PROGRAMACION DE VIDEOJUEGOS III

Tweening

Anteriormente aprendimos como realizar animaciones utilizando spritesheets. En el siguiente tutorial veremos cómo realizar otro tipo de animaciones muy útil a la hora de agregar detalles a nuestros juegos.

El término "tweening" es una abreviación de "in-betweening", lo cual podría traducirse como intermediación o interpolación, y se refiere al proceso de crear frames intermedios a dos cuadros clave de una animación.

Al hacer tweening, especificaremos el estado final de un objeto o variable (tomaremos como estado inicial el actual) y el tiempo que debe durar la transcición y HaxeFlixel calculará automáticamente los valores o estados intermedios. Es importante notar que éste tipo de interpolación sólo puede llevarse a cabo con valores de tipo continuo, tal como la posición, escala, ángulo o cualquier otra variable de tipo **Float** que querramos.

En el ejemplo que estudiaremos, el tweening se realizará sobre la posición y ángulo de un sprite, a pesar de que, como mencionamos antes, el proceso puede hacerse también con muchas otras de sus propiedades. Dicho ejemplo puede apreciarse en la *Fig. 1* y consiste en mover automáticamente un sprite hacia la posición en la que se hizo click con el mouse. La imagen del ejemplo fue tomada de OpenGameArg.org.

Fig. 1: Captura del ejemplo analizado a lo largo del tutorial

Descargar código del ejemplo

Una versión simplificada del código del ejemplo puede observarse en la *Fig. 2*.

Como era de esperar, el código que realiza el tweening se encuentra en el método *update()* de la escena. En primer lugar, se puede ver que el acceso al estado del mouse se realiza a través del atributo *mouse* de la clase **FlxG**, el mismo es de tipo **FlxMouse**. Los atributos *x* e *y* de dicho objeto permiten conocer la posición actual del mouse, mientras que *justPressed* permite conocer el momento exacto en que el botón izquierdo del ratón fue presionado.

Cada animación realizada mediante tweening se representa con un objeto de tipo **FlxTween**. Dichos objetos, no son creados mediante *new* como generalmente se hace, sino que son retornados por el método estático *tween()* de la misma clase. Es decir, la propia clase administra los objetos de tipo **FlxTween** creados y se encarga de reciclarlos para utilizar de manera más eficiente la memoria.

```
import flixel.FlxState;
import flixel.FlxSprite;
import flixel.FlxG;
import flixel.text.FlxText;
import flixel.tweens.FlxTween;
import flixel.tweens.FlxEase;

class PlayState extends FlxState
{
    private var sprite: FlxSprite;
    private var tween: FlxTween;
    private var easeFunction: EaseFunction;
```

```
    override public function create():Void
    {
        super.create();
        sprite = new FlxSprite(100, 100,
"assets/images/15.png");
        add(sprite);
    }

    override public function update():Void
    {
        super.update();
        if (FlxG.mouse.justPressed && tween ==
null)
        {
            var options = {complete:
onTweenEnd, ease: FlxEase.quadIn};
            sprite.angle = 0;
            tween = FlxTween.tween( sprite ,
{"x": FlxG.mouse.x - sprite.width/2.0, "y":
FlxG.mouse.y - sprite.height/2.0, "angle":
360}, 1, options);
        }
    }

    private function onTweenEnd(theTween:
FlxTween): Void
    {
        tween = null;
    }
}
```

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

El método *tween()*, como dijimos anteriormente, crea y devuelve un objeto de tipo **FlxTween**. Dicho método recibe como primer parámetro el objeto cuyos atributos se desea interpolar. El segundo parámetro consiste en un objeto que contiene pares con los nombres de los atributos que se desea interpolar y sus valores finales. El tercero de los parámetros es un real que expresa el tiempo que debe durar el proceso tweening. El último parámetro es una estructura que contiene algunas opciones adicionales.

Como último parámetro del método *tween()* hemos pasado como parámetro la estructura *options* declarada más arriba. La misma es de tipo **TweenOptions** y especifica una función (*onTweenEnd()*) que será llamada al terminar el tweening, y la función de suavizado o easing, cuyo significado explicaremos en breve. La función *tween()*, como dijimos antes, devuelve un objeto

de tipo **FlxTween** el cual puede utilizarse, entre otras cosas, para saber si el tween ha terminado o cancelarlo si no es así.

La función *onEndTween()* que será llamada al finalizar el tweening, recibe como único parámetro el tween que ha finalizado. En nuestro caso, también hemos guardado una referencia a dicho tween como atributo de la clase. Al finalizar el tween, asignaremos el valor *null* a dicha referencia para dejar claro que el atributo ya no representa un tween válido. De ésta manera podremos, cuando el usuario presione el botón del mouse, iniciar un nuevo tween únicamente en caso de que el anterior haya finalizado.

Finalmente, el campo *ease* de la estructura *options* corresponde a una "Easing Function", también conocida a veces como función de aceleración o de suavizado. Dicha función expresa con qué aceleración se realizará el movimiento. En la realidad, el movimiento generalmente no se realiza de manera lineal sino con cierta aceleración. Por ejemplo, cuando abrimos un cajón le imprimimos aceleración al principio y lo frenamos llegando al final. De manera similar, en el ejemplo se ha especificado la función *quadIn()* como aceleración del movimiento. Todas las funciones de easing están en la clase **FlxEase**. En la *Fig. 3* pueden verse las gráficas de algunas funciones de aceleración más populares.

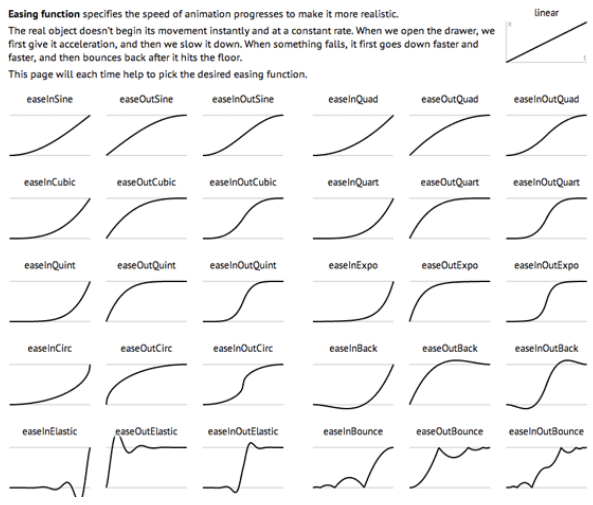

Fig. 3: Algunas de las funciones de easing más utilizadas

Al ejecutar el ejemplo de la *Fig. 1*, es posible ver que se nos permite probar entre diversas funciones de aceleración, lo cual no muestra en el código de la *Fig. 2*. Para lograr ésto, se ha utilizado un mapa que asocia cadenas de texto con los nombres de las funciones a las funciones propiamente dichas. Se recomienda analizar el código del ejemplo para apreciar la

manera de declarar un mapa en haXe.

Finalmente, recomendamos nuevamente consultar la documentación de las clases intervinientes en el ejemplo: FlxMouse, FlxTween y FlxEase; y el manual de referencia de haXe para la clase Map.

Resumen:

- El atributo *mouse* de la clase **FlxG** permite acceder al mouse
- El proceso de tweening en HaxeFlixel se hace a través de los objetos de tipo **FlxTween**, los cuales son devueltos por el método estático *tween()* de dicha clase.
- Las funciones de la clase **FlxEase** permiten variar la aceleración del proceso de tweening

Volver al índice de tutoriales...