PROGRAMACION DE VIDEOJUEGOS III

Pooling

Tomemos como ejemplo el caso de un videojuego de tipo shooter similar al que puede observarse en el ejemplo de la *Fig. 1*, en el cual la nave principal puede emitir una gran cantidad de disparos por minuto. En casos como éste, no resultaría eficiente crear un nuevo proyectil cada vez que la nave dispara y eliminarlo cada vez que el mismo impacta sobre un enemigo o sale de la pantalla. En lugar de eso, sería mejor poseer conjunto de balas que, en lugar de ser eliminadas, simplemente dejen de actualizarse y dibujarse pero queden en memoria para luego, al necesitar disparar nuevamente, se tome alguno de los proyectiles no utilizados del conjunto y se reinicie su posición y otras propiedades para reutilizarlo, evitando de ésta manera reservar y liberar memoria con demasiada frecuencia.

Éste tipo de comportamiento se denomina "Memory Pooling" y, como se explico anteriormente, consiste en conservar los objetos en memoria cuando ya no se necesitan para reutilizarlos luego.

Fig. 1: Captura del ejemplo analizado a lo largo del tutorial, utilizar las teclas a y D para mover la nave y G para disparar

Descargar código del ejemplo

En HaxeFlixel, la clase **FlxGroup** permite realizar ésta tarea de manera muy sencilla. Como mencionamos anteriormente, el método *kill()* utilizado para matar un sprite realmente no lo quita de la escena/grupo ni de la memoria sino que modifica algunos de sus atributos de manera que el mismo ya no sea actualizado, dibujado, ni se tenga en cuenta para la detección de colisiones. El método *recycle()* de la clase **FlxGroup** permite buscar, dentro del grupo, objetos marcados como muertos para poder reutilizarlos.

El método *recycle()* recorrerá el grupo y buscará el primer elemento muerto con el atributo *alive* en falso y devolverá una referencia al mismo, dicha referencia podrá reinicializarse y reutilizarse. En caso de que no encuentre dentro del grupo ningún elemento apto para ser reautilizado, creará uno nuevo y lo agregará al grupo.

En el ejemplo de la *Fig. 1* puede apreciarse, en la esquina superior izquierda, la cantidad total de elementos que hay en el grupo de proyectiles que lanza la nave, y la cantidad de elementos vivos en dicho grupo.

En la *Fig. 2* se puede observar el contenido del archivo *Ship.hx* que contiene un fragmento del código correspondiente a la nave. Nos interesa específicamente analizar la funcion *createNewBullet()*, encargada de disparar un nuevo proyectil.

```
import flixel.FlxSprite;
import flixel.FlxG;
import flixel.group.FlxGroup;

class Ship extends FlxSprite{
    public var bullets: FlxGroup;
    private var timeToNextShoot: Float;
    private static inline var SHOTS_PER_SECOND =
7;
    private static inline var SHIP_SPEED = 150;
    private static inline var BULLET_SPEED = 200;

    public function new(){
        super(0, 0, "assets/images/ship.png");
        x = (FlxG.width - width) / 2.0;
        y = 220;
        bullets = new FlxGroup();
        timeToNextShoot = 0;
    }

    public override function update(): Void {
        super.update();
        timeToNextShoot += FlxG.elapsed;
```

```
        // si presionamos la tecla G y el
contador de tiempo es suficientemente grande
        if (FlxG.keys.pressed.G &&
timeToNextShoot>=(1.0/SHOTS_PER_SECOND))
        {
            // disparamos otra bala
            createNewBullet();
        }
    }

    private function createNewBullet()
    {
        var bullet: Bullet;
        // pedimos al grupo una bala que ya no
se esté utilizando
        bullet = cast(bullets.recycle(Bullet),
Bullet);
        // la revivimos y reposicionamos
        bullet.reset(x+this.width/2.0-
bullet.width/2.0, y);
        bullet.velocity.y = -BULLET_SPEED;
        timeToNextShoot = 0;
    }
}
```

Fig. 2: Fragmento del archivo Ship.hx correspondiente al ejemplo analizado

El método *recycle()* recibe como parámetro la clase del elemento que nos interesa reutilizar, ya que un grupo puede contener elementos de distintas clases. De ésta manera, *recycle()* recorrerá el grupo en busca de un elemento que sea de dicha clase y tenga su atributo *alive* en falso. Al obtener dicho elemento, devolverá una referencia al objeto, pero la misma será de tipo **FlxBasic** por lo que será necesario realizar un casting para convertirla al tipo **Bullet**. Una vez obtenido y convertido el proyectil a reutilizar, el método *reset()* se encarga de reinicializarlo (reinicializa el atributo *alive* entre otros) y reposicionarlo, dejándolo listo para volver a ser utilizado.

Los proyectiles, por supuesto, serán marcados como muertos al salir de la pantalla, lo cual es posible averiguar mediante el método *isOnScreen()*. El código correspondiente a la clase **Bullet** se observa en la *Fig. 3*. Ésta vez no se ha utilizado un archivo de imagen para la gráfica de los proyectiles, sino que la misma se ha generado dinámicamente utilizando el método *makeGraphic()* de la clase **FlxSprite**. Dicho método genera una imágen de un único color recibiendo como parámetro, entre otras cosas, el tamaño de la misma en píxeles y el color.

```
import flixel.FlxSprite;

class Bullet extends FlxSprite{
    public function new(){
        super();
        makeGraphic(4, 4, 0xffffffff, false,
"bullet_graph");
    }

    public override function update(): Void {
        super.update();
        // si el proyectil sale de la pantalla
        if(!isOnScreen())
            // lo marcamos como muerto
            kill();
    }
}
```

Fig. 3: Contenido del archivo Bullet.hx correspondiente al ejemplo analizado

El proceso de pooling explicado en éste tutorial es muy importante en juegos en donde existe un alto grado de creación/eliminación de objetos. Como en tutoriales anteriores, recomendamos consultar la documentación de las nuevas funciones presentadas en éste tutorial: *recycle()*, *isOnScreen()* y *makeGraphic()*.

Finalmente, para el análisis del estudiante, se deja como ejemplo un juego simple similar a *Space Invaders* basado en el ejemplo analizado anteriormente.

Fig. 4: Ejemplo de juego similar a Space Invaders, utilizar las teclas a y D para mover la nave y G para disparar

Descargar código del ejemplo

Resumen:

- El método *recycle()* de la clase **FlxGroup** busca y devuelve un elemento del grupo que ya haya sido marcado como muerto mediante la función *kill()* para poder reutilizarlo
- El método *isOnScreen()* de la clase **FlxSprite** permite saber si el mismo se encuentra dentro de la pantalla
- El método *makeGraphic()* de la clase **FlxSprite** permite crear una imagen dinámicamente

Volver al índice de tutoriales...