Università degli Studi di Torino

Facoltà di Scienze Matematiche Fisiche e Naturali

Corso di Laurea in Informatica

Anno Accademico 2003/2004

Relazione di tirocinio:

Sviluppo di un planetario tridimensionale interattivo: DE ASTRIS

Relatore:
Prof. Nello BALOSSINO

Candidato:
Paolo PROTTO

Corelatore:
Prof. Maurizio LUCENTEFORTE

Corelatore esterno:
Ing. Fulvio DOMINICI CARNINO

INDICE GENERALE

1. INTRODUZIONE

Questo progetto sviluppato presso la Fondazione Ultramundum mira allo sviluppo delle Tabulae (elementi procedurali modulari) che consentano la rappresentazione delle entità celesti visibili dal nostro pianeta.
La Fondazione Ultramundum, con sede a Grugliasco, si occupa, tra le sue attività, dello sviluppo e diffusione di una nuova tecnologia software che consente di realizzare un nuovo tipo di televisione, denominato Ultravisione. Il presidente della Fondazione Fulvio Dominici ha avanzato la richiesta di brevetto di tale tecnologia, chiamata Ultrapeg, sia in Europa sia negli Stati Uniti. Il servizio dell'Ultravisione sarà erogato su Internet in tutto il pianeta, suddividendo i contenuti in canali numerati, molto simili a quelli della televisione classica. Ogni canale potrà veicolare film, documentari, ricostruzioni architettoniche o archeologiche, videogiochi, didattica o altro.
Lo stage da noi svolto presso la Fondazione ci ha permesso di sviluppare un modello di un planetario tridimensionale interattivo, denominato “De Astris”, che sarà scaricabile velocemente via Internet, a disposizione di tutti e integrabile nelle realizzazioni virtuali prodotte da terzi, a partire dai materiali pubblici della Fondazione stessa.
Il nostro impegno ha richiesto una fase iniziale di analisi per la stesura di un documento che indicasse gli standard attuali del settore, le strade e i protocolli per la realizzazione pratica del progetto e un tracciato dati da utilizzarsi per lo sviluppo. La fase successiva che ha portato alla realizzazione pratica del simulatore De Astris ha richiesto lo sviluppo di algoritmi inseriti nel contesto dell'ambiente già presente presso la Fondazione.
La peculiarità del progetto consiste nella scomposizione in moduli, secondo una metodologia euristica, del database monolitico (delle dimensioni di parecchi megabyte), contenente i dati delle stelle. A seguito di ciò, per la creazione del mondo virtuale non si è costretti ad entrare prima in possesso di tutti i dati; in-

fatti, essendo questi ultimi divisi in moduli, in ogni momento vengono scaricati solo quelli necessari secondo la posizione dell'osservatore nello spazio. La renderizzazione può iniziare subito rendendo il mondo navigabile all'utente, senza costringerlo a lunghe attese. Il livello di dettaglio viene raffinato nel tempo, con lo scaricamento di ulteriori dati, anche in rapporto alla potenza di calcolatore dell'utente.
Va detto che la piattaforma “De Astris”, essendo costituita da Tabulae, potrà evolvere nel tempo, integrando nuovi elementi o migliori rappresentazioni di quelli già inseriti.
Inoltre questo applicativo non risulta fine a se stesso, in quanto i moduli da noi creati possono essere sfruttati anche in altri contesti particolari quali ad esempio il riutilizzo in videogiochi e documentari.
I linguaggi utilizzati sono stati OpenGL e C++, mentre le piattaforme utilizzate sono state Linux e Windows.

1.1 L'Ultravisione

Il “De Astris” nasce all'interno di un progetto più ampio che la Fondazione Ultramundum sta sviluppando da qualche anno: l'Ultravisione.
L'Ultravisione è una grande innovazione nel panorama internazionale e rappresenta la televisione tridimensionale del futuro. La principale differenza con la televisione classica risiede nel fatto che i contenuti ultravisivi sono tridimensionali mentre quelli della televisione classica sono bidimensionali, oltre che nel mezzo trasmissivo che è Internet invece delle onde radio. Al di là della tecnologia di visualizzazione, infatti, fino ad oggi il problema principale è sempre stato quello della definizione dei contenuti da visualizzare.
Per vedere qualcosa in tre dimensioni, occorre riprendere la scena con qualche tipo di 'telecamera tridimensionale' che fino ad oggi non era stata inventata. La rivoluzionaria intuizione che sta alla base dell'Ultravisione è quella di memorizzare e trasmettere le scene tridimensionali in un nuovo formato brevettato, chiamato Ultrapeg.
La televisione tridimensionale sarà caratterizzata da diversi canali, attraverso i

quali utenti di tutto il mondo potranno trasmettere quello che desiderano.
Il canale principale sarà quello scientifico 4DGEA, che realizzerà in tre dimensioni il mondo fisico nel quale viviamo, a partire dalle galassie più lontane dell'universo fino al singolo atomo presente in un tombino di piazza San Carlo a Torino. Questo canale è caratterizzato dalla massima aderenza alla realtà, dal momento che l'impronta è rigorosamente scientifica.
Un altro canale, 4DGEA+, affianca 4DGEA, rendendolo più bello, più ricco e più spettacolare a scapito della precisione e dell'attendibilità scientifica di ciò che rappresenta. Tutto ciò che non ha riscontri scientifici nella realtà sarà creato proceduralmente, rendendo l'ambiente il più verosimile possibile. Sarà pertanto possibile fare una passeggiata su Marte o navigare tra gli atomi e le molecole di un pezzo di ferro, ottenendo una rappresentazione tridimensionale molto bella e spettacolare, seppure poco "scientifica".

1.2 Obiettivi del "De Astris"

Il nostro progetto, "De Astris", si colloca proprio all'interno dell'Ultravisione.
L'idea, infatti, è quella di realizzare un planetario virtuale tridimensionale. Esistono già molti software che implementano planetari utilizzando la grafica a tre dimensioni. Uno di questi, "Celestia", rappresenta uno dei software più evoluti, completi e all'avanguardia nel suo genere. Tuttavia, questi software sono caratterizzati da una particolarità non trascurabile: sono applicazioni fini a se stesse. Pertanto, esse possono essere utilizzate esclusivamente per navigare all'interno dello spazio, tra stelle, pianeti e galassie, ma non ad altri scopi. L'idea di "De Astris" è invece più innovativa e ambiziosa. Lo scopo di questo progetto è infatti quello di creare un ambiente tridimensionale come quello di "Celestia", con stelle e corpi celesti aderenti alla realtà, sfruttabile non solo per navigazioni spaziali, ma anche per qualsiasi altra applicazione. Si potranno pertanto creare videogiochi che utilizzano l'ambientazione del "De Astris" (e.g. uno sparatutto tra alieni nello spazio); film d'amore nei quali il cielo stellato che fa da sfondo a un bacio rispecchi esattamente il cielo del periodo nel quale la pellicola è ambientata; documentari sulla vita delle stelle o sulle orbite dei pianeti; eccetera... Ma l'impor-

tanza del progetto non si limita a questo. Il progetto è principalmente finalizzato alla fruibilità via Internet. Pertanto, per garantire una buona efficienza nella trasmissione in rete, gli elementi procedurali trattati dalla tecnologia devono essere di dimensioni ridotte e i dati al loro interno devono essere ottimizzati. Per questo motivo, la realizzazione del progetto è stata compiuta con un'attenzione particolare all'efficienza nella gestione dei dati; non sono mancati, infatti, durante tutto il periodo di implementazione, momenti di discussione e di confronto, per monitorare in ogni momento le scelte fatte e la loro ripercussione sull'efficienza del progetto. Per tutti i motivi sopra citati, l'idea alla base del “De Astris” assume connotati notevolmente interessanti.

1.3 Fasi del progetto

Il punto di partenza nella realizzazione di questo progetto è stata la fase di reverse engeenering realizzata sul programma open source “Celestia”. Questo periodo di studio e approfondimento ci è stato particolarmente utile per iniziare a familiarizzare con concetti astronomici e astrofisici indispensabili per l'elaborazione del progetto. In particolare, in questa fase ci siamo concentrati sul modo in cui Celestia prende i dati delle stelle e delle entità celesti da database di cataloghi per organizzarli in strutture dati, trattarli e renderizzarli nell'ambiente tridimensionale. Il reverse engeenering ci ha fornito indicazioni utili non solo in ambito informatico, per la rappresentazione tridimensionale delle stelle, ma anche da un punto di vista scientifico.

