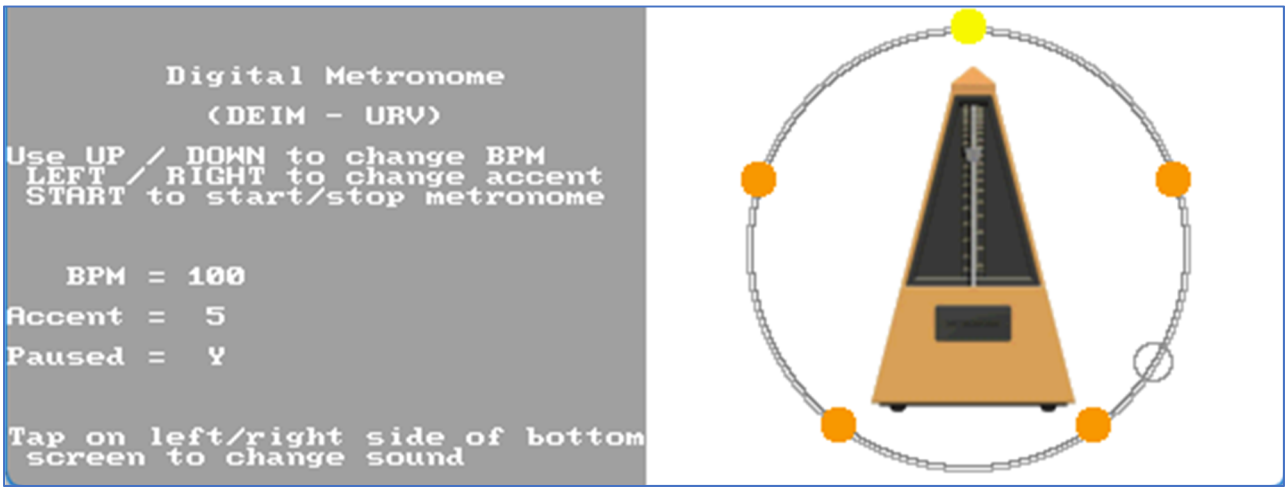
Metrónomo digital

Se propone implementar un metrónomo sobre la plataforma NDS, es decir, un programa para marcar pautas sonoras periódicas. Para realizar este programa no se necesita ningún hardware adicional, puesto que solo se requiere de las pantallas, los botones y el sonido de la NDS. La interfaz de usuario propuesta es la siguiente (las dos pantallas, una al lado de la otra):



El usuario podrá fijar el valor BPM ("beats per minute" o pulsaciones por minuto), entre 10 y 180, usando los botones UP y DOWN, en saltos de 10 unidades.

Usando los botones LEFT y RIGHT, el usuario también podrá fijar un número para formar un grupo de varios "beats". La primera marca sonora del grupo será diferente del resto. A esta marca inicial la denominaremos *acento*. Por ejemplo, para un grupo de 4 "beats", sonará una marca sonora de acento y tres marcas sonoras normales. Después se repetirá el proceso para el siguiente grupo. Esto permitirá, por ejemplo, marcar el inicio de un compás musical. Los grupos de "beats" podrán ser entre 1 y 8, en saltos de 1 unidad.

En el gráfico de la pantalla derecha los "beats" del grupo se representan con círculos, uno amarillo para el "beat" de acento y el resto en color naranja. La circunferencia grande representa el tiempo total de un grupo. La circunferencia pequeña irá girando continuamente en el sentido de las agujas del reloj, para indicar el paso del tiempo. Cada vez que la circunferencia pequeña pase por encima de un círculo, sonará el "beat" correspondiente (de acento o normal).

El número de "beats" por minuto será independiente del número de "beats" por grupo, es decir, el BPM determinará la frecuencia de los "beats" individuales, no la frecuencia de los grupos (no cuantos acentos por minuto).

El usuario podrá cambiar los valores de BPM y número de "beats" por grupo en cualquier momento, de modo que el programa adaptará la reproducción de los "beats" dinámicamente.

Por último, el usuario podrá iniciar y parar el metrónomo en cualquier momento, pulsando el botón START sucesivas veces.

