Capítulo 18

Gestión de tiempos

xLogo dispone de varias primitivas que permiten conocer la hora y la fecha o utilizar un cronómetro descendente (útil para repetir una tarea a intervalos fijos).

18.1. Las primitivas

Primitivas	Argumentos	Uso
`espera`	`número entero`	Hace una pausa en el programa, la tortuga espera `(n/60)` segundos.
`cronometro`, `crono`	`número entero`	Inicia un conteo descendiente de `n` segundos. Para saber que la cuenta ha finalizado, disponemos de la primitiva `fincrono?`
`fincronometro?`, `fincrono?`	`no`	Devuelve `"cierto` si no hay ningún conteo activo. Devuelve `"falso` si el conteo no ha terminado.
`fecha`	`no`	Devuelve una lista compuesta de 3 números enteros que representan la fecha del sistema. El primero indica el día, el segundo el mes y el último el año. `[día mes año]`
`hora`	`no`	Devuelve una lista compuesta de 3 números enteros que representan la hora del sistema. El primero representa las horas, el segundo los minutos y el último los segundos. `[horas minutos segundos]`
`tiempo`	`no`	Devuelve el tiempo, en segundos, transcurrido desde el inicio de xLogo.

En las sección 13.1 comentábamos que los dibujos eran más rápidos con la tortuga oculta que con ella en pantalla. Si añadimos las siguientes líneas al principio y al final del procedimiento:

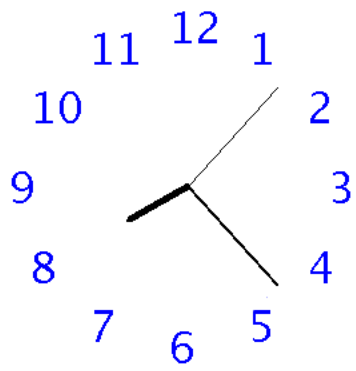
```
para nombre.proc :a :b ...
  haz "tardanza tiempo
   ...
   ...
  escribe :tardanza - tiempo
fin
```

obtendremos los segundos que tardó en ejecutarse el procedimiento.

Veamos otro ejemplo:

```
para reloj
 # muestra la hora en forma numerica (actualizada cada 5 segundos)
  si fincrono? [
   bp ponfuente 75 ot
   haz "ho hora
   haz "h primero :ho
   haz "m elemento 2 :ho
 # muestra dos cifras para los minutos (completando el 0)
   si :m - 10 < 0 [
    haz "m palabra 0 :m ]
   haz "s ultimo :ho
 # muestra dos cifras para los segundos
   si :s - 10 < 0 [
    haz "s palabra 0 :s ]
    rotula palabra palabra palabra palabra :h ": :m ": :s crono 5 ]
  reloj
fin
```

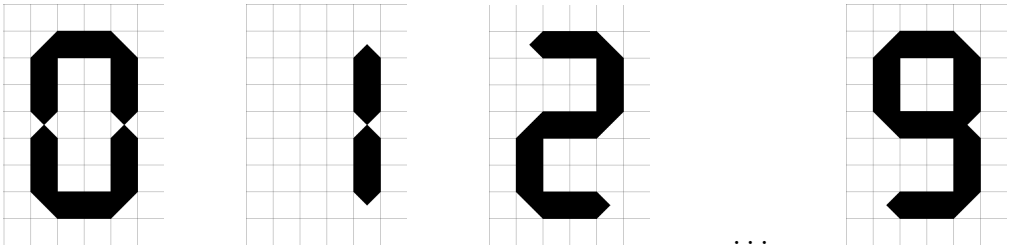
Podríamos plantearnos la forma de representar un reloj analógico:



¿Se te ocurre cómo hacerlo?

