Table des matières

Table des matières....1
I Travail demandé, énoncé....2
II Mise en situation....2
1 Notions de physique....2
a) Quelques définitions....2
Métrique de Robertson-Walker....2
Constante de Hubble....2
Paramètre de décélération....2
Décalage vers le rouge....2
Équation de Friedman....2
Densité d'énergie....3
Constante cosmologique....3
b) Modèles cosmologiques....3
c) Besoin exprimé par les cosmologistes....4
d) Données disponibles....4
Ascension droite....4
Déclinaison....4
Décalage vers le rouge....4
2 Programme initial....5
a) Contexte....5
b) État....5
III Travail réalisé....5
1 Etude du programme initial....5
a) Fonctionnement....5
b) Le toolkit Motif....5
c) Carences....6
2 Choix de travail....6
a) Architecture globale....6
b) Choix d'un nouveau toolkit....6
i. TCL/TK....6
ii. KDE/Qt....6
iii. GTK....6
c) Données....6
3 Réalisation....6
a) Portage et fonctionnalités globales....6
b) Sélection pour la mise en valeur des grandes structures....7
c) Formats de fichiers....8
IV Bilan....10

I Travail demandé, énoncé

Le TER consistera en l'implémentation dans un environnement UNIX d'un logiciel (Univers Viewer) dont les sources sont écrites en C et utilisent une bibliothèque graphique trop particulière. L'intérêt de ce travail est de rendre facile l'utilisation de ce logiciel, qui permet de visualiser espace courbe (géométrie non euclidienne de l'espace-temps) la position des objets astronomiques à grand décalage spectral. D'autre part, les étudiants pourront développer d'autres outils graphiques et statistiques, permettant de mieux appréhender le problème des grandes structures de l'univers.

II Mise en situation

1 Notions de physique

a) Quelques définitions

– *Métrique de Robertson-Walker*

On utilise la métriqueRobertson-Walker pour un espace isotropique, homogène de courbure spatiale constante k avec k>0 pour un espace ouvert (Riemannien), 0 pour un espace plat (Euclidien) et k<0 (Lobatchevsky) pour un espace fermé, en posant:

$$ds^2 = dt^2 - a(t)\, d\sigma^2$$

avec t étant le temps cosmique, a(t) le paramètre d'expansion et d%sigma la métrique dans un espace à 3 dimensions

– *Constante de Hubble*

La constante de Hubble est définie par :

$$H = \frac{\dot{R}}{R}$$

où R est déterminé à partir de la courbure. Pour la valeur actuelle, on ajoute un indice 0.

– *Paramètre de décélération*

Le paramètre de décélération est défini par :

$$q = \frac{-(\ddot{R})}{RH^2}$$

Pour la valeur actuelle, on ajoute un indice 0.

– *Décalage vers le rouge*

Lorsqu'on observe le spectre de différents objets, on constate qu'il est décalé vers le rouge ou vers le bleu. Ce décalage augmente vers le rouge avec la distance de l'objet. Il est dû à l'effet cosmologique : l'espace s'étend, la lumière émise par l'objet voyage pendant un certain temps au cours du quel l'espace se dilate, ce qui implique que la longueur d'onde de la lumière augmente et provoque un décalage vers le rouge.

– *Équation de Friedman*

Cette équation doit normalement être dérivée à partir des équations de champ de la relativité générale. On peut néanmoins dériver cette équation à partir de la cosmologie newtonienne.

Considérons une distribution homogène de matière ρ. On regarde la masse à l'intérieur d'une sphère de rayon R centrée sur l'observateur et on néglige la masse en dehors de cette sphère.

L'accélération d'une particule test située en R est donnée par :

$$\ddot{R}=\frac{-GM(R)}{R^2}+\frac{1}{3}\Lambda c^2 R \quad (1)$$

ou, en multipliant par $\dot{R}$,

$$\dot{R}\ddot{R}=-\dot{R}\frac{GM(R)}{R^2}+\frac{1}{3}\Lambda c^2 R\dot{R} \quad (2)$$

En intégrant on obtient :

$$\frac{1}{2}\dot{R}^2-\frac{GM(R)}{R}-\frac{1}{6}\Lambda c^2 R^2=cte=\frac{-kc^2}{2} \quad (3)$$

La constante d'intégration est déterminée par la relativité générale. En utilisant (2) multipliée par R dans (3), et en multipliant par $\frac{2}{R^2}$, on obtient l'équation de Friedmann :

$$\frac{\dot{R}^2}{R^2}+2\frac{\ddot{R}}{R}-\frac{1}{3}\Lambda c^2=K \quad (4)$$

Avec $M(R)=\frac{4}{3}\Pi R^3\rho$, on on obtient :

$$\frac{\dot{R}^2}{R^2}=\frac{(8\Pi G\rho)}{3}+\frac{1}{3}\Lambda c^2-K \quad (5)$$

Ce qui à l'époque actuelle donne :

$$K_0=H_0^2((2q_0-1)+\frac{\Lambda}{3}\frac{c^2}{H_0^2}) \quad (6)$$

– *Densité d'énergie*

En définissant les densités d'énergie suivantes :

$$\Omega_M=8\Pi\frac{G}{3}H_0^2\rho \qquad \Omega_\Lambda=\Lambda\frac{c^2}{3}H_0^2 \qquad \Omega_k=\frac{K_0}{H_0^2} \quad (7)$$

Ce qui dans (7) donne :

$$\Omega_\Lambda+\Omega_M-\Omega_k=1$$

– *Constante cosmologique*

La constante cosmologique Λ introduite précédemment apparaît comme une constante d'intégration dans les équations de champ de la relativité générale. Elle peut être interprétée comme une force répulsive augmentant avec la distance.

Pour les astronomes, la constante cosmologique sert essentiellement à satisfaire les équations de densité d'énergie, alors que pour le physicien des particules elle est reliée à l'énergie du vide.

b) Modèles cosmologiques

On travaille nécessairement avec des modèles d'univers cohérents. Pour maintenir cet état, on

dispose du modèle de Friedmann-Lemaître :

$$P(a) = \lambda_0 a^4 - K_0 a^2 + \Omega_0 a + \alpha_0$$

dans le quel on cherche à maintenir P(a) = 1.

Avec :

$$\lambda_0 = \frac{1}{3}\frac{\Lambda}{H_0^2}$$

$$k_0 = \frac{K_0}{H_0^2}$$

$$\Omega_0 = \frac{8}{3}\Pi G \frac{\rho}{H_0^2}$$

$$\alpha_0 = \frac{8}{45}\Pi^3 G (kT_0)^4 \frac{h^{-3}}{H_0^2}$$

$T_0 = 2{,}73 \pm 0{,}03\, K°$
k est la constante de Boltzmann
h est la constante de Plank

c) Besoin exprimé par les cosmologistes

Grâce aux observations de l'univers aux grandes échelles, les astronomes ont pu noter que les grands objets (galaxies, quasars, ...) se répartissaient en structures appelées amas. Le fait que leur répartition ne soit pas homogène, comme on l'a cru au début, incite à chercher les causes de cette inhomogénéïté et les effets que cela peut avoir sur les modèles d'univers que l'on connaît (affirmation, infirmation). Pour cela, les astronomes ont besoin d'un outil leur permettant de repérer les grandes structures dans l'espace-temps, de voir l'évolution de ses structures en fonction des divers modèles cosmologiques et d'avoir accès a des informations précises sur la possible existence et les conséquences de ces modèles.

d) Données disponibles

Nous disposons aujourd'hui d'un grand nombre d'informations sur les divers objets de l'univers. Dans notre cas, nous avons essentiellement besoin de trois paramètres pour déterminer la position d'un objet :

- ***Ascension droite***

  Cette information s'apparente à la latitude de la voûte céleste vers le quel on regarde.

- ***Déclinaison***

  Par la même analogie, la déclinaison indique la longitude de la voûte céleste vers la quelle on regarde.

Combinées, ces deux informations nous donnent la position d'une étoile dans le ciel. Reste à savoir à quelle distance elle se trouve :

- ***Décalage vers le rouge***

  Comme expliqué précédemment, cette information nous indique à quelle distance se trouve l'objet regardé.

Nous savons donc désormais repérer un objet dans un espace Euclidien. Mais seulement, l'univers n'est pas forcement un espace Euclidien puisque sa courbure k n'est pas précisément définie. Nous devons donc passer la position de chaque objet dans un espace à 4 dimensions. Ce calcul se fait par l'intermédiaire de la distance comobile :

$$\tau(z) = \sqrt{(k_0)} \int_{(1+z)^{-1}}^{1} \left(\frac{da}{\sqrt{(P(a))}} \right)$$

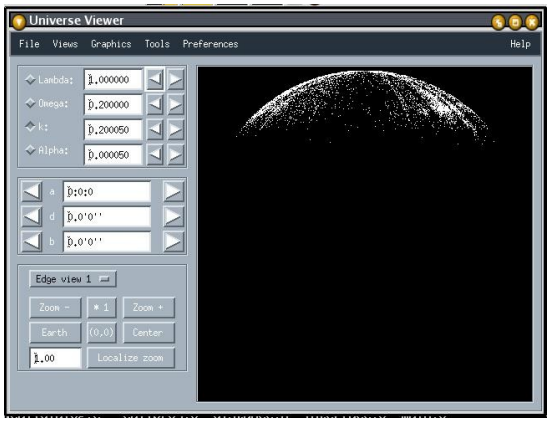
avec P(a) défini précédement.

2 Programme initial

a) Contexte

Ce logiciel à été conçu dans le cadre d'un projet informatique faisant parti de la formation du DEA « Physique des particules, Physique mathématique et Modélisation » de la faculté d'AIX-MARSEILLE II. Les programmeurs : Rémi Lafaye et Lionel Spinelli.

b) État



Dans l'ensemble, l'application était déjà très complète et largement fonctionnelle. Son principal problème résidait dans son interface écrite avec une librairie graphique obsolète et difficilement exploitable avec les systèmes actuels. Nous avons pu noter par ailleurs nombre de maladresses de programmation comme la gestion des allocations mémoires peu prudente.

En fonction des configurations testées, nous avons eu beaucoup de mal à la faire fonctionner de manière satisfaisante. Non pas que ce soit l'application qui pose problème, tout dépendait de la librairie Motif qui s'est révélée capricieuse en fonction qu'elle soit présente elle-même ou qu'elle soit remplacée par Lesstif, censée être compatible, qui a eu tendance à dénaturer les performances d'affichage.

III Travail réalisé

1 Etude du programme initial

a) Fonctionnement

Le fonctionnement global était satisfaisant et se s'est pas particulièrement vu modifié.

L'idée est simple, on part d'une base de données donnant la position de corps célestes par leur ascension droite, déclinaison et redshift que l'on convertit en coordonnées spatio-temporelles. Sachant qu'un espace à 4 dimensions dispose de 6 couples d'axes othogonaux, on projette donc 6 vues selon les axes zt, yt, yz, xt, yt et zt. Autour de ces projections, on peut faire varier les constantes cosmologiques pour changer le modèle d'univers et la position de l'observateur.

b) Le toolkit Motif

Avant de pouvoir comprendre le fonctionnement interne de l'application, il a fallu savoir qui faisait quoi au sein de l'interface. Et donc pour cela se familiariser un tant soit peu avec Motif fut

nécessaire.

c) Carences

Comme expliqué précédemment nous avons pu constater certaines maladresses de programmation avec en particulier la gestion de allocations mémoire.

D'un autre coté nous avons pu détecter certains bogues et des possibilités d'optimisations non exploitées.

2 Choix de travail

a) Architecture globale

Si le fonctionnement e l'application s'est vu respecté, seul le code des fonctions mathématiques et du traitement des données à été conservé. Tout ce qui concerne l'interface à entièrement été reécrit.

b) Choix d'un nouveau toolkit

Le but initial étant de reécrire l'application avec une nouvelle librairie graphique, il nous a fallu faire un choix parmi celles disponibles aujourd'hui.

i. TCL/TK

Relativement facile d'accès, portable et efficace, le couple TCL/TK ne repose pas sur le même langage de programmation que l'application d'origine. Bien qu'il existe des implémentations de TK pour le C, nous n'avons pas retenu cette librairie.

ii. KDE/Qt

Très sophistiquée, cette librairie présentait l'énorme avantage de pouvoir reprendre du code Motif sans vraiment avoir à le retravailler. Seulement, cette fonctionnalité n'était disponible qu'en C++ et aurait donc nécessité de reécrire le code quand même puisque l'application était écrite en C ... Dommage !

iii. GTK

Très utilisée et plutôt simple d'accès à la base, cette librairie se révèle très puissante si on sait correctement s'en servir. Principalement adaptée au C (bien qu'il en existe d'autres implémentations) et au fonctionnement proche du couple Xt/Motif, elle se révèle à nos yeux la mieux adaptée pour ce projet. C'est donc sur elle que s'est porté notre choix.

c) Données

Les entrées de données possibles sont un format texte ASCII et un format binaire (QSO). Au cours de nos recherches, nous avons pu constater qu'il existe d'autres formats comme XML. Nous avons donc choisi de restructurer légèrement les fonctions d'ouverture de bases de données afin d'ajouter un format XML et de faciliter l'ajout d'autres formats à venir.

3 Réalisation

a) Portage et fonctionnalités globales

Pportage de l'application de Motif vers GTK, de l'adaptation de l'ensemble des fonctions existantes dans la première version pour faciliter l'intégration de nouvelles (en particulier pour la gestion des fichiers) et de la correction des problèmes de gestion de mémoire.

Certains paramètres autrefois inaccessibles sont désormais exploitables, en particulier sur le graphe « Lambda vs Oméga ».

b) Sélection pour la mise en valeur des grandes structures

Le zoom utilise un hyper-cube, c'est à dire, en utilisant les différentes vues dont on dispose. Cela consiste a définir grâce aux vues dont on dispose, un espace de forme cubique (en sélectionnant sur chaque vue un rectangle qui représentera la zone dans laquelle on veut zoomer). Une fois cet espace défini, tous les points à l'intérieur de cette zone seront affichés à l'écran.

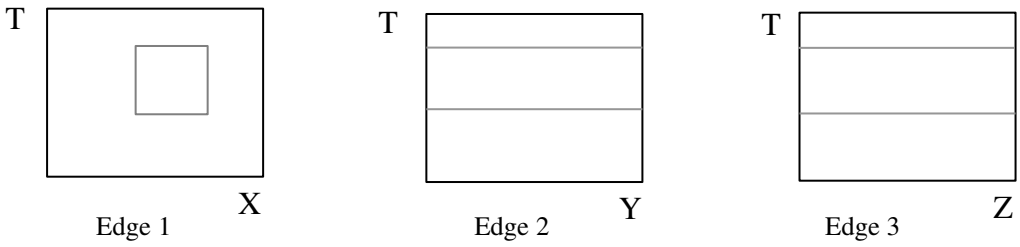
Etant donné que nous travaillons dans un univers à quatre dimensions, la table de vérité correspondante donne six ensembles dont un couple de valeurs est 1 et l'autre couple est 0.

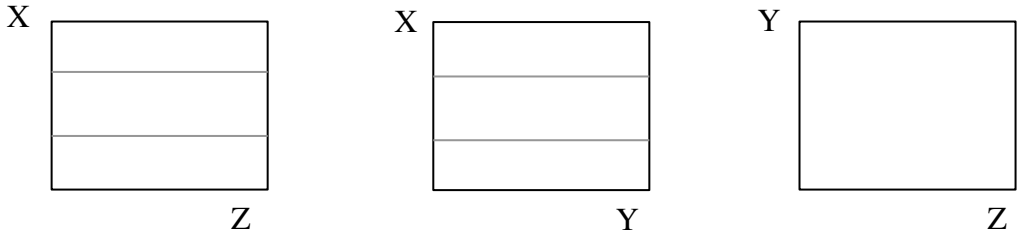
	X	Y	Z	T
	0	0	0	0
	0	0	0	1
	0	0	1	0
Face ZT	0	0	1	1
	0	1	0	0
Face YT	0	1	0	1
Face YZ	0	1	1	0
	0	1	1	1
	1	0	0	0
Face XT	1	0	0	1
Face XZ	1	0	1	0
	1	0	1	1
Face XY	1	1	0	0
	1	1	0	1
	1	1	1	0
	1	1	1	1

Ces six couples correspondent aux axes de coordonnées de chacune des faces d'un cube qui nous permet de voir notre représentation sous tous les angles.

Mode opératoire :

Nous avons créé un tableau représentant le nombre de coordonnées qu'il nous faut afin de pouvoir exécuter le zoom. Les drapeaux représentant ces coordonnées sont initialisés à zéro, et lorsqu'une coordonnée dont nous avons besoin est sélectionnée, le drapeau correspondant est mis à 1 afin de ne plus la modifier ultérieurement.



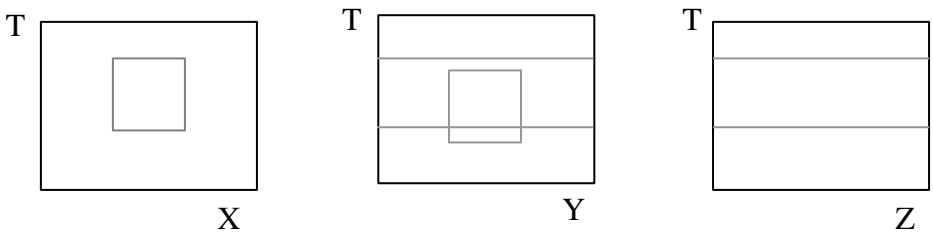


Front 1 Front 2 Front 3

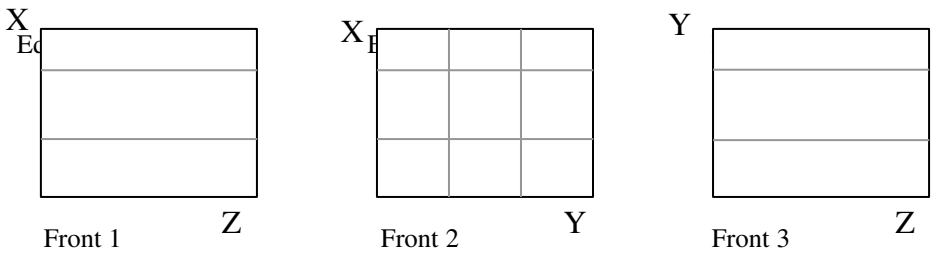
Lorsqu'une zone est sélectionnée, les drapeaux correspondant aux deux coordonnées définissant la vue sont passés à 1 et les coordonnées sont enregistrées.

Alors, une bande apparaît sur les autres vues où ces coordonnées apparaissent, afin d'indiquer à l'utilisateur la zone qui a été sélectionnée précédemment.

Alors, l'utilisateur sélectionne une autre vue et recommence à sélectionner une zone de l'écran, et, comme précédemment, les coordonnées dont le drapeau n'est pas à 1 sont mémorisées, et si les quatre coordonnées ne sont pas encore initialisées, une nouvelle bande apparaît sur les différentes vues dont les coordonnées sont mémorisées.



Edge 1 Edge 2 Edge 3



Front 1 Front 2 Front 3

Si toutes les coordonnées nécessaires au zoom sont mémorisées, alors le zoom peut être effectué, et tous les points qui se trouvent dans la limite de la zone sélectionnée seront affichés.

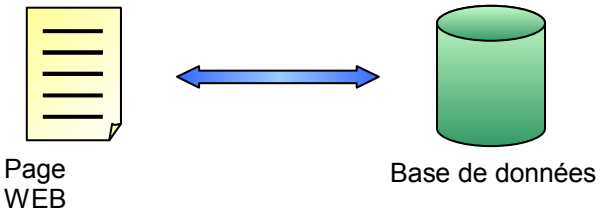
Sinon, une dernière sélection sera nécessaire afin de mémoriser la dernière coordonnée manquante.

c) Formats de fichiers

1) Travail demandé

Il a fallu programmer un certain nombre de fonctions permettant l'acquisition, à partir du WEB, de données contenu dans des fichiers XML (le format de fichier le plus intéressant pour la

lecture des données…) dans un premier temps et dans un second temps, de faire l'inverse, c'est-à-dire, à partir de nos données contenus dans notre base de donnée, de créer un fichier XML publiable sur le WEB.

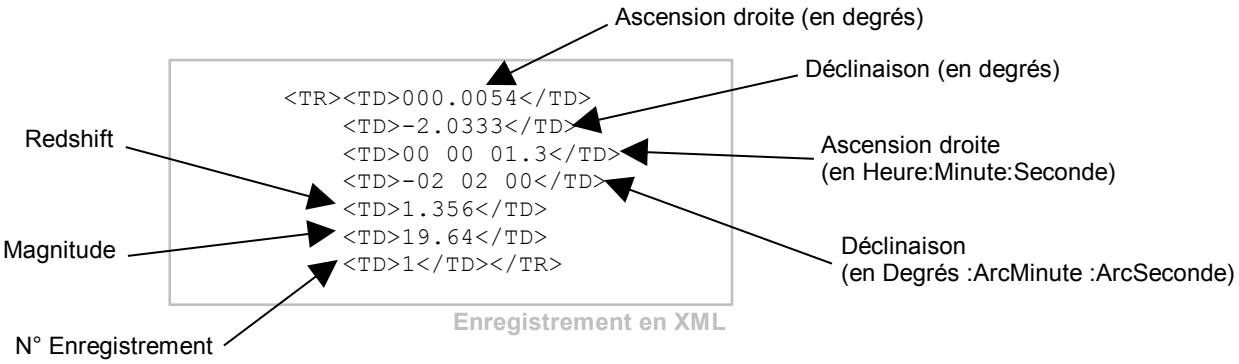


2) Optiques de programmation

Pour réaliser ce travail, nous avons décidé de réaliser deux fonctions principales, autour desquelles de nombreuses autres fonctions intermédiaires seront utilisées.
La première fonction permettra de lire les données contenues sur la page WEB, tandis que la seconde permettra de créer le fichier XML à partir de la base de données.
Autour des ces deux fonctions, de nombreuses autres fonctions de conversion ont été programmées afin de répondre aux exigences de l'utilisateur.

3) Détails d'un fichier XML

Nous allons voir dans cette partie le format des données dans un fichier XML afin de mieux présenter le travail réaliser. Voici sur le schéma suivant la façon dont est codé un enregistrement dans un fichier XML :



Il est à noté que les données sont utilisées pour la plupart en radians dans notre logiciel, il a donc fallu convertir les données dans cette unité pour les insérées dans la base de données et faire la conversion dans l'autre sens pour créer le fichier XML.

Descriptif des deux fonctions principales

1) Fonction *Read_XML_Quasar*
2) Fonction *Write_XML_Quasar*

```
Quasar* Read_XML_Quasar (char* NomFichier, int NBE)
```

Description :
La fonction *Read_XML_Quasar* permet à partir d'un fichier XML, dont le nom est passé en paramètre, de récupérer les données de ce même fichier telle que l'ascension droite, la déclinaison, le redshift et la magnitude, et de les insérer dans la structure Quasar qui est utilisée par les multiples fonctions du logiciel.

Paramètres :
char NomFichier* : Nom du fichier XML qui sera lu.
int NBE : Nombre d'enregistrements à lire dans le fichier XML.

```
void Write_XML_Quasar (Quasar* Q, char* NomFichier)
```

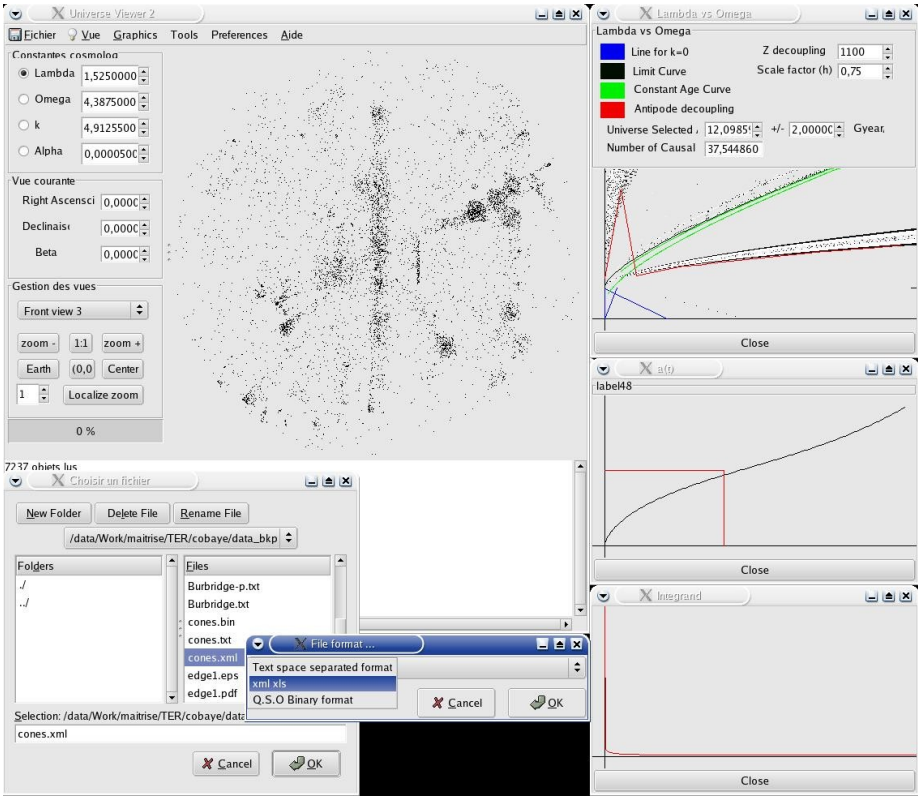
Description :
La fonction *Write_XML_Quasar* permet à partir de la structure de donnée passée en paramètre de créer le fichier XML associé.

Paramètres :
Quasar Q* : Nom du fichier XML qui sera lu.
char NomFichier* : Nombre d'enregistrements à lire dans le fichier XML.

IV Bilan

L'ensemble de l'application se montre fonctionnelle dans son nouvel habit en GTK. De nouvelles fonctions sont désormais disponibles pour faciliter le travail des astronomes à la recherche des grandes structures de notre univers.

Cependant il reste encore un long chemin à parcourir avant de pourvoir prétendre que l'aperçu donné par cette application reflète parfaitement le véritable aspect de notre univers. Par exemple, des effets comme les lentilles gravitationnelles prédites par Einstein restent à gérer.


Universe Viewer 2
Fichier Vue Graphics Tools Preferences Aide
Constantes cosmolog
Lambda 1,5250000
Omega 4,3875000
k 4,9125500
Alpha 0,0000500
Vue courante
Right Ascensci 0,0000
Declinaisc 0,0000
Beta 0,0000
Gestion des vues
Front view 3
zoom - 1:1 zoom +
Earth (0,0 Center
1 Localize zoom
0 %
7237 objets lus
Lambda vs Omega
Lambda vs Omega
Line for k=0
Limit Curve
Constant Age Curve
Antipode decoupling
Z decoupling 1100
Scale factor (h) 0,75
Universe Selected 12,0985
+/- 2,00000
Gyear.
Number of Causal 37,544860
Close
a(t)
label48
Close
Integrand
Close
Choisir un fichier
New Folder Delete File Rename File
/data/Work/maitrise/TER/cobaye/data_bkp
Folders
./
../
Files
Burbridge-p.txt
Burbridge.txt
cones.bin
cones.txt
cones.xml
edge1.eps
edge1.pdf
Selection: /data/Work/maitrise/TER/cobaye/data
cones.xml
Cancel
OK
File format ...
Text space separated format
xml xls
Q.S.O Binary format
Cancel
OK