

VPython : visualisation 3D pour le commun des mortels

Laurent Pierron

INRIA-Lorraine
LORIA

Rencontres Mondiales du Logiciel Libre 2005

Plan

- 1 Introduction
 - Présentation de VPython
 - Premiers objets
 - Premières animations
- 2 Modélisation 3D
 - La scène
 - Les objets
 - Graphiques
- 3 Animation et interaction
 - Contrôle de l'animation
 - Les évènements
 - Interfaces graphiques

Plan

- 1 Introduction
 - Présentation de VPython
 - Premiers objets
 - Premières animations
- 2 Modélisation 3D
 - La scène
 - Les objets
 - Graphiques
- 3 Animation et interaction
 - Contrôle de l'animation
 - Les évènements
 - Interfaces graphiques

Plan

- 1 Introduction
 - Présentation de VPython
 - Premiers objets
 - Premières animations
- 2 Modélisation 3D
 - La scène
 - Les objets
 - Graphiques
- 3 Animation et interaction
 - Contrôle de l'animation
 - Les évènements
 - Interfaces graphiques

VPython : un module 3D pour Python

Modélisation de systèmes physiques

VPython est un environnement logiciel comprenant :

- Le langage de programmation **Python**
- L'environnement de développement **IDLE**
- **Visual**, un module de visualisation 3D temps réel
 - déplacement dans l'espace 3D
 - zoom avant et arrière
 - animation continue
- **Numeric** et **numarray**, des modules de calcul vectoriel rapide

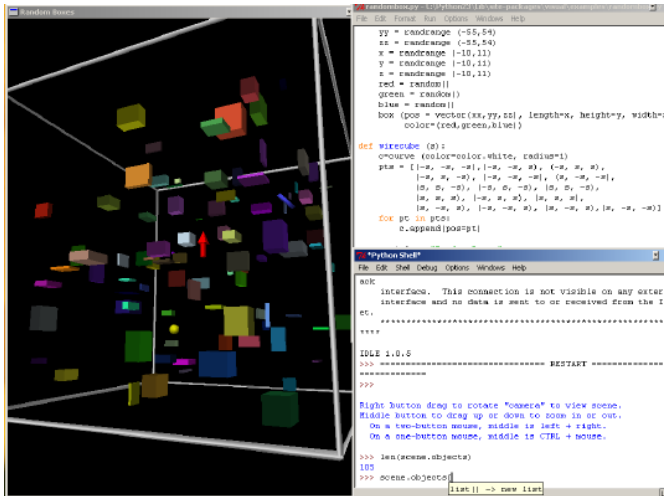
VPython a été développé à Carnegie-Mellon University par David Scherrer en l'an 2000.

VPython : un logiciel libre

Le projet **VPython** est sous copyright de David Scherrer, mais il peut être librement copié, modifié et distribué.

- **VPython** est écrit en C++, le projet est hébergé par SourceForge
- **VPython** est basé sur la bibliothèque graphique 3D OpenGL
- **VPython** fonctionne sous XWindows (X11) sous tout Unix (Linux, BSD, Mac OS X)
- **VPython** fonctionne également sous Windows
- Ancienne version sous Mac OS 9

Environnement VPython



The image shows a screenshot of the VPython environment. On the left, a window titled "Random Boxes" displays a 3D scene with a black background and a white wireframe box. Inside the box, numerous small, multi-colored rectangular boxes are scattered randomly. On the right, a Python Shell window shows the following code:

```
yy = randrange (-55,54)
xx = randrange (-55,54)
x = randrange (-10,11)
y = randrange (-10,11)
z = randrange (-10,11)
red = random()
green = random()
blue = random()
box (pos = vector(xx,yy,zz), length=x, height=y, width=
color=(red,green,blue))

def wirecube (s):
o=curve (color=color.white, radius=1)
pts = [ [-s, -s, -s], [-s, -s, s], (-s, s, s),
[-s, s, -s], [-s, -s, -s], (s, -s, -s),
[s, s, -s], [-s, s, -s], [s, s, s],
[s, -s, s], [-s, -s, s], [s, -s, s], [s, -s, -s]]
for pt in pts:
o.append(pos=pt)
```

Below the code, the Python Shell window shows the following output:

```
ack
interface. This connection is not visible on any other
interface and no data is sent to or received from the I
ct.
*****
****

IDLE 1.0.5
>>> ***** RESTART *****
>>>
>>>
Right button drag to rotate "camera" to view scene.
Middle button to drag up or down to zoom in or out.
On a two-button mouse, middle is left + right.
On a one-button mouse, middle is CTRL + mouse.

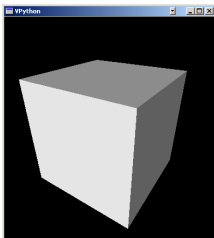
>>> len(scene.objects)
105
>>> scene.objects
```

The bottom of the Python Shell window shows a list of objects, with the first element highlighted as "list | -> new list".

