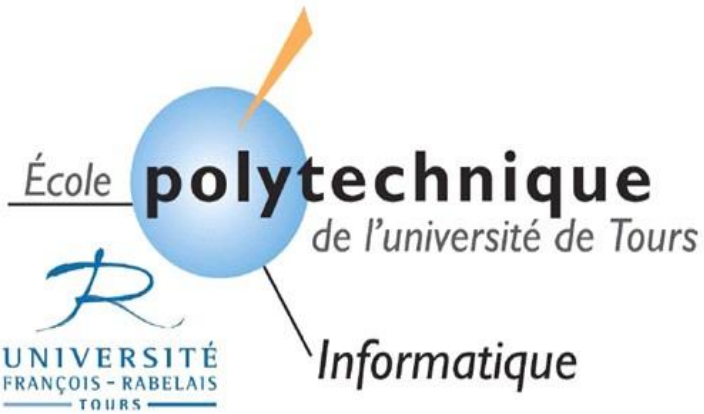

École Polytechnique de l'Université de Tours
64, Avenue Jean Portalis
37200 TOURS, FRANCE

Tél. +33 (0)2 47 36 14 14

Fax +33 (0)2 47 36 14 22

www.polytech.univ-tours.fr

Département Informatique
5ème année
2013-2014

Encadrant:

@univ-tours.fr

Etudiants :
Hongchi Zhou
hongchi.zhou@etu.univ-tours.fr
XU Shuo
shuo.xu@etu.univ-tours.fr

Table des matières

1 Introduction

1.1 Description du projet

A partir du logiciel Image J, il est demandé de proposer des méthodes d'analyse afin de pouvoir segmenter le mieux possible des sections de nerfs présents au sein d'images ultrasonores. Chaque section a les informations qui décrivent cette section. Pour chaque section, il y a 3 coupes sur les axes X, Y et Z.

L'essentiel du travail sera d'explorer l'ensemble des plugins natifs au logiciel ou mis à disposition par la communauté pour proposer plusieurs scénarios de traitement conduisant à des segmentations acceptables.

1.2 Objectifs

L'objectif du projet est de simuler un module de cerveau en 3d, de séparer le cerveau en 64 sections. Chaque section est présentée comme un cube. Le projet nous permet de choisir n'importe quelle section. Après avoir choisi une section, le programme cherche dans la base de données les informations sur cette partie du cerveau et les affiche. Après qu'une section est choisie, on peut obtenir les 3 coupes sur les axes X, Y et Z.

1.3 Présentation java3D

Java 3D est une interface de programmation (Application Programming Interface ou API) pour la plateforme Java visant la synthèse d'image 3D basée sur les graphes de scène.[i]

L'orientation de java3D au modèle objet a induit sa représentation sous forme d'un arbre sans cycle. C'est un graphe formé de nœuds et d'arcs, où un nœud ne pointe jamais vers l'un de ses parents (direct ou non), et où un nœud n'a jamais plusieurs pères.

La racine de l'arbre est un univers virtuel auquel vous rattachez un point de référence qui va servir de repère pour placer vos différents objets.

Un objet peut être vu comme un regroupement d'objets élémentaires. Chaque objet élémentaire est issu de l'association d'une géométrie (ensemble de faces triangulaires ou de primitives) et d'une apparence (couleur, transparence, texture). Java3D fournit les primitives suivantes: sphère, boîte, cône et cylindre.

Voici un schéma qui résume l'organisation d'une scène 3D en Java3D :

