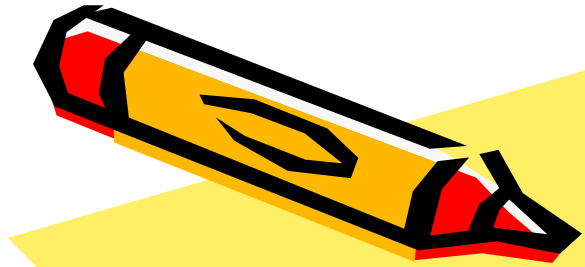
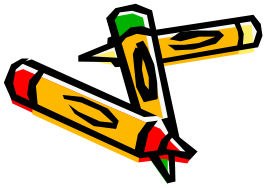


Iniciación a xLogo

Álvaro Valdés Menéndez

Índice

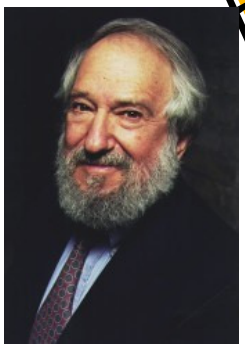


Álvaro Valdés Menéndez

Curso de *xLogo*

BLOQUE 0

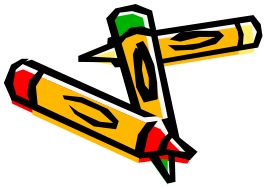
¿Qué es Logo? ...



- **Logo** fue creado por Seymour Papert en el MIT a finales de los años 60
- Papert trabajó con Piaget en la Universidad de Ginebra desde 1959 hasta 1963.
- Basándose en el Constructivismo de Piaget, desarrolló el aprendizaje Construccionista: "el aprendizaje mejora si se aplica activamente a la vida cotidiana" → "*learning-by-doing*"
- **Logo** es una potente herramienta para desarrollar los procesos de pensamiento lógico-matemáticos.
- El usuario mueve un objeto llamado "tortuga" dentro de la pantalla usando órdenes simples como "avanza", "retrocede", "giraderecha" y similares.
- Con cada movimiento la tortuga dibuja una línea tras de sí, y de esta manera se crean gráficos.

… y, ¿XLogo?

- XLOGO es una versión de LOGO escrita en JAVA y distribuida bajo licencia GPL (General Public License)
- GPL implica que XLOGO es libre, tanto en lo referente a disponibilidad del código fuente como a gratuidad.
- XLOGO está en continuo desarrollo, incorporando nuevas primitivas y capacidades constantemente.
- Actualmente admite 7 idiomas, y dispone de opciones para exportarlo a otros fácilmente.
- Poder usar órdenes en el idioma natural favorece su aprendizaje y asimilación
- JAVA es multiplataforma, lo que permite ejecutar XLOGO en todos aquellos sistemas operativos que la soporten (Linux, Windows, MacOS, Solaris, …)
- Recientemente JAVA ha sido liberada bajo licencia GPL, lo que también garantiza su disponibilidad y gratuidad.



Obtener XLogo

Para obtener XLogo, descargarlo libre y gratuitamente desde:

http://xlogo.tuxfamily.org

Curso de *xLogo*

BLOQUE 1

Requisitos previos

Antes de empezar el alumno debe:

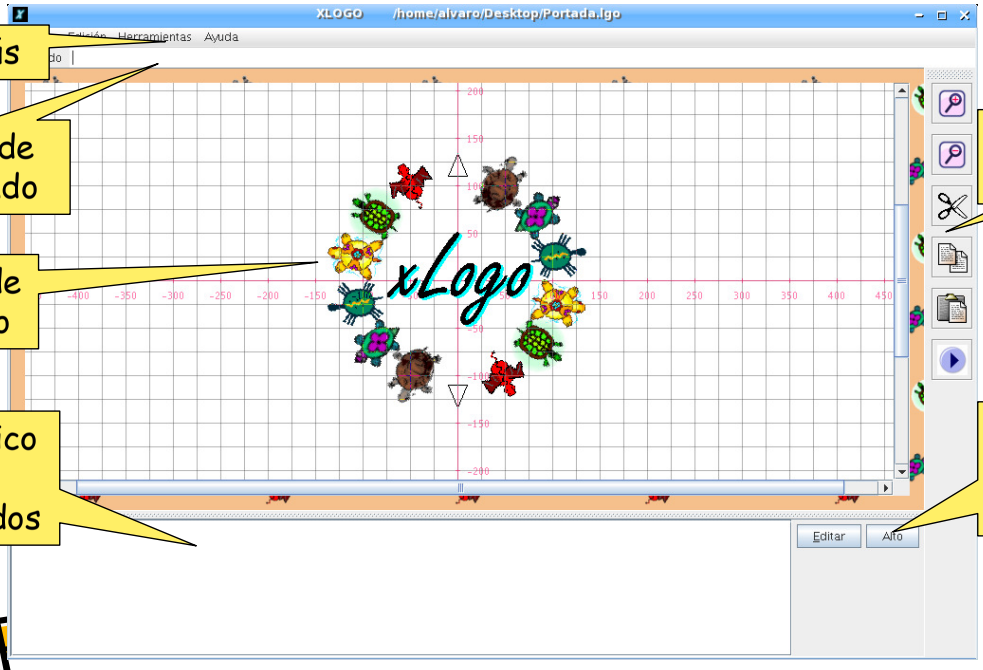
- Saber iniciar sesión con su nombre de usuario y contraseña
- Disponer de conexión a Internet o, en su defecto, que los archivos a trabajar se encuentren disponibles en la intranet local
- Comprobar que el entorno JAVA está instalado en su ordenador
- Tener unos mínimos conocimientos de geometría (polígonos, ángulos, …)

Objetivos

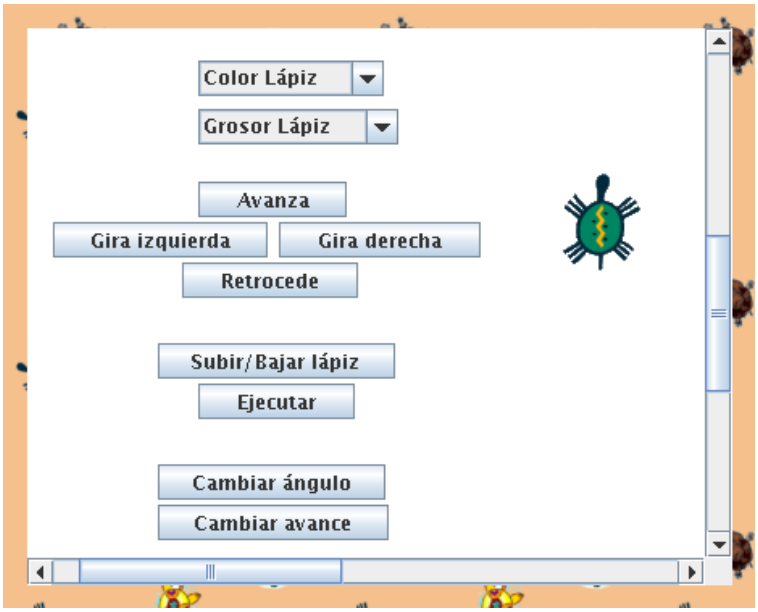
Al concluir este bloque el alumno sabrá:

- Arrancar el intérprete XLogo
- Explicar qué es una primitiva en XLogo y manejarlas correctamente
- Crear formas sencillas.
- Resolver problemas descomponiéndolos en partes más pequeñas

Presentando XLogo



¿Algo fácil para empezar?



¿Cómo consigo que XLOGO muestre esto?

El programa

Descargar
`http://xlogo.tuxfamily.org/sp/curso/presentacion.lgo`

↓

Abrir el fichero `presentacion.lgo`

↓

Escribir `presentacion` en la barra inferior

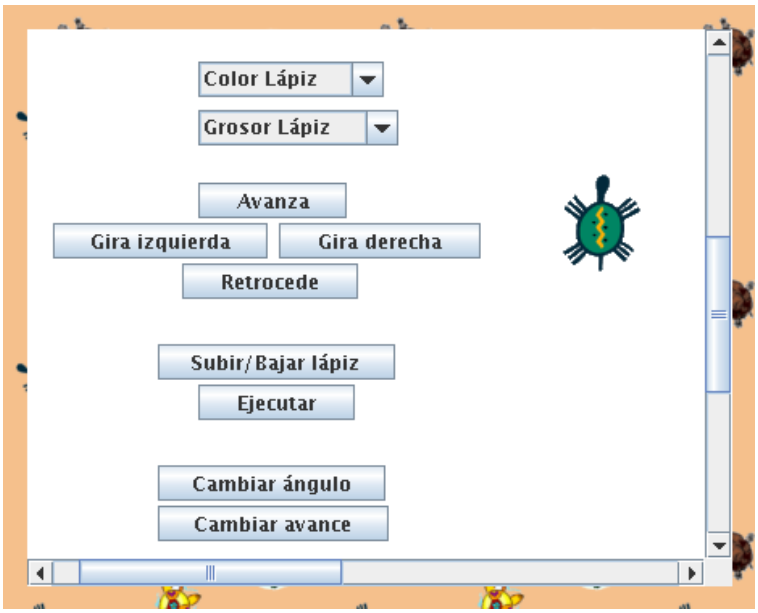
↓

"Click" en el botón del pingüino 

↓

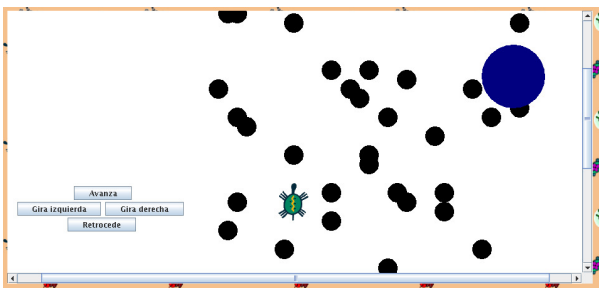
"Click" en el botón 

Debería mostrarse una pantalla como esta:

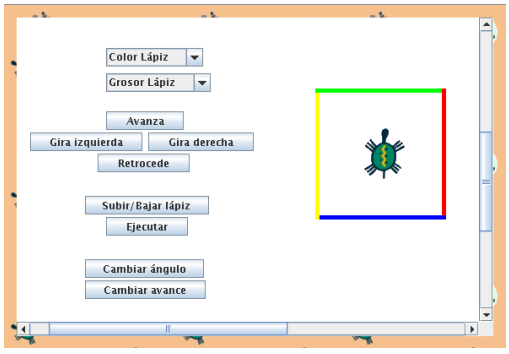


PRUEBA CÓMO SE MUEVE LA TORTUGA Y COMPRENDE "SU" PUNTO DE VISTA

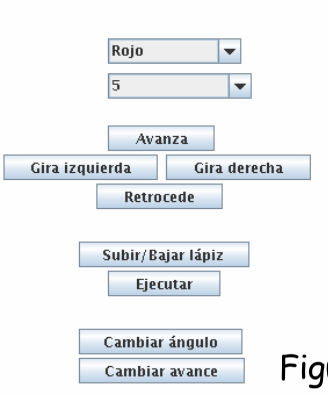
Y con esa pantalla, ¿qué?

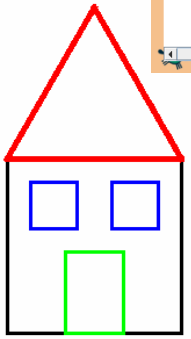


Plantear juegos (Infantil)



Proponer figuras simples





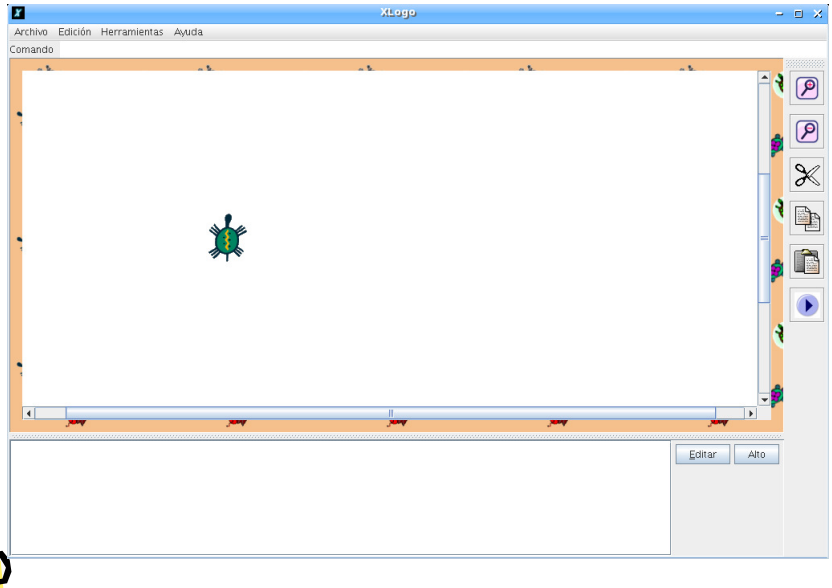
Figuras complejas

El alumno debe:

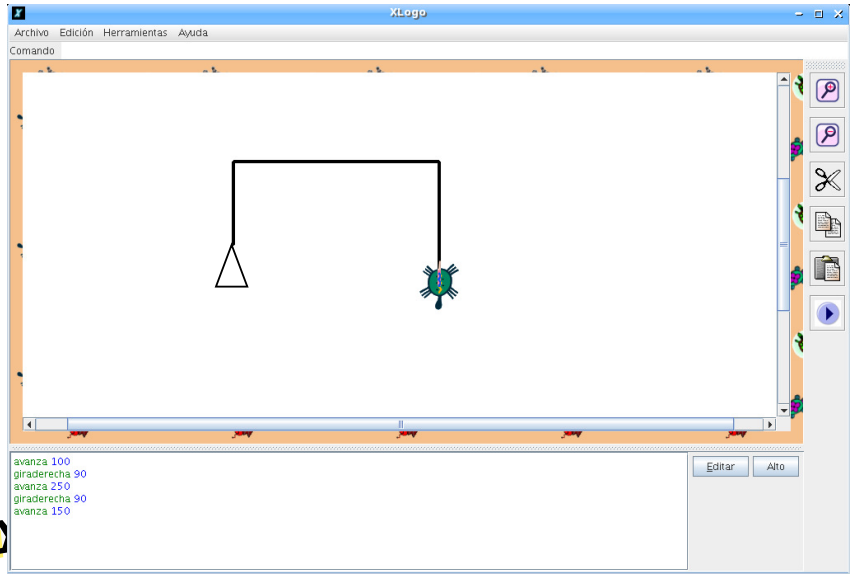
1. Entender él mismo aquello que va a enseñar.
2. Planear una forma de impartirlo.
3. "Trocear" el problema en miniproblemas más fácilmente abordables.
4. Saber cómo comunicarlo claramente.
5. Establecer este nuevo conocimiento como las bases de uno futuro.
6. Ser consciente del conocimiento que su "alumno" (la tortuga) ya tiene, y construir basándose en él.
7. Ser receptivo para explorar nuevas ideas según aparezcan.
8. Responder a los malentendidos de su alumno.

