1 VPython

VPython es un módulo sobre Python. Consta de una serie de subrutinas que permiten construir y visualizar en tiempo real algunos objetos geométricos sencillos.

Python (y por consiguiente VPython) es un lenguaje interpretado. Un programa en Python es un documento de texto que se ejecuta con Python. Para escribir ese programa se puede usar cualquier editor de textos. Normalmente se le pone extensión .py aunque no es necesario.

VPython incorpora un editor llamado IDLE que tiene algunas ventajas. Una de ellas es que lleva incorporadas opciones tales como indentar o desindentar un párrafo, convertir un párrafo en líneas de comentario, etc. que son útiles a la hora de programar. Además, desde el mismo IDLE puede ejecutarse el programa.

Para ejcutar un programa en VPython tenemos tres posibilidades:

1.- Abrir IDLE y en él se carga el programa. Para ejecutarlo se pulsa F5.

2.- Si el programa lleva la extensión .py, con el botón derecho permite abrirlo directamente con IDLE.

3.- Si la extensión es .py al pulsar dos veces sobre el programa se ejecuta directamente.

Hay algunas diferencias entre estas tres formas. La diferencia principal entre las dos primeras y la tercera es el entorno. Si abres IDLE y cargas un programa, al ejecutarlo se abren automáticamente dos ventanas, una de gráficos y un shell. Este shell permite ver los mensajes de error o introducir datos que te pida el programa. Si usamos la tercera opción y pedimos directamente el programa, aparece una ventana del DOS que cumple el papel del shell pero con una diferencia importante: si hay un error se cierra automáticamente, con lo que no llegas a verlo.

La diferencia entre las dos primeras formas de abrir un programa es la dependencia de IDLE:

- Si abres IDLE y cargas un programa, cuando lo cierres IDLE permanecerá abierto.

- Si lo haces al revés, IDLE se cierra automáticamente al cerrar el programa.

Una observación importante es IDLE siempre guarda el programa antes de ejecutarlo. Por tanto, si hacemos modificaciones de las que no estemos seguros, es conveniente grabarlo con otro nombre antes de ejecutarlo.

Al empezar cualquier programa lo primero que hay que decirle a Python es que cargue las rutinas de VPython. Esto se hace con la siguiente línea

```
from visual import *
```

donde * indica que se carguen todas las rutinas del módulo VPython.

Si necesitamos rutinas de otras librerias, necesitaremos también importarlas. Usaremos más adelante las librerías random y LinearAlgebra. La primera implementa números a suertes, números combinatorios, etc. y la segunda manejo de matrices.

2 La ventana de gráficos

Como hemos comentado, al empezar un programa de visual, creamos dos ventanas, una de gráficos y un shell. Por defecto la ventana de gráficos recibe el nombre de scene.

Si queremos crear otra pantalla de gráficos se hace con la orden

ventana = display()

donde ventana es el nombre que usamos para referirnos a esa pantalla.

El manejo básico de una pantalla gráfica se hace con el ratón.

- La tecla izquierda no hace nada
- La tecla derecha permite girar el punto de vista de la escena
- Pulsando las dos teclas y arrastrando el ratón podemos hacer zoom.

Casi todos los aspectos de la ventana gráfica pueden cambiarse mediante código. Veamos algunos de estos comandos:

2.1 scene.background

sirve para indicar el color de fondo de la ventana scene.

Por ejemplo

scene.background = color.red

cambia a rojo el color del fondo. Más adelante veremos cómo especificar un color en VPython.

2.2 scene.foreground

Indica el color por defecto en el que VPython va a dibujar los objetos que creemos

2.3 scene.fullscreen

Si ponemos scene.fullscreen = 1 la ventana de gráficos funciona en pantalla completa. Por defecto se tiene scene.fullscreen = 0.

2.4 scene.autoscale

Por defecto VPython escala los objetos para que quepan todos en pantalla. Si no queremos que lo haga debemos poner scene.autoscale = 0

2.5 Iluminación

Al empezar el programa, la escena se visualiza con una mezcla entre luz ambiente y dos focos de luz, pero nosotros podemos indicar cuánta luz ambiente queremos y qué fuentes de luz y con qué intesidad queremos colocar. Para ello se usa:

scene.ambient nos da el valor de luz ambiente. Este valor debe estar entre 0 y 1

scene.lights es una lista de vectores que nos indican desde qué direcciones proceden las fuentes de luz. La magnitud de cada luz es la longitud del vector correspondiente.

La suma total de todas las luces del escenario debe de ser siempre ≤ 1

2.6 Colocando la cámara

Para indicar dónde ponemos una cámara usaremos tres órdenes:

scene.center es el punto hacia el que está mirando la cámara. Por defecto es scene.center = (0,0,0). Este es también el punto alrededor del que gira la cámara cuando la movemos con el ratón.

scene.forward es la dirección a la que apunta la cámara. Por tanto, podemos pensar que la cámara está colocada en la posición scene.center - scene.forward

scene.range nos dice la sección de espacio que abarca la cámara. Al empezar VPython su valor es scene.range = (10,10,10) pero se modifica inmediatamente para que quepan todos los objetos a no ser que pongamos scene.autoscale=0.

Hay más ódenes que permiten indicar el título, tamaño o posición de la ventana.

Veamos a continuación los tipos de datos que vamos a usar en VPython:

3 Números

VPython trabaja por defecto con números enteros y reales. La distinción es muy importante cuando vayamos a dividir, porque si los números son enteros la división nos da como resultado el cociente de la división entera, descartando el resto.

Así, por ejemplo, $9/2 = 4$.

Si un número tiene decimales como 3.7 el programa lo toma directamente como un número real. Si no tiene decimales, para indicar que un número es real debe de ponerse un punto al final, es decir

x = 7 le da a la variable x el valor entero 7

x = 7. le da a la variable x el valor real 7

4 Vectores

Un vector se define con la orden vector. Por ejemplo, la siguiente línea define u como el vector (1,2,1)

u = vector(1,2,1)

Es muy importante no olvidarnos de poner vector, ya que si escribimos

u = (1,2,1)

Vpython pensará que nos referimos a una lista. Cuando queramos operar con él (sumar, multiplicarlo por un número, etc.) nos dará un error de tipo.

Las instrucciones para trabajar con vectores son:

+ permite sumar vectores

* permite multiplicar un número por un vector

mag(u) devuelve la longitud del vector u

dot(u,v) calcula el producto escalar de u y v

cross(u,v) para hacer el producto vectorial de u y v

norm(u) calcula $\frac{u}{\|u\|}$

5 Listas

Una lista de objetos se declara encerrando dichos objetos entre corchetes.

Por ejemplo:

L = [(1,1,1), (2,1,1), (1,0,1)]

Es una lista que contiene tres puntos.

No es necesario que todos los objetos de una lista sean del mismo tipo, así la lista L = [(1,1,1), 2] tiene dos elementos de los cuales el primero es un punto y el segundo un número.

Con una lista se pueden realizar las siguientes operaciones:

- len nos devuelve la longitud de la lista. En nuestro ejemplo tendríamos que len(L) = 2

- Para escoger un elemento de la lista lo haremos con el nombre de la lista y el número del elemento que queremos entre corchetes. Por ejemplo, en la lista L = [(1,1,1), 2] tendríamos L[0] = (1,1,1) y L[1] = 2. Observemos que VPython empieza a numerar los objetos de una lista empezando en 0.

- La suma de dos listas consiste en hacer una nueva lista poniendo primero los elementos de la primera lista y después los de la segunda.

6 Programando

6.1 Líneas de comentario

El carácter # sirve para indicar que una línea es de comentario.

6.2 Condiciones

Los comandos para hacer condicioneles son if, elif, else

Hay que tener en cuenta que, para todos los comandos de control, es fundamental la indentación del párrafo. Cualquier comando de control termina en : y afecta a las lineas de programa que aparezcan hasta que acabe la indentación.

Así, por ejemplo:

```
if a > 0:
    a = a+1
    b = 2*a
else:
    a = 0
c=1
```

la primera línea es una sentencia de control que afecta a las dos siguientes, es decir, el bloque del if comprende todas las filas que estén indentandas más a la derecha que esa sentencia de control.

Los comandos para hacer condicioneles son if, elif, else

6.3 Bucles

Para hacer un bucle se usa el comando while.

Por ejemplo, el siguiente bucle suma los primero diez números naturales:

```
suma = 0
i = 1
while i<=10:
    suma = suma + i
    i = i+1
```

También podemos hacer un bucle sobre los elementos de una lista. Esto se hace con el comando for in .

Supongamos que tenemos una lista de números L=[3, 2, 1, 4] y queremos sumarlos. Podríamos hacerlos con el siguiente programa:

```
suma = 0
for i in L:
    suma = suma +i
```

Observemos que for no tiene el sentido usual, sino que sólo puede usarse sobre una lista.

6.4 Introducir datos

Para mostrar un mensaje en el shell se usa la orden print

Para pedir al usuario que inbtroduzca un dato se usa raw_input(). Se puede incluir un mensaje como se ve en el siguiente ejemplo:

a = raw_input("Introduce un número : ")

cuando ejecutamos raw_input el dato introducido se guarda siempre como una cadena. Para convertirlo en un número hay que usar el comando eval

Así, el siguiente programa pide al usuario dos números y los suma:

```
a = raw_input("Introduce el primer número : ")
b = raw_input("Introduce el segundo número : ")
c = eval(a) + eval(b)
print c
```

6.5 Subrutinas

Para definir una subrutina se usa la siguiente orden

def subr(parámetros):

dende subr es el nombre de la subrutina y parámetros es una lista de datos que puede quedar vacía.

