DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

Triángulo de Sierpinski

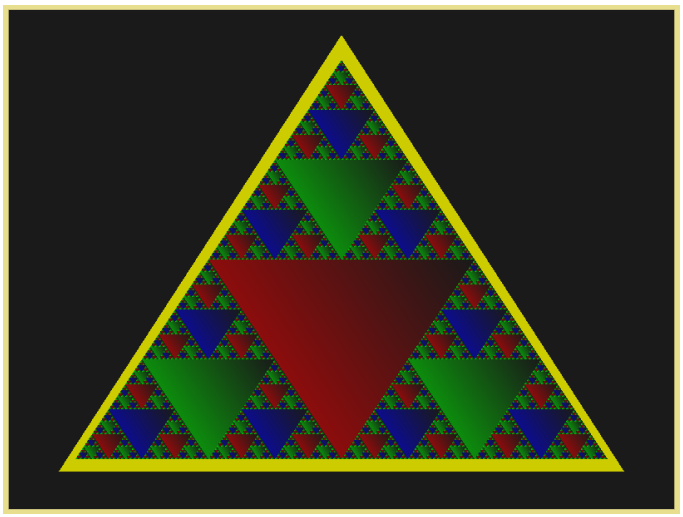
COMPUTACIÓN GRÁFICA

ANTHONY BENAVIDES ARCE
anthony.benavides@ucsp.edu.pe

7MO SEMESTRE

2020

La apariencia de mi Triángulo de Sierpinski es la siguiente:



Modo de uso

1. El programa al ejecutarse renderizará todos los triángulos aparecerá con un máximo número de profundida definido en el main.cpp
2. Presionando la tecla **espacio**, se puede ver la construccion del triangulo emepezando desde el nuivel 0.
3. Presionando la tecla **R**, se puede observar una animación del orden en que fueron colocados los triángulos.

Shaders

Empleé un efecto sombreado de tal forma que se pueda distinguir los triangulos mas profundos de los triangulos centrales. Para lograr esto, implementé de la siguiente manera mis shaders:

```
const GLchar* vertexSource = R"glsl(
    #version 150 core
    in vec2 position;
    in vec3 color;

    out vec3 Color;

    void main()
    {
        Color = color;
        gl_Position = vec4(position, 0.0f, 1.0);
    }
)glsl";

const GLchar* fragmentSource = R"glsl(
    #version 150 core
    in vec3 Color;
    out vec4 outColor;
    void main()
    {
        outColor = vec4(Color, 1.0);
    }
)glsl";
```

Donde la estructura de los vértices son dos floats para la posición y tres floats para el vector, por lo que cada triángulo estaría conformado por un array de 15 floats. Esta primera parte está implementada en una sola VAO.

La animación presionando la tecla R es lograda implementando una clase por cada triangulo, cada clase tiene su propio vertex array object y su vertex buffer object, estos tríangulos se dibujan usando shaders estáticos como:

```
const char *fragmentShader1Source = "#version 330 core\n"
    "out vec4 FragColor;\n"
    "void main()\n"
    "{\n"
    "   FragColor = vec4(1.0f, 0.0f, 0.0f, 1.0f);\n"
    "}\n\0";
const char *fragmentShader2Source = "#version 330 core\n"
    "out vec4 FragColor;\n"
    "void main()\n"
    "{\n"
    "   FragColor = vec4(0.0f, 1.0f, 0.0f, 1.0f);\n"
    "}\n\0";
const char *fragmentShader3Source = "#version 330 core\n"
    "out vec4 FragColor;\n"
    "void main()\n"
    "{\n"
    "   FragColor = vec4(0.0f, 0.0f, 1.0f, 1.0f);\n"
    "}\n\0";
```

Cada shader estático representa un color del RGB.

El resultado es el siguiente: