# VRML 1.0



RETOUR

Dernière modification 20/09/10 06:57:36

## Introduction

VRML (Virtual Reality Markup ou Modeling Language): Langage de description de scènes 3D

"Equivalent" pour la 3D à ce que permet de faire HTML en 2D.

Deux versions de VRML:

- VRML 1.0
- VRML 97

VRML 97 non totalement syntaxiquement compatible avec VRML 1.0 quoique très ressemblant. Reprise en version 97 des fonctio l'intégration de:

- l'animation,
- les contrôles souris,
- ...

X3D: Successeur de VRML

- Concepts identiques à ceux de VRML'97
- Syntaxe xml
- Ouvert et extensible
- En évolution permanente (version actuelle: 3.2)

## Fonctionnalités (VRML 1.0)

Modélisation de scènes 3D illuminées, affectées de matériaux ou texturées

Utilisation de caméras de visualisation

Création de zones cliquables permettant de réaliser:

- le lien vers d'autres mondes VRML ou pages WEB,
- le chargement d'images fixes et animées, de sons, ...

## Fichier VRML

Une scène VRML est décrite dans un fichier VRML caractérisé par une entête définissant son type.

```
#VRML V1.0 ascii
```

est l'entête pour un fichier VRML 1.0 au format texte.

## Concepts et syntaxe de la modélisation

### L'arborescence de scène

Document VRML: Ensemble de nœuds (objets) organisés sous forme hiérarchique (gestion de listes avec la possibilité d'inclure d -> arborescence de scène

Un fichier VRML est lu du début à la fin. L'ordre d'apparition des nœuds correspond au parcours prioritairement en profondeur possède donc une influence directe sur la scène modélisée.

Tout nœud possède un type (Sphere, Cube, WWWInline, Separator,...).

Deux importantes catégories de nœuds:

- les nœuds "de forme" (shape node) (cube, sphère, ensemble de facettes, ...)
- les nœuds "propriété" (property node) (matériel, transformation géométrique, lumière, caméra, ...)

Les nœuds propriété modifient l'état interne du processus d'affichage de telle manière que le nouvel état ainsi configuré reste a Chaque nœud de forme créé est affecté de l'ensemble des caractéristiques (matériel, texture, éclairage, caméra de visualisatio -> Un nœud de forme ne peut être concerné que par les nœuds propriété le précédant dans l'arborescence de scène (et donc le p précisément, il ne peut être affecté que par les nœuds propriété situés sur le chemin unique entre la racine de l'arborescence d

### Les nœuds

Un nœud peut comporter des champs qui ont pour rôle d'indiquer les informations le concernant.

Le nom du nœud est suivi, entre accolades, des noms et des valeurs de ses différents champs. L'espace est le délimiteur unique de virgule, de point-virgule ou de parenthèses). Un champ non renseigné (nom et valeur(s) non indiqués) est initialisé à sa valeur p

Exemples: Sphere contient le seul champ radius.

```
Sphere { radius r }
```

- radius: rayon (valeur par défaut 1)

Cube en contient 3.

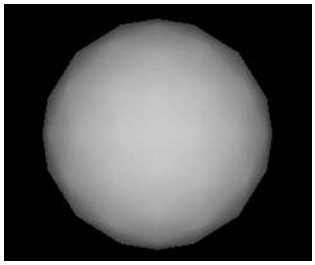
```
Cube { width w height h depth d }
```

- width, height, depth: largeur, hauteur et profondeur (x, y, z) (valeurs par défaut 2)

**Exemple**

```
#VRML V1.0 ascii Sphere { radius 10 }
```

Scène composée d'une unique boule blanche



### Les nœuds de forme de génération de volumes

```
Sphere { radius r }
```

- radius: rayon

**Exemple**

```
Cube { width w height h depth d }
```

- width: largeur (x)
- height: hauteur (y)
- depth: profondeur (z)

**Exemple**

```
Cone { parts p height h bottomRadius br }
```

- parts: composition de SIDES, BOTTOM et ALL
- height: hauteur (y)
- bottomRadius: rayon de la base

**Exemple**

```
Cylinder { parts p height h radius br }
```

- parts: composition de SIDES, BOTTOM, TOP et ALL
- height: hauteur (y)
- radius: rayon

Accès au testeur

**Exemple**

### Les nœuds de groupe

**Nœud de groupe**: Nœud qui, outre ses propres champs s'il en possède, peut contenir d'autres nœuds.

Tout le contenu d'un nœud de groupe est vu comme un seul objet.
-> manipulation d'un grand nombre d'objets comme un seul objet.

