



UNIVERSIDAD NACIONAL DE SAN AGUSTÍN

ESCUELA PROFESIONAL DE CIENCIA DE LA
COMPUTACIÓN

COMPUTACION GRAFICA

Practica 7

Alumnos:

Chayña Batallanes Josnick
Perez Rodriguez Angelo Aldo
Pucho Zevallos Kelvin Paul
Vilcapaza Flores Luis Felipe
Sihuinta Perez Luis Armando

Abril 2021

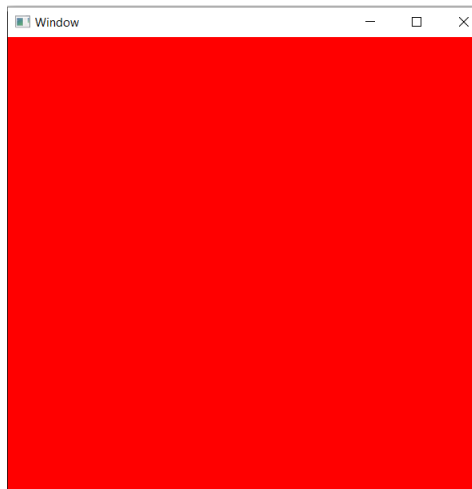
1 Implemente todos los ejemplos del capítulo II, del libro “Computer Graphics Programming in OpenGL with C++” [Gordon and Clevenger(2020)].

1.1 Programa 2.1: Creación de una ventana.

Creacion de un ventana simple mediante la funcion display ,con el color rojo de fondo.

```
1 #include <GL\glew.h>
2 #include <GLFW\glfw3.h>
3 #include <iostream>
4 using namespace std;
5 void init(GLFWwindow* window) { }
6 void display(GLFWwindow* window, double currentTime) {
7     glClearColor(1.0, 0.0, 0.0, 1.0);
8     glClear(GL_COLOR_BUFFER_BIT);
9 }
```

1.2 Capturas



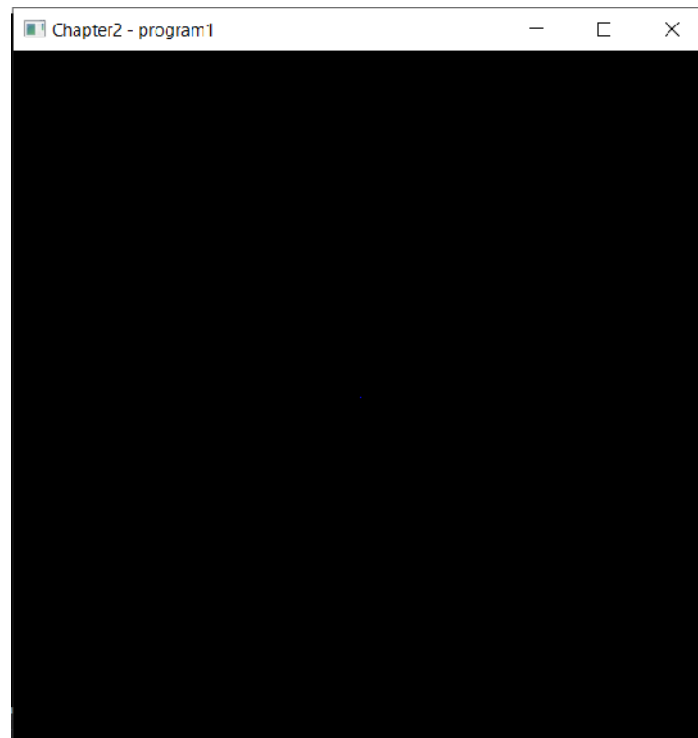
1.3 Programa 2.2: Shaders para la creación de punto (pixel).

Se hace uso de archivos glsl para la creación de shaders ,para este ejemplo el archivo glsl se crea dentro del mismo código aunque esto no es lo recomendable cuando se trabajan con muchos archivos o son muy extensos.

```
1  #pragma once
2  #include <GL/glew.h>
3  #include <GLFW/glfw3.h>
4  #include <iostream>
5
6  #define numVAOs 1
7
8  GLuint renderingProgram;
9  GLuint vao[numVAOs];
10
11 using namespace std;
12
13 //creacion de shaders en el mismo codigo
14 GLuint createShaderProgram() {
15
16     const char* vshaderSource =
17         "#version 430 \n"
18         "void main(void) \n"
19
20         "{gl_Position = vec4(0.0, 0.0, 0.0, 1.0);}";
21     const char* fshaderSource =
22         "#version 430 \n"
23         "out vec4 color; \n"
24         "void main(void) \n"
25         "{color = vec4(0.0, 0.0, 1.0, 1.0);}";
26
27     GLuint vShader = glCreateShader(GL_VERTEX_SHADER);
28     GLuint fShader = glCreateShader(GL_FRAGMENT_SHADER);
29
30     glShaderSource(vShader, 1, &vshaderSource, nullptr);
31     glShaderSource(fShader, 1, &fshaderSource, nullptr);
32
33     glCompileShader(vShader);
34     glCompileShader(fShader);
35
36     GLuint vfProgram = glCreateProgram();
37
38     glAttachShader(vfProgram, vShader);
39     glAttachShader(vfProgram, fShader);
40
41     glLinkProgram(vfProgram);
42
43     return vfProgram;
44 }
45 void init(GLFWwindow* window) {
46     renderingProgram = createShaderProgram();
47
48
49     glGenVertexArrays(numVAOs, vao);
50     glBindVertexArray(vao[0]);
51 }
```

```
52 void display(GLFWwindow* window, double currentTime) {
53
54     glUseProgram(renderingProgram);
55
56     //glPointSize(10.0f); //aumentar el tamaño del pixel
57
58     glDrawArrays(GL_POINTS, 0, 1);
59 }
```

1.4 Capturas



1.5 Programa 2.3: Obtención de GLSL errores

En esta parte solo se nos indica que la GPU compila el programa y no la CPU por lo tanto a veces pueden haber errores en la compilación pero no nos va indicar la terminal por lo cual se le debe agregar funciones que nos indiquen si hay error:

- **printShaderLog:** Muestra el contenido del registro de OpenGL cuando falla la compilación de GLSL y esto se va ver gracias a la variable `fragCompiled` y analizando la variable `fShared`
- **printProgramLog:** Muestra el contenido del registro de OpenGL cuando falla el enlace GLSL, esto se ve gracias a la variable `linked` y gracias al análisis de la variable `vfProgram`
- **checkOpenGLError:** Comprueba el indicador de error de OpenGL para detectar la aparición de un error de OpenGL

```
1  #include <glad/glad.h>
2  #include <glfw/glfw3.h>
3
4  #include <iostream>
5
6  using namespace std;
7
8
9  #define numVAOs 1
10 GLuint renderingProgram;
11 GLuint vao[numVAOs];
12
13 // functions to catch errors in GLSL
14
15 void printShaderLog(GLuint shader) {
16     int len = 0;
17     int chWrittn = 0;
18     char *log;
19     glGetShaderiv(shader, GL_INFO_LOG_LENGTH, &len);
20     if (len > 0) {
21         log = (char *)malloc(len);
22         glGetShaderInfoLog(shader, len, &chWrittn, log);
23         cout << "Shader Info Log: " << log << endl;
24         free(log);
25     }
26 }
27
28 void printProgramLog(int prog) {
29     int len = 0;
30     int chWrittn = 0;
31     char *log;
32     glGetProgramiv(prog, GL_INFO_LOG_LENGTH, &len);
33     if (len > 0) {
34         log = (char *)malloc(len);
35         glGetProgramInfoLog(prog, len, &chWrittn, log);
36         cout << "Program Info Log: " << log << endl;
37         free(log);
38     }
39 }
```

```

39 bool checkOpenGLError() {
40     bool foundError = false;
41     int glErr = glGetError();
42     while (glErr != GL_NO_ERROR) {
43         cout << "glError: " << glErr << endl;
44         foundError = true;
45         glErr = glGetError();
46     }
47     return foundError;
48 }
49
50
51 GLuint createShaderProgram() {
52     GLint vertCompiled;
53     GLint fragCompiled;
54     GLint linked;
55
56     const char *vshaderSource =
57         "#version 430 \n"
58         "void main(void) \n"
59         "{ gl_Position = vec4(0.0, 0.0, 0.0, 1.0); }";
60     // creamos un vertice, no especificamos salida, porque
61     // gl_position es por defecto de salida
62     const char *fshaderSource =
63         "#version 430 \n"
64         "out vec4 color; \n"
65         "void main(void) \n"
66         "{ if (gl_FragCoord.x < 200) color = vec4(1.0, 0.0, 0.0, 1.0)
67           ; else color = vec4(0.0, 0.0, 1.0, 1.0); }";
68     //especificamos el color de los pixeles
69
70     GLuint vShader = glCreateShader(GL_VERTEX_SHADER);
71     GLuint fShader = glCreateShader(GL_FRAGMENT_SHADER);
72
73     glShaderSource(vShader, 1, &vshaderSource, NULL);
74     glShaderSource(fShader, 1, &fshaderSource, NULL);
75
76     glCompileShader(vShader);
77     checkOpenGLError();
78     glGetShaderiv(vShader, GL_COMPILE_STATUS, &vertCompiled);
79     if (vertCompiled != 1) {
80         cout << "vertex compilation failed" << endl;
81         printShaderLog(vShader);
82     }
83
84     glCompileShader(fShader);
85     checkOpenGLError();
86     glGetShaderiv(fShader, GL_COMPILE_STATUS, &fragCompiled);
87     if (fragCompiled != 1) {
88         cout << "fragment compilation failed" << endl;
89         printShaderLog(fShader);
90     }
91
92     GLuint vfProgram = glCreateProgram();
93     glAttachShader(vfProgram, vShader);

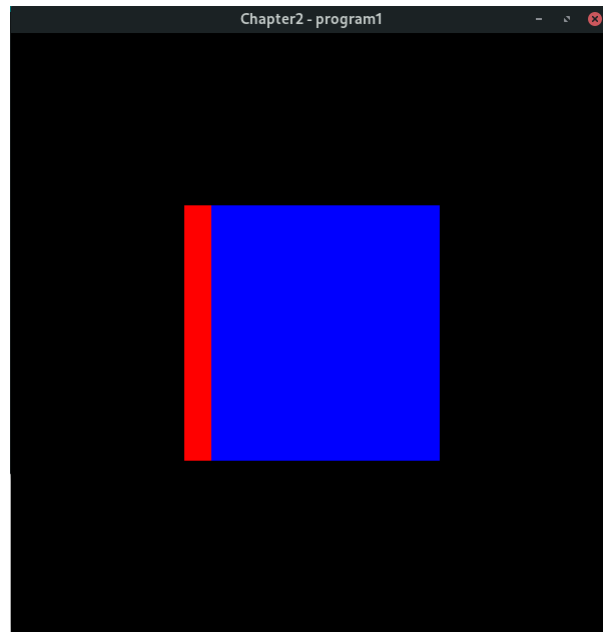
```

```

93     glAttachShader(vfProgram, fShader);
94
95     glLinkProgram(vfProgram);
96     checkOpenGLError();
97     glGetProgramiv(vfProgram, GL_LINK_STATUS, &linked);
98     if (linked != 1) {
99         cout << "linking failed" << endl;
100         printProgramLog(vfProgram);
101     }
102
103     return vfProgram;
104 }
105
106 void init(GLFWwindow* window) {
107     renderingProgram = createShaderProgram();
108     glGenVertexArrays(numVAOs, vao);
109     glBindVertexArray(vao[0]);
110 }
111
112 void display(GLFWwindow* window, double currentTime) {
113     glUseProgram(renderingProgram);
114     glPointSize(400.0f); // un vertice es un pixel, con esto
                          // especificamos el tamaño del pixel
115     //glPolygonMode(GL_FRONT_AND_BACK, GL_LINE); //wire frame
116     glPolygonMode(GL_FRONT_AND_BACK, GL_FILL); // noraml
117     glDrawArrays(GL_POINTS, 0, 1);
118
119 }
120
121 int main(void) {
122     if (!glfwInit()) { exit(EXIT_FAILURE); }
123     glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, 4);
124     glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, 3);
125
126     GLFWwindow* window = glfwCreateWindow(600, 600, "Chapter2 -
                          program1", NULL, NULL);
127     glfwMakeContextCurrent(window);
128
129     //if (glewInit() != GLEW_OK) { exit(EXIT_FAILURE); }
130     if (!gladLoadGLLoader((GLADloadproc)glfwGetProcAddress)){ exit(
        EXIT_FAILURE); }
131
132     glfwSwapInterval(1);
133     init(window);
134     while (!glfwWindowShouldClose(window)) {
135         display(window, glfwGetTime());
136         glfwSwapBuffers(window);
137         glfwPollEvents();
138     }
139
140     glfwDestroyWindow(window);
141     glfwTerminate();
142
143     exit(EXIT_SUCCESS);
144
145 }

```

1.6 Capturas



1.7 Programa 2.4: Lectura de los shaders desde un archivo.

Bueno el objetivo del ejercicio 2.4 es reducir toda la parte de strings llamando archivos .gsls para no escribir tanto por lo cual se crea la variable readShaderSource() lo cual va leer el archivo .gsls luego estos van a ser leídos en nuestros caso por vertShaderStr y fragShaderStr estos van a tener toda la información del archivo .gsls y si corremos es lo mismo que el ejercicio pasado

```
1  #include <glad/glad.h>
2  #include <glfw/glfw3.h>
3
4  #include <iostream>
5  #include <fstream>
6  #include <unistd.h>
7  #include <cstring>
8  using namespace std;
9
10 char* vertShaderPath = "/home/angelo/Escritorio/opengl/src/04
    _shader_file/vertShader.glsl";
11 char* fragShaderPath = "/home/angelo/Escritorio/opengl/src/04
    _shader_file/fragShader.glsl";
12 #define numVAOs 1
13 GLuint renderingProgram;
14 GLuint vao[numVAOs];
15
16 string readShaderSource(const char *filePath) {
17     string content;
18     ifstream fileStream(filePath, ios::in);
19     string line = "";
20     while (!fileStream.eof()) {
21         getline(fileStream, line);
22         content.append(line + "\n");
23     }
24     fileStream.close();
25     return content;
26 }
27
28 // functions to catch errors in GLSL
29
30 void printShaderLog(GLuint shader) {
31     int len = 0;
32     int chWrittn = 0;
33     char *log;
34     glGetShaderiv(shader, GL_INFO_LOG_LENGTH, &len);
35     if (len > 0) {
36         log = (char *)malloc(len);
37         glGetShaderInfoLog(shader, len, &chWrittn, log);
38         cout << "Shader Info Log: " << log << endl;
39         free(log);
40     }
41 }
42
43 void printProgramLog(int prog) {
44     int len = 0;
45     int chWrittn = 0;
46     char *log;
47     glGetProgramiv(prog, GL_INFO_LOG_LENGTH, &len);
```

```

47     if (len > 0) {
48         log = (char *)malloc(len);
49         glGetProgramInfoLog(prog, len, &chWrittn, log);
50         cout << "Program Info Log: " << log << endl;
51         free(log);
52     }
53 }
54 bool checkOpenGLError() {
55     bool foundError = false;
56     int glErr = glGetError();
57     while (glErr != GL_NO_ERROR) {
58         cout << "glError: " << glErr << endl;
59         foundError = true;
60         glErr = glGetError();
61     }
62     return foundError;
63 }
64
65 GLuint createShaderProgram() {
66     GLint vertCompiled;
67     GLint fragCompiled;
68     GLint linked;
69
70     GLuint vShader = glCreateShader(GL_VERTEX_SHADER);
71     GLuint fShader = glCreateShader(GL_FRAGMENT_SHADER);
72
73     // read shaders from files
74     string vertShaderStr = readShaderSource(vertShaderPath);
75     string fragShaderStr = readShaderSource(fragShaderPath);
76     const char *vertShaderSrc = vertShaderStr.c_str();
77     const char *fragShaderSrc = fragShaderStr.c_str();
78     glShaderSource(vShader, 1, &vertShaderSrc, NULL);
79     glShaderSource(fShader, 1, &fragShaderSrc, NULL);
80
81     glCompileShader(vShader);
82     checkOpenGLError();
83     glGetShaderiv(vShader, GL_COMPILE_STATUS, &vertCompiled);
84     if (vertCompiled != 1) {
85         cout << "vertex compilation failed" << endl;
86         printShaderLog(vShader);
87     }
88
89     glCompileShader(fShader);
90     checkOpenGLError();
91     glGetShaderiv(fShader, GL_COMPILE_STATUS, &fragCompiled);
92     if (fragCompiled != 1) {
93         cout << "fragment compilation failed" << endl;
94         printShaderLog(fShader);
95     }
96
97
98     GLuint vfProgram = glCreateProgram();
99     glAttachShader(vfProgram, vShader);
100    glAttachShader(vfProgram, fShader);
101
102    glLinkProgram(vfProgram);