Si queremos que una subrutina nos devuelva algún dato lo indicaremos con return.

Por ejemplo, la siguiente subrutina recibe como parámetros tres números y devuelve su suma:

```
def suma(a,b,c):
    suma = a + b + c
    return suma
```

Para llamar a la subrutina podemos hacerlo con el nombre de la misma y, entre paréntesis, los datos. Por ejemplo:

s = suma(3,2,5)

asigna a s el valor de la suma de 3, 2 y 5 calculada por la subrutina suma.

Para más referncias sobre programación en python se puede consultar la siguiente dirección:

www.cmat.edu.uy/~walterm/python-2002/pynotas.pdf

7 Color

Para indicar un color en VPython podemos hacerlo de dos formas. La primera es escoger entre los colores predefinidos que son:

color.red
color.green
color.blue
color.cyan
color.black
color.white

color.yellow

color.orange

color.magenta

Para indicar otro color, se hace con su valor RGB. Una herramienta útil para hacer esto es el programa c:\Python2.4\lib\site-packages\visual\examples\colorsliders.py

Para asignar un color a un objeto podemos hacerlo:

- Dentro del paréntesis en la declaración del objeto. Por ejemplo en sphere(color = color.red)

- Si el objeto tiene nombre, poniendo nombre.color = color. Por ejemplo, si hemos creado una esfera con el nombre bola, la orden bola.color = color.red le asigna color rojo.

8 Objetos

Un programa en VPython consiste en indicar unos objetos que queremos construir, cada uno de los cuales tienen unas ciertas características.

Hay fundamentalmente de dos tipos, por un lado tenemos órdenes que dibujan unos objetos predefinidos de los que solamente hay que indicar unas ciertas características. Son:

- cylinder
- arrow
- cone
- pyramid
- sphere
- ring
- box
- ellipsoid

En segundo lugar hay otros objetos que permiten hacer construcciones más complejas:

- curve
- convex
- label
- frame

Para invocar uno de ellos, basta poner la orden y, entre paréntesis, sus atributos.

En principio un objeto no necesita tener un nombre. Solamente le pondremos uno si posteriormente queremos hacer referencia a ellos.

Así, por ejemplo,

sphere(pos = (1,1,1), radius = .4, color = color.red)

crea una esfera con centro (1,1,1), radio .4 y color rojo. Sin embargo no hay forma de referirse posteriormente a ella para modificarla. Para poder hacer esto escribiríamos:

bola = sphere(pos = (1,1,1), radius = .4, color = color.red)

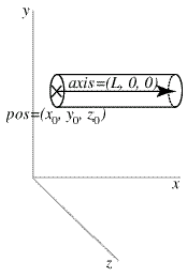
que no solamente crea la esfera sino que le asigna el nombre bola.

9 Atributos de objetos

Hay dos atributos que pueden usarse con cualquier objeto que son el color, que lo aceptan todos los objetos menos frame, y el atributo visible que indica si el objeto se dibuja o no.

Aparte de ellos, cada objeto admite unos atributos específicos:

9.1 cylinder



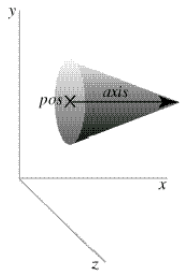
Sus atributos son:

pos el centro del círculo de la base

radius el radio de la base

axis el vector altura, es decir, el vector que une los centros de las dos bases. Vpython permite definir cilindros oblicuos, por lo que axis no es un número, sino que tiene tres coordenadas.

9.2 cone



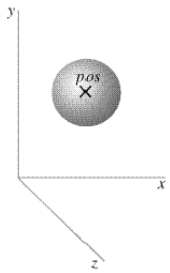
Sus atributos son:

pos el centro del círculo de la base

radius el radio de la base

axis el vector altura, es decir, el vector que une los centros de la bases y el pico. Vpython permite definir conos oblicuos, por lo que axis no es un número, sino que tiene tres coordenadas.