**Separator**: Nœud de groupe sans champs destiné à contenir d'autres nœuds.
Les modifications apportées à l'état interne du rendeur VRML au moyen de noeuds propriété (transformations géométriques, ma
l'intérieur d'un nœud Separator sont annulées avec reconfiguration à l'état du moment de l'entrée dans le Separator, une fois re
fichier VRML.

C'est via les nœuds Separator que sont structurées les arborescences de scène dans les fichiers VRML.

**Exemple**: Voir plus loin, voir plus loin, voir plus loin

### Autres nœuds

### Transformations géométriques (propriété)

Tout ce qui se trouve dans un nœud de groupe, faute d'indication contraire, est défini dans le même repère de modélisation: le r

Les transformations géométriques sur le repère de modélisation peuvent être réalisées au moyen des nœuds:

- Translation

```
Translation {
  translation tx ty tz
}
```

  - translation: déplacement de tx en x, ty en y et tz en z

- Rotation

```
Rotation {
  rotation ax ay az a
}
```

  - rotation: rotation de l'angle a (en radians) autour de l'axe (ax,ay,az)

- Scale

```
Scale {
  scaleFactor fx fy fz
}
```

  - scaleFactor: zoom de facteurs fx, fy et fz selon les axes x, y et z vis à vis de l'origine

- MatrixTransform

```
MatrixTransform {
  matrix m00 m01 m02 m03
         m10 m11 m12 m13
         m20 m21 m22 m23
         m30 m31 m32 m33
}
```

  - matrix: transformation selon la matrice donnée en paramètre (notée en coordonnées homogènes)

- Transform

Le nœud Transform définit une transformation géométrique consistant dans l'ordre en un zoom par rapport à un point arbit
d'un axe arbitraire et une translation.

```
Transform {
  translation tx ty tz
  rotation ax ay az a
  scaleFactor fx fy fz
  scaleOrientation ax ay az a
  center x y z
}
```

  - translation: translation terminale
  - rotation: rotation intermédiaire
  - scaleFactor: zoom initial
  - scaleOrientation: orientation du zoom
  - center: centre du zoom

```
Transform {
  translation T1
  rotation R1
  scaleFactor S
  scaleOrientation R2
  center T2
}
```

est équivalent à la séquence

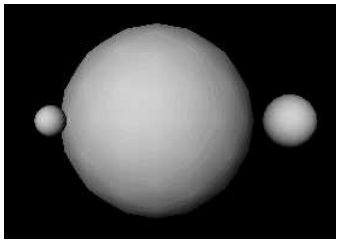
```
Translation { translation T1 }

Translation { translation T2 }

Rotation { rotation R1 }

Rotation { rotation R2 }

Scale { scaleFactor S }

Rotation { rotation -R2 }
```

```
Translation { translation -T2 }
```

**ATTENTION:** Un nœud transformation géométrique compose la transformation géométrique courante de l'environnement VRML spécifiée.

**Exemple**

```
#VRML V1.0 ascii Separator { Sphere { radius
          10 } Separator { Transform { translation 10 0 15 } Sphere { radius 2 } } Separator { Trans
          { radius 1.5 } } }
```



### Material (propriété)

Material: Spécifie les caractéristiques de rendu optique d'une surface.

```
Material { ambientColor r v b diffuseColor
          r v b specularColor r v b emissiveColor r v b shininess f transparency f }
```

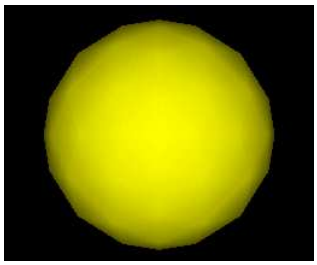
Accès au testeur

- ambientColor: réponse à la lumière ambiante (si elle existe)
- diffuseColor: réponse en réflexion diffuse aux lumières
- specularColor: réponse en réflexion spéculaire aux lumières
- emissiveColor: couleur émise
- shininess: réflectivité
- transparency: transparence

**Exemple 1**

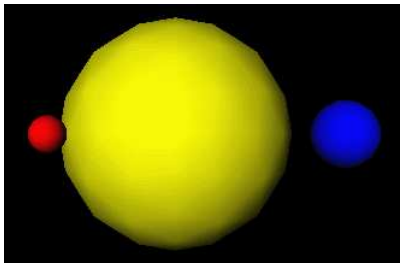
```
#VRML V1.0 ascii Separator { Material { diffuseColor
          1 1 0 } Sphere { radius 10 } }
```

La sphère définie après Material est affectée de la couleur jaune. Il s'agit de la couleur qu'elle réfléchit par diffusion.



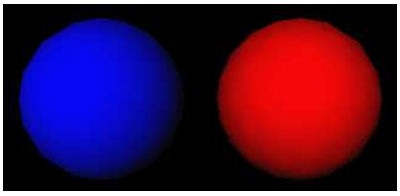
**Exemple 2**

```
#VRML V1.0 ascii Separator { Material { diffuseColor
          1 1 0 } Sphere { radius 10 } Separator { Transform { translation 10 0 15 } Material { dif
          2 } } Separator { Transform { translation -10 0 5 } Material { diffuseColor 1 0 0 } Sphere
```



**Exemple 3**

```
#VRML V1.0 ascii Separator { Translation { translation 0 0 -30 } Material { diffuseColor (
{ translation 12 0 0 } Material { diffuseColor 1 0 0 } Sphere { radius 10 } } Separator {
0 0 } Sphere { radius 10 } } }
```



### Les lumières

Plusieurs nœuds lumière sont utilisables en VRML:

- PointLight: Lumière ponctuelle isotropique

Accès au testeur

```
PointLight {
  on TRUE
  intensity v
  color r v b
  location x y z
}
```

- on: allumé ou non
- intensity: énergie émise
- color: couleur de la lumière émise
- location: position de la source lumineuse

- DirectionalLight: Lumière émise dans une direction (pas d'atténuation en fonction de la distance).

```
DirectionalLight {
  on TRUE
  intensity v
  color r v b
  direction dx dy dz
}
```

- on: allumé ou non
- intensity: énergie émise
- color: couleur de la lumière émise
- direction: direction du vecteur éclairage

- SpotLight: Lumière ponctuelle émise isotropiquement ou non dans un cône.

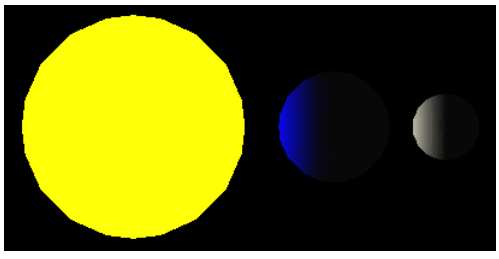
```
SpotLight {
  on TRUE
  intensity v
  color r v b
  direction dx dy dz
  location x y z
  dropOffRate v
  cutOffAngle v
}
```

- on: allumé ou non
- intensity: énergie émise
- color: couleur de la lumière émise
- direction: direction privilégiée du vecteur éclairage (axe du cone)
- location: position de la source lumineuse
- dropOffRate: diminution de l'éclairage avec l'éloignement à la direction privilégiée
- cutOffAngle: dimension angulaire du cône d'éclairage

Ils éclairent les objets définis après eux.

**Exemple**

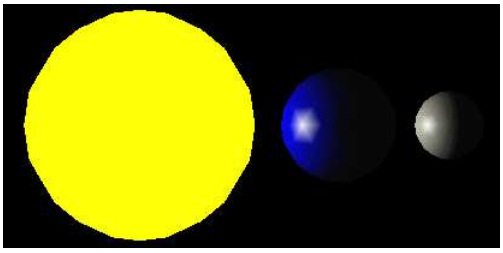
```
#VRML V1.0 ascii Separator { PointLight {
          intensity 1 location 0 0 0 color 1 1 0.9 } Material { emissiveColor 1 1 0 } Sphere { radiu
          { translation 18 0 0 } Material { diffuseColor 0 0 1 shininess 0 } Sphere { radius 5 } Sep
          10 0 0 } Material { diffuseColor 0.7 0.7 0.7 shininess 0 } Sphere { radius 3 } } } }
```



Utilisation d'un PointLight au centre du soleil

**Exemple**

```
Material { diffuseColor 0 0 1 specularColor
          0.9 0.9 0.9 shininess 0.8 }
```



Modification de la surface de la Terre et de la Lune
par apport d'une réflexion spéculaire

### Les liens

Les objets d'une scène VRML peuvent être rendus cliquables.

