PROGRAMACION DE VIDEOJUEGOS III

Moviendo un sprite (parte II)

En el tutorial anterior estudiamos como mover un sprite por la pantalla utilizando el teclado. En éste nuevo tutorial, introduciremos algunas modificaciones que le permitirán moverse de manera uniformemente acelerada, a diferencia del movimiento uniforme que tenía antes. Además, introduciremos algunos cambios en el código para hacerlo más ordenado y estructurado.

En primer lugar, empezaremos creando una nueva clase para nuestro personaje. El código correspondiente a la misma puede observarse en la *Fig. 1*, la hemos llamado **Character** y hereda, por supuesto, de la clase **FlxSprite**. La clase tiene dos métodos: un constructor y *update()* que, como indica la palabra reservada **override** en su delaración, sobreescribe la función con el mismo nombre definida en la clase **FlxSprite**.

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

class Character extends FlxSprite
{
    public function new(X: Float, Y: Float)
    {
        super(X, Y,
"assets/images/ImpGuy_0.jpg");
        drag.x = 200;
        maxVelocity.x = 200;
    }

    override public function update():Void
    {
        super.update();
        if(FlxG.keys.pressed.A)
        {
            acceleration.x = -200;
        }
        else if(FlxG.keys.pressed.D)
        {
```

```
            acceleration.x = 200;
        }
        else
        {
            acceleration.x = 0;
        }
    }
}
```

Fig. 1: Contenido del archivo Character.hx del ejemplo analizado

En haXe, el constructor de una clase es el método que lleva por nombre *new()* y sólo puede haber un único constructor para cada clase. En éste caso, vemos que el mismo recibe como parámetros dos valores reales (de tipo Float) con la posición del sprite, los cuales pasa al constructor de la clase padre **FlxSprite**, el cual es invocado mediante la palabra reservada **super**. El constructor también inicializa algunas variables del movimiento del sprite que explicaremos luego.

Por otra parte, y como podría imaginarse, el método *update()* será llamado automáticamente por el motor en cada cuadro para actualizar el sprite. De hecho, el método *update()* de la clase **FlxState** no hace más que invocar a *update()* para cada uno de los sprites/actores que fueron agregados a la escena mediante *add()*.

El código del método *update()* que se observa en la *Fig. 1* es muy similar al visto en el ejemplo anterior, sólo que, en este caso, lo que se modifica no es la velocidad del sprite sino su aceleración.

Es nuevamente necesario remarcar la importancia de invocar al método *update()* de la clase padre **FlxSprite**, ya que el mismo será el encargado de actualizar la posición del sprite en base a las demás propiedades modifiquemos (velocidad, aceleración, etc).

Tal cómo dijimos anteriormente, en éste caso buscamos que el sprite tenga un movimiento uniformemente acelerado. Para lograr ésto, puede utilizarse el atributo *acceleration* que, al igual que *velocity*, es de tipo **FlxPoint**.

Al modificar la aceleración, el método *update()* de la clase **FlxSprite** (que acentuamos anteriormente la de invocar) actualizará la velocidad en base a la aceleración y luego la posición en base a la velocidad, logrando el movimiento suave que esperabamos conseguir.

Éste nueva lógica de movimiento presenta algunos

problemas. Uno de ellos es evitar que la velocidad crezca indefinidamente una vez que se ha impuesto una determinada aceleración. HaxeFlixel ya resuelve automáticamente éste problema mediante el atributo *maxVelocity*, el cual fue ajustado en el constructor de la clase **Character**. Nuevamente será el método *update()* de la clase **FlxSprite** quien evite que la velocidad del sprite siga creciendo cuando su magnitud supere dicho valor.

Por otro lado, también será necesario que la velocidad empiece a disminuir una vez que la aceleración se ha anulado. Para ésto también existe otra propiedad: *drag* la cual representa una desaceleración que será aplicada al sprite únicamente cuando el valor de *acceleration* sea nulo.

```
import flixel.FlxState;


class PlayState extends FlxState
{
    private var character: Character;

    override public function create():Void
    {
        super.create();
        character = new Character(100, 100);
        add(character);
        bgColor = 0xffffffff;
    }
}
```

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

En la *Fig. 2* se puede ver el contenido del archivo *PlayState.hx*. Ahora que hemos trasladado el código de movimiento a la clase **Character**, el constructor simplemente crea un objeto de dicha clase y lo añade a la escena. El método *update()* no ha sido redefinido ésta vez, conservando su comportamiento de invocar a *update()* para cada actor.

Como detalle adicional, se puede ver que se ha modificado el color de fondo de la escena mediante el atributo *bgColor*. En HaxeFlixel, los colores se expresan como enteros en hexadecimal en la forma #AARRGGBB, donde los primeros dos dígitos corresponden a la opacidad del color y los tres siguientes a los niveles de rojo, verde y azul respectivamente.

En la *Fig. 3* puede observarse una captura del ejemplo realizado, sobre la cual se puede también hacer click para probar

el resultado del programa.

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

Descargar código del ejemplo

En éste tutorial, hemos utilizado por primera vez uno de los tipos primitivos del lenguaje de programación haXe (Float). Para conocer los tipos restantes, se recomienda hechar una breve mirada a la página http://haxe.org/ref/basic.

Al igual que en los tutoriales anteriores, dejamos una breve reseña con los conceptos más importantes tratados durante el tutorial.

Resumen:

- En haXe, el constructor de una clase es el método que lleva por nombre *new()*
- El método *update()* de la clase **FlxSprite** es llamado automáticamente en cada frame para actualizar el estado del sprite
- El método *update()* de la clase **FlxState** invoca a *update()* para cada uno de los objetos de la escena
- El método update() de la clase **FlxSprite** se encarga, entre otras cosas, de actualizar la posición del sprite en base a su velocidad y aceleración
- La aceleración de un objeto de tipo **FlxSprite** se puede modificar mediante el atributo *acceleration*
- El atributo *drag* de un objeto de tipo **FlxSprite** representa una fuerza de desaceleración que se activa únicamente cuando el valor del atributo *acceleration* es nulo

Volver al índice de tutoriales...