

Programación de Videojuegos



Universidad Autónoma de San Luis Potosí

Facultad de Ingeniería

Parcial 2

Angel de Jesús Maldonado Juárez

Actividad 1

14 de marzo del 2023



Incorporación de texturas al proyecto de iluminación

Como primer paso simplemente se agregó la directiva `#include` con la librería `Texture.h` en el archivo `main.cpp`, la cual tiene la clase `Texture` para poder importar, cargar, y manejar texturas:

```
200 #include <libraries.h>
201
202 #include <Camera.h>
203 #include "Variables.h"
204 #include "Cubo.h"
205
206 #include <Shader.h>
207 #include <Texture.h>
208
209 void initGLFWVersion();
210 bool gladLoad();
211 void updateWindow(GLFWwindow* window, Shader ourShader, Shader ourLightShader, Texture ourTexture);
212
213 void framebuffer_size_callback(GLFWwindow* window, int width, int height);
214 void mouse_callback(GLFWwindow* window, double xposIn, double yposIn);
215 void scroll_callback(GLFWwindow* window, double xoffset, double yoffset);
216 void processInput(GLFWwindow* window);
217
218 void GenerationBufferCube();
```

En la función `main()` de este mismo archivo, se agregó la variable `ourTexture`, la cual se utiliza para cargar y guardar la textura en tiempo de ejecución:

```
1
2 limite = (sizeof(texture) / sizeof(texture[0]));
3
4 Shader ourShader("vertexShader.vs", "fragmenShader.fs");
5 Shader ourLightShader("vertexLight.vl", "fragmenLight.fl");
6 Texture ourTexture(texture, limite);
7
8 GenerationBufferCube();
9 GenerationBufferLight();
10
11 for (int i = 0; i < limite; i++)
12     ourTexture.GeneraTextura(texture[i], nombre[i], tipo[i], expan[i]);
13
14 if (limite > 1)
15 {
16     ourShader.use();
17     for (int i = 0; i < limite; i++)
18         ourShader.setInt(ourTexture.UniformTexture(), i);
19 }
20
21 updateWindow(window, ourShader, ourLightShader, ourTexture);
```

En este mismo archivo, en la función `updateWindow()` se le agregó como parámetro un objeto `Texture` llamado `ourTexture`, el cual se utiliza para renderizar en tiempo de ejecución la(s) textura(s) que vayan a cargarse utilizando el método `ViewTexture()` del objeto del parámetro:

```
7 #include <Shader.h>
8 #include <Texture.h>
9
10 void initGLFWVersion();
11 bool gladLoad();
12 void updateWindow(GLFWwindow* window, Shader ourShader, Shader ourLightShader, Texture ourTexture);
13
14 void framebuffer_size_callback(GLFWwindow* window, int width, int height);
```

```

24 void updateWindow(GLFWwindow* window, Shader ourShader, Shader ourLightShader, Texture ourTexture)
23 {
22     while (!glfwWindowShouldClose(window))
21     {
20         float currentFrame = glfwGetTime();
19         deltaTime = currentFrame - lastFrame;
18         lastFrame = currentFrame;
17
16         processInput(window);
15
14         glClearColor(0.1f, 0.1f, 0.1f, 1.0f);
13         glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
12
11         ourTexture.ViewTexture();
10
9         ourShader.use();
8         ourShader.setVec3("viewPos", camera.Position);
7         ourShader.setVec3("objectColor", 0.1f, 0.5f, 1.0f);

