Programación 2D
Práctica 2: Carga de texturas

En esta práctica aprenderemos a cargar y dibujar con texturas generadas a partir de ficheros de imagen.

Se incluyen en el motor dos nuevas clases, `Image` y `ResourceManager`, que deben ser añadidas al proyecto. Se deben descomentar los métodos de la clase `Renderer` relativos al dibujo con imágenes, incluir el fichero **image.h** en **renderer.cpp** y en **u-gine.h**, y en este último añadir también **resourcemanager.h**.

La práctica consta de dos partes, con una valoración máxima de 0,5 puntos cada una. Explicaremos cada una de las partes por separado.

Primera parte (0,5 puntos):

Se pide al alumno que implemente una clase `Image` con soporte para la generación y pintado con texturas. Como las llamadas a OpenGL son encapsuladas en la clase `Renderer`, añadiremos los métodos necesarios a dicha clase:

```
uint32 Renderer::GenImage(uint8* buffer, uint16 width, uint16 height);
void Renderer::BindImage(uint32 glhandle);
void Renderer::DeleteImage(uint32 glhandle);
```

Los datos pedidos por `Renderer::GenImage` los podemos obtener cargando el fichero de imagen con **STB_Image**. El método devuelve el identificador de OpenGL para la textura, que podremos utilizar en los otros dos métodos. Se deben realizar las llamadas pertinentes a OpenGL en cada método (en el caso de `Renderer::GenImage`, con los pasos vistos para la generación de texturas). Además, se deberán implementar las llamadas correctas al renderer en cada método de `Image`.

Una vez implementado el soporte de texturas en el motor, partiremos de la siguiente plantilla para realizar las tareas que se describirán a continuación:

```
int main(int argc, char* argv[]) {
	Screen::Instance().Open(800, 600, false);

	// TAREA: Cargar la imagen “data/ball.png”
	// TAREA: Centrar imagen

	while ( Screen::Instance().IsOpened() /* TAREA: Salida con ESC */ ) {
		// TAREA: Actualizar ángulo y escala de la imagen

		// TAREA: Limpiar pantalla y dibujar la imagen

		// Refrescamos la pantalla
		Screen::Instance().Refresh();
	}

	ResourceManager::Instance().FreeResources();

	return 0;
}
```

En primer lugar, se debe cargar la imagen **“data/ball.png”** utilizando la clase `ResourceManager`, que es la encargada de gestionar todos los recursos del motor que se cargan desde disco.

