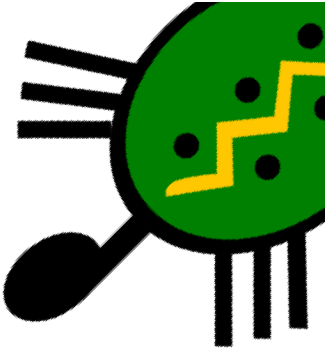


Capítulo 16

Geometría de la tortuga en 3-D

16.1. La tortuga en Tres Dimensiones

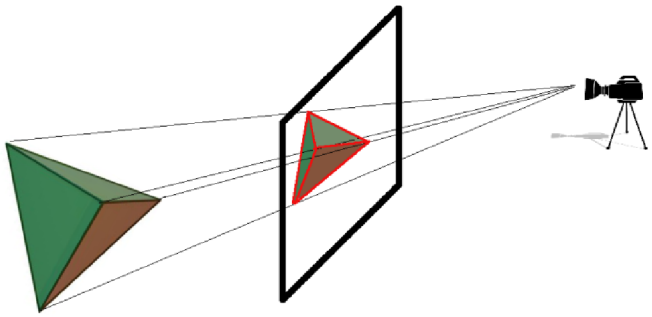
Desde la versión 0.9.92, nuestra tortuga puede dejar el plano para trasladarse a un espacio en tres dimensiones (3D). Para cambiar a esta modalidad, Usaremos la primitiva `perspectiva`. ¡Bienvenido a un mundo en 3D!

La tortuga cambia de forma y orientación, indicándonos en qué modo nos encontramos:

Para recuperar el modo bidimensional (2D), debemos indicarle que vuelva a uno de los modos "planos": `modojaula`, `modoventana` o `modovuelta`.

16.1.1. La proyección en perspectiva

Para representar un espacio 3D en un plano 2D, xLogo utiliza una proyección en perspectiva. Es equivalente a tener una cámara grabando la escena en 3D, y mostrando en la pantalla la imagen de la proyección. Veamos un esquema gráfico para explicarlo mejor:



Disponemos de primitivas para fijar la posición de la cámara, mientras que la pantalla de proyección se encuentra en el punto medio entre la cámara y el objeto.

16.1.2. Entender la orientación en el mundo tridimensional

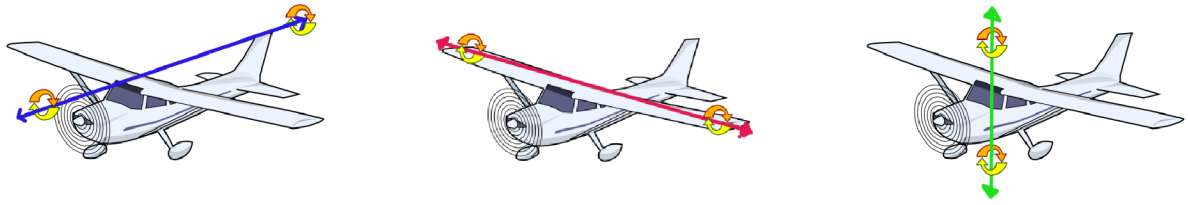
En el plano, la orientación de la tortuga se define únicamente por su rumbo. Sin embargo, en el mundo tridimensional la orientación de la tortuga necesita de tres ángulos. Si usamos la orientación por defecto de la tortuga en 3D (en el plano `XY` mirando hacia el semieje `Y` positivo):

Balanceo: la inclinación alrededor del eje OY

Cabeceo: la inclinación según el eje OX

Rumbo: la inclinación según el eje OZ

De hecho, para moverse en el mundo tridimiensional, la tortuga se comportará de modo muy similar a un avión. De nuevo, ilustremos con una imagen los 3 ángulos:



Balanceo **Cabeceo** **Rumbo**

Parece bastante complicado la primera vez que se estudia, pero veremos que muchas cosas son similares a los movimientos en el plano bidimensional.