Plan

- 1 Introduction
 - Présentation de VPython
 - **Premiers objets**
 - Premières animations
- 2 Modélisation 3D
 - La scène
 - Les objets
 - Graphiques
- 3 Animation et interaction
 - Contrôle de l'animation
 - Les évènements
 - Interfaces graphiques

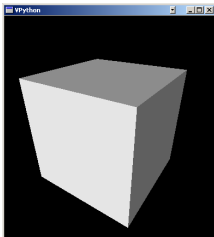
Premier programme : dessine-moi une boîte



Premier programme

```
from visual import *  
  
box()
```

Premier programme : dessine-moi une boîte



Premier programme

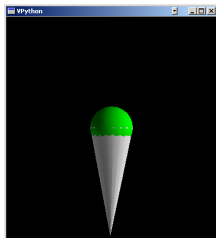
```
from visual import *  
  
box()
```

Second programme : une glace

dessine-moi une glace

```
from visual import *  
  
cornet=cone(length=-5)  
glace=sphere(color=color.green)  
  
cornet.color=color.yellow  
glace.radius=1.1
```

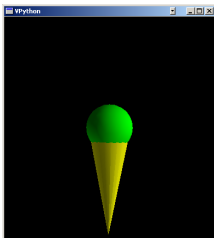
Second programme : une glace



dessine-moi une glace

```
from visual import *  
  
cornet=cone(length=-5)  
glace=sphere(color=color.green)  
  
cornet.color=color.yellow  
glace.radius=1.1
```

Second programme : une glace



dessine-moi une glace

```
from visual import *  
  
cornet=cone(length=-5)  
glace=sphere(color=color.green)  
  
cornet.color=color.yellow  
glace.radius=1.1
```

Plan

- 1 Introduction
 - Présentation de VPython
 - Premiers objets
 - **Premières animations**
- 2 Modélisation 3D
 - La scène
 - Les objets
 - Graphiques
- 3 Animation et interaction
 - Contrôle de l'animation
 - Les évènements
 - Interfaces graphiques

Première animation

Principe

- 1 Modifier les attributs d'un objet graphique
- 2 **VPython** met à jour la scène automatiquement

Fusion de la glace

```
from visual import *

cornet=cone(length=-5)
glace=sphere(color=color.green)

cornet.color=color.yellow
glace.radius=1.1

for i in range(100):
    rate(10)
    glace.radius *= 0.99
```

Première animation

Principe

- 1 Modifier les attributs d'un objet graphique
- 2 **VPython** met à jour la scène automatiquement

Fusion de la glace

```
from visual import *  
  
cornet=cone(length=-5)  
glace=sphere(color=color.green)  
  
cornet.color=color.yellow  
glace.radius=1.1  
  
for i in range(100):  
    rate(10)  
    glace.radius *= 0.99
```

Première animation

Principe

- 1 Modifier les attributs d'un objet graphique
- 2 **VPython** met à jour la scène automatiquement

Fusion de la glace

```
from visual import *  
  
cornet=cone(length=-5)  
glace=sphere(color=color.green)  
  
cornet.color=color.yellow  
glace.radius=1.1  
  
for i in range(100):  
    rate(10)  
    glace.radius *= 0.99
```

Première animation

Principe

- 1 Modifier les attributs d'un objet graphique
- 2 **VPython** met à jour la scène automatiquement

Fusion de la glace

```
from visual import *  
  
cornet=cone(length=-5)  
glace=sphere(color=color.green)  
  
cornet.color=color.yellow  
glace.radius=1.1  
  
for i in range(100):  
    rate(10)  
    glace.radius *= 0.99
```

Plan

- 1 Introduction
 - Présentation de VPython
 - Premiers objets
 - Premières animations
- 2 **Modélisation 3D**
 - La scène
 - Les objets
 - Graphiques
- 3 Animation et interaction
 - Contrôle de l'animation
 - Les évènements
 - Interfaces graphiques

Plan

- 1 Introduction
 - Présentation de VPython
 - Premiers objets
 - Premières animations
- 2 **Modélisation 3D**
 - **La scène**
 - Les objets
 - Graphiques
- 3 Animation et interaction
 - Contrôle de l'animation
 - Les évènements
 - Interfaces graphiques

Navigation dans la scène

Système de coordonnées

- Axe X de la gauche vers la droite
- Axe Y de haut en bas
- Axe Z du fond vers l'avant