Se dispone de las siguientes rutinas ya implementadas:

Rutina	*Descripción*
`inicializaciones()`	Inicializa el *hardware* (pantalla, interrupciones, etc.)
`tareas_independientes()`	Tareas que no dependen de la reproducción de las marcas sonoras ni de la representación gráfica de dichas marcas, por ejemplo, detección del compás en sonido capturado por el micrófono (tiempo de ejecución < 250 ms)
`gestionar_botones()`	Detecta los botones de la NDS y actualiza las variables de configuración en consecuencia (var. globales); devuelve un valor booleano (`unsigned char`), que será diferente de cero si se ha cambiado el BPM o el número de "beats" por grupo
`swiWaitForVBlank()`	Espera hasta el próximo retroceso vertical
`actualizar_pantallas()`	Actualiza la visualización de las variables de configuración y de los círculos que indican las posiciones de las marcas sonoras, en las pantallas correspondientes de la NDS
`activar_beat()`	Si la circunferencia pequeña está encima de un círculo, activa el sonido correspondiente (acento o normal); aunque el sonido se reproducirá durante decenas de milisegundos, esta rutina retorna en menos de 5 microsegundos
`SPR_moverSprite(int indice, int px, int py)`	Mueve el *sprite* cuyo índice se indica por primer parámetro a las coordenadas de pantalla que se indican con los otros dos parámetros
`SPR_actualizarSprites( u16* base, int limite)`	Modifica los registros OAM de uno de los procesadores gráficos, según la dirección inicial de OAM que se pasa por primer parámetro, desde el *sprite* 0 hasta el *sprite* `limite-1` (segundo parámetro menos 1).

La tarea a programar consiste básicamente en mover la circunferencia pequeña sobre el perímetro de la circunferencia grande, a la velocidad adecuada para que pase por encima de cada círculo de marca sonora a la frecuencia que indique el valor actual de BPM. La circunferencia pequeña solo se debe mover si el metrónomo no está en pausa.

Se pide explícitamente que el movimiento de dicha circunferencia pequeña, que será visualizada con el *sprite* 0 (ya inicializado), se gestione desde la RSI del *Vertical Blank*.

Se propone el siguiente programa principal, junto con las variables globales básicas:

```
typedef struct        // coordenadas de sprites
{
     short px;                  // [-64..288]
     short py;                  // [-32..224]
} t_pos;

t_pos ang_pos[360];   // vector con posiciones (px, py) para cada ángulo

unsigned char bpm = 60;          // "beats" por minuto
unsigned char accent = 4;        // divisiones para generar el acento
unsigned char paused = !0;       // metrónomo pausado? (!0: sí, 0: no)

unsigned int ang_actual = 0;     // ángulo actual (en formato Q12)
unsigned int fraccion = ???;     // incremento del ángulo en un VBL (Q12)


int main(void)
{
  inicializaciones();
  do
  {
     tareas_independientes();
     if (gestionar_botones())
     {                           // actualizar fracción según nuevos
          fraccion = ???;        // valores de bpm i accent
     }
     swiWaitForVBlank();
     actualizar_pantallas();
     // activar_beat();          // ???
  } while (1);
  return(0);
}
```

El vector `ang_pos[360]` estará inicializado a los valores `(px, py)` adecuados para posicionar un *sprite* sobre la circunferencia grande, para los 360 grados sexagesimales (enteros). La variable `ang_actual` contendrá el ángulo actual de la circunferencia pequeña, expresado en coma fija con 12 bits para la parte fraccionaria (Q12). La variable `fraccion` será el incremento (Q12) que hay que sumar al ángulo actual a cada activación del *Vertical Blank*, para conseguir que la circunferencia pequeña pase por encima de los círculos de marca sonora a la frecuencia indicada en `bpm`. Hay que decidir si la rutina `activar_beat()` se llama desde el programa principal (descomentando la línea correspondiente) o desde la RSI.

Se pide:

RSI de VBL en lenguaje ensamblador, expresión del cálculo de la fracción en lenguaje C, razonar si `activar_beat()` se debe llamar desde el programa principal o desde la RSI.