16.1.3. Primitivas

Estas son las primitivas básicas para moverse en el mundo 3D:

Primitiva	Forma larga
Pasar al modo 3D	`perspectiva`
Salir del modo 3D	`modojaula`, `modovuelta` o `modoventana`
Baja el "morro" `n` grados	`cabeceaabajo n` ó `bajanariz n`, `bn n`
Sube el "morro" `n` grados	`cabeceaarriba n` ó `subenariz n`, `sn n`

Sube "ala" izquierda y baja "ala" derecha `n` grados	`balanceaderecha n`, `bd n`
Sube "ala" derecha y baja "ala" izquierda `n` grados:	`balanceaizquierda n`, `bi n`
Borrar Pantalla (y tortuga al centro, orientada "hacia nosotros" con el "timón de cola" hacia arriba)	`borrapantalla`, `bp`
Devuelve la inclinación de las "alas"	`balanceo`
Devuelve la inclinación del "morro"	`cabeceo`

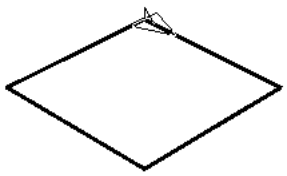
Además de las anteriores, podemos usar algunas de nuestras "viejas conocidas":

- `avanza`, `av`
- `retrocede`, `re`
- `giraderecha`, `gd`
- `giraizquierda`, `gi`

que realizan los mismos movimientos que en el "mundo 2D".
Por ejemplo, en el plano bidimensional, para dibujar un cuadrado de 200 pasos de tortuga, escribimos:

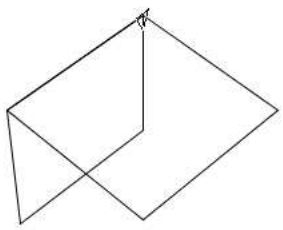
```
repite 4 [ avanza 200 giraderecha 90 ]
```

Estas órdenes siguen existiendo el mundo 3D, y el cuadrado puede dibujarse perfectamente en modo `perspectiva`:



Si la tortuga baja "el morro" 90 grados, podemos dibujar otro cuadrado, y obtenemos:

```
borrapantalla
repite 4 [ avanza 200 giraderecha 90 ]
bajanariz 90
repite 4 [ avanza 200 giraderecha 90 ]
```



Puedes (debes) probar otros ejemplos para entender perfectamente la orientación de la tortuga y el uso de los ángulos y ¡convertirte en un experto!

También debes entender que las tres primitivas que controlan la rotación en 3D están relacionadas entre sí; por ejemplo, al ejecutar:

```
borrapantalla
balanceaizquierda 90 subenariz 90 balanceaderecha 90
```

El movimiento de la tortuga es equivalente a:

```
giraizquierda 90
```

es decir, que como ocurre en 2D, no todas las primitivas son necesarias, por ejemplo:

```
bajanariz 90
```

es equivalente a:

```
balancea 90 giraderecha 90 balanceaizquierda 90
```

(Puedes probar con tu mano si no lo entiendes bien)

16.2. Primitivas disponibles tanto en 2D como 3D

Las siguientes primitivas están disponibles en el plano o en el mundo 3D. La única diferencia son los argumentos admitidos por las primitivas. Estas precisan de los mismos argumentos que en el plano:

`circulo`	`arco`	`centro`
`ponx`	`pony`	`coordenadax`
`coordenaday`	`rumbo`	`ponrumbo`
`rotula`	`largoetiqueta`	

Las siguientes primitivas siguen esperando una lista como argumento, pero ahora debe contener **tres** argumentos, correspondientes a las tres coordenadas de un punto en el espacio: `[x y z]`.

`hacia`	`distancia`	`pos`, `posicion`
`ponpos`, `ponposicion`	`punto`	

16.3. Primitivas sólo disponibles en 3D

- `ponxyz` Esta primitiva mueve a la tortuga al punto elegido. Esta primitiva espera tres argumentos que representan las coordenadas del punto.

  `ponxyz` es muy similar a `ponposicion`, pero las coordenadas no están escritos en una lista.

  Ejemplo, `ponxyz -100 200 50` traslada a la tortuga hasta el punto `x = -100`; `y = 200`; `z = 50`

- `ponz` Esta primitiva mueve a la tortuga al punto de "altura" (desconozco si el término *applikate* usado en Alemania tiene traducción al castellano más allá de **tercera coordenada**) dada. `ponz` recibe un número como argumento, de modo idéntico a `ponx` y `pony`

- `coordenadaz` o `coordz` Devuelve la "altura" de la tortuga. Es equivalente a `ultimo posicion`, pero simplifica la escritura.

- `ponorientacion` Fija la orientación de la tortuga. Esta primitiva espera una lista que contiene tres números: `[balanceo cabeceo rumbo]`

  Ejemplo: `ponorientacion [100 0 58]`: la tortuga tendrá balanceo: 100 grados, cabeceo: 0 grados y rumbo: 58 grados.

  Por supuesto, el orden de los números es importante. Si, por ejemplo, el valor de la orientación es `[100 20 90]`, esto significa que si quieres esa misma orientación partiendo del origen (después de un `borrapantalla`, por ejemplo) deberás escribir la siguiente secuencia:

  ```
  cabeceaderecha 100
  subenariz 20
  giraderecha 90
  ```

  Si en esta instrucción cambiamos el orden, no obtendremos la orientación deseada.

- `orientacion` Devuelve la orientación de la tortuga en una lista que contiene: `[balanceo cabeceo rumbo]`

- `ponbalanceo` La tortuga gira en torno a su eje longitudinal y adquiere el ángulo de balanceo elegido.

- `balanceo` Devuelve el valor actual del balanceo

- `ponbalanceo` La tortuga gira en torno a su eje transversal, y se orienta con el ángulo de cabeceo indicado.

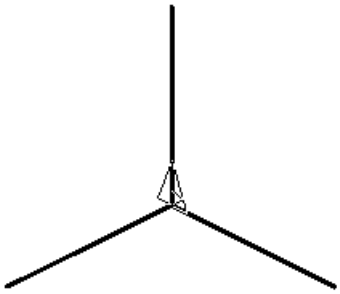
- `balanceo` Devuelve el valor actual del cabeceo

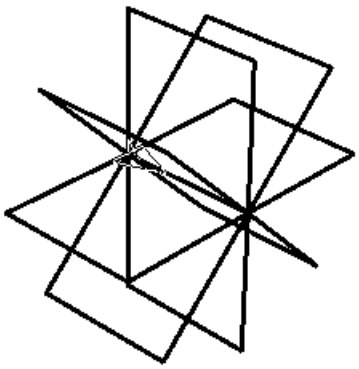
Podemos dibujar los semiejes en la posición habitual tecleando, por ejemplo:

semieje y+: `borrapantalla avanza 200 retrocede 200`

semieje x+: `giraderecha 90 avanza 200 retrocede 200`

semieje z+: `subenariz 90 avanza 200 retrocede 200`





Podemos dibujar 8 planos verticales, cada uno girado respecto del anterior 45 grados, así:

```
borrapantalla
repite 8
  [ balanceaderecha 45
    repite 4
      [ avanza 100 giraderecha 90 ] ]
```

16.4. Ejercicios

1. Dibuja un taburete cuadrado con una pata en cada esquina.

   Idea: repetir 4 veces "lado – pata".

2. A partir del taburete anterior, dibuja una silla con el respaldo inclinado.

3. Reforma la silla anterior para dibujar una tumbona.

4. Dibuja una pirámide cuadrangular regular.

   Idea: proceder como si fuera un taburete invertido, con las patas inclinadas 45 grados respecto del plano horizontal.

5. Dibuja un octaedro a partir de la pirámide anterior

16.5. El Visor 3D

xLogo incluye un visor 3D que permite visualizar los dibujos realizados en tres dimensiones. Este módulo usa las librerías de Java3D, por lo tanto es necesario tener instalado todo el Java3D.

16.5.1. Reglas

Las reglas a tener en cuenta para utilizar el Visor 3D son:

Al crear una figura geométrica sobre el Área de Dibujo, hay que indicar al Visor 3D qué formas desea grabar para una futura visualización. Es posible grabar polígonos (superficies), líneas, puntos o texto. Para utilizar esta función, las primitivas son:

- `empiezapoligono`, `definepoligono`: Los movimientos de la tortuga posteriores a esta llamada se guardan para crear un polígono.
- `finpoligono`: Desde la ejecución de `definepoligono`, la tortuga habrá pasado por varios vértices. Este polígono se habrá "registrado" y su color se definirá en función del color de todos sus vértices.

  Esta primitiva finaliza el polígono.
- `empiezalinea`, `definelinea`: Los movimientos de la tortuga posteriores a esta llamada se guardan para crear una banda (una línea).
- `finlinea`: Desde la ejecución de `definelinea`, la tortuga habrá pasado por varios vértices. Se guardará esta línea y su color se definirá en función del color de todos sus vértices.

  Esta primitiva finaliza la banda
- `empiezapunto`, `definepunto`: Los movimientos sigueintes de la tortuga se guardan para crear un conjunto de puntos.
- `finpunto`: Esta primitiva finaliza el conjunto de puntos.
- `empiezatexto`, `definetexto`: Cada vez que el usuario muestre un texto sobre el Área de Dibujo con la primitiva `rotula`, se almacenará y luego será representado por el visor 3D.
- `fintexto`: Esta primitiva finaliza la grabación de texto.
- `vista3d`, `vistapoligono` Inicia el visor 3D, todos los objetos guardados se dibujan en una nueva ventana.

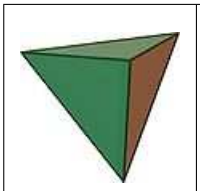
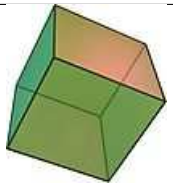
  Disponemos de controles para mover la "cámara" que muestra la escena:

  - Para hacer rotar la imagen haciendo *clic* con el botón izquierdo del ratón y arrastrando.

- Para desplazar la imagen haciendo *clic* con el botón derecho del ratón y arrastrando.
- Para hacer *zoom* sobre la escena, usaremos la rueda del ratón

16.5.2. Poliedros. Los sólidos platónicos

Los sólidos platónicos son el tetraedro, el cubo (o hexaedro regular), el octaedro, el dodecaedro y el icosaedro. También se conocen como cuerpos platónicos, cuerpos cósmicos, sólidos pitagóricos, sólidos perfectos, o **poliedros regulares convexos**. Son poliedros convexos cuyas caras son polígonos regulares iguales en cuyos vértices se unen el mismo número de caras, y reciben este nombre en honor al filósofo griego Platón (ca. 427 adC/428 adC – 347 adC).

				
Tetraedro	Hexaedro o cubo	Octaedro	Dodecaedro	Icosaedro

Esta lista es exhaustiva, ya que es imposible construir otro sólido diferente que cumpla todas las propiedades exigidas, es decir, convexidad y regularidad.

Para su construcción seguiremos un mecanismo general:

- Crearemos un procedimiento que trace el polígono que define sus caras entre las primitivas `empiezapoligono` y `finpoligono`.
- Comenzando desde una cara, iremos recorriendo los vértices dibujando las caras anexas.
- En caras dibujadas en el punto anterior, repetimos el proceso hasta concluir el poliedro.

En todo momento, la mayor dificultad será determinar la orientación adecuada, es decir, el ángulo diedro:.

Tetraedro	Hexaedro o cubo	Octaedro
$\varphi = \arccos\frac{1}{3} \simeq 70{,}52878°$	$90°$	$\varphi = \arccos\frac{-1}{\sqrt{3}} \simeq 109{,}47122°$

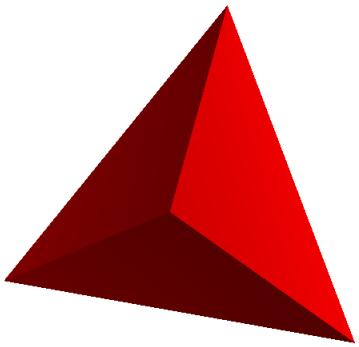
Dodecaedro	Icosaedro
$\varphi = \arccos\frac{-1}{\sqrt{5}} \simeq 116{,}56505$	$\varphi = \arccos\frac{-\sqrt{5}}{3} \simeq 138{,}189685°$