Point d'observation

- En avant de la scène, modifiable à la souris.
- Approche, éloignement ou rotation :
 - 1 Autour de l'axe Y en déplaçant la souris de gauche à droite
 - 2 Autour de l'axe X en déplaçant la souris d'avant en arrière

Propriétés de la scène

Propriétés initiales

- la scène est un objet de nom **scene**
- centrée sur le point zéro
- point d'observation permet de voir tout les objets.
- modification des attributs de **scene**

quelques attributs intéressants

center point au centre de la fenêtre
background couleur du fond de la scène
foreground couleur par défaut des objets
fullscreen si 1 la scène remplit l'écran

Autres éléments de la scène

Propriétés avancées

autocenter,autoscale centre et zoome automatiquement

objects liste des objets visibles sur la scène

lights des sources lumineuses d'intensité variable peuvent être disposées n'importe où, par défaut deux sources

stereo deux images sont affichées pour permettre une vision stéréoscopique avec des lunettes spéciales

scènes multiples

- création de plusieurs scènes avec la fonction `display ()`
- définition de la taille de la vue (`width,height`)
- positionnement de la vue sur l'écran (`x,y`)

Plan

- 1 Introduction
 - Présentation de VPython
 - Premiers objets
 - Premières animations
- 2 **Modélisation 3D**
 - La scène
 - **Les objets**
 - Graphiques
- 3 Animation et interaction
 - Contrôle de l'animation
 - Les évènements
 - Interfaces graphiques

Objets graphique de base

Curvilignes

<code>cylinder</code>	cylindre
<code>sphere</code>	sphère
<code>cone</code>	cône
<code>ellipsoid</code>	ellipsoïde, cigare ou pilule
<code>ring</code>	anneau ou tore à section circulaire

Rectilignes

<code>box</code>	parallélépipède rectangle
<code>pyramid</code>	pyramide à section rectangulaire
<code>arrow</code>	flèche à section carrée

Attributs des objets graphiques

vecteurs

pos position
axis direction
color couleur
sous forme
(R,G,B)
up face avant

dimensions simples

x,y,z positions sur chaque axe
radius rayon
width,length,height dimensions
red,green,blue couleurs individuelles
display,frame scène ou objet de
placement
visible objet présent

Objets complexes

Multi-points

- helix** spires ou ressort
- curve** liste de points connectés
- convex** surface à partir de points

Composites

- frame** composition d'objets
- faces** surface définie par des triangles
- label** étiquette de texte associée aux objets

Plan

- 1 Introduction
 - Présentation de VPython
 - Premiers objets
 - Premières animations
- 2 **Modélisation 3D**
 - La scène
 - Les objets
 - **Graphiques**
- 3 Animation et interaction
 - Contrôle de l'animation
 - Les évènements
 - Interfaces graphiques

Graphiques 2D

Les graphiques 2D se font dans un environnement spécial défini par la fonction `gdisplay()`.

types de courbes

`gcurve` Points connectés

`gdots` Points non connectés

`gvbars` Barres verticales

`ghbars` Barres horizontales

`ghistogram` affichage de données après regroupement par quantile sous forme de barres verticales

Graphiques 2D dans un espace 3D

Utilisation de **curve**

quelques exemples

spirale

```
c = curve( x = arange(0,20,0.1) )
```

```
c.y = sin( 2.0*c.x )
```

```
c.z = cos( 2.0*c.x )
```

```
curve(x=range(100), y=range(100)**0.5,  
      color=color.red)
```

```
eqn = raw_input('Equation_en_x:_')
```

```
x = arange( 0, 10, 0.1 )
```

```
curve(x=x, y=eval(eqn))
```

Graphiques 2D dans un espace 3D

Utilisation de **curve**

quelques exemples

spirale

```
c = curve( x = arange(0,20,0.1) )
```

```
c.y = sin( 2.0*c.x )
```

```
c.z = cos( 2.0*c.x )
```

```
curve(x=range(100), y=range(100)**0.5,  
      color=color.red)
```

```
eqn = raw_input('Equation_en_x:_')
```

```
x = arange( 0, 10, 0.1 )
```

```
curve(x=x, y=eval(eqn))
```

Graphiques 2D dans un espace 3D

Utilisation de **curve**

quelques exemples

spirale

```
c = curve( x = arange(0,20,0.1) )
```

```
c.y = sin( 2.0*c.x )
```

```
c.z = cos( 2.0*c.x )
```

```
curve(x=range(100), y=range(100)**0.5,  
      color=color.red)
```

```
eqn = raw_input( 'Equation_en_x:_:' )
```

```
x = arange( 0, 10, 0.1 )
```

```
curve(x=x, y=eval(eqn))
```