```

En el archivo `Cube.h`, se le agregó al arreglo `vertices`, las coordenadas de la textura después de las normales:

```

1 float vertices[] = {
2     -0.5f, -0.5f, -0.5f, -1.0f, 0.0f, 0.0f, 0.0f, 0.0f,
3     0.5f, -0.5f, -0.5f, -1.0f, 0.0f, 0.0f, 1.0f, 0.0f,
4     0.5f, 0.5f, -0.5f, -1.0f, 0.0f, 0.0f, 1.0f, 1.0f,
5     -0.5f, 0.5f, -0.5f, -1.0f, 0.0f, 0.0f, 0.0f, 1.0f,
6     -0.5f, -0.5f, 0.5f, 1.0f, 0.0f, 0.0f, 0.0f, 0.0f,
7     0.5f, -0.5f, 0.5f, 1.0f, 0.0f, 0.0f, 1.0f, 0.0f,
8     0.5f, 0.5f, 0.5f, 1.0f, 0.0f, 0.0f, 1.0f, 1.0f,
9     -0.5f, 0.5f, 0.5f, 1.0f, 0.0f, 0.0f, 0.0f, 1.0f,
10    -0.5f, 0.5f, 0.5f, 0.0f, 0.0f, -1.0f, 1.0f, 0.0f,
11    -0.5f, 0.5f, -0.5f, 0.0f, 0.0f, -1.0f, 1.0f, 1.0f,
12    -0.5f, -0.5f, -0.5f, 0.0f, 0.0f, -1.0f, 0.0f, 1.0f,
13    0.5f, 0.5f, 0.5f, 0.0f, 0.0f, -1.0f, 1.0f, 0.0f,
14    0.5f, -0.5f, -0.5f, 0.0f, 0.0f, 1.0f, 0.0f, 1.0f,
15    0.5f, -0.5f, 0.5f, 0.0f, 0.0f, 1.0f, 0.0f, 0.0f,
16    0.5f, -0.5f, -0.5f, 0.0f, 0.0f, 1.0f, 1.0f, 1.0f,
17    -0.5f, 0.5f, 0.5f, 0.0f, 0.0f, 1.0f, 0.0f, 0.0f
18 };

```

Nuevamente en el archivo `main.cpp` se modificaron las funciones `GeneracionBufferCube()` y `GeneracionBufferLight()`, agregando la nueva distribución del arreglo `vertices`:

```

21 void GeneracionBufferCube()
20 {
19     glGenVertexArrays(1, &VAO);
18     glGenBuffers(1, &VBO);
17     glGenBuffers(1, &EBO);
16     glBindVertexArray(VAO);
15
14     glBindBuffer(GL_ARRAY_BUFFER, VBO);
13     glBufferData(GL_ARRAY_BUFFER, sizeof(vertices), vertices, GL_STATIC_DRAW);
12
11     glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, EBO);
10     glBufferData(GL_ELEMENT_ARRAY_BUFFER, sizeof(indices), indices, GL_STATIC_DRAW);
9
8     glVertexAttribute(0, 3, 8, 0);
7     glVertexAttribute(1, 3, 8, 3);
6     glVertexAttribute(2, 2, 8, 6);
5
4     glBindBuffer(GL_ARRAY_BUFFER, 0);
3     glBindVertexArray(0);
2 }

17 void GeneracionBufferLight()
16 {
15     glGenVertexArrays(1, &lightVAO);
14     glBindVertexArray(lightVAO);
13
12     glBindBuffer(GL_ARRAY_BUFFER, VBO);
11     glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, EBO);
10
9     glVertexAttribute(0, 3, 8, 0);
8
7     glBindBuffer(GL_ARRAY_BUFFER, 0);
6     glBindVertexArray(0);
5 }

```

En el archivo `vertexShader.vs` se agregó el nuevo `layout` con las coordenadas de la textura llamado `aTexture1`, esta misma variable se declara como salida `ourTexture1` para que el *Fragment Shader* lo pueda recibir, finalmente, en el `main()` se iguala `ourTexture1` con el valor de `aTexture1`:

```

19 #version 330 core
18 layout (location = 0) in vec3 aPos;
17 layout (location = 1) in vec3 aNormal;
16 layout (location = 2) in vec2 aTexture1;
15
14 out vec3 FragPos;
13 out vec3 Normal;
12 out vec2 ourTexture1;
11
10 uniform mat4 model;
9 uniform mat4 view;
8 uniform mat4 projection;
7
6 void main()
5 {
4     FragPos = vec3(model * vec4(aPos, 1.0f));
3     Normal = aNormal;
2     gl_Position = projection * view * model * vec4(aPos, 1.0f);
1     ourTexture1 = aTexture1;
0 }