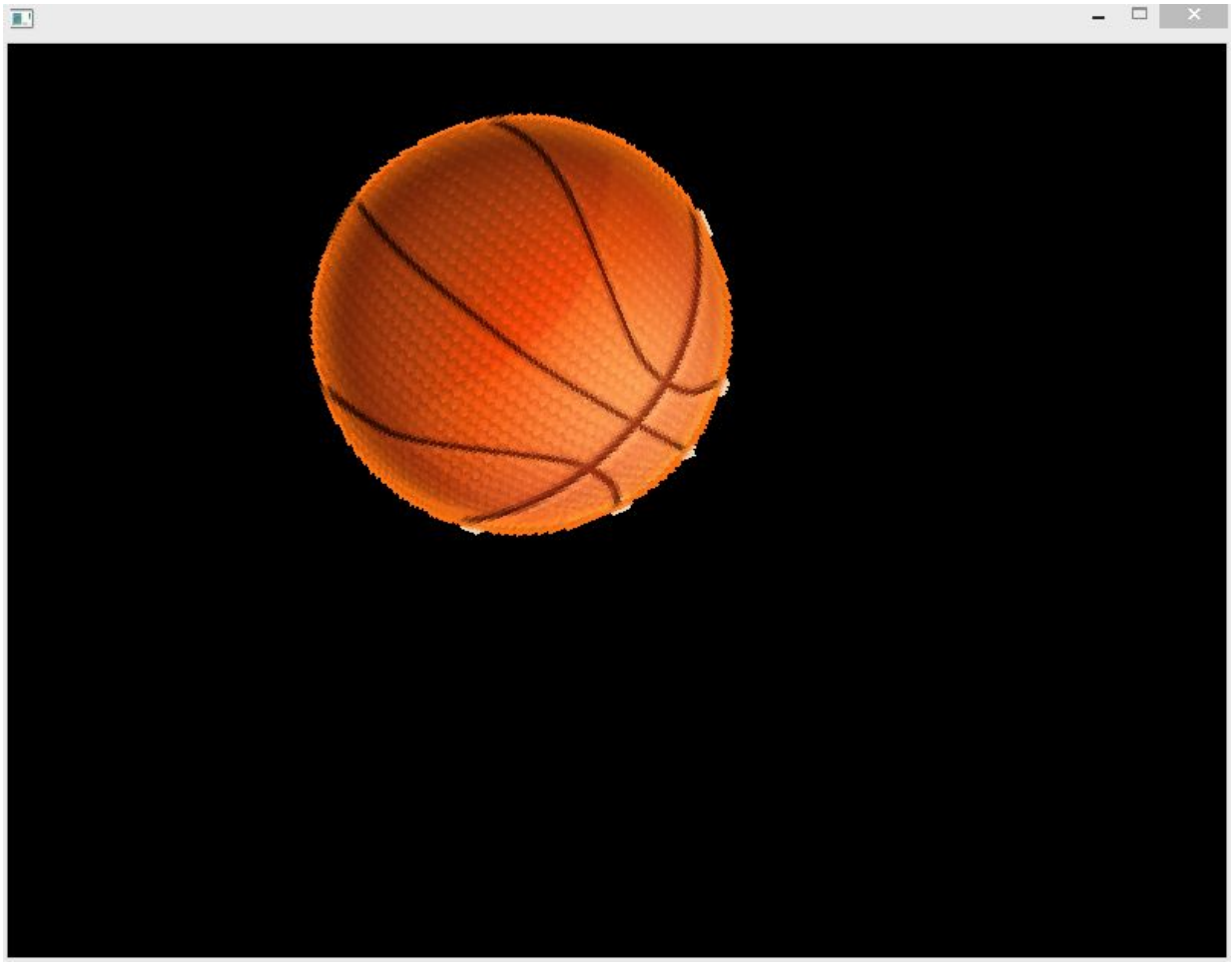
A continuación, debemos establecer el punto de referencia de la imagen en su centro. Cuando pintamos la imagen en unas coordenadas de la pantalla, este punto de anclaje es el que aparece sobre esas coordenadas. A este punto de referencia se le suele llamar “handle”, “anchor” o “pivot”.

Dentro del bucle `while`, queremos dibujar la imagen cargada en las coordenadas del puntero del ratón. La imagen debe sufrir una serie de transformaciones:

- Debe rotar continuamente a una velocidad de 30 grados por segundo.
- Debe ir modificando su escala continuamente entre 0.5 y 5.0, a un ratio de 2 puntos de escala por segundo.

Además, deberemos hacer pruebas con los parámetros de textura `GL_NEAREST` y `GL_LINEAR` para probar cómo afectan a la escala de la imagen los distintos tipos de filtrado.

El resultado del ejercicio quedará como sigue:



Segunda parte (0,5 puntos):

La segunda parte de la práctica requiere realizar manipulaciones avanzadas del buffer de imagen y de los métodos de pintado de imágenes de la clase `Renderer`. El alumno deberá investigar cómo realizar la resolución de esta parte.

Se pide soporte para generar texturas a partir de ficheros de imagen cuyo ancho y alto no sea una potencia de dos.

Para esto, una vez cargado el buffer de la imagen, comprobaremos si su ancho o alto no son potencia de dos, y deberemos generar un nuevo buffer que sí lo sea. A continuación, copiamos cada línea del buffer original en el nuevo, dejando en transparente los píxeles sobrantes y las líneas añadidas.

También se deberá modificar el método `DrawImage` de la clase `Renderer`, ya que el punto máximo en anchura y altura de nuestra textura ya no será 1.0, sino que habrá que hallar el nuevo coeficiente en función de las líneas añadidas y del mayor ancho de éstas.

Para calcular el nuevo tamaño del ancho o el alto si éste no es potencia de dos, podemos utilizar la siguiente fórmula:

```
nuevoAncho = 2^ceil(Log2(ancho))
```

Para probar que se ha realizado correctamente el ejercicio, modificaremos la primera parte de la práctica para que cargue la imagen **“data/soccer_npot.png”**, que tiene un tamaño de 100x100 píxeles.