



## Programación 2D

### Práctica 6: El mundo. Control de cámara y scroll

En esta práctica, aprenderemos a manejar los elementos de juego por medio del mundo, y a utilizar scroll para mostrar dicho mundo en pantalla.

#### Clase World

Implementaremos una clase para gestionar el mundo con los siguientes métodos (añadir también las variables miembro necesarias):

- **World**(  
float clearRed = 0.15f, float clearGreen = 0.15f, float clearBlue = 0.15f,  
const ltex\_t\* back0 = nullptr, const ltex\_t\* back1 = nullptr,  
const ltex\_t\* back2 = nullptr, const ltex\_t\* back3 = nullptr);
- float **getClearRed**() const;
- float **getClearGreen**() const;
- float **getClearBlue**() const;
- const ltex\_t\* **getBackground**(size\_t layer) const;
- float **getScrollRatio**(size\_t layer) const;
- void **setScrollRatio**(size\_t layer, float ratio);
- const Vec2& **getScrollSpeed**(size\_t layer) const;
- void **setScrollSpeed**(size\_t layer, const Vec2& speed);
- const Vec2& **getCameraPosition**() const;
- void **setCameraPosition**(const Vec2& pos);
- void **addSprite**(Sprite& sprite);
- void **removeSprite**(Sprite& sprite);
- void **update**(float deltaTime);
- void **draw**(const Vec2& screenSize);

Principalmente, la clase gestiona los siguientes elementos:

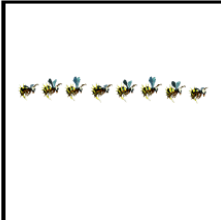
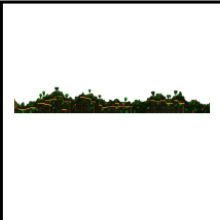

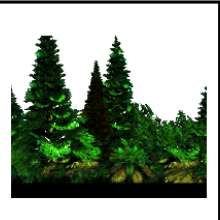
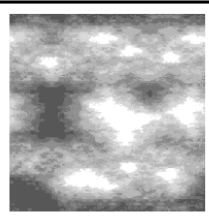
- Un color con el que limpiar el fondo antes del pintado.
- Cuatro fondos con scroll parallax.
- Un ratio de scroll para cada fondo (cuántos puntos moverse por cada punto que se mueve la cámara).
- Una velocidad de scroll automático en cada fondo (especificada en puntos x,y por segundo).
- La posición de la cámara.
- Una lista de sprites contenidos en el mundo.

En el método update, actualizaremos la lógica del mundo (básicamente es actualizar el valor del scroll automático en base a la velocidad que se estableció para cada fondo en setScrollSpeed), y llamamos al método update de todos los sprites del mundo.

En el método draw, debemos borrar el fondo con el color especificado, pintar los fondos (podemos utilizar `ltex_drawrotsized`, pintando cada imagen a pantalla completa y calculando las coordenadas de textura en base al scroll y al tamaño de cada imagen), y pintar los sprites (para que se dibujen correctamente, hay que aplicar la traslación de la cámara previamente con `lgfx_setorigin`).

## Programa principal

Utilizaremos las siguientes imágenes:

				
<code>data/bee_anim.png</code>	<code>data/level.png</code>	<code>data/trees1.png</code>	<code>data/trees2.png</code>	<code>data/clouds.png</code>

Crearemos un mundo, utilizando para los cuatro planos de fondo las imágenes "data/level.png", "data/trees1.png", "data/trees2.png" y "data/clouds.png", respectivamente. Estableceremos las siguientes propiedades:

- El fondo 0 tendrá un ratio de scroll de 1.
- El fondo 1 tendrá un ratio de scroll de 0.8.
- El fondo 2 tendrá un ratio de scroll de 0.6.
- El fondo 3 tendrá un ratio de scroll de 0.4, y una velocidad automática de -16, -8.

Crearemos un sprite con la imagen "data/bee\_anim.png" (y lo añadiremos al mundo), con 8 frames de animación en horizontal y 1 en vertical, animado a 8 fps. Debemos hacer que siga al ratón y rote exactamente igual que en la práctica 4 ("Sprites y animación"). Pero hay una novedad: tras mover el sprite, vamos a posicionar la cámara intentando que el sprite quede centrado en la pantalla siempre, pero con la restricción de que la cámara no puede salirse de los límites del nivel.

Debemos actualizar y pintar todos los elementos con los métodos `World::update` y `World::draw`.

