République **A**lgérienne **D**émocratique et **P**opulaire
Ministère de l'**E**nseignement **S**upérieur et de la **R**echerche **S**cientifique

UNIVERSITE **M**OULOUD **M**AMMERI DE **T**IZI-**O**UZOU

FACULTE DU **G**ENIE **E**LECTRIQUE ET D'INFORMATIQUE
DEPARTEMENT D'ELECTRONIQUE

Mémoire de Fin d'Etudes de MASTER ACADEMIQUE

Domaine : **Sciences et Technologies**
Filière : **Génie Electrique**
Spécialité : **Réseaux et Télécommunications**

Présenté par
HAMMAZ ABDERRAZAK

HADJ SAID MEZIANE

Thème

PLAQUAGE DE TEXTURE 2D SOUS IDL

Mémoire soutenu publiquement le 29 /09/ 2015 devant le jury composé de :

Mr T. OTMAN CHERIF
UMMTO, President

Mr M. LAZRI
UMMTO, Encadreur

Mr F. OUALLOUCHE
UMMTO, Examinateur

Mr Y. ATTAF
UMMTO, Examinateur

REMERCIMENTS

Ce travail a été effectue dans le cadre de la préparation du diplôme de master académique en réseaux et télécommunicationsau niveau de l'université Mouloud Mammeri de Tizi-Ouzou.

On tient à exprimer nos profondes gratitudes et notre immense respect à notreco-encadreur ***Mr. BIR Mehdi*** *pour la qualité de son encadrement, sa disponibilité.*

Nos vifs remercîments s'adressent égalementà notre promoteur ***Mr. LAZRI Mourad*** *pour ses conseilsprécieux et son soutien affectif durant notreétude et réalisation de ce projet.*

Nos remerciements les plus vifs s'adressent aussi aux messieurs le président et les membres de jury d'avoir accepté d'examiner et d'évaluer notre travail.

Dédicaces

Dédicace de Abderrazak.

Je dédie ce modeste travailàmes très chers parents qui ont œuvrés pour ma réussite, leur soutien, tous les sacrifices consentis, et leurs précieux conseils, pour toute leur assistance et leur présence dans ma vie, reçoivent a travers ce travail soit-il, l'expression de mes sentiments et mon éternelle gratitudes.

*A mes très chers frère et sœur: **ZOHEIR et FAIZA***

A toute ma famille et mes amis (e), et à tous ceux qui me sont chers.

Dédicace de Meziane.

Je dédie ce modeste travail à mes très chers parents qui ont œuvrés pour ma réussite, leur soutien, tous les sacrifices consentis, et leurs précieux conseils, pour toute leur assistance et leur présence dans ma vie, reçoivent a travers ce travail soit-il, l'expression de mes sentiments et mon éternelle gratitudes.

*A mes très chers sœur:**KAHINA, SONIA, LINA et ma cher tante DJAMILA.***

A toute ma famille et mes amis (e), et à tous ceux qui me sont chers.

Résumé

Dans le but d'améliorer et d'accroitre le réalisme d'un rendu 3D, on a souvent recours au modèles d'éclairement (local ou global) tels les modèles de Gouraud, Lambert, Phong… etc.Nous nous sommes intéressés dans le cadre de ce projet au plaquage de texture et ce, dans le but d'améliorer les rendu 3D et plus spécifiquement les modèles numériques de terrain (MNT).Dans ce travail, nous avons réalisé le plaquage d'un damier sur des objets à trois dimensions à savoir la sphère, le cylindre, la gaussienne. Ensuite, nous avons effectué le plaquage d'une image satellite sur une sphère en tenant compte des sources de lumières et d'ombrage.

Mots clefs : plaquage de texture, rendu 3D, les modèles numériques de terrain (MNT), modèles d'éclairement

Listes des figures

Sommaire

Introduction générale

Des surfaces lisses aux couleurs uniformes, des lumières brillantes aux reflets éclairants, Voici peut être le principale défaut des images générées par un ordinateur, elles sont trop imparfaite, pourparer à cette imperfection et afinde se rapprocher du réel.Le plaquage, une technique enrichissante pour le visuel des images diffusées par un ordinateur, peut se présenter comme une solution àcette tare. Cela est devenu l'un des enjeux majeurs de la synthèse d'images. Le cinéma, l'industrie du jeu vidéo, les simulateurs toussont avides de réalisme.

Certaines formes sont complexes : un arbre, unnuage, unechevelure et d'autre peuvent être simple ; un soigneux agencement de sphère, cylindre et triangles permet de représenterune grande variété d'objet naturels ou manufacturiers.

Leurs complexités ne résident pas dans les contours et les volumes, mais dans l'apparence de leurs surfaces.Pour cela plusieurslangage de programmation ont été développés IDL en fait partie. Dans cette étude, nous proposons l'exploitation de ce dernier afin d'ajouter de réalisme dans le cadre d'un plaquage de texture 2D.

Dans le but d'améliorer et d'accroitre le réalisme d'un rendu 3D, on a souvent recours au modèles d'éclairement (local ou global) tels les modèles de Gouraud, Lambert, Phong… etc.

Mais au préalable, il s'agira d'abord de construire une surface 3D. Cette étape sera réalisée en employant des fonctions de triangulation et des fonctions d'interpolation en utilisant un système de maillage prédéfini.

Dans un second temps, on emploiera une bibliothèque de textures pour associer à chaque coordonnée de l'image 3D, une couleur puisée dans l'image à plaquer. Puis enfin, un modèle d'éclairement sera ajouté.

Les différents programmes seront mis en œuvre grâce au logiciel IDL (Interactive Data Langage) qui permet la réalisation d'applications graphiques pour produire un logiciel deplacage de texture interactive et convivial.

I .1. Préambule

Le plaquage de texture (modèled'habillage) est une technique permettant d'accroitre le réalisme d'un rendu 3D ; il consiste à coller une image sur un objet 3D à la manièred'une tapisserie afin de rendre plus intuitive la notion de modèle d'habillage, nous proposons une analogie avec le problème concret suivant :

Imaginons que nous souhaitions décorer un objet dont la surface est lisse, par exemple une statuette en métal, et que nous ayons à notre disposition du papier, des ciseaux, de la colle et des feutres.il n'est pas possible d'utiliser les feutres directement sur le métal.il nous faut donc coller du papier sur la surface, et dessiner sur le papier l'aspect que nous souhaitons donner à l'objet.

Ceci correspond à ce qui se passe lorsque l'on doit appliquer un modèle d'habillage à la surface géométrique de l'objet. Généralement, un espace de référence (ici, la feuille), dans lequel est dessinée l'apparence finale, est attaché sur le modèle géométrique de la surface.
Supposons que nous souhaitions donner l'illusion que la statuette est en bois. On peut imaginer plusieurs approches : [1]

- dessiner un aspect de bois en trompe l'œil sur la feuille de papier, puis coller la feuille sur la surface de la statuette. Le recours aux ciseaux va s'imposer si l'on veut que la forme de l'objet soit respectée. Or, ces coupures vont se voire puisque les viennes du bois, soigneusement peintes sur la feuille à plat, ne vont plus se correspondre sur la surface. Cependant la feuille peut être photocopiée et utilisée sur un autre objet, permettant un gain de temps considérable.
- coller la feuille vierge sur la statuette, puis dessiner l'aspect de bois sur l'objet.
- découper un petit morceau de papier puis peindre dessus un aspect de bois. Ce petit morceau est dupliqué plusieurs fois, par exemple à l'aide d'une photocopieuse.

Les petits morceaux seront ensuite collés sur la surfaces, tels des autocollants. Ce collage est plus facile car de petites tailles les autocollants seront moins sujets aux déformations dues aux courbes de statuette. Ils peuvent être agencés de manière à ce que les bords se correspondent à peu près pour donner l'illusion continue. Cette méthode est la plus rapide que de peindre une apparence de bois sur toute la surface. Par contre il est possible que des répétitions soient visibles sur le résultat final.

Par ailleurs, il semble extrêmement délicat de peindre à la main un aspect ressemblant à du bois ; il serait préférable d'utiliser une photographie. Mais dans ce cas, des discontinuités

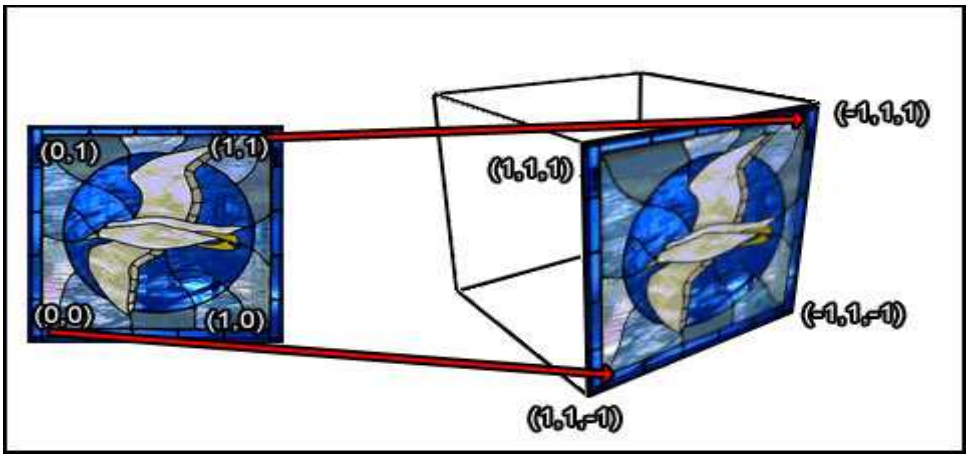
risquent d'apparaitre, comme avec la première méthode. Idéalement, il faudrait plutôt construire une machine spécialisée, capable de peindre automatiquement l'aspect de surface, pour un objet donné, en tenant compte de la méthode employée pour attacher l'apparence sur la surface.

I.2.Le plaquage de texture

La technique du plaquage(ou mappage) de texture est beaucoup utilisée pour améliorer la richesse visuelle des images générées par ordinateur. Cette technique consiste à mettre en correspondance chaque surface 3D avec une image 2D à l'aide d'une fonction appelée une paramétrisation associe à chaque point d'une surface paire de coordonnées (u ; v) désignant un pixel de la texture.

I.2.1.Principe

Le placage de texture a été introduit par CATMULL en 1974. L'idée principale est d'associer une image à deux dimensions appelée textureà une surface géométrique. L'apparence de la texture est définie dans un espace de référence : l'espacetexture. La correspondance entre l'objet et la texture est définie par la paramétrisationde la surface géométrique. On associe chaque point de la surface à un point de l'espace texture, appelé coordonnée de texture(u; v). La paramétrisation permet de lire directement dans la texture la couleur correspondante a un point quelconque de la surface. La texture est ainsi appliquée à la surface de l'objet. [2]



Figures (I.1) : Principe de plaquage de texture

I.3. Définition de la texture

Bien que la notion de texture soit naturelle pour l'être humain, elle résiste depuis longtemps à toute tentative de définition. On peut s'en rapprocher en disant qu'une texture est une zone de l'image qui présente certaines caractéristiques d'homogénéités qui la font apparaitre comme une zone unique. Nous pouvons aussi la décrire comme étant un ensemble de primitives de taille et de forme variables, présentant une organisation spatiale particulière.

La texture est définie aussi comme étant une région d'une image pour laquelle il est possible de définir une fenêtre de dimensions minimales, telle qu'une observation au travers de celle-ci se traduit par une perception (impression)visuelle identique pour toutes les translations possibles de cette fenêtre à l'intérieur de la région considérée.

I.3.1.Principe

- La texture est un tableau de données (réservoir d'information).
- En chaque pixel (lors du remplissage des polygones sur l'écran) on extrait une information pertinente de ce réservoir.
- Cette information peut servir à moduler n'importe quel paramètre du rendu (couleur, spécularité, transparence, vecteur normal, …).
- L'accès dans la texture est contrôlé par des *coordonnées de texture* (données à chaque sommet et interpolées dansle triangle).

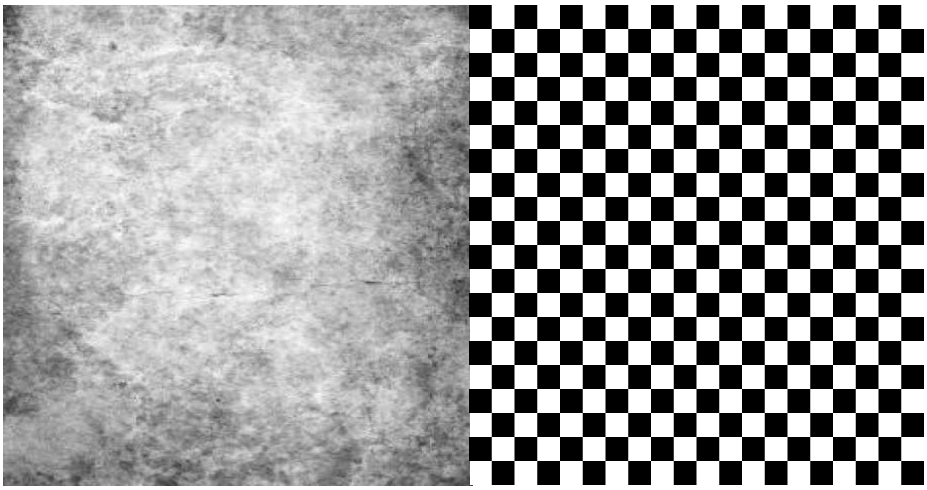


Figure (I.2) : Exemple de texture

I.4. Les différents types de textures

I.4.1. les textures planes

Les textures planes sont des images qui sont en quelques sortes collées à la surface des objets. Les textures planes sont très utilisées, car il est facile de les créer et de les utiliser. Elles peuvent cependant faire surgir des problèmes de raccords. [3]

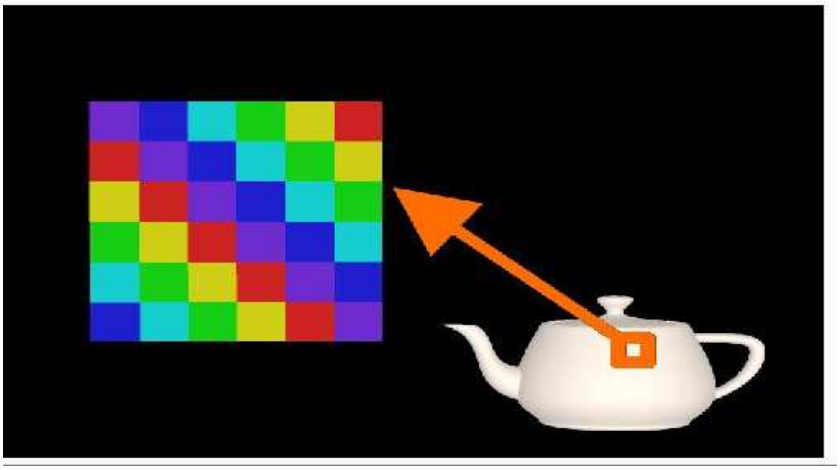


Figure (I.3) : Texture plane

I.4.2 les textures volumiques

Un objet utilisant des textures volumiques apparaît comme étant directement sculptés dans la matière.la textures étant indépendante de la forme de l'objet, il n'y a plus de problème de raccords. Ce type de texture n'est que peu utilisé dans les jeux vidéo. [3]

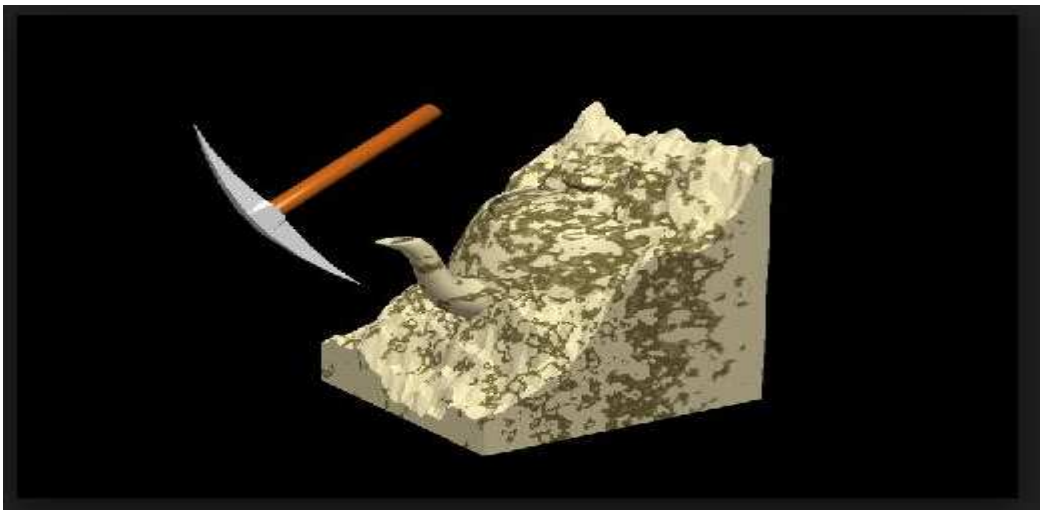


Figure (I.4) : Texture volumique

I.4.3.Les textures numérisées

Ce type de textures est obtenu assez facilement, car elles sont obtenues à l'aide de textures préexistantes. Elles peuvent êtreobtenues avec un appareil photo numérique (enfin de préférence), un scanner ou une caméra numérique. Ce procédé permet d'obtenir des textures réalistes relativement facilement, cependant, des problèmes dans l'habillage des objets.De plus, ces textures sont extrêmement lourdes. Ce type de texture utilisé dans les jeux vidéo. [3]

I.4.4.les textures synthétisées

Ce type de texture est issu de calcul mathématique, elles peuvent représenter des motifs géométriques, comme les briques, des tuiles, etc., ou des motifs pseudo aléatoires tels que des roches, de l'écorce, des feuilles, etc. Ce type de texture est surtout utilisé dans des démos. [3]

Remarque

Dans le cas de textures planes, on parlera principalement de plaquage de texture (mapping en anglais). Le plaquage de texture consiste à appliquer une texture 2D sur un objet 3D.il faut donc faire attention au repérage. En effet, l'objet est définit dans un espace à trois dimensions de coordonnées (X, Y, Z), la texture étant définie dans un espace à deux dimension (celles de l'écran par exemple) de coordonnées(X, Y).

Le calcul de la visualisation d'une texture via la technique du mapping est donc le résultat d'une transformation texture-objet et objet-image finale, comme le montre la figure ci-dessous. [4]

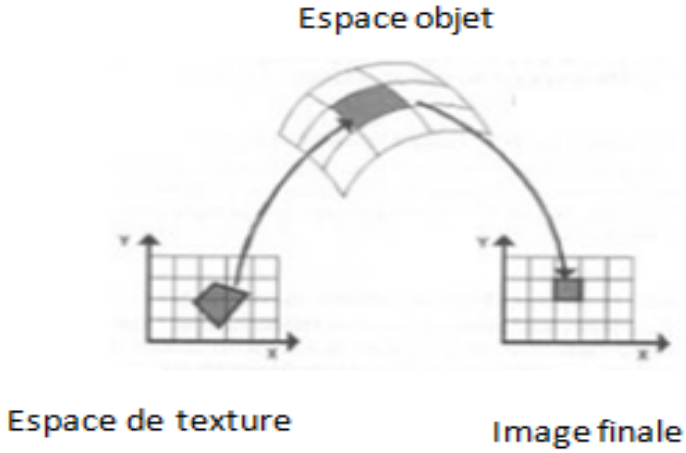


Figure (I.5) : La technique de mapping

I.5.Types de mapping

I.5.1.Introduction

Il existe essentiellement trois types de mapping. Le mapping de texture, plus général, le mapping lié au point de vue, pour afficher l'environnement et le bumpmapping.