Dibujando un tetraedro

El poliedro más simple, una vez dibujada la base, basta recorrerla con el balanceo adecuado:

```
para triangulo :lado
   empiezapoligono
    repite 3 [ avanza :lado giraderecha 120 ]
   finpoligono
fin

para tetraedro :lado
  borrapantalla perspectiva
  poncolorlapiz rojo
  haz "angulo arcocoseno 1/3
# Base triangular
  triangulo :lado
# caras laterales
  repite 3
    [ balanceaizquierda :angulo
      triangulo :lado
      balanceaderecha :angulo avanza :lado giraderecha 120 ]
  vista3d
fin
```

Ejecutando: `tetraedro 500`:



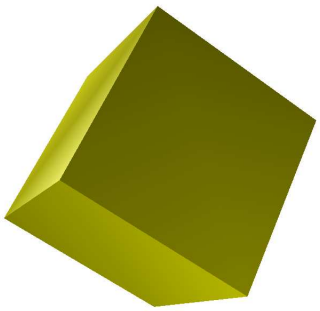
Dibujando un cubo

El caso más simple en lo referente al ángulo diedro es el cubo: son todos de 90 grados. A cambio, debemos recorrer más vértices.

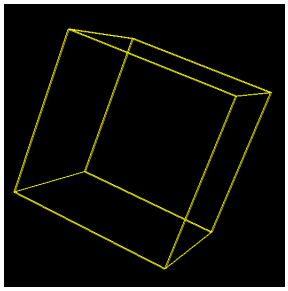
```
para cuadrado :lado
# Grabamos los vertices del cuadrado
```

```
   empiezapoligono
   repite 4 [ avanza :lado giraderecha 90 ]
   finpoligono
 fin

para cubosimple :lado
 # Cubo Amarillo
   borrapantalla perspectiva
   poncolorlapiz amarillo
 # Caras laterales
   repite 4
     [ cuadrado :lado subelapiz
       giraderecha 90 avanza :lado giraizquierda 90
       balanceaderecha 90 bajalapiz ]
 # Parte inferior
   bajanariz 90 cuadrado :lado subenariz 90
 # Cara Superior
   avanza :lado bajanariz 90 cuadrado :lado
 # Visualizacion
   vista3d
fin
```

Estamos listos para ejecutar el comando: `cubosimple 400`:



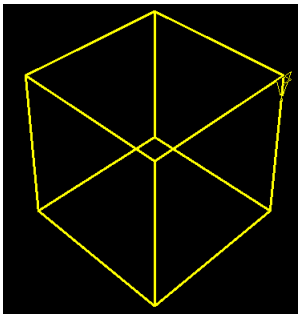
Al sustituir en el procedimiento `cuadrado`, `empiezapoligono` por `empiezalinea`, y `finpoligono` por `finlinea`:



Si hubiéramos usado `empiezapunto` y `finpunto` en lugar de `empiezalinea` y `finlinea`, deberíamos ver en la pantalla sólo los ocho vértices del cubo.

Estas primitivas son muy útiles para mostrar el conjunto de puntos en el espacio 3D.

En todos los casos, en el Área de Dibujo se muestran las aristas del cubo que luego se verá "macizo" con el Visor:

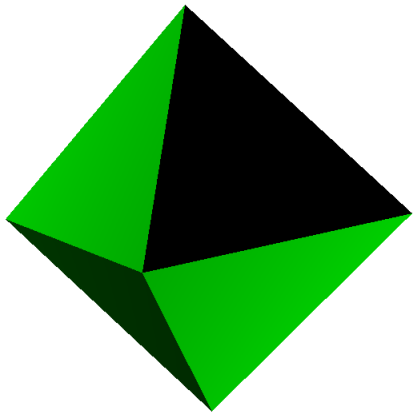


Dibujando un octaedro

Este poliedro tampoco presenta una dificultad excesiva. Recorremos el cuadrado central dos veces, una vez balanceados hacia la derecha y otra a la izquierda:

```
para triangulo :lado
   empiezapoligono
    repite 3 [ avanza :lado giraderecha 120 ]
   finpoligono
fin

para octaedro :lado
  borrapantalla perspectiva
  poncolorlapiz verde
  haz "angulo 0.5*arcocoseno (-1)/3
  repite 4
    [ balanceaizquierda :angulo
         triangulo :lado
      balanceaderecha 2*:angulo
         triangulo :lado
      balanceaizquierda :angulo
      avanza :lado giraderecha 90 ]
  vista3d
fin
```

El resultado:

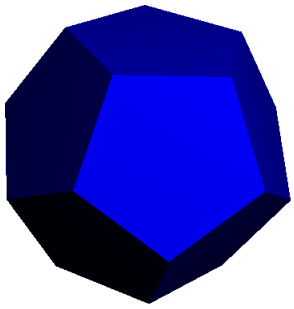


Dibujando un dodecaedro

Gracias a Juan Casal por "prestarnos" este programa. En él vemos como se introduce un procedimiento `cambiacara` para simplificar el código, ya que en este caso también se hace necesario modificar el cabeceo de la tortuga:

```
para pentagono :lado
  empiezapoligono
   repite 5
    [ avanza :lado giraderecha 72 ]
  avanza :lado
  finpoligono
fin

para cambiacara
  giraizquierda 18
  bajanariz 63.44
  giraizquierda 18
fin

para dodecaedro :lado
  limpia
  perspectiva
  poncolorlapiz azul
  pentagono :lado          #cara sup
  giraizquierda 180
  balanceaderecha 63.44    #pos ini p corona sup
  repite 5                   #corona sup
  [ pentagono :lado cambiacara ]
  subelapiz
```

```
     balanceaizquierda 63.44 giraderecha 180
    cambiacara
     avanza :lado giraderecha 72 avanza :lado
    cambiacara
     avanza :lado giraderecha 72
  bajalapiz
  repite 5
   [ pentagono :lado cambiacara ]
  balanceaizquierda 63.44
  giraizquierda 180
  retrocede :lado
  pentagono :lado
#
  vista3d
fin
```



Dibujando un icosaedro

Finalmente, y adaptando un programa de Tom Lynn, conseguimos el más complejo de los sólidos platónicos. Dibujaremos cinco bloques de cuatro triángulos, y nos ayudamos de dos procedimientos para movimientos básicos:

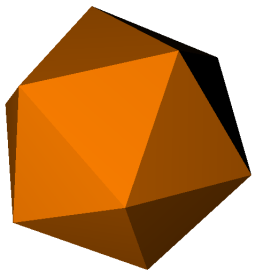
- `proxarista`, que desplaza a la tortuga hacia la siguiente arista en sentido horario sobre la cara actual
- `carasig`, que la desplaza hacia la cara adyacente por la derecha

  `repite 5 [carasig]` devuelve la tortuga a la posición original

```
para icosaedro :lado
  borrapantalla
  perspectiva
  poncolorlapiz naranja
# Inicializamos los angulos
```

```
  haz "phi (1 + raizcuadrada 5) / 2
  haz "beta arcoseno (:phi / raizcuadrada 3)
  haz "alpha 180-2*:beta
  repite 5 [cuatrotrian :lado]
  vistapoligono
fin

para triangulo :lado
  bajalapiz # los procedimientos auxiliares lo dejan arriba
  empiezapoligono
   repite 3 [avanza :lado giraderecha 120]
  finpoligono
  carasig :lado
fin

para cuatrotrian :lado
  repite 2 [triangulo :lado] proxarista :lado
  repite 2 [triangulo :lado]
  repite 2 [ repite 2 [carasig :lado] proxarista :lado]
  carasig :lado
fin

para proxarista :lado
  subelapiz avanza :lado giraderecha 120
fin

para carasig :lado
  subelapiz giraderecha 60 avanza :lado giraizquierda 120 balanceaderecha :alpha
fin
```