```

```

103     checkOpenGLError();
104     glGetProgramiv(vfProgram, GL_LINK_STATUS, &linked);
105     if (linked != 1) {
106         cout << "linking failed" << endl;
107         printProgramLog(vfProgram);
108     }
109
110     return vfProgram;
111 }
112
113 void init(GLFWwindow* window) {
114     renderingProgram = createShaderProgram();
115     glGenVertexArrays(numVAOs, vao);
116     glBindVertexArray(vao[0]);
117 }
118
119 void display(GLFWwindow* window, double currentTime) {
120     glUseProgram(renderingProgram);
121     glPointSize(400.0f); // un vertice es un pixel, con esto
                          // especificamos el tamaño del pixel
122     //glPolygonMode(GL_FRONT_AND_BACK, GL_LINE); //wire frame
123     glPolygonMode(GL_FRONT_AND_BACK, GL_FILL); // noraml
124     glDrawArrays(GL_POINTS, 0, 1);
125
126 }
127
128 int main(void) {
129     if (!glfwInit()) { exit(EXIT_FAILURE); }
130     glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, 4);
131     glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, 3);
132
133     GLFWwindow* window = glfwCreateWindow(600, 600, "Chapter2 -
                          program1", NULL, NULL);
134     glfwMakeContextCurrent(window);
135
136     //if (glewInit() != GLEW_OK) { exit(EXIT_FAILURE); }
137     if (!gladLoadGLLoader((GLADloadproc)glfwGetProcAddress)){ exit(
        EXIT_FAILURE); }
138
139     glfwSwapInterval(1);
140     init(window);
141     while (!glfwWindowShouldClose(window)) {
142         display(window, glfwGetTime());
143         glfwSwapBuffers(window);
144         glfwPollEvents();
145     }
146
147     glfwDestroyWindow(window);
148     glfwTerminate();
149     exit(EXIT_SUCCESS);
150
151 }

```

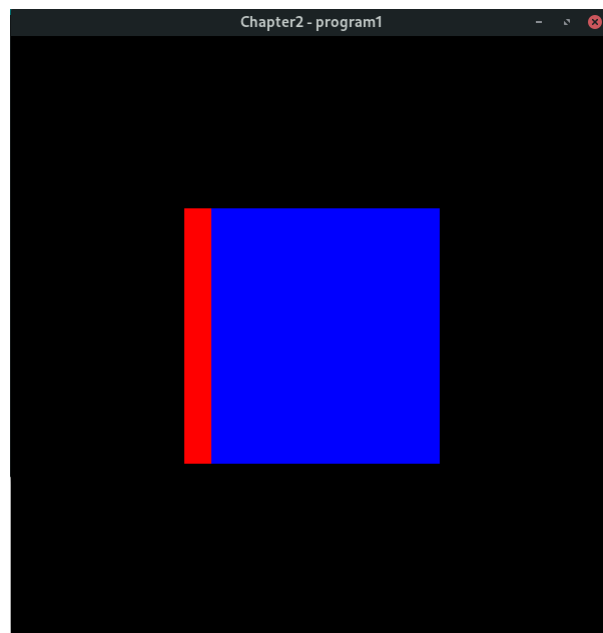
1.8 Vertex Shader

```
1 #version 430
2 void main(void){
3     gl_Position = vec4(0.0, 0.0, 0.0, 1.0);
4 }
```

1.9 Fragment Shader

```
1     #version 430
2
3 out vec4 color;
4 void main(void){
5     if (gl_FragCoord.x < 200)
6         color = vec4(1.0, 0.0, 0.0, 1.0);
7     else
8         color = vec4(0.0, 0.0, 1.0, 1.0);
9 }
```