-> création de liens vers d'autres mondes virtuels ou, plus simplement, d'autres pages WEB ou tout autre document.

Le nœud WWWAnchor

WWWAnchor est un nœud de groupe.

-> il contient les nœuds à lier avec l'URL indiquée.

```
WWWAnchor {
  name "xxx"
  description "yyy"
}
```

- name: URL à lier
- description: texte de description

name et description doivent être donnés entre guillemets.

**Exemple**

```
#VRML V1.0 ascii Separator { Material { emissiveColor
          1 1 0 } WWWAnchor { name "http://www.univ-fcomte.fr" description "Un lien vers la Franche-
          { radius 10 } } }
```

### Les caméras

Les caméras VRML effectuent par défaut une visualisation selon l'axe -z. Il est toutefois possible de définir une rotation appliq

Il est aussi possible de choisir la position de la caméra.

Deux types de caméra:

- projection en perspective

```
PerspectiveCamera {
  position x y z
  orientation dx dy dz a
  focalDistance f
  heightAngle a
}
```

- position: position de la caméra
- orientation: orientation dans l'espace définie par la rotation de l'axe de visualisation de la caméra (-z) pour une valeur
- focalDistance: distance aux objets à visualiser (à ne pas confondre avec la distance focale en photographie)
- heightAngle: angle d'ouverture en y de la caméra

- projection orthographique

```
OrthographicCamera {
  position x y z
  orientation dx dy dz a
  focalDistance f
  height a
}
```

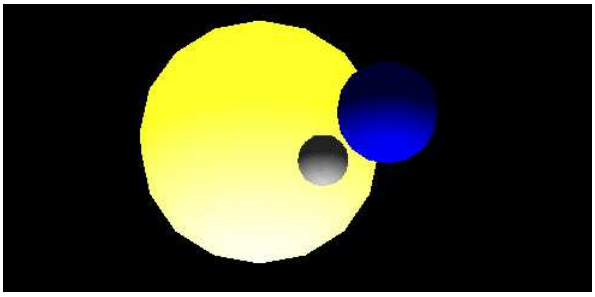
- position: position de la caméra

- orientation: orientation dans l'espace définie par la rotation de l'axe de visualisation de la caméra (-z) pour une valeur
- focalDistance: distance aux objets à visualiser (à ne pas confondre avec la distance focale en photographie)
- height: ouverture en y de la caméra

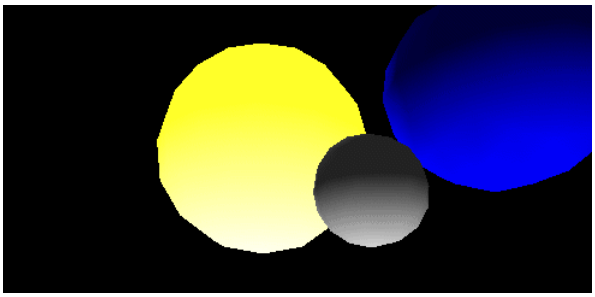
Ces caméras permettent de définir des points de vue dans les scènes VRML.

Une caméra peut être définie globalement pour une scène par définition immédiate après l'entête de fichier VRML. Si elle appar
les noeuds situés après elle dans l'arborescence de scène. Dans ce dernier cas, les objets sont affichés comme visualisés par ce

```
PerspectiveCamera {
  position 2 375 140
  orientation 1 0 0 -1.215
  focalDistance 60
  heightAngle .06
}
```

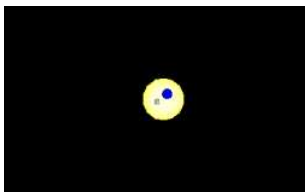


```
PerspectiveCamera {
  position 2 32 12
  orientation 1 0 0 -1.215
  focalDistance 60
  heightAngle 0.8
}
```

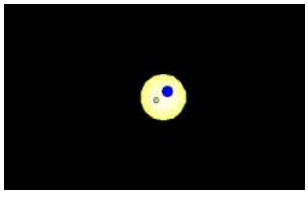


**Exemples**

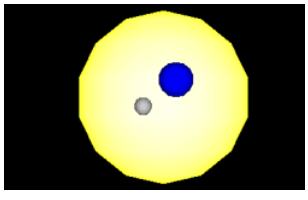
- Visualisation par défaut: projection en perspective. Les objets sont éloignés donc petits à l'écran et peu déformés.



