Sommaire

Introduction General

Contexte

La réalité virtuelle est une expression qui signifie un terme de présence d'un monde parallèle au monde réel ce qui est le monde virtuel tridimensionnel, elle représente le nouveau facteur de la civilisation de notre monde puisqu'elle est utilisée dans plusieurs domaines telle que l'informatique graphique, la simulation, la télé-opération, la conception assistée par ordinateur, l'audiovisuel ou le travail collaboratif. Et depuis la naissance de ce terme par Jaron Lanier en 1989, les chercheurs s'intéressaient a désigner un espace de représentation réaliste, tridimensionnel, calculé en temps réel et immersif.

La réalité virtuelle est une partie de traitement des Images de synthèse qui ont généralement deux utilisations :

3D précalculé :

Pour la conception des images fixes, des films d'animation et des effets spéciaux on utilise des logiciels spéciaux comme Softimage ou 3DS Max. mais dans ce procédé utilisation on a besoin de cartes graphiques très puissantes avec un temps de traitement d'une Image qui est beaucoup plus longue parce qu'on utilise l'algorithme de radiosité (Raytracer) pour le rendue. Et pour créer un film d'animation on a besoin d'une vingtaine d'images par seconde avec 5 milliards de rayons lancés pour calculer l'éclairage de chaque image. Cela prend beaucoup de temps tel qu'on ne peut pas l'appliquer en temps réel.

3D Temps réel :

Pour la création de La réalité virtuelle, on utilise des API standard comme DirectX, OpenGL et XNA Game Studio. Pour la production, et la réalisation 3D en temps réel on à besoin de calculer le nombre des Images par seconde (FPS) qui sont au moins une vingtaine. Avec ce nombre cela dépend de la machine mais il y a un inconvénient celui de sacrifier la production des Images réaliste. Nous constaterons que le calcul serait rapide car nous utilisons la méthode de Rastérisation qui permet de dessiner les polygones 3d. Et on trouve dans cette méthode les techniques de rendue comme le Mip-mapping et le Bump-mapping.

Problématique :

Jusqu'à maintenant il y a beaucoup de problèmes pour la réalisation d'un monde virtuel plus réaliste avec la même qualité de l'image de synthèse en 3D précalculé, et c'est pour cela que les chercheurs améliorent chaque année des techniques pour faciliter la tâche de la création. Pour réaliser un exemple d'un monde virtuel, on a besoin d'un ensemble d'applications pour faire :

- **Modélisation des objets 3D :** on utilise 3DS Max Studio, Maya, Softimage …etc. Pour construire les Objets 3D à partir des formes standards comme le cube, la sphère et aussi le cône! on utilise des techniques plus évoluées pour la représentation des Objets 3D comme les courbes de Bézier.
- **Squelettisation et Animation :** on utilise des techniques de création des bonnes, qui sont intégrées dans les applications de modélisation pour définir les Squelette des Objets 3D. Afin de faire l'animation de ces bonnes une par une on applique les transformations géométriques sur elles comme la translation et la rotation.
- **Dessiner la texture :** on utilise Photoshop, CorelDraw, Gimp pour dessiner et colorer la texture.
- **Placage de texture :** on utilise des moteur de placage soit Mip-Maping ou de bump maping des textures comme Mental Ray qui est intégré dans les logiciel de modélisation de la communauté Autodesk c'est-à-dire dans Maya et Softimage et il y a un autre procédé comme Bryce Daz Studio.
- **Illumination de la scène :** les moteurs de placage de la texture ont le rôle de lancer la lumière dans la scène avec tous ses différents types.
- **Rendue :** On applique l'algorithme de rendue pour réaliser l'affichage sur l'écran et pour éliminer les parties cachées. Et il y a d'autres algorithmes comme le Ray-Tracing et Peintre.

Le but de ce travail est de réaliser et de définir un monde virtuel en temps réel. Nous essayerons de comprendre la méthode Rastérisation et toutes ses techniques et ses étapes à partir de la modélisation de la scène et des objets 3D appartenant à la scène. Nous devons utiliser une des technique de modélisation comme le modèle volumique ou le modèle par frontière, passant par l'animation avec ces deux techniques : soit en appliquant les transformations géométriques sur les objets ou bien sur les bones (OS) de squelette de ces objets ou en appliquant la technique de frame par frame en utilisant les Key frame et on

termine par le placage des textures avec les techniques de Mip-Mapping et le Bump-Mapping avec l'application des différents types de l'ombrage. A la fin nous utiliserons l'algorithme de rendu Z_Buffer pour l'affichage sur l'écran et l'élimination des parties cachées.

Plan des Chapitres :

Le premier chapitre concerne la réalité virtuelle qui comprend l'interaction 3D avec ses techniques (Navigation, sélection, Manipulation et Menu de Contrôle).

Le deuxième chapitre concerne les transformations géométriques et Les représentations des Objets 3D. Dans ce chapitre on étudiera les transformations géométrique et nous devons prendre en considération les coordonnées homogènes puis les techniques de modélisation des Objets 3D et a la fin comment animer un objet 3D.

Le troisième chapitre concerne le rendu réaliste. Dans ce chapitre on verra toutes les techniques dont on a besoin pour faire un rendu en tempe réel a partir de placage des textures, de filtrage des textures, l'éclairage et le Z-Buffer.

Le Quatrième chapitre sera consacré a la conception et l'implémentation de notre système.

CHAPITRE 1: **Réalité Virtuelle**

1.1 Introduction :

RÉSUMÉ

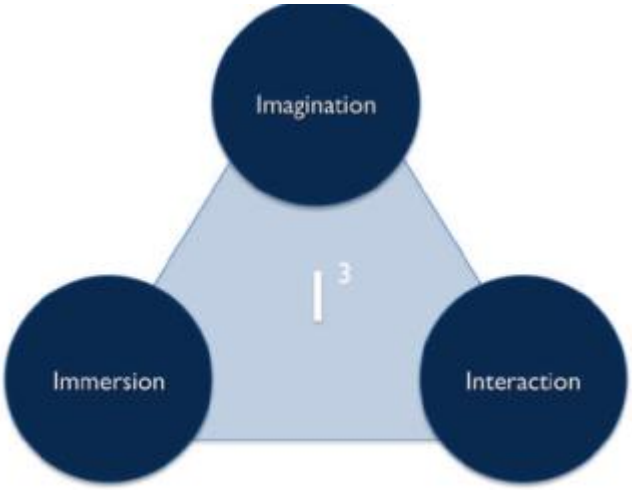
L'expression « *Virtual reality »était* proposée en juillet 1989 lors d'un salon professionnel *San Francisco (USA),* par Jaron Lanier, alors responsable de la société VPL Research spécialisée dans les périphériques d'immersion. Depuis l'apparition de la technologie de la réalité virtuelle, les chercheurs se sont intéressés particulièrement à l'interaction 3D qui peut être considérée comme la composante motrice de tout système interactif. En effet, l'interaction donne une meilleure sensation d'immersion et un sentiment d'être réellement dans l'univers virtuel. Dans ce chapitre, nous proposons une revue des techniques d'interaction 3D utilisées par la communauté de réalité virtuelle. Nous présenterons par la suite un bilan sur l'état actuel de la recherche dans le domaine de l'interaction 3D ainsi qu'une nouvelle voie à explorer pour interagir facilement et efficacement avec des environnements complexes.

1.2 Définition :

1.2.1 La réalité virtuelle

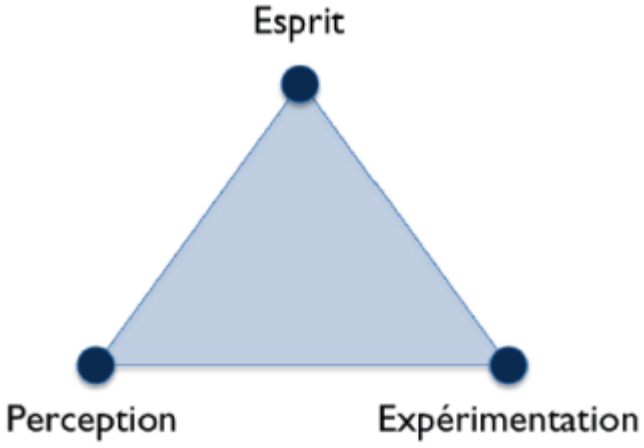
La RV est une technologie qui se situe a la croisée de plusieurs domaines comme l'informatique, la simulation, la CAO ou encore la telérobotique. Elle repose sur l'utilisation de différents dispositifs matériels hétérogènes ainsi que des techniques logicielles. La RV dispose de nombreuses définitions dues a l'ambiguïté du terme virtuel.

[Burdea and Coiffet, 1993]. Ils ont introduit trois composantes a prendre en compte pour permettre `a un utilisateur d'interagir en temps réel dans l'environnement



I.1- Les trois I [Burdea and coiffet, 1993]

[Tisseau, 2001] définit la Réalité Virtuelle comme un univers de modèles qui propose la triple médiation des sens, de l'action et de l'esprit. Cette médiation des sens permet la perception du réel, la médiation des actions permet de mener des expérimentations alors que la médiation de l'esprit permet une représentation mentale de la réalité.



I.2- Les mediations du reel [Tisseau, 2001]

[Fuchs et al., 2006b] ont propose une définition qui englobe toutes les définitions. Pour cela, ils se basent sur la finalité de la RV:

- La finalité de la RV est de permettre a une ou plusieurs personnes, une activité sensori-motrice et cognitive dans un monde artificiel, crée numériquement, qui peut ˆêtre imaginaire, symbolique ou une simulation de certains aspects du monde reel.

1.2.2 Environnement Virtuel mono et multi utilisateurs

[Heim, 1995]. L'environnement virtuel est le lieu qui accueille un ou plusieurs utilisateurs afin de leur permettre de réaliser des taches, en leur donnant la sensation d'être dans un nouveau lieu.

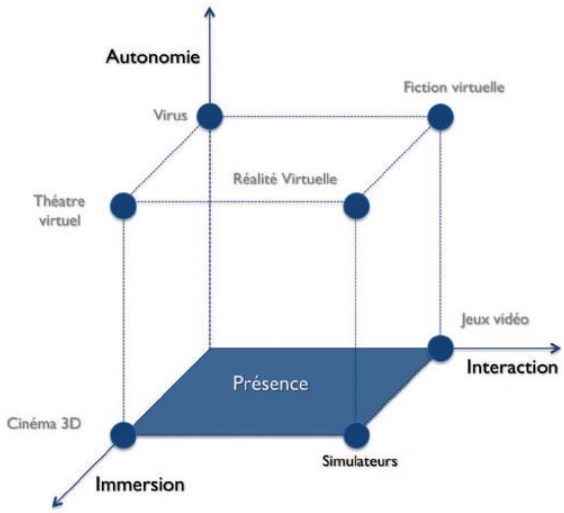
[Kalawsky, 1993] : On distingue différents types d'environnement virtuels en fonction du degré d'immersion que l'on souhaite donner

- Environnement non immersif, semi immersif et immersif.

Un Environnement Virtuel Collaboratif (EVC) est un environnement virtuel qui permet D'accueillir plusieurs utilisateurs et d'interagir avec des objets ou bien entre utilisateurs. Ils sont le résultat de la convergence de deux communautés : la Réalité Virtuelle et le Travail Collaboratif.

1.3 Les Composantes De LA Réalité Virtuelle :

- Zeltzer, Burdea et Coiffet: Ont définie trois composantes de base pour la Réalité virtuelle qui sont en correspondance avec les objectifs de la plupart des travaux dans ce domaine. Ces composantes sont : [6][25]
  - ***Immersion***
  - ***Autonomie***
  - ***Interaction***.



I.3- Composante de réalité virtuelle

1.3.1 Immersion :

- Bowman [4] : L'immersion est la sensation d'être présent, qui est assurée par certains environnements virtuels. Selon cet auteur, un utilisateur est dit " immergé " lorsqu'il sent que le monde virtuel qui l'entoure a remplacé le monde physique avec un certain degré. La présence d'un utilisateur dans un monde virtuel est un autre facteur qui joue un rôle important pour une meilleure sensation d'immersion.

  La présence procure à l'utilisateur un sentiment d'être " à l'intérieur " de l'environnement virtuel.

1.3.2 Autonomie :

L'autonomie est liée aux différentes composantes de l'environnement virtuel. L'utilisateur fait partie de ces composantes et il est considéré comme l'entité la plus active de cet espace .

L'utilisateur doit intégrer un modèle d'utilisateur *(Avatar)* dans son système pour que l'utilisateur puisse effectivement être pris en compte et participer à l'évolution de cet univers de modèles.***[I.28]***

1.3.3 Interaction :

Le plus important dans la Réalité Virtuelle est l'interaction 3D qui peut être considérée comme la composante motrice de tout système interactif. L'interaction est définie comme un langage de communication entre l'homme et la machine qui correspond à l'ensemble des (actions/réactions) réciproques entre l'homme et l'ordinateur par l'intermédiaire d'interfaces sensorielles, d'interfaces motrices et de techniques d'interactions .

1.4 CLASSIFICATION DES TECHNIQUES D'INTERACTION 3D :

2.2.1 Interface homme-machine

Les interfaces homme-machine permettent `a un ou plusieurs hommes de communiquer avec la machine de façon transparente et naturelle. Cela permet notamment de concevoir des applications de simulation extrêmement réalistes.

2.2.2 Métaphore, Paradigme et Technique d'interaction

Dans un environnement virtuel, les utilisateurs interagissent avec les objets de manière analogue au monde réel. En effet, nous interagissons avec différents objets pour effectuer des actions comme se nourrir ou s'amuser.

- La métaphore : d'interaction signifie qu'un objet ou concept réel est utilise comme un outil virtuel pour interagir avec l'environnement virtuel [Sternberger et al., 2008].
- Le paradigme :
  - une représentation du monde, une manière de voir les choses qui repose sur une base définie. On parle alors de paradigme d'interaction pour designer un ensemble de règles et de techniques permettant a l'utilisateur d'accomplir des taches d'interaction au sein d'un environnement virtuel [Bowman et al., 2001b]
  - Dans les interfaces homme-machine 2D, un paradigme d'interaction couramment utilise est WIMP (Windows Icons Menu Pointing devices). [Poupyrev et al., 1998].
- Les techniques d'interaction homme-machine : [Bowman et al., 2001b]. Les techniques d'interaction doivent faire le lien entre l'action humaine (par l'intermédiaire de l'interface) et l'action dans le système.

La réalité virtuelle est la traduction des actions des utilisateurs dans le monde réel en des tâches spécifiques dans l'espace virtuel. Les chercheurs essayent chaque fois de reproduire dans un environnement virtuel des gestes identiques à ceux de la vie quotidienne.

- Hand [[26]] : introduit les bases de la classification moderne. Il classe les différentes techniques d'interaction selon quatre tâches principales d'interaction 3D :
  - La navigation.
  - La sélection.
  - La manipulation
  - Menu de contrôle.

1.4.1 Navigation :

La navigation désigne l'ensemble des méthodes qui permettent de Connaître la position d'un objet par rapport à :

- *un système de référence*.
- *un point fixe déterminé*.

- Bowman et associés[5] : définissent deux composantes principales pour la navigation :

1) *Le déplacement :* représente la composante motrice de la navigation. Il se rapporte aux déplacements physiques de l'utilisateur d'un endroit à un autre.
2) *La recherche d'itinéraire :* correspond à la composante cognitive de la navigation. Elle permet aux utilisateurs de se repérer dans l'environnement et de choisir une trajectoire pour se déplacer.

1.4.2 Sélection : « sélectionner un objet »

Pour pouvoir manipuler un objet qui existe dans un environnement virtuel avec d'autres objets il faut d'abord le sélectionner .Cette tâche représente :

- La désignation d'un objet ou d'un ensemble d'objets .
- La validation de sélection est l'action qui suit la tâche de désignation.

1.4.3 Manipulation :

L'utilisateur doit être un acteur capable de changer les propriétés de l'environnement virtuel par exemple (la position, l'orientation, la couleur, l'échelle et la texture… etc.). Elle représente la composante active de tout système interactif.

1.5 Classification des techniques d'interaction pour la sélection et la manipulation :

Il existe deux classifications de base des techniques de sélection et de manipulation :

1.5.1 ***Classification par décomposition en tâche :***

Les techniques de sélection et de manipulation se composent de plusieurs blocs. Chaque bloc se charge d'exécuter une action élémentaire pendant le processus d'interaction.

1.5.2 ***Classification par métaphore :***

Les techniques de sélection et de manipulation sont divisées en deux grandes familles en fonction de la position et de la distance utilisateurs-objets virtuels , ce sont :

- Les techniques *exocentriques* pour lesquelles le monde virtuel est contrôlé depuis l'extérieur.
- Les techniques *égocentriques* pour lesquelles l'utilisateur agit depuis l'intérieur de l'environnement virtuel. Cette dernière catégorie est composée de deux sous-familles : basées sur la métaphore de la *main virtuelle et pointeur virtuel*.

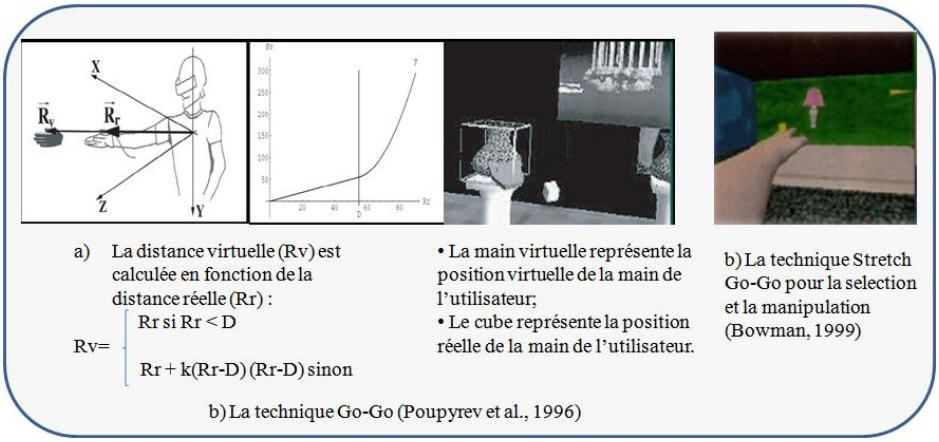
1.6 Les techniques de sélection et de manipulation :

1.6.1 Les techniques egocentriques :

L'utilisateur peut utiliser sa propre main pour sélectionner un objet virtuel. Alors :

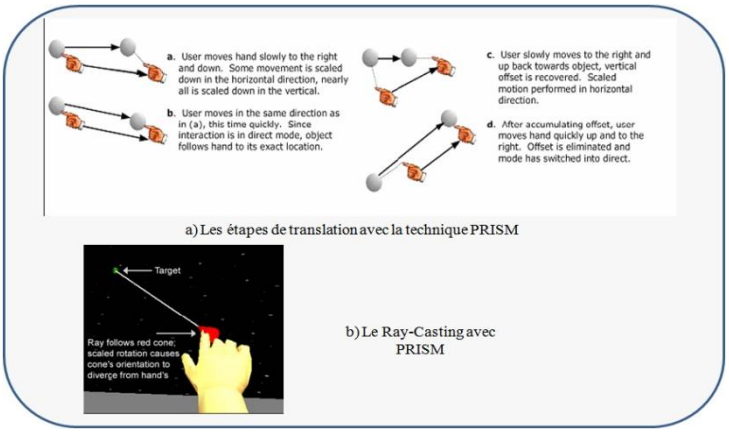
Sturman et associés [22] : Une technique de sélection basée sur la métaphore de la main virtuelle. L'utilisateur touche l'objet virtuel avec sa main réelle pour le désigner, puis valide la sélection.

Poupyrev et associés [18] : La technique du Go-Go, appelée également technique d'extension du bras. Elle est basée également sur la métaphore de la main virtuelle.



I.4- Technique : GoGo et ses variantes.

Frees et associés [9] : Ont proposé une technique de sélection et manipulation directe qui peut être utilisée comme complément aux techniques d'interaction manuelles telles que le Ray-casting afin d'améliorer la précision des mouvements de la main de l'utilisateur.



a) Les étapes de translation avec la technique PRISM

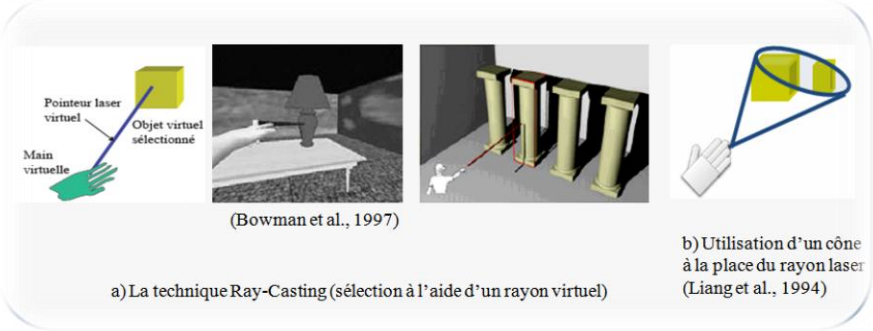
b) Le Ray-Casting avec PRISM

I.5- Technique : PRISM

Bolt en 1980 [2] : a Introduit Une des premières techniques imaginées pour l'interaction avec des mondes virtuels qui est le Ray-Casting.

Le Ray-Casting : est une technique de pointage basée sur la métaphore du rayon virtuel. Un rayon laser infini part de la main virtuelle et traverse tout le monde virtuel. Le premier objet intersecté dans le monde sera prêt à être sélectionné.

Pendant le processus de sélection, l'utilisateur peut rencontrer des obstacles qui cachent les objets qu'il veut sélectionner.



(Bowman et al., 1997)

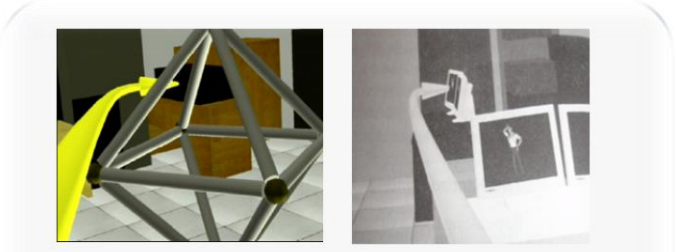
a) La technique Ray-Casting (sélection à l'aide d'un rayon virtuel)

b) Utilisation d'un cône à la place du rayon laser (Liang et al., 1994)

I.6- Techniques : la métaphore du pointeur Virtuel

Pour éviter cette difficulté :

1) Olwal et associés [15] : ont proposé la technique du pointeur flexible, qui est une extension du rayon virtuel avec la possibilité de pointer plus facilement des objets cachés par d'autres objets de l'environnement Si on dirige le rayon dans l'espace virtuel,
2) *D'autres chercheurs:* préfèrent éliminer des objets non désirables, où les objets que l'utilisateur ne veut pas sélectionner. Pour cela l'utilisateur tient en main une lampe de poche virtuelle qui va éclairer certains objets



Le pointeur flexible permet de sélectionner des objets complétement ou partiellement cachés par d'autres objets (Olwal and Feiner 2003)

I.7- Technique : rayon flexible

Techniques de sélection indirecte : qui permettent à l'utilisateur de désigner ou d'indiquer des objets à distance.

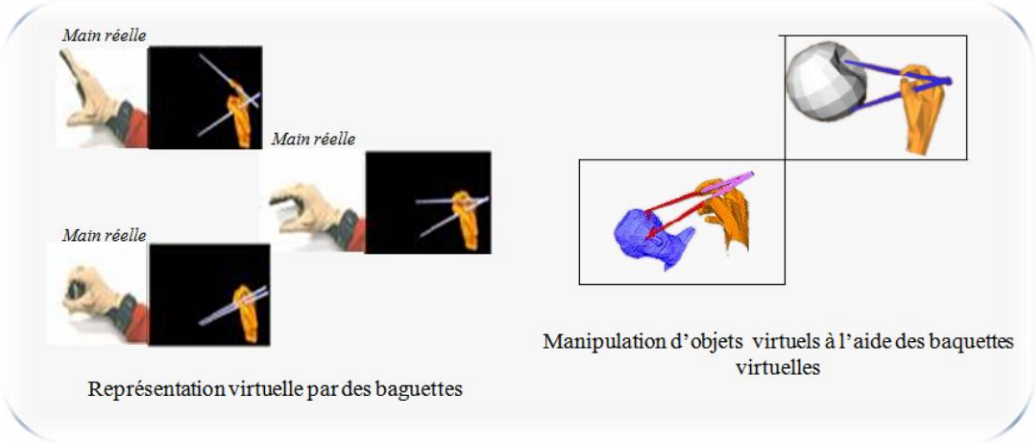
dirigée du doigt : Utilise les techniques de la métaphore qui nécessite que l'index soit repérable dans l'espace virtuel en utilisant un capteur de position attaché au doigt .

Pierce et associés [16] : Proposent une autre technique de sélection qui utilise les mouvements de la main pour sélectionner des objets.

Remarque : D'autres techniques basées sur le même principe ont été proposées.
la technique des mains encadrantes dans laquelle les deux mains sont utilisées pour former un cadre de sélection.

Techniques de manipulation directe :

1) Kitamura et associés [13] : Qui utilisent des baguettes chinoises, permettant de saisir, déplacer et tourner les objets.



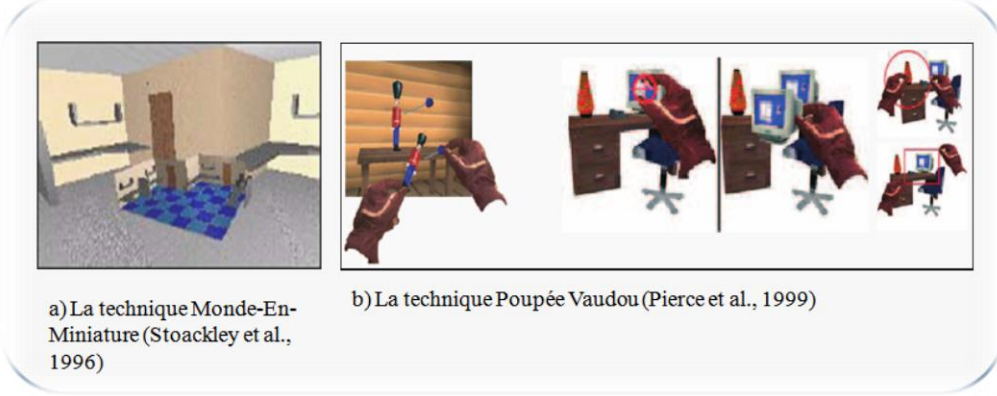
I.8- Manipulation par baguettes chinoises

1.6.2 Les techniques exocentriques

L'utilisateur interagit avec l'environnement 3D de l'extérieur.

Stoackley et associés [21] : proposent un métaphore de sélection et de manipulation qui repose sur le principe de l'interaction exocentrique, appelée « monde-en-miniature ».

Pierce et associés [17] : Ont proposé la technique des Poupées Vaudou qui offre à l'utilisateur la possibilité de créer ses propres objets miniatures du monde virtuel, qui sont nommés « poupées ».



I.9- Techniques d'interaction exocentriques

1.7 Le Menu

Le contrôle d'application ou système de commande est une tâche qui permet d'exécuter une commande dans le but de changer le mode d'interaction ou l'état du système.

Bowman et associés [3] : Le contrôle d'application est comme les changements de l'état du système ou le mode d'interaction. Cette tâche se situe à un niveau conceptuel différent des trois autres tâches *« Navigation, Manipulation, Sélection »*.

Dans le cas d'une interface 3D, on ne peut pas utiliser les techniques classiques pour interagir avec l'application, car la souris ou le clavier ne sont pas utilisables.

Jacoby et associés [27] : proposent un menu déroulant 2D nommé "*menu 2D converti"* qui est librement positionnable et orientable dans l'espace.

L'utilisateur sélectionne et active un élément dans le menu en utilisant la métaphore du pointeur virtuel, en le pointant du doigt .

I.10- Menus 2D adaptés aux environnements 3D.

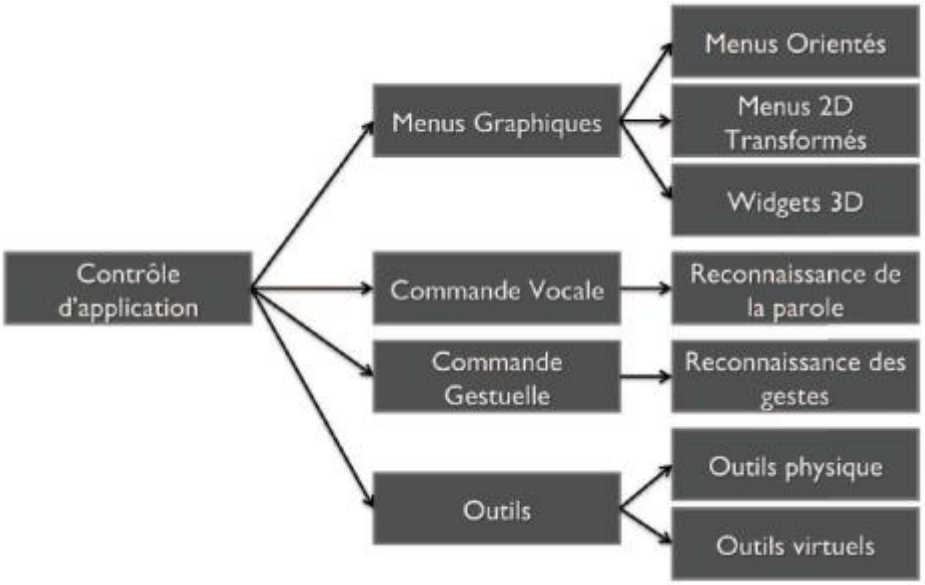
- *Menu TULIP :* est un menu déroulant basé sur l'utilisation des doigts. La main de l'utilisateur est équipée d'un gant de données et chaque doigt correspond à un élément du menu.
- *Menu C3 :* proposé par Grosjean et associés [12] [11] utilise le concept des Marking Menus. Ce dernier propose de disposer les menus en cube plutôt qu'en liste pour en accélérer l'accès.

I.11 Menus 3D.

- *Les menus circulaires* : Largement utilisés en Réalité virtuelle. Les différents éléments sont placés sur un cercle.

I.12- Menus circulaire.

Liang et associés [[14]] : Utilisent les menus circulaires, sans hiérarchie.
Les éléments sont disposés en cercle autour d'un axe vertical. Le menu est alors représenté par un anneau avec un trou au milieu.



I.13- Classification des méthodes de contrôle de l'application [Bowman et al, 2004]

1.8 Conclusion :

Dans ce chapitre, nous avons abordé le thème de la réalité virtuelle, ses définitions et ses composantes (l'immersion, l'autonomie et l'interaction 3D). Ensuite, nous avons mis l'accent tout particulièrement sur l'interaction 3D. Elle se définit en quatre tâches de base : la navigation, la sélection, la manipulation et le contrôle d'application.

CHAPITRE 2: **Transformation et Représentation des Objets 3D**

2.1 Introduction :

Un Espace Virtuel est généralement une scène 3D qui est définie par une ou plusieurs Camera et des Objet 3D qui sont représentés avec des mouvements dynamiques et de la texture.

Dans ce chapitre nous étudierons deux parties :

- La première est conçue sur les transformations géométriques en vision mathématique et infographique telle qu'on manipule les mouvements en utilisant des coordonnées homogènes. Et même les transformations complexes tel que le mouvement de la camera. Apres cela nous verrons comment passer de la dimension 3D vers la dimenssi2D utilisant les techniques de projection avec de ces 2 types.
  - Projection Parallèle est utilisée pour l'architecture
  - Projection Perceptive est utilise pour le rendue réaliste
- Et la deuxième partie est conçue sur les technique de représentation des objets 3D tels que les mail triangulaire, fil de fer , arbre CSG ...etc. pour construire un objet 3D on utilise des logiciels libres comme **Blender** ou commerciaux comme **3DSMax studio ,** ce sont des logiciels spéciaux destiné pour la modélisation de ces objets et on peut exporter ces objets 3D.

2.2 Transformation Géométrique :

2.2.1 Infographie :

Concernant Cette partie on fait une introduction sur les transformations géométriques en 2D en utilisant des coordonnées homogènes telles que les matrices 3x3.

2.2.1.1 Utilisation des coordonnées homogènes :

Outil géométrique très puissant Utilisé dans plusieurs domaines : Synthèse d'images, Vision par ordinateur, Robotique

- On ajoute une troisième coordonnée, w- Par défaut, on pose w=1
- Un point 2D devient un vecteur à 3 coordonnées :

$$P = \begin{pmatrix} x \\ y \\ w \end{pmatrix}$$

*Remarque : Nous quittons maintenant le domaine de la **géométrie euclidienne** pour entrer dans celle de la **géométrie projective** [II2]*

2.2.1.2 Coordonnées homogènes :

- Deux points sont égaux si et seulement si :

$$\frac{x'}{w'} = \frac{x}{w} \text{ Et } \frac{y'}{w'} = \frac{y}{w}$$

- En règle générale, on va s'assurer que la coordonnée homogène (w) soit toujours égale à 1.
  - Sinon, on divise le résultat par w pour obtenir 1.
  - Si la coordonnée homogène vaut 0, alors le point est à l'infini (∞)

- Point à l'infini : projection perspective

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = [x'/w \quad y'/w]$$

2.2.2 Théorème mathématique : [II1] [II4]

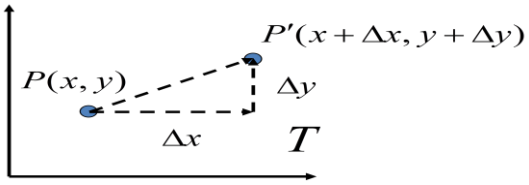
Dans cette partie on fait un rappel sur les transformations géométriques en 2D et en 3D on utilise des matrices 2x2 et les matrices 3x3.

2.2.2.1 Opération en 2D : [II3]

2.2.2.1.1 Translation :

- Translation d'un point d'une position vers une autre :

$$P' = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x' \\ y' \end{bmatrix} + \begin{bmatrix} Tx \\ Ty \end{bmatrix}$$

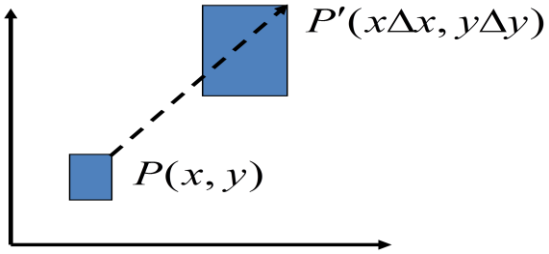


II.1- Translation 2D

2.2.2.1.2 Homothétie :

- Pour obtenir un changement d'échelle(ou homothétie), il suffit de multiplier les coordonnées par un facteur d'échelle.

$$\boldsymbol{P'} = \begin{bmatrix} \boldsymbol{x'} \\ \boldsymbol{y'} \end{bmatrix} = \begin{bmatrix} S_x & \mathbf{0} \\ \mathbf{0} & S_y \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix}$$

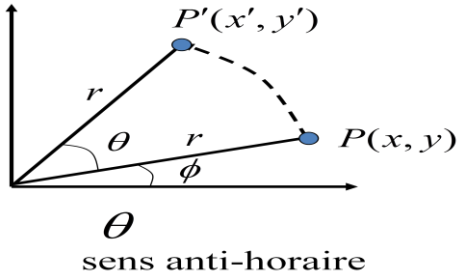


II.2- Homothétie

2.2.2.1.3 Rotation :

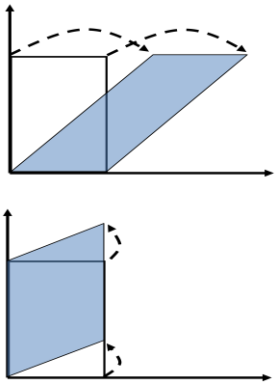
- Rotation d'un point autour de l'origine :

$$P' = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$



II.3- Rotation2D

2.2.2.1.4 Cisaillement en 2D (shearing)

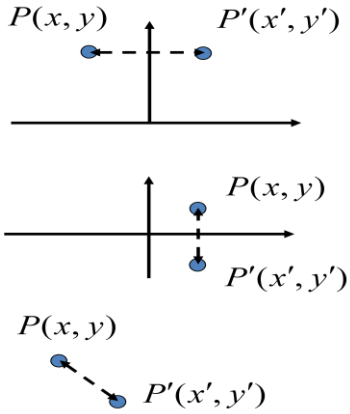


$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & a \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$
$$P' = Sh_x P$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ b & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$
$$P' = Sh_y P$$

II.4- Cassillement

2.2.2.1.5 Réflexion en 2D



$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

2.2.2.1.6 Glissement :

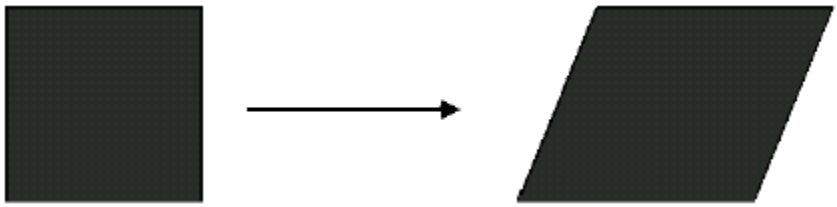
Une autre transformation, le glissement *(shear),* permet de déformer les objets :

- ne conserve pas les angles
- ne conserve pas l'aire des surfaces

On peut faire un glissement dans l'axe des X ou selon l'axe des Y

$$G_x = \begin{bmatrix} 1 & Sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$G_y = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & Sh_x \\ 0 & 0 & 1 \end{bmatrix}$$



II.5- Glissement selon l'axe X

2.2.2.1.7 Réflexion :

Réflexion d'un point par rapport à un axe- « transformation miroir »

- par rapport à un axe X :

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

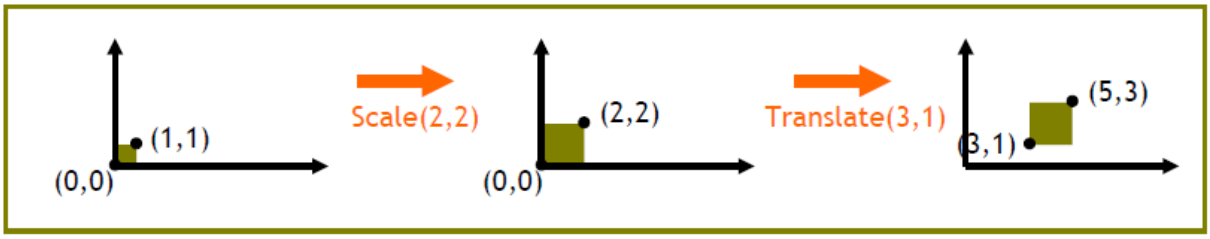
- par rapport à un axe Y :

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- par rapport à l'origine :

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Exemple :



II.6- Exemple Transformation

Multiplication des Matrices : P' = Translate (Scale (p))
P' = T S(P)

$$TS = \begin{pmatrix} 1 & 0 & 3 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 0 & 3 \\ 0 & 2 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

2.2.2.2 Ordre de transformation :

Les transformations ne sont pas commutatives :

- Rotation * Translation ≠ Translation * Rotation

On peut inverser les transformations semblables :

- Rotation1 * Rotation2 = Rotation2 * Rotation1
- Translation1 * Translation2 = Translation2 * Translation1

1. Opération en 3D : [II1] [II4]

1) Translation :

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

2) Homothétie :

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

3) Rotation :

- Autour de l'axe X

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi & 0 \\ 0 & \sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- Autour de l'axe Y

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\psi & 0 & \sin\psi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\psi & 0 & \cos\psi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- Autour de l'axe Z

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

2. Paramètres extrinsèques de la caméra :[II5]

Dans un montage réel, il est impossible de garantir que le repère réel soit aligné avec celui de la caméra.

La position de la caméra dans l'espace réel est la composition de :

- Une translation du centre C par rapport à l'origine du repère « *Xe ,Ye , Ze ».*
- Trois rotations des axes caméras par rapport au repère de l'environnement les paramètres caractérisant la relation entre la caméra et le référentiel sont dits *extrinsèques.*

Le passage des coordonnées environnement aux coordonnées caméra est :

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x \\ 0 & \sin\theta_x & \cos\theta_x \end{bmatrix} \begin{bmatrix} \cos\theta_y & 0 & \sin\theta_y \\ 0 & 1 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y \end{bmatrix} \begin{bmatrix} \cos\theta_z & -\sin\theta_z & 0 \\ \sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_e \\ y_e \\ z_e \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

L'ensemble de ces paramètres est représenté en coordonnées homogènes :

$$\begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} = \begin{bmatrix} r_{x/x} & r_{x/y} & r_{x/z} & t_x \\ r_{y/x} & r_{y/y} & r_{y/z} & t_y \\ r_{z/x} & r_{z/y} & r_{z/z} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_e \\ y_e \\ z_e \\ 1 \end{bmatrix}$$

2.3 Représentation des objets 3D

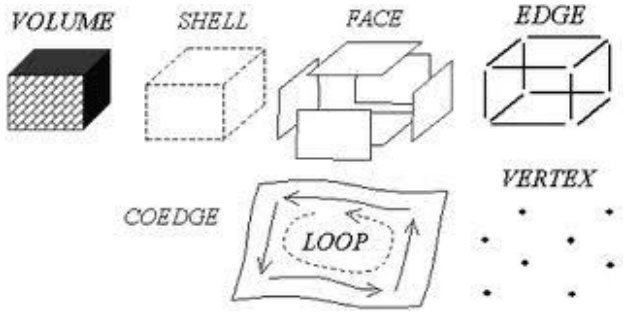
2.3.1 Type de Représentation :

Il existe 2 grande familles pour la représentation des Objets 3D la Première est la représentation en modèle par frontière. et la deuxième en modèle volumique. Telle que dans le premier modèle on trouve la Définition du volume par l'énumération de ses frontières B-Rep et pour le modèle volumique il y a 2 sous catégories pour la représentation la première est par construction exemple des arbres CSG et Level-Sets et la deuxième par décomposition comme l'énumération exhaustive et maillage

2.3.2 Modèle par frontière *(Bord)* : [Il6]

2.3.2.1 B-Rep :

Boundary Représentation ou Représentation Frontière ou Représentation par les Bords est une technique de modélisation 3D géométrique des solides par les surfaces. Le principe est que le volume est défini par son intérieur, son extérieur et sa frontière (ou bord), telle que Le bord d'un volume est une surface fermée.



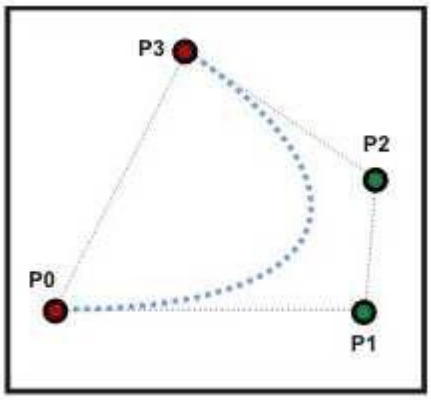
II.7- B-Rep

2.3.2.1.1 Approximation surfacique :

Cette méthode consiste à représenter la peau des objets géométriques et les volumes sont représentes par des surfaces canoniques (Bézier, B-spline, Nurbs,) et les problèmes qui vont avec (raccordement et construction).

Bézier : Les courbes de Bézier sont des courbes polynomiales qui ont été introduites par l'ingénieur français Pierre Bézier, qui travaillait chez Renault dans les années 60. Aujourd'hui les courbes de Bézier ont encore beaucoup

d'applications, par exemple en graphisme ou dans la synthèse d'images (PostScript, GIMP,…etc.).
Une courbe de Bézier cubique est caractérisée par 4 points, appel´es points de contrôle. Le premier et le dernier point sont des nœuds. Les deux autres points permettent de définir la forme de l'arc, la courbe ne passant pas en général par ces deux points de contrôle**. [II10]**



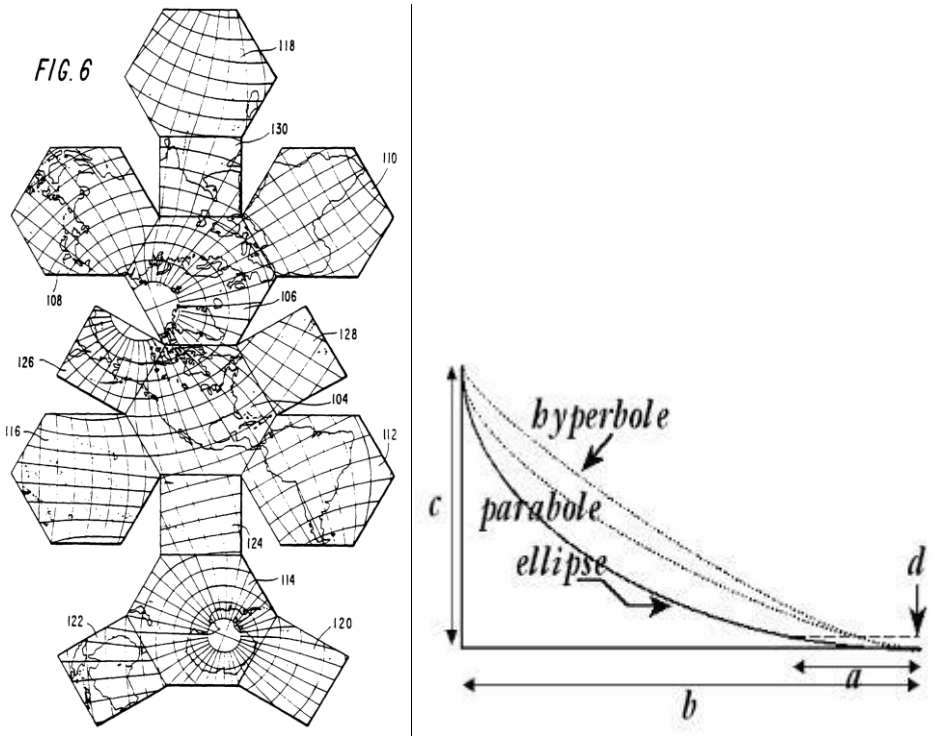
II.8- Bézier

Une courbe de Bézier cubique avec les points de contrôle P0 , … , P3.

$$\mathbf{P}(t) = \mathbf{P}_0(1-t)^3 + 3\mathbf{P}_1 t(1-t)^2 + 3\mathbf{P}_2 t^2(1-t) + \mathbf{P}_3 t^3$$

2.3.2.1.2 <u>Approximation polyédrique :</u>

Un polyèdre est traditionnellement une forme tridimensionnelle qui se compose d'un nombre fini de faces polygonales qui sont des parties de plans; les faces se rencontrent par paires le long des arêtes qui sont des segments de droite, et les arêtes se rencontrent aux points nommés sommets. Les cubes et les cônes sont des exemples de polyèdres. Le polyèdre entoure un volume limité dans l'espace à trois dimensions; quelquefois ce volume intérieur est considéré être une partie du polyèdre, quelquefois, seule la surface est considérée.



II.9- Polyèdre

2.3.2.1.3 Structurer les surfaces :

Même pour les applications les plus simples, on se retrouve rapidement confronté à connaitre la structure (**la topologie**) des surfaces composées par facettes :

- Trouver les facettes incidentes à un sommet ;
- les arêtes d'une facette ;
- recherche d'un chemin entre 2 sommets de la surface

2.3.2.1.4 Conditions :

- Toute arête réunit 2 sommets et seulement 2 sommets.
- Toute arête est incidente à une facette et à au maximum deux facettes

o Une arête incidente à une seule face est une arête du bord de la surface.

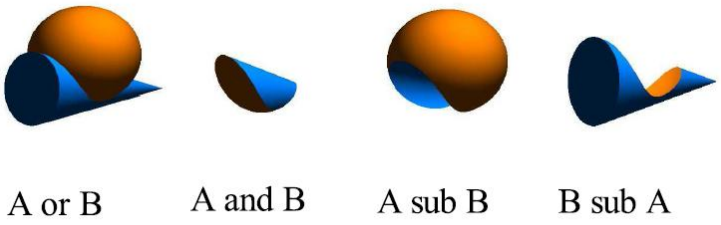
o Pour les volumes par B-rep : toute arête est incidente à exactement deux faces (le bord du volume est une surface fermée).

- Géométrique1 : toute facette est plane (pas toujours requis ; peut être imposée en prenant des triangles).
- Géométrique2 : la surface ne s'auto-intersecté pas pour garantir la définition d'un intérieur et d'un extérieur (rarement validé par les outils de construction).

2.3.2.2 ***Arbres CSG :*** Arbre Constructive solid Geometry **[II6]**

2.3.2.2.1 Opérations booléennes :

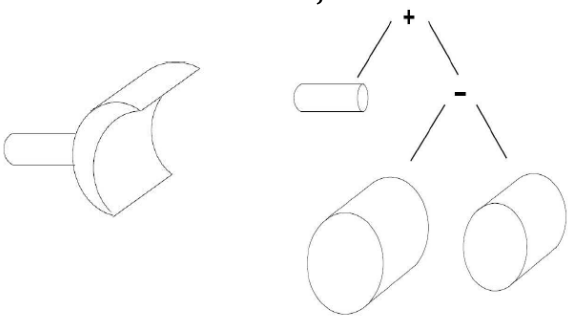
- Si on dispose déjà d'objets (quelque soit leur représentation) une manière naturelle de les combiner est d'utiliser les opérations dites booléennes : intersection (and), réunion (or) et différence (sub).
- Exp : A une sphère et B un cône.



II.10- Arbre CSG

2.3.2.2.2 Arbre de construction :

- Des objets complexes peuvent être obtenus par opération booléennes de primitives simples.
- Représentation arborescente d'un objet.



II.11- Exemple Arbre CSG

2.3.2.2.3 Définition :

- C'est un arbre binaire.
- Chaque nœud contient une matrice de passage M_{pere}➔nœud (positionnement du nœud par rapport au père).
- Les feuilles sont constituées de primitives géométriques.
- Les nœuds internes contiennent un opérateur booléen (and, or, sub).

2.3.2.3 Enumération d'occupation spatiale :

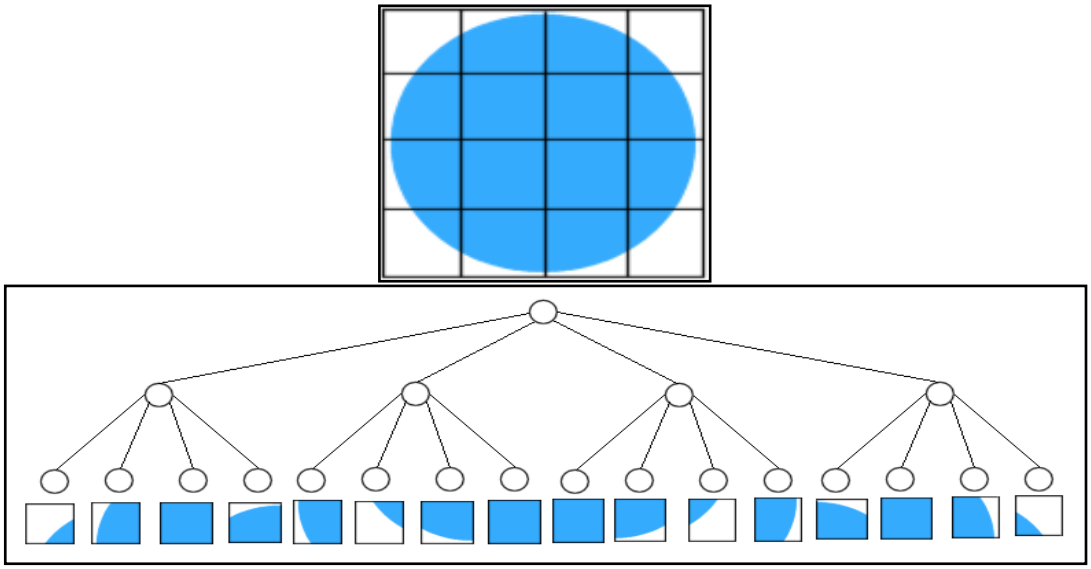
Les techniques qui existent pour l'énumération spatiale sont trois technique Quadtree, Octree et maillage

- Les solides sont représentés par une liste de voxels tells que les voxels sont des cubes de taille uniforme repartis selon une grille.

2.3.2.3.1 Quadtree (2D):

La Structure de donnée de ce type de représentation est :

- *encode une image 2D en plaçant des sous partie de l'image dans un arbre.*
- *chaque nœud de l'arbre dispose 4 fils.*
- *généralement on partitionne l'image en 4 quadrants ayant chacun un nœud.*
- *et donc représentant 1/4 de l'image et ainsi de suite pour ses fils…*
- *on continue jusqu'à avoir une région suffisamment simple :*
    - *taille d'un pixel*
    - *Couleur*
    - *résolution souhaité (certaine hauteur dans l 'arbre)* **[II8]**



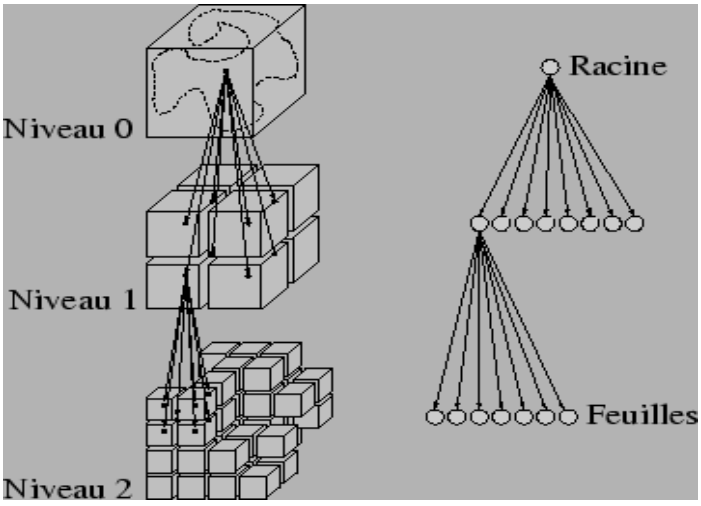
II. 12- Arbre quadtree

2.3.2.3.2 Octree (3D):

La Structure de donnée de ce type de représentation est :

Un Octree est un Quadtree mais avec 8 fils au lieu de 4. Ce qui en fait un bon outils pour décomposer une scène en 3D On décompose l 'Octree de la même façon que le Quadtree jusqu'à obtenir la partition désirée. **[II8]**

Remarque : pour générer facilement la coordonnée des textures.



II.13- Arbre Octree

2.3.2.3.3 Maillage:

La Méthode des Éléments Finis (MEF) est probablement la technique la plus utilisée pour la modélisation du comportement mécanique des solides. Elle s'appuie pour cela sur une discrétisations du domaine modélisé en éléments géométrique simples. Cette partition porte le nom de maillage.
un maillage est considéré comme une partition d'un domaine arbitraire dans des objets géométriques simples (ou éléments). Ces éléments sont composés de nœuds, arêtes, faces et les relations entre eux. **[II11]**

Exemple : Les scanners donnent une grille régulière de voxels (éléments de volume par analogie avec les pixels de l'écran). Un voxel « allumé » contient de la matière, un voxel « éteint » n'en contient pas.**[II6]**

II.14- Grille de voxels

2.4 Animation :

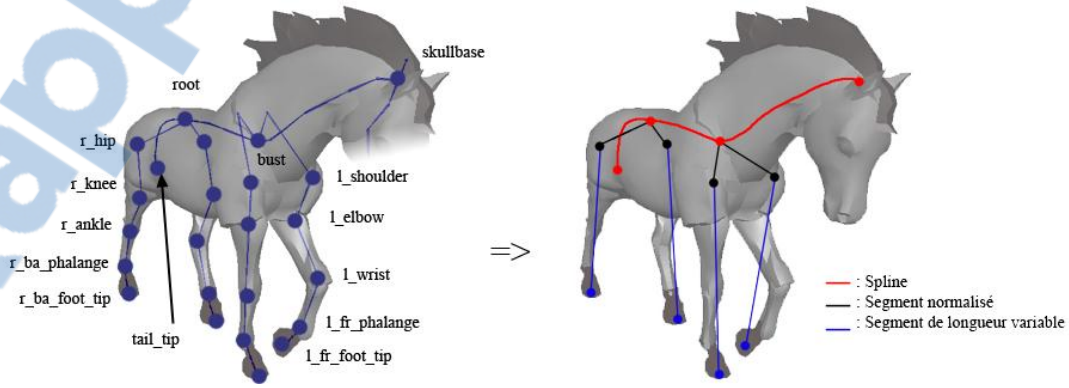
2.4.1 Définition :

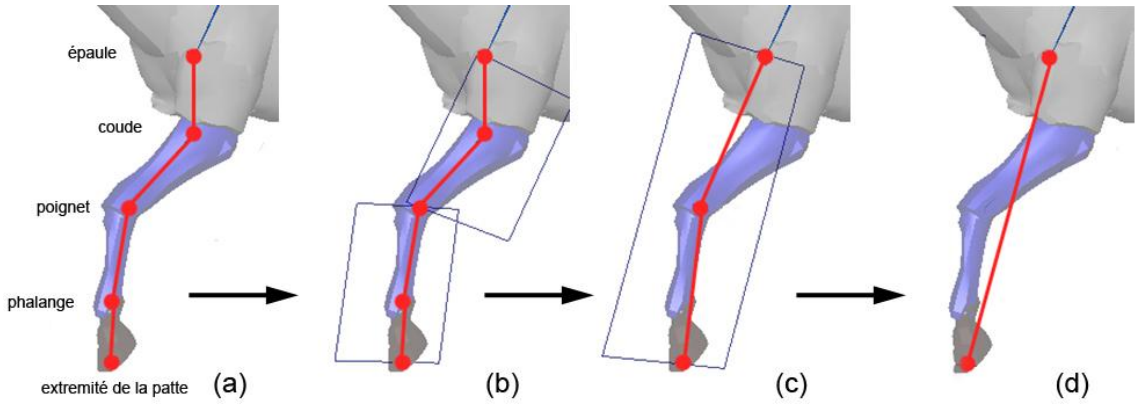
Frame : étape d'une animation d'un modèle.

Keyframe : étape "clé" dans une animation ; c'est-à-dire, une étape importante dans une animation donnée (c'est une frame particulière).

Mesh : modèle 3D, composé d'un ensemble de polygones. Les polygones sont généralement des triangles, donc composés de trois sommets. Dans notre cas simplifié, un sommet a trois coordonnées (x, y, z) et deux coordonnées de texture (u, v).

Squelette : arbre hiérarchique de points (qu'on appellera Joints), qui représente le squelette du modèle ; les joints sont les articulations du squelette.





II.15- Encodage des articulations intermédiaires

Trouzaine : signifie "grande quantité de". Définitivement plus qu'une trouzaine. Exemple : "Le développement de ce jeu nécessite des trouzaines d'heures de travail." **[II7]**

2.4.2 Théorie :

L'animation d'un mesh consiste à déplacer ses sommets dans le temps. Pour cela, deux approches sont possibles :

- Animation par frame : pour chaque étape de l'animation, on exporte le modèle entier. A la fin, on a un ensemble de modèles, qu'on affiche successivement pour donner l'illusion du mouvement, comme pour un film.
- Animation par squelette : on sépare le mesh et le squelette ; seul le squelette est animé. Il va contrôler et faciliter le déplacement des sommets.

Dans ce dernier cas, pour chaque étape, on anime uniquement le squelette, et on calcule les nouvelles positions des sommets du mesh. Il y a plusieurs avantages à cela :

- on ne duplique pas les sommets pour chaque étape d'une animation, donc on gagne de l'espace mémoire,
- on peut animer les modèles plus facilement par le squelette,
- lors du rendu du mesh, on peut calculer son animation à la volée. **[II 7]**

2.5 Conclusion :

Dans ce chapitre, nous avons abordé la transformation géométrique et la représentation des Objets 3D et les techniques de l'animation.

Pour la transformation géométrique on a vu les matrices de transformation comme la translation, rotation, scaling et la projection avec ces deux types parallèle et perceptive.

Pour la représentation des Objets 3d on a vu les deux modèles de représentation, volumique et modèle par frontière tels que :

- Pour le modèle volumique on a vu trois représentations les arbres CSG, l'énumération exhaustive (Octree et Quadtree) et maillage
- Pour le modèle par frontière B-Rep on a vu les polyèdres tridimensionnels, les courbes de Bézier et Winged-edges :

Pour l'animation on a vu deux types soit utilisation des squelettes ou des frames.

CHAPITRE 3: **Rendu Réaliste**

3.1 Introduction

Le rendu réaliste c'est la position des ressources de lumière dans la scène 3D avec le placage des textures sur les objets 3D et finalement application d'un algorithme de rendue pour faire la projection de la scène sur l'écran :

- Jusqu'à maintenant il y a un problème de définition de la lumière , elle peut être plus ou moins intense et colorée , avoir une direction et une position. Alors la solution est de créer plusieurs sources de lumière différentes afin d'avoir un rendue plus réaliste.
- La texture est une image que l'on applique a un triangle ou facette polygonale (placage ou de mapping d'une texture). Le principe du placage est d'associer a chaque point d'une surface 3D une couleur invariable que l'on appelle un texel. Et la bibliothèque permet de charger tout format des images.
- Pour le rendue il y a plusieurs algorithmes de rendue comme Z_Buffer (depth buffer), Peintre et radiosité (Ray-Tracing).

3.2 La Lumière

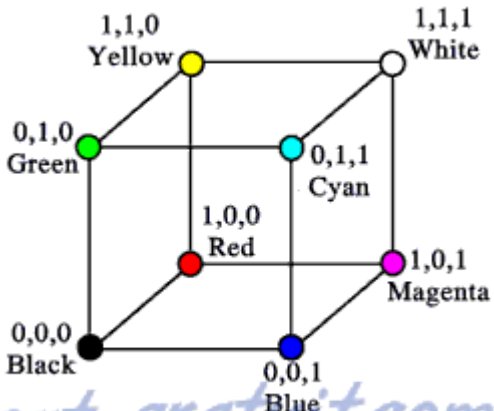
3.2.1 Le modèle de couleur et modèle d'illumination:

3.2.1.1 Le modèle de couleur :

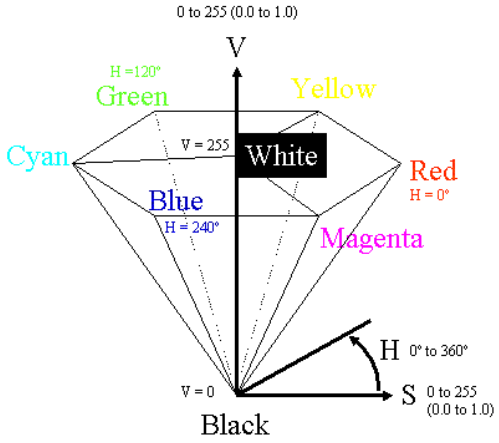
Le modèle de couleur le plus commun est le modèle RGB pour (Red, Green, Blue). La lumière est considérée comme la synthèse additive des trois lumières de couleur rouge, verte et bleu dont l'intensité varie entre 0 et 1. Ce modèle peut être considéré comme un cube d'origine la couleur noire et d'axes rouge, vert et bleu. Il confond l'intensité et la couleur de la lumière.

D'autres modèles de couleurs, moins usités existent :

- *Le modèle XYZ [6]* : développé par l'International Lighting Comittee, est un modèle mathématique indépendant du matériel utilisé.
- *Le modèle CMY (modèle soustractif Cyan-Magenta-Jaune)* : définit sous la forme d'un cube, mais cette fois ci l'origine est le blanc et les trois axes sont (cyan, magenta et jaune). La lumière est ainsi une synthèse soustractive des couleurs primaires.
- *Le modèle HSV[4] (Hue, Saturation, Value)* : correspond à une notion plus intuitive de la couleur pour un artiste. H représente la couleur, S la pureté de la couleur et V sa luminosité.[1]



III.1- Cube de couleur RGB



III.2- Représentation du modèle de couleur HSV

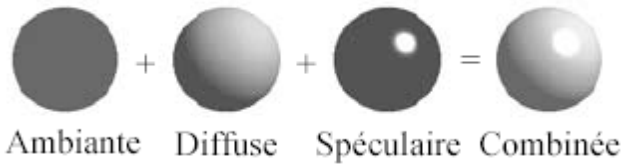
3.2.1.2 Le modèle d'illumination : [1]

Le processus de perception de la couleur dépend de plusieurs paramètres. Le modèle d'illumination fait intervenir :

- La position et l'orientation de(l'observateur, l'objet, la source de lumière).
- La couleur et l'intensité de la lumière émise par la source de lumière.
- Les propriétés d'absorption et de réflexion de l'objet.

3.2.2 Types de lumières : [1]

Pour coller le plus possible à la réalité différente types de lumières ont été créés. La combinaison de plusieurs types de lumière est possible pour obtenir un meilleur rendu possible.



III.3- Différents types de lumière et leur somme.

3.2.2.1 La lumière ambiante:

Semblable au rayonnement de fond de l'univers. La lumière arrive de tous les cotés en plus son intensité et sa couleur sont uniformes, elle n'a ni position ni direction. Ce type de lumière est le plus simple, il consiste à :

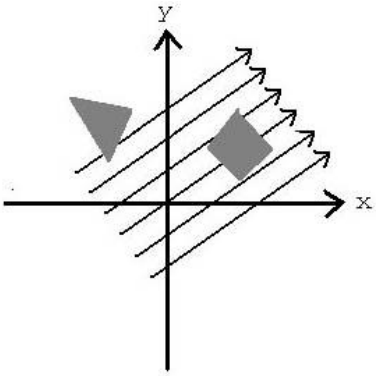
- ajouter une constante de lumière uniforme à tous les points.
- faire varier l'intensité et la couleur de l'éclairage ambiant (bleu sous l'eau,).

- *Kamb* : coefficient entre 0 et 1 propre à la surface,
- *Iamb* : constante
- *Col* : couleur de la surface [3]

$$I_{e=} I_{amb} \cdot K_{amb} \cdot Col$$

3.2.2.2 La lumière directionnelle :

La lumière directionnelle illumine notre scène comme le soleil illumine le monde réel.

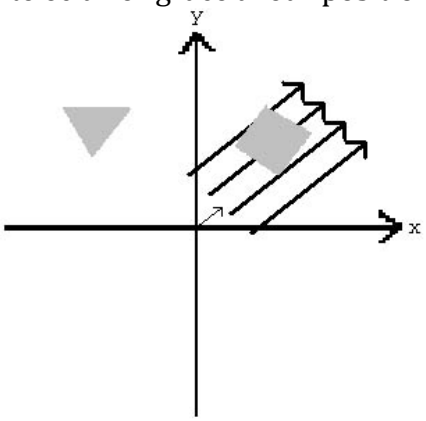


III.4- La lumière directionnelle.

Pour utiliser la lumière directionnelle est plus il faut calculer des produits scalaires au lieu d'ajouter une constante au contraire de l'éclairage d'ambiant.

3.2.2.3 Les sources de lumières parallèles:

Elle est définie par une position dans notre Monde, qui sera le point de départ de la lumière, et une direction. L'utilisation de cette source de lumière permet de choisir les objets que l'on souhaite éclairer grâce à leur position.

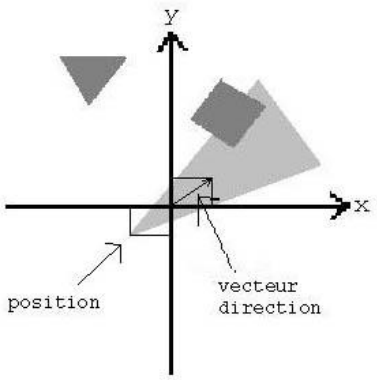


III.5- Une source de lumière parallèle définie par une position et une direction.

3.2.2.4 Les lumières divergentes: Spot

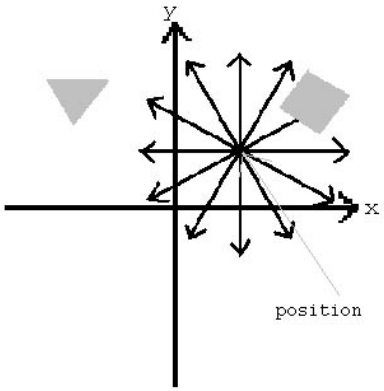
Un spot éclaire la scène en produisant un cône de lumière à partir d'une position donnée, d'une direction donnée et d'un angle. En pratique deux cônes employés:
Les spots sont utilisés dans le rendu des (projecteurs, phares de voitures, lampes de poche, faisceaux laser).
possède :

- deux cônes de lumière
- un facteur d'atténuation
- une position et une direction.



III.6- Une lumière divergente ou spot possède une position et une direction.

Exp : La lumière d'une ampoule éclaire la salle dans toutes les directions.



III.7- Une source de lumière ponctuelle est définie par une position.

3.2.2.5 Le matériau :[3]

L'utilisation d'une source de lumière n'a aucun effet si le point illuminé n'est pas doté d'un matériau qui décrit son comportement face à la lumière. Dans le monde réel, le bois et le métal n'ont pas les mêmes propriétés, ils n'absorbent et ne reflètent pas la lumière de la même façon.

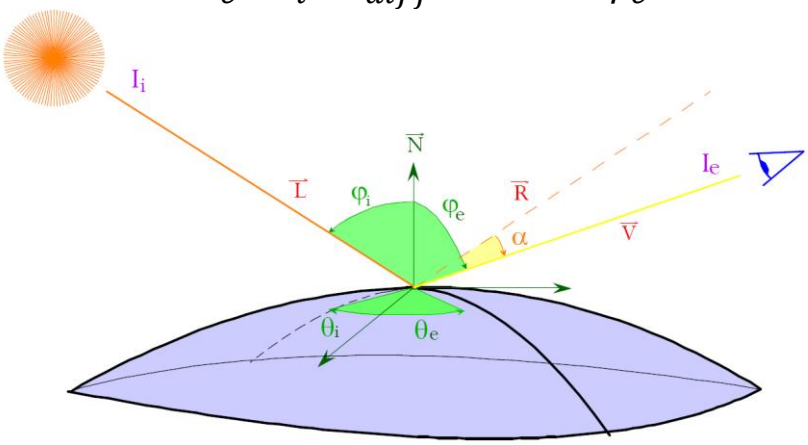
Un matériau possède les quatre caractéristiques suivantes :

- La couleur (diffuse, ambiante, émissive, spéculaire et son intensité).
- **Couleur diffuse :** Correspond à la capacité d'absorption du spectre de la Lumière incidente. Si on éclaire un objet avec une lumière blanche et que sa couleur diffuse est rouge l'objet sera vu rouge.
- **Couleur ambiante :** Est la couleur d'un objet en l'absence de lumière. Le résultat obtenu n'est pas très réaliste (dans le monde réel un objet non éclairé est invisible). Cependant le résultat est intéressant dans un univers de science fiction.
- **Couleur émissive :** Permet simplement de simuler la surexposition d'un objet. Si on attribut une couleur émissive à un objet il renverra plus de lumière que ce qu'il en reçoit.
- **Couleur spéculaire :** Est beaucoup plus intéressante parce qu'elle permet de définir la couleur et l'intensité des reflets. Une faible intensité nous donnera des objets mats, une forte intensité, des objets brillants : par exemple d'apparence métallique.

- Composante diffusée : On utilisera le modèle de Lambert :
  L : direction d'incidence

V : direction d'émission
Ie : l'intensité émise
Ii : l'intensité reçue
Col : de la couleur de la surface
Kdiff : coefficient de réflexion diffuse *Kdiff = (Kdrouge, Kdvert, Kdbleu),*
est donnée par :

$$I_e = I_i \cdot K_{diff} \cdot Col \cdot \cos \varphi_e$$



III.8- Variable de l'illumination

Pour connaître toute l'intensité émise selon une direction, il faut intégrer cette formule pour toutes les directions entrantes. Dans ce modèle, La direction d'incidence n'a pas d'importance.[2]

- *Composante spéculaire :*

R : la direction de réflexion idéale
Kspec : le coefficient de réflexion spéculaire du matériau
ns : son paramètre de taille du faisceau

$$I_{e=} I_i \cdot K_{spec} \cdot \cos^{ns} \alpha$$

On a un effet miroir lorsque ns ➔ ∞ (réflexion parfaite). On ne prend pas en compte la couleur de la surface, et en général, c'est plutôt la couleur de la source qu'on aperçoit dans un reflet. Pour le rayon réfracté, même formule mais par rapport à la direction de réfraction idéale (*b* dans la conclusion).

3.3 Textures

3.3.1 Texture

Les objet ont rarement une surface lisse car celle-ci possède une texture plus ou moins complexe. Le rendu réaliste doit tenir compte de cette texture.

- On peut tout d'abord réaliser une application de texture bi-dimensionnelle sur une surface 3D (texture mapping). Ceci se réalise par une transformation entre une région R de la surface 3D de coordonnées paramétriques (u,v) et une région R' du plan de texture de coordonnées (x,y) coordonnées paramétriques (u,v) et une région R' du plan de texture de coordonnées (x,y).

A chaque point de R correspond un point de R', la correspondance étant définie par la transformation définie par les fonctions f et g:

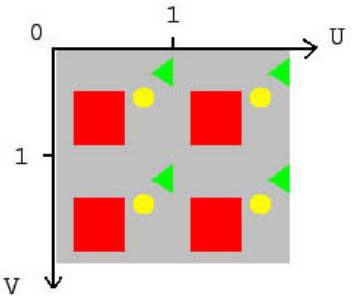
3.3.2 Les Coordonnées: [1][3]

Une texture est une image généralement rectangulaire ou carrée, et souvent de taille égale à une puissance de 2.Donc une dimension comprise entre 8 et 256 permet d'éviter certains problèmes, en effet certaines cartes graphiques ne supportent pas de textures de dimensions supérieures à 256. Direct 3D est optimisé pour les carrés de 256 texels de coté.

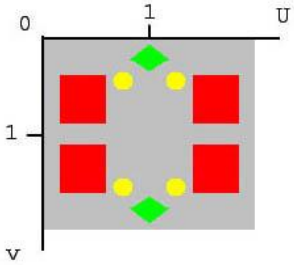
Les coordonnées de textures permettent de sélectionner seulement la partie de la texture qui nous intéresse. Ces coordonnées sont notées u et v.

- L'origine est le coin supérieur gauche de la texture.
- L'axe U est horizontal, dirigé vers la droite
- L'axe V est vertical dirigé vers le bas.

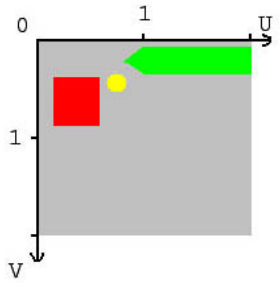
Ces coordonnées sont des nombres flottants, en plus le point en bas à droite à toujours pour coordonnées (1.0f, 1.0f), et ce même si la texture n'est pas carrée. Une texture peut être substituée à une autre sans avoir à changer la moindre ligne de code.



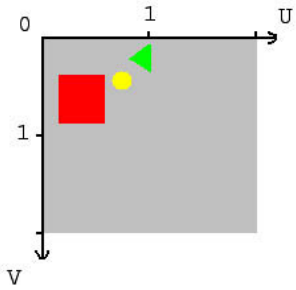
III.9- Le mode normal : la texture est appliquée et se répète à l'infini.



III.10- Le mode miroir : On effectue une symétrie par rapport à l'axe de la coordonnée dépassée (aussi bien en U qu'en V). La texture semble être composée d'un motif 2x2 textures symétriques se répétant à l'infini.



III.11- Le mode clamp: La texture est étirée en répétant sa dernière ligne ou sa dernière colonne si le dépassement se fait en V ou en U.



III.12- Le mode bordure : La texture n'est pas répétée, en cas de dépassement une couleur prédéfinie est renvoyée.

3.3.3 Le filtrage : [1]

C'est le processus qui détermine la couleur d'un pixel de la surface de rendu (l'écran) en fonction d'un ou plusieurs texels. Il nécessite plusieurs paramètres :

- Les coordonnées du point dans la surface de rendu
- La profondeur de ce point dans la scène
- Les coordonnées de placage de la texture de ce point

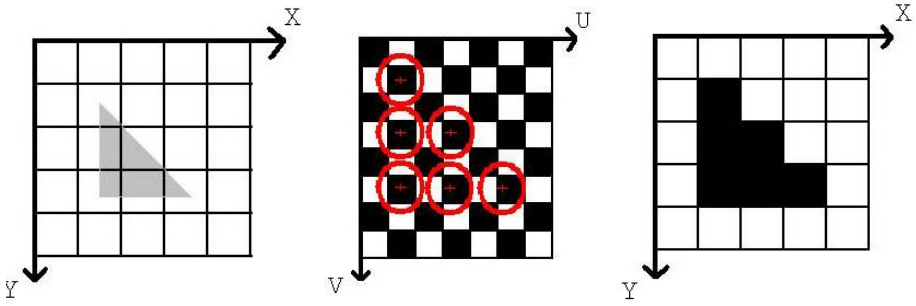
On utilise différentes techniques de filtrage :

3.3.3.1 Le filtre discret :

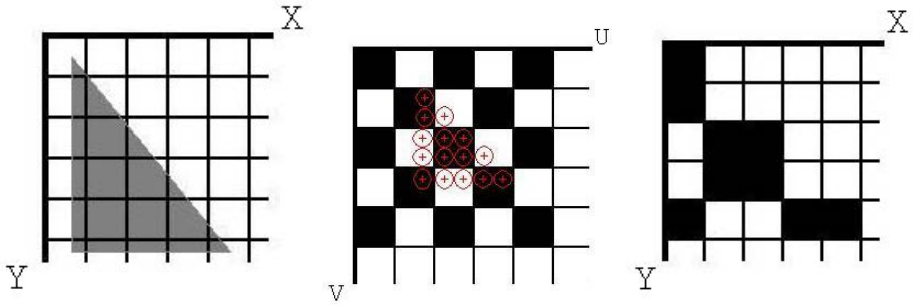
A chaque pixel on doit associer un seul texel.

- *Dans le cas où l'on est loin de l'objet* : un pixel recouvre plusieurs texels ; on choisit alors le texel au centre du pixel.
- *Dans le cas ou l'on est proche de l'objet :* plusieurs pixels représenteront le même texel (effet de pixellisation). L'effet d'un tel filtre n'est correcte que lorsque la distance est idéale (un ratio pixel/texel proche de 1).

Le rendu de cette technique est médiocre.



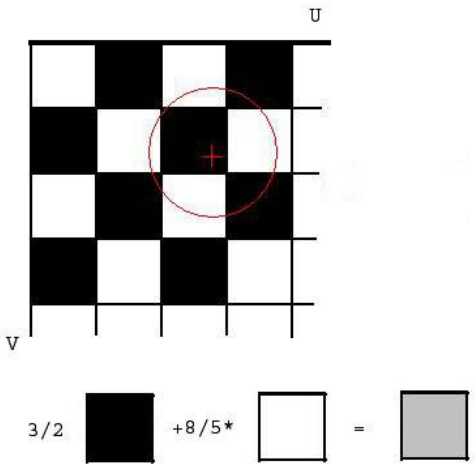
III.13- Dénaturation d'une texture quand un pixel recouvre plusieurs texels lors d'un filtrage discret.



III.14- Dénaturation d'une texture quand un pixel recouvre moins d'un texel lors d'un filtrage discret.

3.3.3.2 Le filtre bilinéaire :

La couleur du pixel est assimilée à la moyenne pondérée des couleurs des texels qu'il recouvre Cette moyenne est calculée par interpolation bilinéaire.
Une telle approximation pose un problème, quand le pixel recouvre plus de texels que le nombre considéré. Un filtrage bilinéaire risque de dénaturer le motif de la texture d'origine.
Cette technique permet d'éviter un amas de pixel uniforme à l'approche d'un objet.



III.15- Utilisation d'un filtre bilinéaire.

3.3.3.3 Le filtre trilinéaire :

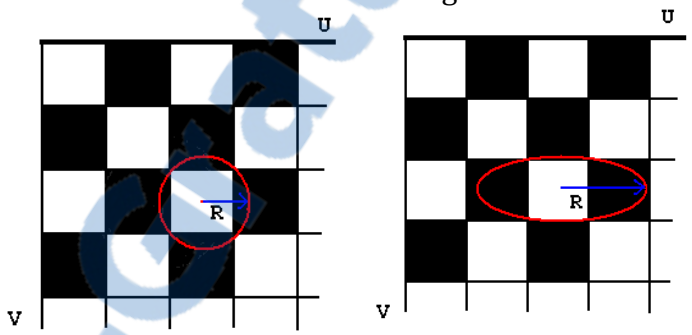
Filtre lié à l'utilisation du Mip-Mapping (dont nous parlerons dans la partie optimisation et effet). Il fonctionne comme un filtre bilinéaire entre deux niveaux d'un mipmap. Cette technique demande plus de ressource (temps) qu'un filtre

bilinéaire car elle traite deux fois plus d'informations (les deux niveaux de mipmap).

3.3.3.4 Le filtre anisotropique :

Est qualifié d'anisotrope ce qui n'a pas les mêmes propriétés dans toutes les directions. Un cercle projeté orthogonalement sur un plan parallèle au plan du cercle sera toujours un cercle. Projeté sur un plan non parallèle, un cercle donnera une ellipse comme l'illustre la figure suivante. Le filtrage anisotropique est plus réaliste que les autres car il considère les déformations subies par les projections de textures sur une surface 3D lors de placages non orthogonaux de textures.
Le principe de calcul de la couleur d'un pixel sera ensuite la même que pour le filtre bilinéaire. Il consiste en un calcul de la moyenne des couleurs des texels contenue dans la projection du pixel.

Une optimisation est d'approcher l'ellipse par un rectangle. La couleur du pixel est alors calculée en fonction de la moyenne pondérée des couleurs par rapport à la distance R des texels au centre du rectangle.



III.16- Face à l'objet. L'objet est vu avec un angle.

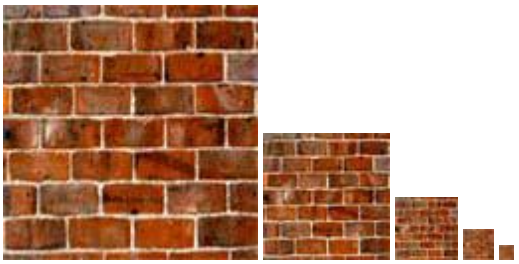
3.3.4 Optimisation et effets :

Le rendu des techniques présentées ci-dessus dénature les textures. Pour y remédier d'autre techniques sont utilisées :

3.3.4.1 Mip-Mapping :

Mip vient du latin Multim In Parvo [2] (ce qui signifie « beaucoup de choses dans un petit endroit »). Cette technique utilise un jeu de textures représentant la même image à des résolutions différentes, ce qui permet d'avoir une image de bonne qualité à n'importe quelle échelle.

L'ensemble des textures successivement réduites forment un Mipmap, chaque texture est appelé niveau de mipmap ou LOD (Level Of Detail). La largeur et la hauteur d'un niveau de mipmap sont respectivement égales à la moitié de la largeur et à la moitié de la hauteur du niveau de mipmap précédent. Le dernier de niveau de mipmap est celui dont la largeur et/ou la hauteur valent 1.

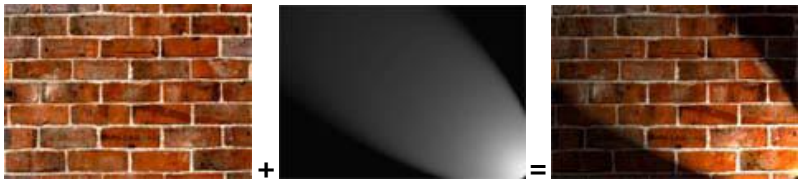


III.17- Différents niveaux de mipmap.

3.3.4.2 Le placage de texture multiple (multi-texturing) :

Le placage de texture multiple est le processus qui permet de mélanger par diverses opérations plusieurs textures appliquées sur une même surface 3D. Il existe en deux implémentations :

- Le placage de texture multiple en plusieurs passes. Un pixel est affiché en combinant à sa couleur précédente la couleur que lui attribue le mapping. Ce processus est réitéré autant de fois qu'il y a de textures appliquées, la projection d'une surface 3D doit donc être rendue plusieurs fois.
- Le placage de texture multiple en une seule passe. Les placages de textures s'enchaînent, les couleurs qu'ils attribuent au pixel sont successivement mélangées entre elles et la couleur délivrée à l'issue du processus et mélangée à la couleur précédente du pixel. La projection de la surface 3D n'est donc rendue qu'une seule fois.

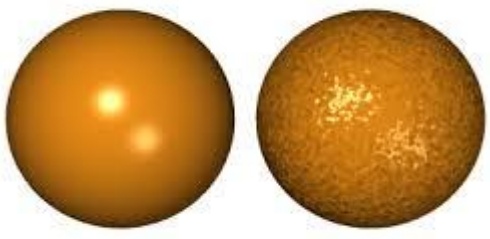


III.18- Effet du placage multiple de texture.

3.3.4.3 Le placage de texture bosselée (bump mapping) :

En 1978, Jim Blinn publie un article [7] présentant un algorithme mathématique permettant de simuler des effets de relief à une texture, illustré sur la figure. Cette technique a été appelée le bump mapping.

Cette technique est utilisée pour donner (par exemple) un effet de relief à l'eau, ou à l'écorce d'un arbre.



III.19- Sans Bump Mapping Avec Bump Mapping

3.4 Rendue

3.4.1 Elimination des parties cachées

L'élimination des parties cachées dépend de la modélisation adoptée modèle fil de fer, modèle à facettes, ou modèle volumique en arbres octaux.

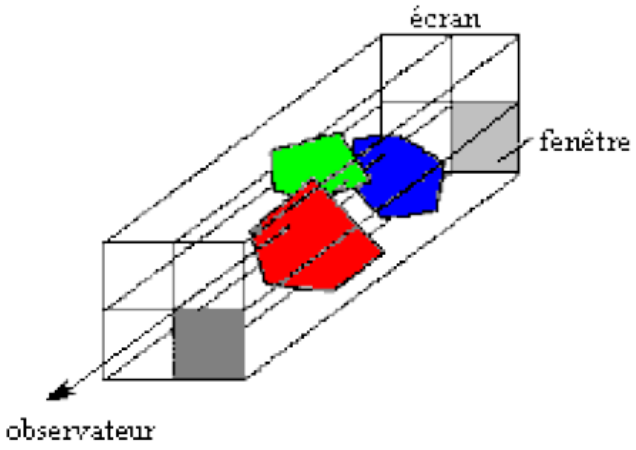
3.4.1.1 Modèle fil de fer

Les objets sont décrits par des lignes qui se coupent. Pour indiquer qu'une ligne se trouve devant une autre, on peut adopter une épaisseur de trait variant suivant les traits que l'on " voit " devant et ceux que l'on voit derrière. On peut aussi couper les traits qui sont derrière d'autres traits. En combinant les deux méthodes,

3.4.1.2 Modèle à facettes

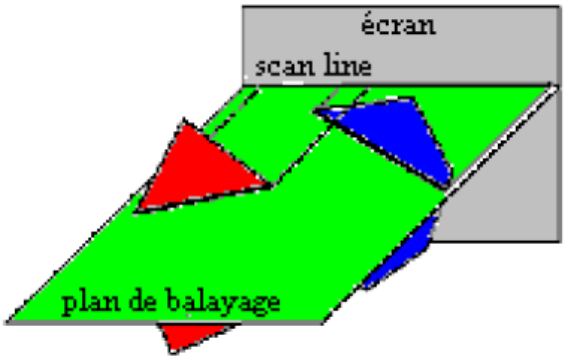
- On peut tout d'abord examiner la position des arêtes relativement aux facettes : chaque arête a(i) est examinée par rapport à toutes les facettes f(j); si la facette f(j) cache l'arête a(i), alors on découpe l'arête a(i) en partie visible et partie invisible. L'examen étant terminé pour la facette a(i), on peut alors afficher les parties visibles de celle-ci. Le temps de calcul est évidemment important si le nombre de facettes est grand.

- La méthode de Warnock consiste à effectuer un découpage récursif de l'espace .



III.20- Warnock

- La méthode de Watkins part d'un principe différent : la scène que l'on désire voir sur l'écran est découpée par des plans, appelés plans de balayage, perpendiculaires à l'écran.



III.21-Watkins

3.4.1.3 Le Z-Buffer : [1][3]

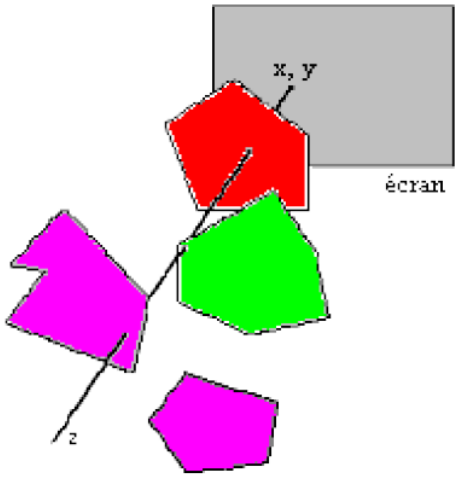
Le Z-Buffer sert à :

- Trier les facettes de la plus lointaine à la plus proche de la caméra
- Dessiner une à une (s'effectue en plusieurs étapes) :
  - Appliquer Les textures,
  - Effectué un test de transparence afin d'éventuellement mélanger la couleur du point en cours de traitements avec celle du point précédent sur la fenêtre de tampon d'affichage.
- recouvrir les facettes lointaines par les proches.

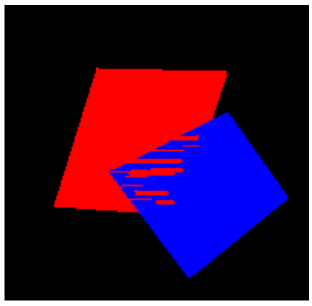
Cette technique naïve provoque des incohérences dans le cas d'intersections de facettes, dues à un buffer imprécis et des faces trop proches. Ces incohérences sont appelées flimmering ou z-fighting.

Algorithme du Z-Buffer :

- Pour chaque facette de la géométrie
- Calculer le centre de gravité
- Trier les facettes par rapport à la composante z du centre
- Pour chaque facette de la plus lointaine à la plus proche
- Tracer la facette



III.22- Fonctionnement du Z-Buffer



III.23- Effet du flimmering.

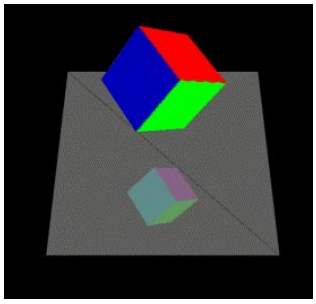
3.4.2 Transparence et opacité

Dans la réalité, un objet est rarement soit totalement transparent, soi totalement opaque. Il faut donc prendre en compte la réfraction de la lumière à travers une surface d'indice donné n.

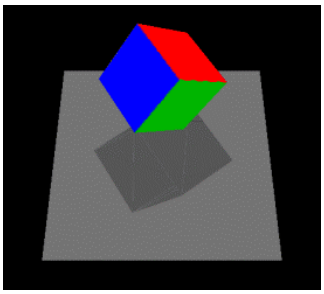
III.24- Transparence

3.4.3 Le pochoir (stencil buffer) :

Les principales applications du stencil buffer sont les ombres, les reflets et l'application d'un masque. Le reflet est le double d'un objet, recouvert d'une surface légèrement transparente pour donner l'illusion qu'il s'agit d'un reflet. Le stencil buffer joue un rôle de délimitation, pour l'ombre, c'est la projection de tout un objet sur un plan suivant la direction de la lumière. Cette opération se fait en multipliant la matrice de vue par une matrice de projection.



III.25- Réflexion d'un cube sur une surface.

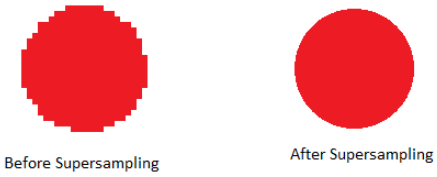


III.26- Ombre d'un cube.

3.4.4 Effet d'anti-crénelage (antialiasing) :

C'est une technique par laquelle on diminue l'effet « marche d'escalier » d'une image, en créant des dégradés de couleurs le long des contours pour les lisser. Le principe de l'anti-crénelage est de calculer une moyenne de couleurs. On distingue deux types d'anti-crénelage:

- L'edge antialiasing qui se fait en 2 passes, on affiche d'abord toute la surface 2D avant de n'en retracer que les bords. Dans cette deuxième passe, la couleur de chaque pixel affiché devient la moyenne de celle des pixels qui l'entourent.
- Le supersampling qui se fait en une seule passe. Les surfaces 2D sont rendues dans une surface plus grande (il existe un ratio x2, x4, x8, x16) que la surface de rendu finale. La couleur d'un pixel est calculée en faisant la moyenne des couleurs des pixels qui le représentent en haute résolution. Cette technique requiert énormément de mémoire vidéo. La figure 5.8 est un exemple de cette méthode.



III.27- Le supersampling

L'image étant ici en noir et blanc, les dégradés de couleur se font dans des niveaux de gris, mais il est bien entendu possible de le faire avec toutes les couleurs. L'anticrénelage présente cependant un inconvénient. En effet il traite les surfaces en 2D et peut nuire à l'effet de relief en créant des jonctions entre deux surfaces peu contrastées.

3.5 Conclusion :

Dans ce chapitre, nous avons abordé les trois bases pour faire un rendue réaliste en temps réel qui sont la lumière, la texture, et le rendue tels que :

- Pour la lumière on a vu les types de source lumière qu'on peut utiliser dans une scène 3D, les différents types matériaux et l'ombrage.
- Pour la texture on a vue les coordonnées de plaquage, les différents types de filtrage et les type de plaquage le Mip-Mapping et le bump mapping
- Et pour le rendue pour l'affichage sur l'écran est appliquer au moyen d'un algorithme de Z-Buffer.

CHAPITRE 4: **Implémentation et mise en œuvre**

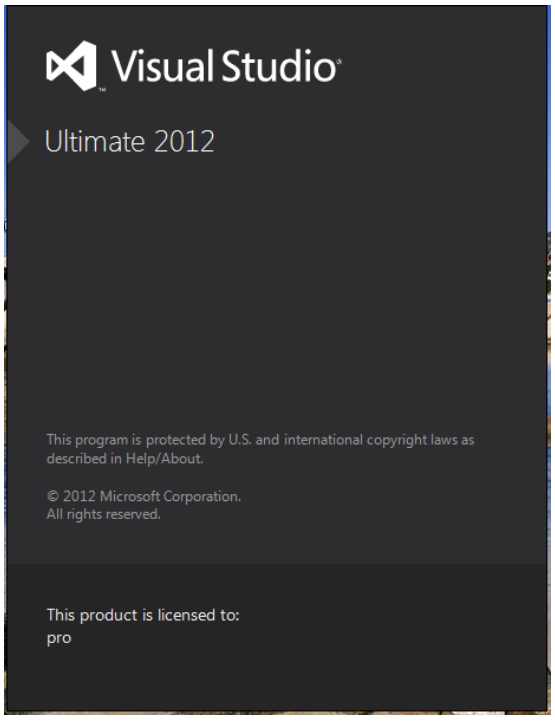
4.1 Introduction :

Dans notre projet, on va réaliser un générateur de navigation des mesh simple en 2d « rectangle , triangle …etc. . » , en 3D « cube , cône …etc. »et des mesh complexe sous forme des fichier (*.x « x files ») « voiture , terrain …etc. » dans une pipeline graphique (seine 3D) obéissant aux lois de la physique tels que l'illumination , l'ombrage , texture , rendue avec application d'un algorithme d'élimination des parties cachées et animation.

4.2 Choix du langage :

Les langages les plus utilisés dans la programmation des réalités virtuels sont « c++, java, Visual basic, et C# »

Dans notre système nous utilisons le **Microsoft Visual Studio C# 2012** avec la bibliothèque **XNA Game Studio** comme outil pour faciliter la programmation des taches graphiques.



IV.1- Microsoft visual studio 2012

4.2.1 Microsoft Visual C# :

La dernière version est née en 2013 mais nous utilisons la version 2012
Nous avons choisi **Microsoft Visual Studio C# 2012** parce qu'elle répond bien a nos exigences et nous offre :

- La syntaxe et le jeu d'instruction du langage c#.
- Les avantages d'une programmation orientée objet tel que l'encapsulation l'héritage, l'énumération et les classes partielles.
- Une forte utilisation de la machine virtuelle .Net qui gère automatiquement les ressources matérielles.
- Génère des fichiers exécutables, rapides et stables.

4.2.2 Microsoft XNA :

parfois présenté dans les médias Xbox Next-Generation Architecture, désigne une série d'outils fournis gratuitement par Microsoft qui facilite les développements de jeux pour les plates-formes Windows, Zune, Windows Phone 7, et Xbox 360 en réunissant un maximum d'outils en provenance de Microsoft et de ses partenaires (DirectX, Visual Studio, PIX, XACT).
Il contient principalement un framework, des outils d'intégrations de contenu, et la documentation nécessaire. L'IDE utilisé, à télécharger séparément, est Visual Studio.
Avec XNA, Microsoft est le premier constructeur à ouvrir la porte au développement indépendant sur sa console Xbox 360. Les jeux produits sont distribués via le Xbox Live.

4.3 Structure de System :

Notre application est une *« Windows Form Application »* mais on peux realiser ce Projet sur un Smartphone ou une appareil de jeux video professionnel XBOX 360. Et parmi les classe et les package qu'on a utilisées :

XNA Game Studio 4.0

Un profil est un ensemble de fonctionnalités implémenté dans le matériel. Le profil Reach supporte HLSL Shader Model 2.0 ; le profil HiDef, supporte quant à lui HLSL Shader Model 3.0.

Il existe deux types de profils, Reach and HiDef : l'un est conçu pour du matériel puissant aux fonctionnalités enrichies, l'autre pour du matériel plus courant.

- Reach est conçu pour couvrir tous les types de plates-formes. Il propose un ensemble d'outils et de fonctionnalités graphiques implémentés dans le matériel. Ce profil est conçu pour prendre en charge une vaste gamme de périphériques
- Le profil HiDef est conçu pour offrir les meilleures performances possibles et un ensemble de fonctionnalités graphiques sans égal. L'utilisation de ce profil est recommandée avec du matériel offrant des fonctionnalités graphiques avancées, comme par exemple, la Xbox 360 et un ordinateur Windows doté au minimum d'un GPU de classe DirectX 10

Description technique

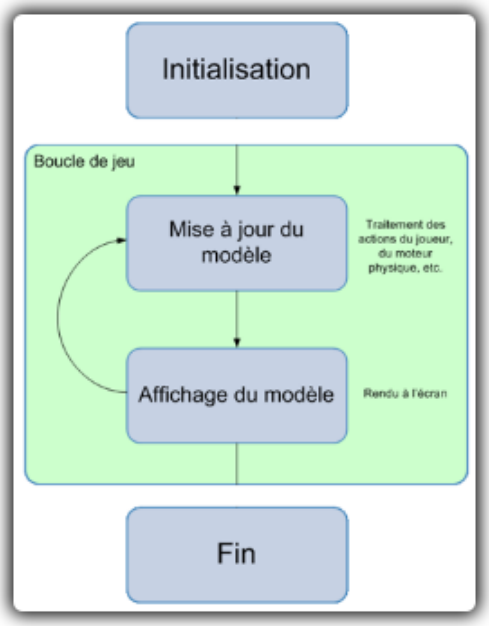
A travers cette partie, nous allons véritablement entrer dans la description des différents éléments des couches présentées dans la partie de description de l'architecture. Pour cela nous allons partir de la couche de plus haut niveau (couche jeux) pour terminer par les couches de plus bas niveau du Framework XNA.

III-A. Le Framework étendu

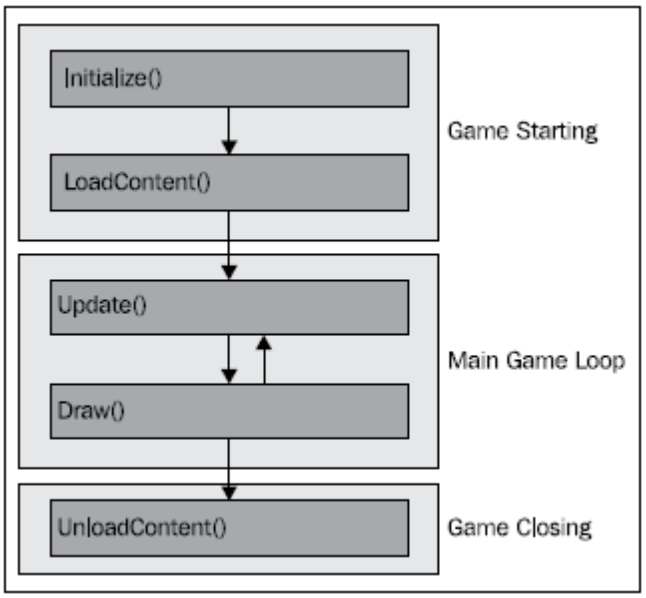
III-A-1. Le modèle applicatif

L'intention de ce modèle applicatif présent dans cette partie du Framework XNA est de faire abstraction de tous traitements habituels des jeux vidéo, à savoir le suivi de son cycle de vie. Avec XNA, pas besoin de s'occuper de créer la fenêtre, de gérer les messages, de créer les différents timers et horloge d'exécution, la seul chose sur laquelle le développeur se concentre, c'est le code de son jeu.

Les jeux vidéo en général respectent un schéma de base simple dans leur exécution :



IV.2- Architecture application avec XNA Game Studio



IV.3- Méthodes utilisées dans XNA Game Studio

Après avoir été initialisé, le cœur du jeu s'exécute en boucle (la boucle de jeu) en effectuant les opérations suivantes :

Mise à jour du modèle : prise en compte des actions du jeu, de actions utilisateurs, du moteur physique, etc.

Affichage du modèle à l'écran : Dans certains cas, cette boucle peut se voir ajouter une troisième étape qui correspond à une étape de post-traitement pour des effets après rendu.

Ce fonctionnement logique du jeu vidéo doit en général être traité par le développeur, grâce à XNA, toute cette architecture logicielle est prise en charge.

Lors de la création d'un jeu vidéo à travers l'IDE dédié, on remarque que la génération du code respecte un schéma particulier et base le projet sur un squelette de base. Une classe principale, dérivée de la classe Game permet d'utiliser de nombreux éléments conteneurs et autres services mais propose également cinq méthodes qui décrivent le cycle de vie du jeu et qui sont très utiles dans la simplification du développement :

Initialize

```
protected override void Initialize()
    {
        // TODO: Add your initialization logic here
        base.Initialize();
}
```

Cette méthode permet de créer les composants non graphiques, d'initialiser les paramètres du jeu, de créer les différentes instances nécessaires et de récupérer des informations sur la plateforme d'exécution.

LoadGraphicsContent

```
protected override void LoadGraphicsContent(bool loadAllContent)
    {
        if (loadAllContent)
        {
            // TODO: Load any ResourceManagementMode.Automatic content
        }
        // TODO: Load any ResourceManagementMode.Manual content
}
```

Cette méthode est appelée automatiquement à travers la méthode Initialize(). C'est à travers cette dernière que l'on charge les éléments et ressources nécessaires au jeu, par exemple les modèles 3d, les textures et autres éléments utiles.

UnloadGraphicsContent

```
protected override void UnloadGraphicsContent(bool unloadAllContent)
    {
        if (unloadAllContent)
        {
            // TODO: Unload any ResourceManagementMode.Automatic content
            content.Unload();
        }
        // TODO: Unload any ResourceManagementMode.Manual content
}
```

C'est à travers cette méthode que nous procédons à la libération des ressources, on décharge ainsi les éléments de la mémoire, on « dispose » les différents éléments graphiques utilisés.

Update

```
protected override void Update(GameTime gameTime)
    {
        // Allows the game to exit
        if (GamePad.GetState(PlayerIndex.One).Buttons.Back == ButtonState.Pressed)
            this.Exit();
        // TODO: Add your update logic here
        base.Update(gameTime);
}
```

des actions du jeu. C'est clairement une méthode présente dans la boucle de jeu qui prends en comptes les actions et entrées de l'utilisateur, du moteur physique etc...

Draw

```
protected override void Draw(GameTime gameTime)
    {
        graphics.GraphicsDevice.Clear(Color.CornflowerBlue);
        // TODO: Add your drawing code here
        base.Draw(gameTime);
}
```

Cette méthode correspond à l'étape d'affichage du modèle, il s'agit réellement de celle qui effectue le rendu à l'écran à travers le GraphicsDevice géré par XNA.

Le content pipeline : Les contenus multimédia graphiques et sonores occupent une place plus qu'importante dans les jeux actuels et l'incorporation de ces éléments dans le projet n'est pas chose aisée. La plupart du temps, l'accent est mis sur la recherche de l'exporter approprié, de l'outil nécessaire pour effectuer une tâche particulière. Dans un environnement de travail en équipe, plusieurs personnes travaillent en ce sens pour réussir à inclure ces éléments dans le projet et nécessite des moyens conséquents et réduisent la flexibilité de travail.

Le content pipeline faisant partie du Framework XNA propose une approche différente et améliorée.

Il ne vous propose pas les outils de création de contenu (récemment l'apparition de l'outil de Softimage XSI Mod Tools gratuit et dédié à la création pour le jeu) mais cependant, une fois de plus, il facilite les actions du développeur en prenant en charge les étapes nécessaires à l'exploitation du contenu.

Les importers

Lorsque vous ajoutez du contenu à votre jeu vidéo pour la première fois, vous avez besoin de choisir un importer. Cet importer est responsable de récupérer les données, de les traiter et de les normaliser pour la suite du traitement, il vérifie également leur cohérence. Il n'est donc plus nécessaire de s'occuper de l'importation et des soucis liés, l'importer s'occupe de tout à condition d'être rigoureux dans la création de contenu (fichier respectueux du format).

Il existe de nombreux importers :

- Autodesk FBX. Ce format présente la possibilité de pouvoir sauvegarder des scènes 3D complètes à travers la plupart des outils de créations graphiques tant open source que commerciaux, s'avérant ainsi une bonne alternative en attendant un réel standard dans les formats de fichier 3D.
- XAP , Les fichiers audio de format XAP sont issus de l'outil XACT (Microsoft Cross-Platform Audio Creation Tool) qui sait traiter les fichiers sonores dans des formats standards.

Le content DOM

Une fois que les importers ont réalisé leur tâche d'importation, les données sont présentes dans le content DOM. Le terme DOM est ici simplement utilisé pour représenter une collection de classes et de schémas (de la même manière qu'un fichier XML).

Les données présentes dans le content DOM sont des données fortement typées ce qui signifie qu'elles sont stockées sous un format bien connu ainsi, par exemple,

une collection de sommets ou des données de textures auront le même format dans ce content DOM.

Ce point est important pour la suite du traitement de ces données, ce résultat peut d'ailleurs être inscrit sur le disque au format XML pour le débogage uniquement, il permet un traitement homogène dans les autres étapes d'exploitation des contenus.

Les processors

Un processor est chargé de récupérer les données présentes dans le content DOM et de créer un objet qui pourra être utilisé en exécution. Cet objet peut alors être simple comme un processor model ou complexe en combinant plusieurs de ces processors.

Par défaut, le Framework XNA propose plusieurs processors de base tels :

- Le processor model : utile pour les objets simples avec textures
- Le processor Texture2D : utile pour les sprites ou pour combiner avec les models
- Le processor Effect : utile pour manipuler les matériaux des models

La plupart du temps donc, ces processors sont utilisés en combinaison et sont très utiles pour effectuer des tâches importantes de traitements des données en plus du fait qu'ils sont très permissifs et facilement personnalisables pour exploiter n'importe quel contenu car en utilisant le content DOM qui normalise les données, ces processors peuvent traiter tout type de fichiers de contenus.

Le content building

Cette partie du content pipeline est également important et elle est géré par le coordinateur de construction du projet. Ainsi lorsque le projet est compilé, le contenu est construit et enregistré sur le disque de la même façon que votre code.

Le content manager

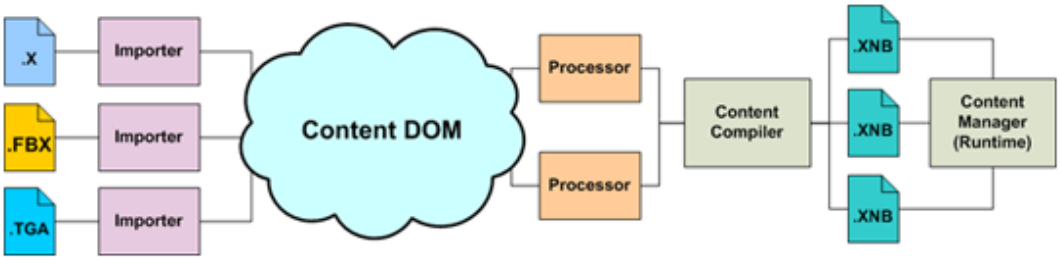
Le content manager est véritablement l'élément qui vous permet d'utiliser vos contenus dans votre jeu, il s'occupe de charger rapidement ces éléments en mémoire.

Son utilisation se veut très simple et voici un exemple de code qui vous permettra de récupérer un élément que vous avez ajouté à votre projet.

```
ContentManager contentManager = new ContentManager(Services);
Model model = contentManager.Load("element");
```

Il est donc très aisé d'utiliser du contenu à votre jeu vidéo en utilisant cette technologie qu'est le content pipeline intégré à XNA et ceci devrait encore s'améliorer au fil des versions.

Voici un schéma qui résume le fonctionnement global du content pipeline :



IV.4- Architecteur XNA game Studio

IV.5- Application(1)

IV.6- Application(2)

1- **Z_Buffer :** Est un algorithme de rendue réaliste le principe de l'application est d'éliminer les parties cachées et le facteur principale pour l'application de cet algorithme est la variable Z de profondeur de chaque facette existant dans la scène , alors dans notre système on fixe le Z=10f comme exemple et l'affichage devenait tout les objets qui existent loin de cette valeur ne sont pas afficher .

4.4 Conclusion

Dans le dernier chapitre nous avons abordé l'environnement de l'implémentation de sorte qu'on a utilisé la gamme Visual studio C# avec l'API standard Microsoft DirectX et le langage XAML pour la représentation des interfaces graphiques.

Et on a défini les différents types de classe de notre système avec la représentation de dbml et des Screenshot de notre application.

Conclusion Générale

Dans notre modeste étude nous avons réalisé un système de navigation dans une scène en utilisant l'API XNA Game Studio qui peut être utilisé dans plusieurs domaines comme simulation industrielle, architecture et programmation des jeux vidéo.

Mais le problème de ce système est qui n'est pas réaliste comme les images de synthèse en 3D précalculé, parce qu'on a utilisé la méthode de Rastérisation et nous souhaiterons bien que les étudiants des prochaines promotions amélioreront les résultats de l'affichage de la scène sur l'écran.

Ce travail peut être complété en rajoutant :

- Un éditeur d'objets pour permettre aux utilisateurs de générer leurs propres objets.
- Faire une conception d'un jeu, en rajoutant les characters et les règles pour jouer avec un système sui permet d'enregistrer les traces de navigation dans la scène 3D sous forme d'un fichier vidéo.
- Régler l'interface graphique avec le langage XAML en utilisant Microsoft Blend.
- Essayer d'appliquer le pixel shader au lieu des vertex shader pour améliorer le placage de la texture.
- Essayer d'appliquer la méthode de Raytracer pour améliorer le rendue en temps réel.

Et on aimerais bien que le projet de fin d'étude sera réalisé avec l'API XNA Game Studio et la modélisation sera implémenter dans Softimage XSI qui est free pour les étudiant.

Enfin nous espérons que ce travail sert comme document de base pour les prochaines promotions dans le domaine de la réalité virtuel et de la CAO.

Reference

[I.1] W. Barfield, D. Zeltzer, T. Sheridan, and M. Slater. Presence and performance within virtual environments. In *Virtual environments and advanced interface design*, pages 473–513, 1995.

[I.2] R. A. Bolt. Put-that-there : voice and gesture at the graphics interface. In *7th annual conference on Computer graphics and interactive techniques*, pages 262–270, 1980.

[I.3] A. Bowman, E. Kruijff, and I. Laviola, J. Poupyrev. *3D User Interfaces : Theory and Practice*. Addison-Wesley, 2005.

[I.4] D. A. Bowman. *Interaction Techniques for Common Tasks in Immersive Virtual Environments : Design, Evaluation, and Application*. PhD thesis, Georgia Institute of Technology, 1999.

[I.5] D. A. Bowman, D. Koller, and L. Hodges. Travel in immersive virtual environments : An evaluation of view-point motion control techniques. In *Proceedings of IEEE Virtual Reality Annual International Symposium*, volume 7, pages 45–52, 1997.

[I.6] G. Burdea and P. Coiffet. *La réalité virtuelle*, chapter 2, pages 243– 251. Hermès-Paris, 1993.

[I.7] M. F. Deering. Holosketch : a virtual reality sketching/animation tool. In *Proceedings of CHI'95*, pages 220–226, 1995.

[I.8] T. T. Elvins, D. R. Nadeau, and D. Kirsch. Worldlets : 3d thumbnails for 3d browsing. In *SIGCHI conference on Human factors in computing systems*, pages 163–170, 1998.

[I.9] S. Frees and G. Kessler. Precise and rapid interaction through scaled manipulation in immersive virtual environments. In *In IEEE Virtual Reality 2005*, pages 99–106, March 2005.

[I.10] P. Fuchs, B. Arnaldi, and J. Tisseau. *La réalité virtuelle et ses applications*, Chapter 1, pages 3–52. Les Presses de l'Ecole des Mines de Paris, 2003.

[I.11] J. Grosjean, J.-M. Burkhardt, S. Coquillart, and P. Richard. Evaluation of the command and control cube. In *Proceedings of the Fourth International Conference on Multimodal Interfaces (ICMI 2002), IEEE Press*, pages 14–16, 2002.

[I.12] J. Grosjean and . Coquillart, S. . Command & control cube : a shortcut paradigm for virtual environments. In *In Proceedings of IPTEGVE' 01*, 2001..

[I.13] Y. Kitamura, T. Higashi, and F. Kishino. Virtual chopsticks : Object manipulation using multiple exact interactions. In *IEEE Virtual Reality*, pages 198–204, 1999.

[I.14] J. Liang and M. Green. Jdcad : a highly interactive 3d modeling system. In *Computer Graphics (SIGGRAPH'93 Proceedings)*, pages 499–506, 1994.

[I.15] A. Olwal and S. . Feiner. The flexible pointer : An interaction technique for selection in augmented and virtual reality. In *User Interface Software and Technology (UIST' 03) Conference Supplement, ACM Press*, pages 81–82, 2003.

[I.16] J. S. Pierce, A. Forsberg, M. J. Conway, S. Hong, R. Zeleznik, and M. R. Mine. Image plane interaction techniques in 3d immersive environments. In *Proceedings of 1997 Symposium on Interactive 3D Graphics*, pages 39–43, 1997.

[I.17] J. S. Pierce, B. C. Stearns, and R. Pausch. Voodoo dolls : Seamless interaction at multiple scales in virtual environments. In *Proceedings of the ACMSymposium*
on Interactive 3D Graphics, pages 141–146, 1999.

[I.18] I. Poupyrev, S. Weghorst, M. Billinghurst, and T. Ichikawa. The gogo interaction technique non-linear mapping for direct manipulation in vr. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST '96) Press*, pages 79–80, 1996.

[I.19] A. Steed and C. Parker. 3d selection strategies for headtracker and non-head tracked operation of spatially immersive displays. In *Immersive Projection Technology*, 2004.

[I.20] L. Sternberger and D. Bechmann. Deformable ray-casting interaction technique. In *In YoungVR February*, 2005. [72] R. Stoackley, M. J. Conway, and R. Pausch. Virtual reality on a wim : Interactive worlds in miniature. In *Proceedings of CHI : Human Factors in Computing Systems*, pages 265–272, 1995.

[I.21] D. J. Sturman, D. Zeltzer, and S. Pieper. Hands-on interaction with virtual environments. In *ACM Symposium on User Interface Software and Technology*, pages 19–24, 1989.

[I.22] J. Tisseau. *Réalité virtuelle : autonomie in virtuo*. PhD thesis, Université de Rennes I (France), 6 décembre 2001.

[I,23] G. Wesche and M. Droske. Conceptual free-form styling on the responsive workbench. In *Proceedings of ACM Symposium on Virtual Reality Software and Technology*, pages 83–91, 2000.

[I,24] L. Yan, R. Allison, and S. Rushton. New simple virtual walking method-walking on the spot. In *Symposium on Immersive Projection Technology*, 2004.

[I,25] D. Zeltzer. Autonomy, interaction, and presence. *Presence : Teleoperators and Virtual Environments*, 1(1) :127–132, 1992.

[I,26] C. Hand. A survey of 3d interaction techniques. In *Proceedings of Computer Graphics Forum*, pages 269–281, 1997.

[I,27] R. H. Jacoby and S. Ellis. Using virtual menus in a virtual environment. *Proceedings of SPIE : Visual Data Interpretation, Alexander, editor*, 1668 :39–48, 1992.