1.10 Capturas



1.11 Programa 2.5: Un triángulo

```
1  #include <GL/glew.h>
2  #include <GLFW/glfw3.h>
3  #include <string>
4  #include <iostream>
5  #include <fstream>
6
7  #define numVAOs 1
8
9  GLuint renderingProgram;
10 GLuint vao[numVAOs];
11
12 using namespace std;
13
14 void printShaderLog(GLuint shader) {
15     int len = 0;
16     int chWrittn = 0;
17     char* log;
18     glGetShaderiv(shader, GL_INFO_LOG_LENGTH, &len);
19     if (len > 0) {
20         log = (char*)malloc(len);
21         glGetShaderInfoLog(shader, len, &chWrittn, log);
22         cout << "Shader Info Log: " << log << endl;
23         free(log);
24     }
25 }
26
27 void printProgramLog(int prog) {
28     int len = 0;
29     int chWrittn = 0;
30     char* log;
31     glGetProgramiv(prog, GL_INFO_LOG_LENGTH, &len);
32     if (len > 0) {
33         log = (char*)malloc(len);
34         glGetProgramInfoLog(prog, len, &chWrittn, log);
35         cout << "Program Info Log: " << log << endl;
36         free(log);
37     }
38 }
39
40 bool checkOpenGLError() {
41     bool foundError = false;
42     int glErr = glGetError();
43     while (glErr != GL_NO_ERROR) {
44         cout << "glError: " << glErr << endl;
45         foundError = true;
46         glErr = glGetError();
47     }
48     return foundError;
49 }
50
51 string readShaderSource(const char* filePath) {
52     string content = "";
53     ifstream fileStream(filePath, ios::in);
54 }
```

```

55     string line = "";
56     while (!fileStream.eof()) {
57         getline(fileStream, line);
58         content.append(line + "\n");
59     }
60     fileStream.close();
61     return content;
62 }
63
64 GLuint createShaderProgram() {
65     GLint vertCompiled;
66     GLint fragCompiled;
67     GLint linked;
68
69     /**
70     Dado que escribir archivos glsl en el mismo codigo es
71     ineficiente cuando estos son grandes
72     se pueden llamar estos archivos con extension glsl desde fuera
73     especificando la ruta
74     */
75     string vertShaderStr = readShaderSource("/Users/estilos/source/
76     repos/Grafica_opengl/glsl/vertexShader.glsl");
77     string fragShaderStr = readShaderSource("/Users/estilos/source/
78     repos/Grafica_opengl/glsl/fragShader.glsl");
79
80     const char* vertShaderSrc = vertShaderStr.c_str();
81     const char* fragShaderSrc = fragShaderStr.c_str();
82
83     GLuint vShader = glCreateShader(GL_VERTEX_SHADER);
84     GLuint fShader = glCreateShader(GL_FRAGMENT_SHADER);
85
86     glShaderSource(vShader, 1, &vertShaderSrc, nullptr);
87     glShaderSource(fShader, 1, &fragShaderSrc, nullptr);
88
89     glCompileShader(vShader);
90     checkOpenGLError();
91     glGetShaderiv(vShader, GL_COMPILE_STATUS, &vertCompiled);
92     if (vertCompiled != 1) {
93         cout << "vertex compilation failed" << endl;
94         printShaderLog(vShader);
95     }
96
97     glCompileShader(fShader);
98     checkOpenGLError();
99     glGetShaderiv(fShader, GL_COMPILE_STATUS, &fragCompiled);
100     if (fragCompiled != 1) {
101         cout << "fragment compilation failed" << endl;
102         printShaderLog(fShader);
103     }
104
105     GLuint vfProgram = glCreateProgram();
106     glAttachShader(vfProgram, vShader);
107     glAttachShader(vfProgram, fShader);
108
109     glLinkProgram(vfProgram);
110     checkOpenGLError();

