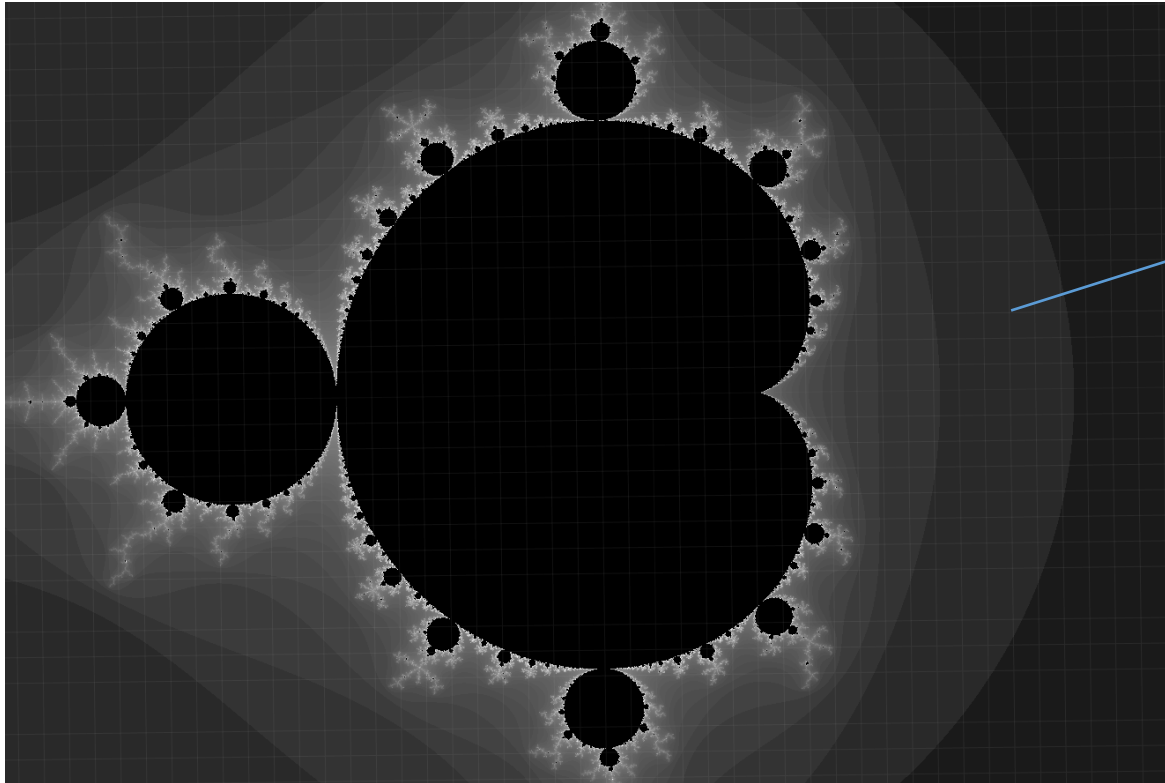


Programar en OpenGL

Ejemplos – Tarea N°2 2018

- ¿Cuáles son nuestros vértices?

R: Modelar un plano cartesiano como una grilla (plano), cada intersección de la grilla es un vértice



Un punto (x, y) . X: Parte real, Y: parte imaginaria

Programar en OpenGL

Ejemplos – Tarea N°2 2018

- ¿Cuáles son nuestros vértices?
- ¿Cómo modelamos los colores?
R: Fragment shader

Programar en OpenGL

Ejemplos – Tarea N°2 – Vertex Shader

```
/*
MANDELBROT
VERTEX SHADER

Ejecuta mandelbrot,  $C = c_r + i*c_i$  se pasa por
cada (x,y) del plano complejo.

@author Pablo Pizarro R. @ppizarror.com
@license MIT
@since 0.1.0
*/

// Activa precisión alta
#ifdef GL_FRAGMENT_PRECISION_HIGH
precision highp float;
#else
precision mediump float;
#endif
precision mediump int;
```

```
// (x,y) de cada valor del plano
attribute float vertex_z_r;
attribute float vertex_z_i;

varying float c_r;
varying float c_i;

void main() {

// El valor complejo corresponde a (x,y)
    c_r = vertex_z_r;
    c_i = vertex_z_i;

    gl_Position = projectionMatrix *
modelViewMatrix * vec4(position, 1.0);

}
```