[I .28] Jacques TISSEAU : *Réalité Virtuelle -autonomie* in virtuo- ;Document de Synthèse , 2001

[II.1] Synthèse d'images *Transformations 2D et 3D* par Alain Boucher

[II.2] Graphisme 3D UFR IMA

[II.3] Formation et Analyse d'Images *Coordonnées Homogènes* par James L. Crowley

[II .4] Cour Infographie *licence* par Ms benziane

[II.5] Informatique Graphique Ecole Polytechnique Fédérale de Lausanne par Daniel Thalmann

[II.6] Représentation des objets 3D *Modélisation 3D et Synthèse* par Fabrice Aubert Master Info - Parcours IVI , Université Lille I – IEEA, 2009-2010

[II.7] http://www.runes-mmorpg.com/index.php/en/blog/70-animations-par-squelette-sur-android-en-opengl.html

[II,8] *Stéphanie Recco et Xavier Larrodé : Octree Textures for Painting and*

Rendering Textures on Unparameterized Models, Diaporama Master MM U.Bordeaux 1

[II.9] Nicolas Roussel : Structures de données : *Diaporama* (Univ. Paris-Sud – CNRS) & INRIA

[II.10] Claudia NEGULESCU, Interpolation et approximation : Courbes de Bézier , CMI, Université de Provence : 2007-2008

[II.11] Claudio LOBOS YÁÑEZ : Amélioration des Techniques de Génération de maillages 3D des structures anatomiques humaines pour la Méthode des Éléments Finis , THÈSE de DOCTORAT , 2009

[III.1] Brard Axel – Miara Jeremy : Moteur3D Licence Informatique 2002 – 2003
[III.2] Ufrima : Rendu réaliste
[III.3]Rendue réaliste : D226_Chapitre-4.Pdf
[III.4] François Faure : Textures, Résume.

Annexe

Dot Net :

Est le nom de la nouvelle plateforme de développement de Microsoft. Les applications développées pour cette plateforme sont indépendantes :

- du système d'exploitation et de l'architecture matérielle sur laquelle elles sont lancées. Cependant il faut une implémentation du CLR (Common Language Runtime) pour cette configuration,
- du langage de programmation utilisé. En effet, quel que soit le langage de programmation utilisé, le code source de l'application est compilé en langage intermédiaire appelé MSIL (Microsoft Intermediate Language), CIL? (Common Intermediate Language) ou en abrégé IL (Intermediate Language).

C Sharp :

A été développé par la société Microsoft, et notamment un de ses employés, Anders Hejlsberg, pour la plateforme .NET (point NET / dot NET).

Ce langage est orienté objet, avec un typage fort. Il est très proche du langage Java. Il est précompilé en MSIL (Microsoft Intermediate Language), puis exécuté sur une machine virtuelle, ou compilé en code natif à l'exécution. Il dispose d'un ramasse-miettes (garbage collector). Il utilise l'API .NET en remplacement des MFC (Microsoft Foundation class).

Comparaison :

Profils	**Reach**	**HiDef**
Plates-formes	Windows Phone 7, Xbox 360 et tout ordinateur exécutant Windows doté d'un GPU de classe DirectX 9 supportant au moins Shader Model 2.0.	Xbox 360 et tout PC tournant sous Windows doté au minimum d'un GPU de classe DirectX 10 (ou équivalent). Pour plus d'informations, consultez le paragraphe ci-dessus.
Shader Model	2.0 (Windows Phone ne prend toutefois pas en charge les nuanceurs personnalisés.)	3.0 (Xbox 360 prend en charge les extensions de nuanceurs personnalisées comme vfetch, non disponibles sous Windows.)
Taille de texture maximale	2,048	4,096
Taille de mappage de cube maximale	512	4,096
Taille de texture de volume maximale	Les textures de volume ne sont pas prises en charge.	256
Textures non puissance de deux	Oui avec limitations : les fonctionnalités de mode d'adressage wrap, de mappage MIP, de compression DXT les textures non puissance de deux ne sont pas prises en charge.	Oui

Mappages de cube non puissance de deux	Non	Oui
Textures de volume non puissance de deux	Les textures de volume ne sont pas prises en charge.	Oui
Nombre maximal de primitives par appel Draw	65,535	1,048,575
Nombre maximal de flux de vertex	16	16
Stride de flux de vertex maximal	25	255
Formats des mémoires tampon d'index	16 bits	16 et 32 bits
Formats des éléments de vertex	Color, Byte4, Single, Vector2, Vector3, Vector4, Short2, Short4, NormalizedShort2, NormalizedShort4	Tous les formats d'élément de vertex Reach, ainsi que HalfVector2 et HalfVector4.
Formats de texture	Color, Bgr565, Bgra5551, Bgra4444, NormalizedByte2, NormalizedByte4, Dxt1, Dxt3, Dxt5	Tous les formats de texture Reach, ainsi que Alpha8, Rg32, Rgba64, Rgba1010102, Single, Vector2, Vector4, HalfSingle, HalfVector2 et HalfVector4. Les formats de texture à virgule flottante ne prennent pas en charge le filtrage.
Formats de texture de vertex	La texturation de vertex n'est pas prise en charge.	Single, Vector2, Vector4, HalfSingle, HalfVector2, HalfVector4
Formats de cibles de rendu	Appelez QueryRenderTargetFormat pour connaître les conditions de prise en charge.	Appelez QueryRenderTargetFormat pour connaître les conditions de prise en charge.
Cibles de rendu multiple	Non	Jusqu'à 4. Toutes doivent avoir la même profondeur de couleur. Semi-transparence et masques d'écriture indépendants par cible de rendu pris en charge.
Requêtes d'occlusion	Non	Oui
Alpha blend séparé	Non	Oui

Liste de bibliothèques 3D

Les bibliothèques 3D sont à la base des moteurs 3D. Les bibliothèques 3D sont également capable de réaliser des tracés 2D (perte d'une dimension), comme avec les moteurs de rendu 3D.

- OpenGL : spécification d'API permettant le tracé d'éléments 3D. Plusieurs bibliothèques implémentent cette spécification, la plus connue étant Mesa 3D (voir la page sur OpenGL). Cette API est disponible pour de nombreux

systemes d'exploitation tels que les différents Unix, BSDs, Linux, Mac OS X, et même Microsoft Windows.
- OGRE : un moteur 3D libre pouvant utiliser les API OpenGL ou Direct3D.
- Direct3D : API permettant le tracé d'éléments 3D. C'est une partie intégrante de DirectX, API pour la programmation d'application multimédia.
- Java 3D : extension standard aux API de Java SE permettant le tracé d'éléments 3D
- Windows Presentation Foundation : utilisant Direct3D.
- La plateforme Flash d'Adobe dispose de bibliothèques pour dessiner en 3D.

- Après de nombreuses tentatives plus ou moins réussies, des propositions émergent actuellement pour apporter la 3D sur le Web:

- WebGL : futur standard basé sur OpenGL ES 2.0.
- O3D : une approche plus haut niveau développée par Google Labs.
- CAO : TracePartsOnline.net, bibliothèque de composants comprenant plus de 350 catalogues de fabricants et + 100 millions de plans CAO 2D et modèles 3D gratuits diffusée par TraceParts

Évolution des cartes graphiques PC :
De plus en plus de fonctionnalités du pipeline graphique ont été délégués à la carte graphique
- **1995-1998** : placage de texture et Z-Buffer
- **1998** : multi-textures
- **1999-2000** : transformations géométrique et éclairement
- **2001** : « vertex shader » programmables
- **2002-2003** : « pixel shader » programmables
- **2004** : Modèle shader 3.0 et support des couleurs 64-bits
- **2005** : Multi-GPU

Animation 3D
L'**animation 3D** est une technique d'animation numérique équivalant à l'animation en volume dans un monde virtuel.

Cel-shading

Le Cel-shading permet, à partir d'une animation 3D, de générer des images ayant l'aspect d'un dessin animé contrairement à l'aspect généralement photoréaliste de la synthèse d'image.

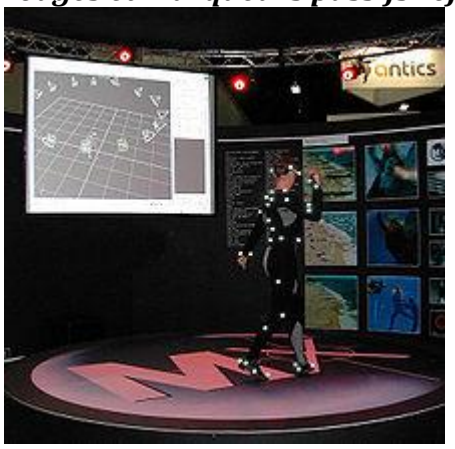
Capture de mouvement

Tous les systèmes de capture de mouvement du corps humain sont constitués d'un dispositif matériel (capteurs, caméras USB ou réseau de caméras ethernet) relié à une application serveur dont le rôle est d'extraire les données brutes matérielles pour les traiter (filtrage, calculs géométriques et d'interpolation) puis les stocker ou les transmettre en temps réel à une application cliente universelle de visualisation, généralement MotionBuilder. Ce logiciel (développé

à l'origine par la société Kaydara, rachetée aux environs de 2008 par Autodesk) est devenu un standard industriel puisqu'il interface via plugin la quasi-totalité des systèmes professionnels, et que son format FBX (pour FilmBoX, nom originel du logiciel) a été imposé comme standard d'échange et d'animation entre 3dsMax, Maya et XSI.

Capture optique

Basée sur caméras infrarouges et marqueurs passifs réfléchissants



Un danseur portant une combinaison

avec des marqueurs passifs réfléchissants,

Basée sur la technologie Kinect

La Kinect est un dispositif de capture dont la technologie sophistiquée est utilisée pour des applications ludiques grand public. Une trame de lumière infrarouge est projetée sur les objets, image que le capteur va traiter pour en reconstituer la profondeur (plus un point infarouge est gros, plus il est près). À la différence des systèmes optiques traditionnels avec marqueurs, cette technique permet de n'utiliser qu'une seule caméra. Conçue à l'origine pour la Xbox, ce périphérique a très vite été détourné pour fonctionner sur PC, puis officiellement par Microsoft avec Kinect for Windows.

Basée sur réseau de caméras vidéo en lumière naturelle
La société Organic Motion commercialise un dispositif de capture sans marqueurs professionnel de haute qualité utilisant un réseau d'une vingtaine de caméras vidéo rapides. Système temps réel, avec affichage des acteurs en *Voxel voxels*.

Basée sur des cellules photosensibles et marqueurs actifs

Un système de marqueurs actifs de haute résolution

fournit en temps réel des positions infra-millimétriques.

Technologie 3D Vision

L'affichage des images en 3D repose sur le principe de la stéréoscopie. Au lieu d'afficher une seule image pour les 2 yeux, on projette une image séparée pour chaque œil du spectateur. La technologie employée pour séparer les images et les projeter aux yeux détermine la qualité d'image et le rendu de la 3D. Les fabricants de téléviseurs proposent désormais des écrans plats capables d'afficher des images (Blu-ray, programme télé, jeu vidéo, etc.) avec un effet de profondeur appelé communément 3D.

Equipement nécessaire pour regarder des programmes en 3D

- le téléviseur doit être compatible 3D.
- La mention 3D Ready s'applique aux écrans capables d'afficher des images en 3D.
- une paire de lunettes actives par spectateur, synchronisée à l'écran.
- lecteur Blu-ray 3D, console de jeu ou PC 3D.

Lunettes 3D : des modèles anaglyphes aux lunettes actives

1- Lunettes anaglyphes

Il s'agit de la génération de lunettes 3D pionnières, utilisée depuis les premiers films produits en relief (années 1920). Cette technologie a pour principal avantage d'être peu coûteuse (les lunettes à verres rouge et vert sont fabriquées en carton) et compatible avec tous les téléviseurs plats ou à tube. Deux images légèrement décalées sont diffusées,

projetées sur chaque œil. Les principaux inconvénients : mauvaise colorimétrie, phénomène d'images fantômes (ghosting) et fatigue visuelle. Le rendu est sans comparaison avec l'aboutissement de la " 3D Full HD ", en termes de qualité d'image et d'effet de profondeur.

2- Lunettes passives (polarisantes)

Plus récentes que les lunettes anaglyphes, les lunettes passives permettent un meilleur rendu 3D. Elles sont utilisées dans de nombreuses salles de cinéma projetant des films en 3D. Elles ne nécessitent pas de piles. On les appelle aussi lunettes polarisantes. L'image affichée sur l'écran est en fait constituée de deux images tramées. Une ligne sur deux est donc destinée à un oeil pendant que l'autre ligne est destinée à l'autre oeil. Pour chaque ligne de l'écran, la lumière émise est polarisée de manière inverse à la ligne précédente. Ainsi, les lunettes ne laissent passer sur chaque oeil qu'une ligne sur deux. Avec les lunettes passives, chaque image affichée à l'écran est constituée à la fois de l'image destinée à l'oeil droit et de l'image destinée à l'oeil gauche. Contrairement aux lunettes actives, les lunettes passives sont incapables d'afficher de la " 3D FULLD HD " puisque les 1080 lignes sont divisées entre chaque œil (540).

3- Lunettes actives

Les lunettes actives aussi appelées lunettes à obturateur (shutter), embarquant des piles sont utilisées par certaines salles de cinéma 3D et par les téléviseurs 3D de salon. Les lunettes sont synchronisées avec le téléviseur ou l'écran de projection, et chaque verre se ferme et s'ouvre alternativement, jusqu'à plusieurs centaines de fois par seconde (100 fois par seconde sur un téléviseur 100Hz). Le cerveau n'a plus qu'à associer les deux images capturées sur des points de vue différents pour recréer l'effet 3D. Les verres sont moins sombres que les lunettes passives et anaglyphes et permettent un meilleur rendu des images 3D. Ce système a été retenu pour la norme Blu-ray 3D.

3D dans les jeux vidéo

Le jeu vidéo PC ou sur console de salon constitue une excellent source d'image pour les téléviseurs 3D. En reliant un PC portable ou fixe labellisé 3D (doté d'une carte graphique compatible 3D Vision) à un téléviseur 3D, on peut jouer avec un effet de profondeur sur la plupart des jeux vidéo PC récents. Du côté des consoles de jeux, seule la PS3 de Sony est compatible avec la 3D. Cette console est capable,

grâce à une mise à jour de son firmware, de lire des films Blu-ray 3D et d'afficher certains jeux de son catalogue en 3D, connectée à un téléviseur 3D.

Unity :

Unity est un logiciel 3D temps réel et multimédia ainsi qu'un moteur 3D/2D et physique utilisé pour la création de jeux en réseau, d'animation en temps réel, de contenu interactif comportant de l'audio, de la vidéo et des objets 3D/2D.

Description technique

Le logiciel a la particularité d'utiliser un éditeur de script compatible mono (C#), UnityScript (un langage proche du JavaScript et inspiré d'ECMAScript) et Boo au lieu de Lua très utilisé dans les jeux vidéo. Son approche orientée asset, par le biais d'un EDI dédié, le différencie des moteurs comme le Quake engine dont les éléments centraux sont les codes sources. Il est l'équivalent du logiciel de création Director pour la 2D qui utilise Lingo. Il se rapproche plus pour la 3D des logiciels tel que Shiva, Virtools, Cheetah3D. Parmi les logiciels d'animations, il ne permet pas la modélisation mais permet de créer des scènes supportant des éclairages, des terrains, des caméras, des textures.

Plates-formes supportées

Le logiciel de conception développé d'abord pour la plate-forme Mac a été porté sous Windows et permet d'obtenir des applications compatibles Windows, Mac OS X, iOS, Android, Wii, PlayStation 3, Xbox 360, Windows Phone 8, PlayStation Mobile nativement, dans une page web grâce à un plugin, ou plus récemment -depuis la version 3.5- le format Flash d'Adobe.

Compatibilité

Il est capable d'importer de nombreux formats 3D (Maya, Cinema 4D, Cheetah3D (en), FBX), des ressources variées : des textures Photoshop, PNG, TIFF, audios, vidéos) qu'il optimise par l'utilisation de filtres.

Unity possède une large palette de déploiement :

- il est compatible avec les API graphiques Direct3D, OpenGL et Wii ;
- les navigateurs web peuvent, grâce au plugin Unity Web Player, afficher les productions du moteur ;
- il est compatible QuickTime et utilise en interne le format Ogg Vorbis.

List des Figures

List des Abréviations

Chapitre 1 :

NIVE : Environnement virtuel non-immersif

SIVE : Environnement virtuel semi-immersif

FIVE : Environnement virtuel totalement immersif.

FPS: Frame per Second

***Sensorimotrice*:** qui concerne les sensations et l'activité motrice

MIT: Massachusetts Institute of Technology

Miniature : Espace limite dans la seine 3D

Chapitre 2 :

Coordonnée homogène : afin de faire les transformations géométriques.

CAO : conception assistée par ordinateur

B-Rep: Boundary Representation

CSG: Arbre Constructive solid Geometry

Chapitre 3:

Alpha : Composante de transparence, défini entre 0 et 1.
Antialiasing : Technique par laquelle on diminue l'effet d'escalier des images, en créant des dégradés de couleurs le long des contours, pour les lisser.
Frame rate(FPS) : Nombre d'image par seconde.
Mapping : Placage de texture.
Mip-Mapping : Multim In Parvo (beaucoup de choses dans un petit endroit).
Mesh : Collections d'objets.
NURBS : Non-Uniform Rational B-Spline. Courbe plane définie par des points appelés « nœuds », à la manière des courbes de Bézier.
Pixel : Un point à l'écran.
Vertex : Un point d'un objet.
Vertex Buffer (VB) : Buffer contenant un ensemble de points.
Vertices : vertex au pluriel.