18.2. Actividad sobre las cifras de una calculadora

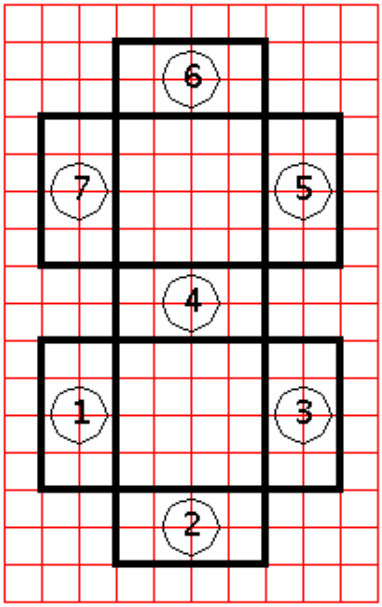
Al hablar de cifras de calculadora, nos referimos a las formas "tradicionales":



aunque nada impediría hacerlo con las formas *pixeladas* más modernas. De todas formas, simplificaremos aún más las cifras para centrarnos en los aspectos relacionados con el tiempo.

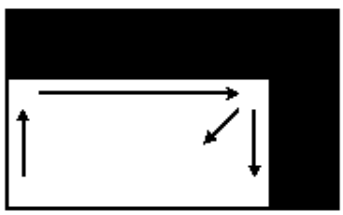
Esta actividad se basa en el hecho de que todos los números de calculadora pueden obtenerse con ayuda de un patrón sí–no:

- para dibujar un "4", "encenderemos" los rectángulos 3, 4, 5 y 7, pero no los 1, 2 y 6
- para dibujar un "8", "encenderemos" los rectángulos 1, 2, 3, 4, 5, 6 y 7.
- para dibujar un "3", encenderemos los rectángulos 2, 3, 4, 5 y 6, pero no los 1 y 7
- ...



18.2.1. El programa

Crearemos un rectángulo sólido de modo recursivo:



```
para rectangulo :alto :ancho
  si :ancho = 0 | :alto =0 [alto]
  repite 2
    [ avanza :alto giraderecha 90
      avanza :ancho giraderecha 90]
  rectangulo :alto-1 :ancho-1
fin
```

Supondremos aquí que la tortuga parte de la esquina inferior izquierda. Vamos a definir un procedimiento llamado `cifra` que lee 7 argumentos `:a`, `:b`, `:c`, `:d`, `:e`, `:f` y `:g`.

- Cuando `:a` vale `1`, dibuja el rectángulo 1. Si `:a` vale `0`, no se dibuja.
- Cuando `:b` vale `1`, dibuja el rectángulo 2. Si `:b` vale `0`, no se dibuja.
- Cuando `:c` vale `1`, dibuja el rectángulo 1. Si `:c` vale `0`, no se dibuja.
- ...

Se obtiene el siguiente procedimiento:

```
para cifra :a :b :c :d :e :f :g
# Dibujamos el rectangulo 1
  si :a=1 [rectangulo 160 40]
# Dibujamos el rectangulo 2
  si :b=1 [rectangulo 40 160]
  sl gd 90 av 120 gi 90 bl
# Dibujamos el rectangulo 3
  si :c=1 [rectangulo 160 40]
  sl av 120 bl
# Dibujamos el rectangulo 5
  si :e=1 [rectangulo 160 40]
# Dibujamos el rectangulo 4
  gi 90 sl re 40 bl
  si :d=1 [rectangulo 160 40]
# Dibujamos el rectangulo 6
  gd 90 sl av 120 gi 90 bl
  si :f=1 [rectangulo 160 40]
# Dibujamos el rectangulo 7
  sl av 120 gi 90 re 40 bl
  si :g=1 [rectangulo 160 40]
fin
```