Programar en OpenGL

Ejemplos – Tarea N°2 – Fragment Shader

```
/*
MANDELBROT
FRAGMENT SHADER

Ejecuta mandelbrot,  $C = c_r + i*c_i$  se pasa
por cada (x,y) del plano complejo.

@author Pablo Pizarro R. @ppizarror.com
@license MIT
@since 0.1.0
*/

// Activa precisión alta
#ifdef GL_FRAGMENT_PRECISION_HIGH
precision highp float;
#else
precision mediump float;
#endif

precision mediump int;

// Pasa C de cada (x,y)
varying float c_r;
varying float c_i;

// Interacciones máximas
uniform int max_iterations;

// Rango de colores
uniform float r_min;
uniform float r_max;
uniform float g_min;
uniform float g_max;
uniform float b_min;
uniform float b_max;
```

Programar en OpenGL

Ejemplos – Tarea N°2 – Fragment Shader

```
// Inicio del shader
void main() {
    float r;
    float g;
    float b;
    float t;
    float w_r;
    float w_i;
    float u;
    float v;

    w_r = 0.0;
    w_i = 0.0;

    // Si converge es negro
    r = 0.0;
    g = 0.0;
    b = 0.0;

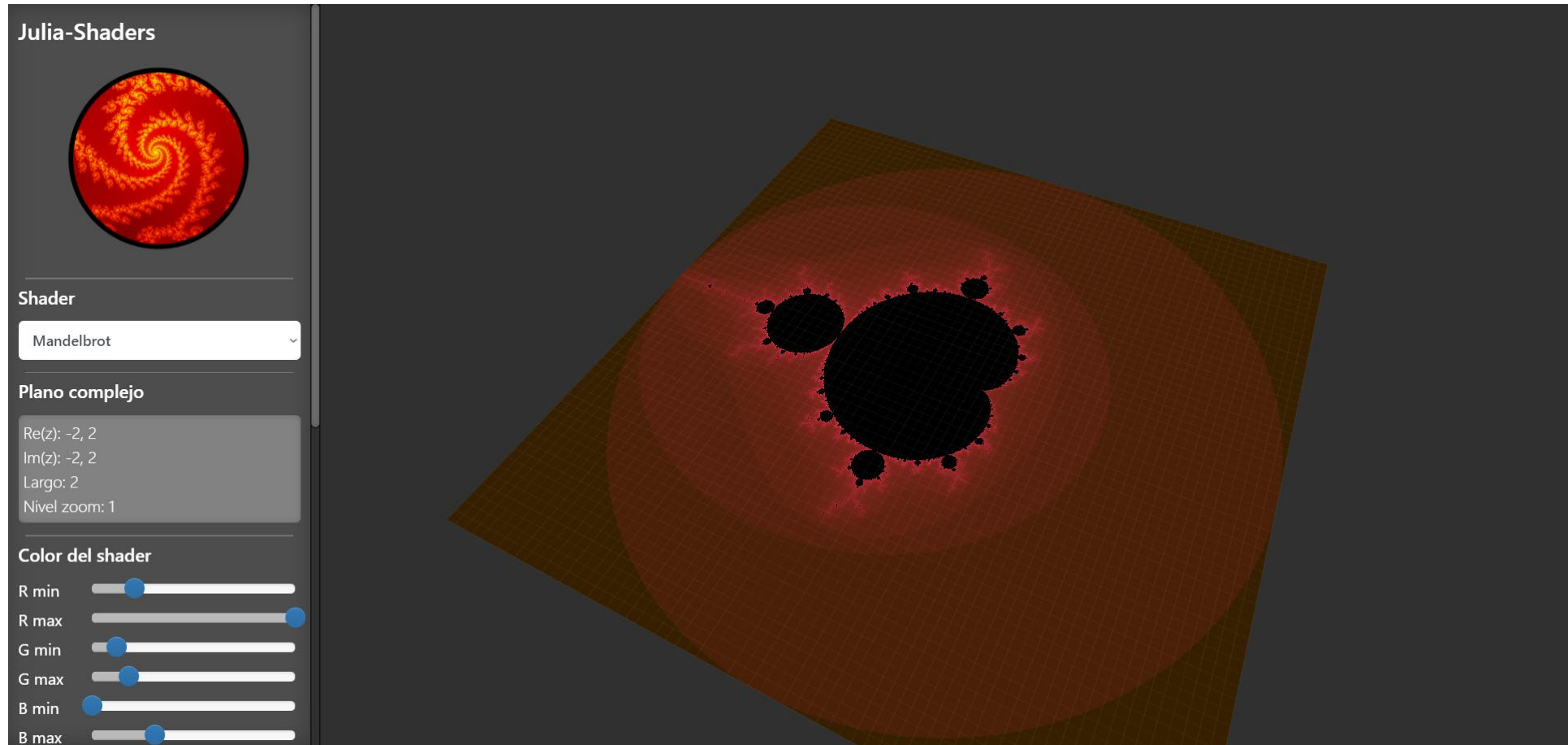
    for (int i = 0; i < 65536; i++) {
        u = w_r;
```

```
        v = w_i;
        w_r = u*u - v*v + c_r;
        w_i = 2.0*u*v + c_i;

        if (w_r*w_r + w_i*w_i > 4.0) { // |z| > 2
            // Computa el rojo
            t = log(float(i + 1)) /
log(float(max_iterations + 1));
            r = t*r_max + (1.0 - t)*r_min;
            // Computa el verde
            g = t*g_max + (1.0 - t)*g_min;
            // Computa el azul
            b = t*b_max + (1.0 - t)*b_min;
            break;
        }
        if (i >= max_iterations) {
            break;
        }
    }
    gl_FragColor = vec4(r, g, b, 1.0);
}
```