El resultado:

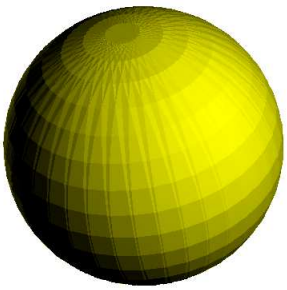


16.5.3. La esfera

El paso final es extender las `n` caras al caso *infinito*, es decir, cómo obtener una esfera. Evidentemente no podemos dibujar *infinitos* polígonos, así que vamos a aproximar la esfera

mediante cuadrados:

```
para esfera
  borrapantalla
  perspectiva
  poncolorlapiz amarillo
  repite 20
    [ retrocede 25 giraizquierda 90 avanza 25 giraderecha 90
      repite 36
        [ cuadrado 50 avanza 50 cabeceaarriba 10 ]
      giraizquierda 90 retrocede 25
      giraderecha 90 avanza 25
      giraderecha 9 ]
  vistapoligono
fin

para cuadrado :lado
  empiezapoligono
  repite 4 [ avanza :lado giraderecha 90 ]
  finpoligono
fin
```



Intenta crear una esfera usando la primitiva `circulo`. ¿Qué observas? ¿Por qué crees que es? Plantéate las mismas preguntas al ver el resultado cuando utilizas un procedimiento como el del `360-gono` que vimos en la sección 4.6.3 (pág. 36)

16.6. Ejercicios

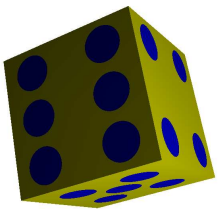
1. Crea un procedimiento `prisma_rect` que dibuje un prisma de base rectangular cuyas aristas midan los valores introducidos con tres variables `:a`, `:b` y `:c`.

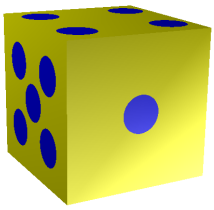
2. Diseña un procedimiento `prisma_reg` que dibuje un prisma cuya base sea un polígono regular y lea tres variables: `:n` (número de lados del polígono), `:l` (lado del polígono) y `:h` (altura).

   Generaliza el procedimiento anterior para obtener un cilindro cuya base tenga radio `r` y altura `h`.

   Observa como cambia el aspecto que tiene el cilindro (la calidad del mismo) a medida que aumenta el número de lados de la base.

3. Transforma el cubo creado en 16.5.2 en un dado.





   Recuerda que las caras opuestas de un dado suman siete.

4. Crea un procedimiento `piramide` que dibuje una pirámide de base rectangular, cuyos parámetros sean las variables `:a`, `:b` (las dimensiones del rectángulo de la base) y `:h` (la altura de la pirámide).

5. Modifica el procedimiento anterior para obtener una pirámide regular que acepte tres valores: `n`, número de lados de la base; `l`, lado de la base y `h`, altura.

   Generaliza el procedimiento anterior para obtener un cono de radio de base `r` y altura `h`.

   Observa como cambia el aspecto que tiene el cono (la calidad del mismo) a medida que aumenta el número de lados de la base.

6. Combina los procedimientos creados antes para obtener:

   a) Una casa en 3D combinando un prisma y una pirámide.

   b) Un silo en 3D combinando un cilindro y un cono.

7. Modifica los procedimientos `cono` y `piramide` para obtener un cono truncado y una pirámide truncada, respectivamente. Ten en cuenta que debes introducir nuevas variables.

8. Combina los procedimientos creados en el ejercicio anterior para crear una casa de base rectangular cuyo tejado termine con uan pirámide truncada.

