Alumno: Rubén Garrido Valdepeñas
Asignatura: Efectos visuales y sonoros
Curso: 2022-2023
Profesor: Rafel Pérez Vidal

PEC 2: Efectos visuales de la naturaleza, el fuego

1. Efecto

Inicialmente pensaba realizar un efecto de lava partiendo del ejemplo del agua. Tras varios intentos fallidos he buscado una forma más simple, una llama en dos dimensiones. Con esto simplifico el problema y solo tengo que trabajar con texturas en el plano 2D (bueno es 3D pero la Z es 0).

2. Ficheros necesarios

Carpeta de Assets:
- Texturas: fire-gradient.png y noiseTexture.png.
- Shaders: FireShader.fragment y FireShader.vertex

Código:
- Shader.h, Shader.cpp y Flame.cpp (main).

3. Librerías utilizadas

- SDL
- Glad
- SDL_Image

La base del proyecto es el código de ejemplo del triángulo en movimiento por lo que no hace falta añadir nada ni hacer ningún paso extra para compilar.

4. Explicación

Lo primero que tengo que hacer es definir los dos triángulos que van a formar parte de mi recuadro. Para ello es necesario definir las posiciones de cada vértice. En mi caso pondré que vayan en las posiciones -1 a 1. Quiero que el rectángulo ocupe toda la pantalla. La información que guardo de cada vértice es la posición en el plano y la textura.

```
//VBO data
GLfloat vertexData[16] =
{
	// x,y,			u,v(s,t)
	-1.0f,  1.0f,		0.0,1.0,
	1.0f,  1.0f,		1.0,1.0,
	1.0f, -1.0f,		1.0,0.0,
	-1.0f, -1.0f,		0.0,0.0
};
```

Con esto solo falta indicar la unión de vértices:

```
//EBO data
GLuint indexData[6] =
{
	0, 1, 2,   0, 2, 3
};
```

Con esto el siguiente paso es reservar memoria en la GPU para guardar esta información, solo tengo que añadir la posición del punto y la textura. Sería definir que tengo dos floats para las posiciones y dos floats para la textura (a esta le tengo que añadir el desplazamiento de los dos primeros valores):

```
glVertexAttribPointer(0, 2, GL_FLOAT, GL_FALSE, 4 * sizeof(GLfloat), NULL);
glVertexAttribPointer(1, 2, GL_FLOAT, GL_FALSE, 4 * sizeof(GLfloat), (GLvoid*)(2 * sizeof(GLfloat)));
```

Faltaría indicar también los índices:

```
glBufferData(GL_ELEMENT_ARRAY_BUFFER, sizeof(indexData), indexData, GL_STATIC_DRAW);
```

Con esto ya tengo los dos buffers, con un bind del VAO esta parte ya estaría.

Seguidamente tengo que cargar las texturas que voy a utilizar en el shader. En mi caso será una textura que utilizaré para generar el ruido y otra que tiene el degradado de colores.

```
	// Create Texture 1
	SDL_Surface* tempSurface;
	tempSurface = IMG_Load("Assets/textures/noiseTexture.png");

	glGenTextures(1, &texture);
	glBindTexture(GL_TEXTURE_2D, texture);
	GLint Mode = GL_RGB;
	if (tempSurface->format->BytesPerPixel == 4) Mode = GL_RGBA;
	glTexImage2D(GL_TEXTURE_2D, 0, Mode, tempSurface->w, tempSurface->h, 0, Mode,
GL_UNSIGNED_BYTE, tempSurface->pixels);
	glGenerateMipmap(GL_TEXTURE_2D);
	glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
	glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);

	glBindTexture(GL_TEXTURE_2D, 0);
```

Cargo la textura en la surface temporal. Como el efecto ocupará toda la pantalla me interesa que la textura se repita si su tamaño es más pequeño.

La parte de inicialización del shader es la misma que en el código de ejemplo.

La siguiente parte que he implementado es la etapa de Render aquí es bastante simple, solo tengo que borrar el fondo y utilizar el shader. En este punto le tengo que pasar las texturas y variables que requiera el shader:

```
//Sets texture
glActiveTexture(GL_TEXTURE0);
glBindTexture(GL_TEXTURE_2D, texture);
glUniform1i(glGetUniformLocation(fireShader.getID(), "noiseMap"), 0);
```

Esto sería todo lo que he añadido en el fichero principal. Ahora falta el shader, empezaré por el vertex.
Como la textura no se tiene que mover no hay mucho que hacer aquí. Solo hace falta hacer un set de la posición y pasar la posición de la textura para el fragment.

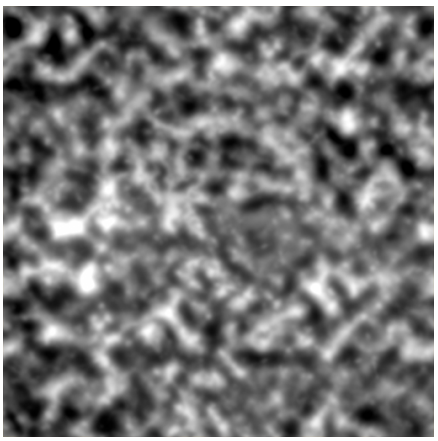
```
#version 330 core

layout (location = 0) in vec2 LVertexPos2D;
layout (location = 1) in vec2 texCoor;


out vec2 texCoord;


void main(){
	gl_Position = vec4( LVertexPos2D.x, LVertexPos2D.y, 0, 1 );
	texCoord = texCoor;
}
```

Ya solo queda el fragment. Aquí sí que hay bastantes cosas que explicar.
En primer lugar hay que saber como generar las llamas. Como se trata de un efecto en 2D se puede dar la ilusión de movimiento modificando cada píxel de la textura. Se puede hacer de dos maneras, mediante una imagen o un algoritmo.
En mi caso y siguiendo ejemplos de clase he utilizado una imagen para generar estos valores.



Solo he podido encontrar texturas en blanco y negro, es ligeramente diferente a la dudv que se utiliza en los ejemplos. Claro que usar directamente esta textura no es suficiente dado que hace falta añadirle un valor que cambie con el tiempo. Por lo que cada vez que se hace el Render le paso al shader un float que se incrementa en cada ciclo.
Con la textura y una variable que se incrementa con el tiempo falta implementar el algoritmo que genera el efecto. Es una mezcla de ruido con distorsión y se le añade esto a la textura que tiene el degradado del fondo.

El ruido se genera mediante el acceso a diferentes valores de la textura de ruido y accediendo al color rojo. Con esto lo que tengo es un efecto como de una lámpara de lava. Se mueve muy despacio y en bloques grandes y no es lo que busco exactamente. he probado a repetir esta operación varias veces acumulando los valores y el efecto generado se hace más pequeño conforme más veces lo repita:

```
distortion += texture(noiseMap, (pos * repetitions) / 64.0).r * weight;
```

Con esto tengo un valor fijo de ruido. Hace falta añadirle variabilidad. Para esto tengo la variable de tiempo que se actualiza en cada etapa de renderizado. Quiero que el fuego vaya subiendo y se mueva lateralmente. Por lo tanto este valor lo puedo añadir de forma fija en el eje de la Y. Para el eje de la X quiero que oscile por lo que es más recomendable usar trigonometría. He probado tanto seno como coseno y me funcionan bien los dos

```
texCoord.x + (offsetDistortion * cos(timef) * texCoord.y),
texCoord.y - (timef * offsetDistortion)
```

Ahora solo queda solapar el efecto con la textura.

```
vec4 colorFragment = texture(fireRamp, vec2(offsetColor, offsetColor));
```