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

Mémoire de fin d'études

pour l'obtention du diplôme de Licence en Informatique

Thème

Système de visualisation 3D AVEC OpenGL pour la navigation dans un environnement fermé (cas d'un labyrinthe)

Réalisé par :

Mr MOUALEK Djalloul Youcef

Présenté le 27. mai 2015 devant la commission d'examination composée de MM.

Mr BENZIAN Yaghmoracène	*(encadreur)*
Mr KHELASSI Abdeldjalil	*(Examinateur)*
Mr HADJILA Fethallah	*(Examinateur)*

Année universitaire : 2014-2015

REMERCIEMENTS ET DEDICACES:

Je dedicace ce modeste travail à toutes les personnes qui m'ont soutenu jusqu'ici de prés ou de loin
Je remercie aussi Mr BENZIAN *Yaghmoracène pour ses précieux conseils tout au long de ce travail*

TABLE DES MATIERES

Introduction generale

OpenGL est un API très performant, multiplateforme, ouvert et libre d'utilisation. Grâce à ses qualités, OpenGL a réussi à se vendre dans plusieurs domaines.

Beaucoup d'applications libres - scientifiques, artistiques, ou non - sont développées pour des systèmes d'exploitations libres (comme Linux). OpenGL est une très bonne bibliothèque graphique pour ces logiciels graphiques.

Il est toujours possible d'utiliser un même programme, avec OpenGL, sur plusieurs systèmes d'exploitations différents (dont Windows), si le langage de programmation utilisé permet le cross-plateform (comme le C) et si l'application possède les fichiers natifs (dll pour Windows, so pour Linux, etc...) nécessaires à son exécution

Les labyrinthes que j'ai crée sont composés de murs rectangulaires et afin de coller le plus possible à la realité ces murs sont texturés

Ce mémoire est composé de 3 chapitres:

Premierement nous allons parler de l'infographie d'une façon generale, nous étudierons les transformations géométriques , Deuxiemement nous verrons toutes les techniques dont nous avons besoin pour faire un rendu realiste, et pour terminer le 3eme chapitre sera consacré à l'implementation du systeme

CHAPITRE 1 : infographie

1) Introduction

On appelle infographie le domaine de l'informatique concernant la création et la manipulation des images numériques. L'infographie regroupe de nombreux savoirs, parmi lesquels la représentation des éléments graphiques (texte, image ou vidéo), ainsi que leurs transformations (rotation, translation, zoom, ...) par l'intermédiaire d'algorithmes.

L'image s'affiche sur un écran , il s'agit d'un périphérique de sortie permettant de fournir une représentation visuelle. Ces informations proviennent de l'ordinateur, mais de façon "indirecte". En effet le processeur n'envoie pas directement les informations au moniteur, mais traite les informations provenant de sa RAM, puis les envoie à une carte graphique qui est chargée de convertir les informations en impulsions électriques qu'elle envoie au moniteur.

2) Les différents secteurs d'application

L'infographie peut être utilisée dans de nombreux domaines, tels que le cinéma, la presse, l'imprimerie, l'architecture... L'infographiste est en mesure de mener à bien toutes sortes de projets de communication, Et cela dans tous les domaines pratiquement.

Le secteur du cinéma fait souvent appel au savoir et au talent des infographistes,ils utilisent souvent la technologie 3D, et les images de synthèse, car Elles permettent d'obtenir des espaces, des images, se rapprochant au maximum de la réalité.

Dans le domaine du jeu vidéo Ils sont également très friands de la technologie 3D, car elle permet une fois De plus, de donner une dimension « réelle » aux personnages, ou aux espaces qui en font Partie.

De plus en plus de sites internet optent également pour l'infographie, surtout celle en 3D, Qui permet d'obtenir des présentations animées époustouflantes.

L'architecte utilise les images de synthèse, car elles donnent aux plans et aux projets de Construction une dimension photo réaliste. L'effet est bluffant et permet à la clientèle de visionner en avant première son projet de Construction sur écran. Ce sont des « maquettes », crées sur écran, qui collent très fortement à la réalité, et qui offrent aux clients une qualité d'image haute définition.

En réalité, les domaines d'application de l'infographie sont très variés et très différents. Que ce soit pour le cinéma, la presse, le monde du multimédia, ou les jeux vidéo, l'infographiste est amené a être polyvalent, et très créatif. Il lui faut, en

plus des compétences techniques, des compétences artistiques, et créatives. Un bon infographiste, réunissant ces critères, est en mesure de travailler dans n'importe quel domaine.

3) Les transformations

Les transformations géométriques permettent d'associer à toute figure initiale, une figure image (figure finale).

A) Les matrices

Une matrice est un tableau de nombres ordonnés en lignes et en colonnes entourés par des parenthèses. Sa syntaxe est semblable à celle d'un vecteur mais avec plus de nombres :

$$A = (a_{i,j}) = \begin{pmatrix} a_{1,1} & 0 & \cdots & \cdots & 0 \\ a_{2,1} & a_{2,2} & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ \vdots & & & \ddots & 0 \\ a_{n,1} & a_{n,2} & \cdots & \cdots & a_{n,n} \end{pmatrix}$$

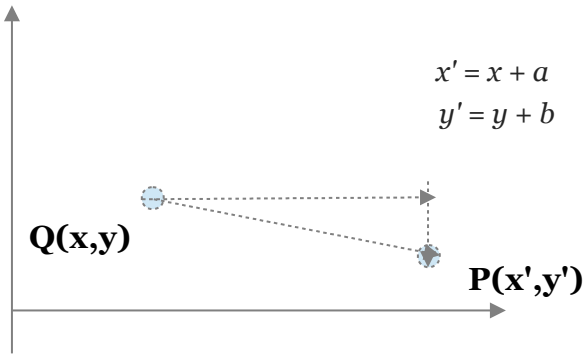
C'est une sorte de super-vecteur qui permet de faire pas mal de choses intéressantes. Un vecteur est en général utilisé en 3D pour gérer les points, les directions, les normales... Les matrices permettent de faire bien plus que ça. Elles servent principalement à convertir des données géométriques en données numériques car c'est plus facile de travailler avec des nombres

Rappel sur quelque transformations géométriques en 2D et en 3D

B) opération en 2D :

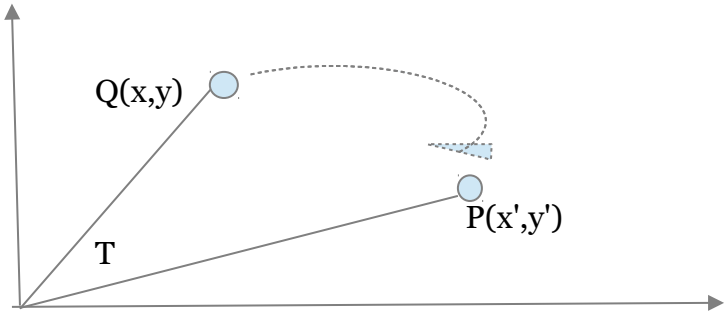
i. **translation** :

consiste à faire "glisser" un ensemble de points sur un "rail" (un vecteur).



ii. rotation :

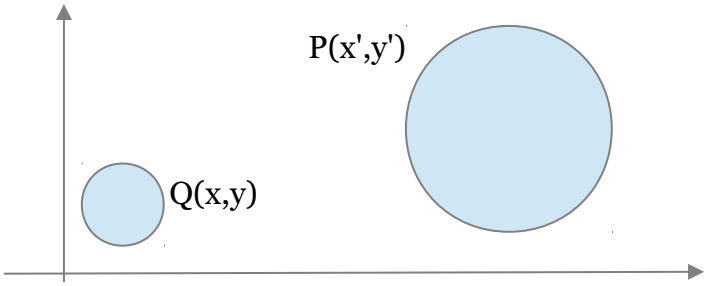
consiste à faire pivoter un ensemble de points d'un angle Thêta par rapport à un point.



$$P = \begin{pmatrix} \cos(T) & -\sin(T) \\ \sin(T) & \cos(T) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

iii. homothétie

consiste simplement à agrandir ou à réduire une forme géométrique.



$$P = \begin{pmatrix} S_X & 0 \\ 0 & S_Y \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

C) opération en 3D :

i. translation

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

ii. rotation

- autour de l'axe X

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(T) & -\sin(T) & 0 \\ 0 & \sin(T) & \cos(T) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

- autour de l'axe Y

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos(T) & 0 & \sin(T) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(T) & 0 & \cos(T) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

- autour de l'axe Z

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos(T) & -\sin(T) & 0 & 0 \\ \sin(T) & \cos(T) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

iii. homothétie

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

4)Les projections

A) Introduction

En général, une projection transforme les points dans une dimension N vers des point dans une dimension inférieur à N par exemple N=3 vers N=2. C'est grâce à cela que nous allons pourvoir "transformer" un monde 3D en un monde 2D (jusqu'à nouvel ordre, un écran ne dispose que de deux dimensions).

On effectue une "projection" car on réalise une transformation ayant pour but de projeter la silhouette d'un objet sur un écran virtuel placé entre l'objet et l'observateur (appelé aussi centre de projection).

B) Projection planaire

Projection projetant la texture sur un plan. Destiné aux surfaces planes, sinon des distorsions de l'image peuvent apparaître.

Principaux types de projections planaires:

i. **Les projections parallèles**: le centre de projection est à l'infini

- ce type de projection préserve les lignes, les distances, les angles
- cependant, on perd toute information de profondeur. L'œil humain ne voit pas comme ça !

ii. **Les projections perspectives :** Le centre de projection est à une distance finie des objets

- un bon modèle de l'œil humain ou d'un appareil photo
- respecte l'effet de perspective : plus un objet est loin plus il est petit
- préserve les lignes mais pas les angles ni les distances

5) Modélisation d'objets 3D :

A) Introduction

La synthèse d'images tridimensionnelles souvent abrégée 3D (3D pour trois

dimensions : x,y,z, les trois axes qui constituent le repère orthonormé de la géométrie dans l'espace) est un ensemble de techniques notamment issues de la Conception assistée par ordinateur qui permet la représentation d'objets en perspective sur un moniteur d'ordinateur. Elle est actuellement très utilisée en art numérique dans l'industrie du film et dans beaucoup de jeux vidéo.

B) Énumération Spatial :

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

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

Les techniques qui existent pour l'énumération spatiale sont trois techniques :

i. Quadtree :

Un **quadtree** est une structure de données de type arbre dans laquelle chaque nœud a exactement zéro ou quatre fils. L'idée de base est la suivante : si une image associée a un nœud n'est pas simple, on la découpe en quatre morceaux de même taille et les fils
du nœud correspondant sont les quadtrees associés aux quatre sous-images. Si au contraire l'image est simple, alors elle est caractérisée par une information unique, sa couleur ; dans ce cas, le nœud porte cette information et n'a pas de fils.

Par exemple, si on parcourt les quatre quarts d'une image en décrivant un Z, et si on suppose
que les dix régions en lesquelles on a découpe le carre de la figure 1 sont simples, de couleurs a; b; :::, alors cette figure peut être représentée par le quadtree de la figure 2.

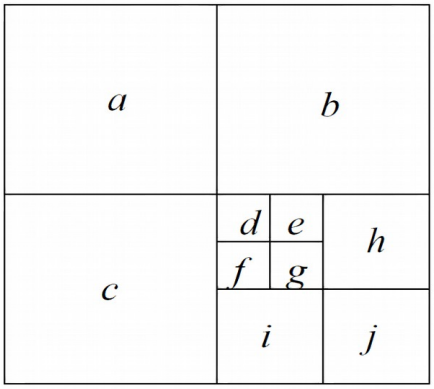


Figure 1

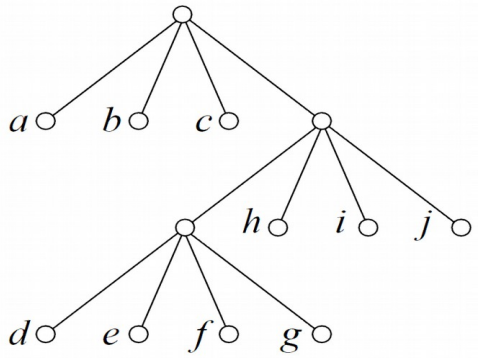


Figure 2

ii. **Octree :**

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

iii. Maillage :

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

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

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

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

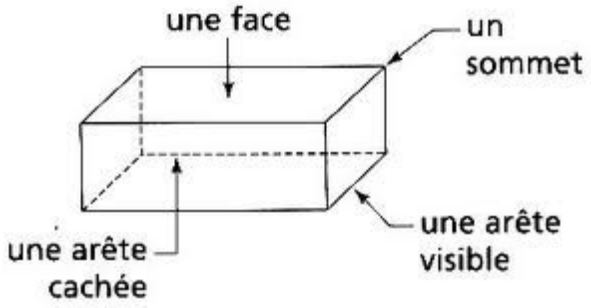


Figure 1.3

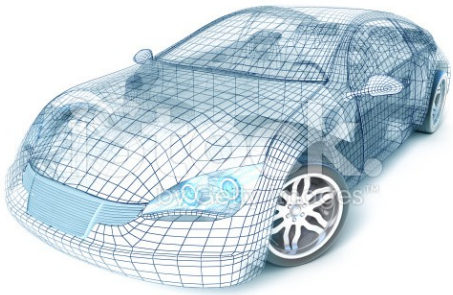


figure 1.4

C) Géométrie Solide Constructive (les arbres CSG)

En infographie, la géométrie de construction de solides (CSG en anglais: Constructive Solid Geometry) est une branche de la modélisation de solide (ou modélisation 3D). Cette géométrie concerne la représentation d'objet solide comme une combinaison d'objets solides simples (cylindre, sphère, etc.).

On applique des opérations booléennes et des transformations géométriques.

La grammaire peut être décrit par :

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

Note : les objets ci dessous sont texturés

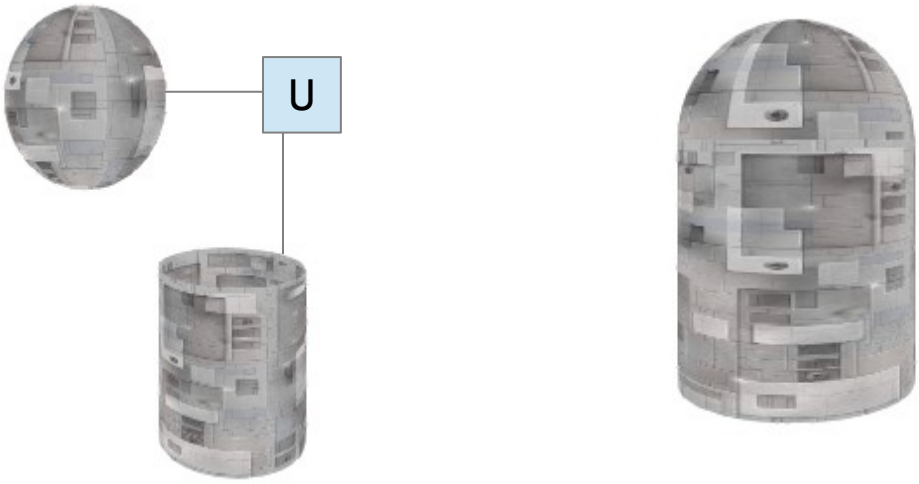


figure 1.5

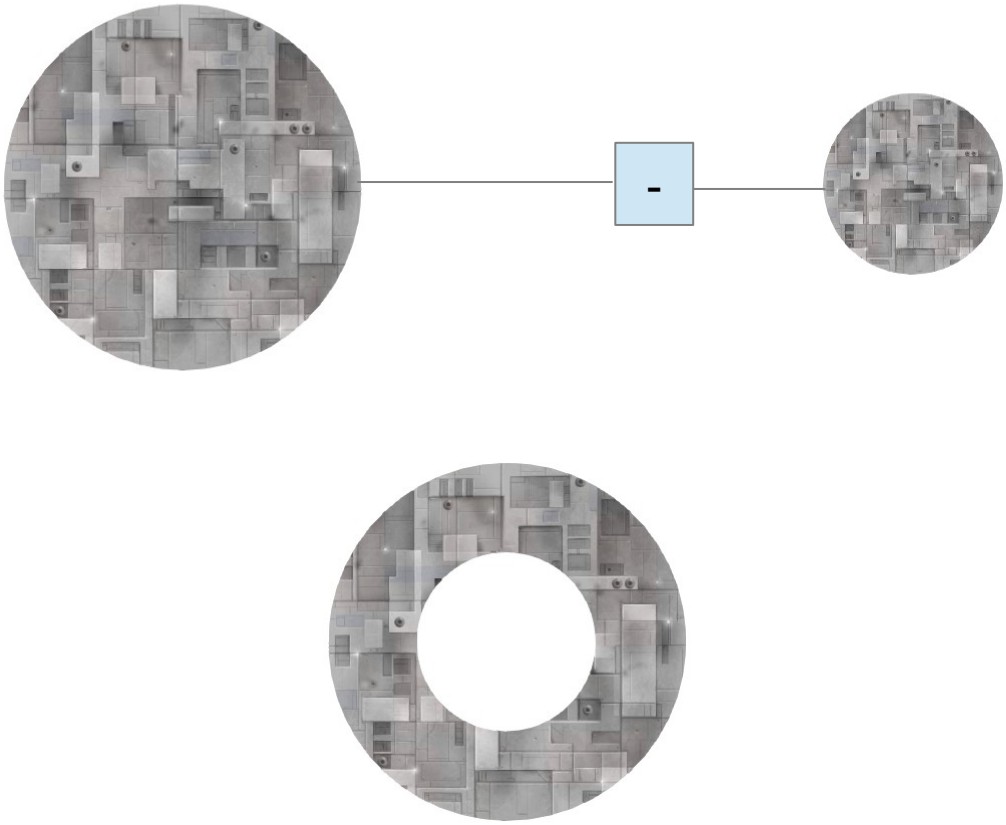


figure 1.6

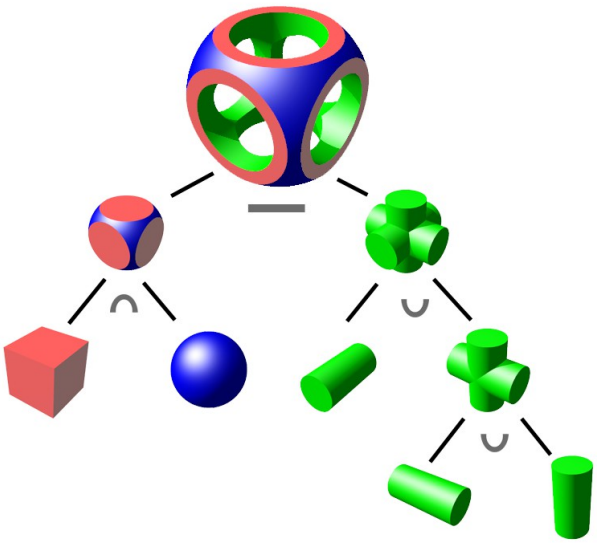


figure 1.7

D) Courbes et surfaces polynomiales (Courbes et surface de Bézier)

i. Courbes de Bézier :

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

Quatre points P0, P1, P2 et P3 définissent une courbe de Bézier cubique. La courbe se trace en partant du point P0, en se dirigeant vers P1 et en arrivant au point P3selon la direction P2-P3. En général, la courbe ne passe ni par P1 ni par P2 : ces points sont simplement là pour donner une in8formation de direction.

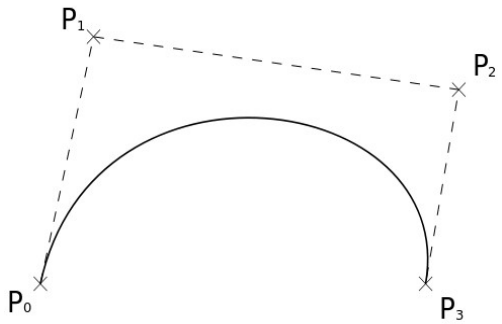


Figure 1.8

$$P(t) = t^3.P_3 + 3.t.(1-t)^2.P_1 + 3\,t^2(1-t).P_2 + (1-t)^3 P_0$$

ii. Surfaces de Bézier :

Les surfaces de Bézier sont une méthode de définition d'une surface grâce aux courbes de Bézier.Ce qu'on remarque immédiatement c'est qu'à $t=t_0$ constant on a simplement la définition d'une courbe de Bézier

$$P(s,t) = \Sigma\Sigma Bi,n(s).Bj,m(t).Pi$$

j, s et t évoluant sur R. (On se restreindra à s et t variant sur l'intervalle [0,1]) Les Pi,définissent ce que l'on appelle une grille de contrôle (on les représente souvent sous forme d'une grille). L'allure locale de la surface est toujours déterminée par un point de contrôle.

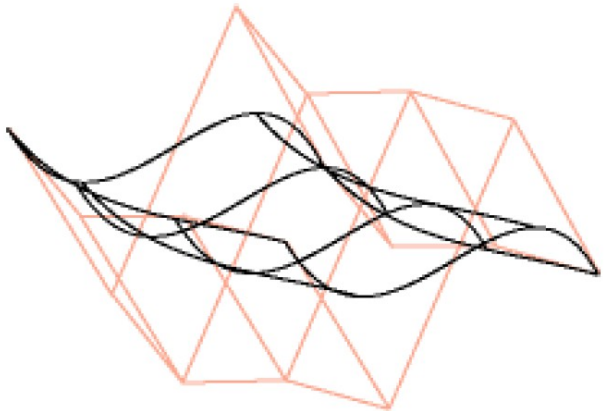


Figure 1.9

6)Conclusion

Dans ce chapitre nous avons introduit le thème de l'infographie et le rôle qu'occupe un infographiste dans le secteur du divertissement et du travaille.

Nous avons abordé les transformations géométriques et la représentation des objets 3D

Pour les transformations géométriques on a vu les matrices de transformations (translation, rotation, homothétie) ainsi que la projection planaire avec ces deux types (parallèle, perspective)

Pour la représentation des objets 3D on a vu deux représentations :les arbres CSG, l'Énumération Spatial (Quadtree, Octree, maillage).

CHAPITRE 2 : RENDU REALISTE

1) Texture (exemple avec OpenGL)

A) Introduction

Une texture est simplement une sorte de papier peint que l'on va "coller" sur les modèles (2D ou 3D). Dans un jeu vidéo, toutes les surfaces que vous voyez (sols, herbe, murs, personnages, ...) sont constituées de simples images collées sur des formes géométriques. Pour un mur par exemple, il suffit de créer une surface carrée puis de coller une image représentant des briques.

Figure 2.1
un simple rectangle de couleur verte qui se transforme en un mur grâce à la texture

B) Plaquage de texture

Chaque point du rectangle correspond à un point de la texture. Donc pour réaliser le plaquage de la texture sur la face il faut faire correspondre à chaque vertex (sommet) composant notre face une coordonnée sur la texture

Pour définir les coordonnées du vertex nous utiliserons glVertex3d(x,y,z), et pour définir les coordonnées de texture nous utiliserons :

glTexCoord2d (u,v);
L'espace de coordonnées sur la texture est en 2D, le coin en bas à droite a les coordonnées (0,0) et le coin en haut à gauche est en (1,1), quelque soit la taille en pixels de l'image.

```
glBegin(GL_QUADS);
glTexCoord2d(0,1);  glVertex3d(0,1,1);
glTexCoord2d(0,0);  glVertex3d(0,1,-1);
glTexCoord2d(1,0);  glVertex3d(3,1,-1);
glTexCoord2d(1,1);  glVertex3d(3,1,1);
glEnd();
```

