# Modificación de un hitter

*Este tutorial aplica a la versión 0.88c (o posteriores) de MK2*

MK2 implementa dos hitters (por el momento, exclusivos) que podemos activar desde /dev/config.h:

```
#define PLAYER_CAN_PUNCH
#define PLAYER_HAZ_SWORD
```

Activando uno de estos dos (sólo uno) estaremos activando el hitter. Por defecto, el primero define un puñetazo que puede darse a uno u otro lado, y el segundo un espadazo que puede darse a izquierda, derecha o arriba (siempre en vista lateral, ya nos curraremos algo genital cuando sea necesario).

Por supuesto, los gráficos, recorrido y comportamiento de los hitters es fácilmente modificable. Vamos a ver aquí cómo modificar la espada, los puños son más sencillos y el proceso es prácticamente el mismo.

**Modificando la trayectoria y número de frames**

El comportamiento principal de los hitters y, en este caso, la espada, está en el archivo /dev/engine/hitter.h. Casi al principio del mismo verás algo así:

```
#ifdef PLAYER_HAZ_SWORD
unsigned char swoffs_x [] = {8, 10, 12, 14, 15, 15, 14, 13, 10};
unsigned char swoffs_y [] = {2,  2,  2, 3,  4,  4,  5,  6,  7};
#endif
```

Estos dos arrays definen la posición de la espada (que es un sprite de 8x8) con respecto al sprite del protagonista durante todos los cuadros que dura su animación. En concreto, si sabemos contar, vemos que hay nueve pasos de animación.

Cuando el jugador pulsa disparo, se activa la espada. Esta activación dura nueve cuadros. Cada cuadro, la espada se coloca en la posición indicada por el índice correspondiente de esas dos arrays, siempre tomando como origen la posición del jugador.

Cuando el jugador mira hacia la derecha o a la izquierda:

- La posición "y" donde se dibuja la espada será gpy + swoffs_y [cuadro], donde gpy es la posición "y" del jugador y cuadro va de 0 a 9.
- Si el jugador mira a la derecha, la posición "x" será gpx + swoffs_x [cuadro], y si mira a la izquierda será gpx + 8 - swoffs_x [cuadro], donde gpx es la posición "x" del jugador y cuadro va de 0 a 8.

Cuando el jugador lanza la espada hacia arriba, swoffs_x y swoffs_y se intercambian para permitir que el movimiento sea vertical:

- La posición "y" donde se dibuja la espada será gpy + 6 - swoffs_x [cuadro].
- La posición "x" donde se dibuja la espada será gpx + swoffs_y [cuadro].

Modificando swoffs_x y swoffs_y podemos modificar el recorrido de la espada. Tal y como está, la espada hace una especie de curva.

También podemos aumentar o disminuir el número de frames que ocupa la animación. Vamos a poner una espada más rápida, quitaremos todos los cuadros pares y nos quedaremos sólo con cinco cuadros de animación. Hacemos esto, literalmente: nos cargamos cada valor par de los arrays. Nos quedaría:

```
#ifdef PLAYER_HAZ_SWORD
unsigned char swoffs_x [] = {8, 12, 15, 14, 10};
unsigned char swoffs_y [] = {2,  2,  4,  5,  7};
#endif
```

Como hemos cambiado el número de cuadros, tenemos que modificar un poco el código. A partir de la linea 88 está el código que detecta que han pasado todos los cuadros de la espada. Ahora tenemos 5 cuadros en lugar de 9, así que tendremos que dejar esto así:

```
#ifdef PLAYER_HAZ_SWORD
   if (hitter_frame == 5) {
#endif
```

**Modificando el "intervalo de hitter caliente"**

Llamamos "intervalo de hitter caliente" a los cuadros de la animación del hitter en los que se considera que el hitter está golpeando. Por ejemplo, en Ninjajar! el puño no golpea mientras se retrae. Con la espada, y considerando que los cuadros originalmente iban de 0 a 8, el intervalo de hitter caliente eran los frames 3 a 6, ambos inclusive:

```
FRAME:   0  1  2  3  4  5  6  7  8
GOLPEA:  NO NO NO SI SI SI SI NO NO
```

Tendremos que modificar estos intervalos en dos sitios: cuando se detecta que golpeamos un tile destructible (si los estamos usando), y cuando se detecta que golpeamos a un enemigo.