9.3 sphere

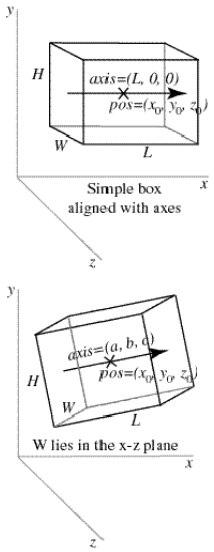


Sus atributos son:

pos el centro del círculo de la base

radius el radio de la base

9.4 box



Crea una "caja". Sus atributos son:

pos el centro de la caja

axis es un vector que nos dice sobre qué "eje X" se coloca la caja.

up la dirección para poner la "parte de arriba" de la caja

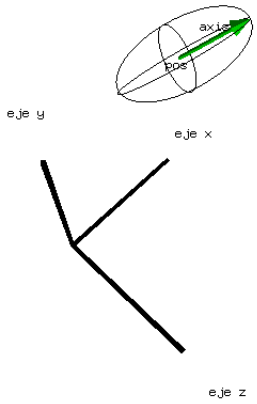
length tamaño "en el eje X"

width tamaño "en el eje Y"

height tamaño "en el eje Z"

Podemos indicar directamente el tamaño de box con size que es un vector de la forma size = (length, width, height)

9.5 ellipsoid



Crea un elipsoide. Sus atributos son:

pos el centro del elipsoide

axis es un vector que nos dice sobre qué "eje X" se coloca el elipsoide.

up la dirección para poner la "parte de arriba" del elipsoide

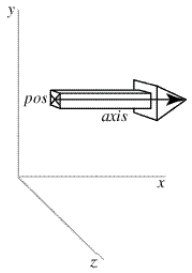
length tamaño "en el eje X"

width tamaño "en el eje Y"

height tamaño "en el eje Z"

Podemos indicar directamente el tamaño del elipsoide con size que es un vector de la forma size = (length, width, height)

9.6 arrow



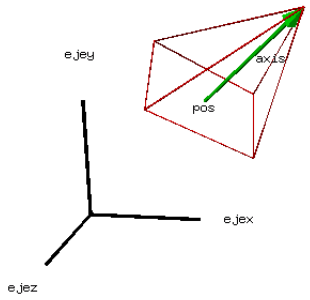
Sus atributos son:

pos el centro de la base de la flecha

axis el vector, es decir, el vector que une el centro de la base con el final de la flecha.

shaftwidth el ancho del "tronco" de la flecha
headwidth el ancho de la punta de la flecha
headlength la longitud de la punta de la flecha

9.7 pyramid



Crea una pirámide de base rectangular. Sus atributos son:
pos es el centro de la base de la pirámide
axis el vector, es decir, el vector que une el centro de la base con el pico de la pirámide.
up la dirección para poner el primer vértice de la pirámide
length tamaño "en el eje X"
width tamaño "en el eje Y"
height tamaño "en el eje Z"
Podemos indicar directamente el tamaño de la pirámide con size que es un vector de la forma size = (length, width, height)

9.8 convex

Dada una lista de puntos, crea un convexo cuyos vértices son los puntos de la lista. Vpython borra de la lista aquellos vértices que no permitan que el objeto sea convexo.
Sus atributos son:
pos una lista de vértices
Es el objeto que más usaremos al principio porque es muy flexible.

9.9 curve

Crea una "curva" a base de unir mediante segmentos una lista de puntos. Sus atributos son:
pos una lista de vértices
radius el grosor de la curva

Caso de poner solamente dos vértices, sirve para dibujar segmentos.

9.10 frame

Realmente no es un objeto, sino que permite agrupar varios objetos en uno solo. Por ejemplo, si escribimos:

arbol = frame()

cylinder(pos = (0,0,0), radius = .3, axis = (1,2,1), frame = arbol)

sphere(pos = (1,2,1), radius = 1, color = color.green, frame = arbol)

estamos creando un objeto llamado arbol que contiene tanto al cilindro como a la esfera. Si ahora ponemos

arbol.pos = (1,1,1)

estaremos colocando en esa posición el "dibujo" formado por el cilindro y la esfera.

Sus atributos son:

pos el "punto" donde poner el objeto

axis el "eje" sobre el que poner el objeto

up la dirección en que el objeto mira hacia arriba

objects es una lista de los objetos que contiene el frame

9.11 faces

Es el objeto más complejo que vamos a usar. Consiste en una lista de triángulos para dibujar y lo usaremos cuando lleguemos a superficies. Sus atributos son:

pos una lista de triángulos

normal una lista de vectores normales, uno para cada vértice de cada triángulo, para determinar la iluminación

color una lista de colores, uno para cada vértice de cada triángulo, para determinar su coloración

La longitud de las listas pos y normal debe de ser la misma. Respecto a acolor hay dos opciones, poner un único color que se usará para todos los triángulos que se generen o especificar una lista de colores de la misma longitud que pos.

9.12 Otros objetos

Hay más objetos, tipo helix, ring o label que, en principio, no vamos a necesitar.

Así mismo, es posible definirnos nuestros propios objetos y ampliar el programa.