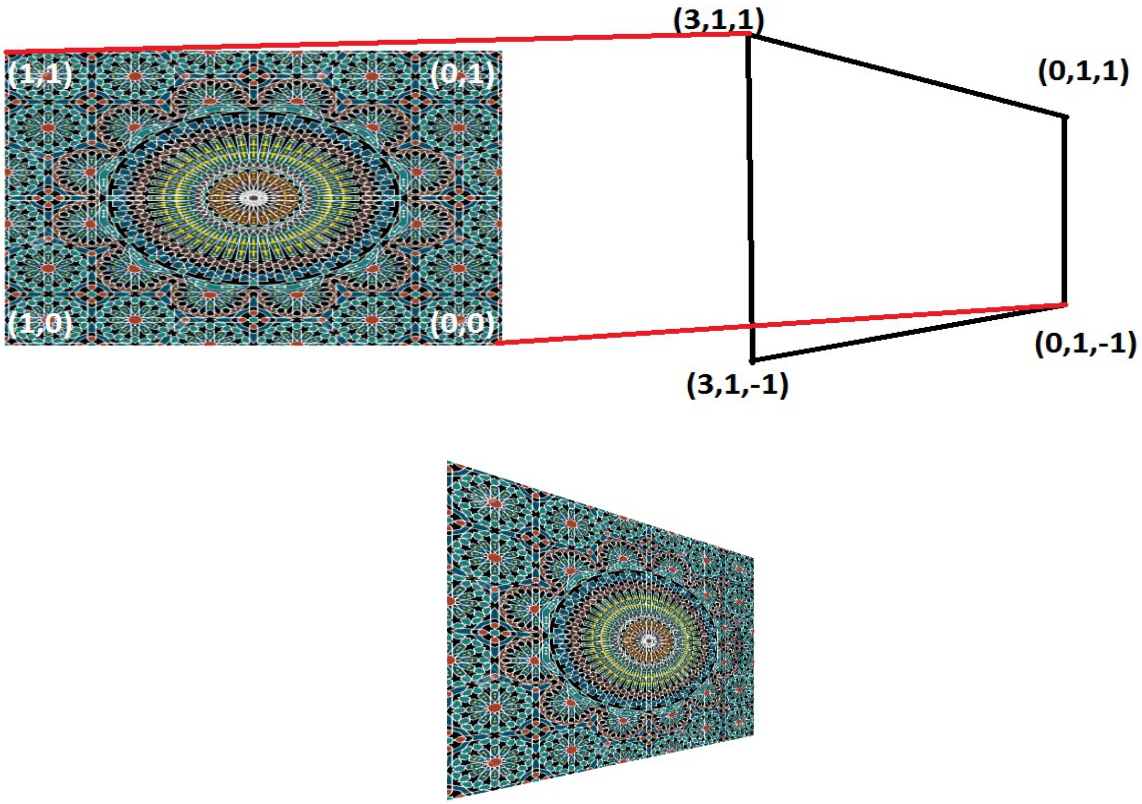
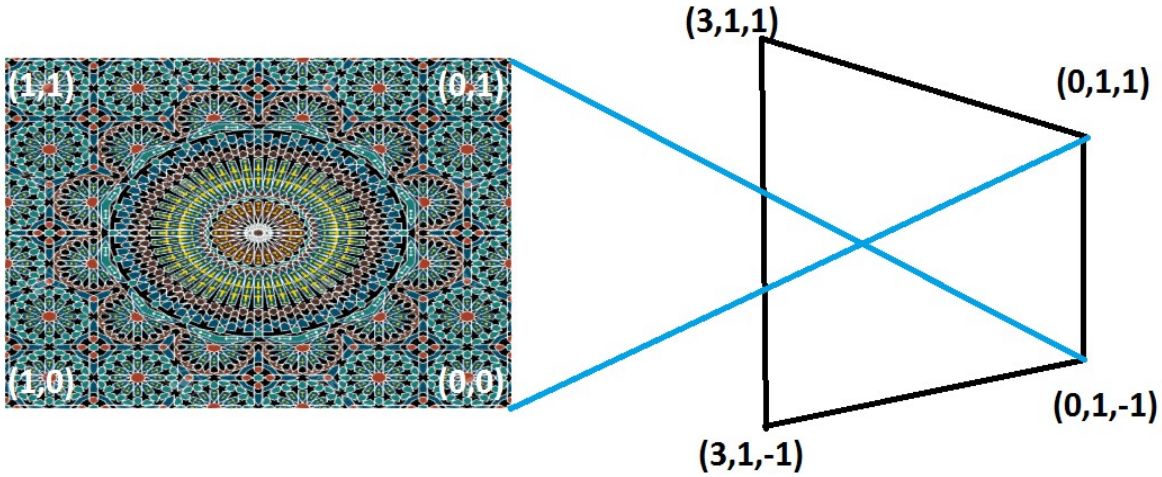


figure 2.2

C)Erreur fréquente :

une mauvaise correspondance entre les coordonnées d'un sommet et celles d'une texture provoque une déformation du rendu



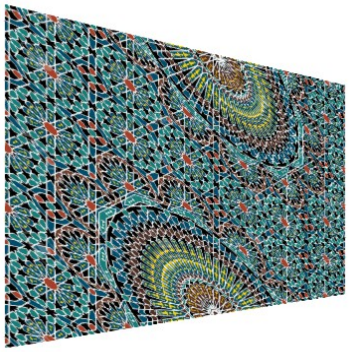


figure 2.3

D) Cas d'un triangle :

Dans les jeux vidéos, la primitive de base la plus utilisée est le triangle. Le plaquage de texture fonctionne de la même manière en ne choisissant que3 points sur la texture, points qui ne sont donc pas forcément des coins de l'image.

```
glBindTexture(GL_TEXTURE_2D, texture2);
   glBegin(GL_TRIANGLES);
   glTexCoord2d(0,0);      glVertex3d(1,1,-1);
   glTexCoord2d(1,0);      glVertex3d(-1,1,-1);
   glTexCoord2d(0.5,1);    glVertex3d(0,0,1);
   glEnd();
```

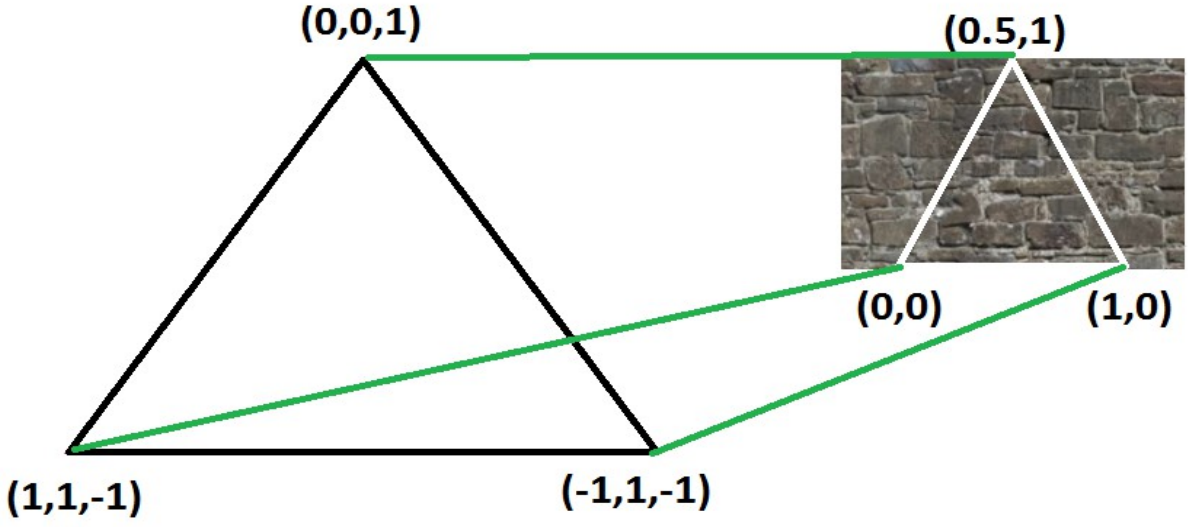


figure 2.4

E)Répétition des textures

Jusqu'à présent tout ce qu'on a fait c'est appliquer soit la texture soit une partie. Mais si on essayer de créer un sol avec la même techniques le résultat ne sera pas très convaincant

```
glBegin(GL_QUADS);
   glTexCoord2d(0,1);  glVertex3d(50,-50,-1);
   glTexCoord2d(1,1);  glVertex3d(-50,-50,-1);
   glTexCoord2d(1,0);  glVertex3d(-50,50,-1);
   glTexCoord2d(0,0);  glVertex3d(50,50,-1);
   glEnd();
```

figure 2.5

comme on peut le voir cette texture ne peut pas être appliquer sur une très grande surface , donc il faut faire en sorte qu'elle se répète.

Dans les exemples précédant les paramètres u et v de la fonction glTexCoord2d était entre 0 et 1 mais en réalité ils peuvent être bien plus grands. Ici nous voulons que la texture se répète 15 fois en longueur et en largeur. Il suffit de prendre des coordonnées entre 0 et 15

```
glBegin(GL_QUADS);
glTexCoord2d(0,15);  glVertex3d(50,-50,-1);
   glTexCoord2d(15,15);  glVertex3d(-50,-50,-1);
   glTexCoord2d(15,0);  glVertex3d(-50,50,-1);
   glTexCoord2d(0,0);  glVertex3d(50,50,-1);
   glEnd();
```

Nous obtenons un bien meilleur résultat visuel :

figure 2.6

2) Éclairage

A) Le modèle d'éclairage

La perception de la couleur de la surface d'un objet du monde réel dépend de la distribution de l'énergie des photons qui partent de cette surface et qui arrivent aux cellules de la rétine de l'œil. Chaque objet réagit à la lumière en fonction des propriétés matérielles de sa surface.
Le modèle d'éclairage d'OpenGL considère qu'un objet peut émettre une lumière propre, renvoyer dans toutes les directions la lumière qu'il reçoit, ou réfléchir une partie de la lumière dans une direction particulière, comme un miroir ou une surface brillante.

pour coller le plus possible à la réalité différentes types de lumière ont été créés. OpenGL distingue quatre types de lumières :

i. Lumière ambiante

C'est la lumière qui a tellement été dispersée et renvoyée par l'environnement qu'il est impossible de déterminer la direction d'où elle émane. Elle semble venir de toutes les directions. Quand une lumière ambiante rencontre une surface, elle est renvoyée dans toutes les directions.

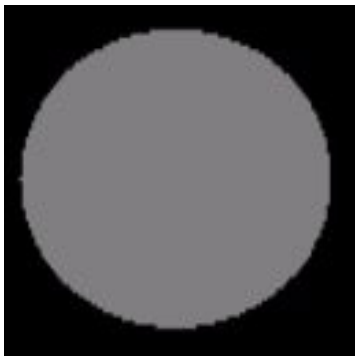


Figure 2.7

Une sphère 3D éclairé par une Lumière ambiante seulement; ressemble à de la 2D

ii. Lumière diffuse

C'est la lumière qui vient d'une direction particulière, et qui va être plus brillante si elle arrive perpendiculairement à la surface que si elle est rasante. Par contre, après avoir rencontré la surface, elle est renvoyée uniformément dans toutes les directions.

Figure 2.8

iii. Lumière spéculaire

La lumière spéculaire vient d'une direction particulière et est renvoyée par la surface dans une direction particulière. Par exemple un rayon laser réfléchi par un miroir.

Figure 2.9

Par exemple, la Lumière ambiante est la couleur de l'objet sans éclairage, la Lumière diffuse est la couleur de l'objet illuminée par une Lumière ambiante (lumière non directe) et la couleur spéculaire est la couleur donnée par un éclairage direct, par exemple des métaux ont en général une Lumière spéculaire blanche (tache blanche sur une sphère en métal par exemple). On peut également caractériser la lumière d'émission.

iv. Lumière émise

Les objets peuvent émettre une lumière propre, qui augmentera leur intensité, mais n'affectera pas les autres objets de la scène.

v. Brillance

Cette valeur entre 0.0 et 128.0 détermine la taille et l'intensité de la tâche de réflexion spéculaire. Plus la valeur est grande, et plus la taille est petite et l'intensité importante.

B) Les lampes

i. Couleur de lampes

la couleur d'une composante de lumière d'une lampe est définie par les pourcentages de couleur rouge, verte, bleue qu'elle émet.

ii. La lumière ponctuelle

Une source de lumière ponctuelle est une source dont les rayons partent tous du centre de la source.

iii. La lumière directionnelle

Une source de lumière directionnelle est une source dont les rayons ont la même direction en tout point de l'espace

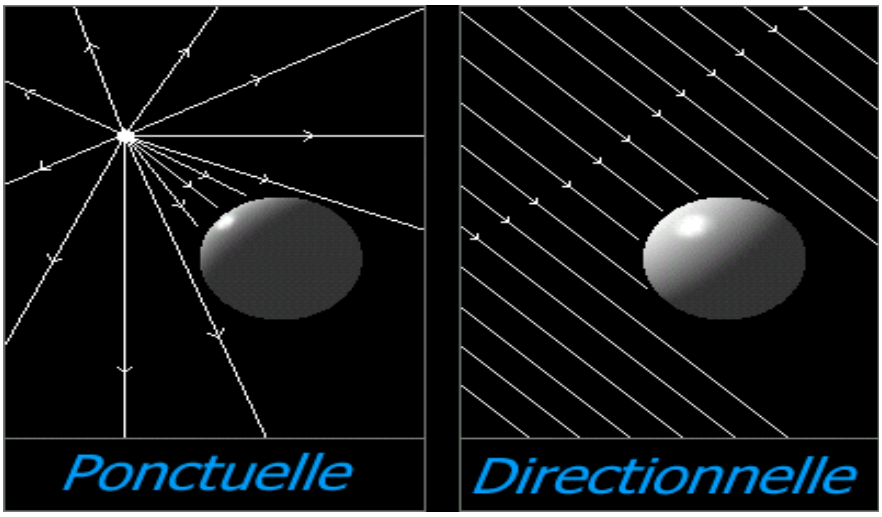


Figure 2.10

Les rayons d' une source ponctuelle qui serait située tellement loin de la scène pourraient être considérés comme parallèles. L'exemple parfait de lumière directionnelle dans notre monde, qui est d'ailleurs cité dans n'importe quel bouquin expliquant ce genre de choses, est bien sûr le soleil : tous ses rayons vont dans la même direction.

iv. Lampes omnidirectionnelles et spots

Par défaut, une lampe illumine l'espace dans toutes les directions.

L'autre type de lampe proposé par OpenGL est le spot.

Un spot est caractérisé, en plus de sa position, par sa direction, le demi angle du cône de lumière et l'atténuation angulaire de la lumière. Les spots sont utilisés dans le rendu des (phare de voitures, lampes de torches, projecteurs ...).

C) Le matériau :

Pour pouvoir gérer correctement les lumières il faut d'abord definir pour chaque objet ses caractéristiques de luminositée. Il faut donc définir de quelle facon le matériau de l'objet va interagir avec la lumière.

Par exemple, un objet rouge renvoie toute la lumière rouge qu'il reçoit et absorbe toute la lumière verte et bleue qu'il reçoit.

Si cet objet est éclairé par une lumière blanche (composé en quantités égales de rouge, vert et bleu), il ne renverra que la lumière rouge et apparaîtra donc rouge.

Si cet objet est éclairé par une lumière verte, il apparaîtra noir, puisqu'il absorbe le vert et n'a pas de lumière rouge à réfléchir.

ı. Propriétés matérielles d'un objet

Les quatre propriétés matérielles d'un objet sont celles qui ont été évoquées dans la partie Modèle d'éclairage c.à.d (Lumière ambiante, Lumière diffuse, Lumière spéculaire, Lumière émise)

3)Conclusion

Dans ce chapitre nous avons abordés les étapes pour faire un rendue réaliste en temps réel qui sont les textures et les les lumières.

Pour les textures on a vu les coordonnées de plaquage, comment appliquer une texture et l'erreur à ne pas faire . Pour les lumière on a vu les différentes types de lumière dans une scène 3D, les lampes ainsi que les matériaux.

CHAPITRE 3 : APPLICATION REALISTE

1) Introduction

L'apprentissage de la navigation est une étape incontournable au bon fonctionnement d'une application 3D. Deux fonctions sont nécessaires pour naviguer dans une scène virtuelle, la rotation et la translation.

Pendant la programmation j'ai utilisé l'IDE permettant de programmer en C et en C++ , je parle bien sur de Dev C++ avec l' API OpenGL

2)Choix du langage

J'ai préféré utiliser un langage dont on se sert pour concevoir des jeu vidéo et parmi ces langage il y' a le C

Le C est un très bons choix, car couramment utilisé dans le monde du jeu vidéo. De par ce fait, le langage dispose d'une multitude de bibliothèques permettant de réaliser des jeux. Malheureusement, ce n'est pas le langage le plus simple (même si ce point de vue est relatif) et celui-ci demande une grande rigueur.

3)Dev C++

Dev-C++ est un environnement de développement C++ complet et gratuit

J'ai choisi Dev C++ parce qu'il répond bien à mes exigences et offre :

- Un Compilateur C et C++ (Win32)
- Un Débugger intégré
- Une Colorisation de la syntaxe paramétrable
- Un Travail en mode source ou en mode projet
- Une Création de fichier Setup (pour une installation simplifiée)
- De Nombreuses bibliothèques intégrées.

4)OpenGL

OpenGL (**Open G**raphics **L**ibrary) est un ensemble normalisé de fonctions de calcul d'images 2D ou 3D lancé parSilicon Graphics en 19921. Cette interface de programmation est disponible sur de nombreuses plateformes où elle est utilisée pour des applications qui vont du jeu vidéo jusqu'à la CAO en passant par la modélisation.

OpenGL permet à un programme de déclarer la géométrie d'objets sous forme de points, de vecteurs, de polygones, de bitmaps et de textures. OpenGL effectue ensuite des calculs de projection en vue de déterminer l'image à l'écran, en tenant compte de la distance, de l'orientation, des ombres, de la transparence et du cadrage.

L'interface regroupe environ 250 fonctions différentes qui peuvent être utilisées pour afficher des scènes tridimensionnelles complexes à partir de simples primitives géométriques. Du fait de son ouverture, de sa souplesse d'utilisation et de sa disponibilité sur toutes les plates-formes, elle est utilisée par la majorité des applications scientifiques, industrielles ou artistiques 3D et certaines applications 2D vectorielles.

Cette bibliothèque est également utilisée dans l'industrie du jeu vidéo où elle est souvent en rivalité avec la bibliothèque de Microsoft : Direct3D. Une version nommée OpenGL ES a été conçue spécifiquement pour les applications embarquées (téléphones portables, agenda de poche, consoles de jeux...).

5) Bibliothèque utilisée

J'ai du utiliser Plusieurs bibliothèques dont certaines développées à partir d'OpenGL afin d'apporter des fonctionnalités qui ne sont pas disponibles dans la bibliothèque OpenGL elle-même :

- **<math.h>**

Pour calculer des fonctions mathématiques courantes (sinus , cosinus ...). C99 a ajouté de nombreuses fonctions mathématiques .

- **<stdio.h>**

Fournit les capacités centrales d'entrée/sortie du langage C

- **<GL/freeglut.h>**

Freeglut est une alternative open source à la bibliothèque OpenGL Utility Toolkit (GLUT) . GLUT (et donc freeglut) permet à l'utilisateur de créer et gérer des fenêtres contenant des contextes OpenGL sur une large gamme de plates-formes et également de lire les fonctions souris , clavier et joystick .

- **<gl\glaux.h>**

Pour lire les fichiers .BMP

6)Menu

Le menu est une tache qui permet d'exécuter une commande dans le but de changer le mode d'interaction ou l'état du système

Je dois faire un menu pour que l'utilisateur puisse passer du labyrinthe 1 au labyrinthe 2.facilement.et pour pouvoir quitter l'application

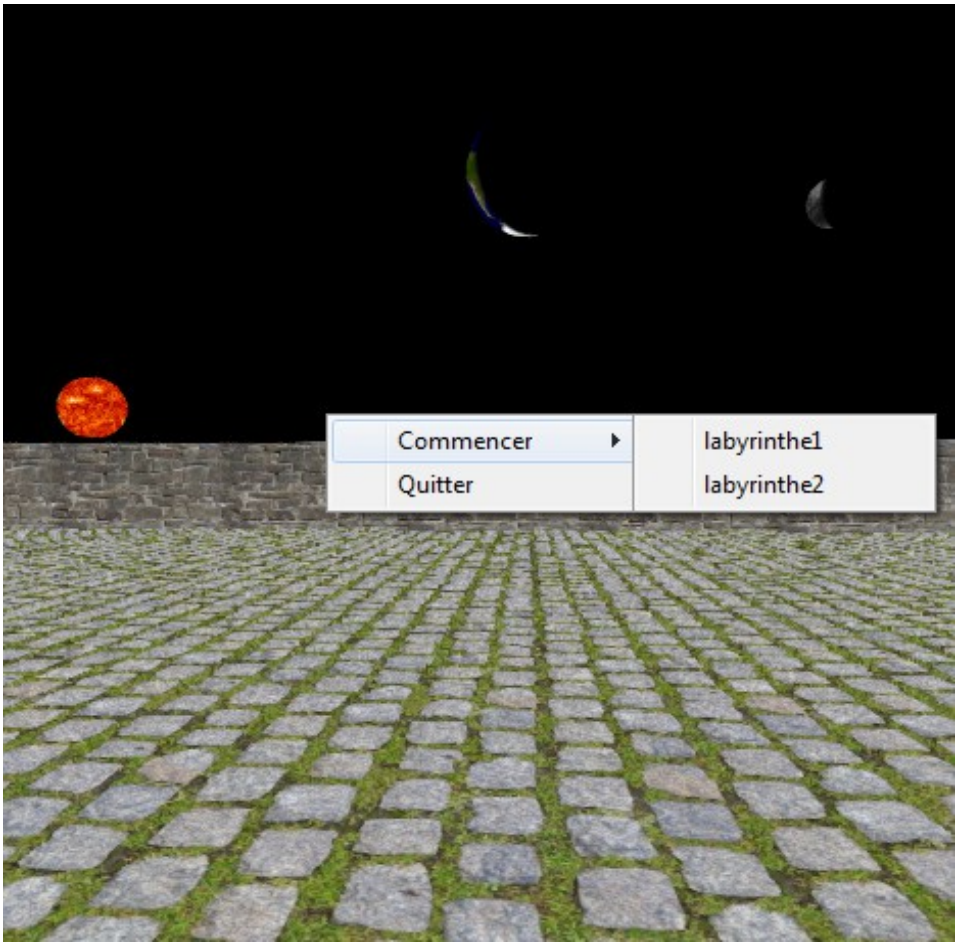


figure 3.1

menu avec l'option « commencer » qui permet de choisir entre le labyrinthe 1 ou le

labyrinthe 2 ainsi que l'option « quitter » qui permet de sortir de l'application.

7) Navigation dans un environnement

A) *Introduction*

En effet, nous avons déjà vu quelque notions essentielles *d'OpenGL*. Nous maîtrisons l'affichage de formes simples, l'affichage de textures, la gestion de l'éclairage, Il ne reste plus qu'une seule chose à voir pour compléter ces essentielles : **une caméra déplaçable**.

B) *Type de camera*

Nous utiliserons une caméra de déplacement particulière appelé caméra subjective.

En **caméra subjective**, la caméra est le sujet de l'action ; le point de vue de la caméra est alors celui d'un personnage, de telle sorte que le spectateur ait la sensation de partager la perception visuelle de celui-ci. Il participe à accentuer le processus d'identification au personnage de la part du spectateur.

C) *Les contrôles*

Pour pouvoir se déplacer dans notre monde 3D, nous devrons utiliser le clavier et la souris. Grâce à OpenGL nous allons pouvoir lier un événement à un déplacement, ces événements seront divisés en deux catégories. Nous utiliserons :

- La souris : pour orienter la caméra en fonction du mouvement effectué
- Le clavier : pour déplacer la caméra selon 4 axes (gauche, droite, avant et arriéré).

D) *La gestion de l'orientation*

La première chose que l'on va voir concerne l'orientation de la caméra. Les événements déclenchés par la souris permettront de l'orienter en fonction du mouvement effectué. Lorsque nous bougerons la souris vers la droite ou vers le gauche, la caméra s'orientera de la même façon :

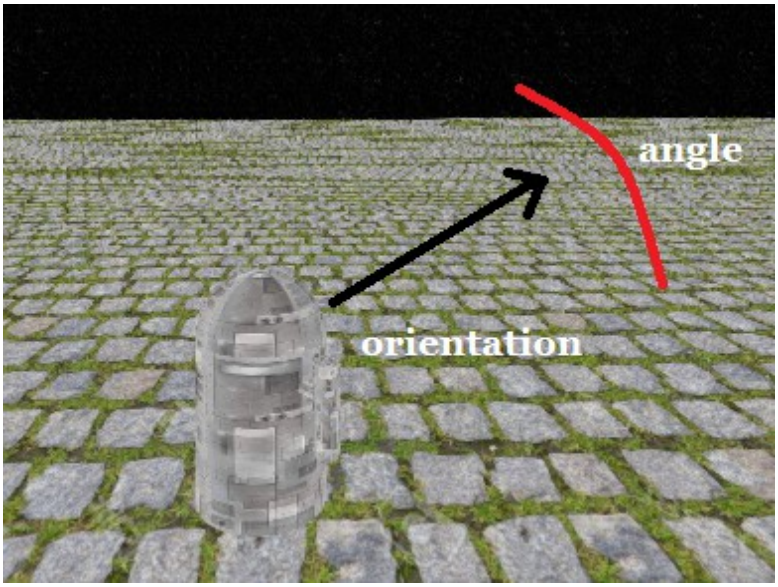


figure 3,2

E) *La gestion des déplacements*

- Le déplacement 'vertical' pour avancer avec le bouton 'z' et reculer avec 's'.

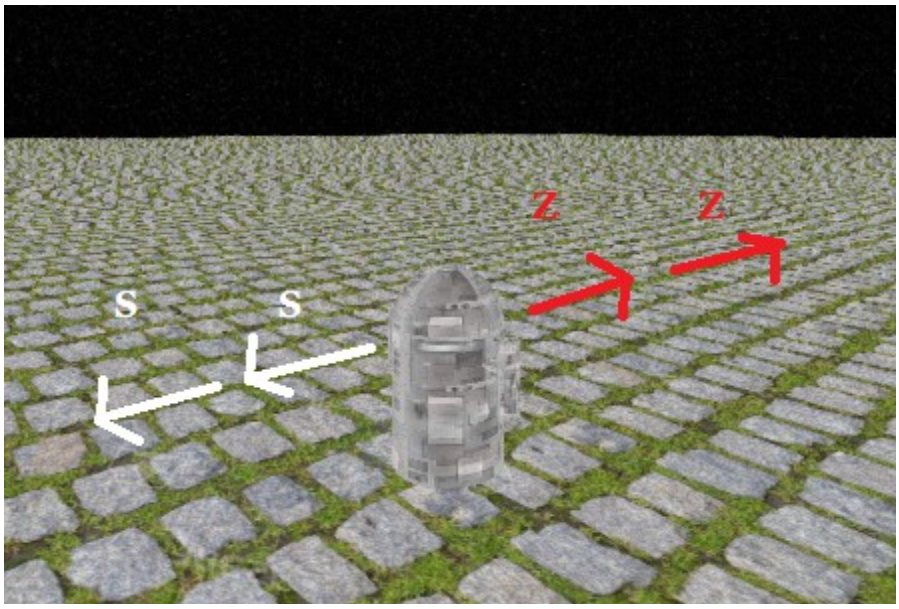


figure 3,3

- Le déplacement latéral pour aller vers la gauche avec 'q' ou la droite 'd'.

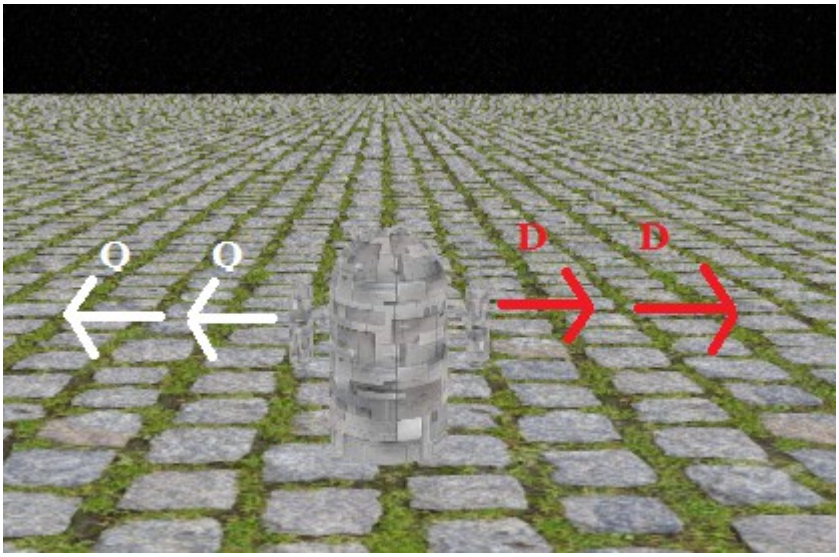


figure 3,4

8)Navigation dans un labyrinthe

Mon projet consiste à effectuer une navigation dans des labyrinthe 3D en utilisant OpenGL

Mon objectif était de définir des labyrinthes avec une entrée et une sortie qui se compose de murs rectangulaires texturés dont les descriptions (coordonnées des murs et des texture) se trouvant dans un fichier texte.

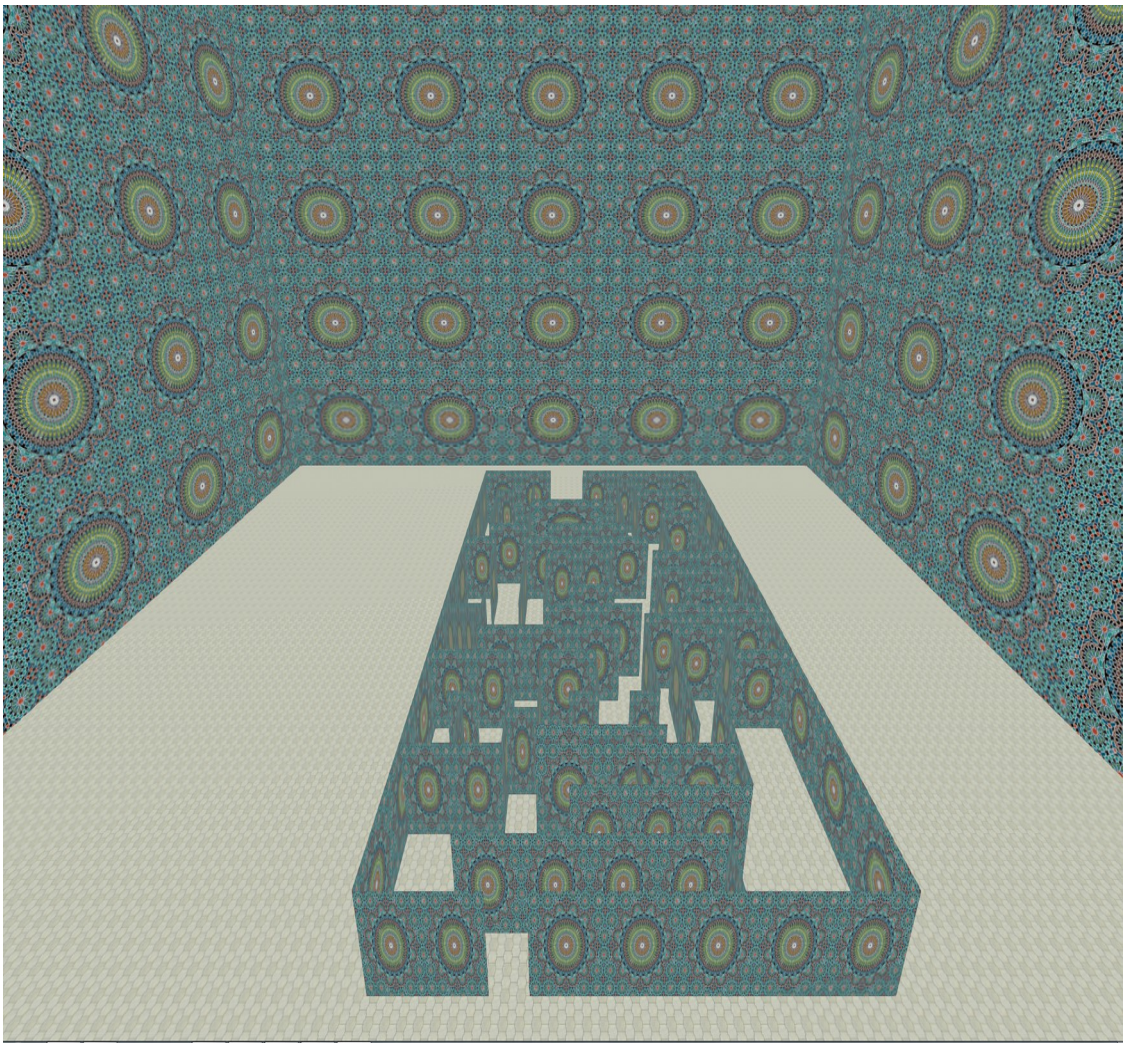


figure 3.5

Vue aérienne du 1er labyrinthe 1

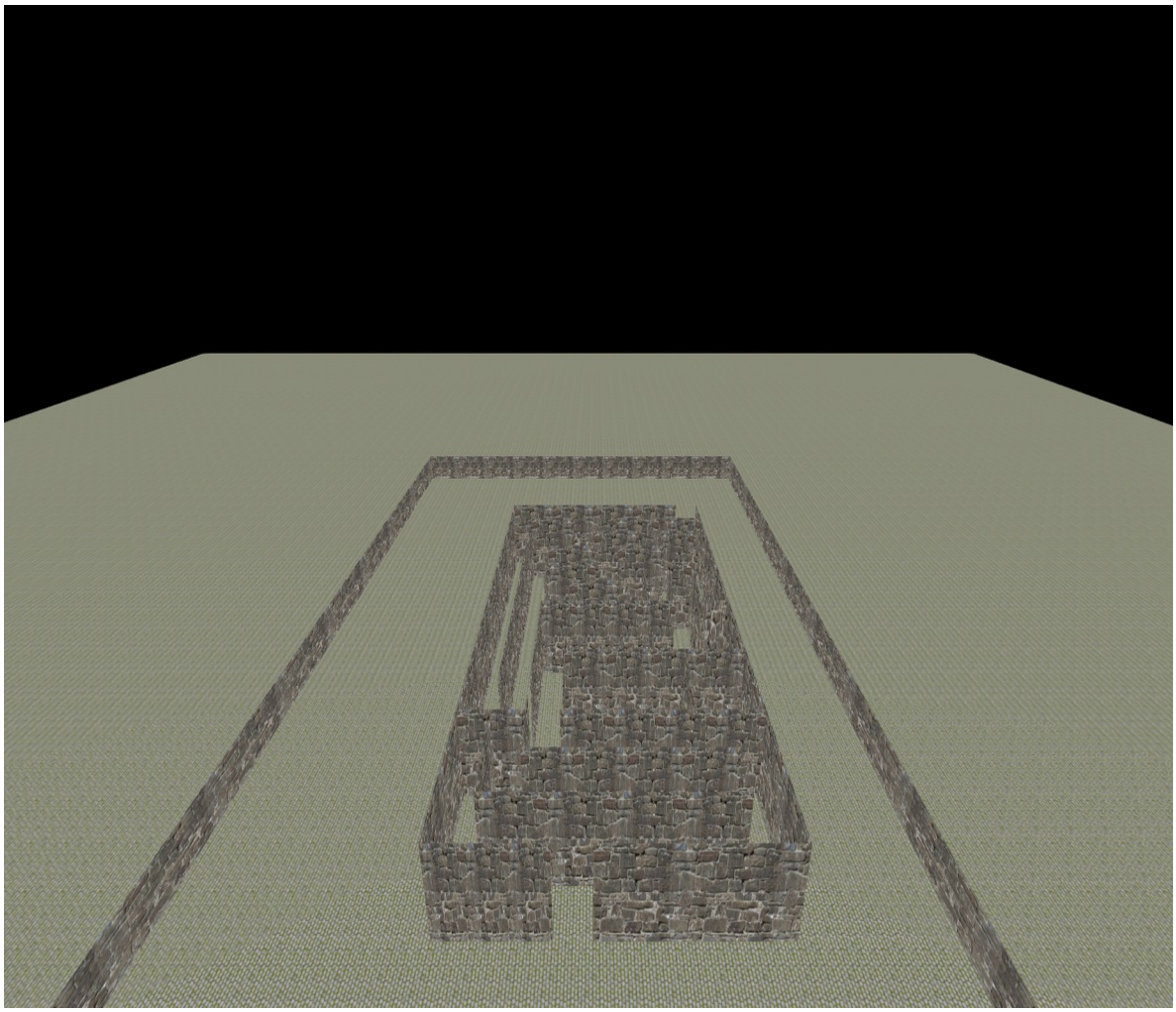


figure 3.6

Vue aérienne du 2eme labyrinthe 2

A) description de la scène

L'avantage de mettre la description d'une scène 3D dans un fichier texte et de le charger après c'est de pouvoir modifier les coordonnées aisément par l'utilisateur.

Pour pouvoir expliquer le fonctionnement voici une capture d'écran du fichier texte ou se trouve la description des quatre grands murs qui entoure le labyrinthe 2

Nous avons quatre murs, chaque mur est constitué de 4 sommets et chaque sommets possède 3 composantes (x, y, z) et deux composantes de texture (u, v).

Une ligne contient 5 nombres. Les 2 premiers nombres qui sont encadrés en vert sont les coordonnées (u,v) de la texture (on peut voir que j'ai répété la texture 10

fois sur u), et les 3 derniers qui sont encadrés en bleu sont les coordonnées (x, y ,z) des vertex

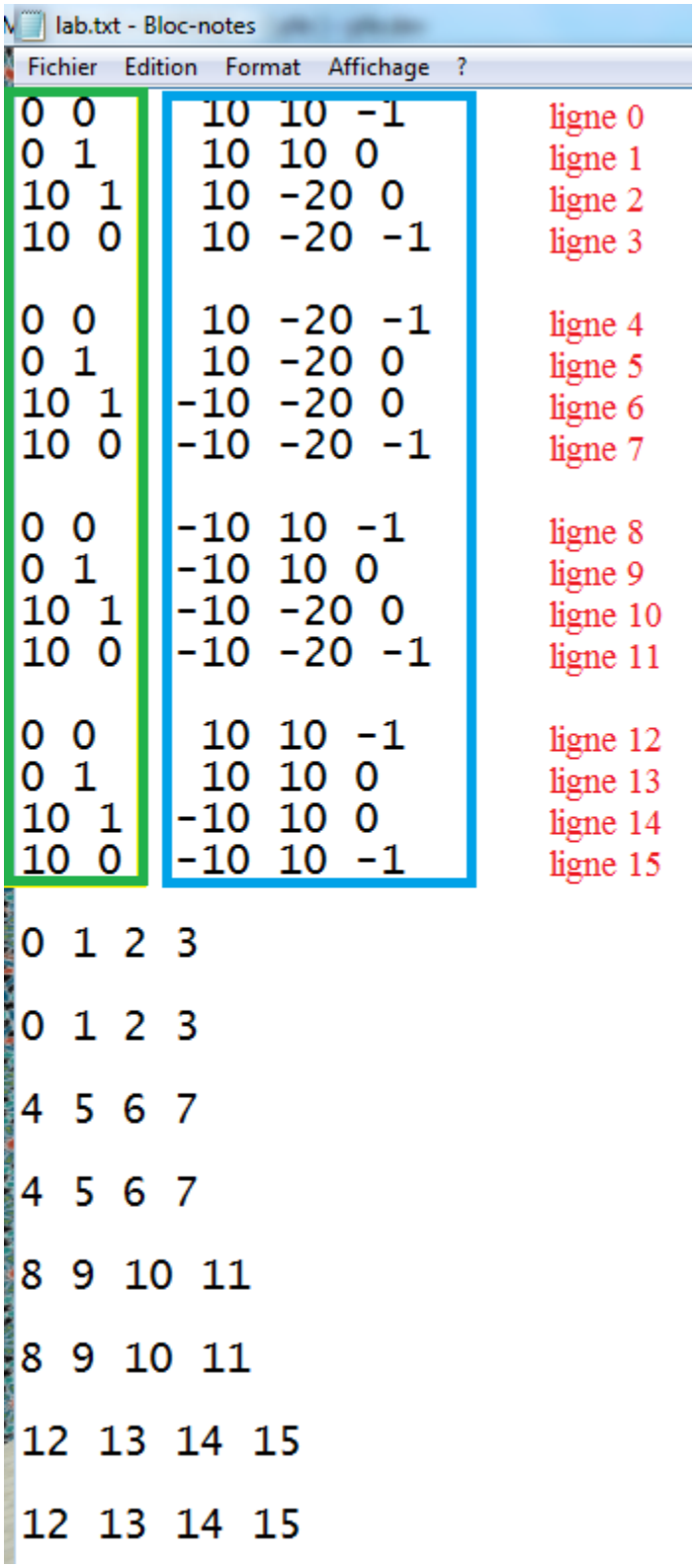


figure 3,7

Donc une seul ligne contient la description complète d'un sommet (texture et vertex).

Les quatre 1ere ligne décrivent le mur de droite puis respectivement le mur du devant, mur gauche, mur du derrière.

Les huit dernière ligne ceux avec seulement quatre paramètres (un pour chaque sommet) indiquent dans quelle ligne se trouve la description du sommet. Il faut indiquer ou se trouve les coordonnées textures et les coordonnées vertex pour chaque sommet

Le premier 0 indique que les coordonnées texture (u,v) du premier sommet sont dans la ligne 0

le 1 qui suie indique que les coordonnées texture (u,v) du deuxième sommet sont dans la ligne 1

ainsi de suite pour les deux autres sommets

Le 0 de la ligne suivante indique que les composantes du premier vertex (x,y,z) se trouve dans la ligne 0 .

Le 1 qui suie indique que les composantes du deuxième vertex (x,y,z) se trouve dans la ligne 1.

même procédé pour le reste des sommets

9)Conclusion

Mon projet consiste à modéliser des labyrinthe 3D avec l'API OpenGL. La richesse et la puissance d'OpenGL permettent la description et la modélisation des objets et des scènes les plus complexe grâce à une multitude de fonctions et de bibliothèques développer et partager par la communauté

Conclusion générale

Dans mon modeste travail j'ai réalisé un système de navigation dans un labyrinthe 3D en utilisant l'API opengl qui est beaucoup apprécié des infographistes pour sa richesse et sa portabilité ce qui signifie que cette technologie peut être utilisée sur n'importe quel système.

Avec cette étude j'ai pu donner libre cour à mon imagination pour concevoir un monde virtuel mais surtout explorer un nouveau domaine de la programmation en ce qui me concerne que je ne connaissais pas

Bibliographie et webographie

- Mr BENAISSA Mohamed el Amine et Mr LAMARI Amine (2013/2014) Moteur 3D, Système de navigation dans une scène 3D avec XNA Game studio
- Mme ALIOUI Wahiba et Mme BELKHODJA Souad (2012/2013) Visite virtuelle dans un établissement avec VRML
- http://openclassrooms.com/courses/creez-des-programmes-en-3d-avec-opengl/introduction-a-opengl
- http://openclassrooms.com/courses/creez-des-programmes-en-3d-avec-opengl/les-quadriques-1
- http://nehe.gamedev.net/tutorial/texture_mapping/12038/
- Eric BITTAR Cours OpenGL Université de Reims
- http://nehe.developpez.com/tutoriel/07-filtre-eclairage/
- Cour Infographie licence par Mr BENZIAN (2014/2015)