```

En el archivo `fragmenShader.fs` se declara la variable de entrada `ourTexture1`, que representa las coordenadas de la textura, y la variable uniforme `texture1`, que será la imagen de la textura como tal. En el `main()` como prueba, simplemente se deja comentada la línea en donde se manda la salida `FragColor` con el resultado de las operaciones con la luz, y se iguala a la creación de una única textura:

```

6 #version 330 core
5 out vec4 FragColor;
4
3 in vec3 FragPos;
2 in vec3 Normal;
1 in vec2 ourTexture1;
0
1 struct Material {
2     vec3 ambient;
3     vec3 diffuse;
4     vec3 specular;
5     float shininght;
6 };
7
8 struct Light{
9     vec3 ambient;
10    vec3 diffuse;
11    vec3 specular;
12
13    vec3 pose;
14 };
15
16 uniform Material material;
17 uniform Light light;
18 uniform sampler2D texture1;
19
20 uniform vec3 lightColor;
21 uniform vec3 lightPos;
22 uniform vec3 viewPos;
23
24 void main()
25 {

```

```

5 void main()
6 {
7     vec3 ambiental = material.ambient * light.ambient;
8
9     vec3 norm = normalize(Normal);
10    vec3 lightDir = normalize(light.pose - FragPos);
11    float diff = max(dot(norm, lightDir), 0.0f);
12    vec3 difuse = (diff * material.diffuse) * light.diffuse;
13
14    vec3 viewDir = normalize(viewPos - FragPos);
15    vec3 reflectDir = reflect(-lightDir, norm);
16    float spec = pow(max(dot(viewDir, reflectDir), 0.0f), material.shininght);
17    vec3 specular = spec * light.specular * material.specular;
18
19    vec3 result = ambiental + difuse + specular;
20    //FragColor = vec4(result, 1.0f);
21    FragColor = texture(texture1, ourTexture1);
22 }

```

Finalmente, en el directorio del proyecto se agregó otro directorio llamado `Imágenes` el cual contendrá todos los archivos de imagen relacionados con las texturas de los objetos:

Name	Date modified	Type	Size
Imágenes	07/04/2023 11:59 a. m.	File folder	
img	07/04/2023 12:51 p. m.	File folder	
x64	06/04/2023 11:40 p. m.	File folder	
A7LucesMaterial.es.sln	06/04/2023 07:05 p. m.	Visual Studio Solut...	2 KB
A7LucesMaterial.es.vcxproj	06/04/2023 11:58 p. m.	VCXPROJ File	8 KB
A7LucesMaterial.es.vcxproj.filters	06/04/2023 11:58 p. m.	VC++ Project Filte...	2 KB
A7LucesMaterial.es.vcxproj.user	06/04/2023 07:05 p. m.	Per-User Project O...	1 KB
A8LucesMaterial.esTexturas.sln	07/04/2023 11:15 a. m.	Visual Studio Solut...	2 KB
A8LucesMaterial.esTexturas.vcxproj	07/04/2023 11:50 a. m.	VCXPROJ File	7 KB
A8LucesMaterial.esTexturas.vcxproj.filters	07/04/2023 11:50 a. m.	VC++ Project Filte...	2 KB
A8LucesMaterial.esTexturas.vcxproj.user	07/04/2023 11:15 a. m.	Per-User Project O...	1 KB
Cubo.h	07/04/2023 11:30 a. m.	C/C++ Header	2 KB

Al ejecutar el proyecto se puede observar el cubo cargado con la textura **caja.jpg** de manera exitosa junto con la luz:

