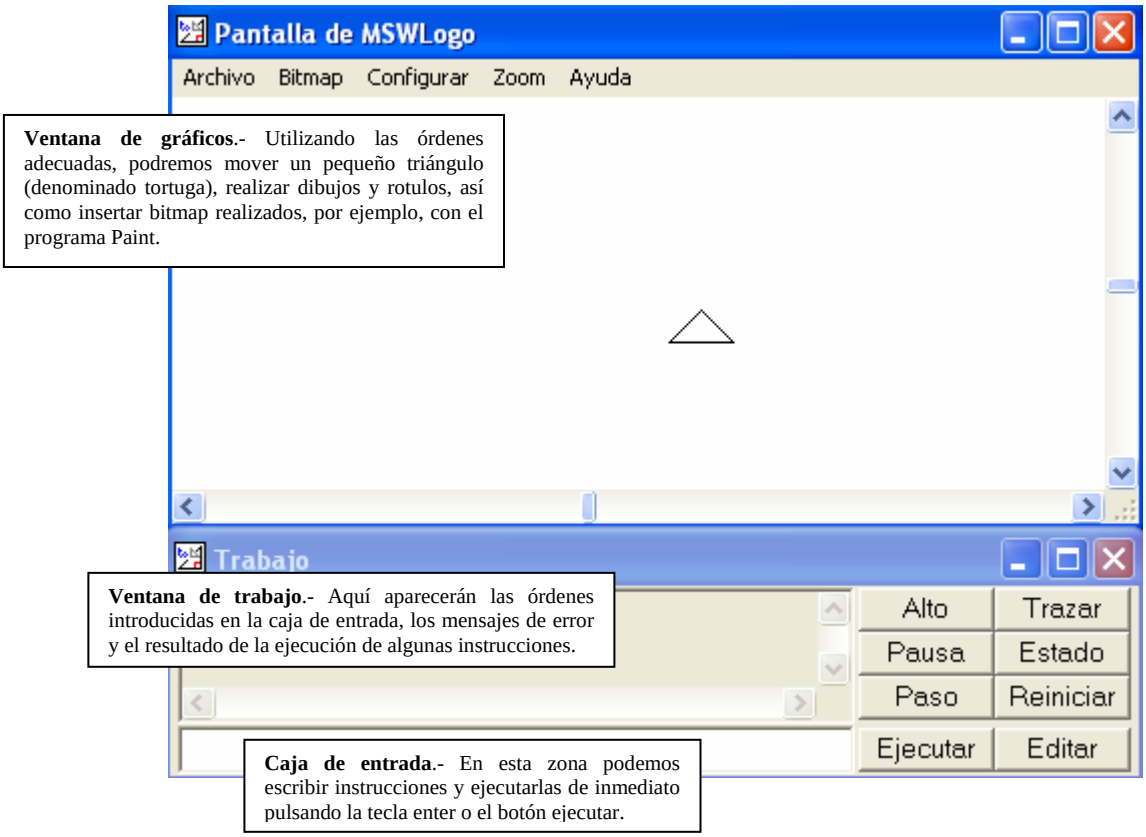
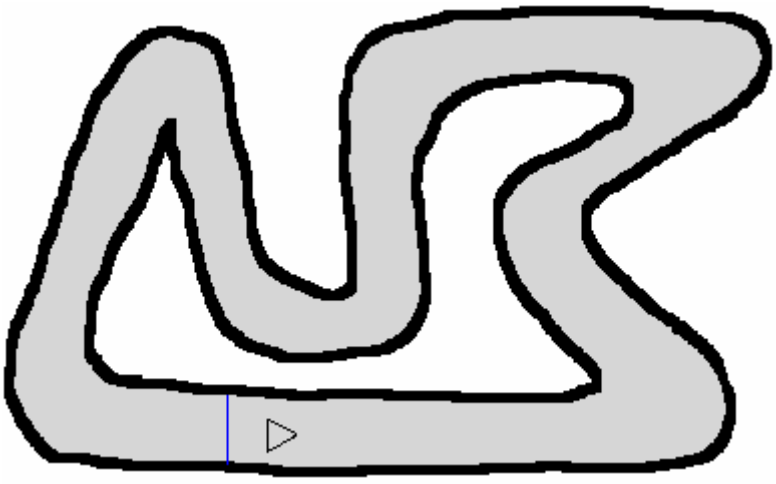
INTRODUCCIÓN

MSWLogo es un lenguaje de programación que nos permite construir programas que pueden ser ejecutados por el ordenador. Estos programas están formados por una sucesión de órdenes o instrucciones que el ordenador es capaz de entender y, en consecuencia, realizar las acciones que dichas ordenes le indiquen. Combinando adecuadamente su repertorio de instrucciones podemos hacer juegos, programas matemáticos, gráficos, programas para controlar pequeños motores o bombillas, etc.