Le bumpmapping consiste à perturber la surface apparente à l'aide d'une autre texture. [4]

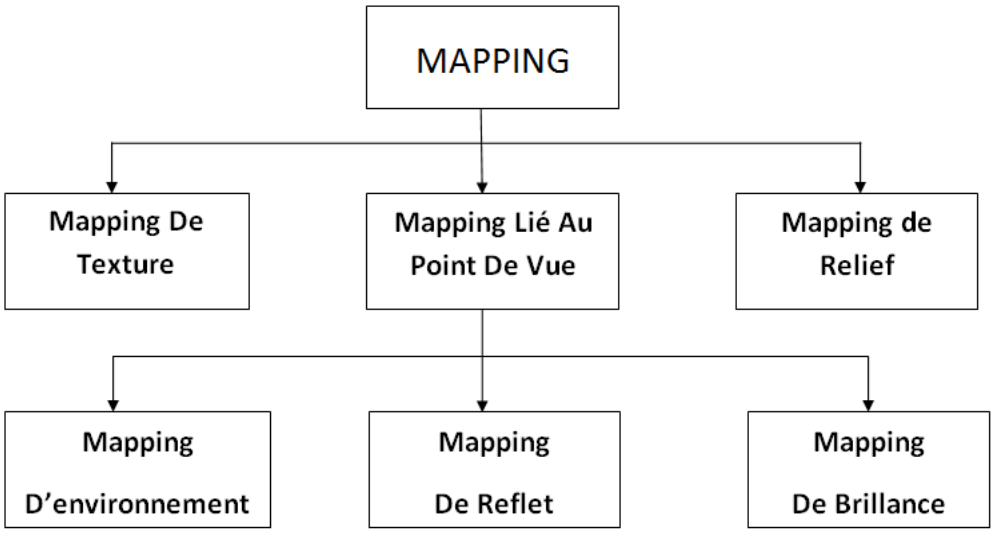


Figure (I.6) : Type de mapping

I.5.2.Le mapping de texture simple

Lemapping de texture simple est la méthode de mapping la plus utilisée. Elle consiste à projeter une image 2D sur un objet 3D. On peut ainsi, facilement simuler un tableau, un tapis, ce type de mapping est utilisé dans les jeux vidéo. [4]

Figure (I.7) : Texture simple

I.5.3. Le bumpmapping

Les aspérités ont un intérêt certain en synthèse d'image car elles permettent d'accroitre le réalisme de certains revêtements. Leur seul gros défaut est d'être très gourmandes en polygones et nécessitent donc un temps de modélisation et de rendu relativement élevé. Elles sont de fait impossibles à utiliser dans le cadre d'applications en temps réel comme les jeux vidéo. De plus, une bonne texture et un bon mapping permettent de faire aussi bien et pour moins cher.

Ainsi au lieu d'essayer de modéliser la scène, il estsouvent bien plus simple et bien plus rapide d'avoirrecours au bumpmapping. Cette méthode consiste àsimuler des creux et des bosses en jouant sur lesniveaux de luminosité de l'image considérée en noiret blancTechnique du bumpmapping consiste à modifierlégèrement en hauteur et en orientation les normales(droite perpendiculaire à la surface) des facettes del'objet, le blanc indiquant la normale la plus haute, lenoir la plus basse. Le blanc est alors utilisé pourdonner l'illusion d'une bosse et le noir l'illusion d'uncreux. Il est cependant essentiel d'éviter les écarts de hauteur trop importante.

On peut réaliser cela en perturbant la surface apparente à l'aide d'une autre texture, comme le montre le schéma ci-dessous

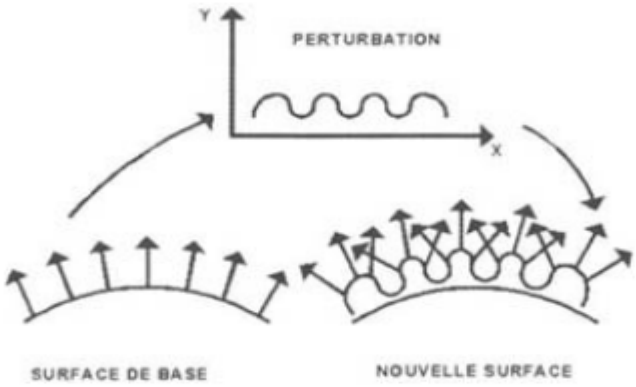


Figure (I.8) : Bumpmapping

Il est cependant essentiel d'éviter les écarts de hauteur trop importante, car comme l'illustre la théière ci-dessous, celles-ci ne sont qu'illutions et n'affectent pas le contour des objets. [4]

Figure (I.9) : image illustrant le bumpmapping

On peut voir ici que les contours de la théière sont lisses, alors que le texte apparait en relief.

I.5.4.Le mapping d'environnement

Le mapping d'environnement a été inventé pourremédier aux problèmes posés par le calcul de laréflexion de la lumière sur certaines surfaces. Cettetechnique de mapping permet de simuler facilementla réflexion d'un décor complet sur un ou plusieursobjets, en plaquant l'image de ce décor ensurimpression sur les objets. Il devient alors assezaisé de simuler des objets réfléchissants comme parexemple des miroirs ou certains types de verres.

Le mapping d'environnement est fixé à la vue plutôtqu'a l'objet, ce qui fait que si l'on se déplace autourde l'objet, la vue réfléchie, représentée par le bitmapreste fixe, comme elle le ferait dans la réalité.

Ce type de mapping a cependant des limites, ainsi,un objet dynamique venant s'intercaler entre lascène est la surface réfléchissante ne sera pas visibledans le reflet.

Figure (I.10) : Mapping d’environnement

I.5.5.Le mapping d'opacité

Ce type de mapping permet de simuler assezfacilement des objets partiellement opaques (vitresale, grillages, feuillage,…). Comme dans le cas dubumpmapping, la texture est considérée en noir etblanc, le blanc correspondant aux zonestransparentes, le noir aux zones opaques.

Sur cette capture d'écran de Medal of Honor, on peutvoir que ce type de mapping est utilisé pour lesbarbelés et les branchages.

Figure (I.11) : Mappingd’opacité

I.5.6. Le mapping de brillance

Ce type de mappingmodifie l'intensité du reflet basée sur l'intensité du bitmap. Les pixels blancs du mapping de brillance produisent un reflet, tandis que les pixels noirs suppriment complètement celui-ci. Sile mapping de reflet affecte la couleur du reflet, le mapping de brillance affecte son intensité.

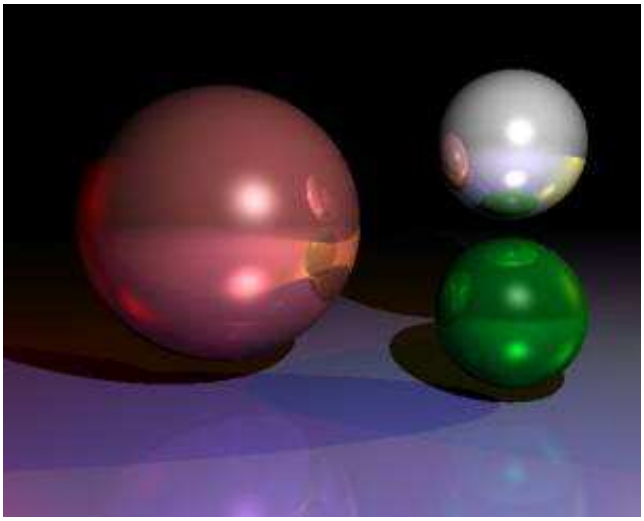


Figure (I.12) : mapping de brillance

I.5.7. Le mapping de reflet

Le mapping de reflet fonctionne comme le mappingde texture, cependant, il n'est visible que dans leszones de reflet. Il affecte la couleur du reflet et estsurtout visible dans les zones les plus éclairées del'objet. [4]

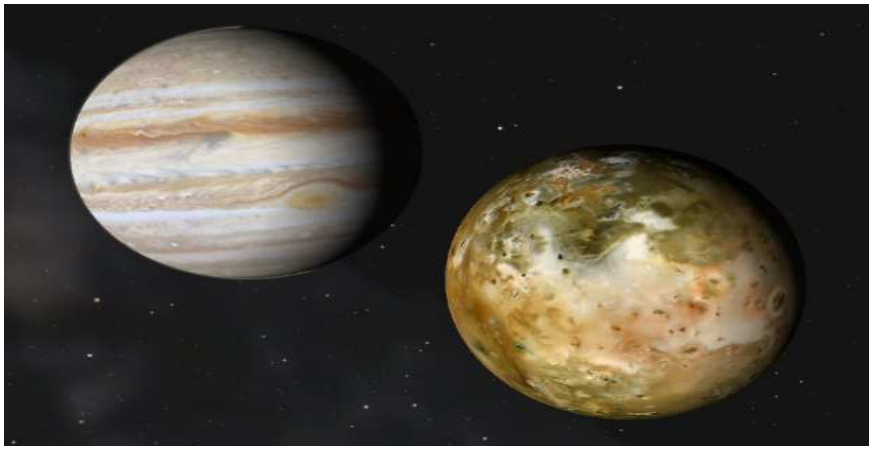


Figure (I.13) : Mappingde reflet

I.5.8. Le mapping d'auto illumination (ou Lightmap)

Permet d'appliquer des effets d'auto illumination basés sur l'intensité des pixels de l'image bitmap correspondant à l'image considérée en noir et blanc.

Ainsi, plus les valeurs des pixels du bitmap sont proches du blanc, plus l'auto illumination n'est importante. Ce type de mapping donne l'impression que l'objet génère sa propre lumière. Ce type de mapping a été utilisé pour la première fois dans Quake II afin d'améliorer la qualité de rendu des éclairages de manière significative sans pour autant vampiriser les ressources de la machine. Depuis lors tous les jeux utilisent cette méthode d'éclairage peu ou prou.

Figure (I.14): Mappingd'auto illumination,

Une capture d'écran de Quake IIles lightmap sont ici utilisée pour le néon et les lumières rouge au fond de la cage.

I.6. Modes de projection

Cette opération s'effectue en choisissant la forme de la surface de projection. Chaque image est ainsi attachée à une primitive de surface, telle le cube, la sphère, le cylindre ou le plan. La surface de projection est ainsi paramétrée, ce qui veut dire les points de la surface sont liés aux coordonnées (u, v) de l'image.

De plus, il est également possible de déplacer et de changer la taille de l'mage sur la surface de projection elle-même, par un contrôle ultérieur. Il existe cinq types de projections différentes [4] :

I.6.1. La projection planaire

En projection planaire, l'image la texture est appliquée sur un plan et est projetée perpendiculairement à ce plan sur l'objet. Ce type de projection permet de garder une mappe plane, mais des distorsions peuvent apparaître. Ce type de projection est surtout utilisé dans les jeux vidéo afin de simuler une surface de sol.

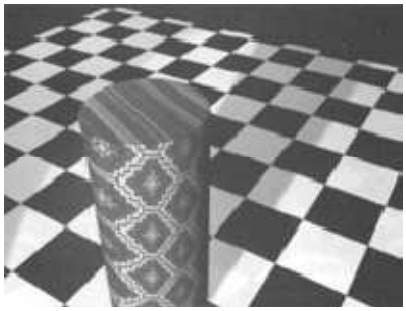


Figure (I.15) : Une image illustrant la
Méthode de projection planaire sur un cylindre

I.6.2.La projection cylindrique

La projection cylindriquepermet d'arrondir la mappeen cylindre pourl'envelopper autour d'unobjet de la même manièrequ'une étiquette enveloppeune bouteille. Ce type deprojection estparticulièrement utile pourles objets cylindriques. Cetype de projection est assezutilisé dans les jeux vidéo,en association avec d'autresméthodes de projections,afin d'éviter les problèmes de raccords.

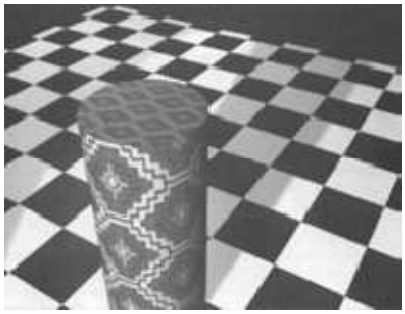


Figure (I.16) : Une image illustrant
La méthode de projection cylindrique sur un cylindre.

I.6.3. La projection sphérique

« Ce type de projection permet d'arrondir la mappe en sphère pourl'envelopper autour d'unobjet pour l'enveloppperautour d'un objet de lamême manière qu'unplanisphère entoure unesphère pour créer unglobe terrestre. »[4].

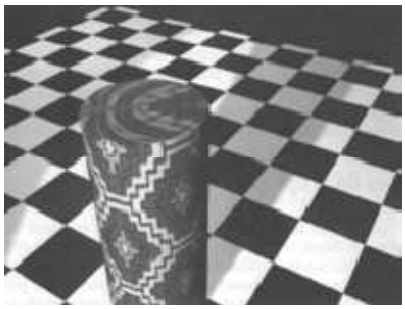


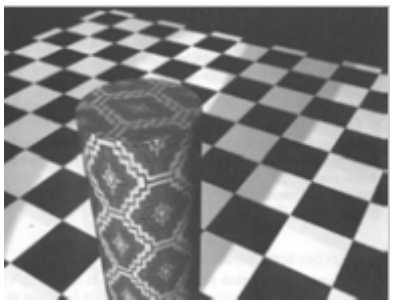
Figure (I.17) : Une image illustrant

La méthode de projection sphérique sur un cylindre

I.6.4. La projection cubique

Correspond à six projectionsplanaires arrangée enparallélépipède. Elle est trèsutilisée dans les jeux vidéopour simuler les pièces d'unemaison. Par exemple, unepièce a un mur recouvertd'un papier peint et la piècevoisine qui possède un murmitoyen possède un autrepapier peint, alors qu'enréalité, c'est le même objet.

Nous pouvons voir sur lecylindre ci-dessous que cemode de projection a étéutilisé afin de rendre lesraccords invisibles.



Figure(I.18) : Un exemple de projection cubique sur un cylindre.

I.6.5. La projection automatique

Dans le cas de la projection automatique, unalgorithme utilise la normale de chaque polygonepour mapper la texture. Ce type de projection peutêtre utilisé sur des objets complexes.

Dans cette partie, nous abordons le problème de l'ombrage des surfaces de la scène traitée en fonction de leurs positions, orientations et caractéristiques ainsi que de celles des sources lumineuses présentes dans la scène.

Nous distinguerons par la suite :

- le modèle d'illumination qui permet de déterminer la couleur d'un point de la surface
- le modèle d'ombrage qui détermine la manière d'appliquer le modèle d'illumination sur les surfaces :

➢ par pixel

➢ par interpolation des valeurs en certains pixels.

Les méthodes existantes pour effectuer un rendu différent sur le modèle d'illumination choisi : modèles sans interaction entre les objets (modèles simples) ou avec interaction (méthodes globales) et le modèle d'ombrage (Gouraud, Phong).

I.7. Les modèles d'illumination

Un modèle d'illumination comprend tout le processus de transport de la lumière, quicomprend son émission, sa réflexion, réfraction, diffusion ou absorption. Il permetd'obtenir une solution approchée de l'illumination d'une scène. Deux types demodèles existent : les modèles d'illumination locale qui ne considèrent que lacontribution directe des sources de lumière et les modèles d'illumination globale quicomprennent en plus la lumière inter réfléchie entre les différentes surfaces de lascène. Dans ce processus, un modèle de réflexion se concentre sur la lumière quiatteint une surface et sur la lumière qui est réfléchie.

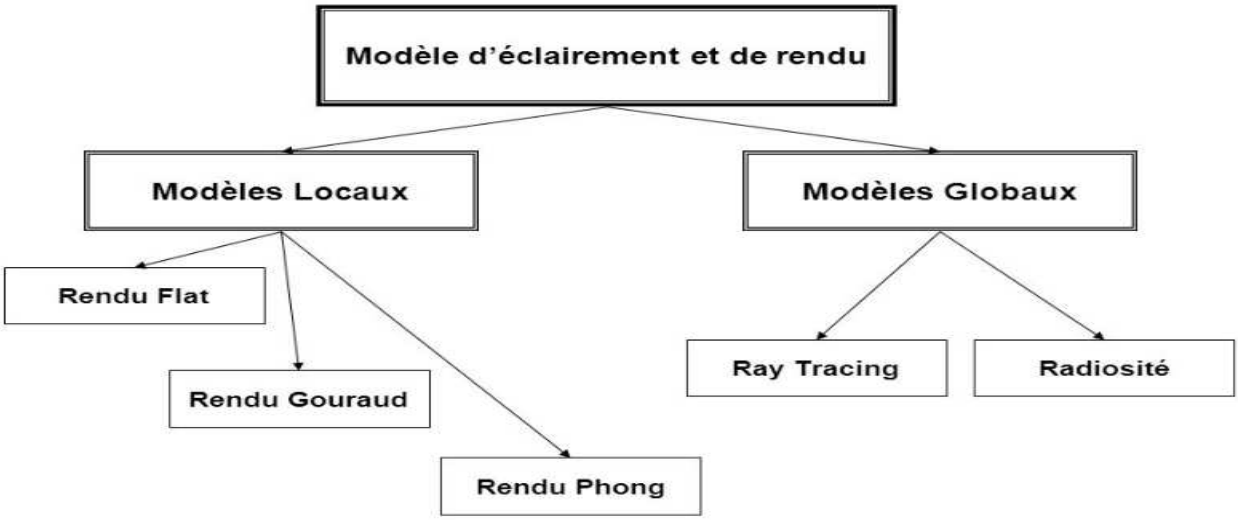


Figure (I.19) : Les modèles d'illumination

I.7.1. Illumination locale

Un modèle d'illumination locale prend en entrée l'éclairage incident et les propriétés du matériau en un point de la surface. Il modélise la réaction de la surface à la lumière envoyée par l'objet. Ici, le terme lumière regroupe à la fois la couleur et l'intensité lumineuse.

Les modèles d'illumination utilisés dans le domaine du graphisme sont inspirés de modèles physiques simplifiés et adaptés à l'usage informatique.

La validation de ces modèles découle autant de l'usage pratique que d'raisonnement mathématique.

Dans cette partie, nous allons avoir des modèles d'illumination simples ne prenant pas en compte les interactions entre objet mais seulement les interactions entre une surface et les sources lumineuses.

Ces modèles sont largement utilisés car ils produisent des résultats au réalisme satisfaisant dans de nombreux cas.

I.7.2. modèle RGB

La raison pour laquelle nous parlerons de ce modèle d'illumination est simple : c'est le plus employé pour la réalisation graphique. La lumière est une synthèse de trois couleurs (rouge, vert, bleu) qui possèdent des intensités variable allant de 0 à 1. Dans son principe, ce modèle d'illumination mélange couleurs et intensité, ce que nous appellerons : la luminosité.

I.7.3. principe d'illumination : la loi de Lambert

Après avoir défini le principal d'illumination, on va expliquer comment la lumière prend effet sur un objet .lorsqu'un objet reçois de la lumière, il l'absorbe et en réfléchit une partie.

Ainsi, l'œil ne reçoit qu'une simple partie de cette lumière qu'on perçoit avec une couleur et une intensité.