Lo más importante en el proceso de aprendizaje con Logo no es el resultado final, sino **cómo haces lo que haces**; es decir, **ver cómo se crea el diseño es más interesante y educativo que el diseño en sí.**

El Lenguaje de la Tortuga



El Lenguaje de la Tortuga

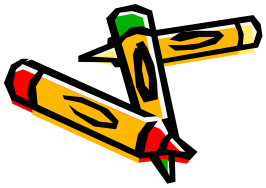


Las primitivas

- Las órdenes que recibe la tortuga se llaman "*primitivas*"
- Las primitivas asociadas a los movimientos requieren más información: los "*argumentos*"
  - **`avanza`** y **`retrocede`** esperan un número: cuántos *pasos* debe desplazarse
  - **`giraderecha`** y **`giraizquierda`** necesitan el número de grados que queremos que gire
- Para borrar lo dibujado: **`borrapantalla`**

Primera Actividad

- Imagínate caminando por el aula; ¿Qué "órdenes" tendrías que darte a ti mismo para salir de ella desde el sitio que ocupas?
- Imagina que tienes un lápiz pegado a la espalda con la punta hacia abajo y que va dibujando en el suelo. ¿Qué figura saldría?
- ¿Y si quisieras dar una vuelta alrededor de todo el aula? ¿Qué ordenes darías y qué dibujo saldría?



La primitiva `repite`

- Cuando rodeaste el aula, pudiste observar que algunas órdenes se repetían.
- La primitiva `repite` sirve para acortar las órdenes cuando una secuencia de ellas se repite
- Ejemplo:

```
repite 2 [avanza 30 giraderecha 90]
```

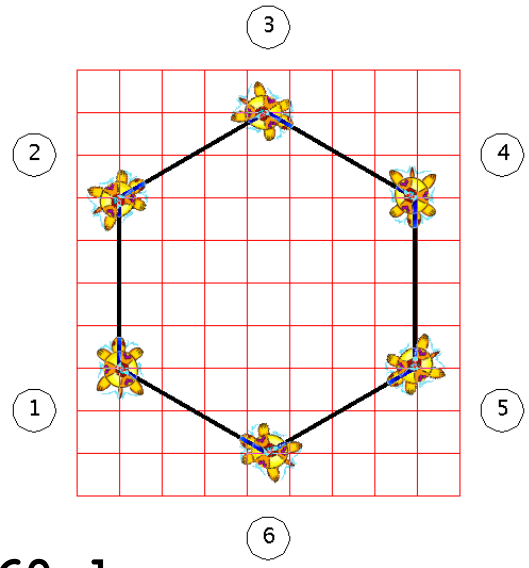
es equivalente a escribir

```
avanza 30 giraderecha 90
avanza 30 giraderecha 90
```

Polígonos

- Debemos pensar en:
  - Número de vértices
  - Ángulo de giro
  - Longitud del lado
- Programa:

```
repite 6
  [ avanza 100
    giraderecha 60 ]
```



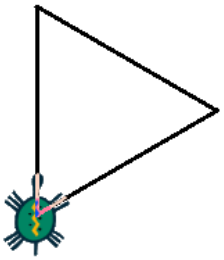
Segunda Actividad

- Acabas de ver la construcción de un hexágono usando **repite**. "Adivina" qué se obtiene con estas órdenes:

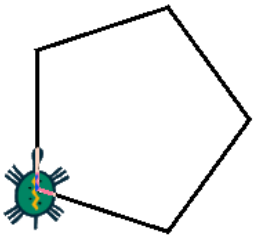
```
repite 3 [ avanza 150 giraderecha 120 ]

repite 5 [ avanza 100 giraderecha 72 ]

repite 8 [ avanza 75 giraderecha 45 ]

repite 9 [ avanza 150 giraderecha 80 ]
```

Haz *click* en siguiente para ver si acertaste

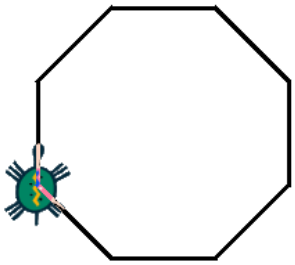
`repite 3 [...]`



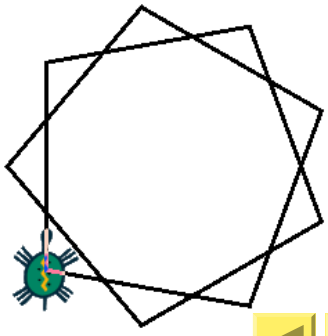
`repite 5 [...]`



`repite 8 [...]`



`repite 9 [...]`



Pensamiento deductivo (I)

- Observa los ejemplos anteriores:
  - El número que va tras **`repite`** indica el número de lados
  - Mira el ángulo de giro y el número de lados.
    - Prueba a multiplicarlos
    - El resultado es siempre 360
    - ¿Será siempre igual, o sólo en estos ejemplos?
  - Analiza tu conjetura intentando dibujar un decágono
- En la diapositiva siguiente puedes ver la respuesta

Pensamiento deductivo (II)

- Esto "debería" funcionar:

  `repite 10 [avanza 50 giraderecha 36]`

- ¿Por qué?
  - `repite 3 [av 100 gd 120]` → 3 x 120 = 360
  - `repite 5 [av 100 gd 72]` → 5 x 72 = 360
  - `repite 6 [av 100 gd 60]` → 6 x 60 = 360
  - `repite 8 [av 100 gd 45]` → 8 x 45 = 360
  - `repite 10 [av 100 gd 36]` → 10 x 36 = 360

  Al dibujar un polígono regular, la tortuga da una vuelta completa sobre sí misma

Pensamiento deductivo (III)

- En el ejemplo anterior hemos visto cómo podemos usar XLogo para fomentar el pensamiento deductivo
- El siguiente paso es (puede ser):
  - Proponer distintas figuras para dibujar
  - Jugar con características de las figuras
  - Buscar patrones que se repiten
  - Realizar predicciones

  . . .

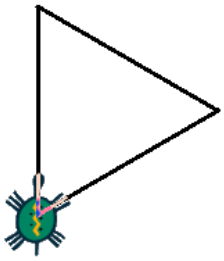
Tercera Actividad

- ¿Qué crees que pasará si usamos **giraizquierda** en vez de **giraderecha**?
- ¿Qué cambiarías para hacer los polígonos más grandes?
- ¿Qué pasaba en el ejemplo con

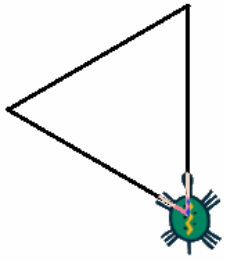
  repite 9 [. . .]?
- ¿Por qué crees que es?
- ¿En qué casos crees que puede hacerse?
- ...

Haz *click* en siguiente para verlo

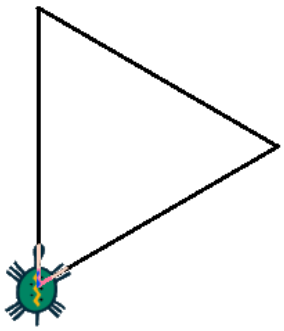
```
repite 3 [av 100
  giraderecha 120]
```



```
repite 3 [av 100
  giraizquierda 120]
```



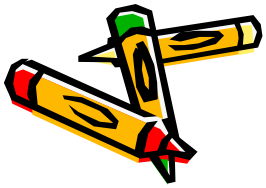
```
repite 3 [avanza 150
  giraderecha 120]
```



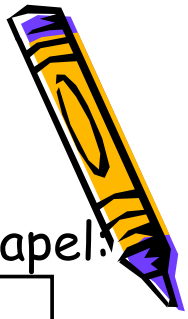
Para conseguir polígonos estrellados debemos tener un número de vértices impar y mayor que 3

Más primitivas

- La tortuga dibuja con un *lápiz* al moverse
- El lápiz puede estar *abajo*, *arriba*, *invertido* o ser una *goma*.
- Para hacer que dibuje o no, disponemos de:
  - **`bajalapiz`** → la tortuga dibuja al moverse
  - **`subelapiz`** → la tortuga no dibuja al moverse
  - **`inviertelapiz`** → si hay dibujo, invierte los colores
  - **`goma`** → la tortuga borra en vez de dibujar
- Para ver o no a la tortuga
  - **`muestratortuga`** → la tortuga es visible
  - **`ocultatortuga`** → no se ve a la tortuga



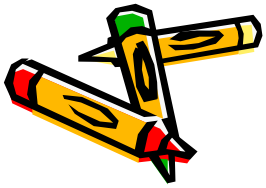
Más primitivas aún



- Podemos cambiar el color del lápiz y del papel:
  - **`poncolorlapiz`** →
  - **`poncolorpapel`** →

  esperan un argumento: número o nombre del color
- También rellenar regiones cerradas:
  - **`rellena`** → la zona cerrada en que se encuentra
  - **`rellenazona`** → la zona limitada por el color activo
- Y cambiar el grosor y la forma:
  - **`pongrosorlapiz`** →
  - **`ponformalapiz`** →

  esperan un número como argumento



Otro desafío

- ¿Cómo dibujarías un trozo de carretera?:

- Piensa un momento cómo lo harías
- Puedes descomponer el problema en varias partes (*divide y vencerás*)
- También hay más de una forma de conseguirlo

Abordando el desafío

1. Modelizar:
   - ¿Es una carretera?
   - ¿Es un rectángulo negro con líneas blancas?
2. Dividir en etapas:
   1. Dibujar el rectángulo
   2. Dibujar (o borrar) las líneas blancas
3. Conectar las etapas
   1. Situar la tortuga en el sitio correcto

Solucionando el desafío (I)

- Rectángulo:

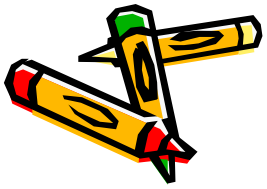
```
repite 2
  [ avanza 100 giraderecha 90
    avanza 800 giraderecha 90 ]
```

- Rellenar el rectángulo:

```
subelapiz
giraderecha 45 avanza 10
rellena
```

- Dibujar las líneas blancas

```
poncolorlapiz blanco
repite 8
  [ bajalapiz avanza 50
    subelapiz avanza 50]
```



Solucionando el desafío (II)

- Ubicando todo correctamente:
  1. El rectángulo

```
subelapiz giraizquierda 90
avanza 400 giraderecha 90
bajalapiz
```

  2. Tras rellenar el rectángulo:

```
retrocede 10 giraizquierda 45
bajalapiz
```

  3. Las líneas blancas

```
avanza 50 giraderecha 90
```

- Puliendo detalles
  1. Grosor de las líneas blancas:

```
pongrosor 4
```

Solucionando el desafío (III)

```
subelapiz giraizquierda 90
avanza 400 giraderecha 90
bajalapiz
repite 2
 [ avanza 100 giraderecha 90
   avanza 800 giraderecha 90 ]
subelapiz
giraderecha 45 avanza 10
rellena
retrocede 10 giraizquierda 45
bajalapiz
avanza 50 giraderecha 90
pongrosor 4
poncolorlapiz blanco
repite 8
  [ bajalapiz avanza 50
    subelapiz avanza 50]
```

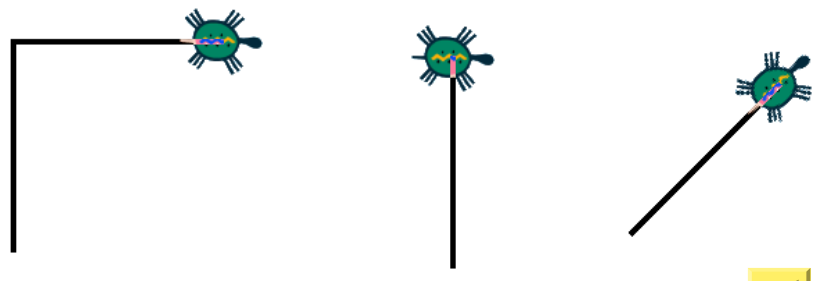
Autoevaluación 1

- Imagina que tecleas estas tres órdenes en la línea de comandos:

  borrapantalla

  avanza 150 giraderecha 90

- ¿Qué forma aparecerá?

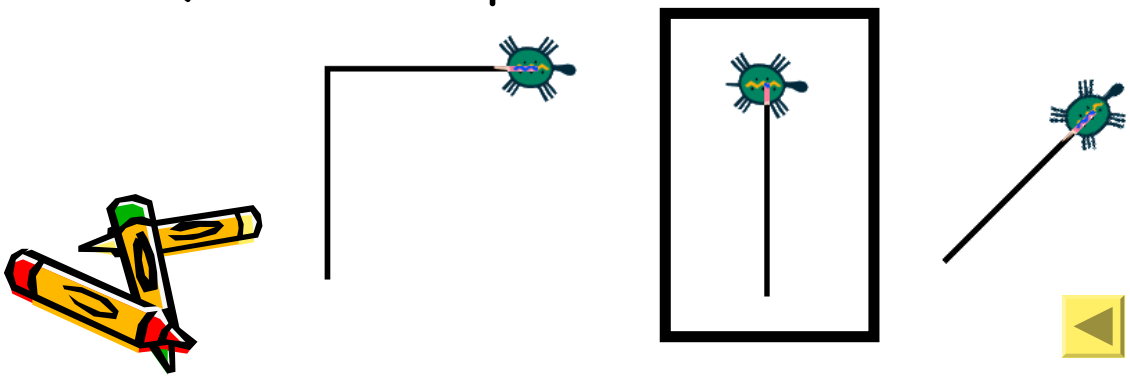


Autoevaluación 1

- Imagina que tecleas estas tres órdenes en la línea de comandos:

```
borrapantalla
avanza 150 giraderecha 90
```

- ¿Qué forma aparecerá?



Autoevaluación 2

- ¿Qué orden es una simplificación de la siguiente?

```
av 50 gd 60 av 50 gd 60 av 50 gd 60 ?
```

```
repite 3 av 50 gd 60
```

```
repite 3 [ av 50 gd 60 ]
```

```
repite [ av 50 gd 60 ] 3
```

Autoevaluación 2

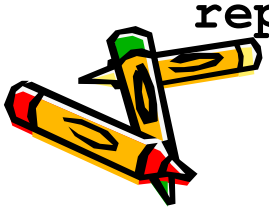
- ¿Qué orden es una simplificación de la siguiente?

```
av 50 gd 60 av 50 gd 60 av 50 gd 60 ?
```

```
repite 3 av 50 gd 60
```

```
repite 3 [ av 50 gd 60 ]
```

```
repite [ av 50 gd 60 ] 3
```



Autoevaluación 3

- ¿Qué orden de las siguientes avanzará 50 pasos sin dibujar?

```
avanza 50 subelapiz
```

```
goma avanza 50
```

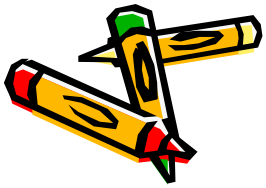
```
subelapiz avanza 50
```

Autoevaluación 3

- ¿Qué orden de las siguientes avanzará 50 pasos sin dibujar?

```
avanza 50 subelapiz
```

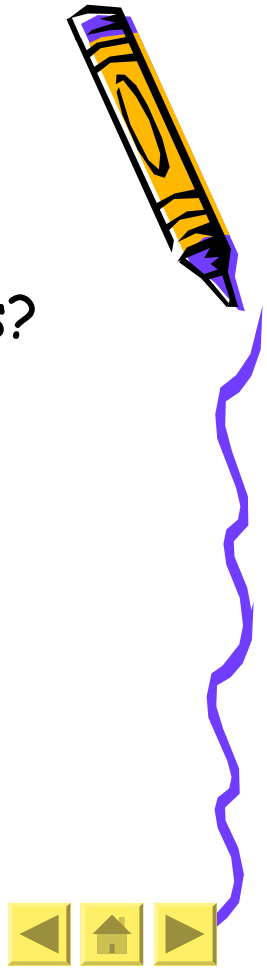
```
goma avanza 50

subelapiz avanza 50
```



Autoevaluación 4

- ¿Qué verías en pantalla tras introducir las órdenes siguientes?

```
poncolorpapel 0
poncolorlapiz 0
avanza 100
```



Autoevaluación 4

- ¿Qué verías en pantalla tras introducir las órdenes siguientes?

```
poncolorpapel 0
poncolorlapiz 0
avanza 100
```

Sólo verías a la tortuga en otro sitio, porque estás dibujando con color negro en una pantalla también negra

Autoevaluación 4

- ¿Qué verías en pantalla tras introducir las órdenes siguientes?

```
poncolorpapel 0
poncolorlapiz 0
avanza 100
```

Curso de *xLogo*

BLOQUE 2

Requisitos previos

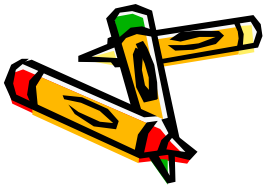
Antes de empezar esta parte el alumno debe:

- Poder iniciar XLogo correctamente
- Saber crear formas sencillas con **`repite`**, **`avanza`**, **`retrocede`**, **`giraderecha`** y **`giraizquierda`**.
- Dominar las primitivas **`subelapiz`**, **`bajalapiz`**, **`goma`**, **`poncolorlapiz`** y **`poncolorpapel`**.
- Haber comprendido que los problemas pueden resolverse más fácilmente si se descomponen en partes más pequeñas

Objetivos

Al concluir este bloque el alumno sabrá:

- Explicar qué es un procedimiento en Logo
- Crear procedimientos nuevos y editar los existentes
- Ejecutar procedimientos
- Guardar y cargar procedimientos en/de el disco duro
- Explicar, entender y utilizar variables



Procedimientos

- En el bloque anterior, cuando escribíamos una orden (o varias), la tortuga las ejecutaba de inmediato
- Pero igualmente, LAS OLVIDABA al terminar de dibujar
- Si queríamos dibujar algo de nuevo, debíamos re-escribir las órdenes o seleccionarlas en el **Histórico de Comandos** con el ratón o las flechas del teclado
- Los **PROCEDIMIENTOS** son la forma de **ENSEÑAR** nuevas primitivas a la tortuga

¿Qué son los procedimientos?

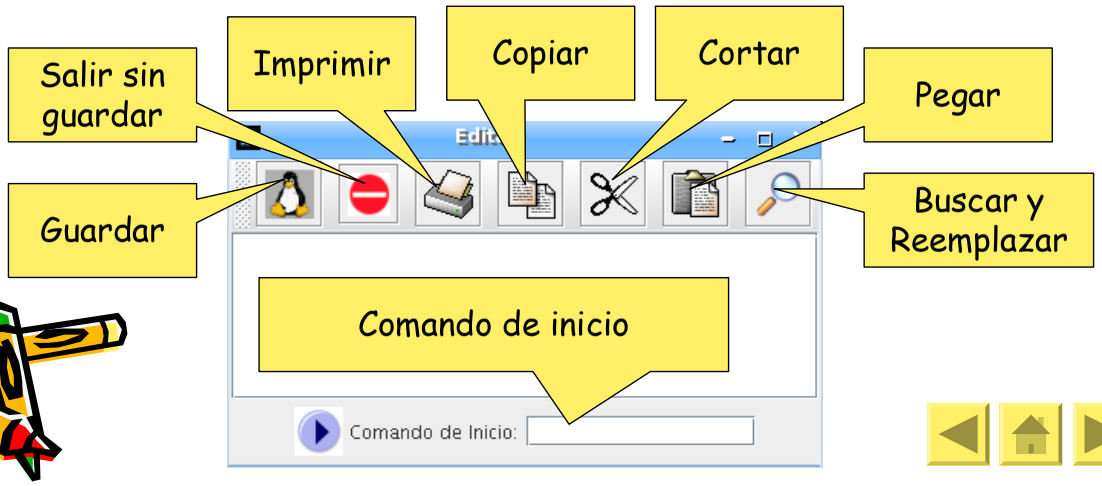
- Un procedimiento es un conjunto de órdenes caracterizadas por un **nombre**
- Al crear un procedimiento, enseñamos a la tortuga cómo hacer cosas nuevas
- Por ejemplo, podemos enseñarle que un `cuadrado` se dibuja con las órdenes:

```
repite 4
    [avanza 100 giraderecha 90]
```

- Además, será más fácil cambiar el tamaño si deseamos cuadrados más grandes o más pequeños que re-escribiendo la línea completa

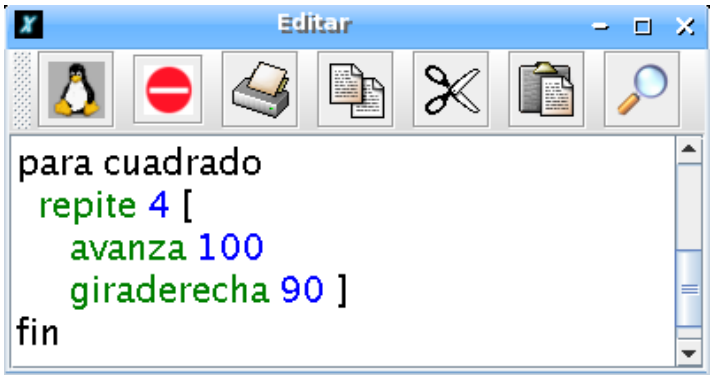
¿Cómo crear procedimientos?

- Tres formas:
  1. Escribiendo `ed` en la Línea de Comandos
  2. Haciendo *click* en el botón `Editar`
  3. Escribiendo `para nombre` en la Línea de Comandos
- En todos los casos, se iniciará el **Editor de Procedimientos**:



Sintaxis del procedimiento

- En el Editor debes empezar los procedimientos con la primitiva **`para`** y terminarlos con la primitiva **`fin`**
- En el ejemplo anterior:



Para guardar, hacemos *click* en el botón del pingüino

Procedimiento creado, ¿ahora qué?

- En el **Histórico de Comandos** debe aparecer la frase:

`Acaba de definir cuadrado`

- Desde este momento, cada vez que escribamos **`cuadrado`** la tortuga dibujará un cuadrado de lado 100
- Los procedimientos pueden usarse como cualquier otra primitiva
- Pueden formar parte de otros procedimientos

SUBPROCEDIMIENTOS

Subprocedimientos

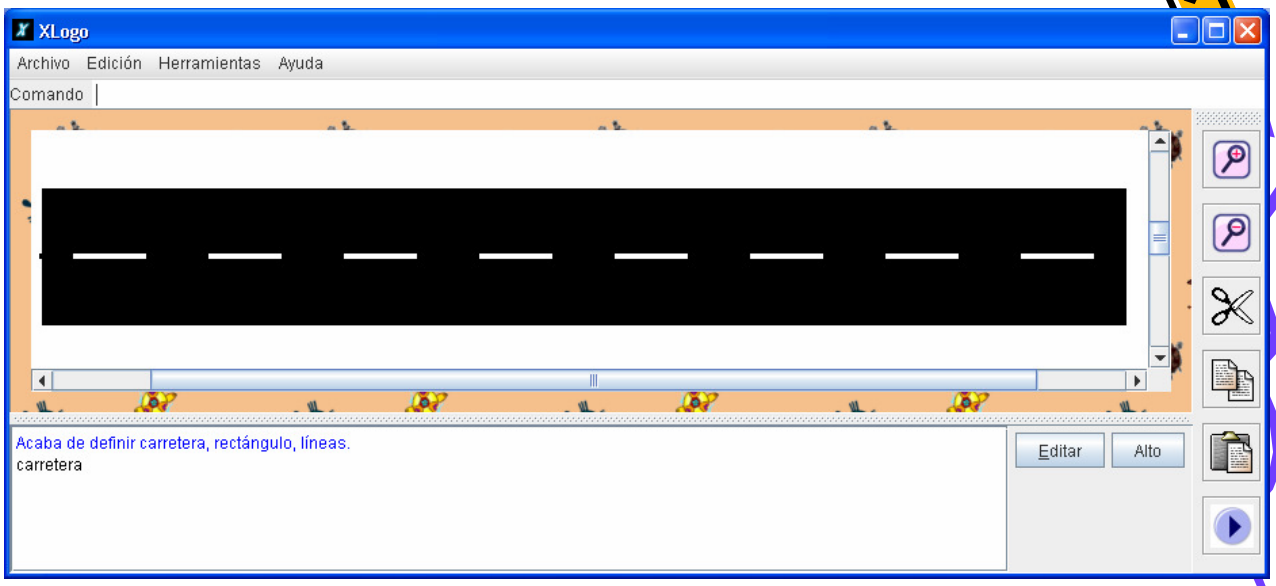
- Cuando un procedimiento es llamado por otro, se le denomina *subprocedimiento*
- Se usan para que la división hecha del problema:
  - se plasme en el código
  - éste sea más fácilmente entendible
- En el ejemplo del bloque anterior que dibujaba una carretera podemos definir dos procedimientos:
  - **rectángulo**
  - **líneas**
- Separamos los movimientos de las "acciones importantes"

Solucionando el desafío (IV)

```
para carretera
 ocultatortuga
 subelapiz giraizquierda 90
 avanza 400 giraderecha 90
 bajalapiz
 rectángulo
 subelapiz
 giraderecha 45 avanza 10
 rellena
 retrocede 10 giraizquierda 45
 bajalapiz
 avanza 50 giraderecha 90
 líneas
fin
```

```
para rectángulo
 repite 2
  [ avanza 100 giraderecha 90
   avanza 800 giraderecha 90 ]
fin

para líneas
 avanza 25
 pongrosor 4
 poncolorlapiz blanco
 repite 8
   [ bajalapiz avanza 50
     subelapiz avanza 50]
fin
```

Solucionando el desafío (IV)



Autoevaluación 5

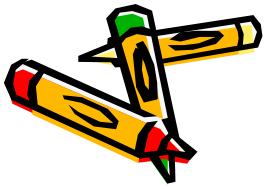
- ¿Qué es un procedimiento?
  - Una figura que dibuja la tortuga
  - Un conjunto de órdenes guardadas bajo un nombre
  - Una primitiva de XLOGO. Por ejemplo:

```
procedimiento 45
```

Autoevaluación 5

- ¿Qué es un procedimiento?
  - Una figura que dibuja la tortuga
  - Un conjunto de órdenes guardadas bajo un nombre
  - Una primitiva de XLOGO. Por ejemplo:

```
procedimiento 45
```



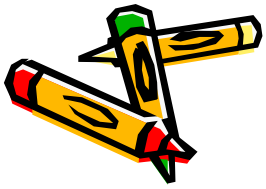
Autoevaluación 6

- Para definir un procedimiento escribimos:
  - `empieza nombre.proc`
  - `para nombre.proc`
  - `ed nombre.proc`

Autoevaluación 6

- Para definir un procedimiento escribimos:
  - **`empieza nombre.proc`**

  - **`para nombre.proc`**
  - **`ed nombre.proc`**

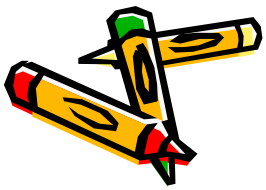


Autoevaluación 7

- Una vez definido un procedimiento llamado **`estrella`**, para ejecutarlo escribimos:
  - **`ejecuta estrella`**
  - **`empieza estrella`**
  - **`estrella`**

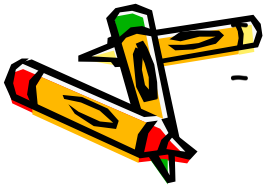
Autoevaluación 7

- Una vez definido un procedimiento llamado **`estrella`**, para ejecutarlo escribimos:
  - **`ejecuta estrella`**
  - **`empieza estrella`**
  - **`estrella`**



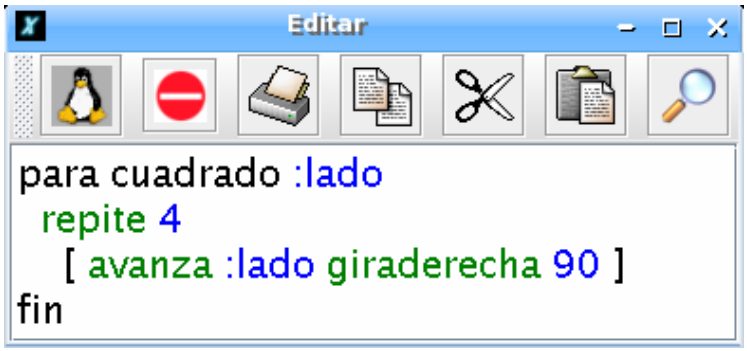
Variables

- ¿Qué hacemos si queremos cambiar el lado del cuadrado?
  - Editar cada vez el procedimiento **`cuadrado`**
  - Definir procedimientos sin saber cuánto van a valer determinados números, pero dejando "huecos" preparados para asignarles los valores más tarde
- Se llama *variable* a una letra o palabra que representa a un valor que puede cambiar (variar)
- En XLOGO se usan dos convenios distintos para definir y para llamar a las variables:
  - Para definir variables: **`haz "variable`**
  - Para llamarla (usarla): **`escribe :variable`**



Procedimientos con variables

- Para usar variables en un procedimiento, sólo hay que indicarle cuáles va a usar.
- En el ejemplo del cuadrado:



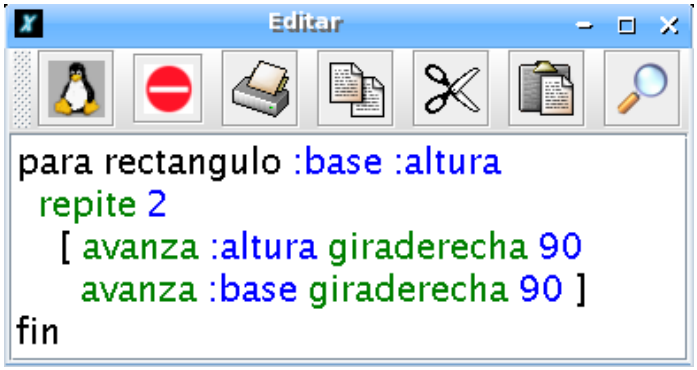
- "`lado` aún no tiene valor, debemos dárselo al usar el procedimiento:

```
cuadrado 50
```

dibujará un cuadrado cuyo lado mide 50 pasos de tortuga

Más procedimientos con variables

- En el ejemplo del rectángulo:



- Ahora definimos dos variables "`base` y "`altura`, cuyo valor es asignado al llamar al procedimiento:

```
rectangulo 50 100
```

Suponemos que la tortuga está mirando hacia arriba para que base y altura tengan el significado conocido

Operaciones

- La primera operación con variables es cómo asignarle un valor:

```
haz "variable 450
```

- Las operaciones que pueden efectuarse son las habituales:
  - Sumas: `haz "x :x+:y+:z+50`
  - Restas: `haz "x 50-:x`
  - Productos: `haz "área :base*:altura`
  - Divisiones:
    - Real: `haz "reparto :total/:personas`
    - Entera : `haz "base cociente :área :altura`
  - Raíz cuadrada:

```
haz "ladocuad raizcuadrada :área
```

Más operaciones

- Otras operaciones son:
  - Potencias: `haz "x potencia :x 2`
  - Módulo: `haz "x resto 13 3`
  - Truncamiento y redondeo:

```
haz "euros truncar :pesetas/166.386
haz "euros redondea :pesetas/166.386
```

  - Cambiar signo:

```
haz "opuesto cambiasigno :valor
haz "opuesto (-:valor)
```

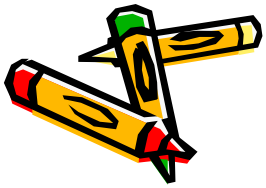
  - Trigonométricas y logarítmicas:

```
seno        coseno        tangente
arcoseno    arcocoseno    arcotangente
log         log10
```

Autoevaluación 8

- Tras las siguiente operaciones, ¿cuál es el valor de "**variable**?:

```
haz "variable 25
haz "variable :variable * 2
haz "variable :variable + 10
haz "variable :variable / 2
```



Autoevaluación 8

- Tras las siguiente operaciones, ¿cuál es el valor de "**variable**?:

```
haz "variable 25
haz "variable :variable * 2
haz "variable :variable + 10
haz "variable :variable / 2
```

"**variable** guarda el valor 30

Autoevaluación 9

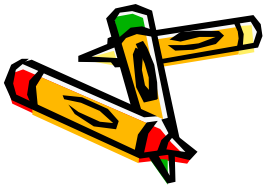
- ¿Cuál no es un nombre correcto para una **variable**?:

  "ladocuadrado

  "lado_cuadrado

  "lado cuadrado

  "lado.cuadrado



Autoevaluación 9

- ¿Cuál no es un nombre correcto para una **variable**?:

  "ladocuadrado

  "lado_cuadrado

  "lado cuadrado

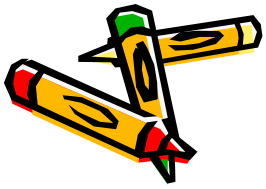
  "lado.cuadrado

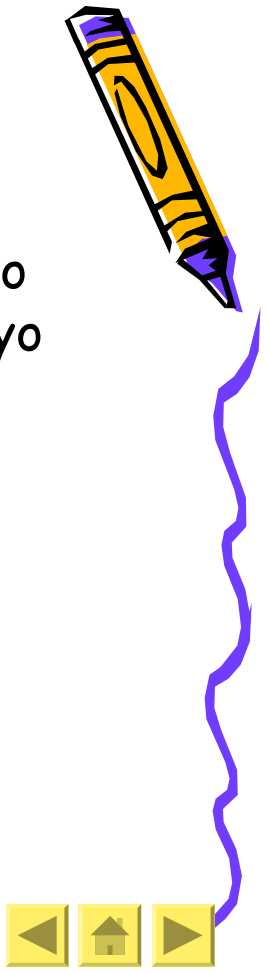
Los nombres de variables no pueden incluir espacios

Autoevaluación 10

- Con el procedimiento **cuadrado** definido antes, ¿cómo dibujarías un cuadrado cuyo lado mide 80 pasos de tortuga?:

```
para cuadrado 80

haz cuadrado 80

cuadrado 80

cuadrado 80 pasos
```

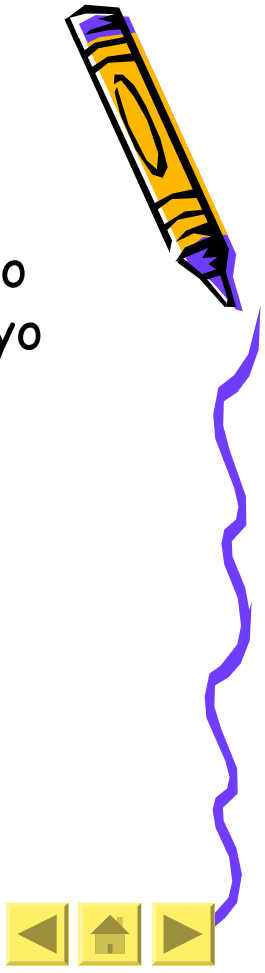




Autoevaluación 10

- Con el procedimiento **cuadrado** definido antes, ¿cómo dibujarías un cuadrado cuyo lado mide 80 pasos de tortuga?:

```
para cuadrado 80

haz cuadrado 80

cuadrado 80

cuadrado 80 pasos
```

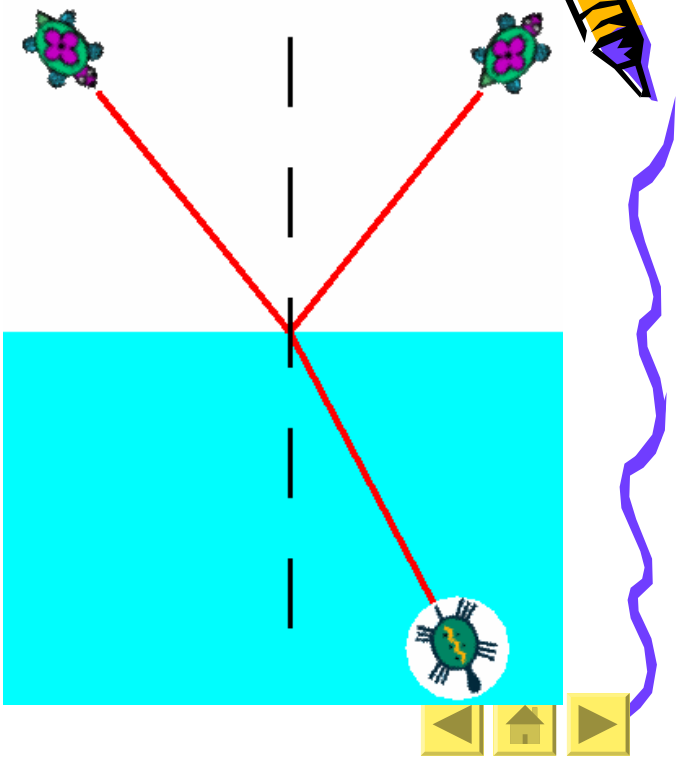


Ejemplo avanzado: Refracción de la luz

- Refracción de la luz:
- Se pide un punto
- Se desplaza hacia el centro de la pantalla
- "Choca" con una "superficie traslúcida"
- Se refleja y refracta de acuerdo a las leyes de Snell

$$\alpha_{inc} = \alpha_{refl}$$

$$n_i \cdot \text{sen}\, \alpha_{inc} = n_r \cdot \text{sen}\, \alpha_{refr}$$



Más sobre procedimientos

- Comentarios:
  - Los comentarios ayudan a entender un programa
  - Son líneas que la tortuga ignora
  - Se crean anteponiendo la "almohadilla": **#**
- Para conocer los procedimientos definidos se usa la primitiva **listaprocs**.
- Existe otra forma de definirlos: la primitiva **define**:

```
define "cuadrado [lado]
  [repite 4 [avanza :lado giraderecha 90]]
```

- Podemos guardar los procedimientos definidos en el disco duro:
  - Menú Archivo → Guardar como …
  - Primitiva **guarda**
- Igualmente podemos cargar desde el disco duro:
  - Menú Archivo → Abrir
  - Primitiva **carga**

Más sobre variables

- El nombre de las variables debe estar relacionado con el "cometido" que van a desempeñar. Eso facilita la posterior lectura del programa
- Para conocer las variables definidas se usa la primitiva **`listavars`**.
- Para borrar alguna variable se usa la primitiva **`borravariable`**.
- Hay *variables locales* y *variables globales*.
  - Globales: Pueden usarse en todos los subprocedimientos de un programa
  - Locales: sólo "existen" durante la ejecución del procedimiento

    Son útiles en programas largos, cuando varias variables comparten nombre
- Para borrar todas las variables y procedimientos se usa la primitiva **`borratodo`**.

  PELIGRO – NO PIDE CONFIRMACIÓN

Curso de *xLogo*

BLOQUE 3

Requisitos previos

Antes de empezar, el alumno debe:

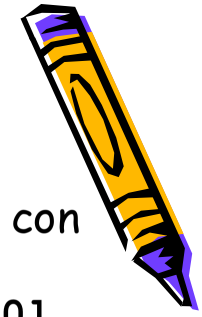
- Crear formas variadas
- Usar correctamente variables y procedimientos
- Manejar sin problemas la carga y guardado en disco
- Haber comprendido los conceptos de variable global y local

Objetivos

Al concluir este bloque el alumno sabrá:

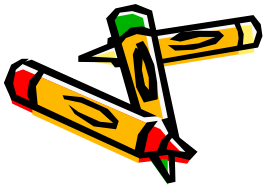
- Mover a la tortuga a cualquier punto de la pantalla
- Orientar a la tortuga en cualquier dirección
- Mostrar y ocultar en pantalla una cuadrícula y los ejes cartesianos
- Comprender qué es una estructura condicional y sus operaciones lógicas
- Entender qué es una lista
- Manejar los elementos de una lista: seleccionar, borrar, añadir, intercalar, …

Coordenadas y Rumbo



- La posición de la tortuga en pantalla se describe con una lista que consta de dos números: `[X Y]`
- Al centro de la pantalla corresponde la lista `[0 0]`
- El resto de coordenadas se determinan a partir del centro
- El primer valor de la lista es la separación horizontal respecto al centro
- El segundo valor de la lista es la separación vertical respecto al centro
  - Por ejemplo: el punto `[50 100]` corresponde al lugar que alcanzaríamos haciendo:

```
giraderecha 90 avanza 50
giraizquierda 90 avanza 100
```



Nos desplazamos 50 pasos hacia la derecha y 100 pasos hacia arriba

Coordenadas

- Podemos ayudarnos de una cuadrícula y unos ejes para situarnos mejor
  - `cuadricula a b` → muestra una rejilla de anchura a y altura b
  - `ejes a` → muestra los ejes cartesianos con paso a
  - `ejex a` → muestra el eje de abscisas con paso a
  - `ejey a` → muestra el eje de ordenadas con paso a
- Para desplazarnos a un punto dadas sus coordenadas disponemos de las primitivas:
  - `centro` → tortuga al origen, punto `[0 0]`
  - `ponposicion [a b]` → tortuga al punto `[a b]`
  - `ponxy a b` → tortuga al punto `[a b]`
  - `ponx a` → desplaza a al punto de abscisa a sin variar la ordenada
  - `pony b` → desplaza a al punto de ordenada b sin variar la abscisa

Para conocer nuestra posición, teclearemos:

```
posicion
```

Rumbo

- XLOGO define el rumbo como el ángulo que forma la tortuga con la vertical, en sentido horario
- Las primitivas son:
  - **`rumbo`** → devuelve el rumbo
  - **`ponrumbo n`** → orienta a la tortuga en la dirección indicada
  - **`hacia [x y]`** → devuelve el rumbo para que la tortuga "apunte" al punto **`[x y]`**
  - **`distancia [x y]`** → devuelve la distancia hasta el punto **`[x y]`**
- Ejemplo de uso:

  Dibujar los haces en el ejemplo anterior sobre reflexión y refracción, haciendo que la tortuga apunte en las direcciones de dichos haces

Coordenadas y rumbo Ejemplo

- Retomemos el ejemplo de la carretera una vez más
- Además del rectángulo y la línea discontinua, el problema obligaba a desplazar y orientar a la tortuga a puntos distintos:
  - El vértice del rectángulo mirando hacia arriba
  - El punto medio de un lado vertical mirando hacia la derecha
- Podemos definir un procedimiento que lo haga por nosotros: **`veteyapunta`**
  - Admitirá TRES argumentos: **`X`**, **`Y`** y **`n`**
  - Situará a la tortuga en el punto **`[X,Y]`** con rumbo **`n`**
- Por ejemplo, puede ser:

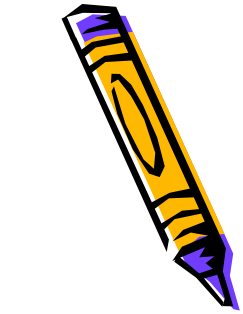
```
para veteyapunta :x :y :n
  subelapiz ponxy :x :y
  ponrumbo :n bajalapiz
fin
```

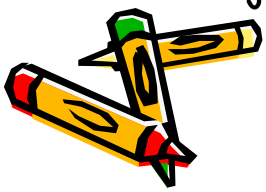
Puliendo el ejemplo

- De ese modo, el programa quedaría:

```
para carretera
  ocultatortuga
  veteyapunta -400 -50 0
  rectángulo
  veteyapunta 0 0 0
  rellena
  veteyapunta -400 0 90
  líneas
fin
```

- Observa que hemos hecho los cálculos para que el rectángulo quedara centrado en pantalla
- Podemos hacer que sea la tortuga la que haga el trabajo, y:
  - Nosotros sólo damos el tamaño del rectángulo
  - Es ella la que centra el dibujo
  - También ajusta el tamaño de las líneas
  - ...





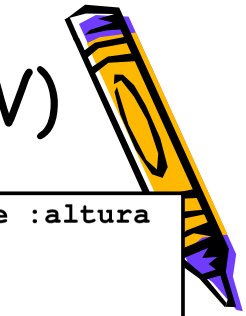
Solucionando el desafío (y V)

```
para carretera :largo :alto
  borrapantalla
  ocultatortuga
  haz "x  cambiasigno :largo/2
  haz "y  cambiasigno :alto/2
  veteyapunta :x :y 0
  rectángulo :largo :alto
  subelapiz centro bajalapiz
  rellenazona
  veteyapunta :x 0 90
  líneas :largo 8
fin
```

```
para veteyapunta :x :y :n
  subelapiz ponxy :x :y
  ponrumbo :n bajalapiz
fin
```

```
para rectángulo :base :altura
 repite 2
  [ avanza :altura
    giraderecha 90
    avanza :base
    giraderecha 90 ]
fin
```

```
para líneas :largo :n
  haz "long :largo/(2 * :n)
  avanza :long/2
  pongrosor 4
  poncolorlapiz blanco
  repite :n
    [ bajalapiz avanza :long
      subelapiz avanza :long]
fin
```



Solucionando el desafío (y V)

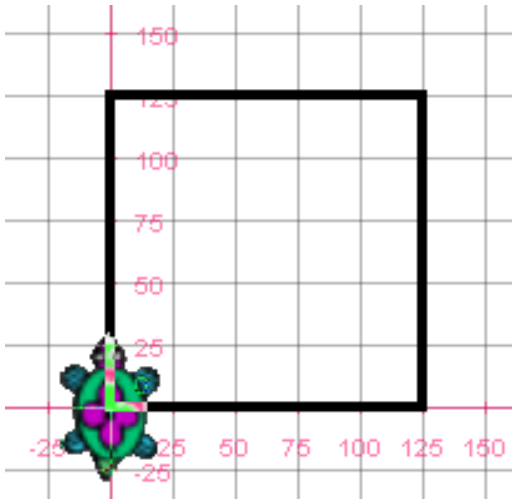
Autoevaluación 11

- ¿Qué dibuja esta secuencia de órdenes?

```
ponxy 0 125
ponxy 125 125
ponxy 125 0
ponxy 0 0
```

Autoevaluación 11

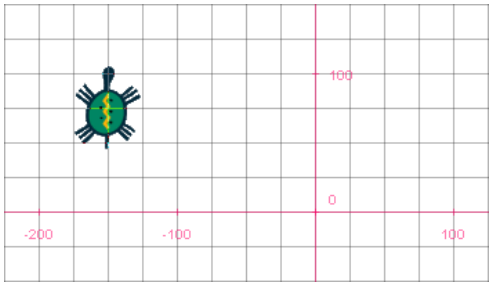
- ¿Qué dibuja esta secuencia de órdenes?



¡Un cuadrado!

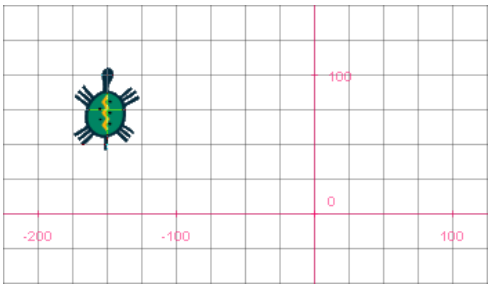
Autoevaluación 12

- ¿Dónde está la tortuga?



Autoevaluación 12

- ¿Dónde está la tortuga?



En el punto

[-150 75]

Condicionales

- Un condicional decide qué acción realizar en función de que se cumpla o no una determinada condición.
- Utilizamos la primitiva **si**:

```
si condición
   [acciones a realizar si es cierta]
```

- O bien:

```
si condición
   [acciones a realizar si condición es cierta]
   [acciones a realizar si condición es falsa]
```

- Ejemplo:

```
si :número > 0
  [escribe [El número es negativo] ]
[escribe raizcuadrada :número]
```

Condicionales (II)

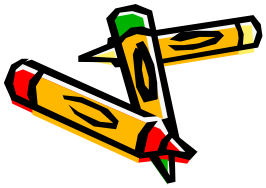
- La estructura condicional permite crear programas complejos.
- Los condicionales pueden encadenarse:

```
si condición1
  [ si condición2
     [ acciones a realizar si ambas son ciertas]
     [ acciones si es cierta la 1 pero no la 2] ]
  [ acciones a realizar si condición1 es falsa]
```

- O bien usar las operaciones lógicas:
  - `y cond1 cond2` → deben cumplirse ambas para obtener `cierto`
  - `o cond1 cond2` → debe cumplirse al menos una para ser `cierto`
  - `no condición` → se obtiene `cierto` cuando `condición` es falsa
- Existen primitivas que devuelven `cierto` o `falso` en función del estado de la tortuga, de las características de la pantalla, ...:

```
bajalapiz?   visible?
cuadricula?  ejes?      ejex?     ejey?
variable?    procedimiento?       primitiva? …
```

Reciben el nombre de ***BOOLEANOS***



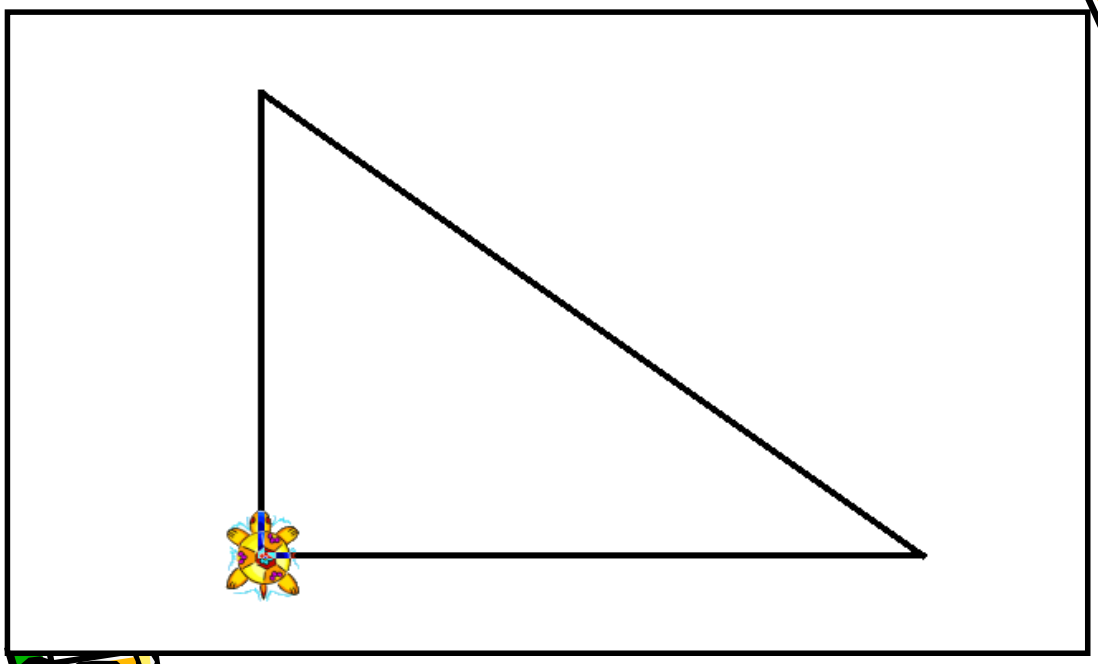
Autoevaluación 13

- ¿Qué mostrará la ejecución del siguiente procedimiento?

```
para test :a :b :c
 si iguales? :a rc (:b*:b + :c*:c)
  [ subelapiz centro bajalapiz
    pony :b ponxy :c 0
    centro ]
  [ escribe [No es un triángulo rectángulo]  ]
fin

test 500 300 400
```

Usa un papel cuadriculado para hacer los dibujos

Autoevaluación 13



Autoevaluación 14

- ¿Qué mostrará la siguiente secuencia de órdenes?

```
haz "valor 27
si :valor < 15
  [ escribe [Demasiado pequeño] ]
  [ escribe [Demasiado grande]  ]
```

Nada

Demasiado pequeño

Demasiado grande

Autoevaluación 14

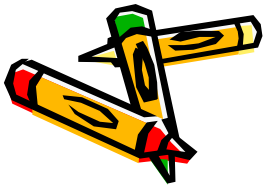
- ¿Qué mostrará la siguiente secuencia de órdenes?

```
haz "valor 27
si :valor < 15
  [ escribe [Demasiado pequeño] ]
  [ escribe [Demasiado grande]  ]
```

Nada

Demasiado pequeño

Demasiado grande



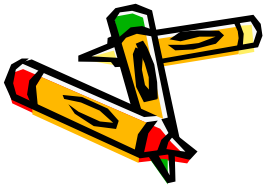
Listas

- Se define una lista como un conjunto de elementos guardados en una variable, ordenados y "etiquetados"
- En otros lenguajes de programación se les llama "matrices" o "vectores"
- Cada valor de la lista se denomina *elemento*
- Cada elemento posee un número que lo caracteriza:

  Ejemplo: En

  `[Hola esto es una lista de ejemplo]`

  a `Hola` le corresponde el número 1 y a `ejemplo` el número 7



Manejo de listas

- Para crear una lista:
  - Desde cero:
    `haz "ejemplo [Tengo 35 años]`
  - Combinando variables, listas, …:
    `haz "ejemplo frase [Tengo] :edad`
- Manejar la lista:
  - Contar los elementos: `cuenta :nombre.lista`
  - Leer el primer elemento: `primero :nombre.lista`
  - Leer el último elemento: `ultimo :nombre.lista`
  - Leer el elemento n-simo: `elemento n :nombre.lista`
- Modificar la lista:
  - Quitar el primero: `menosprimero :nombre.lista`
  - Quitar el último: `menosultimo :nombre.lista`
  - …

Ejemplo con listas

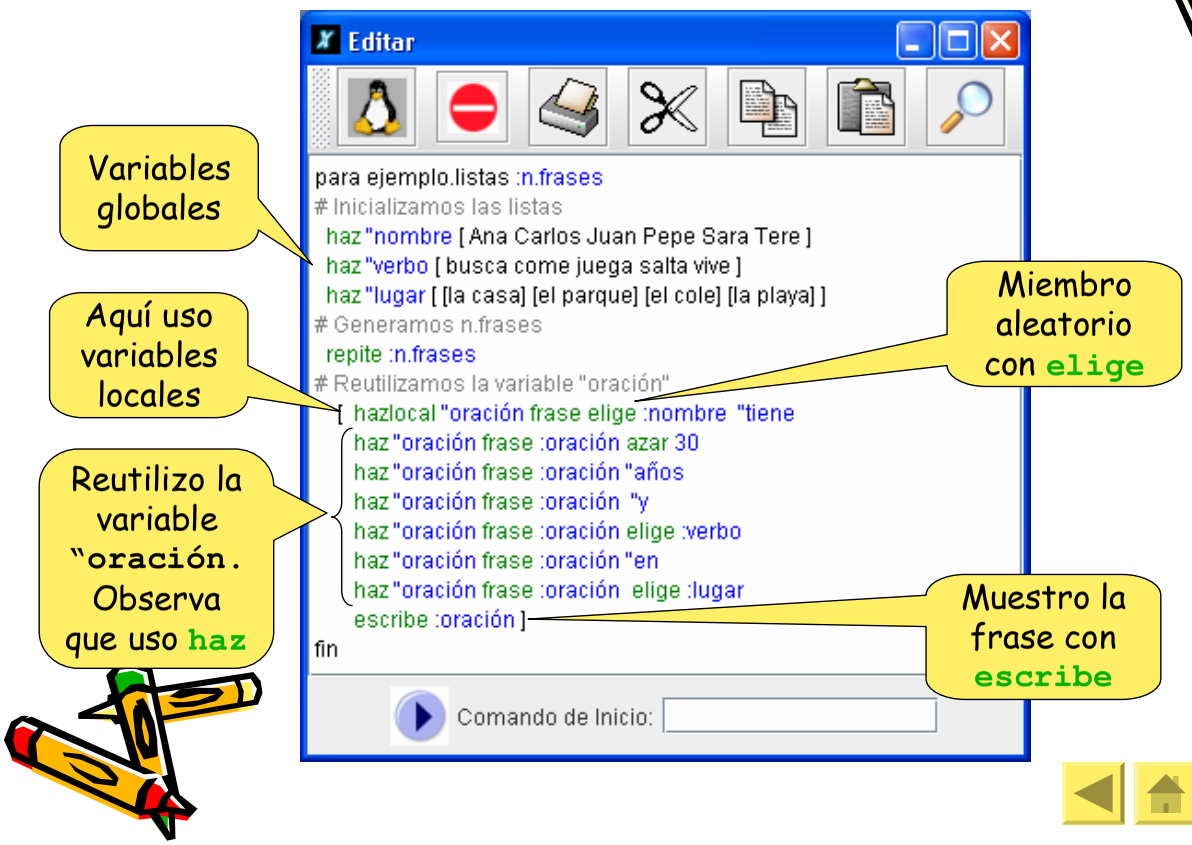
- Presentemos la primitiva **azar**:

  azar n

  genera un número natural aleatorio del conjunto

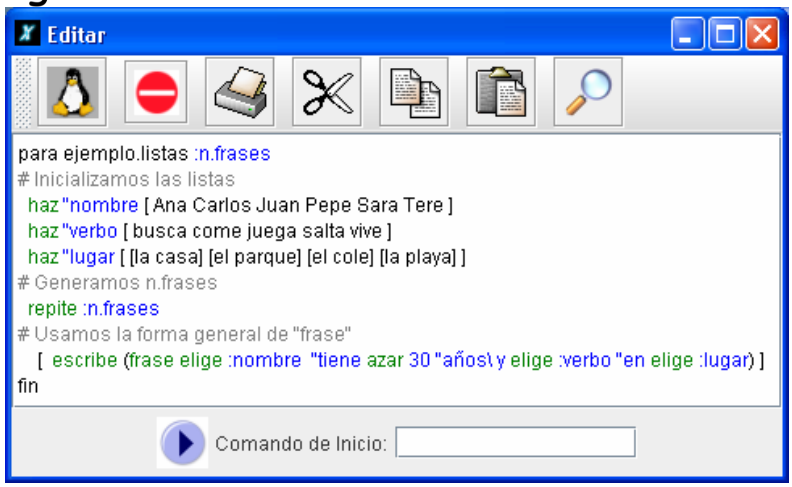
  {0, 1, …, n-1}
- Creemos tres listas:
  - `haz "nombre [Ana Carlos Juan Pepe Sara Tere]`
  - `haz "verbo [busca come juega salta vive]`
  - `haz "lugar`
    `[[la casa] [el parque] [el cole] [la playa]]`
- Inventemos una oración con "huecos":
  - `[nombre]` tiene `[nº]` años y `[verbo]` en `[lugar]`
- Rellenemos esos huecos aleatoriamente, y generemos "n" frases

Primera versión



Simplificando el programa

Usando la forma general de frase se simplifica el código:

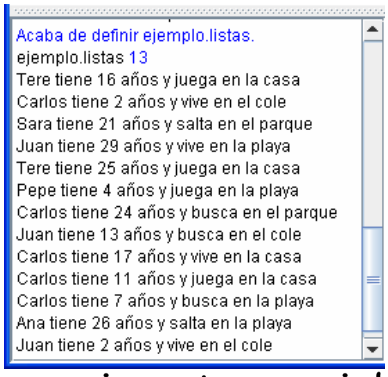


Pueden formarse 3720 frases distintas

Ejecutando el programa

- Tras guardar, ejecutamos haciendo, por ejemplo:

```
ejemplo.listas 13
```



- Para ver la diferencia entre variables globales y locales, prueba a ejecutar:

```
escribe :oración
escribe :nombre
```

en la línea de comandos. ¿Qué observas?

Autoevaluación 15

- ¿Por qué XLogo da un mensaje de error al ejecutar estas secuencias de órdenes ?

```
haz "lado 100
avanza "lado

haz :lado 100
avanza :lado

haz "lado [ 100 ]
avanza :lado
```

Autoevaluación 15

- ¿Por qué XLogo da un mensaje de error al ejecutar estas secuencias de órdenes ?

```
haz "lado 100
avanza "lado
```

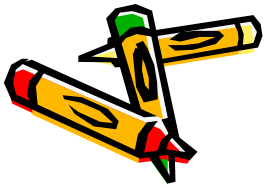
Define bien `"lado`, pero la usa mal. Debe ser `avanza :lado`

```
haz :lado 100
avanza :lado
```

Ahora, usa bien `avanza :lado` pero la define mal: `haz "lado`

```
haz "lado [ 100 ]
avanza :lado
```

`"lado` contiene una lista, no un número. Debe ser `avanza primero :lado`



Autoevaluación 16

- ¿Cómo harías para crear un número aleatorio comprendido entre 1 y 6?

```
azar 6
```

```
azar 6 + 1
```

```
1 + azar 6
```

Autoevaluación 16

- ¿Cómo harías para crear un número aleatorio comprendido entre 1 y 6?

```
azar 6

azar 6 + 1
```

```
1 + azar 6
```

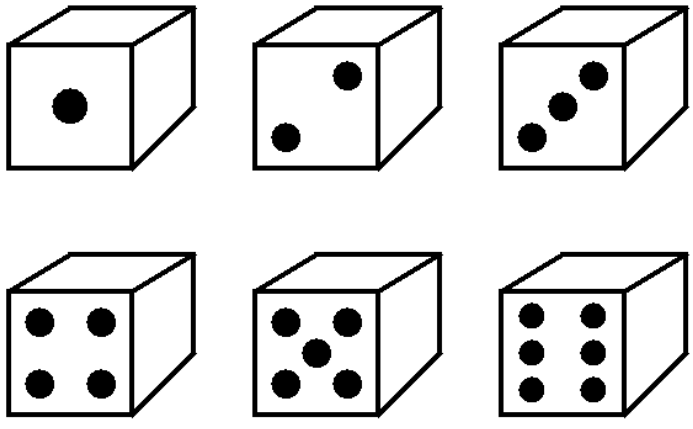
Debemos tener en cuenta la prioridad de las operaciones

Autoevaluación 16

- ¿Cómo harías para crear un número aleatorio comprendido entre 1 y 6?

```
azar 6

azar 6 + 1
```

```
1 + azar 6
```

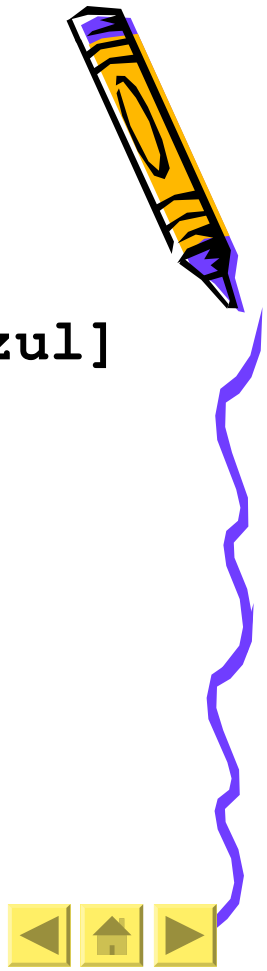



Autoevaluación 17

- ¿Qué aparecerá en el Histórico de Comandos al teclear:

```
haz "colores [rojo verde azul]
escribe :colores ?

colores

rojo verde azul
```

Depende del color seleccionado

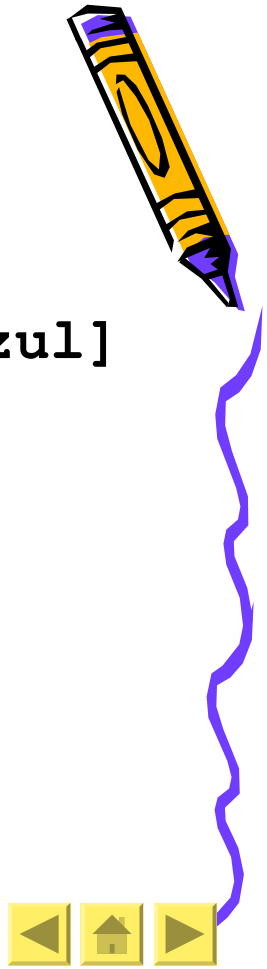


Autoevaluación 17

- ¿Qué aparecerá en el Histórico de Comandos al teclear:

```
haz "colores [rojo verde azul]
escribe :colores ?

colores

rojo verde azul
```

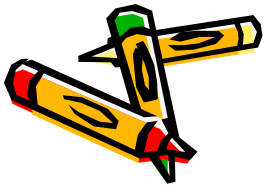
Depende del color seleccionado

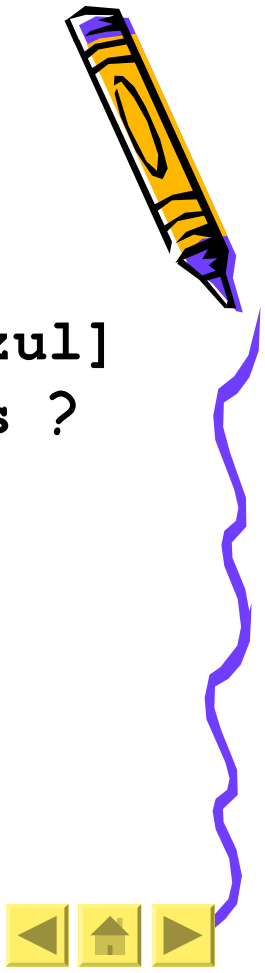


Autoevaluación 18

- ¿Qué aparecerá en el Histórico de Comandos al teclear:

```
haz "colores [rojo verde azul]
escribe elemento 2 :colores ?

        rojo verde

        verde azul

        verde
```

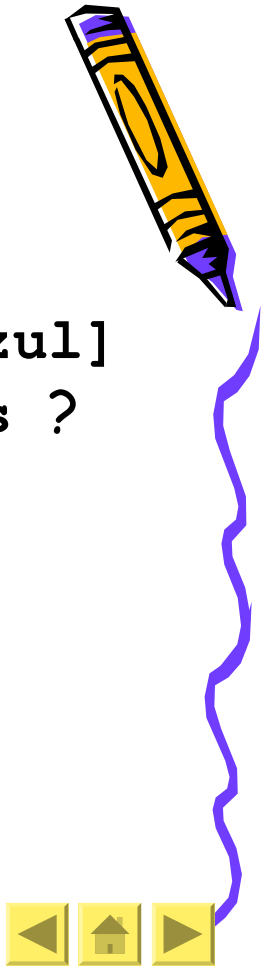




Autoevaluación 18

- ¿Qué aparecerá en el Histórico de Comandos al teclear:

```
haz "colores [rojo verde azul]
escribe elemento 2 :colores ?

        rojo verde

        verde azul

        verde
```

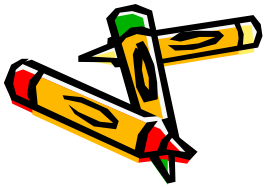


Autoevaluación 19

- ¿Qué aparecerá en el Histórico de Comandos al teclear:

```
haz "colores [rojo verde azul]
haz "colores menosultimo :colores
escribe :colores ?
```

```
rojo verde

verde azul

azul
```



Autoevaluación 19

- ¿Qué aparecerá en el Histórico de Comandos al teclear:

```
haz "colores [rojo verde azul]
haz "colores menosultimo :colores
escribe :colores ?
```

```
rojo verde

verde azul

azul
```

Manejo de palabras

- Uno de los puntos fuertes de LOGO es el manejo de palabras:
  - Contar las letras: `cuenta "palabra`
  - Leer la primera letra: `primero "palabra`
  - Leer la última letra: `ultimo "palabra`
  - Quitar el primero: `menosprimero "palabra`
  - Quitar el último: `menosultimo "palabra`
  - Concatenar palabras: `palabra "palabra1 "palabra2`
- Esto permite desarrollar programas aplicados a Lengua sin excesiva dificultad
- Ejemplos:
  - Determinar la conjugación de verbos (1ª, 2ª o 3ª)
  - Conjugar tiempos verbales
  - Crear nuevas palabras combinando raíz y morfema
  - ...

Ejemplos sobre verbos (I)

El alumno hace *click* sobre el tiempo que quiere conjugar

La tortuga muestra el tiempo verbal

Ejemplos sobre verbos (II)

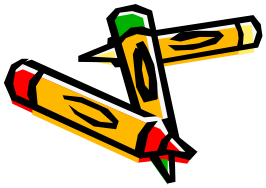
http://wiki.gleducar.org.ar/wiki/Jugando_con_verbos

http://coleccion.educ.ar/coleccion/CD7/actividades/verbos_solapa01.htm

Presente	Otra vez	Verbo
yo am		a
tú am		as
él am		áis
nosotros am		an
vosotros am		o
ellos am		amos

El alumno hace *click* sobre la terminación adecuada

La tortuga indica con el color del guión si está bien o mal



Presente	Otra vez	Verbo
- yo amo		a
- tú amas		as
- él ama		áis
- nosotros amamos		an
- vosotros amáis		o
ellos am		amos

Ejemplos sobre verbos (II)

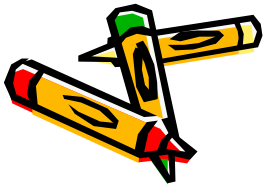
http://wiki.gleducar.org.ar/wiki/Jugando_con_verbos

http://coleccion.educ.ar/coleccion/CD7/actividades/verbos_solapa01.htm

Pasado	Otra vez	Verbo
yo am		aste
tú am		amos
él am		ó
nosotros am		asteis
vosotros am		é
ellos am		aron

El alumno hace *click* sobre la terminación adecuada

La tortuga indica con el color del guión si está bien o mal



Pasado	Otra vez	Verbo
- yo amé		aste
- tú amasteis		amos
- él amaron		ó
- nosotros amamos		asteis
- vosotros amaste		é
ellos am		aron

Ejemplos sobre verbos (II)

http://wiki.gleducar.org.ar/wiki/Jugando_con_verbos

http://coleccion.educ.ar/coleccion/CD7/actividades/verbos_solapa01.htm

Futuro	Otra vez	Verbo
yo amar		emos
tú amar		án
él amar		ás
nosotros amar		éis
vosotros amar		é
ellos amar		á

El alumno hace *click* sobre la terminación adecuada

La tortuga indica con el color del guión si está bien o mal

Futuro	Otra vez	Verbo
- yo amará		emos
- tú amaréis		án
- él amaré		ás
- nosotros amaremos		éis
- vosotros amarás		é
ellos amar		á

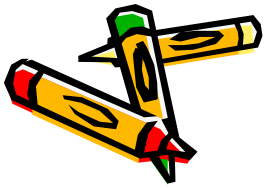
Curso de *xLogo*

BLOQUE 4

Requisitos previos

Antes de empezar esta parte el alumno debe:

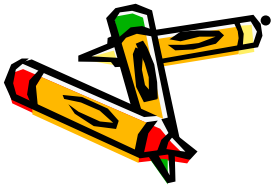
- Poder reproducir formas complejas
- Dominar el uso de variables, listas, procedimientos y condicionales
- Comprender el uso de **`repite`** para simplificar acciones repetitivas
- Saber ubicar y orientar a la tortuga en el sistema cartesiano que constituye el **Área de Dibujo**



Objetivos

Al concluir este bloque el alumno sabrá:

- Utilizar otras estructuras iterativas para simplificar acciones repetitivas
- Describir y utilizar la recursividad
- Manejar opciones de control del **Área de Dibujo** y del **Histórico de Comandos**
- Cómo interactuar con un "usuario" mediante mensajes, botones y menús
- Desenvolverse usando varias tortugas



Bucles y Recursividad

- Un **bucle** es una sentencia que se realiza repetidas veces
- Para repetir una acción sin copiar varias veces el mismo código.
  - Ahorra tiempo
  - Deja el código más claro
  - Facilita futuras modificaciones.
- Debe contener condiciones que establezcan cuándo empieza y cuándo acaba
- XLogo tiene tres primitivas para construir bucles:
  - **`repite`**
  - **`repitepara`**
  - **`mientras`**

Otra forma de lograr repeticiones es la recursividad o recurrencia

Bucles `repite`

- Ya vimos anteriormente su sintaxis:

```
repite n
    [ acciones a repetir ]
```

- Útil si sabemos cuántas repeticiones han de hacerse
- El número de repeticiones puede ser una variable
- Define una variable interna:

`contador`

que determina la iteración en curso

Bucles `repitepara`

- Su sintaxis es:

```
repitepara [lista1]
    [ acciones a repetir ]
```

- `lista1` consta de:
  - Un contador
  - Los límites inferior y superior
  - El paso (cualquier real) - opcional
- Ejemplos:

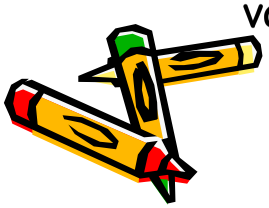
```
repitepara [i 1 4]
```

varía `:i` desde 1 hasta 4 de uno en uno

```
repitepara [i 31 4 -1.5]
```

varía `:i` desde 31 hasta 4 bajando de 1.5 en 1.5

Útil si sabemos cuántas repeticiones han de hacerse, tenemos contadores anidados y queremos variar el paso



Bucles `mientras`

- Su sintaxis es:

```
mientras [condición]
    [ acciones a repetir ]
```

- `condición` debe ser lógica, y dar `cierto` o `falso`
- Se ejecuta hasta que `condición` sea `falso`
- Útil cuando no se sabe cuántas veces hay que repetir la secuencia

Ejemplo:

```
mientras [no vacio? :lista ]
    [ escribe primero :lista
      haz "lista menosprimero :lista ]
```

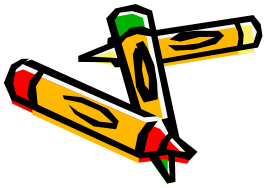
Vacía la variable `"lista` mientras la va escribiendo elemento a elemento sin saber cuantos elementos tiene

Más sobre bucles

- Si queremos detener un bucle en algún momento, disponemos de la primitiva

  `alto`

- Para un mismo cometido suele ser posible usar los tres tipos de bucles
- Cambia la sencillez/dificultad de su programación
- Elegir el bucle correcto es "un arte"
- Un bucle mal definido puede:
  - No llegar a su objetivo
  - Producir resultados "extraños"
  - "Colgar" al sistema: Bucle infinito



Autoevaluación 20

- La orden siguiente debería dibujar un triángulo, pero hay errores. ¿Los ves?

```
repetir 3 [avanza100 giarderecha 60]
```

Autoevaluación 20

- La orden siguiente debería dibujar un triángulo, pero hay errores. ¿Los ves?

```
repetir 3 [avanza100 giarderecha 60]
```

repite

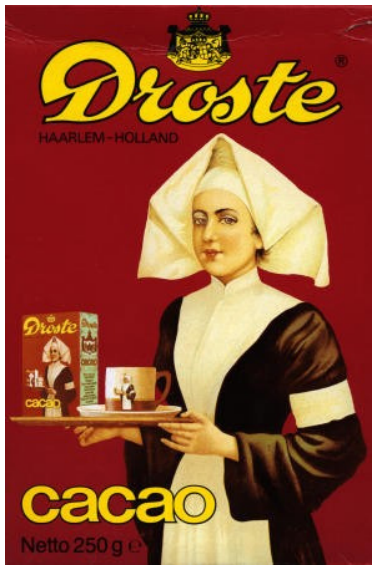
avanza 100

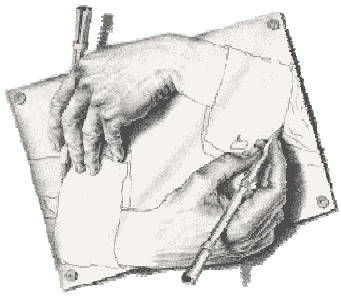
giraderecha

120

Recursividad o recurrencia

- Para entender la recursividad, vete a la diapositiva sobre recursividad.
- Ejemplos gráficos:





Programando recurrencia

- Primero y muy importante: Debemos saber cómo y cuándo parar:
  - Condicional
  - Primitiva `alto`.
- Recordemos el ejemplo del bucle **`mientras`**:

```
mientras [no vacio? :lista ]
     [ escribe primero :lista
       haz "lista menosprimero :lista ]
```

- Podemos convertirlo en un procedimiento recursivo:

```
para escribe.todo :lista
  si vacio? :lista
    [ alto ]
    [ escribe primero :lista
      escribe.todo menosprimero :lista ]
fin
```

Más recurrencia

- Introduzcamos una *lista de listas*.:

```
escribe.todo [ Hola [esto es] [un ejemplo] ]
```

devuelve:

```
Hola
esto es
un ejemplo
```

- Si hago los siguientes cambios:

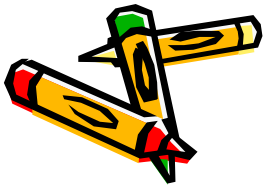
```
para escribe.todo :lista
  si no vacio? :lista
    [ escribe primero :lista
      si lista? primero :lista
        [ escribe.todo primero :lista ]
      escribe.todo menosprimero :lista ]
fin
```

¿qué devolverá la misma orden anterior?

Autoevaluación 21

- ¿Ves algún problema en este programa recursivo?

```
para factorial :n
  devuelve :n * (factorial :n+1)
fin
```

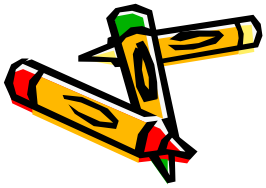


Autoevaluación 21

- ¿Ves algún problema en este programa recursivo?

```
para factorial :n
  devuelve :n * (factorial :n+1)
fin
```

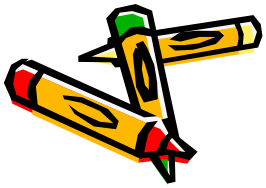

No tiene condición que lo detenga. Irá acumulando el valor del producto hasta producir un *overflow*: sobrepasar el máximo valor que puede mostrar



Autoevaluación 22

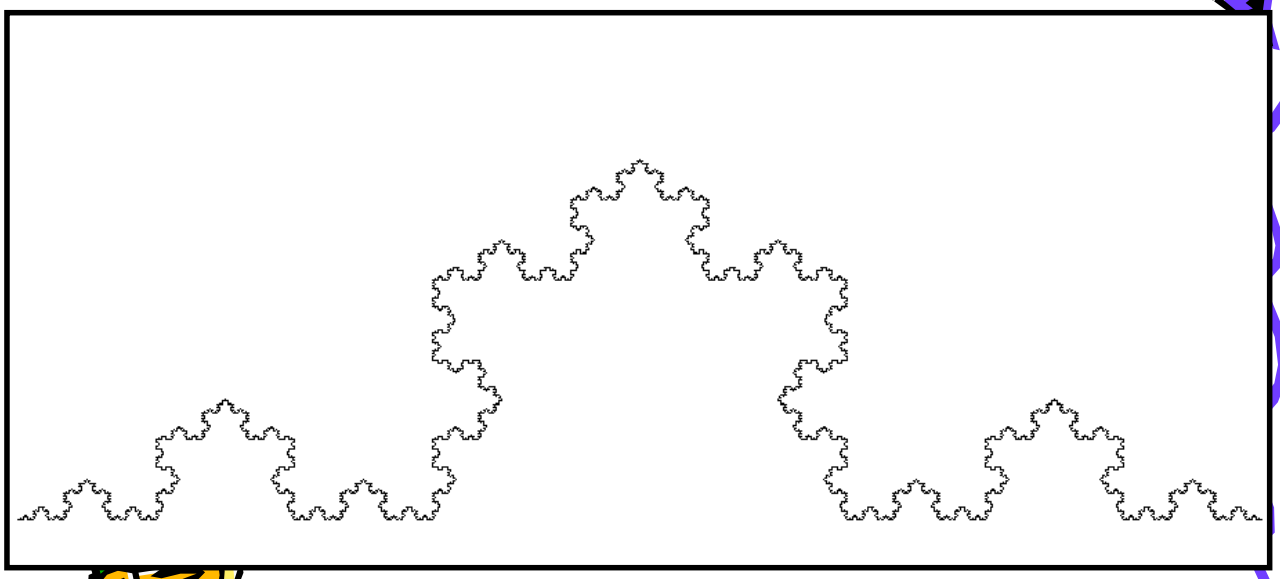
- Este es algo difícil: ¿Qué hace esta recurrencia?

```
para algo :orden :largo
  si (:orden < 1)
   [ avanza :largo
      alto ]
   algo (:orden -1) (:largo/3)
   giraizquierda 60
   algo (:orden -1) (:largo/3)
   giraderecha 120
   algo (:orden -1) (:largo/3)
   giraizquierda 60
   algo (:orden -1) (:largo/3)
fin
```



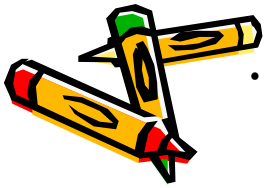
Autoevaluación 22

- Este es algo difícil: ¿Qué hace esta recurrencia?



Interacción con usuario

- Podemos recibir información mediante:
  - Teclado
  - Ratón
  - Ventanas emergentes
  - Botones
  - Menús
- Mostraremos respuestas a través de:
  - Ventanas emergentes
  - Mensajes:
    - En el Área de Dibujo
    - En el Histórico de Comandos



Devolver respuestas (I)

- Ya conocemos la primitiva **escribe**:

  escribe Argumento

  muestra **Argumento** en el Histórico de Comandos y produce un salto de línea
- Similar es **tipea**:

  tipea Argumento

  muestra **Argumento** en el Histórico de Comandos, pero NO produce un salto de línea
- Crear una ventana emergente

  mensaje Argumento

  muestra una ventana emergente con **Argumento**
- Escribir en el Área de Dibujo:

  rotula Argumento

Devolver respuestas (II)

- En todos los casos:
  - **Argumento** puede ser un número, lista o palabra.
  - Las operaciones permiten que el mensaje sea complejo:

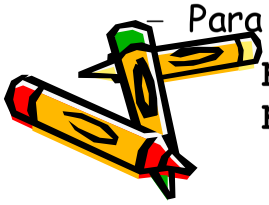
```
escribe (frase "En 2008 "cumplo :edad "años)
tipea (frase "En 2008 "cumplo :edad "años)
mensaje (frase "En 2008 "cumplo :edad "años)
rotula (frase "En 2008 "cumplo :edad "años)
```

- Podemos controlar la fuente, el tamaño, el color y el estilo con varias primitivas:
  - Para el Histórico de Comandos

```
ponfuentetexto          poncolortexto
ponestilo               ponnombrefuentetexto
```

  - Para `rotula`:

```
ponfuente               poncolorlapiz
ponnombrefuente
```



Recibir información

- Podemos pedir información con **leelista**:

```
leelista [título] "variable
```

  abre una ventana titulada **título** donde puede escribirse una respuesta, que se almacena en **"variable**

- Leer qué se pulsa en el teclado con **leetecla**:

```
haz "variable leetecla
```

  guarda en **"variable** el valor numérico asociado a la tecla pulsada

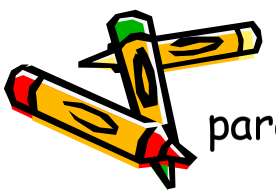
- Interactuar con el ratón

```
haz "variable leeraton
```

  y

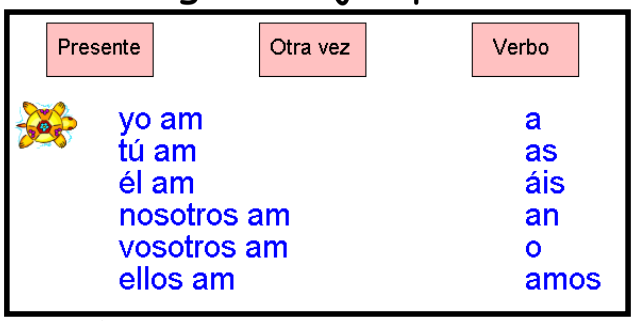
```
haz "variable posraton
```

  para la acción del ratón y su posición, resp



Ejemplo

- Recordemos el segundo ejemplo sobre verbos:



- Se usa intensivamente **`rotula`**
- El juego es una recursión, sólo se detiene pulsando el botón **`Alto`**
- Determina si hay "*click* izquierdo" en una zona concreta
  - Si es en "Nuevo verbo", muestra un **`mensaje`**.
  - ...

Interfaz Gráfica de Usuario

- Más llamativas que las anteriores.
- Botones y menús vistos en los ejemplos
- Las primitivas asociadas terminan con **`igu`**:
  - Para botones: **`botonigu "nombre "Leyenda`**
  - Para menús: **`menuigu "nombre [Lista opciones]`**
- Debemos definir su posición:

  `posicionigu "nombre [coordenadas]`

  definir su acción (acciones si es un menú):

  `accionigu "nombre [Lista de acción/es]`

  y luego dibujarlo en pantalla:

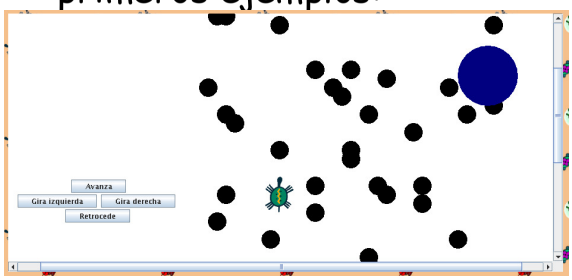
  `dibujaigu "nombre`

  - Para borrarlos:

    `eliminanigu "nombre`

Ejemplo

- Recordemos uno de los primeros ejemplos:



- Sólo hay cuatro botones
- Las acciones son muy sencillas:
  - **`avanza 10`**
  - **`retrocede 10`**
  - **`giraderecha 90`**
  - **`giraizquierda 90`**
- "Piedras" colocadas con **`azar`**
- ...

El resto del código está aquí.

Autoevaluación 23

- ¿Qué orden permite leer un solo caracter del teclado?

`haz "pulsado tecla?`

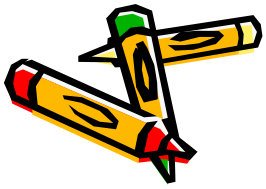
`haz "pulsado leetecla`

`haz "pulsado leelista`

Autoevaluación 23

- ¿Qué orden permite leer un solo caracter del teclado?

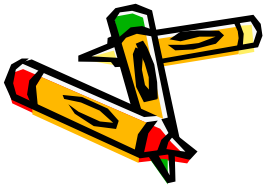
```
haz "pulsado tecla?

haz "pulsado leetecla

haz "pulsado leelista
```



Autoevaluación 24

- Quieres mostrar una ventana que diga "**Dime tu nombre**" y almacene la respuesta en "**nombre**:

```
leelista [Dime tu nombre] "nombre

haz "nombre leelista [Dime tu nombre]

haz leelista [Dime tu nombre] "nombre
```



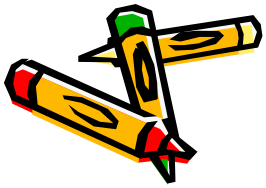
Autoevaluación 24

- Quieres mostrar una ventana que diga "**Dime tu nombre**" y almacene la respuesta en "**nombre**:

```
leelista [Dime tu nombre] "nombre
```

```
haz "nombre leelista [Dime tu nombre]
```

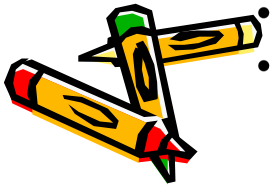
```
haz leelista [Dime tu nombre] "nombre
```



Dibujo Avanzado

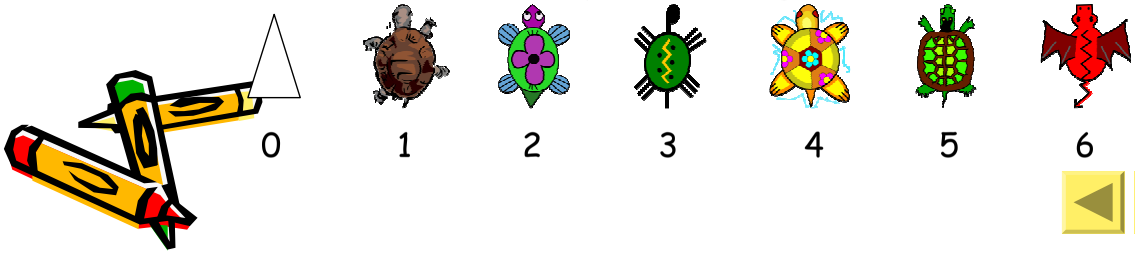
Otras opciones del Área de dibujo

- Borrar sin volver al centro: **limpia**
- Mirar el color del punto en que se encuentra: **encuentracolor**
- Podemos controlar:
  - El tamaño de la pantalla: **pontamañopantalla**
  - Color: **poncolorpapel**
  - Calidad: **poncalidaddibujo**
- Cómo se comporta la tortuga en los bordes:
  - **modojaula**: no puede salir y muestra error
  - **modovuelta**: reaparece por el otro lado
  - **modoventana**: puede salir, pero no dibuja
- Aumentar o disminuir la visualización: **zoom**
- ...



Multitortuga

- Permite utilizar varias tortugas en pantalla
- Cada tortuga puede tener una forma distinta, de entre las 7 posibles en XLogo:
- Las primitivas son:
  - **pontortuga n**: la tortuga **n** pasa a ser la activa
  - **tortuga**: devuelve el nº de la tortuga activa
  - **eliminatortuga n**: quita la tortuga **n**
  - **tortugas**: da una lista con los nºs de las tortugas activas
- Cambiar la forma de la tortuga:
  - **ponforma n**: cambia el dibujo de la tortuga activa
  - **forma**: devuelve el tipo de tortuga de la activa



Ejemplo de multitortuga

Piensa un minuto qué harías para conseguir esta "rosca"

Ejemplo de multitortuga

Editar

```
para portada
  ocultatortuga                  # Primero dibujaré la "sombra"
  ponfuente 110                  # Tamaño de letra
  ponnombrefuente 307            # Tipo de letra
  poncolorlapiz cyan             # Lápiz azul claro
  haz "posicion largoetiqueta "xLogo  # Colocamos la tortuga en
  subelapiz                      #      función de la longitud
  ponxy (-:posicion)/4+3 (-22)   #      de la etiqueta
  rotula "xLogo                  # Rotulamos
#
  poncolorlapiz negro            # Lápiz negro para el rótulo
  subelapiz                      #      principal
  ponxy (-:posicion)/4 (-20)     # Algo más atrás y más arriba
  rotula "xLogo                  # Rotulamos
#
  haz "angulo 360/14    # Distribuyo uniformemente 14 tortugas
  repitepara [n 1 14]
    [ haz "tortuga resto :n 7    # Cuando "n > 7, vuelve a ser 0
      pontortuga :n              # Tortuga activada
      muestratortuga             # Tortuga activa es visible
      ponforma :tortuga          # Tipo de tortuga
      subelapiz
      ponxy 0 0                  # Por si acaso, al centro
      giraderecha :angulo * :n   # Giramos el ángulo adecuado
      avanza 110 ]               # Avanzamos a la posición final
fin
```

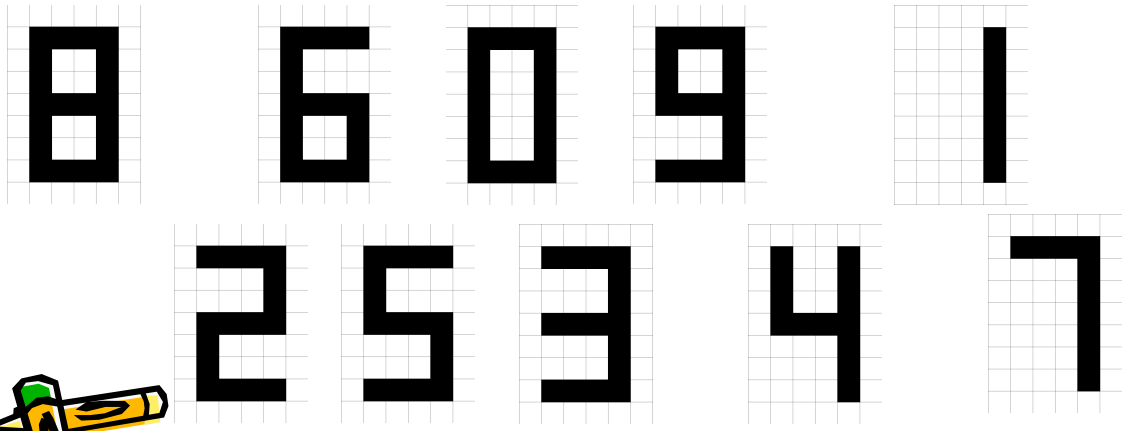
Comando de Inicio:

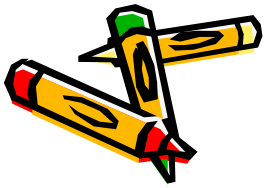
Animación

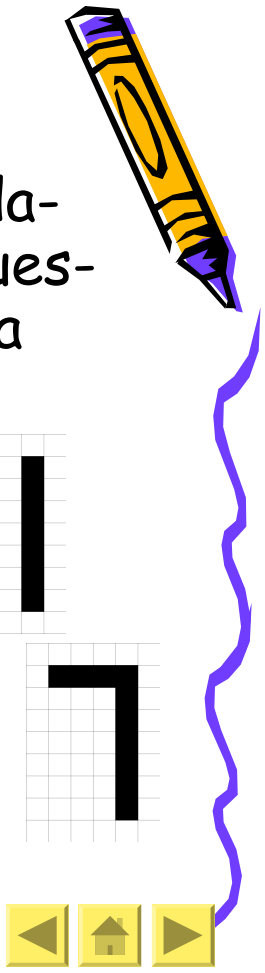
- A veces, determinados dibujos complejos hacen que la tortuga "dé saltos" al dibujar
- Otras veces, directamente buscamos mostrar una animación en pantalla
- Una animación consiste en varios dibujos terminados que son mostrados consecutivamente
- La tortuga dibuja en "*background*", sin mostrar nada hasta que está terminado
- Las primitivas son tres:
  - **`animacion`**: inicia el modo animación. La tortuga "dibujará" en la memoria del ordenador
  - **`refrescar`**: muestra el dibujo almacenado en memoria
  - **`detieneanimacion`**: termina el modo animación

Animación - Ejemplo

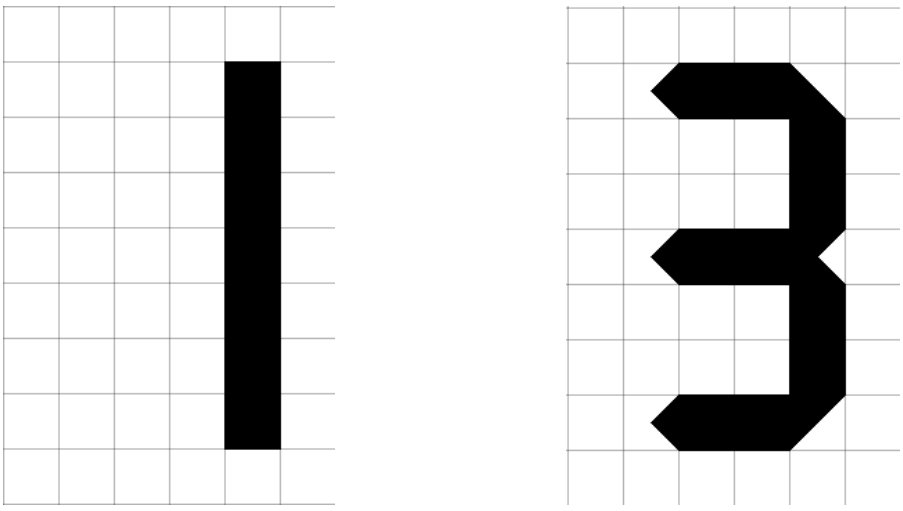
- Combinando rectángulos adecuadamente, podemos simular cómo muestra los números la pantalla de una calculadora:

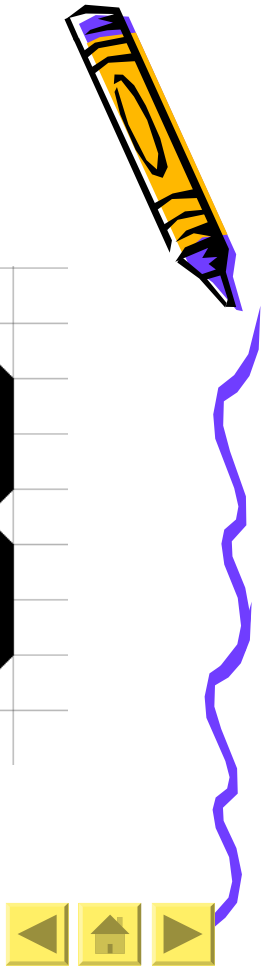






Animación - Ejemplo





Curso de xLogo

BLOQUE 5

Requisitos previos

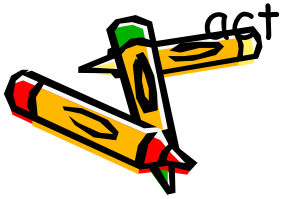
Antes de empezar esta parte el alumno debe:

- Poder reproducir "cualquier" forma planteada
- Saber depurar programas y corregir los errores que existan
- Saber interactuar con un usuario
- Comprender la estructura de archivos de un ordenador
- Tener nociones musicales básicas

Objetivos

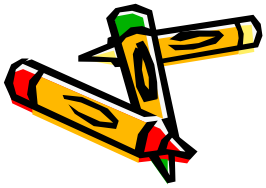
Al concluir este bloque el alumno sabrá:

- Moverse por el árbol de directorios de su disco duro
- Interpretar piezas sencillas de música
- Manejar correctamente las primitivas aplicadas al tiempo
- Chatear con otros ordenadores, ejecutar procedimientos de forma remota y otras actividades asociadas al uso de redes



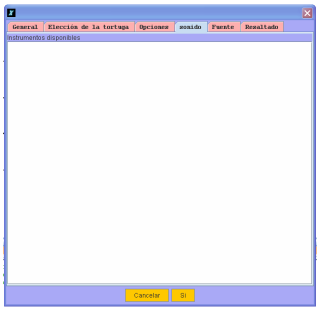
Manejo de Archivos

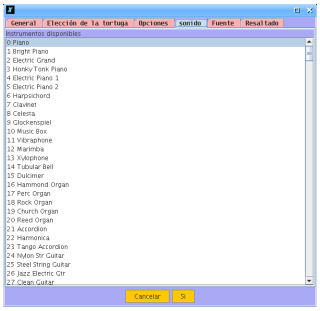
- Primitivas:
  - **catalogo**: lista el contenido del directorio actual
  - **pondirectorio**: cambia la ruta de trabajo al directorio especificado de forma absoluta
  - **cambiadirectorio**: cambia al directorio especificado de forma relativa
  - **guardatodo**: graba en disco los procedimientos
  - **cargaimagen**: muestra en el Área de Dibujo la imagen (jpg o png) especificada
- Podemos abrir y leer archivos de texto:
  - **abreflujo**: abre un fichero para lectura/escritura
  - **cierraflujo**: cierra el fichero
  - **leelineaflujo**: lee una línea del fichero
  - **escribelineaflujo**: escribe una línea en el fichero
  - **agregalineaflujo**: añade una línea al final

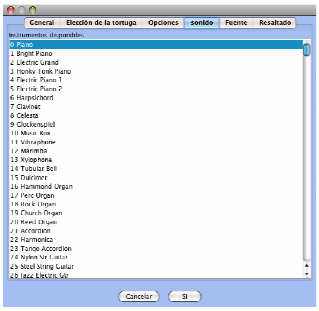


Música

- Desgraciadamente, la versión **JAVA** para Windows no incluye los instrumentos para tocar música
- En Linux y Mac no existe ese problema







- Debemos descargarlos desde:

http://java.sun.com/products/java-media/sound/soundbanks.html

y descomprimirlos al directorio java de nuestro disco duro, C:\Archivos de programa\Java\jre1.6.0\lib\audio

creando el directorio audio si éste no existe

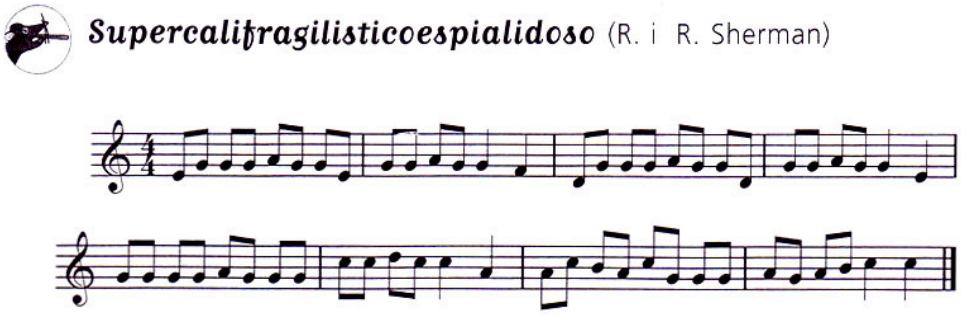
Desde este momento, habrá música en Windows

Primitivas musicales

- Las primitivas son:
  - **borrasecuencia**: limpia cualquier melodía de memoria
  - **secuencia**: crea la melodía y la carga en memoria
  - **tocamusica**: toca la secuencia almacenada
  - **poninstrumento**: cambia el instrumento
  - **ponindicesecuencia**: cambia el puntero a la posición de la nota deseada
  - **indicesecuencia**: devuelve la posición de la nota actual

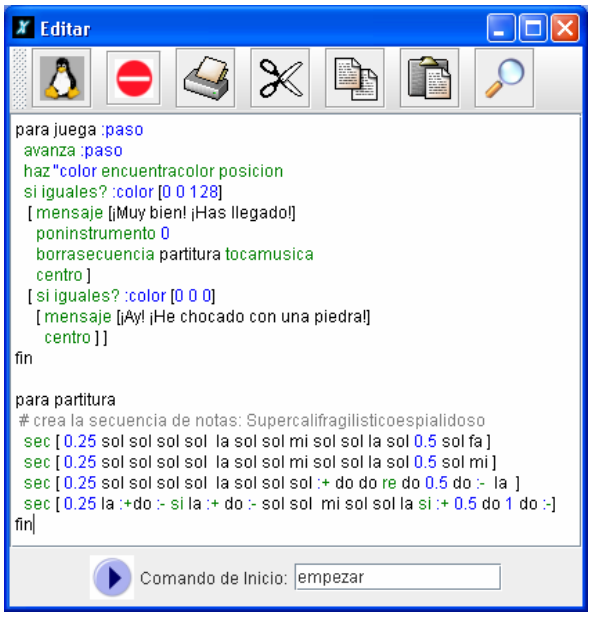
Ejemplo con música

- Tanto el ejemplo de llevar la tortuga hacia el lago como el de verbos tienen "efectos musicales"
- Veamos cómo "interpretar" la canción de
  Supercalifragilisticoespialidoso
- Suele ser bueno tener la partitura:



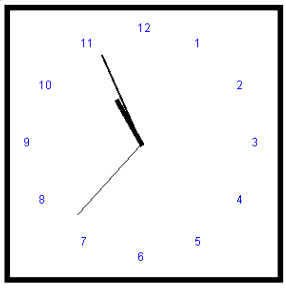
Ejemplo con música: código

- Completando el código mostrado en diapositivas anteriores:



Gestión de Tiempos

- Sirven para:
  - Conocer fecha y hora:
    - **fecha**, **hora**: Sin argumentos
  - Realizar pausas y cuentas atrás
    - **espera**, **crono** y **fincrono?**
  - Determinar la duración de una ejecución
    - **tiempo**: Sin argumentos
- Ejemplo: **reloj.lgo**



La Red

- XLogo permite conectar dos PC's de manera que entre ellos pueden:
  - Ejecutar código.
  - Establecer sesiones de **chat**.
- Las primitivas son cuatro:
  - **escuchatcp**: pone al PC "en espera"
  - **ejecutatcp "nombre [lista]**: ejecuta las órdenes contenidas en **lista** en el ordenador llamado "**nombre**
  - **enviatcp "nombre [lista]**: envía la información guardada en **lista** a "**nombre**, debiendo enviarse como una orden completa:

```
haz "variable enviatcp "liebre [rojo verde azul]
escribe enviatcp "liebre [rojo verde azul]
```

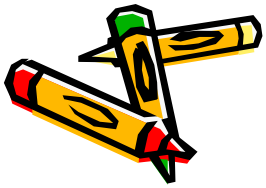
  - **chattcp "nombre [lista]**: abre una ventana de chat en "**nombre** mostrando el contenido de **lista**

La Red - Uso

- No sólo podemos conectar PC's.
- La robótica es posible en XLogo.
- Marcelo Dushkin mantiene la *web*:

  http://www.miprimerrobot.com.ar
- Vídeo de M. D. mostrando cómo XLogo controla una máquina construida con **Rasti**

http://www.youtube.com/watch?v=pAGoU5g6Feo

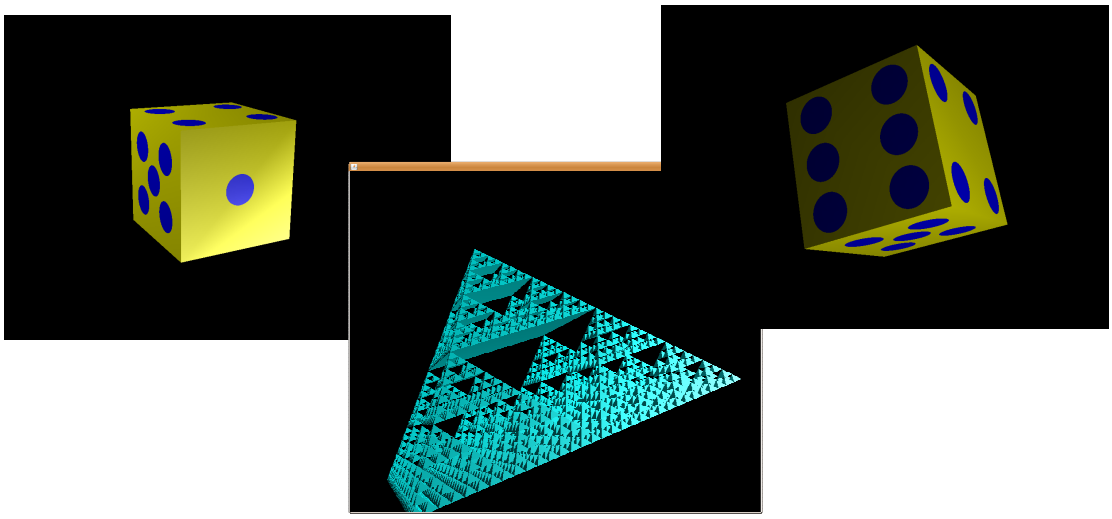


Para concluir

- Cimientos sólidos; límite: el Cielo
- No hagamos que los límites de los alumnos sean nuestros límites.
- La informática para ellos es algo cotidiano, nos sacan ventaja

Futuro Inmediato

Tres dimensiones:



Listas de Propiedades

Gracias por su atención

xLogo