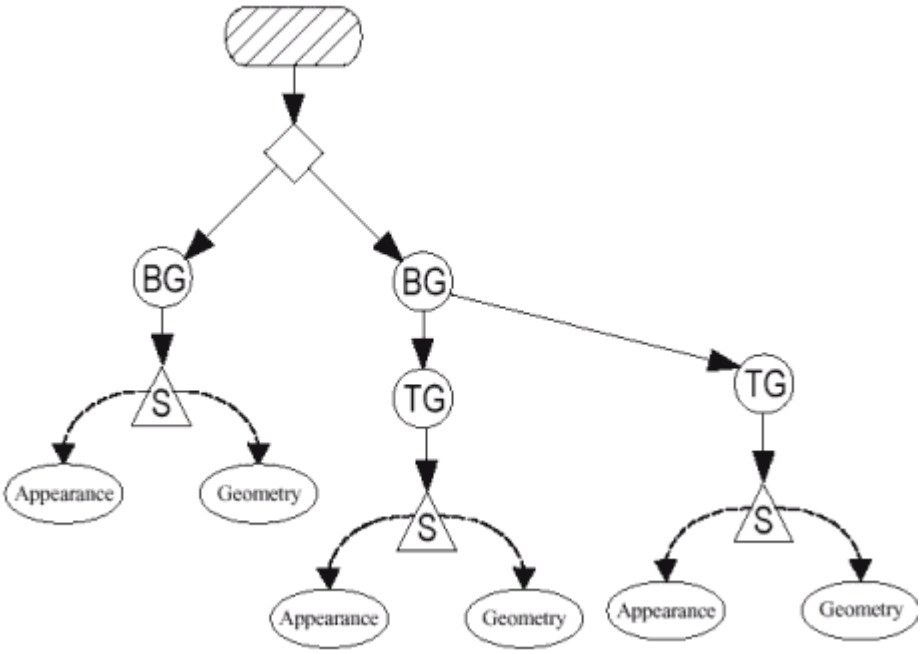

Figure 1.1 – L'organisation d'une scène 3D[ii]

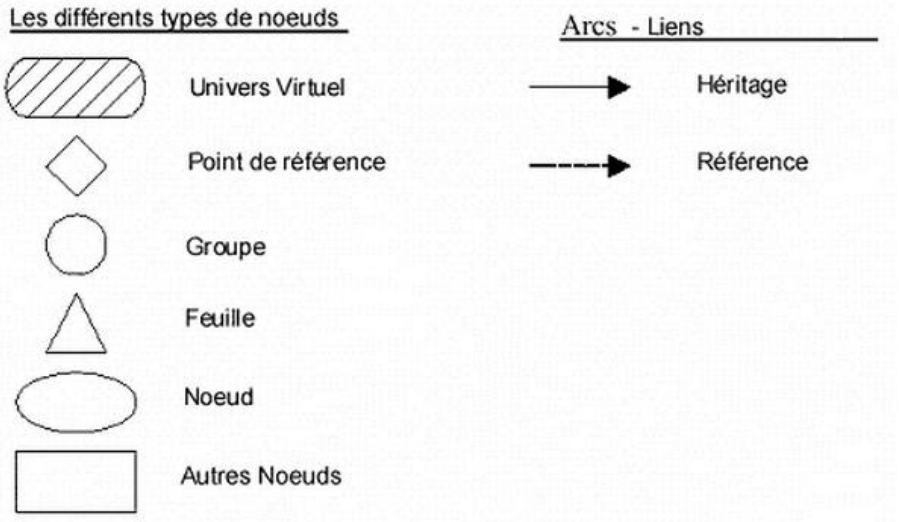

Figure 1.2 – L'explication des éléments[iii]

Univers Virtuel : il définit un univers. Un univers permet un Java 3D program créer une arène séparâtes et distinct pour définir les objets et leur relations.

BG : Regroupement d'objets (BrancheGroup) Le BranchGroup nœud serves comme la racine d'un branche graph. Totalement, le BranchGroup nœud et tous leur fils de la branche graph.

TG : Group de transformation (TransformGroup) Les TransformGroup serviront donc à placer les objets dans ce repère ou à créer des animations. Les TransformGroup permettent de faire des translations et des rotations.

S : Object élémentaire

Dans l'univers de JAVA3D, le système de coordonnées est présenté dans la figure suivante.

A l'aide des TransformGroup, on peut positionner deux objets l'un par rapport à l'autre.

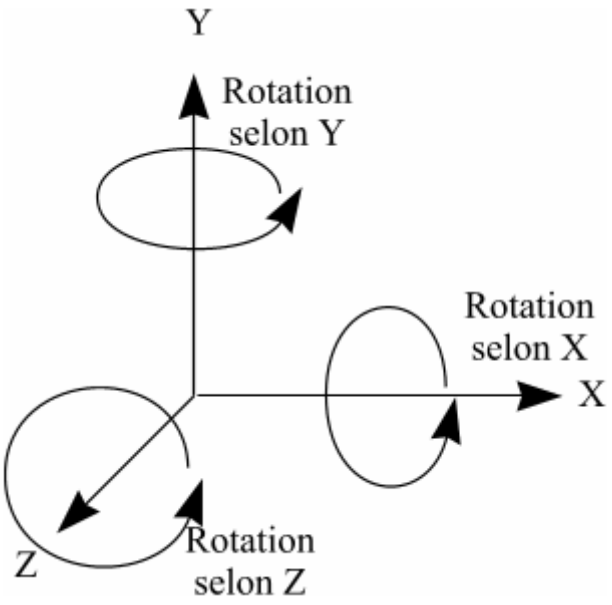

Figure 1.3 – Le système de coordonnées de JAVA3D[iv]

