Control de usuario
Práctica 8 - Control de cámara. Scroll

En esta práctica, aprenderemos a manejar los elementos de juego mediante una escena, y mostrar una región de la misma utilizando scroll. Nuevamente la práctica se divide en dos partes de evaluación independiente.

Primera parte (0,5 puntos)

En esta parte de la práctica, añadiremos al motor las clases Camera y Scene. La primera de ellas tiene la siguiente interfaz:

```
class Camera {
public:
	Camera();

	virtual void SetPosition(double x, double y);
	virtual void SetX(double x);
	virtual void SetY(double y);
	virtual double GetX() const;
	virtual double GetY() const;

	virtual void SetBounds(double bx0, double by0, double bx1, double by1);
	virtual bool HasBounds() const;
	virtual double GetMinX() const;
	virtual double GetMinY() const;
	virtual double GetMaxX() const;
	virtual double GetMaxY() const; }

	virtual void FollowSprite(Sprite* sprite);

	virtual void Update();
private:
	double x, y;
```

```
        double boundx0, boundy0, boundx1, boundy1;
        Sprite* followingSprite;
};
```

Como podemos observar, la clase Camera tiene la siguiente funcionalidad:

- Unas coordenadas **x,y** dentro de la escena.
- Unos **límites** o *boundaries* dentro de la escena. Las coordenadas anteriores no pueden ser menores que **boundx0, boundy0**, ni mayores que **boundx1, boundy1 menos el ancho y alto de la pantalla** respectivamente (el tamaño de la pantalla define el *viewport* de la cámara). Cuando se fijan las coordenadas de la cámara, debemos establecer dichas coordenadas dentro de estos límites. Si **boundx0** y **boundx1** tienen el mismo valor, entonces no se han fijado límites para la cámara.
- Un **sprite** al que la cámara puede seguir opcionalmente. Si se fija el valor de este sprite, dentro del método update se debe mover la cámara para que el sprite quede centrado.

Debemos crear esta clase e importar la cabecera en los ficheros del motor que corresponda.

Respecto a la clase Scene, se proporcionan la cabecera y la implementación.

Debemos también implementar el método SetOrigin de la clase Renderer, de forma que establezca la transformación de la matriz de modelado.

Una vez completada la funcionalidad, realizaremos un ejercicio que pruebe el establecimiento de límites dentro de la escena y el seguimiento de sprites.

Debemos crear una escena cuya imagen de fondo sea el archivo **“data/background.png”** proporcionado junto con este enunciado. Los límites de coordenadas para la cámara se deben establecer al tamaño de esta imagen.

Crearemos un sprite utilizando la imagen **“data/alien.png”** que hemos empleado en prácticas anteriores, y dicho sprite se debe poder desplazar por la escena utilizando las teclas de cursor. La cámara debe seguir al sprite.

Segunda parte (0,5 puntos)

Para la segunda parte, implementaremos el soporte de scroll parallax mediante la clase ParallaxScene, derivada de Scene, que tiene la siguiente interfaz:

```
class ParallaxScene : public Scene {
public:
        ParallaxScene(Image* imageBack, Image* imageFront = 0);

        virtual const Image* GetBackLayer() const;
        virtual const Image* GetFrontLayer() const;
```

```
        virtual void SetRelativeBackSpeed(double x, double y);
        virtual void SetRelativeFrontSpeed(double x, double y);
        virtual void SetAutoBackSpeed(double x, double y);
        virtual void SetAutoFrontSpeed(double x, double y);

        virtual void Update(double elapsed, Map* map = 0);
protected:
        virtual void RenderBackground() const;
private:
        Image* backLayer;
        Image* frontLayer;
        double backX, backY;
        double frontX, frontY;
        double relBackSpeedX, relBackSpeedY;
        double relFrontSpeedX, relFrontSpeedY;
        double autoBackSpeedX, autoBackSpeedY;
        double autoFrontSpeedX, autoFrontSpeedY;
};
```

Esta clase permite definir dos planos de scroll adicionales con ratios de desplazamiento diferentes, consiguiente un efecto de scroll parallax. La velocidad relativa se refiere al ratio respecto del desplazamiento de la cámara (si lo ponemos por ejemplo a **0.5**, por cada punto que se mueva la cámara, el plano de scroll correspondiente se desplazará medio punto). La velocidad automática permite que el plano de scroll se esté desplazando a una velocidad en puntos por segundo dada independiente del movimiento de la cámara.

En el método Update debemos actualizar la información del scroll automático.

En el método RenderBackground, debemos dibujar las capas de fondo con las propiedades actuales del scroll.

Una vez implementada la clase, debemos modificar el ejercicio anterior para crear una escena con scroll parallax, utilizando las imágenes **“data/backlayer.png”** y **“data/frontlayer.png”** como planos de scroll de fondo y frente respectivamente.

El plano de fondo debe tener un desplazamiento automático de **(32, 32)** puntos, y un desplazamiento relativo de **(0.8, 0.8)**. El plano de frente debe tener un desplazamiento automático de **(-32, 32)** puntos, y un desplazamiento relativo de **(1, 1)**.