Programar en OpenGL

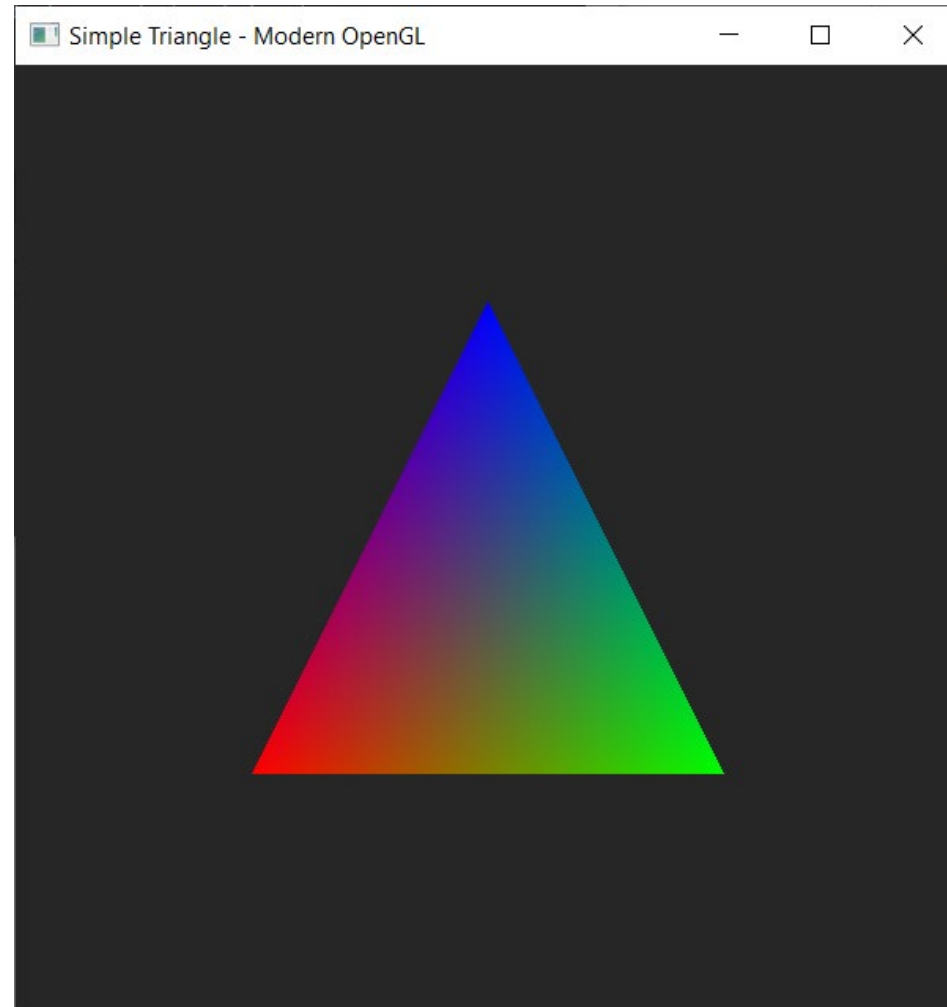
Ejemplos – Tarea N°2 2018



<https://ppizarror.com/julia-shaders-threejs/>

Programar en OpenGL

Ejemplos – Multiplataforma – Ejemplo en Python



Programar en OpenGL

Ejemplos – Multiplataforma – Ejemplo en Python

- Mismo esquema
 1. Iniciar OpenGL
 2. Crear la ventana
 3. Definir la geometría u el objeto (En este caso un triángulo)
 4. Definir los shaders
 5. Crear el VAO
 6. Compilar los shaders
 7. Bucle del programa

Muchas gracias por su
atención

¿Preguntas?