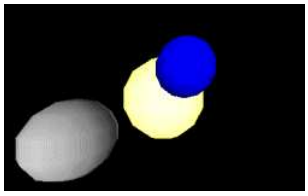
- Visualisation orthographique: projection orthographique avec un grand volume de visualisation -> les objets sont petits à l'éc



- Visualisation en perspective de loin: projection en perspective. Les objets sont éloignés mais l'agrandissement est fort. Ils



- Visualisation en perspective de près: projection en perspective. Les objets près de l'observateur sont grands à l'écran et tr
et peu déformés.



### Les textures

VRML autorise le texturage des objets d'une scène.

Le nœud propriété Texture2 permet cette réalisation.

Une texture peut être donnée sous deux formes:

- octet par octet pour tous les pixels de l'image à placer,

- un fichier image au format JPEG (éventuellement PNG et GIF acceptés).

```
Texture2 {
  image rx ry b
  ...
  filename "file"
  wrapS v
  wrapT v
}
```

- image: résolution en x, puis résolution en y, puis nombre d'octets par pixel, puis la liste des octets de l'image (format h
- filename: non du fichier contenant l'image
- wrapS: répétition ou non de la texture selon l'axe S (REPEAT ou CLAMP)
- wrapT: répétition ou non de la texture selon l'axe T (REPEAT ou CLAMP)

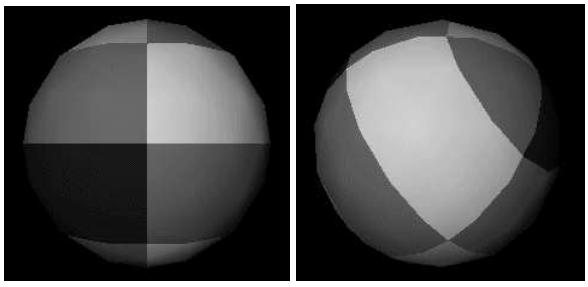
Tous les objets placés après le nœud texture sont texturés au moyen de ce nœud.

#### Champ image

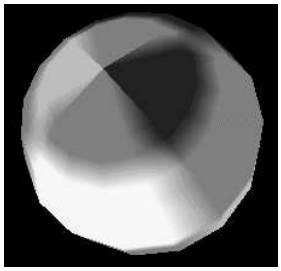
La couleur des pixels est directement donnée dans le fichier VRML sous la forme de la résolution en x, puis de la résolution en y, et enfin, d'une suite de pixels en notation hexadécimale.

**Exemple**

```
Texture2 { image 4 4 3 0xC0C0C0 0x808080
           0xFFFFFF 0x404040 0x808080 0x202020 0x808080 0xC0C0C0 0x202020 0x808080 0xFFFFFF 0x808080
           }
```



Dans le SceneViewer



Dans le plugin WorldView

#### Champ filename

On indique ici l'URL d'un fichier contenant l'image à mapper.

**Exemple**

```
Texture2 {
  filename "bugs.jpg" }
```

### Les ensembles de facettes

(1) Utilisation du nœud Coordinate3 pour indiquer les coordonnées 3D des sommets

```
Coordinate3 {
  point [
  x0 y0 z0,
  x1 y1 z1,
  x2 y2 z2,
  ...
  ]
}
```

- point: liste des coordonnées définies (entre crochets)

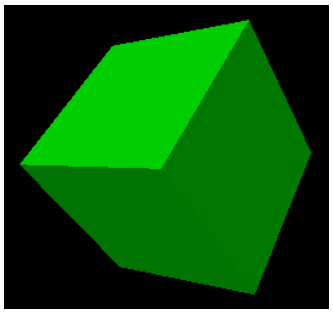
(2) Utilisation du nœud IndexedFaceSet pour indiquer la liste des facettes par indexation dans la liste des sommets

```
IndexedFaceSet { coordIndex [ s1 s2 ... sn
          -1, s1 s2 ... sm -1, s1 s2 ... sp -1, ... ] materialIndex normalIndex textureCoordIndex }
```

- coordIndex: liste des facettes (entre crochets), chaque facette est définie par la liste des indices de ses sommets (lis
- materialIndex: idem ci-dessus mais pour des matériaux
- normalIndex: idem pour les normales
- textureCoordIndex: idem pour les coordonnées de texturage

#### Exemple

```
#VRML V1.0 ascii Separator { Material { diffuseColor
          0.0 1.0 0.0 ambientColor 0.0 0.1 0.0 } Coordinate3 { point [ 1.0 -1.0 1.0, 1.0 -1.0 -1.0,
          1.0 1.0 -1.0, -1.0 1.0 -1.0, -1.0 1.0 1.0, 1.0 1.0 1.0 ] } IndexedFaceSet { coordIndex [ 2
          2, -1, 4, 2, 5, -1, 5, 2, 3, -1, 5, 3, 6, -1, 6, 3, 0, -1, 6, 0, 7, -1, 4, 5, 6, -1, 4, 6,
          -1 ] } }
```



### Instanciation d'objets

### DEF

DEF permet d'attribuer un nom à un nœud (un objet) de manière que ce nœud puisse être réutilisé par la suite (cet objet puisse

#### Syntaxe

DEF nom_objet type_de_noeud {

}

#### Exemple

```
DEF Soleil Separator { Material { diffuseColor
          1 1 0 } Sphere { radius 10 } }
```

### USE

A tout instant, un appel à USE permet d'instancier un objet nommé.

#### Syntaxe

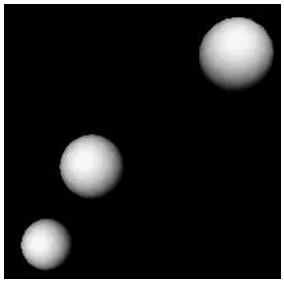
USE nom_objet

**Exemple**

USE Soleil

**Exemple**

```
#VRML V1.0 ascii DEF Planete Separator {
          Material { diffuseColor 1.0 1.0 1.0 ambientColor 0.1 0.1 0.1 } Sphere { } } Translation {
          Translation { translation 4 3 2 } WWWAnchor { name "http://www.univ-fcomte.fr" descriptio
          USE Planete }
```



### Le nœud WWWInline

WWWInline est un nœud de groupe permettant l'inclusion par référence d'objets à l'intérieur d'une scène.

-> non construction entière d'un objet, mais seulement référenciation des parties de cet objet comme provenant d'autre docume

```
WWWInline {
  name "xxx"
  bboxSize dx dy dz
  bboxCenter x y z
}
```

- name: URL du fichier VRML contenant l'objet à inclure
- bboxSize: taille de la boite englobant l'objet référencé
- bboxCenter: position de la boite englobante

La boite englobante est affichée en filaire tant que l'objet référencé n'est pas chargé.

Elle doit être d'une taille voisine de l'objet qu'elle représente.

**Exemple**

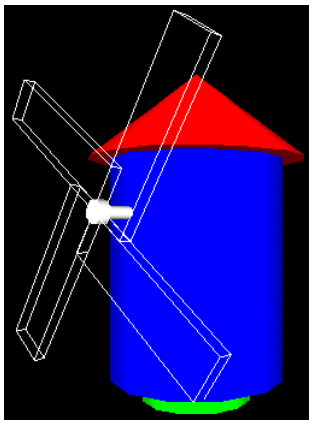


Image intermédiaire

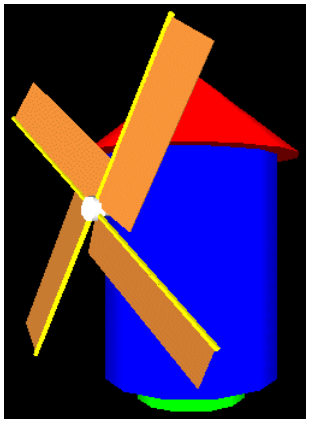


Image finale

### Le nœud LOD

Le nœud de groupe LOD permet l'adaptation d'un objet visualisé à la distance d de visualisation.

```
LOD {
  range [v1,v2,...,vn]
  center x y z
}
```

- range: liste de n distances croissantes
- center: position de référence pour le nœud LOD

Un nœud LOD contient théoriquement n+1 nœuds.

Seul le 1er est évalué si d < v1.

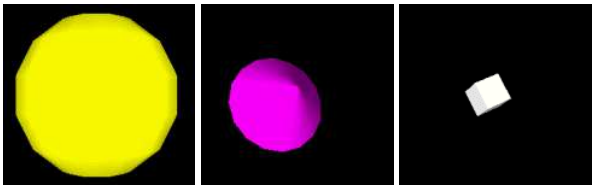
Seul le 2ème est évalué si v1 < d < v2.

...

Seul le n+1ème est évalué si d > v2.

**Exemple**

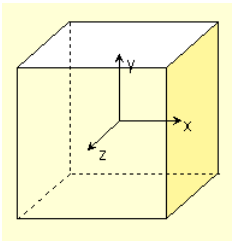
```
#VRML V1.0 ascii Separator { LOD { range
          [ 50,100 ] Separator { Material { diffuseColor 1 1 0 } Sphere { radius 10 } } Separator {
          } Cone { bottomRadius 10 height 10 } } Separator { Material { diffuseColor 1 1 1 } Cube {
          } } }
```



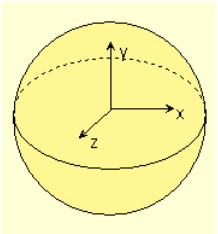
### Récapitulatif des nœuds et valeur par défaut des champs

#### Les nœuds de forme

```
Cube {
  width 2
  height 2
  depth 2
}
```



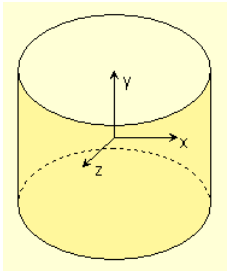

```
Sphere {
  radius 1
}
```




```
Cylinder {
  parts ALL
  radius 1
  height 2
}
```

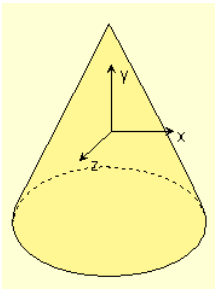
parts: composition de SIDES, TOP, BOTTOM et ALL

Exemple: (SIDES|TOP)




```
Cone {
  parts ALL
  bottomRadius 1
  height 2
}
```

parts: composition de SIDES, BOTTOM et ALL




### Les autres noeuds

```
Coordinate3 {
  point 0 0 0 # M
}

DirectionalLight {
  on TRUE
  intensity 1
  color 1 1 1
  direction 0 0 -1
}

IndexedFaceSet {
  coordIndex 0 # M
  materialIndex -1 # M
  normalIndex -1 # M
  textureCoordIndex -1 # M
}

IndexedLineSet {
  coordIndex 0 # M
  materialIndex -1 # M
  normalIndex -1 # M
  textureCoordIndex -1 # M
}

LOD {
  range [ ] # M
  center 0 0 0
}

Material {
  ambientColor 0.2 0.2 0.2
  diffuseColor 0.8 0.8 0.8
  specularColor 0 0 0
  emissiveColor 0 0 0
  shininess 0.2
  transparency 0
}

MatrixTransform {
  matrix 1 0 0 0
         0 1 0 0
         0 0 1 0
         0 0 0 1
}

OrthographicCamera {
  position 0 0 1
  orientation 0 0 1 0
  focalDistance 5
```

```
  height 2
}

PerspectiveCamera {
  position 0 0 1
  orientation 0 0 1 0
  focalDistance 5
  heightAngle 0.785398
}

PointLight {
  on TRUE
  intensity 1
  color 1 1 1
  location 0 0 1
}

PointSet {
  startIndex 0
  numPoints -1
}

Rotation {
  rotation 0 0 1 0
}

Scale {
  scaleFactor 1 1 1
}

SpotLight {
  on TRUE
  intensity 1
  color 1 1 1
  location 0 0 1
  direction 0 0 -1
  dropOffRate 0
  cutOffAngle 0.785398
}

Texture2 {
  filename ""
  image 0 0 0
  wrapS REPEAT
  wrapT REPEAT
}

WRAP ENUM
REPEAT Repeats texture outside
       0-1 coordinate range
CLAMP Clamps texture coordinates
      to lie within 0-1 range

Transform {
  translation 0 0 0
  rotation 0 0 1 0
  scaleFactor 1 1 1
  scaleOrientation 0 0 1 0
  center 0 0 0
}

Translation {
  translation 0 0 0
}

WWWAnchor {
  name ""
  description ""
  map NONE
}

MAP ENUM
NONE Do not add information to the URL
POINT Add object-space coordinates to URL

WWWInline {
  name ""
  bboxSize 0 0 0
  bboxCenter 0 0 0
}
```

**Exemples**


VRML
VIRTUAL REALITY MODELING LANGUAGE