```

```

107     glGetProgramiv(vfProgram, GL_LINK_STATUS, &linked);
108     if (linked != 1) {
109         cout << "linking failed" << endl;
110         printProgramLog(vfProgram);
111     }
112
113     return vfProgram;
114 }
115
116
117 void init(GLFWwindow* window) {
118     renderingProgram = createShaderProgram();
119     glGenVertexArrays(numVAOs, vao);
120     glBindVertexArray(vao[0]);
121 }
122
123 void display(GLFWwindow* window, double currentTime) {
124     glUseProgram(renderingProgram);
125     glPointSize(30.0f);
126
127     //Dibujar el triangulo
128     glDrawArrays(GL_TRIANGLES, 0, 3);
129 }
130
131 int main(void) {
132     if (!glfwInit()) { exit(EXIT_FAILURE); }
133     glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, 4);
134     glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, 1);
135
136     GLFWwindow* window = glfwCreateWindow(600, 600, "Pregunta 2.5",
137         nullptr, nullptr);
138     glfwMakeContextCurrent(window);
139     if (glewInit() != GLEW_OK) { exit(EXIT_FAILURE); }
140     glfwSwapInterval(1);
141
142     init(window);
143
144     while (!glfwWindowShouldClose(window)) {
145         display(window, glfwGetTime());
146         glfwSwapBuffers(window);
147         glfwPollEvents();
148     }
149
150     glfwDestroyWindow(window);
151     glfwTerminate();
152     exit(EXIT_SUCCESS);
153 }

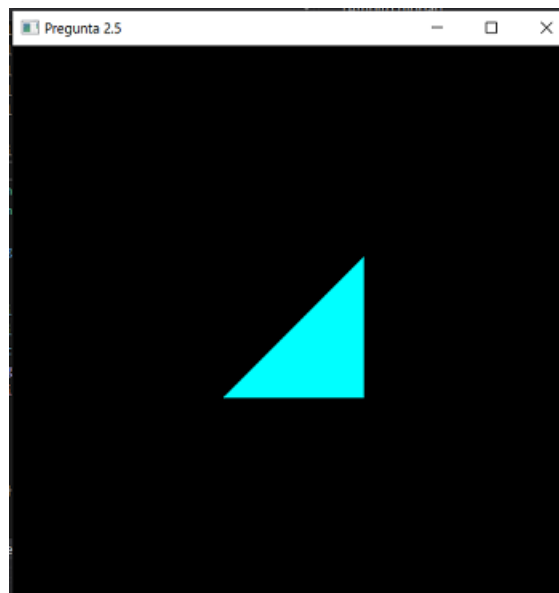
```

1.12 Vertex Shader

```
1  #version 430
2  void main(void)
3  {
4      if (gl_VertexID == 0) gl_Position = vec4(0.25, -0.25, 0.0, 1.0)
5          ;
6      else if (gl_VertexID == 1) gl_Position = vec4(-0.25, -0.25,
7          0.0, 1.0);
8      else gl_Position = vec4(0.25, 0.25, 0.0, 1.0);
9  }
```

1.13 Fragment Shader

```
1  #version 430
2  out vec4 color;
3  void main(void)
4  {color = vec4(0.0, 1.0, 1.0, 1.0);}
```



1.14 Programa 2.6: Una animación simple

Para realizar la animación se hace configura el display para que cada vez que se llama los valores de x cambien mediante un offset

```
1  #include <GL/glew.h>
2  #include <GLFW/glfw3.h>
3  #include <string>
4  #include <iostream>
5  #include <fstream>
6
7  #define numVAOs 1
8
9  GLuint renderingProgram;
10 GLuint vao[numVAOs];
11
12 float x = 0.0f;
13 float inc = 0.01f;
14
15 using namespace std;
16
17 void printShaderLog(GLuint shader) {
18     int len = 0;
19     int chWrittn = 0;
20     char* log;
21     glGetShaderiv(shader, GL_INFO_LOG_LENGTH, &len);
22     if (len > 0) {
23         log = (char*)malloc(len);
24         glGetShaderInfoLog(shader, len, &chWrittn, log);
25         cout << "Shader Info Log: " << log << endl;
26         free(log);
27     }
28 }
29
30 void printProgramLog(int prog) {
31     int len = 0;
32     int chWrittn = 0;
33     char* log;
34     glGetProgramiv(prog, GL_INFO_LOG_LENGTH, &len);
35     if (len > 0) {
36         log = (char*)malloc(len);
37         glGetProgramInfoLog(prog, len, &chWrittn, log);
38         cout << "Program Info Log: " << log << endl;
39         free(log);
40     }
41 }
42
43 bool checkOpenGLError() {
44     bool foundError = false;
45     int glErr = glGetError();
46     while (glErr != GL_NO_ERROR) {
47         cout << "glError: " << glErr << endl;
48         foundError = true;
49         glErr = glGetError();
50     }
51     return foundError;
52 }
```

```

53
54 string readShaderSource(const char* filePath) {
55     string content = "";
56     ifstream fileStream(filePath, ios::in);
57     string line = "";
58     while (!fileStream.eof()) {
59         getline(fileStream, line);
60         content.append(line + "\n");
61     }
62     fileStream.close();
63     return content;
64 }
65
66 GLuint createShaderProgram() {
67     GLint vertCompiled;
68     GLint fragCompiled;
69     GLint linked;
70
71     //direccion de los archivos glsl
72     string vertShaderStr = readShaderSource("/Users/estilos/source/
        repos/Grafica_opengl/glsl/vertexShader.glsl");
73     string fragShaderStr = readShaderSource("/Users/estilos/source/
        repos/Grafica_opengl/glsl/fragShader.glsl");
74
75     const char* vertShaderSrc = vertShaderStr.c_str();
76     const char* fragShaderSrc = fragShaderStr.c_str();
77
78     GLuint vShader = glCreateShader(GL_VERTEX_SHADER);
79     GLuint fShader = glCreateShader(GL_FRAGMENT_SHADER);
80
81     glShaderSource(vShader, 1, &vertShaderSrc, nullptr);
82     glShaderSource(fShader, 1, &fragShaderSrc, nullptr);
83
84     glCompileShader(vShader);
85     checkOpenGLError();
86     glGetShaderiv(vShader, GL_COMPILE_STATUS, &vertCompiled);
87     if (vertCompiled != 1) {
88         cout << "vertex compilation failed" << endl;
89         printShaderLog(vShader);
90     }
91
92     glCompileShader(fShader);
93     checkOpenGLError();
94     glGetShaderiv(fShader, GL_COMPILE_STATUS, &fragCompiled);
95     if (fragCompiled != 1) {
96         cout << "fragment compilation failed" << endl;
97         printShaderLog(fShader);
98     }
99
100    GLuint vfProgram = glCreateProgram();
101    glAttachShader(vfProgram, vShader);
102    glAttachShader(vfProgram, fShader);
103
104    glLinkProgram(vfProgram);
105    checkOpenGLError();
106    glGetProgramiv(vfProgram, GL_LINK_STATUS, &linked);