Vamos a dejar el intervalo así, ahora que tenemos menos frames:

```
FRAME:   0  1  2  3  4
GOLPEA: NO SI SI SI NO
```

En /dev/engine/hitter.h, a partir de la linea 72, es donde se detecta un tile destructibe y se manda a destruir si estamos en el intervalo de hitter caliente. Si te fijas, el intervalo está definido si el cuadro > 2 y cuadro < 7. Vamos a cambiarlo para que esté activo los cuadros 1, 2 y 3, o sea, si cuadro > 0 y cuadro < 4. Nos queda así:

```
#if defined (BREAKABLE_WALLS) || defined (BREAKABLE_WALLS_SIMPLE)
   if ((attr (gpxx, gpyy) & 16) && (hitter_frame > 0 && hitter_frame < 4))
      break_wall (gpxx, gpyy);
#endif
```

En /dev/engine/enemmods/hitter.h, en la linea 7, tenemos una detección de intervalo parecida, esta vez para detectar si la espada está "caliente" al colisionar con un enemigo. De la misma forma, lo cambiamos para que quede así:

```
            if (hitter_frame > 0 && hitter_frame < 4) {
```

**Cambiando el "punto caliente"**

La detección de colisiones del hitter con el escenario o los enemigos se hace basándonos en un pixel dentro del sprite. Por defecto, este pixel es:

- Espada hacia arriba: (hitter_x + 4, hitter_y)
- Espada hacia la izquierda: (hitter_x, hitter_y + 4)
- Espada hacia la derecha: (hitter_x + 7, hitter_y + 4)

Para cambiar esto tendremos que cambiar las asignaciones a gpxx y gpyy que hay en el bloque de código que gobierna la colocación de la espada con respecto al jugador en /dev/engine/hitter.h, o sea, aquí:

```
#ifdef PLAYER_HAZ_SWORD
   if (p_up) {
      hitter_x = gpx + swoffs_y [hitter_frame];
      hitter_y = gpy + 6 - swoffs_x [hitter_frame];
      hitter_next_frame = sprite_sword_u;
#if defined (BREAKABLE_WALLS) || defined (BREAKABLE_WALLS_SIMPLE)
      gpxx = (hitter_x + 4) >> 4; gpyy = (hitter_y) >> 4;        // <<< AQUI (x, y espada hacia arriba)
#endif
   } else {
      hitter_y = gpy + swoffs_y [hitter_frame];
#if defined (BREAKABLE_WALLS) || defined (BREAKABLE_WALLS_SIMPLE)
      gpyy = (hitter_y + 4) >> 4;                                // <<< AQUI, (y espada horizontal)
```

```
#endif
      if (p_facing) {
         hitter_x = gpx + swoffs_x [hitter_frame];
         hitter_next_frame = sprite_sword_r;
#if defined (BREAKABLE_WALLS) || defined (BREAKABLE_WALLS_SIMPLE)
         gpxx = (hitter_x + 7) >> 4;                          // <<< AQUI, (x espada hacia la derecha)
#endif
      } else {
         hitter_x = gpx + 8 - swoffs_x [hitter_frame];
         hitter_next_frame = sprite_sword_l;
#if defined (BREAKABLE_WALLS) || defined (BREAKABLE_WALLS_SIMPLE)
         gpxx = (hitter_x) >> 4;                              // <<< AQUI, (x espada hacia la izquierda)
#endif
      }
   }
#if defined (BREAKABLE_WALLS) || defined (BREAKABLE_WALLS_SIMPLE)
   if ((attr (gpxx, gpyy) & 16) && (hitter_frame > 2 && hitter_frame < 7))
      break_wall (gpxx, gpyy);
#endif
#endif
```

El punto caliente está situado en una posición bastante conveniente, dudo que tengas que cambiar esto a menos que dibujes una espada muy estrambótica.

**Cambiando el gráfico**

La espada está definida en /dev/extrasprites.h a partir de la linea 259. Las etiquetas _sprite_sword_l, _sprite_sword_r y _sprite_sword_u contienen los frames para la espada hacia la izquierda, hacia la derecha y hacia arriba, respectivamente, usando los 4 bloques de 8x8 pixels con máscaras habituales.