9. Combina los procedimientos `cono`, `cilindro` y `esfera` para dibujar un "árbol" por superposición de:

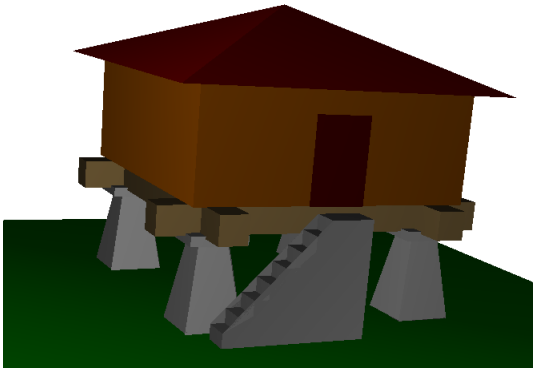
   a) Un cono "pegado" al suelo.

   b) Un cilindro de radio algo menor como tronco.

   c) Una esfera como copa del árbol.

10. Ya sabes crear "casas" y "árboles". ¿Te atreves a crear un pequeño pueblo en 3-D? Puedes hacer que las calles sean perpendiculares entre sí, y que las dimensiones de los edificios sean aleatorias, usando `azar`, para darle más realismo.

    Puedes incluir la versión en 3-D del hórreo asturiano (página 129):



    cuyo código, así como otros ejemplos, puede encontrarse en:

    `http://xlogo.tuxfamily.org/sp/html/ejemplos/3D.html`

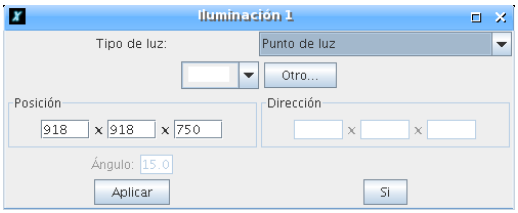
16.7. Efectos de luz y niebla

Desde la versión 0.9.93 se pueden añadir efectos artísticos a las imágenes generadas en el Visor. Estos pueden ser efectos de luz y de niebla, y se accede a ellos con los botones presentes en el visor 3D.

Efectos de luz

Se pueden utilizar cuatro tipos de luz en las imágenes en tres dimensiones, a las que se accede haciendo *clic* en uno de los cuatro botones mostrados bajo la leyenda `Iluminación`.

Al trazar por primera vez una imagen en 3D sólo se utilizan dos tipos de luz, ambos **Luz Puntual**, pero pulsando en cualquiera de los cuatro botones de `Iluminación`, aparece el siguiente cuadro de diálogo:



donde podemos elegir entre los siguientes tipos de luz:

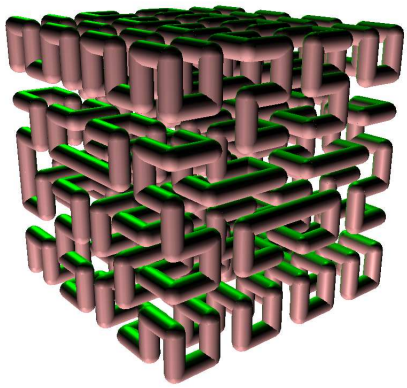
Luz Ambiente: Luz uniforme de la que sólo puede modificarse el color

Luz Direccional: Se genera respecto a una dirección fija. Se parece a la luz ambiente cuando la fuente está muy lejos del objeto (por ejemplo, el sol)

Punto de Luz: La fuente está en una posición determinada, como en el caso de un faro.

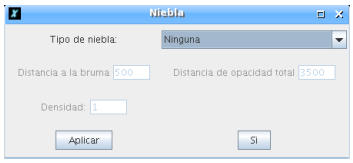
Foco: Es como el punto de luz, pero el haz de luz se abre formando un cono cuya abertura debe fijarse.

La mejor forma de entenderlo, es practicar con ello.



Efectos de niebla

Se pueden añadir efectos de niebla en la imagen tridimensional. Pulsa el botón con "nubes" y obtendrás este cuadro de diálogo:



Disponemos de dos tipos de niebla:

Niebla Lineal o progresiva: La imagen se va difuminando de modo lineal, pudiendo variar dos parámetros:

- La distancia a la que empieza la niebla
- La distancia a la que la niebla no deja ver nada (opacidad total)

Niebla Densa: La niebla es uniforme en toda la escena, y sólo necesitamos especificar la densidad de la misma.

Este es un ejemplo con niebla lineal: