Université Abou Bekr Belkaid
Tlemcen Algérie
جامعة أبي بكر بلقايد
تلمسان الجزائر

République Algérienne Démocratique et Populaire
Université Abou Bakr Belkaid– Tlemcen
Faculté des Sciences
Département d'Informatique

Mémoire de fin d'études

Pour l'obtention du diplôme de Licence en Informatique

Thème

Visite virtuelle dans un établissement avec VRML

Réalisé par :

- **M^me^ Alioui Wahiba.**
- **M ^elle^ Belkhodja Souad.**

Présenté le 27 Juin 2013 devant la commission d'examination composée de MM.

- ***M^r^ BENZIANE Mohammed.*** *(Examinateur)*
- ***M^me^ CHAOUCHE Ramdane Lamia.*** *(Examinateur)*
- ***M^r^ BENAZZOUZ Mourtada.*** *(Examinateur)*

Année universities : 2012-2013

DEDICACE

Je dédie ce modeste travail :

A mes parents

A mes frères : Lakhdar, Ali, Abdelwahab, Mohammed.

A mes sœurs : Zoubida, Nouria, Zahia.

A les petits : Salsabil, Alaa, Iyad.

A tous mes amies.

Et à mon binôme Wahiba.

Souad.

DEDICACE

Je dédie ce modeste travail :

A mes parents et ma grande mère.

A mes frères.

A mes sœurs.

A ma chère petite : Zahra Nardjisse.

A tous mes amies.

A ma cousine Fatiha.

Et à mon binôme Souad.

Wahiba.

SOMMAIRE :

Chapitre 4 :

Introduction Général

Le langage VRML (Virtual Reality ModelingLanguage) est un langage standardisé servant à décrire des objets 3D interactifs. VRML permet de décrire des mondes en trois dimensions dans lesquels l'utilisateur peut se déplacer et interagir avec les objets qui l'entourent.

Son point fort est sa compatibilité avec le web : de la même façon que HTML permet d'échanger des documents écrit dans le monde entier, VRML permet de diffuser des "Mondes virtuels". En effet, une des premières motivations des pionniers du VRML était de marcher et de travailler dans un monde virtuel partagé. La simplicité et la puissance de ce langage à permet de le retrouver dans différents domaine comme la conception des scènes. Notre travail principal, est visite dans un établissement avec VRML en exploitant les propriétés de ce langage. Notre projet est représentée par une bibliothèque avec plusieurs objets en 3D avec une vue virtuelle.

La bibliothèque de notre projet est composée de plusieurs bureaux et chaqu'un constitué par des murs, des ouvertures (portes, fenêtre) et bien entendu tous les meubles.

Dans le but de bien comprendre le fonctionnement de VRML nous avons essayé de mettre en œuvre le plus de nœuds disponibles dans la syntaxe VRML.

Notre mémoire est composée de quatre chapitres.En premier temps nous avons entamé une vision globale sur l'infographie, et en second en parle sur modélisation des objets solide et la réalité virtuelle. En fin nous présentons la syntaxe de langage VRML et nous présentons les étapes de la conception et la réalisation de notre projet virtuelle par le langage VRML.

chapitre 1:
infographie

1. INTRODUCTION :

L'**infographie** est le domaine de la création d'images numériques assistée par ordinateur.

Lors de l'introduction du concept dans la langue française vers les années 1970, elle désigne uniquement les graphismes, que l'on appelle alors « infographies » (information par le graphisme), destinés à mettre en image des informations généralement statiques au moyen de diagrammes, de cartes ou de schémas. Mais le concept d'infographie s'étant rapidement élargi à tous les graphismes produits par des moyens informatiques. En anglais le terme est resté : infographies signifie graphisme d'information. C'est donc un faux-ami.

Il s'agit aussi de la technique qui consiste à finaliser le travail du graphiste à l'aide de l'outil informatique. Ce métier est né avec l'avènement de l'informatique (il est la continuité du graphisme.

2. Les applications de l'informatique graphique :

L'informatique graphique trouve ses applications dans presque tous les domaines de la création, de la conception, de la fabrication, de l'information.

En architecture, on peut représenter des bâtiments avec tous leurs détails; on peut visionner ces bâtiments avec n'importe quel point de vue.

En cartographie, l'ordinateur peut produire très rapidement des cartes avec différents types d'éléments géographiques. Dans une carte météorologique, par exemple, on peut utiliser des symboles comme des nuages ou le soleil pour indiquer quel temps il va faire dans chaque région.

Notons que ce type d'application est utilisée par la télévision et dans le cadre des système vidéotex.

Dans le domaine des transports, des cartes de réseaux de bus, de chemins de fer ou d'avions.

En médecine, de nombreuses applications existent. Grâce au traitement d'images, on peut facilement détecter certaines maladies. Par exemple, en cardiologie, on peut déceler l'infarctus du myocarde par reconstruction d'images. Et on commence déjà à avoir des modèles humains entièrement synthétisés pour simuler des expériences.

Finalement, dans le domaine des arts autres que l'animation, l'ordinateur a encore de la peine à acquérir ses lettres de noblesse. Pourtant, des peintures et des sculptures sont

produites chaque année et des expositions font de plus en plus leur apparition. peuvent être produites par ordinateur, aussi bien pour les réseaux existants que pour la planification de nouveaux réseaux.

3. Modélisation d'Objets solides 3D :

La première étape dans la synthèse d'images par infographie 3D est celle de la modélisation d'un "objet" donné. Nous étudierons dans un premier temps de quoi se compose cet "objet" et comment il est pris en compte par le logiciel d'infographie 3D. Dans un second temps, nous mettrons en lumière les principes mathématiques et géométriques permettant la modélisation, et la manipulation d'objets dans l'espace virtuel 3D.

4. Transformation géométrique :

a- 2D versus 3D :

L'être humain vit dans un monde à trois dimensions, mais lorsqu'il dessine, il utilisegénéralementdesfeuillesdepapierquin'ontquedeuxdimensions.Ilsetrouvedonc confronté à un problème de représentation en deux dimensions d'un monde à trois. Deux solutions s'offrent alors:

- représenter seulement une face plane des objets ,par exemple la façade avant d'une maison ou le dessus d'une table
- tenté de dessiner la scène choisie en tenant compte de lois de projection

En informatique graphique, comme les supports matériels (écrans) sont à deux dimensions, ces deux approches se retrouvent et donnent lieu à deux types de modélisation, de systèmes graphiques et d'applications.

On dira qu'un système graphique est à deux dimensions (2D) si la représentation interne de l'information graphique dans l'ordinateur est à deux dimensions. Un système graphique sera à trois dimensions (3D) lorsque l'ordinateur a connaissance de l'information tridimensionnelle.

Cette distinction est fondamentale. En effet, lorsqu'on voit une image produite par ordinateur d'une maison en perspective, il est impossible de savoir si l'image a été produite avec un système à 2 ou à 3 dimensions. En effet, la maison a pu être dessinée en perspective et fournie ainsi à un système graphique à 2 dimensions qui s'est contenté de la restituer ou la vue en perspective a été synthétisée par un système tridimensionnel à partir de données tridimensionnelles. Ceci nous amène à préciser que lorsque nous parlerons d'images tridimensionnelles, il s'agira toujours d'images produites à partir

d'un modèle tridimensionnel connu de l'ordinateur et non d'images réellement en trois dimensions.

Il faut aussi remarquer que l'espace à deux dimensions peut être considéré comme un cas particulier d'espace à trois dimensions dont la troisième dimension Z est toujours nulle.

b. Le système de coordonnées 3D :

Avant tout, il convient de parler du mode de représentation utilisé pour placer un point dans l'espace. Les calculs nécessaires pour la transformation et la projection d'un point dépendent de cette représentation. Le système de coordonnées que l'on utilise pour la 3D est cartésien. Selon les outils utilisés, ce référentiel sera « main droite » (comme avec OpenGL ou VRML) ou « main gauche » (comme avec Direct3D).

Comme nous pouvons le voir dans la Figure suivante, que le système soit direct ou indirect, pour chacun d'eux les axes X et Y sont les mêmes (X de gauche à droite et Y de bas en haut de l'écran). La façon la plus naturelle de se représenter un système de coordonnées cartésien est d'utiliser ses deux mains (d'où les noms « main droite » et « main gauche») . Pour chacune des mains, il suffit de faire pointer le pouce (donnant l'axe X) vers la droite, l'index (l'axe Y) vers le haut. Le majeur, perpendiculaire aux deux doigts précédents (et à la paume de la main) nous donne automatiquement la direction positive de l'axe Z.

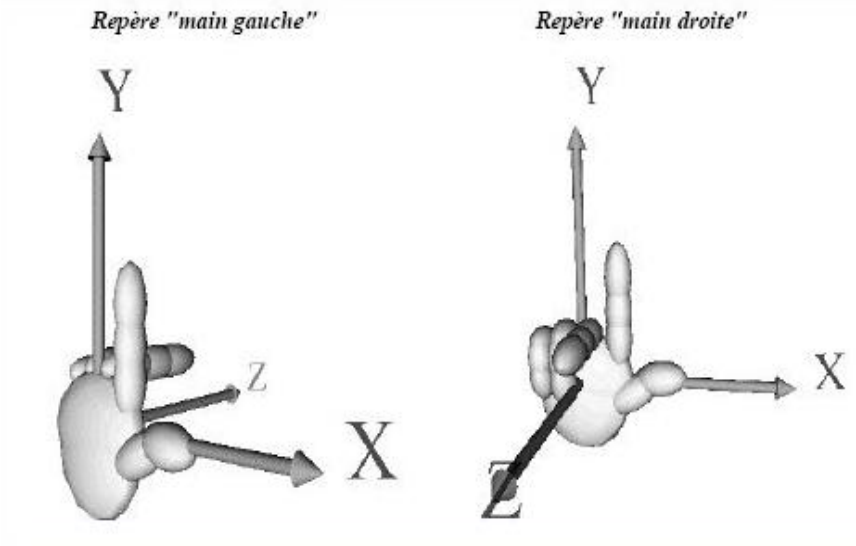


Figure 1 : direction des axes.

La transformation des coordonnées d'un système de coordonnées à l'autre se fera en prenant l'opposée de la coordonnées Z. Quant à la différence d'interprétation que

l'on peut donner à cette opposition de l'axe Z, pour le repère « main gauche » on parlera de profondeur pour la coordonnées Z d'un point alors que, pour le repère « main droite », on y verra plus volontiers une altitude.

Toute primitive graphique complexe est d'abord transformée en un ensemble de triangles . L'apparence obtenue par approximation en un nombre plus ou moins de triangles permet d'avoir soit un rendu plus rapide soit un rendu de meilleure qualité. Chacun des triangles subit alors les différentes étapes du pipeline graphique dont nous donnons pour les plus importantes le détail des calculs :

Transformations : opérations matricielles donnant le transformé d'un point par une ou plusieurs des matrices suivantes :

translation d'un vecteur T:

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ T_x & T_y & T_z & 1 \end{bmatrix}$$

rotation d'un angle A autour de l'axe X:

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

rotation d'un angle B autour de l'axe Y:

$$R_y = \begin{bmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

rotation d'un angle C autour de l'axe Z:

$$R_z = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 & 0 \\ \sin\gamma & \cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

transformation d'échelle d'un facteur E:

$$E = \begin{bmatrix} E_x & 0 & 0 & 0 \\ 0 & E_y & 0 & 0 \\ 0 & 0 & E_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

5. La Projection :

En général, une projection transforme les points dans une dimension N vers des point dans une dimension inférieur à N par exemple N=3 vers N=2

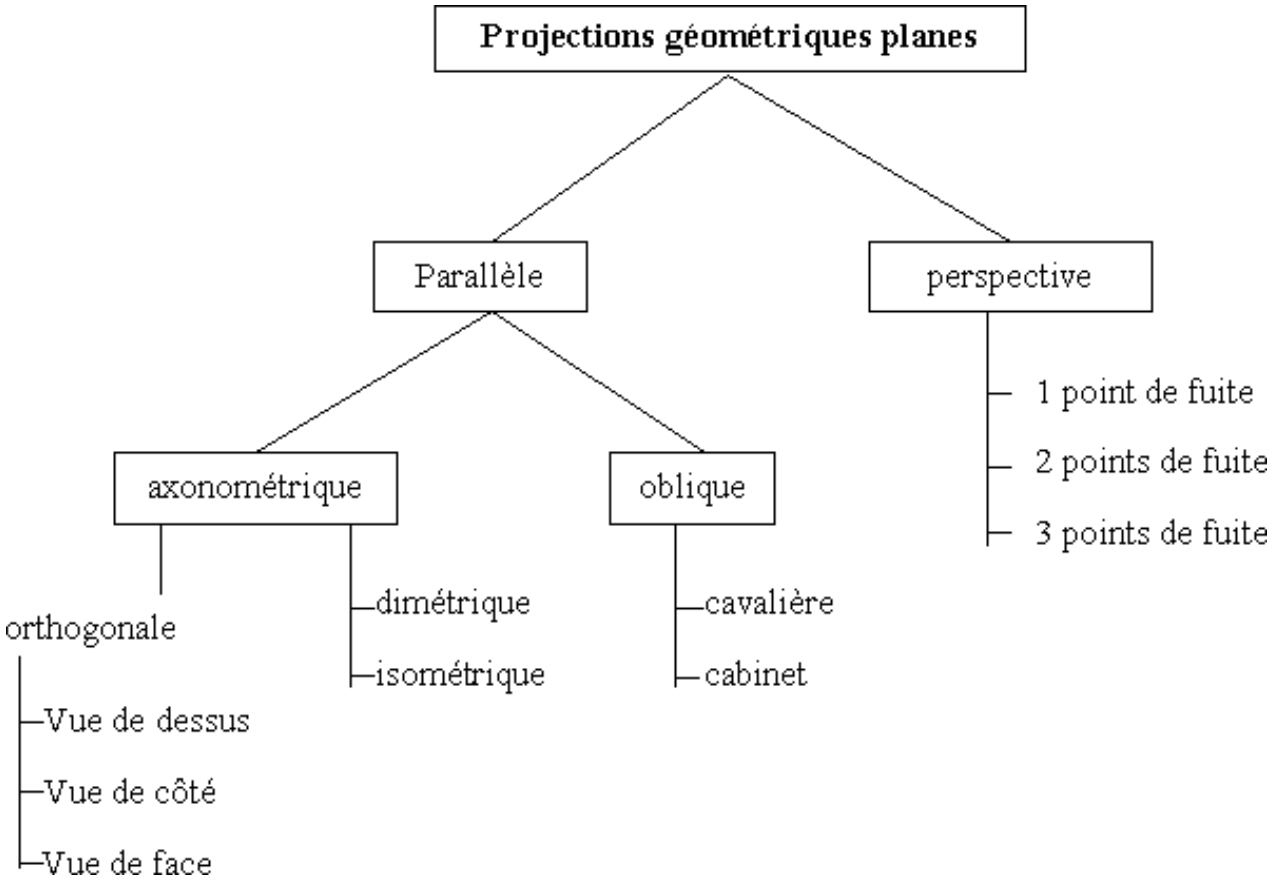


Figure2 : projection géométrique

.

a- Projection parallèle :

Les projections sont parallèles à une direction de projection (si direction de projection est perpendiculaire, alors projection orthographique).

1- projection axonométrique : définition d'une direction projection, plan de projection perpendiculaire à la direction de projection

- dimétrique
- isométrique
- orthographique (vue de dessus, de côté et vue de face)

2- projection oblique : le plan de projection n'est pas perpendiculaire à la direction de projection.

- projection cavalière (45°)
- projection cabinet : les distances fuyantes sont divisées par 2.

b-Projection perspective :

Les projections émanent d'un seul point (centre de projection) :

_Volume de vue : volume contenant tous les objets susceptible d'être projetés.

_Fenêtre 3D : rectangle délimitant ce qui doit être vu dans le plan de vue

Le centre de projection est à distance finie du plan de projection

--> vision "humaine".

Exemple : projection d'un cube

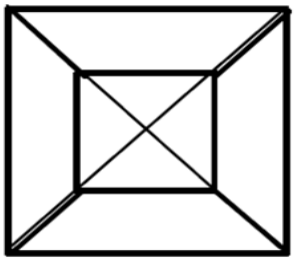


Figure3 : Projection à un point de fuite

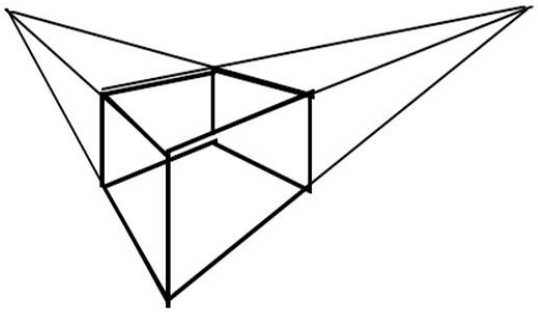


Figure4 : Projection à deux points de fuites

6. Caméra virtuelle ou synthétique

_œil : point d'où l'on regarde (ou on positionne la caméra)

_point d'intérêt : point que l'on regarde

_angle de vue : 45^0 pour être humain

7. La couleur :

a- Le rôle de la couleur en informatique graphique :

En informatique graphique, la couleur joue un rôle fondamental pour deux raisons principales. D'une part, elle permet de distinguer des objets différents, d'autre part, elle est indispensable pour rendre les images réalistes. Pour distinguer N objets différents, il suffit de N couleurs et souvent moins; comme on manipule rarement beaucoup d'objets à la fois.

On peut donc se contenter d'un nombre limité de couleurs. Dans le cas d'images réalistes, on fait intervenir la lumière; mais, si on considère, par exemple, un objet complexe rouge, il faudra un grand nombre de nuances rouges pour le représenter. Plus généralement, la production d'images réalistes avec transparence, texture, ombrage nécessite une très grande quantité de couleurs.

On constate donc que selon le type d'applications envisagées, le besoin en couleurs est variable. Il faut pourtant des moyens standards de spécifier ces couleurs. Avec un nombre limité à 8 par exemple, on peut évidemment se servir des noms des couleurs; mais pour des milliers voire des millions de couleurs, des systèmes numériques sont indispensables. Il existe plusieurs systèmes dont les principaux sont : RGB, CMY, YIQ, HSV et HLS. Nous ne décrirons que les systèmes RGB et HLS.

b. Le système RGB :

Ce système est dérivé du système instauré en 1931 par la Commission Internationale de l'Eclairage (CIE). Le système de la CIE définissait un espace basé sur trois couleurs primaires: Rouge (**R**ed), Vert (**G**reen) et Bleu (**B**lue). Toute couleur visible devenait une combinaison linéaire des trois couleurs primaires. Le système RGB correspond au principe des moniteurs TV, puisque dans ces moniteurs, les couleurs sont créés par des phosphores rouges, verts et bleus.

Le modèle RGB utilise généralement un cube dont les arêtes sont de longueur 1 et qui est représenté à la Figure. Le noir est à l'origine et le blanc au point <1,1,1>. Les trois couleurs primaires se trouvent évidemment le long des trois axes. Enfin, les couleurs cyan, magenta et jaune sont situées aux trois autres sommets du cube. Toutes les couleurs peuvent ainsi être exprimées par des vecteurs de composantes comprises entre 0 et 1. On aura donc:

Rouge = <1,0,0> Vert = <0,1,0> Bleu=<0,0,1>

Jaune = <1,1,0> Magenta = <1,0,1> Cyan = <0,1,1>

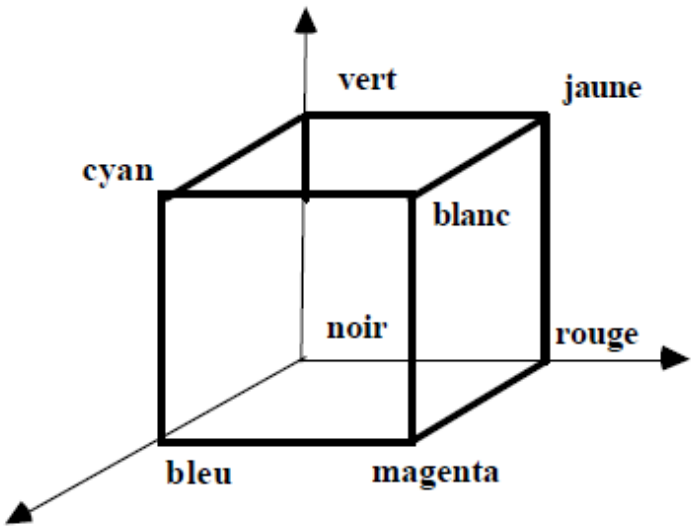


Figure5 : cube de couleur d'espace RGB

Il faut encore remarquer que le long de la diagonale du cube, on trouve une échelle des gris allant du noir (<0,0,0>) au blanc (<1,1,1>).

c. Le système HLS :

Ce système spécifie aussi toute couleur à l'aide de trois nombres, mais ces trois nombres ont une toute autre signification. Ce sont la nuance ou teinte (**H**ue), la clarté (**L**ightness) et la saturation (**S**aturation).

La nuance ou teinte H peut se représenter à l'aide d'un cercle; on considère alors l'angle au centre du cercle. En degrés, la teinte va donc de 0o à 360o comme le montre la Figure . Les trois couleurs primaires et les trois couleurs complémentaires forment un hexagone régulier.

La clarté L est définie selon une échelle allant de 0 (noir) à 1 (blanc) en passant par tous les gris.

La saturation S mesure la pureté des couleurs, c'est-à-dire le pourcentage de couleur pure par rapport au blanc. Une valeur de 1 représente donc une couleur pure ou saturée tandis qu'une valeur de 0 correspond à un gris de même clarté.

Le système HLS peut être illustré à l'aide d'un double cône. A la surface du cône, toutes les couleurs ont une saturation de 1. La saturation est représentée le long du rayon d'une section circulaire du cône. La teinte est décrite par l'angle au centre d'un cercle tandis que la clarté est sur l'axe vertical

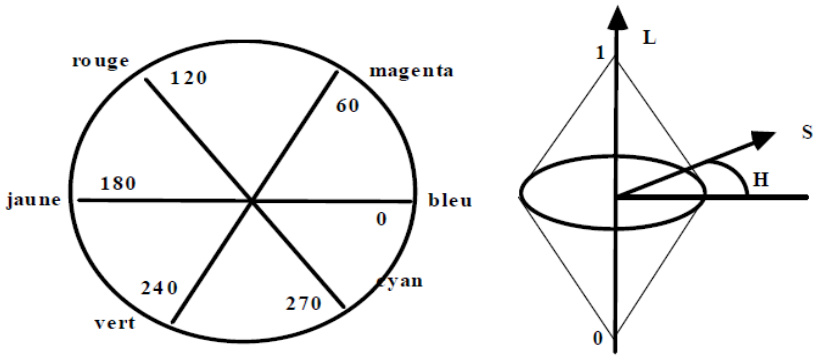


Figure 6 : espace HLS

Remarquons que la teinte est souvent exprimée en fraction de tour de cercle, ce qui permet de noter les couleurs par des vecteurs à trois composantes comprises entre 0 et 1: <H,L,S>. Ainsi, les couleurs primaires et complémentaires peuvent s'écrire:

Rouge = <0.33,0.5,1> Vert = <0.67,0.5,1> Bleu=<0,0.5,1>

Jaune = <0.5,0.5,1> Magenta = <0.167,0.5,1> Cyan = <0.833,0.5,1>

8. L’ombrage :

Une **ombre** est une zone sombre créée par l'interposition d'un objet opaque (ou seulement partiellement opaque) entre une source de lumière et la surface sur laquelle se réfléchit cette lumière. Elle se matérialise par une silhouette sans épaisseur. Une ombre portée contient la couleur complémentaire de la source de lumière qui la crée.

a. Caractéristiques :

La taille de l'*ombre portée* dépend de la taille de l'objet intercalé et de sa distance relative de la source de lumière. Plus l'objet est près de la source de lumière, plus la zone d'ombre sera grande. Son intensité dépend de la proportion de la lumière apportée par la source masquée. Si la source masquée est la seule source présente, alors la zone à l'ombre est dans l'obscurité totale.

En dessin technique, et de façon conventionnelle, on accepte que l'ombre projetée par le soleil soit le résultat de rayons incidents parallèles qui vont générer une ombre portée par l'objet sur le plan sur lequel il est posé, et une ombre propre créée sur les faces de l'objet qui sont soustraites aux rayons incidents.

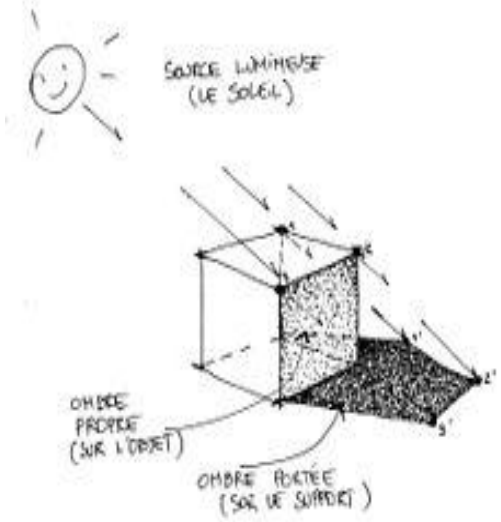


Figure7 : le soleil comme source lumineuse

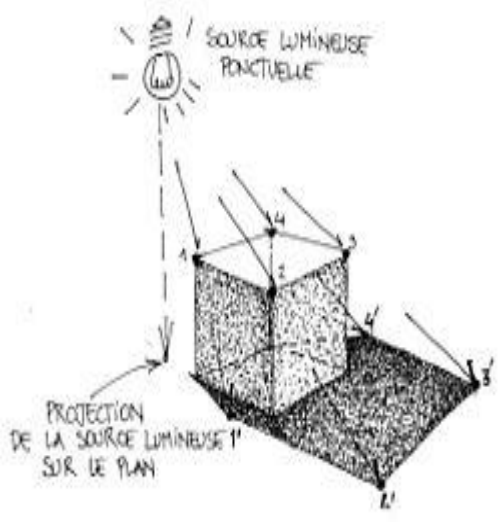


Figure8 :éclairage ponctuel comme source lumineuse

chapitre 2:

modelisation des objets

1. Modèle de système graphique :

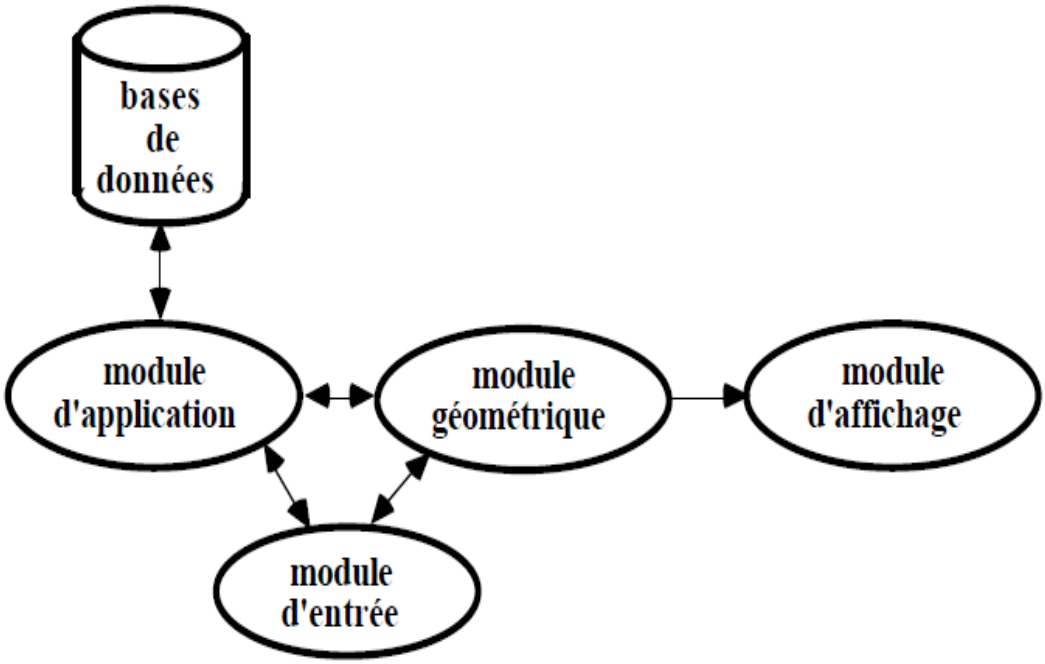


Figure1 : système graphique.

Dans ce modèle, on distingue quatre modules principaux:

Le module d'entrée :il est responsable de l'entrée des dessins et comprend normalement des dispositifs d'entrée tels qu'une tablette graphique, par exemple ,et le logiciel de contrôle de ces dispositifs.

Le module géométrique :il est formé essentiellement de logiciel et son rôle est de créer et de manipuler des objets graphiques ;il faut noter que les stations graphiques les plus récentes offrent de plus en plus de facilités matérielles pour créer et manipuler des objets graphiques :rotations matérielles ,générateur de cercles.

Le module d'affichage: il est responsable de la sortie des dessins et comprend le matériel de sortie , tel que les écrans ,les imprimantes ainsi que le logiciel d'affichage qui peut aller du simple ensemble de sous-programmes de traçage de lignes jusqu'au logiciel super sophistiqué de synthèse d'images avec multiple les sources de lumière, ombre portée, transparence.

Le module d'application: ce module est celui orienté vers l'usager; il diffère évidemment suivant le type d'application et se présente généralement comme un programme interactif. Ce modèle est évidemment théorique et pratiquement, on peut plutôt considéré qu'on a un ensemble de dispositifs matériels d'entrée et de sortie, un logiciel

graphique de base et un logiciel d'application. Le logiciel graphique de base comprend généralement des opérations d'entrée, de sortie et de création et manipulation d'objets graphiques.

2.Énumération Spatial :

Un solide est représenté par une liste de cellules occupées dans l'espace par l'objet, les cellules, appelées *voxels*(*volume éléments*) sont des cubes de taille fixe représentés par les coordonnées d'un point (ex : le centre du cube)

Cette représentation permet de réaliser facilement des opérations booléennes. Elle ne constitue qu'une approximation plus ou moins grossière du solide suivant la taille du voxel utilisé. Plus la précision est grande, plus la taille occupée en mémoire est importante Amélioration intéressante : utilisation d'arbres octaux (*octrees*) qui permettent de représenter les objets par une succession hiérarchique de cubes de taille variable

a. QuadTree :

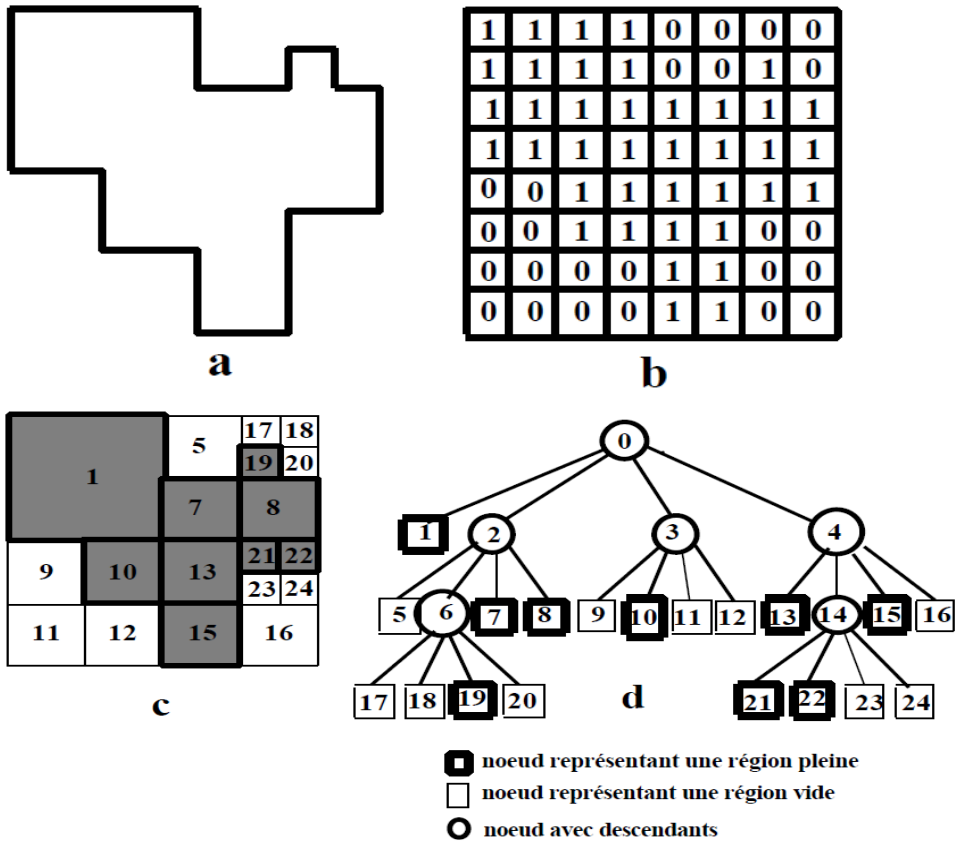
Un quadtree est une représentation d'un objet 2D basée sur la subdivision récursive d'un carré en 4 carrés de même taille (quadrants)

A chaque stade du processus récursif, deux cas sont possibles:

1. L'objet ne recouvre pas complètement le quadrant; il est alors subdivisé en 4 nouveaux carrés

2. Le quadrant est plein ou vide; la subdivision est alors stoppée et le quadrant est marqué comme PLEIN (1) ou VIDE (0).

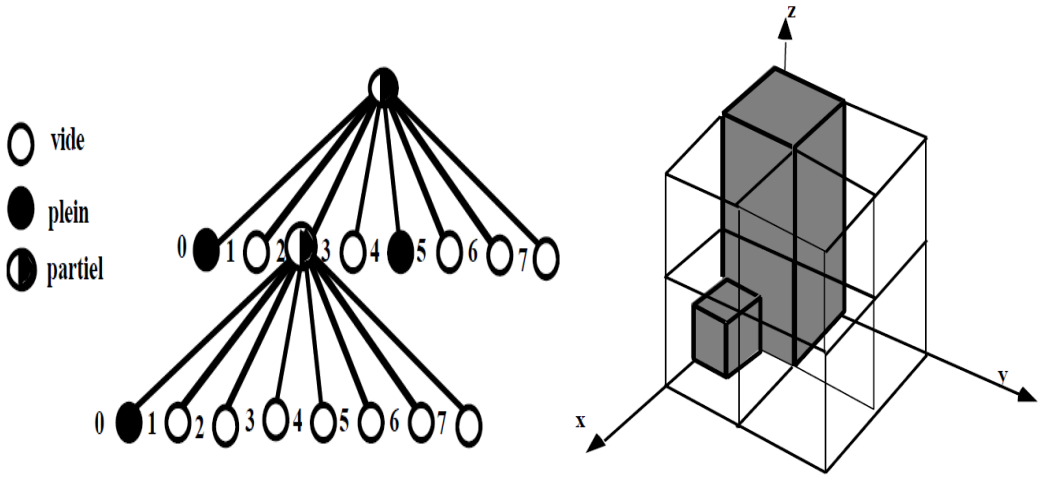
Exemple :



b.Octrees:

Les arbres octaux sons une variante hiérarchies de l'énumération spatiale, conçue pour rependre aux besoins en stockage de cette approche, les octrees sont dérives des arbres quaternaires (quadtree), un format de représentation utilisée pour coder des images.

Exemple :

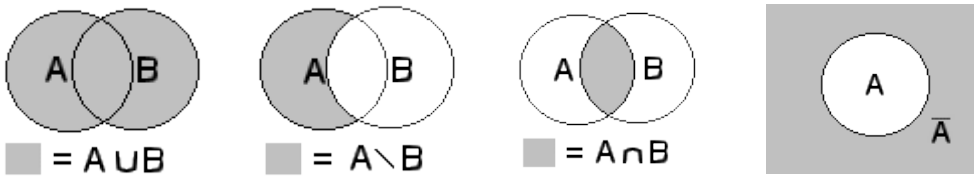


4. Constructive Solide Géométrique (CSG) :

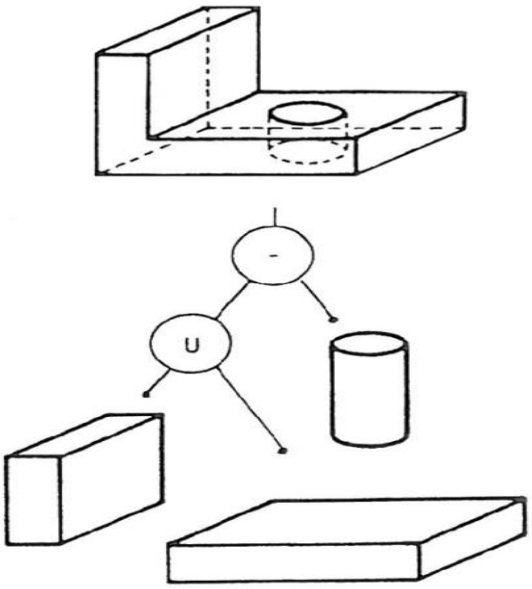
La méthode CSG appelée en anglais "Constructive Solid Geometry" est une technique de modélisation géométrique qui concerne la représentation d'un objet solide comme combinaison d'objets solides simples (ex: cylindre, sphère, cône, tore, etc.) à l'aide d'opérateurs géométriques booléens (ex: union, intersection, soustraction).

On utilise des arbres binaire , avec des objets à chaque feuille, et un operateur pour chaque nœud non terminal.

union **difference** **intersection** **complement**



Exemple de CGS :



La grammaire peut être décrit par :

<objet à modéliser > ::=< primitive> / <objet><opération><objet> /
<objet><transformation>
<opération> ::= union/intersection / différence
<transformation> ::= translation /rotation / changement d'échelle
<primitive> ::= cylindre / cube / polymédre /sphère / parallélépipède . . .

5 .Surface et courbe de Bézier :

a. Les courbes de Bézier :

Les courbes de Bézier ce sont en fait de simples courbes, polynomiales. Ce n'est pas un "nouveau type" de courbe mais simplement un moyen de représenter des courbes connues.

Les courbes polynomiales de degré 1 n'ont qu'un intérêt très limité puisque ce sont des droites.

Courbe polynomiale de degré 2 : $\mathbf{P(t) = t^2.P_2 + 2.t.(1-t).P_1 + (1-t)^2.P_0}$ avec t évoluant sur [0,1]

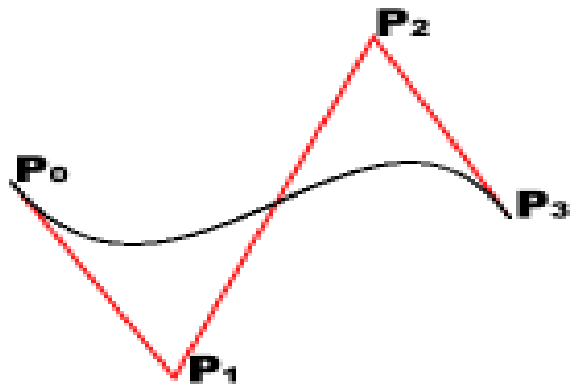


Figure2 : courbes de Bézier

b. Les surfaces de Bézier :

On peut définir une surface comme l'ensemble des points d'une courbe se déplaçant et se transformant simultanément (approche de Bézier).

A partir des courbes de Bézier il est possible d'engendrer de telles surfaces en combinant les expressions paramétriques des courbes dans une somme de combinaisons des facteurs des différentes expressions paramétriques des courbes. L'idée des surfaces de Bézier est strictement la même que pour les courbes de Bézier, la différence c'est qu'on a un degré de liberté supplémentaire.

$\mathbf{P(s,t) = \sum\sum B_{i,n}(s).B_{j,m}(t).P_{i,j}}$, s et t évoluant sur **R**, l'ensemble des réels. (On se restreindra à s et t variant sur l'intervalle [0,1])

Les $\mathbf{P_{i,j}}$définissent ce que l'on appelle une grille de contrôle (on les représente souvent sous forme d'une grille). L'allure locale de la surface est toujours déterminée par un point de contrôle.

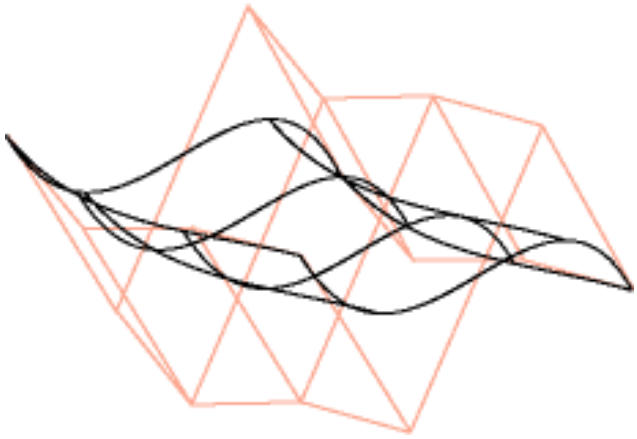


Figure3 : Surface de Bézier définie par une grille à 16 points de contrôle

6. Le Maillage :

Qu'est-ce qu'un maillage ? C'est le nom donné à tout objet modélisé dans un espace tridimensionnel. Le maillage est formé de plusieurs groupes d'éléments. Il est tout d'abord composé de points, appelés *vertices* dans le langage des infographistes. Mais il comprend aussi des arêtes et des faces. De façon générale, un logiciel d'infographie 3D enregistre les données concernant ces trois composants du maillage.

– Un vertices : C'est non seulement un point dans l'espace, mais c'est une unité à laquelle l'ordinateur associe des informations sur la position (du type p(x ; y ; z)), la couleur, etc.

– Une arête : C’est un segment qui relie deux vertices.

– Une face : Pour un logiciel d’infographie 3D, une face est un triangle composé de trois arêtes refermées et connectées par leurs sommets.

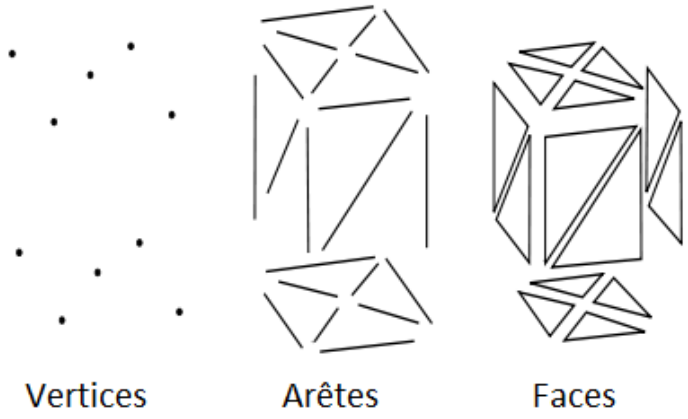


Figure4 : les mailles

chapitre 3:

VRML

1. Introduction :

Depuis quelques années, la recherche concernant l'interface homme-machine a suscitée beaucoup d'attention auprès de la presse et de l'industrie. Toute une technologie, connue sous l'appellation de "Réalité Virtuelle" est apparue. Cette technologie se donne pour but d'interagir avec les sens humains: la vue, l'ouïe, le toucher.

En 1993 Mark Pesce et Tony Parisi développèrent une interface 3D pour le web, laquelle incorporait une grande partie des recherches sur la réalité virtuelle et sur les réseaux (les liens hypertextes par exemple). Ils communiquèrent ces innovations à Tim Berners-Lee, l'inventeur du langage HTML, et Pesce fut invité à présenter une communication à la première conférence internationale sur le World Wide Web à Genève. Lors d'une réunion à propos des interfaces de réalité virtuelle les participants se mirent d'accord sur la nécessité de définir un langage commun permettant de décrire des scènes 3D et des hyperliens, tout comme le HTML, mais pour des applications de réalité virtuelle, est comme sa Le terme Virtual Reality Modeling Language (VRML) est conçut. Le VRML est venu de manière à satisfaire 3critéres : Indépendance de la plate-forme (Windows, MAC OS, UNIX etc..), Extensibilité, Travailler avec une faible bande passante.

Le VRML est aujourd'hui à sa version 2.

2. Langage VRML

VRML (Virtual **R**eality **M**arkup **L**anguage), est un simple langage de programmation pour décrire les mondes 3D virtuels interactif et multimédia,Un moyen de réaliser des applications graphiques interactives sur internet.

Un fichier source (texte) VRML, identifié par son extension : **.wrl**on peut créer le monde VRML a partir d'un éditeur de texte **VRMLPAD** et un visualisateur (VRML-viewer) pour voir notre objet comme **cortorona 3D** et **cosmos Player** et on peut visualiser notre monde sur une page HTML

2.1- A quoi ressemble le VRML ?

Un fichier VRML est un fichier qui contient la description d'une scène en mode texte.

Ce fichier possède l'extension *.wrl*. Le texte peut être lu avec un simple éditeur de texte (comme le bloc-notes de *Windows*). Pour visualiser ses graphismes en 3D, il faut des outils spécifiques.

Comme le VRML est utilisé dans le monde Internet, ces outils se présentent souvent sous forme de plug-in dans un navigateur Internet. Nous en avons essayé deux : ***Cortona VRML*** et ***Cosmoplayer***, et aussi le logiciel ***3D View***.en mode graphique (*Cortona VRML, 3D View*et*Cosmoplayer*) ou la structure de son code source (*VrmlPad*).

2.2. Structure d'un fichier VRML :

Un fichier VRML contient :

L'entête du fichier : **#VRML V2.0 utf8**

#VRML: Le fichier contient un texte VRML.

V2.0 : Le texte est conforme à la version 2.0 du langage.

utf8 : Le texte utilise les caractères UTF8 (est le nom d'un jeu de caractères international).

: Commentaires des commentaires et notes .

2.3- Syntaxe d'un programme VRML :

Nœuds : sont des paramètres de définition de la scène (objet), et on la définit comme suit : **Nom du nœud {…}**

Chaque nœud possède :

- Un type (Shape, cylinder, etc.).
- Une paire d'accolades.
- Zéro ou plus champs entre les accolades.
- Les noms des types ou de champs sont dépendants de la casse.
- Chaque mot commence par une majuscule, le reste est en minuscule.

Sont des paramètres qui définition les attributs des nœuds, Les champs sont optionnels, La valeur par défaut pour un champ est utilisée s'il est absent est ils peuvent être donné dans n'importe quel ordre. Chaque champ possède :

- Un nom (height, radius, etc.)
- Un type de données (float, integer, etc.)
- Une valeur par défaut
- Le premier mot commence par une minuscule

3. Les objets de base dans le VRML :

Les nœuds de type Shapes sont les briques de base d'un monde VRML, est Les formes primitives de ce nœud sont : Box, Cone, Cylinder, Sphere, Text.

3.1-Syntaxe de Shape :

Shape { appearance ….

geometry ….}

appearance : SFNode (pour définir la couleur et la texture de l'objet).

geometry : SFNode (Forme primitive ou structure).

Les champs des nœuds géométry spécifient les dimensions de la forme, Ces dimensions sont généralement en mètre.

3.2-Syntaxe d'un nœud Box

Un nœud **Box** définit un cube centré en (0 0 0) et parallèle aux axes : Shape { appearance Appearance {}

Geometry Box {size 2.0 2.0 2.0 }}

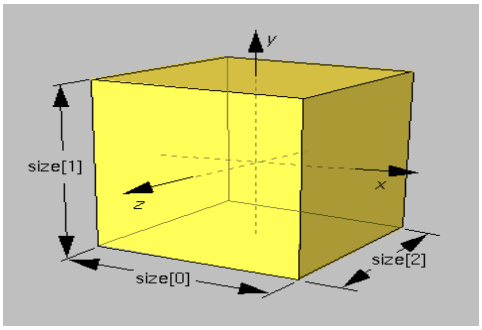
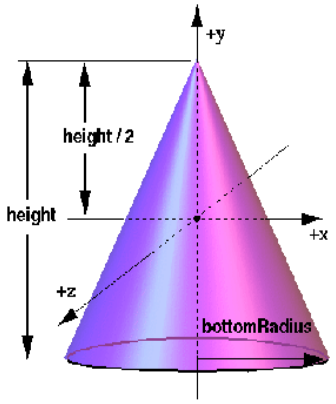


Figure1 : Geometry Box

Size : **SFVect3f** (Longueur, Hauteur, Profondeur)

3.3-Syntaxe d'un nœud Cône :

Un nœud Cone définit un cône orienté selon l'axe Y et centré en (0 0 0):




```
Shape {
      appearance Appearance {
                                    }
      geometry Cone {
height 2.0
bottomRadius 1.0
          bottom TRUE
          side TRUE
}
    }
```

Figure2 : Geometry cone

height et **bottomRadius** : SFFloat (hauteur et rayon de base)

bottom et **side** : SFBool (fermeture du cone)

3.4-Syntaxe d'un nœud cylinder :

Ce nœud définit un cylindre orienté selon l'axe Y et centré en (0 0 0):

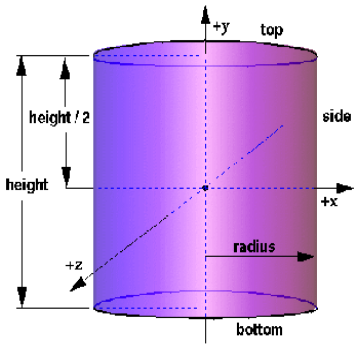


Figure3 : Geometry Cylindre

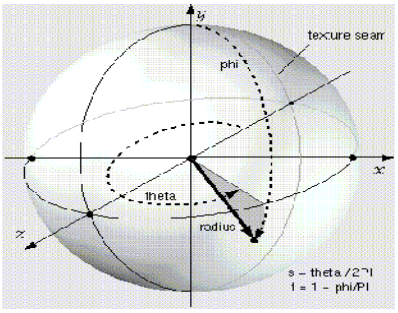
```
Shape {
    appearance Appearance {
      }
    geometry Cylinder {
     height 2.0
     radius 1.0
     bottom TRUE
      top TRUE
     side TRUE
          } }
```

height et **radius** : SFFloat (hauteur et rayon)

bottom, **top** et **side** : SFBool (fermeture du cylindre)

3.4-Syntaxe d'un nœud sphère:

Définit une sphère centrée en (0 0 0)




```
Shape {
 appearance Appearance    {
   }
   geometry Sphere {
radius 1.0
     }
  }
```

Figure4 : Geometry Sphere

Radius: SFFloat (rayon)

3.6-Syntaxe d'un nœud Text:

définit un texte dans le plan (XOY) centré en (0 0 0):


```
Shape {
      appearance Appearance {
}
geometry Text {
          string [ "Text",
               "Shape" ]
fontStyleFontStyle {
            style "BOLD" }
}}
```

string : MFString (texte à écrire)

fontStyle : SFNode (Définition du style du texte)

3.7 Syntaxe d'un nœud Appearance

Un nœud Appearance décrit le matériau (ou la texture) d'un objet par les attributs :

material : SFNode (Définition du matériau de l'objet).

Texture : SFNode (Définition de la texture de l'objet).

textureTransform: SFNode (Définition du placement de la texture sur l'objet).

3.7.1 Syntaxe d'un nœud Material

Contrôle la couleur de l'objet. La couleur définit un mélange de lumière rouge, verte et bleue. Les valeurs vont de 0.0 (pas de couleur) à 1.0 (toute la couleur)

diffuseColor : ***SFColor*** (Couleur diffuse)

emissiveColor : ***SFColor*** (Couleur émise)

transparency :***SFFloat*** (Transparence)

shininess :SFFloat (brillance)

ambientIntensity : SFFloat (Eclairage ambiant)

3.7.2 Texture

A partir d'un bloc **Appearance** on a un bloc spéciale pour les textures appelle « ImageTexture », on utilise une adresse a l'aide d'un champ « url » pour l'image texture soit image jpg ou bien image gif

4. Transformation des objets :

Maintenant que nous savons créer des objets et leur donner une apparence particulière, nous allons pouvoir les transformer et les positionner dans la scène.Les objets précédents peuvent être manipulés par les opérations géométriques usuelles de translation, rotation et homothétie. Il existe un nœud du langage permettant de procéder simultanément au positionnement (translation, rotation) et à la transformation (homothétie) des formes géométriques:

Transform {
center 0 0 0
translation 0 0 0
rotation 0 0 1 0
scale 1 1 1
scaleOrientation 0 0 1 0
children [... **]**
}

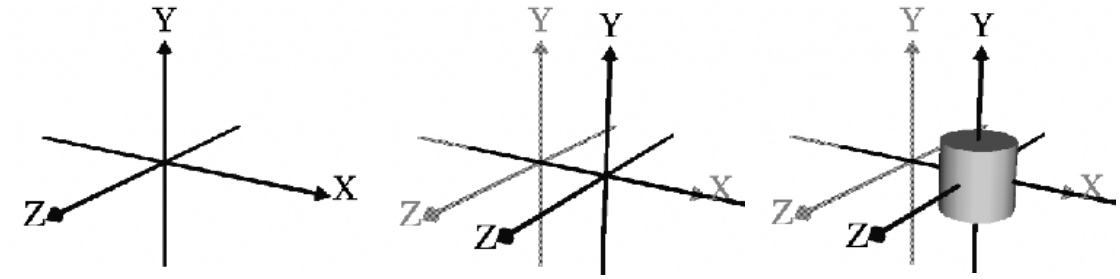
Ce nœud fait partie de ce qu'on appelle les «nœuds de groupe». Ce sont des nœuds qui ont la propriété de pouvoir contenir d'autres nœuds du langage dans leurs paramètres. Ici, le paramètre *children* contient d'autres nœuds (en particulier des *Shape*), auxquels les transformations géométriques seront appliquées.

- Les transformations géométriques effectuées par ce nœud sont les suivantes: *translation* déplace l'objet selon les axes X Y et Z.
- *rotation (x y z a)* tourne de a (en radians) autour du vecteur (x y z).
- *scale* (homothétie) modifie l'échelle selon les 3 axes du repère défini par *scaleOrientation* (cet axe fonctionne de la même manière que le paramètre *rotation*). *Center* permet de spécifier un centre pour l'homothétie.

Exemple de translation :

Transform {
 # X Y Z
translation 2.0 0.0 0.0
children[. . .]
}

Référentiel du monde Référentiel translaté Objet translaté



Exemple de rotation :

Transform {

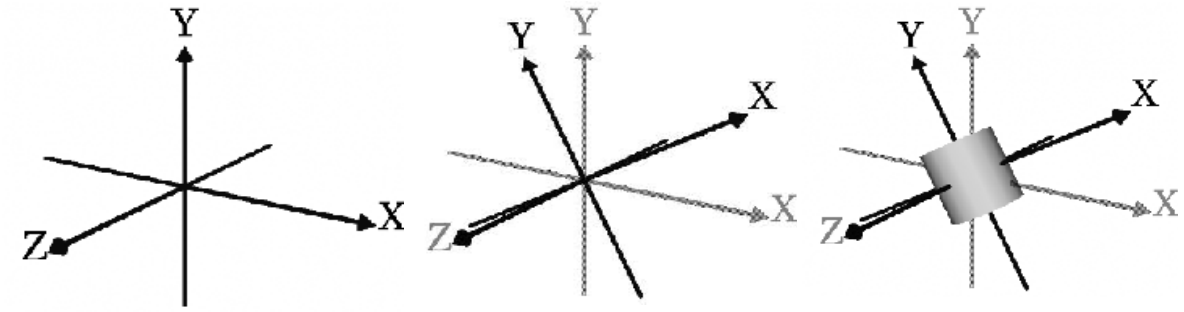
 # X Y Z Angle

rotation 0.0 0.0 1.0 0.52

children[. . .]

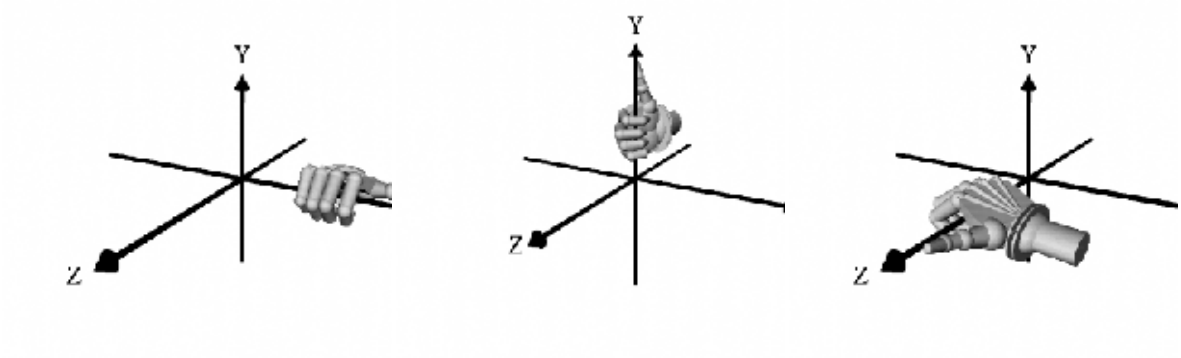
}

Référentiel du monde Référentiel orienté Objet orienté



Sens des rotations :

- Les rotations positives se font dans le sens trigonométrique
- Règle de la main droite pour la définition des rotations.
- Ouvrir la main
- Lever le pouce et l'orienter dans le sens positif de l'axe de rotation
- La direction d'enroulement des doigts est une rotation positive



Rotation autour de l'axe X Rotation autour de l'axe Y Rotation autour de l'axe Z

Mise à l'échelle :

```
Transform {
   #    X   Y   Z
scale 0.5 0.5 0.5
children[ . . . ]
}
```

Il est fortement conseillé au départ de ne pas combiner plusieurs transformations dans un même nœud *Transform*

Camera sur les objets :

- **Viewpoint** : permet de créer une caméra, de la positionner, de l'orienter etc.
- **fieldOfView :** Permet de modifier l'angle de vue (en radians)
- **orientation** : Permet d'orienter la caméra selon un vecteur et un angle
- **position :** Permet de positionner la caméra aux coordonnées x y z
- **description :** Label, apparaissant dans le menu de positionnement de certains visualisateurs.

5. Les formes complexes :

Les objets complexes peuvent être difficiles à obtenir par mélange de formes simples Terrains, Animaux, Plantes, Véhicules ...

Au lieu de construire ces objets à partir de formes primitives, on les construits à partir de formes atomiques : Points, Lignes, Faces.On a 5 types de forme complexes :

- Un noeud**PointSet** définit un nuage de points dans R3 :
  coord : SFNode(Noeud de définition des coordonnées) Un point est placé à chaque coordonnée.

- Un noeud**IndexedLineSet**définit un ensemble de polylignes construites sur des points :
  coord : SFNode(Noeud de définition des coordonnées des points)
  coordIndex : MFInt32 (Noeud de définition des polylignes).
- Un noeud**IndexedFaceSet** définit un objet à partir d'un ensemble de polygones:

coord : SFNode(Noeud de définition des coordonnées des points)
coordIndex : MFInt32 (Noeud de définition des polygones)
solid : SFBool (Indicateur de fermeture de l'objet)
ccw : SFBool (Indicateur d'ordre des sommets des polygones)
convex : SFBool (Indicateur de convexité des polygones)

5.1- Les nœuds spéciaux :

Pour l'éclairage de la scène :

- **DirectionalLight** :permet de créer une source lumineuse directionnelle, non localisée.
- **PointLight :** permet de créer une source lumineuse localisée. Cette source est omnidirectionnelle.
- **SpotLight :**permet de créer une source lumineuse localisée et directionnelle de type spot.
- **Background :**permet de créer un arrière plan sphérique pour le ciel et le sol, et / ou de créer un panorama cubique englobant la scène, dont chacune des faces du cube peut recevoir une image.
- **Fog :** permet de créer un brouillard recouvrant la scène. Il permet donc de limiter la vision d'objets lointains ou / et de créer un effet dramatique. L'utilisation de l'option EXPONENTIAL donne un résultat plus réaliste.

2.5. Multimédia :

Sound : permet de jouer un son. Ce son de type AudioClip (MIDI ou WAVE) ou du type MovieTexture (MPEG avec son).

AudioClip :permet de paramétrer le son d'un bloc Sound.

3.5. Modèles prédéfinis et liens :

- **Anchor :** permet de lier une adresse aux blocs inclus. Cette adresse peut désigner une autre scène externe ou une page HTML par exemple.
- **DEF** permet de nommer un bloc. L'instruction **USE** permet d'utiliser un nom de bloc en lieu et place de sa définition complète.
- **PROTO** :permet de définir un prototype d'objet paramétrable . On peut ainsi utiliser un objet en le désignant par son nom, et en lui fournissant les paramètres nécessaires.
- **EXTERNPROTO :** permet d'utiliser un prototype dont la définition se trouve dans un fichier externe.
- A la différence de PROTO, il ne faut pas spécifier de valeur par défaut, elles doivent être déclarées dans le prototype.
- **Inline :** permet d'inclure dans une scène une autre scène ou un objet dont la description se trouve dans un second fichier dont on fournit l'adresse.
- **Group :**permet de regrouper des objets.
- **ROUTE ... TO :**permet de transmettre la valeur d'un champ de type **eventOut** d'un objet à un champ **eventIn** d'un autre (ou du même) objet. Pour pouvoir effectuer cette opération, il est nécessaire d'avoir auparavant nommé l'objet émetteur et l'objet récepteur à l'aide de l'instruction DEF.

6. Conclusion :

Dans ce chapitre, nous avons présenté une modélisation d'un environnement virtuel VRML et la syntaxe d'un programme écrit dans ce langage. Langage VRML est un langage puissant qui permettre décrire une réalité virtuelle avec une grande richesse d'objets.

Chapitre 4:

Réalisation

1. Introduction :

L'apprentissage de la navigation est une étape incontournable au bon fonctionnement d'une application 3D. Si les viewers 3D proposent des interfaces de navigation différentes de l'un à l'autre, la plupart des fonctions sont communes et répondent aux principes propres à la 3D. Deux fonctions sont nécessaires pour naviguer dans une maquette virtuelle, la rotation et la translation. Généralement, ces deux fonctions sont prés-combinées en variantes pour permettre une navigation plus directe. Mon projet consiste à développer des déférentes scènes en 3D pour la modélisation et la réalisation virtuel d'un robot SCARA et pour cela j'ai utilisé le langage VRML 97 (**V**irtual **R**eality **M**odeling**L**angage).

Pendant la programmation j'ai utilisé l'éditeur VrmlPad et pour la visualisation j'ai choisi VRML Viewer qui contient les fonctions de navigation tel que les outils de Cosmos Player et Cortona .

2. Description de l'éditeur VRML :

➢ **VrmlPad :** est un éditeur professionnel pour la programmation VRML. Les fonctions clés comprennent de puissantes capacités d'édition ainsi qu'un support visuel pour l'arborescence des scènes et pour les opérations de ressource.

VrmlPad permet l'édition des fichiers situés à la fois localement et à distance. Les fonctions sont les suivantes :

- Détection dynamique des erreurs.
- Mise en valeur de la syntaxe.
- Support visuel pour l'arborescence des scènes.
- Opérations sur les ressources.
- Aperçu avant impression.
- Publication.

Voici la figure (3.1) qui représente l'interface de VrmlPad avec ses outils :

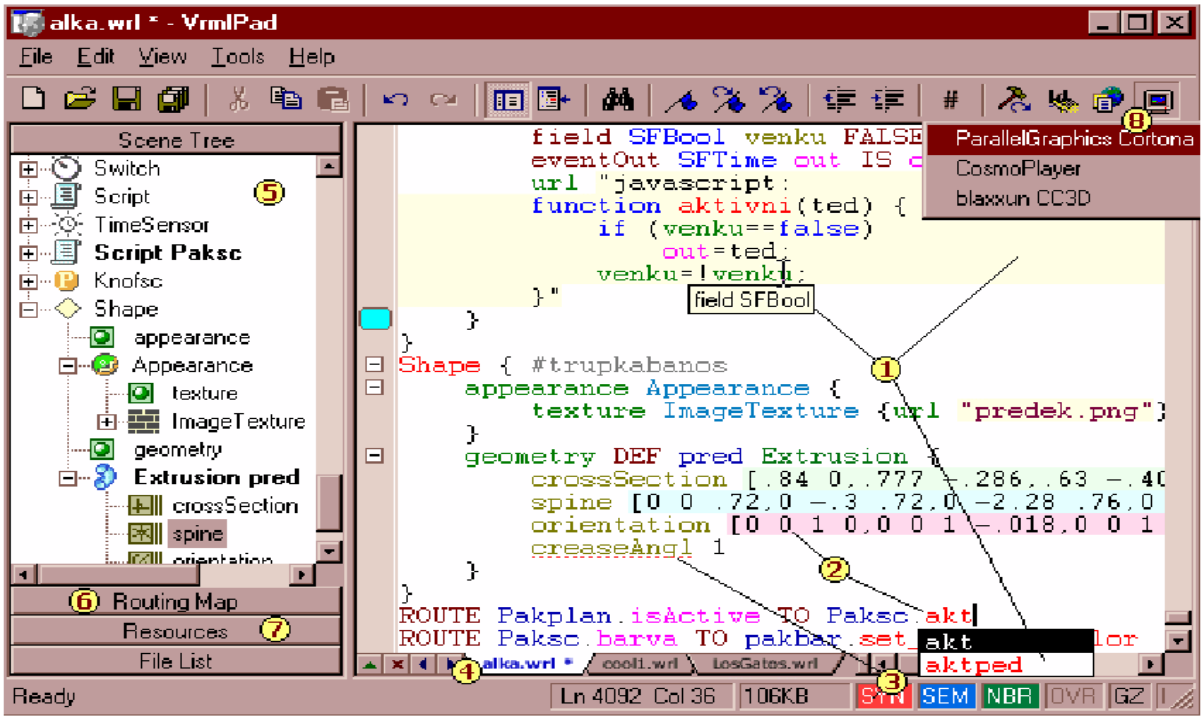


Figure1 : VrmlPad

Pour visualiser les programmes qui sont écris en VrmlPad on utilise VrmlViewer qui contient deux éditeurs de navigation sont :

- **Cortona vrmlviewer :** Dans cortona tous les outils de navigation sont directement accessibles par leurs icônes situés à gauche et en bas de la fenêtre 3D. A gauche les outils de déplacement et d'orientation, en bas les outils de positionnement

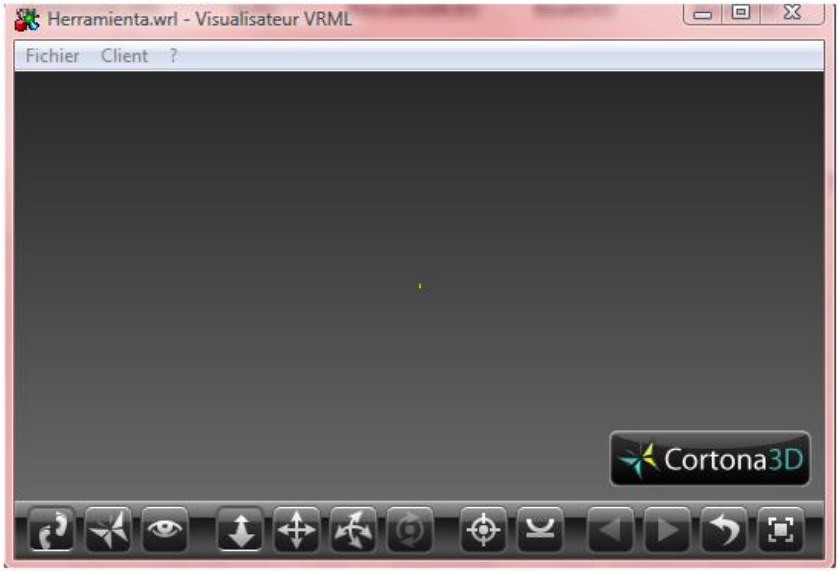


Figure2 : Cortona vrmlviewer

2. Description des outils de navigation :

Les outils de déplacements et d'orientation : Les outils de déplacement se situent en colonne à gauche, ils sont séparés en 2 groupes et fonctionnent de manière combinée. Il faut sélectionner d'abord un mode maître puis ensuite un mode secondaire. Le groupe supérieur sert à sélectionner un mode de déplacement "Maître" (WALK, FLY et STUDY) Le groupe inférieur à sélectionner un mode de déplacement secondaire. (Plan, Pan, Turn et Roll).

- **Cosmo Player** : Cosmo Player s'intègre à votre navigateur Web pour vous permettre de découvrir et d'explorer les mondes 3D. Cosmo Player vous donne accès à tous les mondes 3D créés en langage VRML. Ces mondes utilisent souvent d'autres supports, tels que le son ou la vidéo. Les principales fonctions de Cosmo Player sont décrites brièvement ci-dessous :

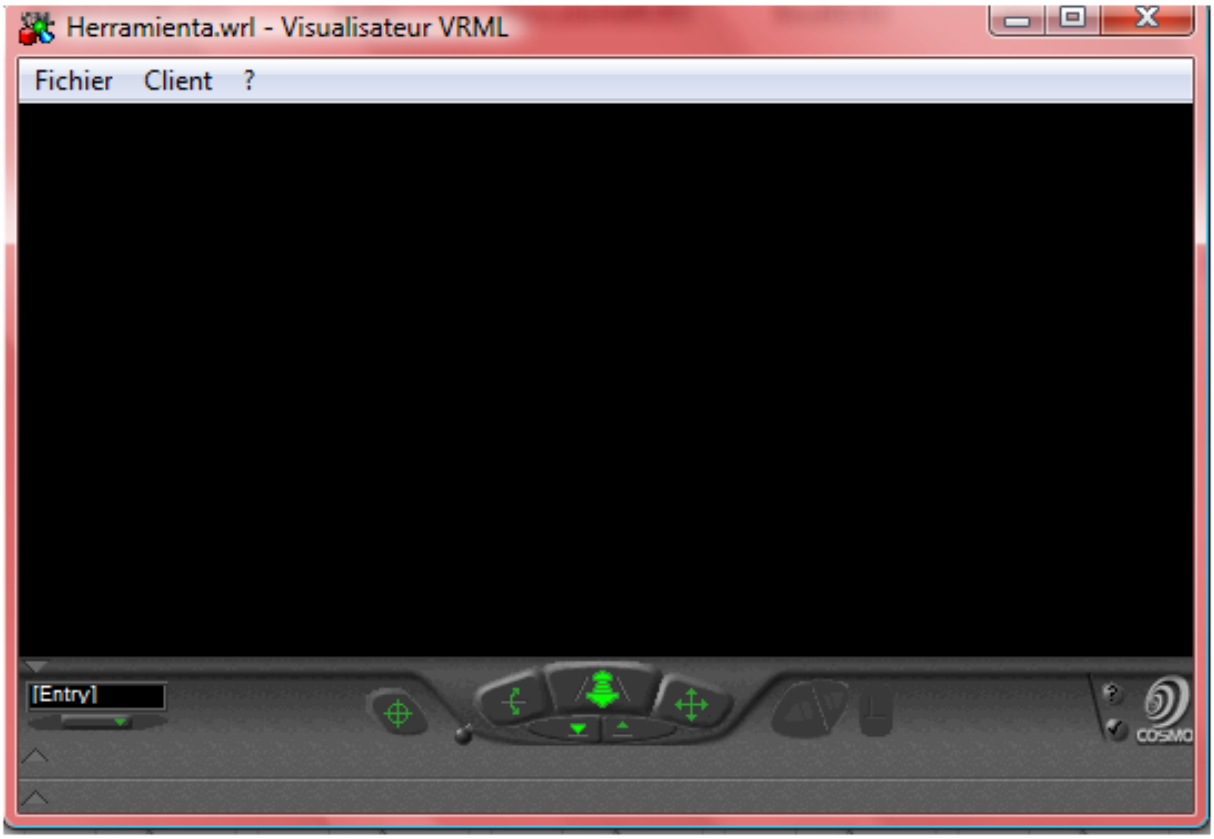
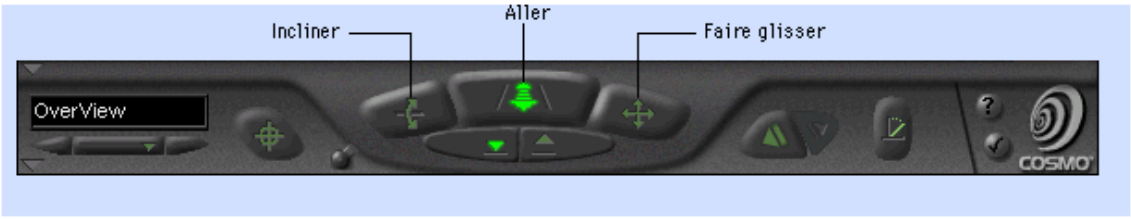


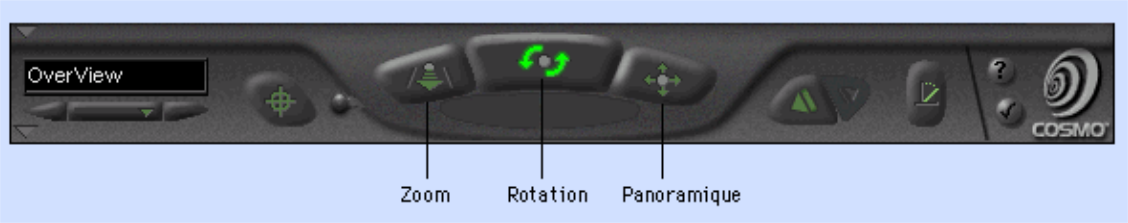
Figure2 : Cosmo Player

Le tableau de bord : Les principaux boutons du tableau de bord de Cosmo Player vous permettent de vous déplacer dans les mondes 3D et d'observer leurs objets. Certains mondes ne proposent pas de tableau de bord, mais peuvent afficher des instructions de navigation.

Le tableau de bord se présente comme suit :

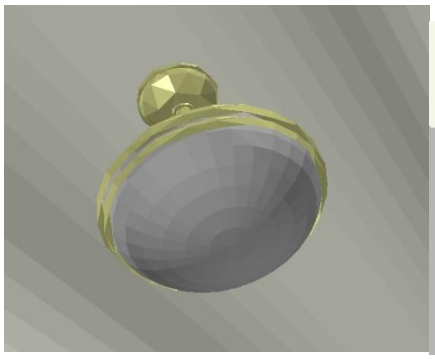


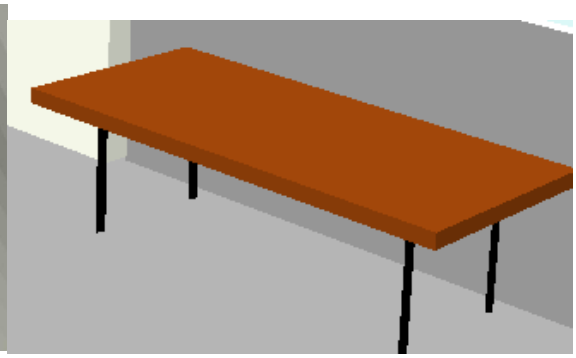
Ce tableau est utilisé pour Aller, Faire glisser et Incliner dans le monde .et pour faire La Rotation, Panoramique et Zoom on utilise le tableau de bord suivant :



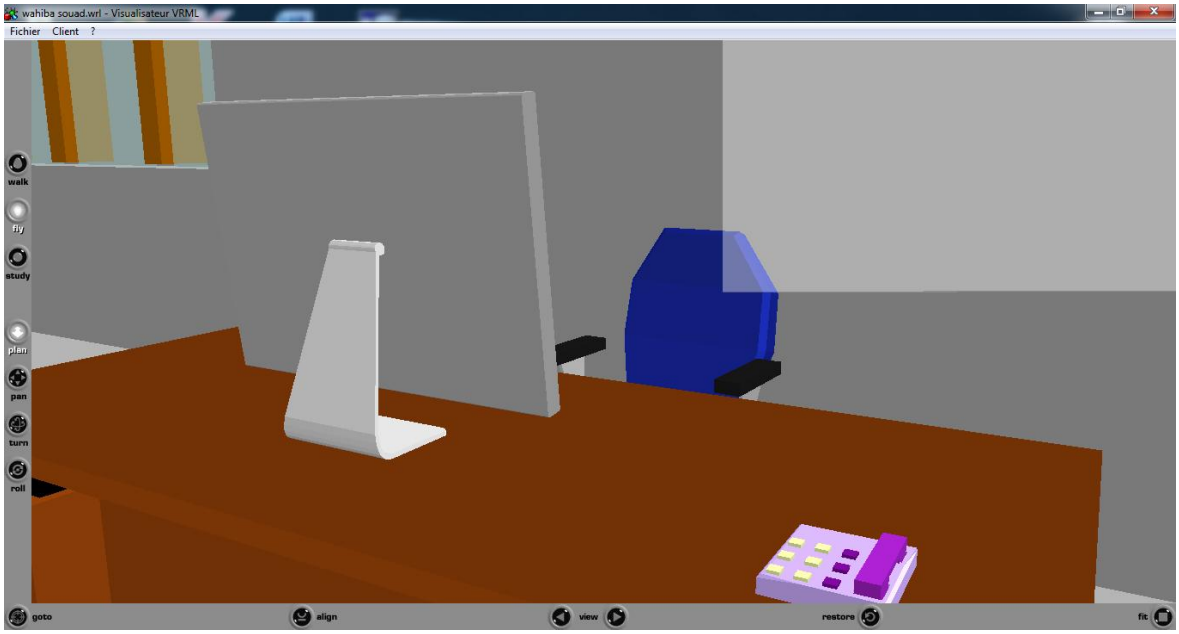
4. Description de l'application

Le but du projet qui nous a été confié était d'effectuer la visite virtuelle dans une bibliothèque en utilisant le langage VRML (y compris l'animation de certains objets …).
L'objectif que nous nous sommes fixé était de définir une bibliothèque rectangulaire qui compose de quatre salles(2bureaux et 2salle de lecture) rectangulaire avec ses murs, ses ouvertures (portes et fenêtres) et bien entendu tous ses meubles(tables, chaises, bureaux,..)et des escaliers à l'entrer .





L'entrée de la bibliothèque



Bureaux d'inscription

Salle de lecture

5.Logiciel exportant en VRML :

Il existe une multitude de logiciels de modélisation 3D. En voici quelques-uns :

- Archicad
- SketchUp
- Blender
- Abaqus
- GPure
- TopStation de JSInfo
- 3ds Max
- Maya
- Autocad . . .

Il existe aussi des logiciels que permet de convertir des projets réaliser par un des logiciels précédant au VRML comme PolyTrans.

Conclusion :

Notre projet consiste à modéliser une bibliothèque par le langage VRML. Ce langage permet une description des scènes et des objets 3D. La richesse du langage VRML en 3D nous a réduit la tache de la conception virtuelle de notre projet. Comme perspective de notre travail nous souhaitons pour les prochains projets licence de suivre notre travail en réalisons une conception virtuelle par OPENGL.

Conclusion Générale

L'importance majeure donnée au langage de programmation virtuelle VRML dans notre mémoire revient à leur puissance et sa simplicité, plus sa richesse dans le développement des scènes de 3 dimensions. VRML est un langage descriptif orienté objet possède tous les outils et les fonctions nécessaires pour amener à construire des scènes compliquées riche avec animation et mouvement. Notre objectif visé dans cette étude de ce langage est de le maîtrisé et en même temps ouvrir notre esprit sur le monde virtuelle est sa beauté pour exprimer la vie ou le monde réel avec la 3D.

De plus, nous avons pu voir un nouvel aspect de la programmation que nous ne connaissons pas.

Comme perspective de notre travail nous souhaitons pour les prochains projets licence de suivre notre travail en réalisons une conception virtuelle par OPENGL.

Bibliographie et Webographie :

- Hachmaoui Sidi Mohammed (2010). *Navigation et visualisation 2D en fonction d'un SIG*.
- http://www.anuelle de référence VRML .com.
- Djamel Boursali et Sidi Mohammed Khenafou. Réalisation d'un système de navigation dans une scène tridimensionnelle (2009).
- http://www. *The VRML 2.0 Sourcebook.html*
- AlexandreTOPOL (2001).*VRML: étude,miseenœuvreet applications.*
- http ://fr.wikipedia.org/wiki/Infographie
- http://www.les espace de couleur.html
- http://www.Courbes et surfaces de BezierLeGregHomepage - Gregory Massal