En la figura siguiente vemos la pantalla que se obtiene al abrir MSWLogo y una descripción de las zonas en que se divide.



En los apartados siguientes vamos a ir conociendo las características de MSWLogo y algunas de sus instrucciones, con el objetivo de realizar un juego de coches en el que podamos conducir la tortuga, con ayuda del teclado, a través de un circuito dibujado previamente con el programa Paint.



La tortuga llevará un movimiento permanente de avance que podremos cambiar, girándola a izquierda o derecha, con las teclas correspondientes del teclado numérico. Tendremos que procurar dar vueltas al circuito sin chocar con los límites negros, ya que, si se tocan esos bordes haremos que el coche sea penalizado volviendo a la línea de salida.

También aprenderemos a contar y visualizar el número de vueltas que dé la tortuga y el tiempo que tarda en recorrerlas.

MOVIENDO LA TORTUGA

Para mover la tortuga y cambiar su dirección Logo cuanta con unas instrucciones muy sencillas que vamos a analizar a continuación. Estas instrucciones y los ejemplos correspondientes podemos probarlos escribiéndolos en la caja de entrada y pulsando ejecutar.

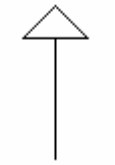
borrapantalla

bp Se borra la pantalla de gráficos y la tortuga vuelve a su posición inicial.

avanza distancia

av distancia Mueve la tortuga hacia delante la distancia especificada

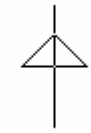
Ejemplo: `av 100`



retrocede distancia

re distancia Mueve la tortuga hacia atrás, en su dirección actual, la distancia especificada.

Ejemplo: `av 100 re 50`



giraderecha ángulo

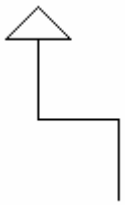
gd ángulo La tortuga gira hacia la derecha (en el sentido de las agujas del reloj) el ángulo especificado, medido en grados.

Ejemplo: `gd 90`
Ejemplo: `av 40 gd 45 av 20`

giraizquierda ángulo

gi ángulo La tortuga gira hacia la izquierda (en el sentido contrario a las agujas del reloj) el ángulo especificado, medido en grados.

Ejemplo: `gi 60`
Ejemplo: `av 40 gi 90 av 40 gd 90 av 40`



subelapiz

sl Sube el lápiz haciendo que la tortuga no dibuje al moverse.

Ejemplo: `sl av 50 gd 90 av 50`

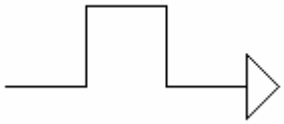
bajalapiz

bp Baja el lápiz haciendo que la tortuga dibuje al moverse. Es la situación por defecto al iniciar el programa.

Ejemplo: `gd 90 av 50 sl av 50 bl av 50`

ACTIVIDADES:

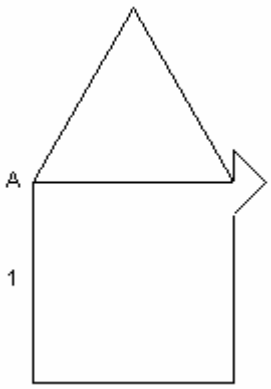
1. Escribir las instrucciones necesarias para dibujar un rectángulo de dimensiones 50 de alto y 100 de largo.
2. Introducir las instrucciones necesarias para realizar el dibujo adjunto. Las líneas son de 40 puntos.



3. Completar las instrucciones para dibujar un triángulo equilátero de lado 100. Tener en cuenta que sus ángulos internos son de 60º, aunque ese no es el valor que hay que poner en las instrucciones gi

```
bp gd 90 av 100
gi    av 100
gi    av 100
```

4. Introducir las instrucciones necesarias para realizar el dibujo adjunto. Las dimensiones de todas las líneas es 100. Comenzar desde el punto A, para dibujar en primer lugar la línea 1.