```

```

107     if (linked != 1) {
108         cout << "linking failed" << endl;
109         printProgramLog(vfProgram);
110     }
111
112     return vfProgram;
113 }
114
115
116 void init(GLFWwindow* window) {
117     renderingProgram = createShaderProgram();
118     glGenVertexArrays(numVAOs, vao);
119     glBindVertexArray(vao[0]);
120 }
121
122 //limpia el display y altera la pos de x
123 void display(GLFWwindow* window, double currentTime) {
124
125     glClear(GL_DEPTH_BUFFER_BIT);
126     glClearColor(0.0, 0.0, 0.0, 1.0);
127     glClear(GL_COLOR_BUFFER_BIT);
128
129     glUseProgram(renderingProgram);
130
131     //mover en x
132     x += inc;
133     //cambia la direccion
134     if (x > 1.0f) inc = -0.01f;
135     //cambi la direccion otra vez
136     if (x < -1.0f) inc = 0.01f;
137
138
139     GLuint offsetLoc = glGetUniformLocation(renderingProgram, "
        offset");
140
141     glProgramUniform1f(renderingProgram, offsetLoc, x);
142
143     glDrawArrays(GL_TRIANGLES, 0, 3);
144 }

```

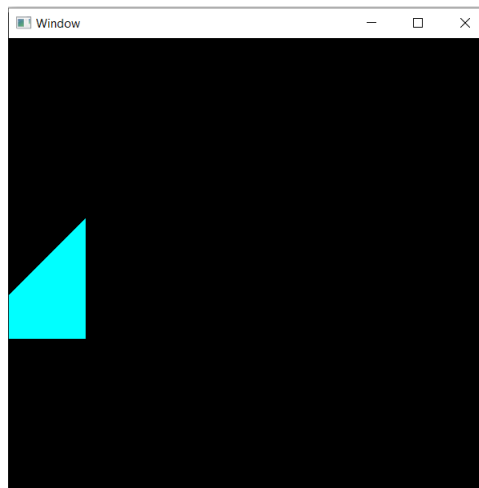
1.15 Vertex Shader

```
1 #version 430
2 void main(void)
3 { if (gl_VertexID == 0) gl_Position = vec4( 0.25, -0.25, 0.0, 1.0);
4   else if (gl_VertexID == 1) gl_Position = vec4(-0.25, -0.25,
5     0.0, 1.0);
6   else gl_Position = vec4( 0.25, 0.25, 0.0, 1.0);
7 }
```

1.16 Fragment Shader

```
1 #version 430
2 out vec4 color;
3 void main(void)
4 {color = vec4(0.0, 1.0, 1.0, 1.0);}
```

1.17 Capturas



2 Modifique el programa 2.2, para agregar una animación donde el punto crezca y se encoja (puede utilizar `glPointSize()`).

2.1 Solucion

Se utilizo la misma logica que el libro utilizo cuando transporto el triangulo en el Programa 2.6, Se declaro variable de tipo float para utilizar como incremento y decremento del vertice para asi en cada reiteracion el parametro de la funcion `glPointSize(x)` la variable `x` que representa el tamaño alterado es establecido. Entonces cuando el rasterizador recibe el vértice del vertex shader, establecerá valores de color de píxeles

```
1  #include <glad/glad.h>
2  #include <glfw/glfw3.h>
3  #include <iostream>
4
5  using namespace std;
6  #define numVAOs 1
7  GLuint renderingProgram;
8  GLuint vao[numVAOs];
9
10 float x = 0.0f;
11 float inc = 0.01f;
12
13 GLuint createShaderProgram() {
14     const char *vshaderSource =
15         "#version 430 \n" // version 4.3 glsl es parecido a c++
16         "void main(void) \n"
17         "{ gl_Position = vec4(0.0, 0.0, 0.0, 1.0); }";
18     const char *fshaderSource =
19         "#version 430 \n"
20         "out vec4 color; \n"// out indica que el variable color es
21         una salida
22         "void main(void) \n"
23         "{ color = vec4(0.0, 0.0, 1.0, 1.0); }";
24
25     GLuint vShader = glCreateShader(GL_VERTEX_SHADER); //generando
26         shader for vertex vacios
27     GLuint fShader = glCreateShader(GL_FRAGMENT_SHADER); //generando
28         shader for fragment vacios
29
30
31     glShaderSource(vShader, 1, &vshaderSource, NULL);
32     glShaderSource(fShader, 1, &fshaderSource, NULL);
33
34     glCompileShader(vShader);
35     glCompileShader(fShader);
36     GLuint vfProgram = glCreateProgram();
37
38     glAttachShader(vfProgram, vShader);
39     glAttachShader(vfProgram, fShader);
40     glLinkProgram(vfProgram);
41     return vfProgram;
42 }
```

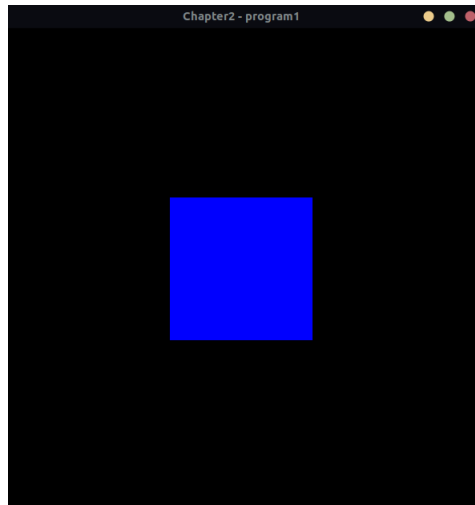
```

40
41 void init(GLFWwindow* window) {
42     renderingProgram = createShaderProgram();
43     glGenVertexArrays(numVAOs, vao);
44     glBindVertexArray(vao[0]);
45 }
46
47 void display(GLFWwindow* window, double currentTime) {
48     glClear(GL_DEPTH_BUFFER_BIT);
49     glClearColor(0.0, 0.0, 0.0, 1.0);
50     glClear(GL_COLOR_BUFFER_BIT);
51     glUseProgram(renderingProgram);
52
53     x += inc;
54     if (x > 100.0f) inc = -0.08f;
55     if (x < 0.0f) inc = 0.08f;
56     cout << x << endl;
57     glPointSize(x);
58     glDrawArrays(GL_POINTS, 0, 1);
59
60 }
61 int main(void) {
62     if (!glfwInit()) { exit(EXIT_FAILURE); }
63     glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, 4);
64     glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, 3);
65
66     GLFWwindow* window = glfwCreateWindow(600, 600, "Chapter2 -
        program1", NULL, NULL);
67     glfwMakeContextCurrent(window);
68
69     if (!gladLoadGLLoader((GLADloadproc)glfwGetProcAddress)){ exit(
        EXIT_FAILURE); }
70
71     glfwSwapInterval(1);
72     init(window);
73     while (!glfwWindowShouldClose(window)) {
74         display(window, glfwGetTime());
75         glfwSwapBuffers(window);
76         glfwPollEvents();
77     }
78     glfwDestroyWindow(window);
79     glfwTerminate();
80     exit(EXIT_SUCCESS);
81
82 }

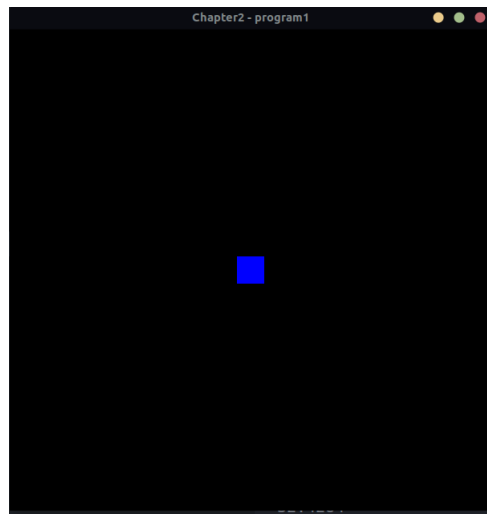
```

2.2 Capturas

- Vertice incrementado



- Vertice decrementado



3 Modifique el programa 2.5, de manera tal que se grafique un triangulo isóseles.

```
1 #include <HSGIL/hsgil.hpp>
2 #include <iostream>
3 using namespace std;
4 using namespace luis;
5 int main(void) {
6     RenderingWindow window(200, 200, "Vpri");
7     Shader sh1("triangulo");
8     float vertices[] = {-0.5, -0.5, 0, 0, 0.5, 0, 0.5, -0.5, 0};
9     unsigned int vao, vbo;
10    glGenVertexArrays(1, &vao);
11    glGenBuffers(1, &vbo);
12    glBindVertexArray(vao);
13    glBindBuffer(GL_ARRAY_BUFFER, vbo);
14    glBufferData(GL_ARRAY_BUFFER, sizeof(vertices), vertices,
15                GL_STATIC_DRAW);
16    glVertexAttribPointer(0, 3, GL_FLOAT, false, 3*sizeof(float),
17                          0);
18    glEnableVertexAttribArray(0);
19    glBindVertexArray(0);
20    while(window.isActive())
21    {
22        glClearColor(0,0,0,1);
23        glClear(GL_COLOR_BUFFER_BIT);
24        sh1.use();
25        glBindVertexArray(vao);
26        glDrawArrays(GL_TRIANGLES, 0, 3);
27        glBindVertexArray(0); //
28        window.pollEvents();
29        window.swapBuffers();
30    }
```

```
1 #version 330 core
2 out vec4 fc;
3
4 void main()
5 {
6     fc = vec4(0.2, 0.3, 0.5, 1);
7 }
```

```
1 #version 330 core
2 layout(location = 0) in vec3 pos;
3
4 void main(){
5     gl_Position = vec4(pos, 1);
6
7 }
```