Plan

- 1 Introduction
 - Présentation de VPython
 - Premiers objets
 - Premières animations
- 2 Modélisation 3D
 - La scène
 - Les objets
 - Graphiques
- 3 **Animation et interaction**
 - **Contrôle de l'animation**
 - **Les évènements**
 - **Interfaces graphiques**

Exemple d'animation

L'animation se définit en modifiant les attributs des objets

```
from visual import *
```

```
floor = box(pos=(0,0,0), length=4, height=0.5, width=4, color=co
```

```
ball = sphere(pos=(0,4,0), radius=1, color=color.red)
```

```
ball.velocity = vector(0,-1,0)
```

```
dt = 0.01
```

```
while 1:
```

```
    rate (100)
```

```
    ball.pos = ball.pos + ball.velocity*dt
```

```
    if ball.y < ball.radius:
```

```
        ball.velocity.y = -ball.velocity.y
```

```
    else:
```

```
        ball.velocity.y = ball.velocity.y - 9.8*dt
```

Plan

- 1 Introduction
 - Présentation de VPython
 - Premiers objets
 - Premières animations
- 2 Modélisation 3D
 - La scène
 - Les objets
 - Graphiques
- 3 **Animation et interaction**
 - **Contrôle de l'animation**
 - Les évènements
 - Interfaces graphiques

Contrôle de l'animation

fonction `rate(frequence)`

- appelée dans la boucle d'affichage
- garanti qu'il n'y aura pas plus d'exécution par secondes de la boucle que la fréquence indiquée.
- réalisation de temps réel, soit un mobile se déplaçant à la vitesse v , alors la position du mobile devra être modifiée de v/f où f est la fréquence.

rotation `objet.rotation(angle,axis,origin)`

`angle` angle de rotation

`axis` axe de rotation, par défaut axe de l'objet

`origin` origine de l'axe, par défaut centre de l'objet

Plan

- 1 Introduction
 - Présentation de VPython
 - Premiers objets
 - Premières animations
- 2 Modélisation 3D
 - La scène
 - Les objets
 - Graphiques
- 3 **Animation et interaction**
 - Contrôle de l'animation
 - **Les évènements**
 - Interfaces graphiques

Clic de souris

évènements gérables

- press** bouton enfoncé
- click** bouton relâché
- drag** bouton enfoncé et déplacement
- drop** bouton relâché après déplacement

exemple

```
while 1:  
    if scene.mouse.clicked:  
        m = scene.mouse.getclick()  
        loc = m.pos  
        print loc  
        sphere(pos=loc, radius=0.1)
```

Déplacer un objet à la souris

Définir les coordonnées de l'objet à partir de la position de la souris.

exemple

```
boule=sphere ()  
dessine=False  
while 1:  
    if scene.mouse.clicked:  
        scene.mouse.getclick ()  
        dessine=not dessine  
        trace=curve ()  
    if dessine:  
        boule.pos=scene.mouse.pos  
        trace.append(pos=boule.pos)
```

Taper du texte

objet : `scene.kb`

`keys` Nombre d'évènements

`getkey()` Récupère un évènement

exemple

```
blabla=label()  
while 1:  
    s=scene.kb.getkey()  
    if len(s) == 1:  
        blabla.text += s  
    elif ((s == 'backspace' or s == 'delete')  
          and len(blabla.text) > 0):  
        blabla.text = blabla.text[:-1]  
    elif s == 'shift+delete':  
        blabla.text = ''
```

Plan

- 1 Introduction
 - Présentation de VPython
 - Premiers objets
 - Premières animations
- 2 Modélisation 3D
 - La scène
 - Les objets
 - Graphiques
- 3 **Animation et interaction**
 - Contrôle de l'animation
 - Les évènements
 - **Interfaces graphiques**

Boutons, curseurs, etc.

Fenêtre de contrôle

- fenêtre spéciale de type **controls**
- méthode **interact()** pour récupérer les évènements
- associe des fonctions aux contrôles

Objets de contrôle

button bouton à cliquer

slider curseur pour valeur numérique

toggle manette à deux positions

menu liste de boutons

Utilisation de Tkinter

- 1 Créer une fenêtre en **Tk**
- 2 Définir les contrôles en **Tk**
- 3 Définir la scène **VPython**
- 4 Lancer l'animation de la scène dans un **thread**
- 5 Lancer la boucle d'évènements **Tk**
- 6 En sortie de boucles terminer le **thread**

Résumé

- **VPython** est relativement intuitif
 - paramètres par défaut,
 - visualisation immédiate.
- **VPython** permet de réaliser facilement des modèles 3D.
- **VPython** est un outil pédagogique libre et indépendant de la plateforme.
- En plus **VPython** est amusant.

- Prolongements
 - Utilisation de **VPython** dans l'enseignement en master de mécanique
 - Applications avec **VPython**