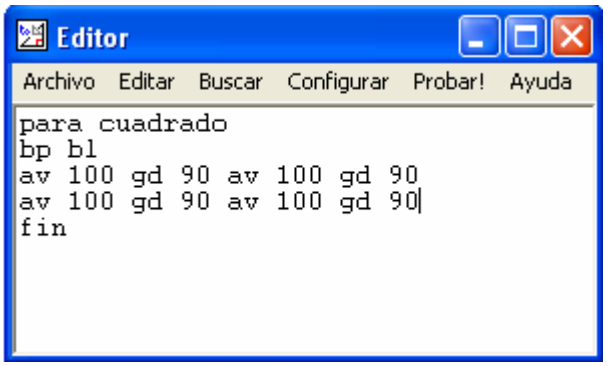
PROCEDIMIENTOS

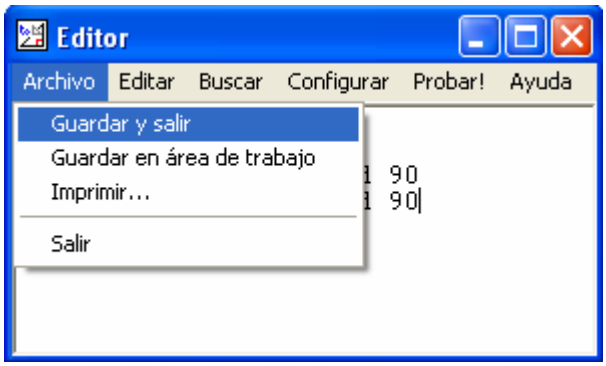
Tal como hemos visto en el apartado anterior, a través de la caja de entrada Logo ejecuta las instrucciones sueltas conforme se van introduciendo. Para crear un verdadero programa tenemos que darle un nombre y conseguir que todas las instrucciones que lo forman se ejecuten cuando escribamos ese nombre en la caja de entrada.

Pues bien, un procedimiento es un conjunto de instrucciones, agrupadas bajo un nombre, que serán ejecutadas al escribir el nombre del procedimiento en la caja de entrada. Su estructura es la siguiente:

```
para nombre del procedimiento
        .........
instrucciones que lo forman
        .........
fin
```

Para crear un procedimiento se utiliza el editor con que cuenta MSWLogo, a él se accede pinchando en el botón Editar, situado en la parte inferior derecha. Como ejemplo vamos a realizar un procedimiento para dibujar un cuadrado de lado 100. Una vez escrito, pulsamos Guardar y Salir para que el procedimiento quede almacenado en la memoria del ordenador y podamos ejecutarlo poniendo su nombre en la caja de entrada.



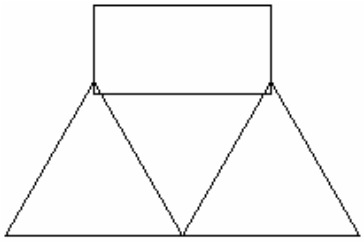


Después de escribir un procedimiento y guardarlo en la memoria del ordenador, éste puede ser tratado como si hubiésemos creado una nueva instrucción y, como tal, lo podemos ejecutar llamándolo desde otros procedimientos. En general, los programas se suelen dividir en varios procedimientos que son llamados o ejecutados desde uno principal o inicial.

ACTIVIDADES:

5. Realizar un procedimiento denominado rectangulo y otro llamado triangulo que dibujen un rectángulo y un triángulo similares a los de las actividades 1 y 2 anteriores.

6. Escribir un procedimiento denominado figura que realice el dibujo adjunto. Los triángulos y el rectángulo se harán llamando a los procedimientos del ejercicio anterior.



LA TORTUGA EN MOVIMIENTO PERMANENTE.

Nuestro vehículo del juego de coches (la tortuga) tiene que arrancar y avanzar de manera constante, nosotros sólo le haremos girar accionando las teclas correspondientes del teclado numérico. Para dotar a la tortuga de este avance permanente vamos a utilizar la instrucción siempre:

`siempre` [instrucciones]

Repite constantemente las instrucciones encerradas entre los corchetes, hasta que pulsemos el botón Alto.

Ejemplo: `siempre [av 1]`

En el ejemplo anterior, la instrucción avanza 1 se repite permanentemente. Si la probamos en el ordenador veremos que la tortuga se mueve a gran velocidad, casi no podremos verla. Para disminuir la velocidad del movimiento podemos incluir la instrucción espera:

`espera` tiempo

Detiene la ejecución de un procedimiento durante el tiempo especificado (éste se expresa en 1/60 segundos), una vez transcurrido continúa con la ejecución del resto de instrucciones.

Ejemplos:

`espera 1`	Detiene el programa durante 1/60 segundos (0’0166 segundos).
`espera 60`	Detiene el programa durante 1 segundo.
`espera 300`	Detiene el programa durante 5 segundos.
`espera 3600`	Detiene el programa durante 1 minuto.

En definitiva, para mover nuestro vehículo introducimos la instrucción espera 1 para provocar un retraso entre los avances de la tortuga. Así lo hemos hecho en un procedimiento que denominamos juego que provoca un avance lento de la tortuga:

```
para juego
sl
siempre [av 1 espera 1]
fin
```

En muchos casos nos interesará que una o varias instrucciones se repitan, pero no de manera indefinida, sino un número determinado de veces. Para ello se utiliza la instrucción repite:

`repite` nº de veces [primitivas a repetir]

Repite el nº de veces indicado las órdenes encerradas entre corchetes.

