PROGRAMACION DE VIDEOJUEGOS III

Spritesheets y animaciones

En éste tutorial presentaremos funciones que nos permitirán cargar un spritesheet desde HaxeFlixel y utilizarlo para crear animaciones que nuestros sprites podrán mostrar. Utilizaremos el ejemplo desarrollado en los tutoriales anteriores como base para crear un personaje animado.

El spritesheet que utilizaremos nuestro personaje animado se muestra en la Fig. 1, cada cuadro tiene un tamaño fijo de 25 pixeles de ancho 45 de alto. El spritesheet está basado en éste otro obtenido de OpenGameArt.org.

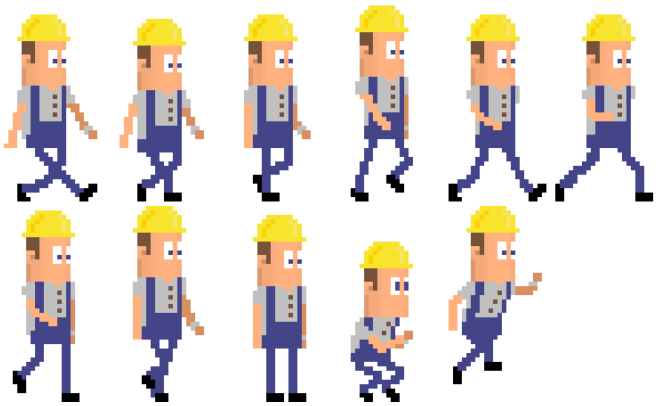

Fig. 1: Spritesheet que utilizaremos en el ejemplo

En la *Fig. 2* puede observarse el código de la clase **Character**, que guarda bastante similitud con el del ejemplo realizado en el tutorial anterior.

Para cargar un spritesheet, en lugar de especificar el archivo de imagen en el constructor del sprite, se debe hacerlo invocando la función *loadGraphic()* tal como se observa en el código. Dicha función permite realizar la carga de una imagen utilizando opciones avanzadas. En la llamada a *loadGraphic()* se especifican, además del arhivo de imagen, un segundo parámetro que indica que el archivo no contiene una imagen única sino varios frames, y otros dos parámetros más que

indican el tamaño de cada cuadro contenido en el spritesheet. El prototipo completo de la función y la explicación de cada uno de sus parámetros posibles se pueden ver en la documentación de referencia de HaxeFlixel.

```
import flixel.FlxSprite;
import flixel.FlxG;

class Character extends FlxSprite
{
    private static inline var ACCEL: Float = 65;

    public function new(X: Float, Y: Float)
    {
        super(X, Y);
        loadGraphic("assets/images/charzera_0.png",
true , 25 , 45);
        animation.add("walking", [0, 1, 2, 3, 4, 5,
6, 7], 10);
        animation.add("standing", [8], 1);
        animation.play("standing");
        drag.x = 2 * ACCEL;
        maxVelocity.x = ACCEL;
    }

    override public function update():Void
    {
        super.update();
        if(FlxG.keys.pressed.A)
        {
            acceleration.x = -ACCEL;
            animation.play("walking");
            flipX = true;
        }
        else if(FlxG.keys.pressed.D)
        {
            acceleration.x = ACCEL;
            animation.play("walking");
            flipX = false;
        }
        else
        {
            acceleration.x = 0;
            animation.play("standing");
        }
    }
}
```

Fig. 2: Contenido del archivo Character.hx del ejemplo analizado

Una vez que hemos cargado el spritesheet mediante el

ḿetodo *loadGraphic()* podremos utilizar el atributo *animation* de la clase **FlxSprite** para reproducir animaciones basadas en el mismo. El método *add()* permite crear una nueva animación, recibe como parámetros una cadena con el nombre de la animación, un arreglo de enteros con los cuadros del spritesheet que la componen, y un real que especifica la velocidad de reproducción de la animación en cantidad de cuadros por segundo.

En el constructor de la clase **Character** podemos observar que se han agregado dos animaciones para el personaje: una para cuando está caminando y otra para cuando se encuentra parado.

Una vez creadas las animaciones mediante el método *add()* del atributo *animation*, se puede indicar al sprite que la reproduzca mediante el método *play()* del mismo atributo. Dicho método recibe como único parámetro una cadena de texto con el nombre de la animación, el cual fue especificado en la llamada a *add()*.

Como puede observarse en el código, se realiza una llamada a *play()* en el constructor para especificar la animación inicial del personaje y otras tres llamadas en *update()*, cada vez que se actualiza la aceleración del personaje.

Para que el personaje animado se vea correctamente, será necesario invertir la imagen horizontalmente cuando el mismo camine o se encuentre mirando hacia la izquierda. Para lograrlo, se utiliza el atributo booleano *flipX* de la clase **FlxSprite**, tal como se puede apreciar en el código de la *Fig. 2*.

En la *Fig. 3* puede apreciarse el ejemplo terminado.

Fig. 3: Captura del ejemplo analizado a lo largo del tutorial

Descargar código del ejemplo

Como detalle adicional. Puede verse que se ha utilizado un valor constante para la aceleración del personaje. En haXe, podemos declarar constantes utilizando las palabras reservadas **static** e **inline** como es posible ver en el código mostrado. La palabra reservada **static** indica que existirá una única variable para todos las instancias de la clase, mientras que inline indica que el valor no se podrá modificar y que se reemplazará en lugar de cada ocurrencia de la variable. Para una mejor explicación, se recomienda hechar una breve mirada a las páginas de referencia del lenguaje de programación haXe sobre la palabra reservada *inline* y sobre otras palabras reservadas relacionadas con Programación Orientada a Objetos.

Resumen:

- Para cargar un spritesheet en HaxeFlixel, debe utilizarse el método *loadGraphic()* en lugar de especificar la imagen del sprite en su constructor
- El atributo *animation* permite controlar el comportamiento de la animaciones y realizar diversas acciones con el spritesheeet
- El método *add()* del atributo *animation* permite crear una nueva animación
- El método *play()* del atributo *animation* permite reproducir una animación creada anteriormente
- En haXe se pueden crear constantes anteponiendo a la declaración de una variable las palabras reservadas **static** e **inline**

Volver al índice de tutoriales...