2. Travail Réalisé

2.1 La base de données

Dans le projet, chaque cube est comme une part d'une image médicale. Chaque cube a un nom et des informations. Pour les informations d'un cube, il faut utiliser la base de données dans laquelle les informations sont enregistrées.

La base de données est créée par Baptiste Chartier, l'étudiant en 4ème année. Il y a 8 tables : article, auteur, contient, definit, ecrit, element, possede, structure. Dans le projet, nous utilisons la table structure. La table structure est comme suivante.

Nom	Type	ALLOW NULL
id_structure	int(11)	NO
Id_parent	int(11)	YES
nom	varchar(255)	NO
description	text	NO
abbreviation	varchar(255)	YES

Figure2.1 - La table structure

Ici, id structure est la clé primaire. Cela augmente automatiquement. Le code de construction est suivant :

```
CREATE TABLE IF NOT EXISTS `structure` (
    `id_structure` int(11) NOT NULL AUTO_INCREMENT,
    `id_parent` int(11) DEFAULT NULL,
    `nom` varchar(255) NOT NULL,
    `description` text NOT NULL,
    `abbreviation` varchar(255) DEFAULT NULL,
    PRIMARY KEY (`id_structure`),
    KEY `id_parent` (`id_parent`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=125 ;
```

Les données sont déjà insérées dans la base de données avec le script de Baptiste Chartier.

2.2 Les classes dans l'application

L'application est réalisée avec le langage JAVA. Dans l'application, il y a 4 classes pour réaliser l'application.

1) DB : réaliser de connecter à la base de données et lire des informations
2) Info : Enregistrer les informations qui sont lues de la base de données
3) ReColorCube : réaliser un cube avec un nom et les informations. Cette classe hérite de la classe ColorCube qui est une classe dans la bibliothèque de JAVA.
4) test1 : réaliser d'afficher les cubes et réaliser les opérations.

2.2.1 La classe DB

Parce qu'il faut lire des informations de la base de données, nous avons écrit une classe DB pour connecter à la base de données et lire des informations. Dans le projet, nous n'avons pas besoin d'enregistrer les informations, nous n'avons pas écrit les fonctions pour enregistrer et modifier les informations.

Dans la classe, il y a 5 propriétés et 5 fonctions.

Les 5 propriétés sont suivantes.

Nom	Type	Visibilité	Description
driver	String	private	Driver JDBC qui permet d'établir une connexion entre un programme java et un système de gestion de base de données
url	String	private	L'url qui indique où se trouve la base de données.
user	String	private	L'utilisateur de la base de données
password	String	private	Le mot de passe de la base de données
conn	Connection	private	Une propriété retournée après la connexion à la base de données.

Figure2.2 – 5 propriétés de la classe DB

Les explications des 5 fonctions sont suivantes.

1) Le constructeur par défaut public DB() : Il n'y a pas aucun argument. La visibilité est public. Cette fonction est d'initialiser les 5 propriétés. La valeur de la propriété est NULL. La valeur de la propriété driver est "com.mysql.jdbc.Driver".
2) Le 2ème constructeur public DB(String var_databaseName, String var_user, String var_password): Cette fonction est pareille au contructeur par défaut, mais dans cette fonction, il y a 3 arguments : le nom de la base de données, l'utilisateur de la base de données et le mot de passe.
3) La fonction private void connection() : Cette fonction n'a pas aucun argument. La visibilité est private. Elle ne retourne rien. Cette fonction sert à connecter à la base de données. Puis la valeur retournée de la classe DriverManager.getConncection() est affectée à la propriété conn.
4) La fonction private void closeConnection() : Cette fonction n'a pas aucun argument. La visibilité est private. Cette fonction sert à clore la base de données.
5) La fonction public Object[] getResultSelect(String sql) : Cette fonction a un argument sql. C'est la phrase de sql pour rechercher les informations. La visibilité est public. La valeur retournée est un tableau de type Object. Nous avons choisi le type Object, car les informations retournées ont les différents types.

2.2.2 La classe Info

Cette classe est d'enregistrer les informations qui sont lues de la base de données. Dans la classe, il y a 3 propriétés et 5 fonctions.

Les 3 propriétés sont dans la table suivante.

Nom	Type	Visibilité	Description
idStructure	int	private	L'identité de structure
nom	String	private	Le nom d'une structure
description	String	private	La description d'une structure

Figure2.3 – 3 propriétés dans la classe Info

Les descriptions de 5 fonctions sont suivantes.

1) Le constructeur par défaut public Info() : Cette fonction n'a aucun argument. Le constructeur sert à initialiser les propriétés. L'idStructure est 0 ; Le nom est "", La description est "".
2) Le 2ème constructeur public Info(int var_idStructure) : Le constructeur a un argument : l'Id d'une structure. Selon l'argument var_idStructure, nous recherchons les informations correspondantes dans la base de données. Si les informations existent, les informations sont affectées aux propriétés. Si les informations n'existent pas, le nom et la description sont affectés par la chaîne "Not Found".
3) Les 3 fonctions suivantes servent à obtenir les propriétés.
   public int getIdStructure() : retourner idStructure
   public String getNom() : retourner nom
   public String getDescription() : retourner description

2.2.3 La classe ReColorCube

Cette classe hérite de la classe ColorCube. La classe ColorCube est une classe de la bibliothèque de JAVA3D. ColorCube représente un cube avec 6 faces de couleurs différentes. Dans la classe ReColorCube, nous avons ajouté 5 propriétés. Et il y a 4 fonctions.

Les 5 propriétés sont suivantes.

Nom	Type	Visibilité	Description
nomCube	String	private	Le nom d'un cube. Le nom est un numéro.
X	float	private	La position sur l'axe X
Y	float	private	La position sur l'axe Y
Z	float	private	La position sur l'axe Z
info	Info	private	Les informations de ce cube

Figure2.4 – 5 propriétés de la classe ReColorCube

Les 4 fonctions sont suivantes.

1) Le constructeur public ReColorCube(double scale, String var_nomCube, float var_X, float var_Y, var_Z) : Créer un ColorCube en appelant super(scale). L'argument scale définit la taille d'un cube. Les coordonnées de deux coins opposés d'un cube sont (-scale, -scale, -scale) et (scale, scale, scale). Par défaut, scale est égal à 1. Puis, le constructeur initialise nomCube, X, Y, Z et info. L'argument du constructeur de la classe est var_nomCube. Mais il faut changer le type.
2) La fonction public float[] getXYZ() : retourner X, Y et Z.
3) La fonction public String getNomCube() : retourner le nom du cube.
4) La fonction public Info getInfo() : retourner l'info.

2.2.4 La classe test1

La classe test1 est de réaliser l'application. Cette classe réalise l'affichage des cubes et nous permet de faire les opérations sur les cubes.

La figure suivante est l'affichage des cubes quand le programme s'exécute.

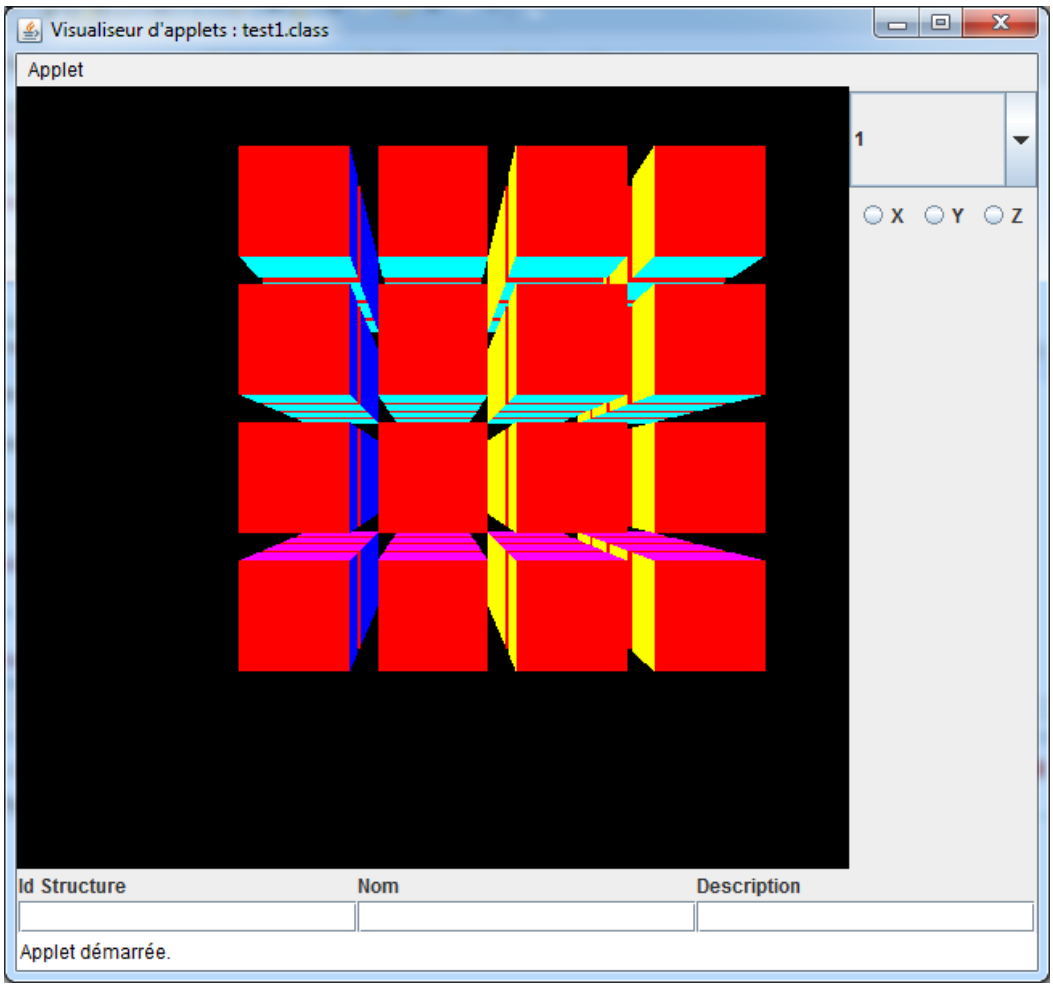

Figure2.5 – L'affichage initial

Dans cette classe, il y a certaines propriétés et 2 fonctions importantes.

Ici, nous présentons les propriétés.

Dans ce projet, il y a 64 cubes. Donc, Nous avons créé un tableau rcc[] dont la taille est 64 et le type est ReColorCube.

Chaque cube peut bouger et a besoin de la translation. Nous avons créé un tableau objRotate[] dont la taille est 64 et le type TransformGroup. La classe TransformGroup est une classe de la bibliothèque de JAVA3D qui permet de faire des translations et des rotations.

Après avoir construit tous les objets, il faut définir un rassemblement d'objets correspondant à la classe BranchGroup. La fonction createSceneGraph() retourne le type BranchGroup.

Les 2 fonctions importantes sont public BranchGroup createSceneGraph() et public void init().

La fonction createSceneGraph() : Dans la fonction, d'abord, il faut créer une instance de la classe BranchGroup. Puis, nous mettons les cubes aux différentes places. Pour cela, il faut utiliser la classe Transform3D qui est en fait une matrice pouvant définir des translations ou rotations. La matrice est les coordonnées.

Puis, nous créons une instance de la classe TransformGroup avec l'argument, l'instance de la classe Transform3D. Nous voulons que les cubes puissent faire la rotation. Nous avons ajouté la classe MouseRotate qui est la classe de la bibliothèque de JAVA3D à l'instance de TransformeGroup. Ensuite, nous ajoutons l'instance de TransformGroup à objRoot, l'instance de la classe BranchGroup. A la fin, on retourne objRoot.

La fonction public void init() : Cette fonction construit l'univers de JAVA3D et réalise l'animations des cubes et les opérations.

Au début, on construit l'interface. Il y a une liste de nom, les champs de textes et la toile. Pour afficher les cubes, il faut créer une instance canvas3D de la classe Canvas3D qui appartient à la bibliothèque de JAVA3D. Puis, nous créons l'instance u de la classe SimpleUniverse qui est aussi la classe de JAVA3D avec l'argument canvas3D. A la fin, l'instance de la classe BranchGroup est ajoutée à u.

L'animation

Quand on clique sur le nom dans la liste, un cube fait la translation sur l'axe X. Nous avons réalisé l'animation en changeant le point central d'un cube toutes les certaines millisecondes. On change le point central, puis utilise Thread.sleep() pour faire une pause. Après que le cube arrête, on recherche les informations correspondantes à ce cube dans la base de données. Les informations sont affichées dans les champs de textes.

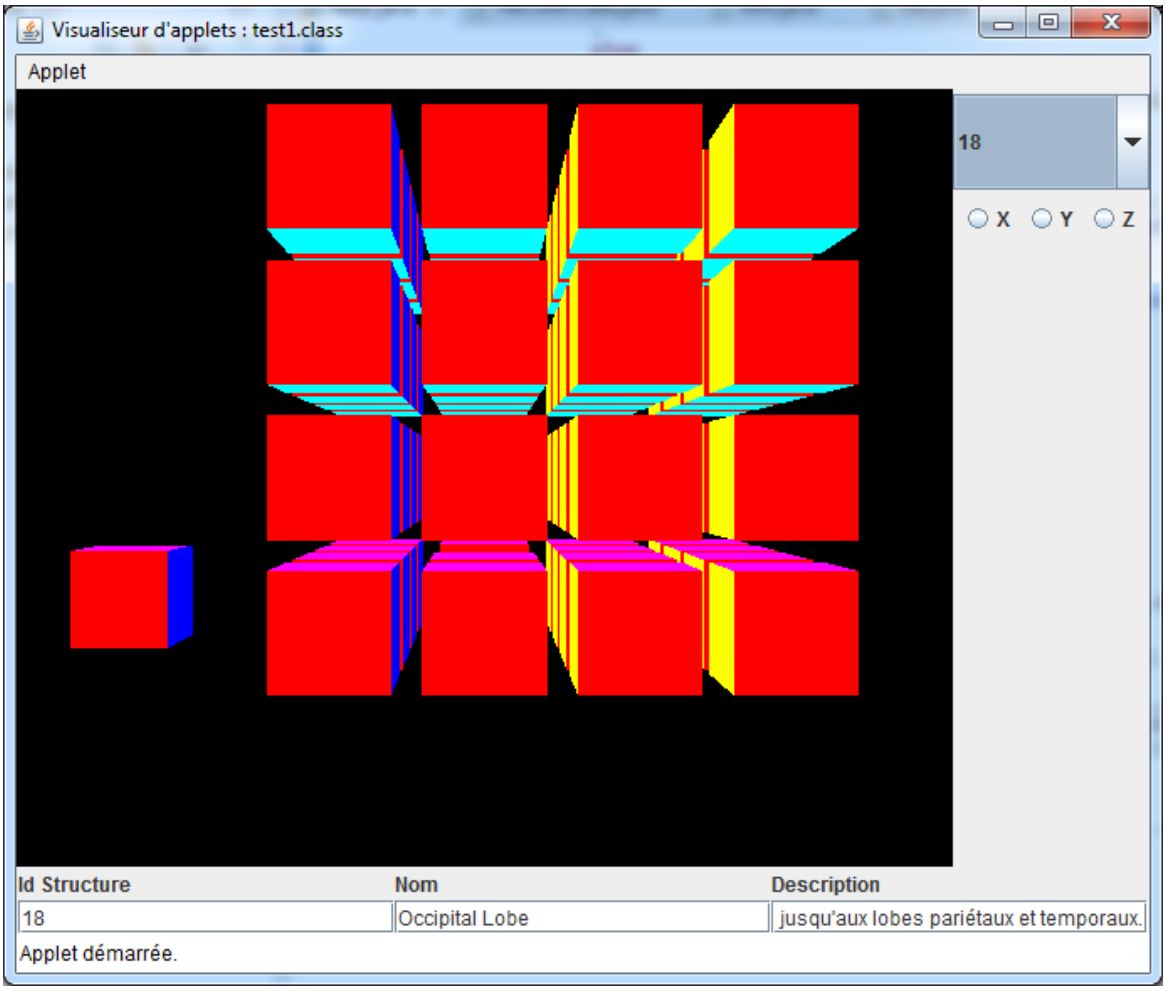

Figure2.9 – Le cube après bouger

Les coupes sur 3 axes

Un objectif du projet est d'obtenir les coupes sur les axes X, Y et Z. Mais nous n'avons pas réussi à bien réaliser cet objectif. Nous pouvons couper les cubes, mais nous n'avons pas trouvé la zone d'influence correcte. Par exemple, quand on choisit Y sur la case d'option, la figure suivante affiche le résultat.

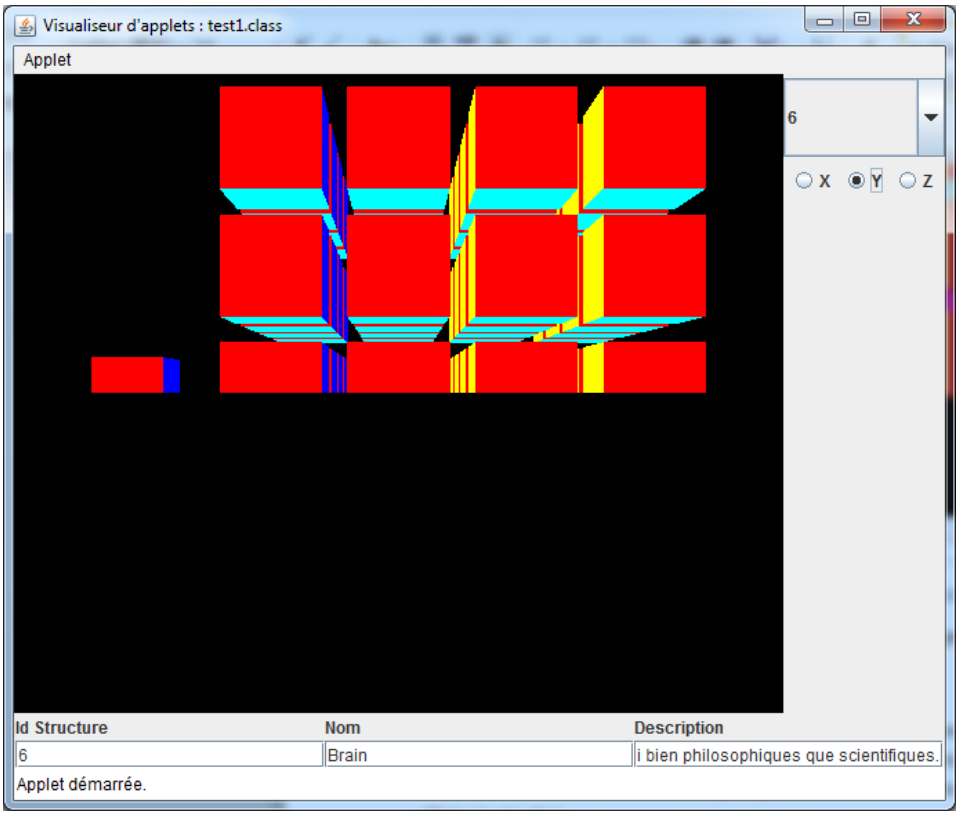

Figure2.10 – La coupe sur l'axe Y

On peut faire la rotation d'un cube pour voir la coupe comme la figure suivante.

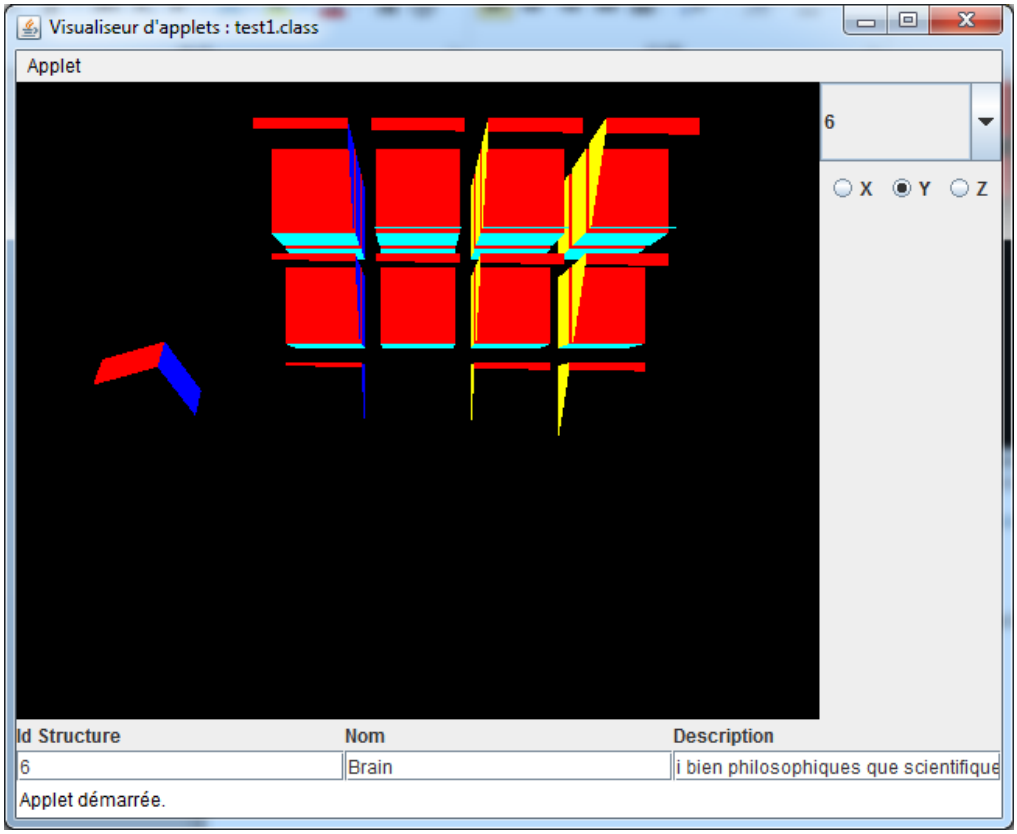

Figure2.11 – La coupe sur l'axe Y après la rotation

On peut voir la coupe est noire. Parce que la classe ColorCube a 6 faces de couleurs différentes, mais l'intérieur du cube est noir. Couper un cube est réalisé avec la classe ModelClip qui appartient à la bibliothèque de JAVA3D. L'instance de ModelClip est ajoutée au tableau de TransformGroup. Les fonctions setPlane(), setEnable() et setInfluencingBounds() réalise à couper un cube.

Les APIs des classes peuvent être trouvés sur le site suivant :
http://download.java.net/media/java3d/javadoc/

3. L'explication d'utilisation

Pour exécuter cette application, il suffit de 4 bibliothèques et 1 fichier du type DLL (Dynamic Link Library). Il faut aussi créer la base de données qui s'appelle projetoption.

C'est une application de JAVA, il faut aussi s'installer JRE et JDK.

Les 4 bibliothèques sont j3dcore.jar, vecmath.jar, j3dutils.jar et mysql-connector-java.jar.

Le fichier du type DLL est j3dcore-ogl.dll. Ce fichier s'installe au répertoire où se trouve le projet.

Après la préparation, on peut exécuter l'application.

4. Conclusion

En conclusion, ce projet d'option est réalisé avec la bibliothèque de JAVA3D. Dans ce projet, nous créons 64 cubes. Chaque cube a un nom. Quand on choisit le nom sur la liste de noms, le cube bouge à gauche sur l'axe X. Il faut aussi obtient les coupes sur 3 axes et rechercher les informations de ce cube dans la base de données qui est créée par Baptiste Chartier, l'étudiant en 4ème année. Nous avons écrit 4 classes qui sont présentées ci-dessus afin de réaliser l'application.

En ce qui concerne ce projet, il y a des points à améliorer. L'interface doit être améliorée pour bien utiliser. La fonctionnalité pour obtenir les coupes doit être améliorée. Pour couper un objet, il faut utiliser la classe ModelClip. Le problème pour nous est de ne pas trouver une zone correcte. Donc, il y a souvent des problèmes d'affichage.

On peut trouver les APIs de toutes les classes de JAVA3D sur l'internet.

---

[i] http://fr.wikipedia.org/wiki/Java_3D
[ii] http://rvirtual.free.fr/programmation/java3d/chap1/chap1.html
[iii] http://rvirtual.free.fr/programmation/java3d/chap1/chap1.html
[iv] http://rvirtual.free.fr/programmation/java3d/chap1/chap1.html