Ejemplo: Para dibujar un cuadrado podemos repetir 4 veces las órdenes av 100 gd 90. Es decir:

```
repite 4 [av 100 gd 90]
```

Puesto que el ordenador tarda un cierto tiempo en ejecutar las instrucciones, si repetimos "nada" un determinado número de veces conseguiremos provocar un retraso que, en este caso, dependerá de la rapidez del ordenador y habrá que determinar haciendo varias pruebas. Si utilizando espera 1 el movimiento es muy lento podemos modificar el procedimiento anterior de la forma siguiente:

```
para juego
sl
siempre [av 1 repite 2000 []]
fin
```

ACTIVIDADES:

7. Hacer un procedimiento denominado parpadeo que repita permanentemente el siguiente ciclo: se dibuja un cuadrado de lado 50 (hacer un procedimiento llamado cuadrado), transcurrido medio segundo se borra y, tras otro medio segundo, se vuelve a iniciar el ciclo.

8. Realizar un procedimiento (lo llamaremos apilar) que dibuje 4 cuadrados de lado 20 uno encima del otro, pero separados una distancia de 5 puntos (utilizar la instrucción repite).

9. Escribir un procedimiento llamado lado que dibuje una línea vertical de 100 puntos lentamente y otro, denominado cuadrado_lento, que dibuje un cuadrado de lado 100 lentamente.

CONTROL DESDE EL TECLADO

El siguiente paso para crear nuestro juego es controlar la dirección del movimiento de la tortuga desde el teclado del ordenador. Para poder utilizar el teclado como elemento de control tenemos que usar dos instrucciones ponfoco y ponteclado:

`ponfoco` [nombre de la pantalla que queremos tener activa]

Pasa el control a la pantalla especificada: Pantalla de MSWLogo o Trabajo.

Ejemplo: `ponfoco [Pantalla de MSWLogo]`
Esta instrucción es necesaria como paso previo a activar los eventos del teclado.

`ponteclado` [instrucciones tecla bajada]

Activa los eventos del teclado, es decir, cada vez que pulsemos una tecla Logo detendrá lo que esté haciendo en ese momento y ejecutará las instrucciones y/o procedimientos escritos en el corchete, posteriormente continuará con el trabajo que había parado.

Ejemplo: `ponteclado [gd 20]` cada vez que pulsemos una tecla la tortuga gira 20º a la derecha

Nuestro procedimiento juego puede ser ampliado para que al pulsar sobre cualquier tecla la tortuga gire 20º a la derecha:

```
para juego
bp sl
ponfoco [Pantalla de MSWLogo]
ponteclado [gd 20]
siempre [av 1 repite 2000 []]
fin
```

En el procedimiento anterior la tortuga gira siempre a derecha al pulsar cualquier tecla, sin embargo, para controlar un vehículo necesitamos que al accionar el número 4 del teclado numérico se produzca un giro a izquierda y al pulsar el número 6 el giro sea hacia la derecha. A lo largo de los próximos apartados veremos como conseguirlo.

ACTIVIDADES:

10. Escribir un procedimiento que dibuje una línea vertical de 20 puntos cada vez que pulsemos una tecla (las líneas estarán separadas unas de otras 10 puntos).

OBTENER Y GUARDAR LA TECLA PULSADA. VARIABLES.

Vamos a ver, a continuación, dos instrucciones que nos permiten conocer la tecla que hemos pulsado y la forma de guardar el carácter correspondiente a la tecla en la memoria del ordenador, dentro de una variable.

leercar

lc Devuelve el valor ASCII (número entre 1 y 127) de la ultima tecla pulsada.

caracter código ASCII

car código ASCII Devuelve el carácter (letra, número, signo,…) correspondiente al código ASCII indicado.

Estas dos instrucciones las usamos de manera combinada, es decir:

car lc Así, obtenemos el carácter correspondiente al código de la última tecla pulsada.

La letra, número o símbolo obtenido con las instrucciones anteriores tenemos que almacenarla en algún lugar, para ello utilizamos las variables:

Una variable es una porción de la memoria del ordenador donde podemos guardar algo: un número, una palabra, un carácter, una frase, etc. A las variables hay que asignarles un nombre y un valor inicial (su contenido, que posteriormente podrá ser modificado). Para crear una variable se utiliza la instrucción haz:

haz "nombre de la variable valor asignado

Ejemplo: `haz "operando 5` define la variable operando y le asigna un valor inicial de 5
`haz "tecla car lc` define la variable tecla y, a través de car lc, le asigna el carácter de la tecla pulsada, que queda almacenado en la variable.

Para referirnos, posteriormente, al valor de una variable, es decir, a lo que guarda en su interior, tenemos que colocar dos puntos (:) inmediatamente delante de su nombre.