18.2.2. Creación de una pequeña animación

Vamos a simular una cuenta atrás, que consiste en hacer aparecer sucesivamente las cifras de 9 a 0 en orden decreciente.

```
para cuentatras
   bp ot cifra 0 1 1 1 1 1 1 espera 60
   bp ot cifra 1 1 1 1 1 1 1 espera 60
   bp ot cifra 0 0 1 0 1 1 0 espera 60
   bp ot cifra 1 1 1 1 0 1 1 espera 60
   bp ot cifra 0 1 1 1 0 1 1 espera 60
   bp ot cifra 0 0 1 1 1 0 1 espera 60
   bp ot cifra 0 1 1 1 1 1 0 espera 60
   bp ot cifra 1 1 0 1 1 1 0 espera 60
   bp ot cifra 0 0 1 0 1 0 0 espera 60
   bp ot cifra 1 1 1 0 1 1 1 espera 60
fin
```

Pequeño problema: hay un efecto de parpadeo desagradable durante la creación de cada cifra. Para suavizarlo se van a utilizar las primitivas `animacion` y `refrescar`.
Se obtiene el siguiente programa modificado:

```
para cuentatras
# Entramos en modo animacion
  animacion
  bp ot cifra 0 1 1 1 1 1 1 refrescar espera 60
  bp ot cifra 1 1 1 1 1 1 1 refrescar espera 60
  bp ot cifra 0 0 1 0 1 1 0 refrescar espera 60
  bp ot cifra 1 1 1 1 0 1 1 refrescar espera 60
  bp ot cifra 0 1 1 1 0 1 1 refrescar espera 60
  bp ot cifra 0 0 1 1 1 0 1 refrescar espera 60
  bp ot cifra 0 1 1 1 1 1 0 refrescar espera 60
  bp ot cifra 1 1 0 1 1 1 0 refrescar espera 60
  bp ot cifra 0 0 1 0 1 0 0 refrescar espera 60
  bp ot cifra 1 1 1 0 1 1 1 refrescar espera 60
# Volvemos al modo de dibujo habitual
  detieneanimacion
fin
```

18.3. Ejercicios

1. Diseñemos un juego. A ver quién "cuenta" mejor un minuto mentalmente.

   a) Debe haber un botón de "Inicio" (sólo nos preocupa el *clic* del ratón)

b) Al hacer *clic* sobre el botón, la tortuga puede:

- Guardar el valor de `tiempo`
- Guardar la `hora` actual

c) Permanece en espera hasta que se hace un segundo *clic*. En ese momento, calcula la diferencia entre el `tiempo` o la `hora` almacenados y muestra si te has pasado, te has quedado corto o has acertado.

Puedes añadir cualquier efecto que deje claro cualquiera de los tres resultados (color de pantalla, de letra, música, ...)

2. Diseña un programa que se comporte como la alarma de un reloj:

   a) Abra una ventana que pida la hora a la que debe sonar

   b) Compruebe la hora recursivamente hasta que coincida con la introducida

   c) Cuando coincidan la hora real y la introducida, que suene una melodía

3. Recupera el procedimiento `bisiesto?`, y haz que xLogo te diga si el año actual es bisiesto o no.

4. Diseña un procedimiento que te pida la fecha de tu cumpleaños y determine cuántos días faltan para él.

   Si hoy es tu cumpleaños, que suene el "Cumpleaños Feliz".

5. Diseña un procedimiento que determine qué día de la semana es hoy.

   (Dificultad alta) Amplíalo para que te pida una fecha cualquiera y diga qué día de la semana le corresponde. Para ello:

   a) Elige una fecha de referencia de la que sepas qué día de la semana es.

   b) Observa que los 365 días que tiene un año son:

   $$365 = 364 + 1 = 7 \times 52 + 1$$

   es decir, cada año normal desplaza un día el día de la semana de una fecha fija: por ejemplo, el 23 de Enero de 2006 fue Lunes, pero el mismo día de 2007 fue Martes.

   c) Ten en cuenta que los años bisiestos desplazan dos días en vez de uno

   d) Finalmente, haz un razonamiento análogo para cada mes, estudiando sus respectivos días