On peut distinguer quatre étapes dans notre façon de percevoir la lumière :

- ✓ L'intensité lumineuse sur l'objet
- ✓ La réfection d'une partie de la lumière
- ✓ La perception d'une partie de la lumière réfléchie
- ✓ L'interprétation qualitative de la lumière perçue

De plus les modèles d'illumination sont basés sur ces principes et suivent différents paramètre contrôlent ce procédé. Il faut connaître :

- ➢ La position et la direction de l'objet

- La position et la direction de l'observateur
- La position et la direction de la source lumineuse
- Les propriétés d'absorption de l'objet
- La couleur et l'intensité de la lumière générée par la source.

La source lumineuse est toujours considérée comme génératrice de rayon lumineux. Lorsqu'un rayon rencontre un point d'un objet, il est réfléchi et plusieurs nouveaux rayons se forment.

Leurs couleurs et intensités dépendent de deux paramètres :

- Les propriétés d'absorption de l'objet
- L'angle d'incidence qui se forme entre le rayon et la normale à l'objet

Le phénomène décrit ci-dessus est appelé : la réflexion diffuse.

Considérons**θ** l'angle formé par le rayon incident et une normale à l'objet, on peut écrire que :

- Lorsque**θ**augmente, l'intensité lumineuse baisse
- Lorsque**θ** baisse, l'intensité lumineuse augmente

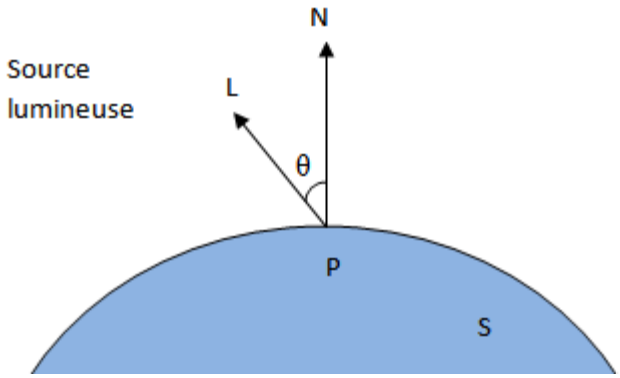


Figure (I.20) : principe d'illumination

On remarque que donc que l'intensité lumineuse est proportionnelle à l'angle θ. Le rapport existant entre ces deux valeurs est appelé : loi de Lambert.

On constate que l'intensité est égale à 0 quand θ est supérieur à90, est à son maximum quand θ est égale à 0, on peut donc écrire la loi de Lambert suivante :

$$Id = Ii \times Kdiff \times cos\theta \qquad (1)$$

Avec:

– ***Id*** : intensité diffuse, c'est-à-dire l'intensité de lumière

– ***Ii*** : intensité incidente

– ***Kdiff***: paramètre de texture paramétrant le comportement. Un faible coefficienttraduit un objet absorbant, tandis qu'un fort coefficient traduit une forteréémission ;

– ***θ*** : angle d'incidence de la lumière, calcule par rapport au vecteur normal à lasurface

Pour déterminer la valeur de l'intensité actuelle d'un point et sa couleur exacte, il faut multiplier l'intensité avec la couleur actuelle du point .le résultat dépendra du type de couleur utilisé.

On pourra alors faire varier l'intensité et la couleur en fonction des différentes valeurs de Kd.

I.7.4. la réflexion diffuse

Nous allons revenir sur trois aspects énoncés dans la partie précédente, et tâcher de les détailler :

- La position de l'objet.
- Laposition de l'observateur.
- La position de la source lumineuse.

Nous avons expliqué précédemment que lorsqu' un rayon lumineux rencontrait un point d'un objet, il fait réfléchi en plusieurs nouveaux rayons lumineux. Ainsi, la couleur d'un point et son intensité ne dépendent pas de la position de l'observateur.

Cependant l'intensité lumineuse d'un point dépend de la distance qui sépare l'objet de la source de lumière.

Figure (21):Réflexion diffuse sur la sphère

Réflexion diffuse total. Aucune réflexion spéculaire visible

Toutefois, un facteur de réduction pour empêcher que les objets qui sont à des distances déférentes de la source de lumière aient les mêmes couleurs.

Ce facteur de réduction a pour expression

$$I=Li*Kd*\boldsymbol{cos(\theta)/r^2}$$

L'implémentation de cette technique n'étant pas satisfaisante, on introduit un coefficient Kr.

Nous obtenons donc :

$$I=Ii*Kd*\boldsymbol{cos(\theta)}/\ (\ \boldsymbol{r^2}+\boldsymbol{Kr})$$

Pour éviter que les polygones qui ont un angle $\boldsymbol{\theta>90°}$ soient noirs, il faut ajouter une lumière environnante telle qu'on obtienne :

$$I=I\boldsymbol{\theta}*K\boldsymbol{\theta}+I\boldsymbol{\theta}*K\boldsymbol{\theta}*\boldsymbol{cos(\theta)/r^2}$$

Ce développement n'est pas très utilisé, car trop complexe.

Le modèle le plus simple consiste à prendre en compte une source lumineuse non ponctuelle qui émet de manière constante dans toutes les surfaces une lumière d'intensité Ia.

Alors pour un point P de la surface, l'intensité sera :

I=Ka Ia.

Qui est constante en tous points de la surface, et ou Ka est un facteur qui détermine la quantité de lumière ambiante réfléchie par la surface et est fonction des propriétés matérielles de la surface (0<= Ka<=).

Ce modèle d'illumination n'est pas utilisé tout seul.il présente en effet peut d'intérêt pas lui-même.il est utilisé en conjonction avec d'autre modèles pour permettre d'illuminer certains parties d'une surface qui ne le sont pas autrement.

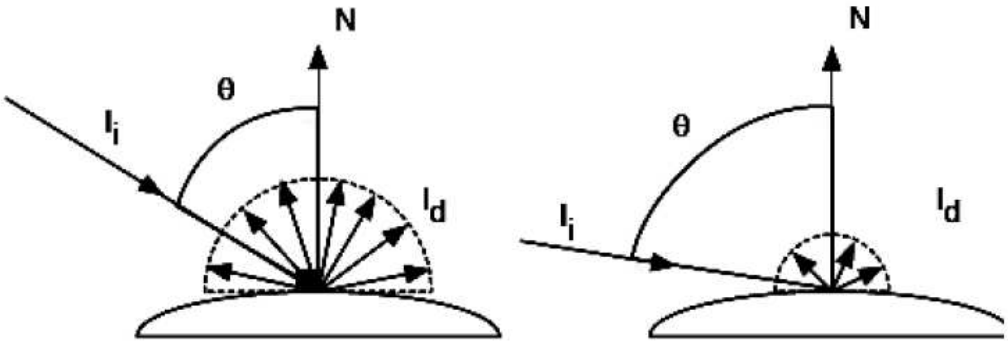


Figure (I.22) : Réflexion diffuse

I.7.5.réflexion spéculaire (model de phong)

Le modèle d'illumination locale le plus utilisé est le modèle défini par B.T. PHONG en 1975. Il définit l'intensité lumineuse réfléchie en un point *P* d'une surfaceen fonction de l'intensité des sources lumineuses. Ce modèle est une généralisation dela loi de Lambert sur la réflexion de la lumière. Cette loi, aussi appelée loi ducosinus, décrit l'intensité diffusée par un point d'une surface mate en fonction del'intensité et de la direction de la source prise en compte. [5]

Phong a proposé un modèle de réflexion très populaire, en effet, l'idée a été d'ajouter une réflexion spéculaire qui permettrait d'éviter qu'un polygone éclairé ne soit trop terne. Cette réflexion spéculaire n'est en fait que le symétrique de la lumière incidente par apport à la normale au polygone. [6]

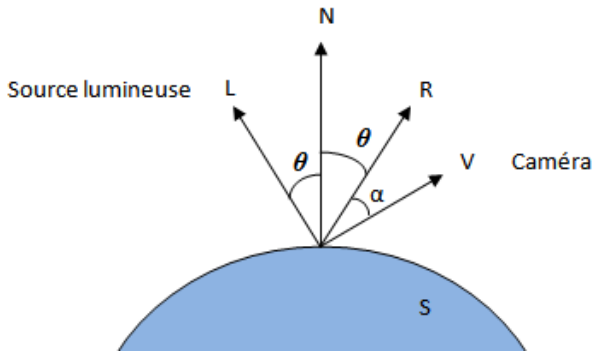


Figure (I.23) : Principe de phong

Par conséquent, l'œil peut percevoir des rayons lumineux réfléchis sur d'autres objets. Cephénomène est appelé la multiplication de la source de lumière.

Il faut savoir que la couleur d'un point ne dépend pas de la position et de la direction de l'observateur, elle est la même partout. Cependant, la position de l'observateur a des incidences directes sur l'intensité d'un point. En effet, plus l'observateur se rapproche du rayon lumineux réfléchi, plus l'intensité lumière augmente, et vice-versa.

On ajoute au modèle vu précédemment, la réflexion spéculaire :

Ks*(cos(α))^n avec :

Ks : une constante liée au polygone objet.

α : l'angle entre le rayon réfléchi et un autre vecteur qui représente la direction de l'œil.

Figure (24) : réflexion spéculaire sur la sphère

Réflexion spéculaire total. Aucune lumière diffuse

I.8.Modèle d'ombrage

La solution la plus simple pour effectuer le rendu d'une surface consiste à calculer l'illumination en chaque point visible de la surface. Cette méthode naïve est extrêmement couteuse en temps .dans cette partie, nous allons voir différente méthodes permettant de diminuer le cout en temps en n'effectuant le calcul d'illumination qu'en un nombre limité de point.

I.8.1. Le modèle flat

C'est sans aucun doute le modèle d'illumination le plus simple qui soit. Le principe consiste à placer dans la scène une source lumineuse située à l'infini, et éclairant ainsi uniformément chaque point d'une facette. Ainsi, chaque des faces composant l'objet sera dessinée avec une couleur unique.

C'est un modèle qui est donc très simple à mettre en œuvre et extrêmement rapide. En fait, on calcule l'intensité lumineuse qu'une fois pour chaque facette de l'objet. [7]

Si l'on veut utiliser la fonction de Lambert, il faut que l'on ait le cosinus de l'angle entre le vecteur de lumière et une normale [7]

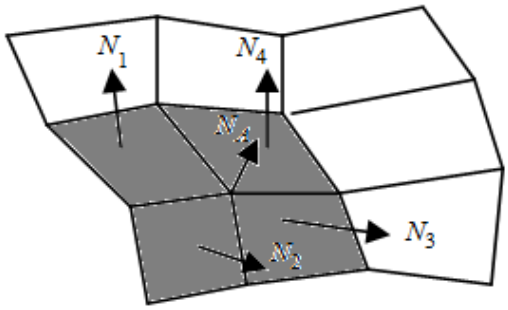
- Le modèle flat peut donc se résume en trois grandes étapes :
- Calculer la normale du polygone à l'aide d'un produit vectoriel.
- Calculer l'intensité de la couleur qui sera utilisé avec la fonction de Lambert.
- Remplir le polygone avec la couleur calculée.

En conclusion, on peut dire que le modèle Flat est très simple d'utilisation et très rapide, cependant, il nous donne un mauvais rendu. [7]

Figure (I.25) : Ombrage flat de la sphère pour :
16-16 facettes, 32-32 et 64-64

I.8.2. Ombrage de Gouraud

L'ombrage Gouraud le même modèle d'illumination que le modèle flat, c'est-à-dire le modèle Lambertien. Ce qui diffère dans le Gouraud c'est la méthode de calcul des normales.
La méthode développée par Gouraud [1971] élimine les discontinuités d'intensité sur une facette polygonale par interpolation des valeurs d'intensité aux sommets de la facette. Avec Gouraud, on calcule la normale d'un point.



Figure(I.26) : principe de calcul de la normal pour l'ombrage grouraud

La méthode développée par Gouraud en 1971 élimine les discontinuités d'intensité sur une facette polygonale par interpolation des valeurs d'intensité aux sommets de la facette. Cette méthode est largement utilisée et se retrouve dans la majorité des matériels graphiques existants.

Cette méthode requiert la connaissance de la normale à la surface aux sommets des facettes polygonales. Lorsque les normales sont connues, les intensités aux sommets des facettes polygonales sont calculées. L'interpolation peut ensuite être effectuée à l'aide de l'algorithme de balayage de ligne utilisée pour le remplissage de polygone et le z-buffer.

Pour calculer la normale d'un point, on fait la moyenne des normales des facettes ayant ce point commun.

Apres avoir obtenu les normales, on utilise dans la fonction de Lambert connue pour calculer l'intensité de chaque point du polygone, puis, on utiliser l'interpolation.

On interpole chaque point du polygone pendant son remplissage.

Cependant, l'interpolation est basée sur une étape d'incrémentation, en fait, considérer l'interpolation comme un dégradé de couleur. Mathématiquement, cela consiste à partir d'une valeur minimum pour aller vers une valeur maximum en usant d'une étape d'incrémentation.

C'est le même principe que lorsque l'on initialise le coté du polygone dans un but de le remplir.

Pour résumer l'algorithme de l'ombrage Gouraud, on doit :

- Calculer la normale de chaque point.
- Utiliser la fonction de Lambert pour calculer la couleur de chaque point
- Interpoler la couleur des sommets pour chaque point du côtéd'un polygone.
- Interpoler chaque ligne remplie avec la procédure.

Le rendu de Gouraud est beaucoup plus réaliste et satisfaisant que celui du modèle flat. C'est un bon compromis entre réalisme et temps de calcul, c'est pourquoi, cette méthode est très utilisée dans les jeux en 3 dimensions.

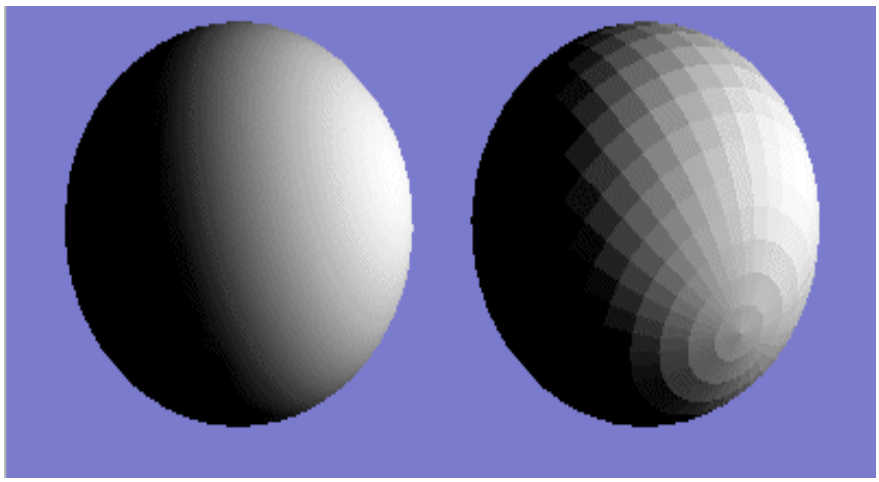


Figure (I.27) Ombrage de Gouraud sur la sphère

I.8.3. ombrage de phong

L'intérêt de cette approche par rapport à l'ombrage de Gouraud réside principalement dans sa capacité à traiter les réflexions spéculaires. Gouraud ne permet pas, en effet, de prendre en compte les réflexions spéculaires lorsque celles-ci sont localisées au centre d'une facette.

L'ombrage de Phongest un modèle de rendu très réaliste parce qu'il utilise le modèle d'illumination Phongavec tous ses paramètres, et l'ombrage est beaucoup plus avancé que le Gouraud.

L'Ombrage de Phongconsiste à interpoler les normales au lieu d'interpoler les couleurs. La couleur du point est alors calculée en fonction de la normale interpolée
Ceci signifie que chaque point dans le polygone a une normale. [7]

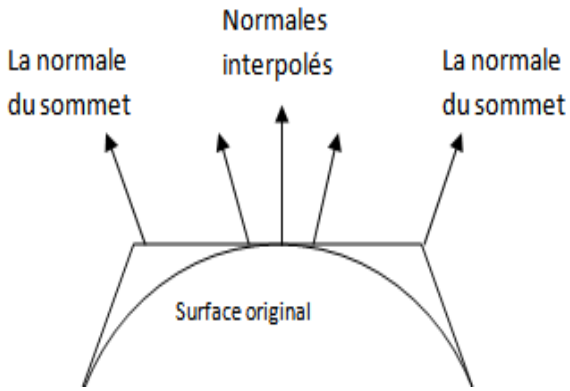


Figure (I.28) : Principe de calcule des normal pour l'ombrage de phong

I.9. Méthodes globales

Les méthodes précédentes permettent d'effectuer un rendu sur des surfaces directement éclairées par une source lumineuse. Elles ne permettent pas de prendre en compte les interactions entre objets : réflexions d'une surface sur une autre, occultation de la lumière par une surface (ombrage).ces interaction nécessite un modèle d'illumination global. Deux classes de méthodes existent qui sont basées sur des modèles d'illumination globaux :

1. La première classe concerne les algorithmes de lancer de rayons qui remplacent des illuminations locales ambiantes, diffuses et spéculaires par un modèle à base de réflexion spéculaire et transmission globale.

2. La deuxième classe concerne les méthodes de radiosité qui modélisent toutes les interactions entre objets par des sources lumineuses dans une étape préliminaire. Une image est ensuite, déterminée pour un point de vue donné par des algorithmes classiques d'élimination de partie cachées et d'interpolation d'ombrages.

Figure (I.29) : Ombres avec l'algorithme d'illumination globale

I.9.1.Lancer de rayon

Les algorithmes de lancer de rayons (ray casting) permettent de déterminer l'illumination ainsi que la visibilité des surfaces en traçant des rayons imaginaires du point d'observation vers la scène. Dans sa version la plus simple, un rayon est lancé pour chaque pixel de l'image, et la couleur du pixel est déterminer à partir de l'illumination du point de surface intersectée par le rayon qui est la plus proche de l'image. On peut pour cela utiliser les modèles d'illumination locaux présentes précédemment.

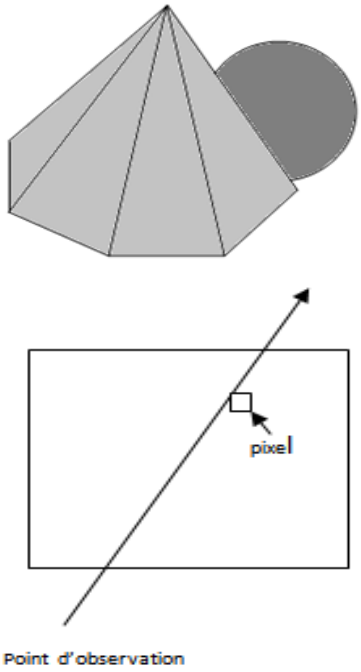
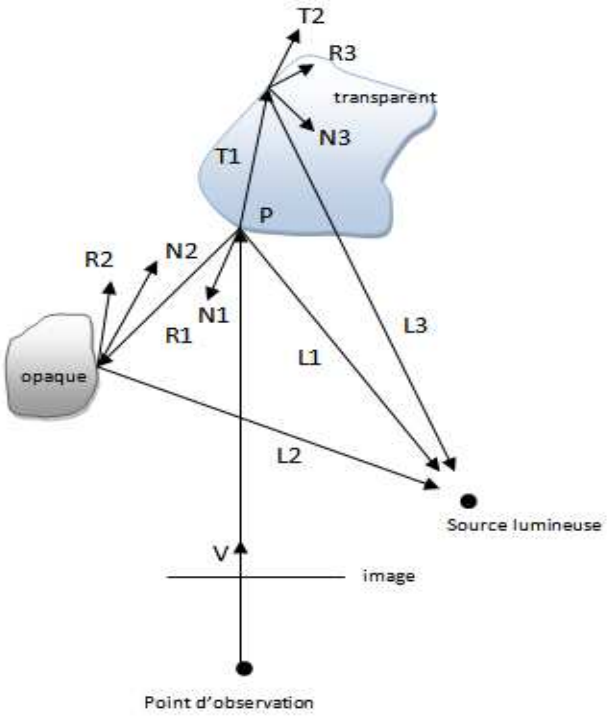


Figure (I.30) : lancer de rayon sous sa forme la plus simple

Une approche plus globale en [1980] étant le lancer de rayons de manière à inclure les réflexions spéculaires globales ainsi que les réfractions. Pour chaque pixel de l'image, un premier rayon émis du point d'observation est générer. Ensuite, au point P de la surface toucher par se rayon, trois niveaux rayons sont générer. (Voire la figure)

1. un rayon réfléchie R symétriquement à la normale à la surface en P ;
2. un rayon d'ombre du point du point de la surface vert la source lumineuse pour vérifier si une autre surface n'occulte pas la source lumineuse ;
3. éventuellement un rayon transmis T, suivant les lois de la réfraction, si la surface transparente.

Chaque nouveau rayon réfléchie ou transmis poursuit sa course dans la scène et est à l'origine de trois nouveaux rayons lorsqu'il touche une surface. Ce processus récursif s'arrête lorsque : le rayon généré touche la source lumineuse ou le rayon sort de la scène ou bien une profondeur limite de processus récursif est atteinte.



Figure(I.31) :lancer de rayon

L'equation d'illumination au point P de la surface fait intérvenir le modèle de phong s'écrit, pour une composante rouge, verte ou bleu (le signe somme consérne les déffirentes source lumineuses) :

IP=Ka S Ia + KrIr + KtIt + sommeX(P) fd Il[Kd S(NP*LP) + Ks ***cos***^n(RP.LP) :

Ou :

- S est l'intensité de la surface pour composante concerné ;
- Ia est l'intensité de la source lumineuse ambiante ;
- Il est l'intensité de la source ponctuelle/directionnelle ;
- Ir est l'intensité de rayon réfléchie, la direction du rayon réfléchi R étant :

R= V-2(N.V) N.

Ou

V est la direction du rayon source

-It est l'intensité du rayon transmis, la direction du rayon transmis T

T= (n1/n2) V-(***cos***θt +n1/n2(V.N)) N

Avec :

Sinθt/Sinθi =n1/n2

Ou n1 et n2 sont les indices de réfraction des matériaux considérés (air = 1)

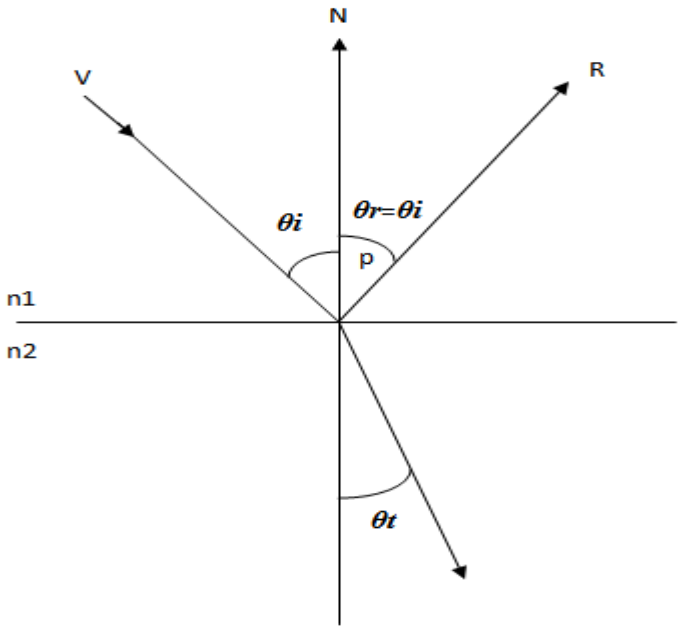


Figure (I.32) : la réfraction

-Kr ;Kt sont des coefficient compris entre 0 et 1 et tel que Kr + Kt = 1

-X(P) est tel que :

-X(P) = 1 si le rayon d'ombre de P vers la source lumineuse (L1 dans la figure 3) n'est pas bloqué ;

-X(P) = 0 sinon.

I.9.2.Radiosité

Les méthodes de radiosité apportent une modélisation plus précise de l'inter-réflexion entre objet. Toutes les énergies lumineuses émises ou réfléchies par chaque surface sont prises en compte.

La radiosité caractérise le taux d'énergie quittant une surface et correspond à la sommedes taux d'émission de la surface, de réflexion et de transmission par d'autres surfaces. Une différence majeure avec les méthodes de lancer de rayons est que ces méthodes pré-calculent les interactions lumineuses pour un environnement donné indépendamment du point de vue. Les images pour différents points de vues sont générées ensuite.

L’équation de radiosité

L’environnement étant échantillonné sous la forme de patchs discrets, de taille finie, émettant et réfléchissant la lumière uniformément sur leurs surfaces. En considérant chaque patch comme étant une surface Lambertienne, alors :

Bi = Ei + Ki * la somme<selon j> des (XjBjFji Ai) ;

Où :

-Bi, Bj sont les radiosités des patchs i et j (exprimée en W =$m2$)

-Ei est le taux d’émission du patch i.

-Fji est le facteur de forme qui caractérise la proportion d’énergie quittant le patch j qui arrive sur le patch i.

-Ai et Aj sont les surfaces des patchs i et j.

Cette équation caractérise le fait que l’énergie quittant un élément de surface est la somme des lumières émises et réfléchies.

En écrivant cette équation pour l’ensemble des patchs constituant la scène traitée, on débouche sur un système d’équations linéaires ayant les Bi pour inconnues. Ce système d’équations est résolu pour chaque longueur d’ordre traitée. Le calcul des facteurs de forme

I.10. Discussion

Le placage de texture est très largement utilisé en synthèse d'images pour améliorer la qualité des vues obtenues.

Le bump-mapping correspond plutôt à un souci de simplification des scènes par diminution du nombre de facettes et introduction d'un relief de surface simulé.

II .1. Préambule

L'utilisation de photographie aériennes, d'image satellites, de carte scannées et de modèles numérique de terrains amène à mettre en place des stratégies de stockage et de visualisation de ces données. Afin d'obtenir une visualisation en trois dimensions, il est nécessaire de lier ces image appelées textures avec la géométrie du terrain nommée Modèle Numérique de Terrain (MTN). Ces informations sont en pratiques stockées dans trois fichiers différents : MNT, texture, position et projection des données dans un système géo-référencé.

La représentation de terrains en trois dimensions est une composante importante pour la mise en place d'environnements graphiques extérieurs visuels. Nous pouvons, par exemple, citer les simulateurs et de conduite. Dans le cadre de nos travaux, nous nous intéressons à la visualisation 3D de terrains, grâce au logiciel IDL (interactive data interface). Ceci nous amène à combiner deux types de données. [8]

II.2.Le langage IDL

II.2.1.Définition du langage IDL

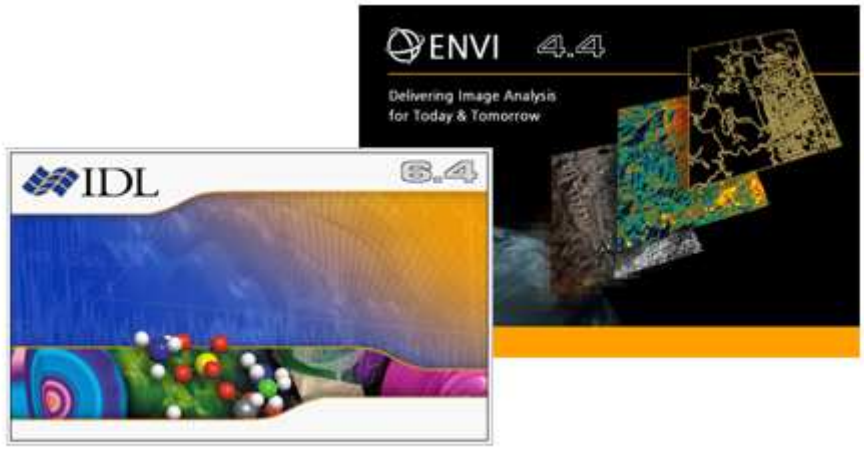


Figure (II.1) : présentationd'IDL

Le langage interactive Data(IDL) est un langage de programmation reconnu dans un grand nombre de discipline, pour crée des visualisations pertinentes de données numériques complexes. Quece soit à partir de programmes d'analyses simples ou d'application à grande échelle, IDL propose l'environnement de programmation complet pour extraire des informations significatives à partir des données.Un grand nombre de découvertes scientifiques découle de l'analyse de données numériques complexes. Si la recherche scientifique est un élément clé de notre activité, nous avons besoin d'outils pour comprendre nos données, mais

également pour utiliser efficacement ces informations : bref un outil performant nous permettant d'inspecter et d'analyser le contenu de nos données. L'utilisation d'un langage de programmation moderne et puissant nous permettra de transformer nos informations numériques en représentations graphiques dynamiques, afin d'interpréter nos données, d'accélérer nos recherches, et de proposer sur le marché des applications pertinentes, étant par ailleurs une solution multiplateforme, elle supporte les systèmes d'exploitation les plus couramment utilisés, aussi bien sous environnement Microsoft Windows®, Mac OS X, Linux ou Solaris.[9]

IDL est un système logiciel propriétaire de Exelis(http://www.exelisvis.com), c'est un langage informatique, facilement compréhensible par n'importe quel utilisateur d'ordinateur-alphabétisé. Il offre toute la puissance et la polyvalence des langages de haut niveau comme Fortran, Matlab et langage C. [9]

IDL inclut un moteur d'analyse puissant et robuste, qui permet d'extraire, à partir des données, les informations statistiques et numériques spécifiques, dont on a besoin.

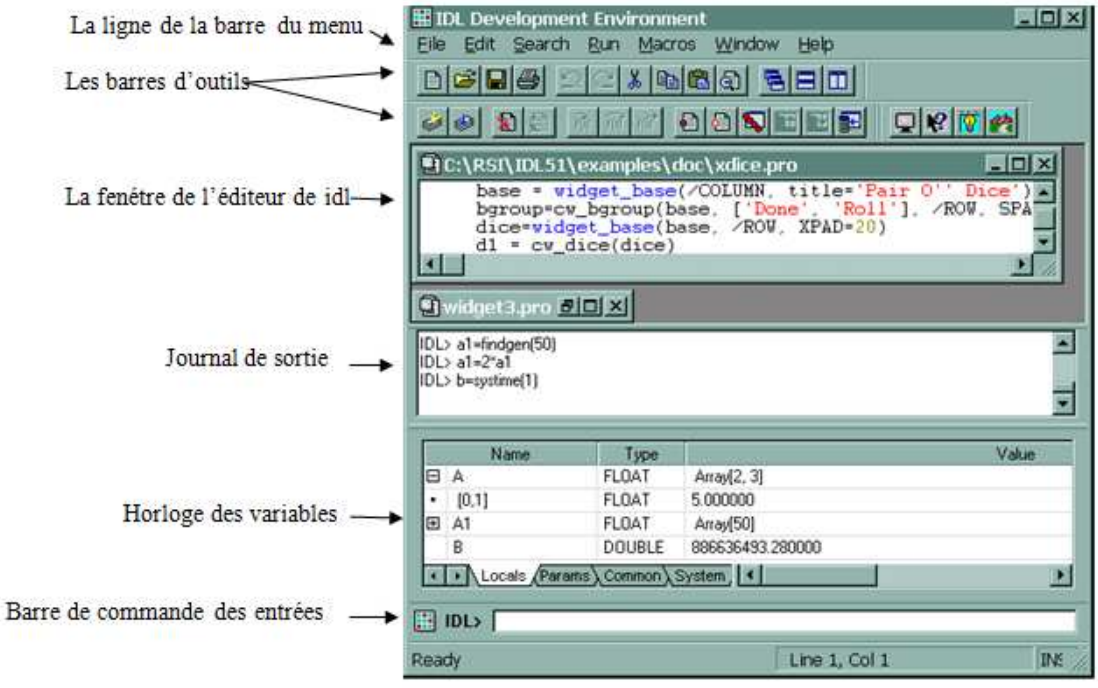


Figure (II.2) : Présentation de la fenêtre d'accueil d'IDL

IDL inclut:

- Une vaste librairie de routines performantes et supportant le 'multithreading', pour l'analyse de données.
- La possibilité d'ajouter de propres routines spécifiques à la librairie existante, en développant des procédures plus rapidement qu'avec d'autres langages.
- Une syntaxe simple, un typage de données dynamique, et des opérations orientées tableaux.
- Des fonctionnalités intégrées pour traiter de nombreuses données, incluant des outils de maillage et d'interpolation 2D et 3D, des routines pour l'ajustement de courbes ou de surfaces, et la capacité d'effectuer des calculs en mode 'multithreading'.

Pour transformer des données numériques complexes en représentations graphiques pertinentes, telles que des lignes, surfaces, images ou contours 2D et 3D, nous avons besoin d'un langage de programmation intuitif et puissant. Nous devons pouvoir produire des résultats de niveau professionnel en un minimum de temps et d'effort. Facile à prendre en main et à utiliser, IDL est le langage de choix des chercheurs et des ingénieurs car il offre un cheminement simple et rapide des données aux résultats.

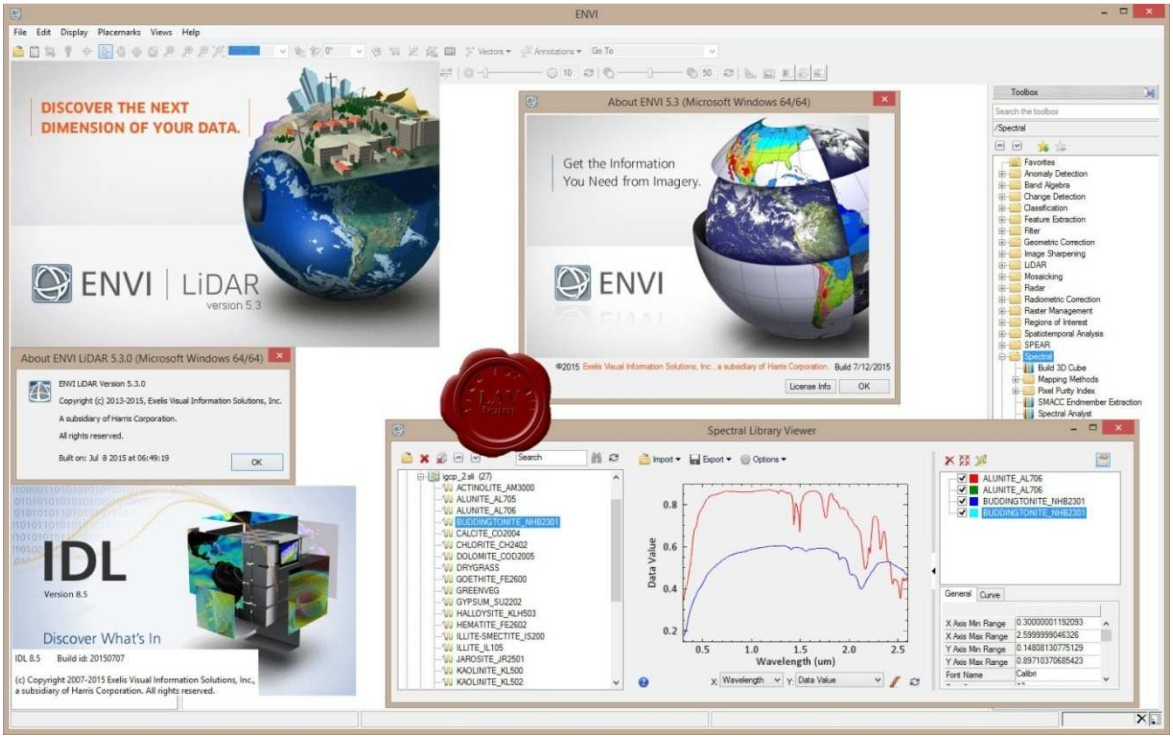


Figure (II.3) : Vue d'ensemble de toutes les fenêtres d'IDL

II.2.2. Fonctionnalités

II.2.2.1.Un Système de typage dynamique

IDL est un langage à typage dynamique. Nous pouvons donc modifier variables et valeurs en cours d'exécution, sans en créer de nouvelles, ni recompiler et ré-exécuter notre code. [10]

Le typage dynamique d'IDL nous offre une plus grande souplesse de programmation. Il nous fait gagner un temps précieux et nous permet de nous concentrer sur l'analyse des données et la visualisation, plutôt que sur des détailsde programmation.

II.2.2.2.Règles et conventions intuitives

Le langage IDL s'appuie sur des règles et conventions intuitives et faciles à assimiler, quels que soient les langages que nous avons pratiqués antérieurement, et même si nous n'avons aucune expérience en programmation. Avec IDL, nous avons besoin de peu de lignes de code pour créer des programmes de visualisation simples ou des applications complètes. Assorti d'une vaste bibliothèque de routines d'analyse et de visualisation précompilées, IDL est le langage de choix pour les programmeurs, quel que soit leur niveau d'expérience. [10]

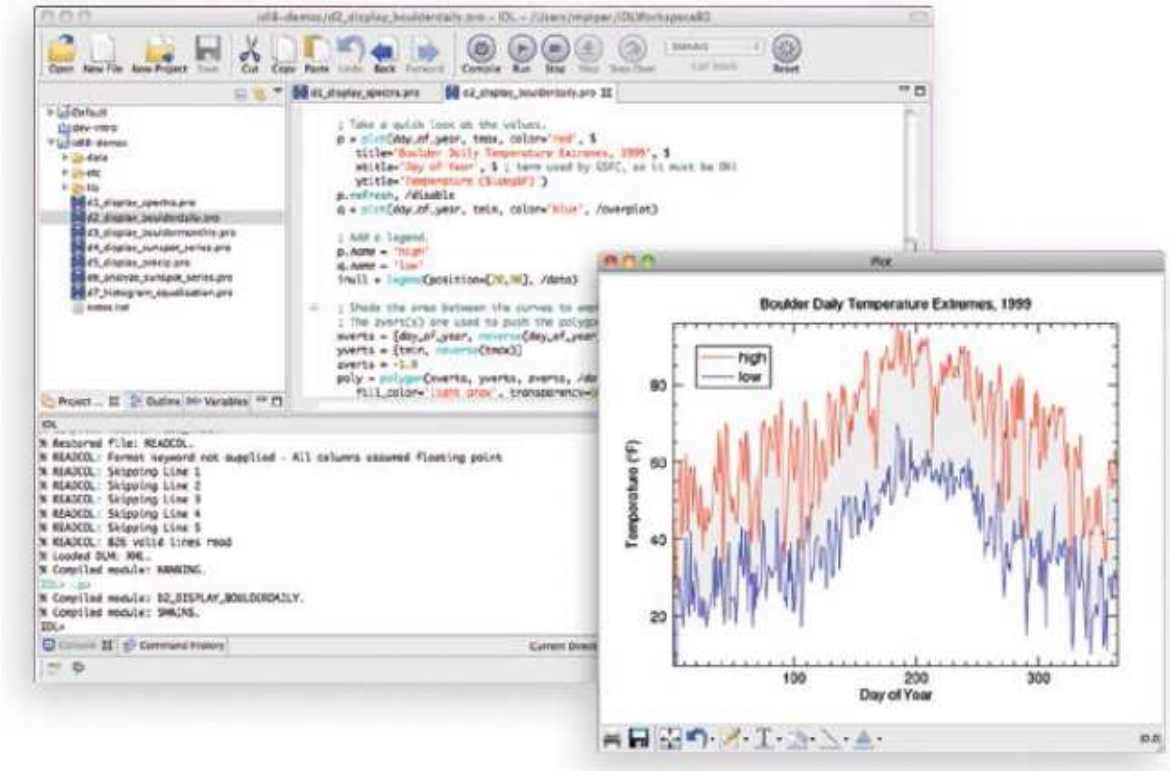


Figure (II.4) :l'environnement de développement IDL et intuitif est simple

II.2.2.3.Un support étendu des formats de données

IDL supporte la majorité des formats et types de donnéesque nous utilisons. Nous avons ainsi accès aux formats courants(TIFF, JPEG, PNG etc.), aux formats de données scientifiqueshiérarchiques (HDF, HDF-EOS, CDF, netCDF) ainsi qu'auxformats binaires et ASC II. IDL étant conçu pour gérer degrands volumes de données multidimensionnelles, aucunetâche ne sera jamais trop complexe : récupération de donnéessur serveurs distants, enregistrement de fichiers sur deslecteurs réseau via des protocoles courants, etc. IDLfonctionne comme client des serveurs HTTP et FTP, et donneaccès à des serveurs standards OGC (Open GeospatialConsortium) tels que les services WMS (Web Map Service) etWCS. [10]

II .2.2.4.L'environnement de développement IDL

Que ce soit pour créer des visualisations spécifiques ou pour concevoir des applications diffusées à grande échelle, nous avons besoin d'un environnement de programmation simple à utiliser. L'environnement de développement IDL - le Workbench - offre une interface intuitive et moderne pour créer rapidement programmes et applications, à partir de routines prédéfinies. [10]

II.2.2.5. Une interface intuitive

Grâce à des fonctionnalités conviviales telles quela barre de tâches, qui incluent les outils que nousutilisons le plus souvent, l'interface intuitive d'IDLnos permet de nous concentrer uniquement surl'analyse des données. [10]

II.2.2.6.Des fonctionnalités de développement flexibles

Avec IDL, le développement est plus facile qu'avec d'autres langages. IDL nous fait gagner du temps grâce à des fonctionnalités prédéfinies telles que l'aide à la saisie, la saisie semi-automatique, le chroma-codage du langage ou les fenêtres d'aide pop-up. Des codes exemples sont également disponibles pour créer rapidement tracés, graphiques et autres visualisations. [10]

II.2.2.7.L'intégration de codes IDL avecd'autres applications

IDL est un langage flexible et extensible qui permetd'interagir dans les deux sens avec d'autreslangages tels que le C, C++, Java, Visual Basic, etc.

Nous pourrons ainsi tirer le meilleur parti de codesexistants en les exportant ou en les importantfacilement depuis ou dans IDL.[10]

II.2.2.8.Le système graphique IDL

Créer des visualisations graphiques est essentiel pourbien comprendre les informations contenues dans nosdonnées numériques.

Le système graphique IDL s'appuie sur la syntaxe intuitivedu langage IDL. Nous pouvons ainsi créer des représentationsgraphiques de haute qualité pour interpréter nos données,partager des résultats avec nos collaborateurs, ou préparerdes publications. La grande facilité de prise en main d'IDLnous permet de créer en toute simplicité des représentationsgraphiques convaincantes. [10]

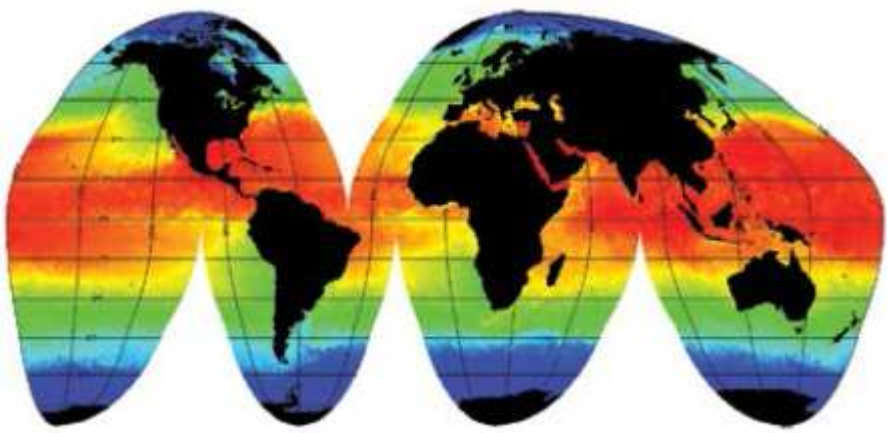


Figure (II.5) : projection cartographique des températures océaniques auteur de globe

II.2.2.9.Créer rapidement des présentations graphiques de qualité avec IDL

IDL est un langage de haut niveau qui facilite la création de visualisations graphiques pertinentes : tracés, cartes, graphiques bidimensionnels, représentations interactives 3D complexes… Conçu pour tirer le meilleur parti de l'accélération matérielle OpenGL, le moteur graphique IDL nous offre un rendu rapide. IDL est ainsi capable de traiter efficacement des grands ensembles de données et de produire les graphiques dont nous avons besoin. [10]

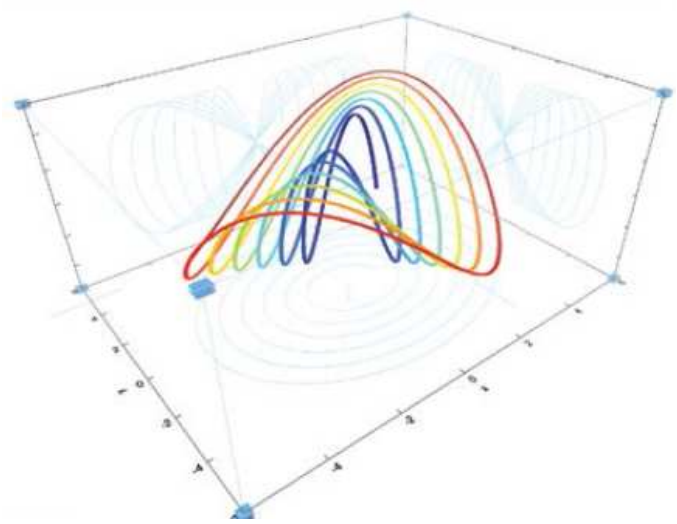


Figure (II.6) :différenteslignes pour mieux distingué les différents ensemble de données

II.2.2.10.Personnaliser les attributs graphiques avec IDL

Le système graphique interactif IDL nous permet de personnaliser l'aspect de nos représentations, que ce soit pour une présentation interne ou une publication internationale. Contrairement à d'autres langages, IDL nous permet de modifier à la volée les styles de lignes, symboles, annotations, polices et couleurs. [10]

II.2.2.11.Générer des sorties graphiques dans tous lesformats images

IDL permet de générer des sorties graphiques en mode programmatique ou interactif dans une large palette deformats images: GIF, JPEG, PNG, etc. afin de les inclure dans nos présentations PowerPoint, Keynote ouHTML. Il est également possible de générer des fichiers PostScript ou PDF pour les inclure dans un article de presse ou des documents TeX ou Word. Grâce aux nombreux formats de sortie disponibles, nous pouvons facilement partager nos résultats avec nos collaborateurs en vue d'une publication ou pour vérification.

II.2.2.12.Traitement d'images et projectionscartographiques

Si nous devons convertir des images brutes en informations pertinentes, IDL dispose d'une vaste bibliothèque de routines de traitement et d'analyse d'images. IDL propose notamment des outils de transformations géométriques, de cartographie, des masques, des méthodes statistiques, des outils de déformation et d'analyse de régions d'intérêt, ainsi que des systèmes de gestion du contraste et des filtres. [10]

II.2.2.13.Traitement du signal

IDL inclut des outils de traitement du signal : outils de décomposition du signal, algorithmes de fenêtrage, routines de lissage, techniques de convolution et filtres numériques pour réduire le bruit, ainsi que des techniques d'analyse de corrélation et de covariance. IDL propose aussi un ensemble d'outils spécifiques pour l'analyse par ondelettes de données multidimensionnelles. [10]

II.2.2.14.Routines mathématiques et statistiques

Le module additionnel d'IDL « Advanced Math and Stats » permet d'ajouter rapidement des fonctionnalités mathématiques et statistiques à nos applications IDL. Il permet de combiner les capacités avancées de visualisationet d'analyse d'IDL avec les routines mathématiques etstatistiques complètes de la librairie numérique C IMSL™:un ensemble d'algorithmes pré-écrits qui s'intègrentfacilement dans nos programmes. [10]

II.3.Modèle Numérique de terrain

II .3.1.Définition de modèle numérique de terrain

Le MNT (modèle numérique de terrain) est une représentation numérique et mathématique de l'altitude d'un point quelconque de la surface terrestre d'une zone géographique, dans un système référentiel bien défini. Même si le MNT représenté sous forme de triangles irréguliers présente certains avantages (Peucker et al… 1978), dans la grande majorité des cas, le MNT est représenté comme un ensemble de mailles régulières, caractérisées chacune par une valeur d'altitude, dans le but d'en faciliter la manipulation et le stockage en mémoire de l'ordinateur pendant les calculs. Le MNT reste l'une des plus importantes sources de données utilisées pour l'extraction de nombreux paramètres utilisés tels que la pente, la direction d'écoulement de l'eau, l'indice topographique, etc. Cependant, il reste des MNT qui en sont issues afin de les quantifier d'une part et d'autre part, d'identifier les méthodes permettant de les propager à travers une application. [11]

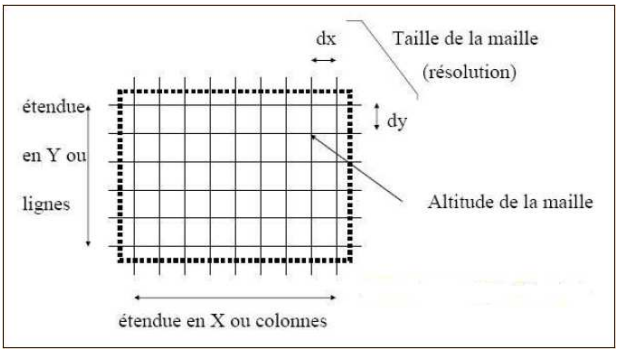
Un MNT permet :

- De reconstituer une vue en images de synthèse du terrain,
- De déterminer une trajectoire de survol du terrain.
- De calculer des surfaces ou des volumes.

- De tracer des profils topographiques.
- Extraction des paramètres du terrain
- Tracés des profils topographiques
- Modélisation de l'écoulement de l'eau ou de la masse du mouvement (par exemple pour les avalanches et glissements de terrain)
- Création de cartes en relief
- Rendu de visualisation en 3D.
- Rectification géométrique de photographie aérienne ou d'imagerie satellitaire.
- Les analyses de terrain en géomorphologie et géographie physique
- Systèmes d'information géographique (SIG)
- Ingénierie et conception des infrastructures
- Systèmes de positionnement global (GPS)
- Simulation de vol
- Précision agricoles et forestières
- Analyse de surface

On peut également distinguer les MNT selon le type de maillage utilisé :

- Maillage régulier carré (raster),
- Maillage triangulaire régulier,
- Maillage triangulaire quelconque (TIN)



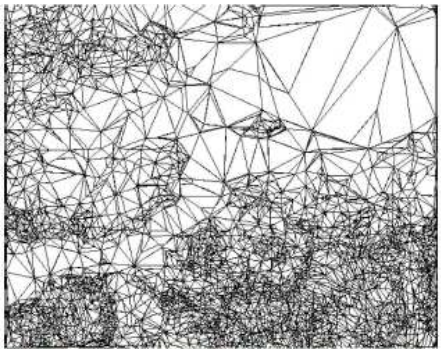


Figure (II.7) : structure de type raster**Figure (II.8) :** structure de type TIN

II.3.2. Importance et domaines d'application

Les **MNT** sont utilisés dans de nombreuses applications en sciences de la terre et eningénierie. Leurs premières utilisations remontent aux années 1950. La seule applicationdu MNT durant cette période concernait les tracés de routes (tracés de profils du terrain etcalcul de courbures).

Les MNT se sont avérés être une méthode importante pour la modélisation et l'analysedes données topographiques spatiales. Leurs domaines d'application sont variés. Ils sont

Par exemple utilisés en :

- Génie civil.
- Sciences de la terre
- Planification et gestion de ressources-
- Applications militaires.

II.3.2.1.Génie civil

Les ingénieurs sont principalement intéressés à utiliser les MNT pour faire face **aux**problèmes concernant la conception des routes, la planification des sites et le calculvolumétrique des barrages. [11]

II.3.2.2.Sciences de la terre

Laterre ou les applications géo-scientifiques se concentrent principalement sur desfonctions spécifiques pour la modélisation, l'analyse et l'interprétation de la morphologiedu terrain. Celles-ci peuvent inclure la simulation, la classification géomorphologique etla topographie géologique. En géologie, la production des profils de pente pour créer descartes ombragées est une utilisation populaire qui fait usage du MNT. [11]

II.3.2.3.Planification et gestion de ressources

Les applications caractérisant mieux ce domaine incluent l'emplacement des sites, lesmodèles d'érosion de sol et les modèles potentiels de dispersion de la pollution. [11]

II.3.2.4.Applications militaires

Le domaine militaire est non seulement unprincipal consommateur des MNT mais il est également un producteur significatif. Presque chaque aspect de l'environnement militairedépend d'une compréhension fiable et précise du terrain, de l'altitude et de la pente de lasurface de la terre. L'utilisation militaire des MNT combine les méthodes de tous lesdomaines d'application précédents et leur objectif final est très spécialisé, par exemple :l'analyse de l'inter-visibilité pour la gestion de champ de bataille, l'affichage en troisdimensions pour les systèmes de contrôles d'mes, la simulation de vol et la préparationde missions sur des régions parfois difficiles. [11]

II.4. La reconstruction tridimensionnelle des surfaces

Habituellement, les nuages de points ne sont pas suffisants comme résultat finale de documentation. Ils pourraient être considérés comme des moyens d'archiver la géométrie d'un objet en prévision d'exploitation future. Le but de la reconstruction d'une surface à partir d'un nuage de points peut être énoncé comme suit : étant donné un ensemble de points P supposés décrire une surface inconnue S, crée un modèle de surface S qui approxime S. Dans ce but, beaucoup de techniques existent aujourd'hui.

II.4.1. Les procédures de la reconstruction automatiques

Le maillage polygonal est habituellement la méthode la plus adapté pour représenter correctement les résultats des mesures, fournissant une description avec les données d'entrée.

Beaucoup de méthodes ont été développées [Mencl et all, 1998] pour une représentation (triangulaire) régulière et continue de maille à partir d'un nuage de points. Une fois la surface polygonale déterminée, diverses techniques de post-traitement peuvent être employées pour l'optimisation d'un résultat : lissage, remplissage de trous, etc...La convention des données mesurées en une surface polygonale cohérente et généralement basés sur quatre étapes :

II.4.2. Prétraitement

Dans cette phase, on élimine des données bruitées ou bien on échantillonne le nuage de points pour réduire le volume de données à traiter et donc le temps de calcul ;

II.4.3. Détermination de la topologie globale de la surface de l'objet :

Les relations de voisinage entre les parties adjacentes de la surface doivent être dérivées. Cette opération a besoin typiquement d'informations additionnelles (normales) pour

prise en compte de contraintes de discontinuité, dans le but de préserver principalement la définition des bords.

II.4.4. Génération de la maille :

Des mailles triangulaires sont créées répondant à certaines exigences de qualité, par exemple la taille et la forme des mailles.

Dans notre cas, la génération de maille se fait selon la triangulation 2D cette technique consiste en la génération de triangles qui se rencontre sur une arête et sur les sommets partagés sont générés sur le plan et re-projetés dans l'espace.

Une des méthodes de reconstruction très connue est la triangulation de Delaunay qui optimise simultanément les critères de qualité mentionnés précédemment.

Le critère de Delaunay assure en effet qu'aucun sommet ne se relie à l'intérieur de chacun des cercles circonscrits des triangles dans les réseaux.

II.4.5. Post-traitement :

Une fois le modèle créé, des opérations d'édition sont généralement appliquées pour raffiner et perfectionner la surface polygonale résultante.

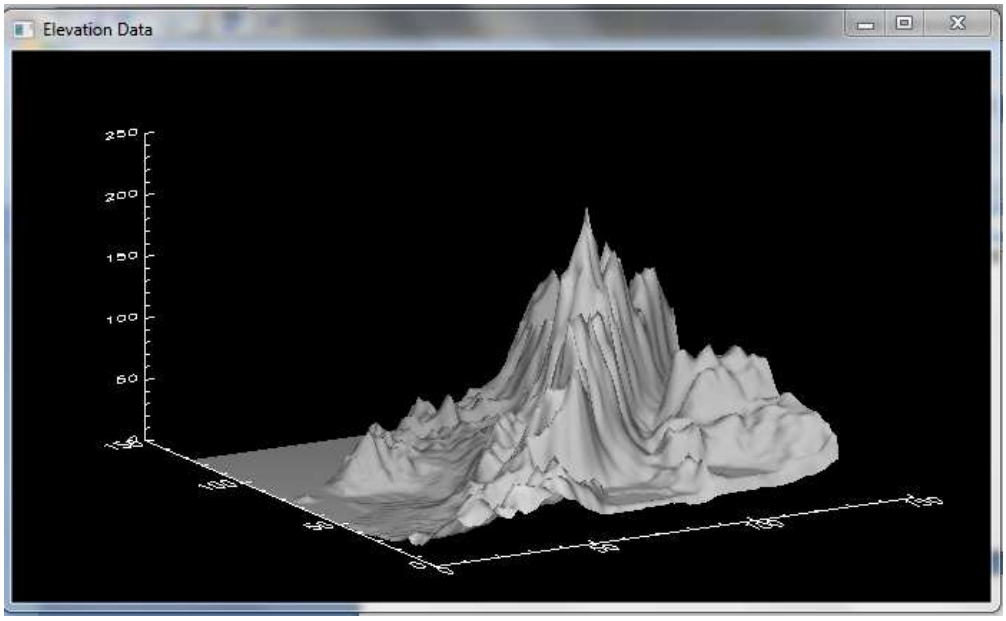
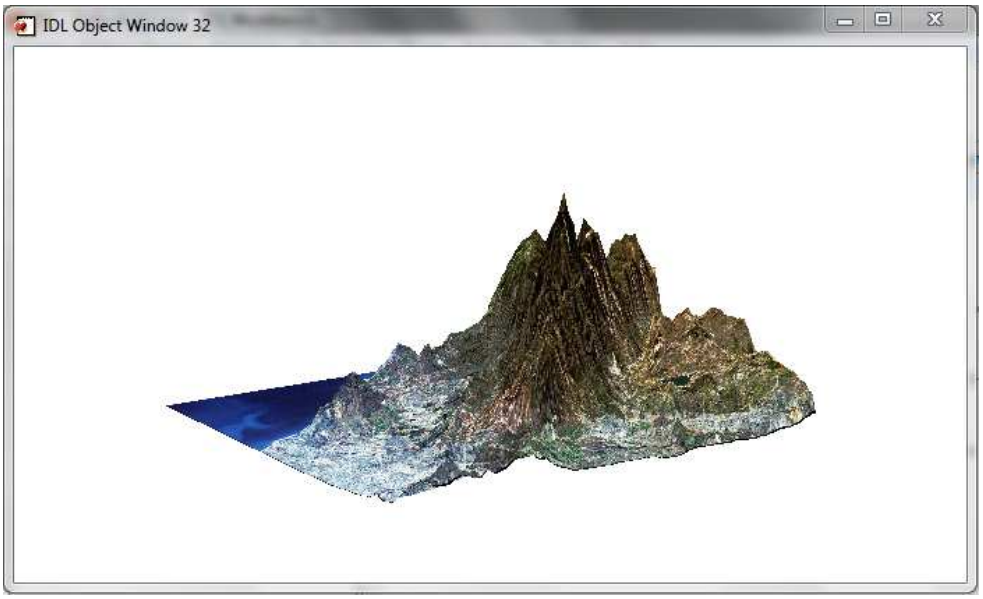


Figure (II.9) : post traitement

II.4.6 Application de la texture :

Une fois la construction géométrique effectuée, il faut plaquer sur les triangles obtenus une texture ou une image.



Figure(II.10) : Application de la texture

La texturation de la surface polygonale crée consiste, techniquement, en un remplissage de polygone. Considérons une feuille de papier que l'on souhaite coller sur une sphère : on ne peut le faire proprement qu'à condition de couper la feuille à des endroits bien précis. C'est ce qu'il se passe en pratique dans le mapping : l'idée est d'associer à chaque sommet de l'objet une coordonnée (u,v) dans la texture à appliquer. Il faut donc trouver une fonction F de mapping de cette forme :

$(u,v) = F(x,y,z)$ [12]

Ou (x,y,z) sont les coordonnées spatiales de l'objet à mapper. Cette projection des valeurs de u et v sur l'objet peut être considérée comme une fonction de modélisation. On peut donc pour trouver cette fonction utiliser les fonctions classiques de projection. Deux fonctions courantes de projection sont les fonctions de mapping cylindrique et sphérique. Considérons d'abord le mapping cylindrique : il est facile de donner une définition paramétrique de la surface courbe d'un cylindre.

En effet tout point de la surface de rayon R et de hauteur h est représenté de la manière suivante :

$X = R \cos \theta$

Y = h

Z = R cos θ

Avec

$\pi <= \theta <= \pi$

La variable h varie entre min et h max, la variableθ varie entre θ deb et θ fin.

On peut associer les valeurs (u,v) de la texture avec un point du cylindre par :

(u,v) = ((θ-θdeb) / (θfin- θ deb),(h - h min) / (h max – h min))

Pour un mapping sphérique, les choses sont peu plus compliquées. En effet mapper un plan sur une sphère produit des distorsions très complexes au niveau des pôles. Considérons comme exemple le mapping d'une texture sur une partie de sphère. La paramétrisation donne :

X = R cos θ cos φ

Y = R sin φ

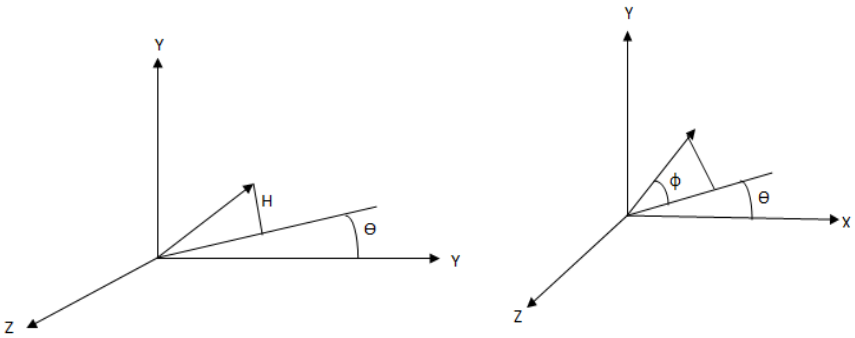
Z = R sin θ cos φ

Avec - π <= θ<= π et – π /2 <= <= π /2

La variable θ varie entre θ min et θ max, la variable φ varie entre φ min et φ max.

Nous pouvons alors définir une fonction de mapping comme au paravent :

(u,v) = ((θ – θ min) / (θ max – θ min), (φ – φ min) / (φ max – φ min))



Figure(II.11) : para-métrisation

Grace à ces fonctions, il est possible de mapper une texture sur un objet sphérique ou cylindrique, et toutes les transformations bi - dimensionnelles standards peuvent être appliquées aux coordonnées de cette texture. Mais en générale, l'objet sur lequel nous souhaitons coller une texture n'est pas parfaitement cylindrique ou sphérique.

On essaye alors de se ramener dans le cas d'un cylindre ou d'une sphère auquel on appliquera diverses transformations pour s'approcher le plus possible de l'objet désiré.

II.5. Discussion

Dans ce chapitre nous avons présenté le logiciel IDL, ses fonctionnalités et les différentes étapes pour la conception et l'implémentation d'un programme sous le même environnement dont nous présenterons un exemple qui, en plus, est notre travail.

III.Application

III.1.Préambule

L'implémentation est la phase la plus importante dans notre application. Le choix des outils de développement influence énormément sur le coût en temps de programmation, ainsi que sur la flexibilité du produit à réaliser.

Dans ce chapitre, nous allons commencer par l'élaboration des programmes destinés à l'implémentation sous environnement IDL puis nous clôturerons notre travail par l'étalage des résultats obtenus. Rappelons que l'objectif de notre travail est de plaquer une image deux dimensions sur des objets à trois dimensions

III.2. Environnement de travail

L'environnement de travail est constitué par deux parties nommées environnement matériel et environnement logiciel.

III.2.1. Environnement matériel

Le développement de l'environnement matériel est caractérisé par :

1. Système d'exploitation : Windows 7 Professionnel.
2. CPU : intel(R) Core i3-2348M CPU@ 2.30GHZ 2.30GHZ
3. Mémoire : 4.00 GO

III.2.2. Environnement logiciel

L'environnement logiciel nécissite le composant suivant :

IDL Workbenchversion 7.01 build id 20080320 developer par Eclipse foundation, Inc

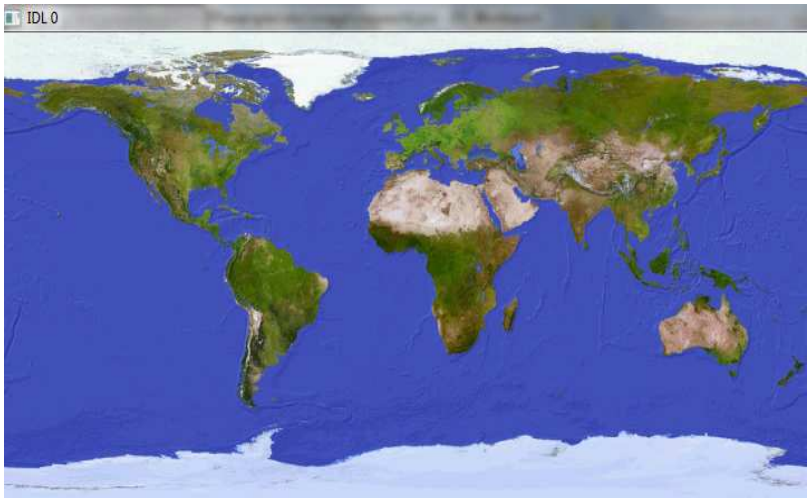
III.3.Développement de l'application

La première étape consiste à sélectionner et à importer des images 2D sous-forme (JPEG, PNG, DATA… etc.), qui sont stockées dans le répertoire exemple, sous-fichier data sous IDL.

Le programme que nous avons conçu pour obtenir le chemin vers le fichier d'image sous IDL :

```
; Lire dans le fichier d'image

World = READ_PNG (FILEPATH ('avhrr.png', $

SUBDIRECTORY = ['examples', 'data']), R, G,B)

; Préparer le dispositif d'affichage et de charger la table des couleurs

DEVICE, DECOMPOSED = 0, RETAIN = 2
TVLCT, R, G, B

; Obtenir la taille de la matrice d'image

worldSize = SIZE (world, /DIMENSIONS)

; Utilisez les dimensions renvoyées pour créer une fenêtre d'affichage, et afficher l'image originale.

WINDOW, 0, XSIZE = worldSize [0], YSIZE = worldSize[1]
TV, world

End
```

L'image sélectionnée et lue par notre programme est donnée en figure III.1 :

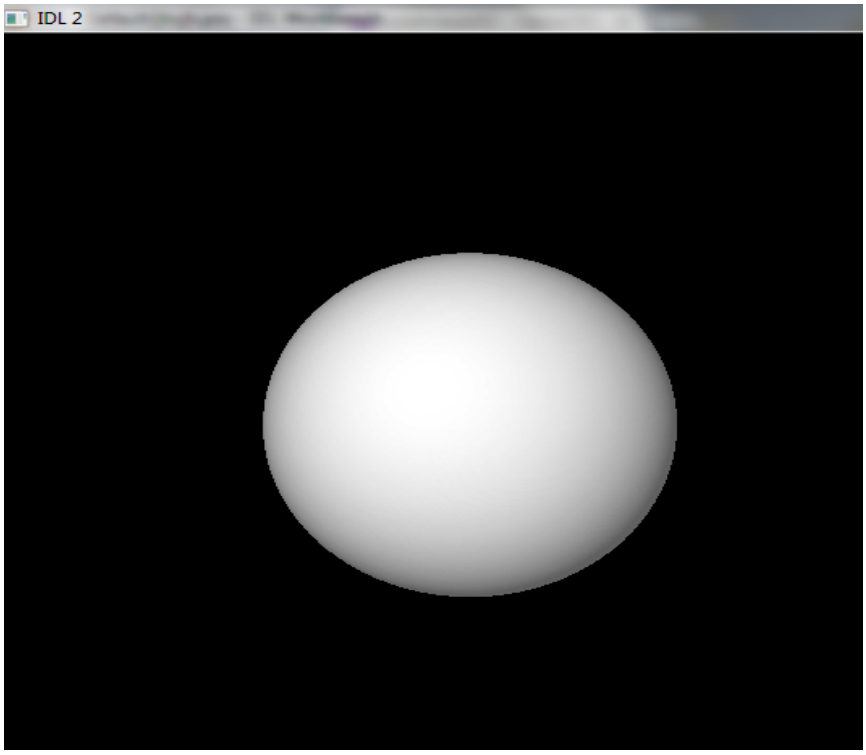


Figure(III.1). Exemple d'image

Dans la seconde étape on va créer des différents objets 3D et de visualiser les fenêtres sur l'écran.

Le programme suivant permet de créer une sphère :

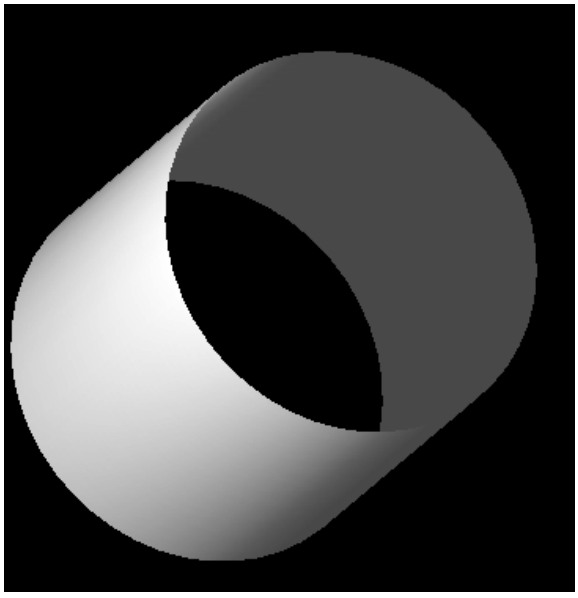
```
; crée une sphère de rayon de valeur 0.25
MESH_OBJ, 4, Vertex List, Polygon_List, $
REPLICATE (0.25, 214,214)

; Créer la fenêtre de taille x =512, y=512
WINDOW, 2, XSIZE=512, YSIZE=512
CREATE_VIEW, WINX=512, WINY=512, AX=140.0, AY= (140.0), ZOOM=1.0

; Rendre les mailles
SET_SHADING, LIGHT= [-0.5, 0.5, 2.0], REJECT=1
TVSCL, POLYSHADE (Vertex_List, Polygon_List, /NORMAL)
,End
```



Figure(III.2) : visualisions de la sphère

Le programme suivant réalise un cylindre :

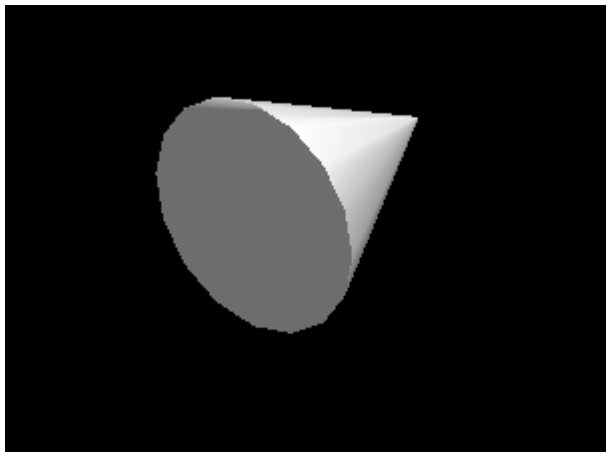
```
Création d'un cylindre de rayon constant de valeur  0.25
MESH_OBJ, 3, Vertex_List, Polygon_List, $
Replicate(0.25, 48, 64), P4=0.5

; Transformation des sommets
T3D, /RESET
T3D, ROTATE=[0.0, 30.0, 0.0]
T3D, ROTATE=[0.0, 0.0, 40.0]
T3D, TRANSLATE=[0.25, 0.25, 0.25]
VERTEX_LIST = VERT_T3D(Vertex_List)

; Crée une fenêtre d'affichage
WINDOW, 0, XSIZE=512, YSIZE=512
CREATE_VIEW, WINX=512, WINY=512

; rendre les mailles :
SET_SHADING, LIGHT=[-0.5, 0.5, 2.0], REJECT=0
TVSCL, POLYSHADE (Vertex_List, Polygon_List, /NORMAL)
End
```



Figure(III.3) : image illustrant un cylindre

Le programme d'un cône est donné comme suit:

```
; Créer un cône (surface de révolution)

MESH_OBJ, 6, Vertex_List, Polygon_List, $

  [[0.75, 0.0, 0.25], [0.5, 0.0, 0.75]], $
  P1=16, P2= [0.5, 0.0, 0.0]


; Crée la fenêtre d'affichage
WINDOW, 1, XSIZE=512, YSIZE=512
CREATE_VIEW, WINX=512, WINY=512, AX=30.0, AY= (140.0), ZOOM=0.5


; Rendre les mailles
SET_SHADING, LIGHT= [-0.5, 0.5, 2.0], REJECT=0
TVSCL, POLYSHADE (Vertex_List, Polygon_List, /DATA, /T3D)

End
```

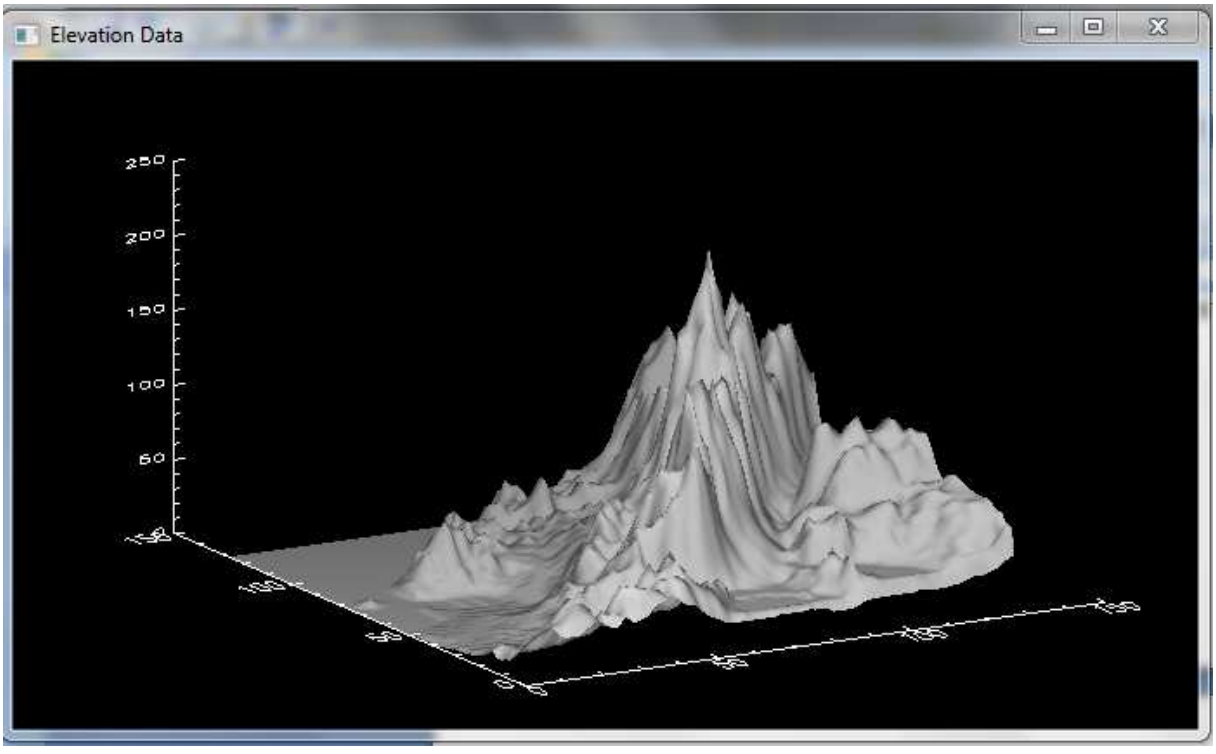


Figure(III.4) : visualisions d un cône

Le programme suivant permet de créer un modèle numérique de terrain(MNT) :

```
PRO MNT
;improrter une image 'elev_t.jpg' qui se trouve dans le sous-fichier data
imageFile = FILEPATH('elev_t.jpg', $
SUBDIRECTORY = ['examples', 'data'])
```

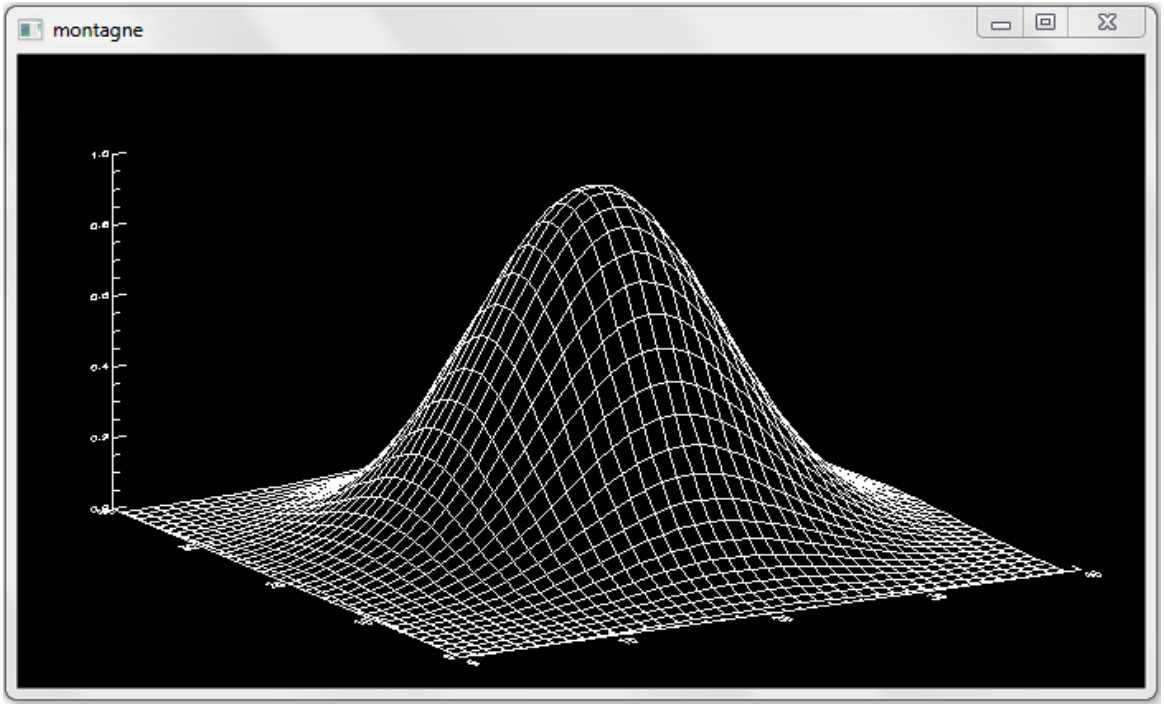
```
;lire l'image
READ_JPEG, imageFile, image

;Sélectionnez le fichier de DEM
demFile = FILEPATH('elevbin.dat', $
SUBDIRECTORY = ['examples', 'data'])
; lire le fichier demfile
dem = READ_BINARY(demFile, DATA_DIMS = [64, 64])

;réechantillonner la surface
;dem = CONGRID(dem, 128, 128,/INTERP
DEVICE, DECOMPOSED = 0
;afficher la surface et donner le titre
WINDOW, 0, TITLE = 'modéle numérique de terrain  '
SHADE_SURF, dem

end
```



Figure(III.5) : visualisation d'un MNT

Le Programme que nous avons conçu pour créerla gaussienne est donné :

```
PRO gaussienne


; Création d'un tableau de dimension 40*40 dont chaque élément égale à la distance.
image=SHIFT(DIST(40),20,20)
image=EXP(-(image/10)^2)

;extraire la dimension de la surface crée.
S=SIZE(image,/DIMENSIONS)
Xsize=s[0]
Ysize=s[1]
DEVICE,DECOMPOSED=0
WINDOW,0,TITLE="montagne"
Surface,image
END
```



Figure(III.6) : visualisation de la gaussienne

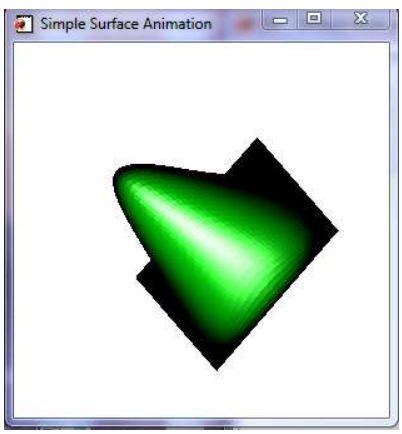
Dans cette étape on a appliqué l'ombrage de Gouraud et flat et on a crée une source de lumière.

Le programme que nous avons conçu pour l'animation d'une surface est le suivant:

```
PRO animation de surface
FUNCTION MyRotator::Init, oModel
self.oModel = oModel
return, 1
END

PRO MyRotator::Cleanup
END

PRO MyRotator::OnTimer, oWin
self.oModel->Rotate, [0,1,0], 10
oWin->Draw
END
PRO MyRotator__define
void = { MyRotator, $
oModel: OBJ_NEW() $
        }
END

Pro animation surface
; Creation de l'objet
zdata = HANNING (50,50)
osurface = obj_new('IDLgrSurface', zdata, STYLE= 3 , $
    COLOR= [0, 255, 0], BOTTOM= [255,0,0], SHADING =1
    )
;creationd'une source lumineuse
oLight = OBJ_NEW('IDLgrLight', type=2, location = [1, -1, 1])
olightmodel = OBJ_NEW('IDLgrModel')
olightmodel->ADD, oLight
```

```
; afficher la surface
oModel = OBJ_NEW('IDLgrModel')
oView = OBJ_NEW('IDLgrView')
oModel->Add, oSurface
oView->Add, oModel
oView->Add, olightmodel

; centrer l'objet crée dans la fenètre d'affichage
oSurface->GetProperty, XRANGE=xr, YRANGE=yr, ZRANGE=zr
xc=NORM_COORD(xr)
xc[0] = xc[0] - 0.5
yc=NORM_COORD(yr)
yc[0] = yc[0] - 0.5
zc=NORM_COORD(zr)
zc[0] = zc[0] - 0.3

osurface->SetProperty, XCOORD_CONV=xc, YCOORD_CONV=yc, $
ZCOORD_CONV=zc

;manoeuvrer l'objet crée
omodel->ROTATE, [1,0,0], -60
omodel->ROTATE, [0,1,0], 60
omodel->ROTATE, [0,0,1], 90
;donner le titre à la surface crée
oWin = OBJ_NEW('IDLitWindow', DIMENSIONS=[300,300], $
    TITLE="Simple Surface Animation")
oWin->Add, oView
oRotator = OBJ_NEW('MyRotator', oModel)
oWin->AddWindowEventObserver, oRotator

; régler lz vitesse de rotation de l'objet
oWin->SetTimerInterval, 0.04
oWin->SetEventMask, /TIMER_EVENTS
```

END



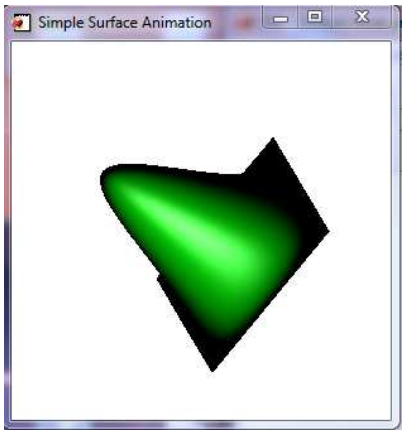
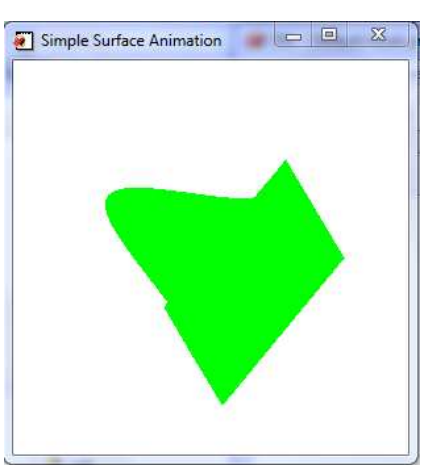


Figure (III.7) : ombrage de flat **Figure (III.8) :** ombrage de Gouraud



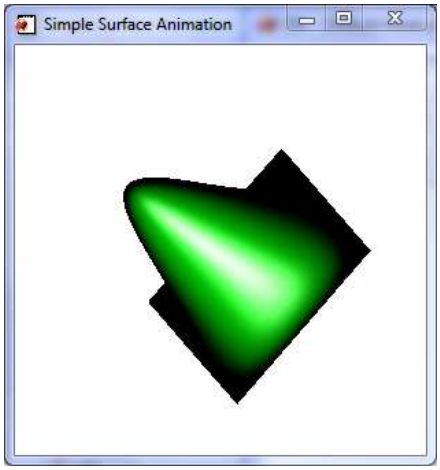


Figure (III.9) : gaussienne sans source lumineusesource **Figure (III.10) :** gaussienne avec source lumineuse

Après avoir élaboré nos programmes, nous allons effectuer le plaquage de texture (pour notre cas, nous avons choisi un damier et une image satellitaire) sur les objets que nous avons créés.

Le programme de plaquage d'un damier sur un cylindre est le suivant :

PRO damier_cylindre

;improrter une image'worldelv.dat', qui se trouve dans le sous-fichier data

```
;file = FILEPATH('worldelv.dat', $
;SUBDIRECTORY = ['examples', 'data'])

; lirel'image
;image = READ_BINARY(file, DATA_DIMS = [360, 360])
image = READ_PNG(FILEPATH('Damier.PNG', $
  SUBDIRECTORY = ['examples', 'data']), R, G, B)
dims = SIZE(img, /DIMENSIONS)

;création d'un cylindre de rayon constant.
MESH_OBJ, 3, Vertex_List, Polygon_List, $
Replicate(0.25, 64, 64), P4=0.5



; positionner l'objet
T3D, /RESET
T3D, ROTATE= [0.0, 30.0, 0.0]
T3D, ROTATE= [0.0, 0.0, 40.0]
T3D, TRANSLATE= [0.25, 0.25, 0.25]
VERTEX_LIST = VERT_T3D(Vertex_List)

; créer  une palette de couleur
oModel = OBJ_NEW('IDLgrModel')
oPalette = OBJ_NEW('IDLgrPalette')
oPalette -> LOADCT, 33
oPalette ->SetRGB, 255, 255, 255, 255
oImage = OBJ_NEW('IDLgrImage', image, PALETTE = oPalette)

;coordonnées de la texture
vector = FINDGEN(64)/63.
texure_coordinates = FLTARR(2, 64, 64)
texure_coordinates[0, *, *] = vector # REPLICATE(1., 64)
texure_coordinates[1, *, *] = REPLICATE(1., 64) # vector
```

```
; creation de l’objet polygone contenant les données
oPolygons = OBJ_NEW('IDLgrPolygon', SHADING = 1, $
  DATA = Vertex_List, POLYGONS = Polygon_List, $
COLOR = [255, 255, 255], $
  TEXTURE_COORD = texure_coordinates, $
  TEXTURE_MAP = oImage, /TEXTURE_INTERP)

oModel -> ADD, oPolygons
;rotation de l’objet
oModel -> ROTATE, [1, 0, 0], -90
oModel -> ROTATE, [0, 1, 0], -90

; afficher les résultats en utilisant XOBJVIEW (un utilitaire interactif qui nous permet de pivoter et de redimensionner

XOBJVIEW, oModel, /BLOCK
OBJ_DESTROY, [oModel, oImage, oPalette]
```

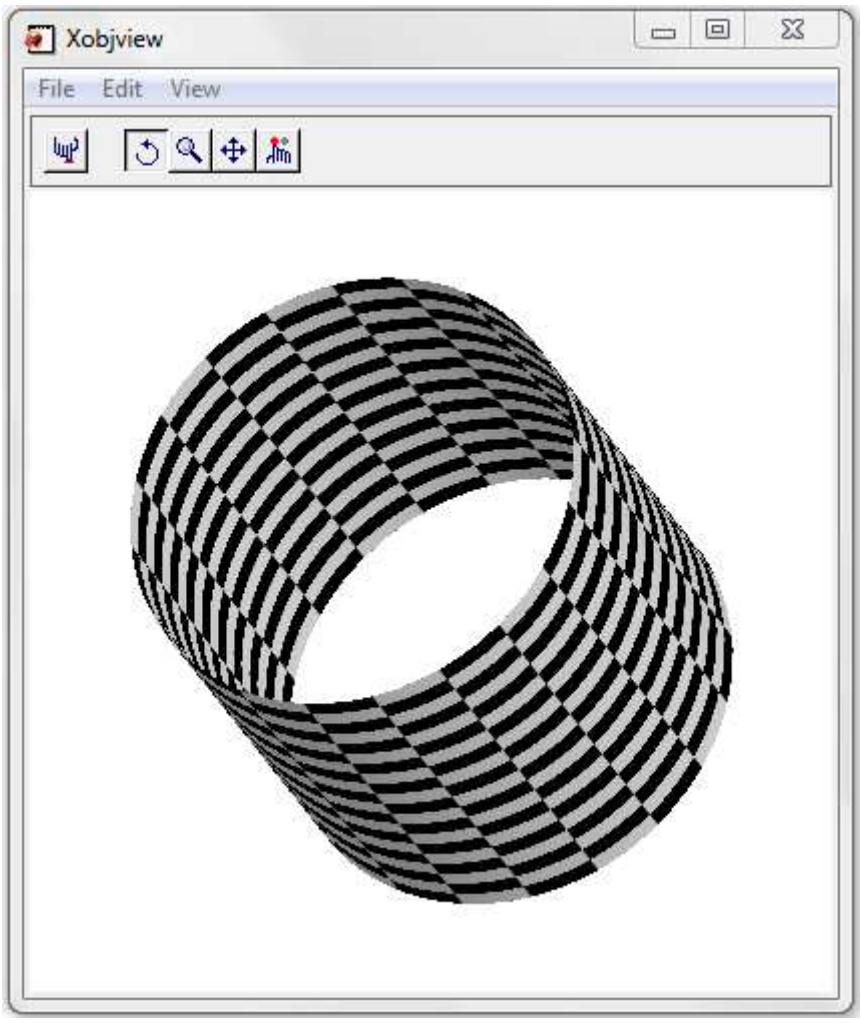


Figure (III.11) : plaquage d'un damier sur un cylindre

Pour la sphère, nous donnons le programme correspondant:

PROdamier_sphere

```
;importer la texture damier créer au préable est stocké dans le sous fichier data
file = FILEPATH('worldelv.dat', $
;SUBDIRECTORY = ['examples', 'data'])
;image = READ_BINARY(file, DATA_DIMS = [360, 360])
image = READ_PNG(FILEPATH('Damier.PNG', $
  SUBDIRECTORY = ['examples', 'data']), R, G, B)
dims = SIZE(img, /DIMENSIONS)
```

```
;créer une sphére avec un rayon de valeur 0.25
MESH_OBJ, 4, vertices, polygons, REPLICATE(0.25, 101, 101)
oModel = OBJ_NEW('IDLgrModel')

;créer une palette de couleur
oPalette = OBJ_NEW('IDLgrPalette')
oPalette ->LOADCT, 33
oPalette ->SetRGB, 255, 255, 255, 255
oImage = OBJ_NEW('IDLgrImage', image, PALETTE = oPalette)

; coordonnées de la texture
vector = FINDGEN(101)/100.
texure_coordinates = FLTARR(2, 101, 101)
texure_coordinates[0, *, *] = vector # REPLICATE(1., 101)
texure_coordinates[1, *, *] = REPLICATE(1., 101) # vector

; application la texture a l'objet
oPolygons = OBJ_NEW('IDLgrPolygon', SHADING = 1, $
DATA = vertices, POLYGONS = polygons, $
  COLOR = [255, 255, 255], $
  TEXTURE_COORD = texure_coordinates, $
TEXTURE_MAP = oImage, /TEXTURE_INTERP)
oModel -> ADD, oPolygons

; faire tourner l'objet
oModel ->ROTATE, [1, 0, 0], -90
oModel ->ROTATE, [0, 1, 0], -90

; afficher les résultats en utilisant XOBJVIEW (un utilitaire interactif qui nous permet de pivoter et de redimensionner
XOBJVIEW, oModel, /BLOCK
OBJ_DESTROY, [oModel, oImage, oPalette]
END
```

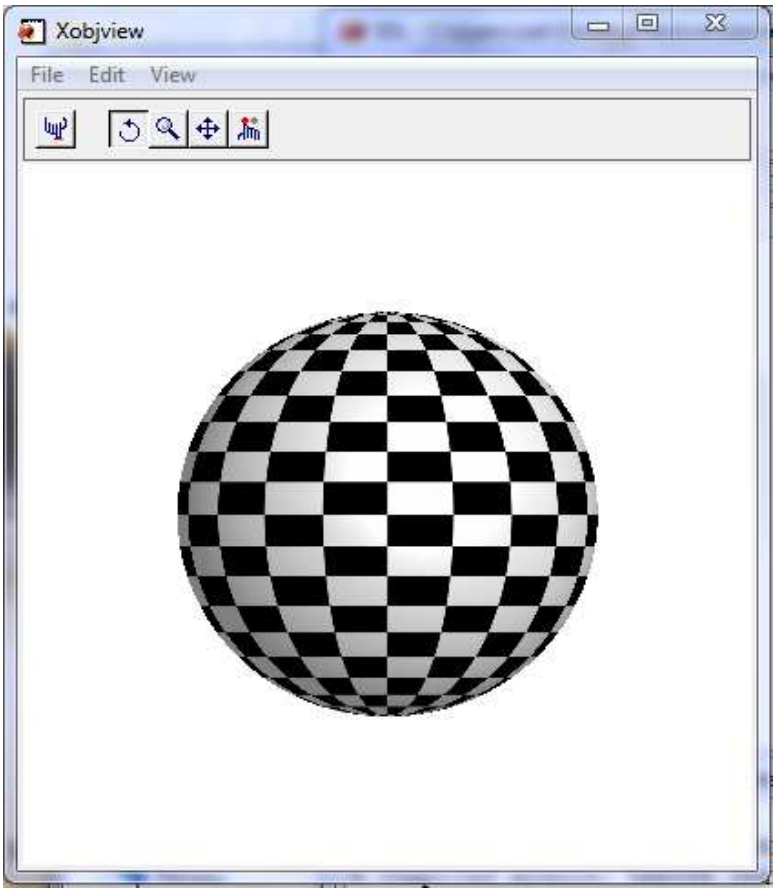


Figure (III.12) : plaquage d'un damier sur une sphère

Le programme que nous avons conçu pour illustrer le plaquage d'un damier sur une gaussienne est le suivant:

```
PRO gaussienne_damier

; importer la texture

image = READ_PNG(FILEPATH('Damier.PNG', $

 SUBDIRECTORY = ['examples', 'data']), R, G, B)

dims = SIZE(img, /DIMENSIONS)


; creation de l'objet  et definir la palette de couleur
zdata = HANNING (40, 40)
```

```
osurface = obj_new('IDLgrSurface', zdata, STYLE=2, $
    COLOR=[0,0,255], BOTTOM=[255, 0, 0], SHADING=1)
oPalette = OBJ_NEW('IDLgrPalette')
oPalette -> LOADCT, 33
oPalette -> SetRGB,255,255 ,255, 255
oImage = OBJ_NEW('IDLgrImage', image, PALETTE = oPalette)

;crée des objet d'affichage
oModel = OBJ_NEW('IDLgrModel')
oView = OBJ_NEW('IDLgrView')

; convertir les coordonnées de données aux coordonnées normales
oSurface->GetProperty, XRANGE=xr, YRANGE=yr, ZRANGE=zr
xc=NORM_COORD(xr)
xc[0] = xc[0] - 0.5
yc=NORM_COORD(yr)
yc[0] = yc[0] - 0.5
zc=NORM_COORD(zr)
zc[0] = zc[0] - 0.5

osurface->SetProperty, XCOORD_CONV=xc, YCOORD_CONV=yc, $
    ZCOORD_CONV=zc

oSurface ->SetProperty, TEXTURE_MAP = oImage, $
COLOR = [255, 255, 255]



; régler la retation de l'objet
omodel->ROTATE, [1,0,0], -90
omodel->ROTATE, [0,1,0], 30
omodel->ROTATE, [1,0,0], 30

oModel->Add, oSurface
oView->Add, oModel
```

```
;crée une nouvelle classe (MayRatator)
oWin = OBJ_NEW('IDLitWindow', DIMENSIONS=[800,800], $
    TITLE="Simple Surface Animation")
oWin->Add, oView
;oModel -> ROTATE, [1, 0, 0], -90
;oModel -> ROTATE, [0, 1, 0], -90

; afficher les résultats en utilisant XOBJVIEW (un utilitaire interactif qui nous permet de pivoter et de redimensionner
XOBJVIEW, oModel, /BLOCK
OBJ_DESTROY, [oModel, oImage, oPalette]
END
```

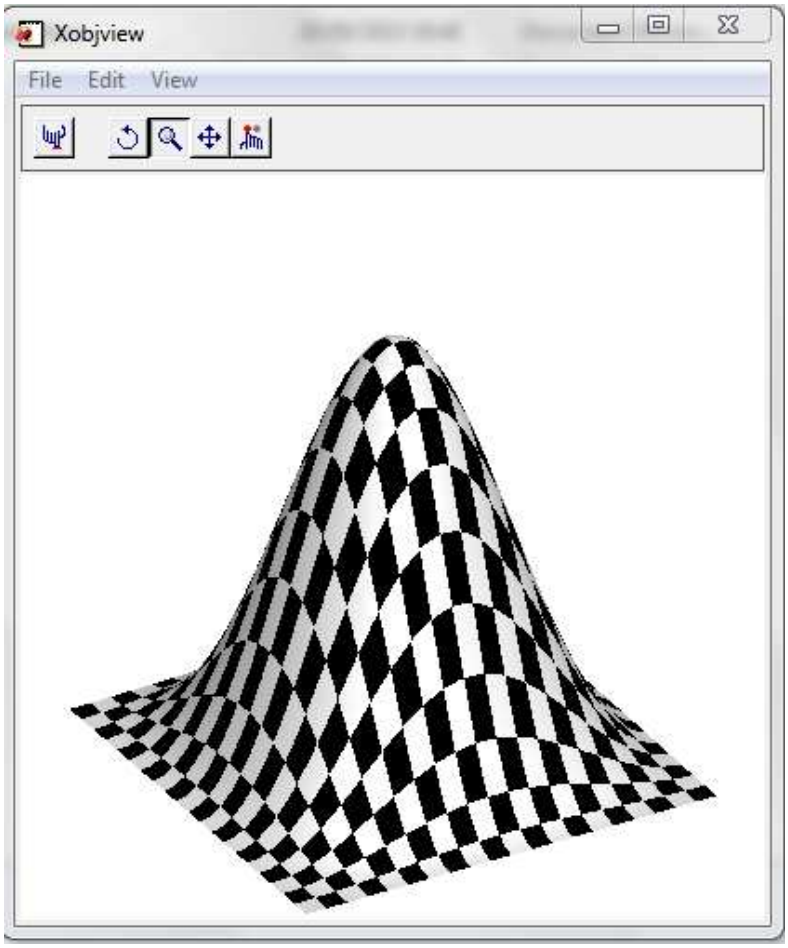


Figure (III.13) : plaquage d'un damier sur une gaussienne

Le programme que nous avons conçu pour réaliser le plaquage d'une image satellite sur MNT est donné ci-dessous :

```
PROElevation_Object

; importer une imge'elev_t.jpg' qui se trouve dans le sous-fichier data
imageFile = FILEPATH('elev_t.jpg', $
  SUBDIRECTORY = ['examples', 'data'])

; lirel'image
READ_JPEG, imageFile, image


; improrter l' objet 'elevbin.dat' qui se trouve dans sous fichier-data
demFile = FILEPATH('elevbin.dat', $
  SUBDIRECTORY = ['examples', 'data'])

; lire le fichier binaire demfile
dem = READ_BINARY(demFile, DATA_DIMS = [64, 64])
;réechantillonner la surface
dem = CONGRID(dem, 128, 128, /INTERP)
DEVICE, DECOMPOSED = 0, RETAIN = 2

; afficher la surface
WINDOW, 0, TITLE = 'Elevation Data'
SHADE_SURF, dem

; initialiser la fenêtre
oModel = OBJ_NEW('IDLgrModel')
oView = OBJ_NEW('IDLgrView')
oWindow = OBJ_NEW('IDLgrWindow', RETAIN = 2, $
COLOR_MODEL = 0)
oSurface = OBJ_NEW('IDLgrSurface', dem, STYLE = 2)
```

```
oImage = OBJ_NEW('IDLgrImage', image, $
  INTERLEAVE = 0, /INTERPOLATE)

; positionner l'objet
oSurface ->GetProperty, XRANGE = xr, $
  YRANGE = yr, ZRANGE = zr
xs = NORM_COORD(xr)
xs[0] = xs[0] - 0.5
ys = NORM_COORD(yr)
ys[0] = ys[0] - 0.5
zs = NORM_COORD(zr)
zs[0] = zs[0] - 0.5
oSurface ->SetProperty, XCOORD_CONV = xs, $
YCOORD_CONV = ys, ZCOORD = zs

; mapper la texture
oSurface ->SetProperty, TEXTURE_MAP = oImage, $
COLOR = [255, 255, 255]

; ajouter l'object au  modèle ,et a modèle auview.
oModel -> Add, oSurface
oView -> Add, oModel

; faire tourner l'objet
oModel ->ROTATE, [1, 0, 0], -90
oModel ->ROTATE, [0, 1, 0], 30
oModel ->ROTATE, [1, 0, 0], 30
;afficher la surface texturé avec application de modèle lumineux
oWindow ->Draw, oView

;afficher la fenêtre avec la possibilité de manœuvrer l'objet
XOBJVIEW, oModel, /BLOCK, SCALE = 1

OBJ_DESTROY, [oView, oImage]
```

END

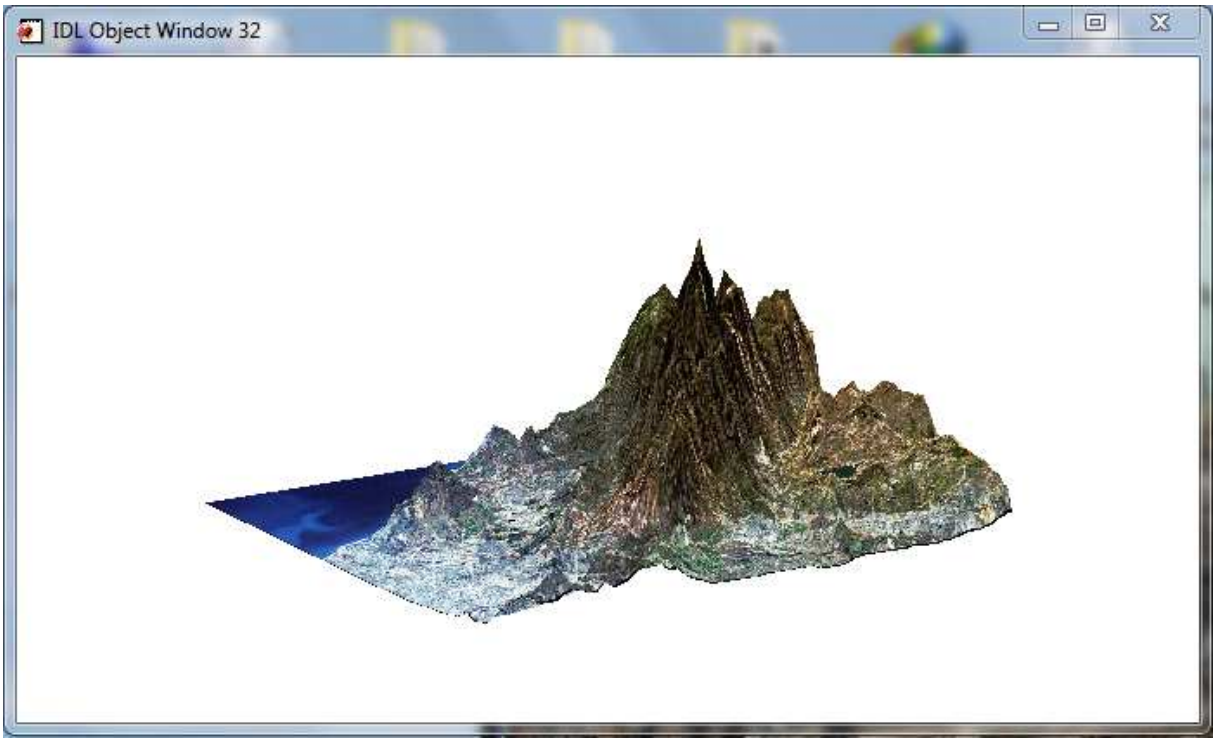


Figure (III.14) : plaquage d'une image satellitaire sur un MNT

Pour récapituler toutes les étapes ci-dessus on a proposée de simuler une image satellitaire de la planète terre sous IDL.

Le programme que nous avons conçu pour simuler une image satellitaire de la planète terre est donné comme suit :

```
FUNCTION MyRotator::Init, oModel
self.oModel = oModel
return, 1
END

PRO MyRotator::Cleanup
END
```

```
PRO MyRotator::OnTimer, oWin
self.oModel->Rotate, [0, 1, 0], 10
oWin->Draw
END

PRO MyRotator__define
void = { MyRotator, $
oModel: OBJ_NEW() $
}
END

PRO planète
; Importer une texture 'worldelv.dat' qui est stocké dans le sous fichier data.
file = FILEPATH('worldelv.dat', $
  SUBDIRECTORY = ['examples', 'data'])
image = READ_BINARY(file, DATA_DIMS = [360, 360])

; Création d'une sphère de rayon constant de valeur 0.5
MESH_OBJ, 4, vertices, polygons, $
REPLICATE (0.5, 101, 101)
oModel = OBJ_NEW('IDLgrModel')

;crée une palette de couleur
oPalette = OBJ_NEW('IDLgrPalette')
oPalette ->LoadCT, 33
oPalette ->SetRGB, 255,255,255,255
oImage = OBJ_NEW('IDLgrImage', image, $
  PALETTE = oPalette)

; coordonnées de la texture
vector = FINDGEN(101)/100.
texure_coordinates = FLTARR(2, 101, 101)
texure_coordinates[0, *, *] = vector # REPLICATE(1., 101)
```

```
texure_coordinates[1, *, *] = REPLICATE(1., 101) # vector

; Application de la texture a la surface
oPolygons = OBJ_NEW('IDLgrPolygon', SHADING = 1, $
   DATA = vertices, POLYGONS = polygons, $
   COLOR = [255,255,255], $
   TEXTURE_COORD = texure_coordinates, $
TEXTURE_MAP = oImage, /TEXTURE_INTERP)

 ; Ajouter le polygone
oModel -> ADD, oPolygons

;faire tourner l'objet
oModel -> ROTATE, [1, 0, 0], -90
oModel -> ROTATE, [0, 1, 0], -90

; afficher le résultat
oview= obj_new('idlgrview',color=[0,0,0])

;creerune source lumineuse
Olight0=OBJ_NEW('IDLgrLight',TYPE=1,location=[4,-1,3])
Olight0->SetProperty,COLOR=[255,255,255],INTENSITY=1.9
oLightModel=obj_new('idlgrmodel')
oLightModel->add,Olight0

Olight1=OBJ_NEW('IDLgrLight',TYPE=0,location=[0.5,1,-1])
Olight1->SetProperty,COLOR=[255,0,0],INTENSITY=1.65

oLightModel1=obj_new('idlgrmodel')
oLightModel1->add,Olight1

;Olight2=OBJ_NEW('IDLgrLight',TYPE=1,location=[0.5,1,-1])
;Olight2->SetProperty,COLOR=[25,30,50],INTENSITY=1.65
;oLightModel2=obj_new('idlgrmodel')
```

```
;oLightModel2->add,Olight2

;afficher l'objet sphérique avec l'exposition de lalumière
oView->add,oModel
oView->add,oLightModel
oView->add,oLightModel1
;oView->add,oLightModel2

;affichage de la sphére et donner le tittre
oWin = OBJ_NEW('IDLitWindow', DIMENSIONS=[650,650], $
TITLE="planete")
oWin->Add, oView

; afficher les résultats en utilisant XOBJVIEW (un utilitaire interactif qui nous permet de pivoter et de redimensionner
;XOBJVIEW, oView, /BLOCK

; nettoyage des references d'objet
;OBJ_DESTROY, [oModel, oImage, oPalette]


oRotator = OBJ_NEW('MyRotator', oModel)
oWin->AddWindowEventObserver, oRotator

 ; régler la vitesse de rotation de la sphére
oWin->SetTimerInterval, 0.9
oWin->SetEventMask, /TIMER_EVENTS
End
```

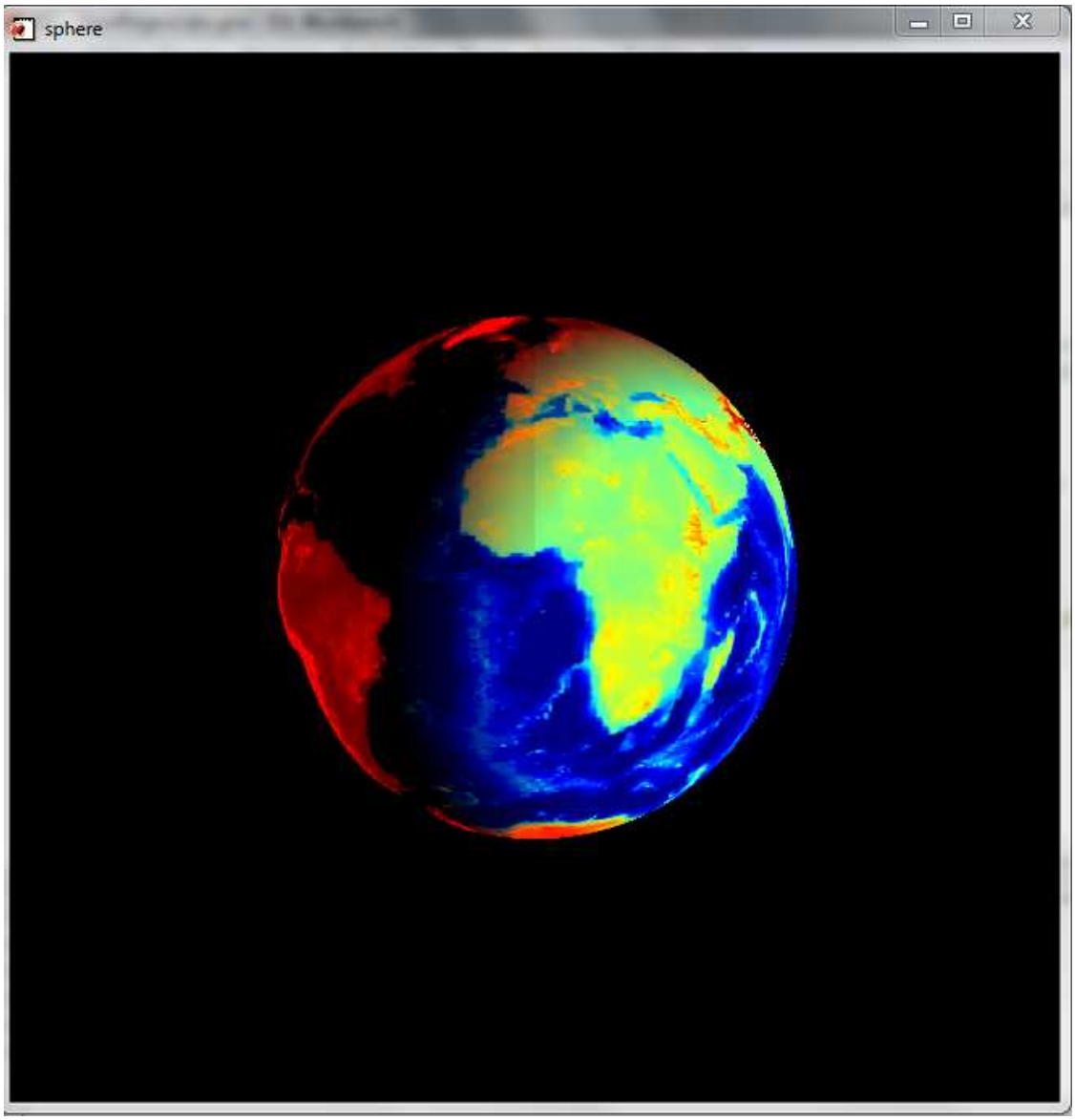


Figure (III.15) : visualisation de la planète

III.4.Discussion

Dans ce chapitre, nous avons présenté les étapes du plaquage de texture 2D sous IDL sur des différents objets 3D créés. Pour tester notre application, nous avons dans un premier temps réalisé sur des exemples tests. Dans un second temps, nous avons plaqué une image satellite sur le MNT.

Conclusion générale

Nous nous sommes intéressés dans le cadre de ce projet au plaquage de texture et ce, dans le but d'améliorer les rendu 3D et plus spécifiquement les modèles numériques de terrain (MNT).

Nous avons choisi de travailler avec le langage IDL (Interactive DataLangage)puisqu' il est associé au logiciel ENVI dédié à l'analyse des images satellitaires. Nous avons, donc, dans un premier temps, mené une étude bibliographique visant à mieux appréhender le concept de plaquage de texture et toutes les notions y afférant. Nous avons, dans un second temps, exploré les fonctionnalités de l'IDL qui nous ont permis de réaliser du plaquage de texture.

Dans ce travail, nous avons réalisé le plaquage d'un damier sur des objets à trois dimensions à savoir la sphère, le cylindre, la gaussienne. Ensuite, nous avons effectué le plaquage d'une image satellite sur une sphère en tenant compte des sources de lumières et d'ombrage.

En perspective, il est très important d'utiliser le plaquage de texture pour améliorer la qualité et ajouté du réalisme aux jeux vidéo et autre application.

Références bibliographiques

[1] :**S. Lefevere**.modèles d'habillage de surface pour la synthèse d'image. Thèse de doctorat Université Joseph Fourier de Grenoble (UJF).

[2] : **C. Dumas**. Un modèle d'interaction 3D : Interaction homme machine et homme-machine-homme dans les interfaces 3D pour le TCAO synchrone. Thèse de doctorat en Informatique. Université des Sciences de technologies de lille.

[3] : **http://www.jeuxvideopc.com**/point-textures-article-678408-1.html#goto

[4] : **J-Pierre Couwenbergh**. La Synthèse d'image. Editeur Marabout, France, 1998. Perception. Inrialpes.fr/people/Boyer/Teaching/Master/c7.pdf

[5] : **ABDEL MOUMÉNE ZERARI**, Mémoire en vue de l'obtention du diplôme de Magister en Informatique, VOLUME D'OMBRE EN RENDUTEMPS RÉEL, 2011.

[6] :**V. Boye**. Le Rendu Réaliste. Tutorial

[7]: **C. Heulin**. Tutorial 3D. www.123people .fr/s/Christophe+heulin.

[8] :**R. Raffin S. Thom, G. Gesquiére**. Visualisation 3D de feux de forets sur des modèles numérique de terrain de l'ING. Dans rencontre GEO-RISQUE 2006" La cartographie des risques naturels", Montpellier(France), Février 2006

[9] : **Yassine ARIBI**, *Doctorant au Laboratoire,* IDL : Solution informatique multiplateforme

[10] :**Guy Lebègue,** « Du Spatial aux Travaux publics : Les Maquettes virtuelles », avec la collaboration de Éric Lebègue, CSTB et Laurent Lebègue, CNES, Lettre AAAF Cannes, spécial mars 2007, publiée sur archive-host.com, reprise dans La Lettre AAAF, n°6 de juin 2007, (ISSN 1767-0675).

[11] : Support de cours M1 SIIG3T - Traitement d'images - J-P Cherel 2010, Création d'un Modèle Numérique de Terrain sous TNTmips.

[12]:**A. Watt et M. Watt.** Advanced Animation and renderingTechnique Theory and Practice.1992 ACM Press.