Vamos a modificar nuestro juego de coches para que cada vez que se pulsa una tecla se ejecute un procedimiento que denominaremos control. De momento éste procedimiento lo único que hace es guardar el carácter de la tecla pulsada en la variable "tecla. Posteriormente, como veremos en el siguiente apartado, tendrá que girar la tortuga a izquierda o derecha según que la tecla pulsada sea el número 4 o el número 6.

```
para juego
bp sl
ponfoco [Pantalla de MSWLogo]
ponteclado [control]
siempre [av 1 repite 2000 []]
fin

para control
haz "tecla car lc
fin
```

TOMANDO DECISIONES. SI

En la mayoría de los programas el ordenador necesita tomar decisiones; es decir, decidir la realización de unas actuaciones u otras en función del resultado de una operación, de que un número sea mayor o menor que otro, de que una condición sea verdadera o falsa, etc. En nuestro juego necesitamos que la decisión (girar a izquierda o derecha) se tome dependiendo del contenido de la variable "tecla, es decir, del carácter correspondiente a la tecla pulsada. Para ello se utiliza la instrucción si:

si condición [instrucciones]

Si la condición es verdadera se ejecutan las instrucciones comprendidas entre los corchetes. Si la condición es falsa se pasa a la siguiente instrucción del programa sin ejecutar las encerradas entre los corchetes. Las condiciones pueden ser de distintos tipos: =, >, <, etc.

Ya podemos dar un salto importante en nuestro programa. Con los procedimientos siguientes la tortuga está en movimiento permanente y con las teclas numérica 4 y 6 podemos hacerla girar a derecha e izquierda:

```
para juego
bp sl
ponfoco [Pantalla de MSWLogo]
ponteclado [control]
siempre [av 1 repite 2000 []]
fin
```

```
para control
haz "tecla car lc
si :tecla = 4 [gi 20]
si :tecla = 6 [gd 20]
fin
```

Estos procedimientos debemos guardarlos en el disco duro del ordenador utilizando la opción Guardar como… del menú Archivo (poner como nombre del archivo "juego de coches"). En cualquier momento, podremos acceder a ellos con la opción Cagar… del mismo menú.

ACTIVIDADES:

11. Sabemos que la velocidad de la tortuga depende del número que pongamos en la instrucción repite, cuanto más grande sea más lenta irá la tortuga, ya que el retraso que se produce es mayor. Queremos ampliar el programa anterior para poder **controlar la velocidad de la tortuga**: si pulsamos la tecla x la tortuga se debe mover más rápida, si pulsamos la tecla z el movimiento será más lento. Para conseguirlo, tenemos que crear una variable, denominada por ejemplo "velocidad, cuyo valor cambie al pulsar las teclas x y z; el contenido de esta variable lo pondremos en la instrucción repite para provocar el retraso.

    Es decir, con la instrucción haz creamos la variable "velocidad y le damos un valor inicial de 2000. Si pulsamos la tecla x utilizamos la instrucción haz para cambiar el valor de "velocidad a 800 (el movimiento será más rápido) y si pulsamos la tecla z lo cambiamos por 4000 (el movimiento será más lento). En la instrucción repite pondremos el contenido de la variable, poniendo dos puntos delante de su nombre :velocidad

DIBUJAR EL CIRCUITO. CARGADIB.

El circuito por el que se va a mover la tortuga lo dibujaremos previamente con el programa Paint, teniendo presente que los bordes deben realizarse en color negro y la línea de salida en color azul (del menor grosor posible). Una vez hecho, lo guardamos en la misma carpeta en la que hemos almacenado los procedimientos anteriores con el nombre de "circuito" y la extensión bmp.

Para llevar el dibujo a la pantalla de MSWLogo utilizamos la instrucción cargadib:

cargadib nombrebitmap

Carga el bitmap que se indica en la entrada que debe ser una palabra. Esta instrucción es similar a la opción Cargar del Menú Bitmap. Nombrebitmap es el nombre del fichero que contiene el bitmap.

Ejemplo: `cargadib "circuito.bmp` El archivo debe estar en la misma carpeta que el procedimiento que lo utilice.

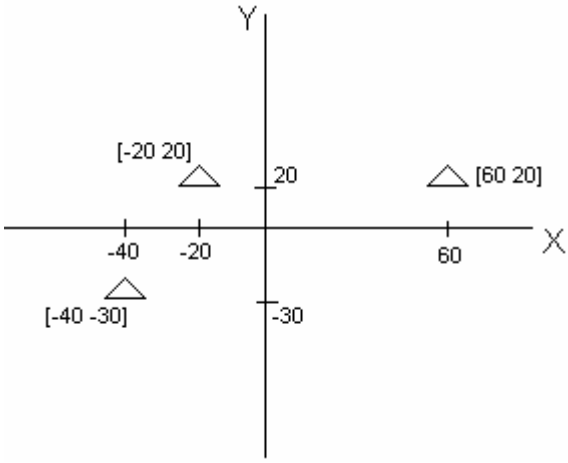
COLOCAR LA TORTUGA EN LA SALIDA. PONPOS.

La tortuga se sitúa inicialmente en el centro de la pantalla, posición que no coincidirá con la línea de salida del circuito que hayamos cargado. Para colocarla en ese lugar podemos usar las instrucciones de movimiento ya conocidas, av, re, gd, gi, etc., aunque resulta más inmediato utilizar una nueva instrucción que permite colocar la tortuga en cualquier posición utilizando sus coordenadas x e y:

En la pantalla podemos imaginar dos ejes de coordenadas estando la tortuga inicialmente en el origen, es decir, en la posición x=0 y=0, en la sintaxis de Logo [0 0].

En la figura podemos ver diversas posiciones de la tortuga y sus coordenadas.

La instrucción que nos permite mover la tortuga utilizando sus coordenadas es:

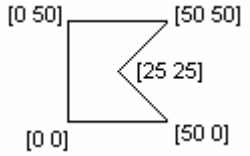


ponpos [coordenadaX coordenadaY]

Ubica la tortuga en el punto cuyas coordenadas se indican. Si el lápiz está bajado dibuja una línea recta en su desplazamiento.

Ejemplo: Vamos a realizar el dibujo de la figura moviendo la tortuga a través de sus coordenadas:

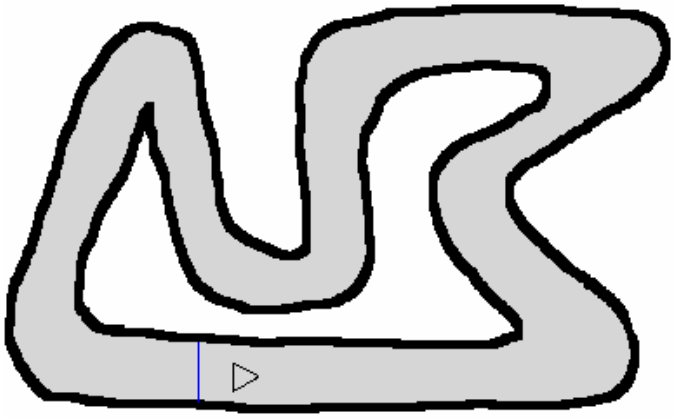
```
ponpos [0 50] ponpos [50 50]
ponpos [25 25] ponpos [50 0]
ponpos [0 0]
```



El juego de coches queda como se muestra a continuación:

```
para juego
bp sl
ponfoco [Pantalla de MSWLogo]
ponteclado [control]
cargadib “circuito.bmp
ponpos [-210 130] gd 90
siempre [av 1 repite 2000 []]
fin

para control
haz “tecla car lc
si :tecla = 4 [gi 20]
si :tecla = 6 [gd 20]
fin
```



Utilizando ponpos llevamos la tortuga de su posición inicial a la salida del circuito. Las coordenadas dependerán del circuito dibujado (se prueba)

ACTIVIDADES:

12. Dibujar un cuadrado de 200 puntos de lado. Su centro debe ser el punto de coordenadas [0 0] y la tortuga quedará situada en ese lugar.

DETECTAR EL CHOQUE CON LOS BORDES. LOS COLORES. PIXEL

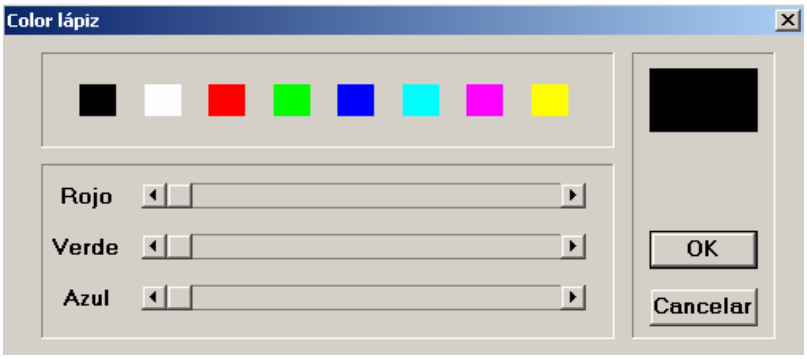
El choque de la tortuga con los bordes negros del circuito lo detectaremos preguntando, cada vez que la tortuga avanza, por el color del punto de la pantalla (píxel) en el que se sitúa. Veamos por tanto algunas instrucciones para trabajar con colores en Logo:

poncolorlapiz [ojo verde azul]

poncl [rojo verde azul] Selecciona el color del lápiz. Éste se obtiene mediante una mezcla de los colores rojo, verde y azul, indicando su proporción mediante un número comprendido entre 0 y 255. Algunos colores se pueden abreviar mediante un solo número:

negro:	`poncl [0 0 0]`	`poncl 0`
azul:	`poncl [0 0 255]`	`poncl 1`
verde:	`poncl [0 255 0]`	`poncl 2`
azul claro:	`poncl [0 255 255]`	`poncl 3`
rojo:	`poncl [255 0 0]`	`poncl 4`
violeta:	`poncl [255 0 255]`	`poncl 5`
amarillo:	`poncl [255 255 0]`	`poncl 6`
blanco:	`poncl [255 255 255]`	`poncl 7`
gris:	`poncl [128 128 128]`	

En el menú Configurar ……. Color del lápiz también se puede fijar el color combinando las proporciones de rojo verde y azul, al mover con el ratón el botón correspondiente

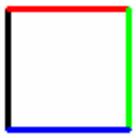


`pongrosor` [altura anchura]

`pong` [altura anchura] Define el grosor del lápiz

Ejemplo: Las siguientes instrucciones dibujan un cuadrado en el que cada lado es de un color, siendo el trazo de grosor 3:

```
pong [3 3] av 60 gd 90
poncl 4 av 60 gd 90
poncl 2 av 60 gd 90
poncl 1 av 50 gd 90
```



`píxel`

Devuelve (entre corchetes) los números que representan la intensidad de los colores rojo, verde y azul del píxel que se encuentra bajo la tortuga.

Ejemplo: `si píxel = [255 0 0] [re 10 gi 50]`

La línea de programa anterior pregunta si el color del punto en el que se encuentra la tortuga es rojo y en caso afirmativo ejecuta las instrucciones del último corchete: retrocede 10 pasos y gira a la izquierda 50 grados.

En nuestro juego tenemos que preguntar si el píxel en el que se encuentra la tortuga es negro y en caso afirmativo colocar la tortuga en el punto de inicio. El programa queda, definitivamente, de la forma siguiente:

```
para juego
bp sl                                          ;borramos pantalla y subimos el lápiz
ponfoco [Pantalla de MSWLogo]
ponteclado [control]                           ;al pulsar una tecla se ejecuta control
cargadib "circuito.bmp                         ;cargamos en la pantalla nuestro circuito
ponpos [-210 130] gd 90                        ;ponemos la tortuga en la salida
siempre [                                      ;estas instrucciones se repiten siempre
        av 1 repite 2000 []                    ;avanzamos la tortuga y esperamos un tiempo
        si píxel = [0 0 0] [ponpos [-210 130]];si borde negro volvemos a la salida
        ]
fin
```

```
para control
haz “tecla car lc                ;guardamos en la variable tecla el carácter pulsado
si :tecla = 4 [gi 20]            ;si es el número 4 giramos a izquierda
si :tecla = 6 [gd 20]            ;si es el número 6 giramos a derecha
fin
```

PROCEDIMIENTOS RECURSIVOS

Hasta ahora, para conseguir que un grupo de instrucciones se repitan de manera permanente hemos utilizado la instrucción siempre. Vamos a ver, a continuación, otra manera de hacerlo utilizando un procedimiento recursivo; es decir, un procedimiento que, antes de finalizar, se vuelve a llamar a si mismo.

En definitiva, las instrucciones que hay entre los corchetes de la instrucción siempre las incluimos en un procedimiento que vamos a llamar movimiento y cambiamos la instrucción siempre por una llamada a ese procedimiento:

```
para movimiento
av 1 repite 2000 []
si píxel = [0 0 0] [ponpos [-210 130]]
movimiento                ;el procedimiento se llama a si mismo y vuelve a empezar
fin

para juego
bp sl
ponfoco [Pantalla de MSWLogo]
ponteclado [control]
cargadib “circuito.bmp
ponpos [-210 130] gd 90
movimiento
fin
```

Para dar por finalizado un procedimiento recursivo o, en general, cualquier procedimiento (sin llegar a la instrucción fin) se utiliza la instrucción alto:

alto

Finaliza la ejecución del procedimiento en el que aparece

Ejemplo: Hemos modificado el procedimiento movimiento para que, si la tortuga encuentra una zona de color rojo, se de por finalizada su ejecución. El programa regresaría al procedimiento juego en la instrucción que sigue a movimiento, en este caso, fin.

```
para movimiento
av 1 repite 2000 []
si píxel = [0 0 0] [ponpos [-210 130]]
si píxel = [255 0 0] [alto]   ;si pasa sobre una zona roja finaliza el procedimiento
movimiento
fin
```

ACTIVIDADES:

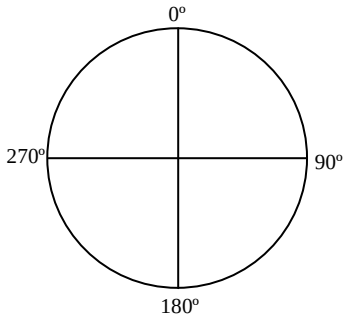
13. Realizar un programa que realice la siguiente secuencia:
    - Dibuja un cuadrado de color rojo, grosor 3 y 50 puntos de lado.
    - Transcurridos 4 segundos lo pone de color amarillo.
    - Pasados otros 2 segundos el color cambia a verde.
    - Después de 1 segundo vuelve a iniciarse el ciclo.

PONRUMBO

Al probar nuestro “juego de coches” habréis comprobado que, al chocar con un borde, la tortuga vuelve a la posición de salida manteniendo la orientación que tenía en el momento del choque. Es preferible que la tortuga se oriente en la dirección del circuito, en nuestro ejemplo, hacia la derecha. Como no sabemos cual es el ángulo que tendríamos que girar, es necesario usar la instrucción ponrumbo:

ponrumbo ángulo

ponr ángulo Cambia el rumbo de la tortuga por un nuevo rumbo absoluto. El argumento es un ángulo, el rumbo se mide en grados según el movimiento de las agujas del reloj desde el eje y.



Ejemplo:

```
ponrumbo 90
```

ACTIVIDADES:

14. Modificar el programa anterior para que, cuando la tortuga choque con un borde, regrese a la posición de salida y quede orientada correctamente.

CONTAR EL NÚMERO DE VUELTAS

Para poder contar el número de vueltas que da la tortuga tenemos que detectar su paso por la línea azul de salida y, cada vez que esto suceda, incrementar (sumar una unidad) una variable que hayamos creado al efecto; el contenido de esa variable será, en cada momento, el número completo de vueltas que ha dado la tortuga. También tendremos que mostrar el contenido de esa variable en la pantalla, eso lo podremos hacer con la instrucción escribe:

escribe "palabra
escribe "|frase|
escribe variable
(escribe "palabra "|frase| o varible … "palabra "|frase| o variable …)

es Escribe en la ventana de texto de la pantalla de trabajo la palabra indicada. Cuando queremos escribir una frase formada por varias palabras hay que colocarla entre barras. Si ponemos toda la instrucción entre paréntesis podemos mezclar palabras, frases y variables. Una vez finalizada la ejecución de una primitiva escribe se produce un salto de línea.

Ejemplo:

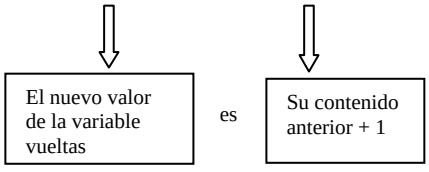
```
es "Antonio
es "|Hola Mundo|
es "Hola "Mundo     da un mensaje de error: no me has dicho que hace con Mundo
es :velocidad       escribe el contenido de la variable "velocidad.
(es "|la variable velocidad vale | :velocidad)
```

borratexto

bt Borra el contenido de la ventana Trabajo

En definitiva, podemos crear una variable denominada "vueltas y darle el valor inicial de cero. Posteriormente, cada vez que la tortuga pase por la línea azul borramos la pantalla de trabajo, incrementamos la variable "vueltas en una unidad y escribimos su contenido. Las líneas de código que tendríamos que añadir, en los sitios adecuados, a nuestro programa serían:

```
haz "vueltas 0
……
si píxel = [0 0 255] [bt haz "vueltas :vueltas +1 es :vueltas]
……
```



ACTIVIDADES:

15. Incluir las líneas de código anteriores en los lugares adecuados de nuestro juego de coches y comprobar su correcta ejecución.

16. Realizar un programa que al pulsar la tecla c escriba en la pantalla Trabajo el número de veces que hemos pulsado la tecla a.

EL CONTROL DEL TIEMPO

Por último, queremos que, al finalizar cada vuelta, aparezca en la pantalla Trabajo el tiempo (en segundos) transcurrido desde el inicio del juego. Para ello, vamos a utilizar la instrucción poncontador:

poncontador numero tiempo [instrucciones o procedimiento a ejecutar]

Ejecuta las instrucciones o procedimientos encerrados entre los corchetes cada vez que transcurre el tiempo indicado (en milésimas de segundo). MSWLogo puede activar a la vez hasta 31 contadores, por ello, hay que indicar el número del contador activado en la instrucción.

Ejemplo: `poncontador 1 1000 [haz “tiempo :tiempo+1]`

El ejemplo anterior activa el contador 1 para que, cada segundo, se incremente la variable tiempo, que se habrá definido con anterioridad.

El ejemplo anterior nos muestra el camino a seguir:

Crear un variable (denominada “tiempo) y darle un valor inicial de cero.
Utilizar la línea del ejemplo para que esa variable cuente los segundos transcurridos.
Escribir el contenido de la variable cada vez que finalice una vuelta.

ACTIVIDADES:

17. Modificar el juego para que, cada vez que finalice una vuelta, nos muestre el tiempo transcurrido.

18. Modificar nuestro juego para que finalice cuando se hayan dado 10 vueltas al circuito. Tendréis que preguntar si la variable vueltas es igual a 10 y, en caso afirmativo, ejecutar la instrucción alto.