Per potenziare le nostre conoscenze approssimative e superficiali in campo astronomico, è seguita una fase di ricerca e di studio, nella quale ci siamo documentati sull'universo, sui suoi meccanismi, i suoi principi, le sue caratteristiche e su tutto ciò che si trova nell'universo (galassie, stelle, pianeti, satelliti...). Durante questa fase, abbiamo prodotto una documentazione di oltre 300 pagine, con la descrizione dettagliata di tutto ciò che abbiamo appreso in questo periodo. L'acquisizione di queste conoscenze, che non possedevamo in quanto studenti di Informatica, sono state indispensabili nel prosieguo del progetto.

Successivamente, è iniziata la fase implementativa. Utilizzando il software Ul-

traport, di proprietà della Fondazione, abbiamo cominciato a scrivere del codice e a discutere gli aspetti fondamentali da tenere in considerazione per l'elaborazione del progetto. Il primo applicativo utilizzato in ambiente Ultraport è stato l'Invector, un importatore di dati presenti su database per creare Tabulae, elementi procedurali che vengono in seguito renderizzati nello spazio tridimensionale. I database che abbiamo scelto di importare in Tabulae sono stati i cataloghi di stelle (Hipparcos e Tycho). Da questi cataloghi, si è discusso su quali fossero i dati utili da memorizzare e quali quelli calcolabili indirettamente e non indispensabili. La compressione dei dati delle stelle, infatti, rappresenta un elemento fondamentale nella tecnologia Ultrapeg.

Dopo aver realizzato le prime Tabulae di dati, che erano collezioni di informazioni di un certo numero di stelle, è seguita una prima fase di renderizzazione, per cominciare a disegnare nell'ambiente tridimensionale le prime stelle. Per far questo, abbiamo utilizzato l'applicativo Nucleus della Fondazione.

I risultati non particolarmente efficienti nella navigazione spaziale e nella rappresentazione delle stelle ci hanno portato a ridiscutere l'organizzazione dell'universo e la sua suddivisione spaziale, con l'obiettivo di ottimizzarla per renderla il più possibile efficiente.

Al termine di questa fase, compiuta “a tavolino”, abbiamo implementato la seconda versione dell'Invector, tenendo conto delle ricerche effettuate e delle decisioni prese in seguito a queste.

L'ultima parte della realizzazione del progetto ha riguardato l'implementazione finale e definitiva per ottenere il planetario navigabile dall'utente.

2. INQUADRAMENTO SCIENTIFICO

2.1 Galassie

Una galassia è un sistema stellare costituito da miliardi di stelle; in particolare, l'iniziale maiuscola indica quella galassia di cui fa parte il Sole: la Galassia o Via Lattea.
E' un sistema complesso, autogravitante (ossia tenuto insieme dalla forza gravitazionale generata dalla sua stessa massa), costituito da miliardi di stelle e gigantesche nubi di gas e polveri.

Figura 2.1: nell'immagine, la galassia NGC 3314, con i suoi bracci a spirale [Sito12].

Nel 1610, Galileo Galilei usò un telescopio per studiare la brillante banda presente nel cielo notturno, conosciuta come Via Lattea, e scoprì che era composta da un enorme numero di deboli stelle. In un trattato del 1755 Immanuel Kant fece l'ipotesi (corretta) che la galassia poteva essere un insieme rotante compo-

sto da un gran numero di stelle, tenute insieme dall'attrazione gravitazionale, simile al Sistema Solare ma su scala molto più grande. Il disco di stelle risultante sarebbe visibile come una banda nel cielo, dalla nostra prospettiva dentro il disco. Kant inoltre congetturò che alcune delle nebulose visibili nel cielo notturno fossero galassie separate.

2.1.1 Via Lattea

La nostra galassia, chiamata Via Lattea, ha un diametro di circa 100 mila anni luce e viene classificata come galassia a forma di spirale barrata, costituita da sei bracci. E' una galassia gigante e si stima contenga oltre 100 miliardi di stelle.
Una di queste è il nostro Sole, situato nel Braccio di Orione a una distanza di circa 27.700 anni luce dal centro galattico. Il Sole ha un periodo di rivoluzione all'interno della galassia di 225 milioni di anni.

2.2 Stelle

All'interno della Via Lattea, le stelle conosciute e osservabili dalla Terra costituiscono una piccola parte di quelle presenti in tutta la Galassia.

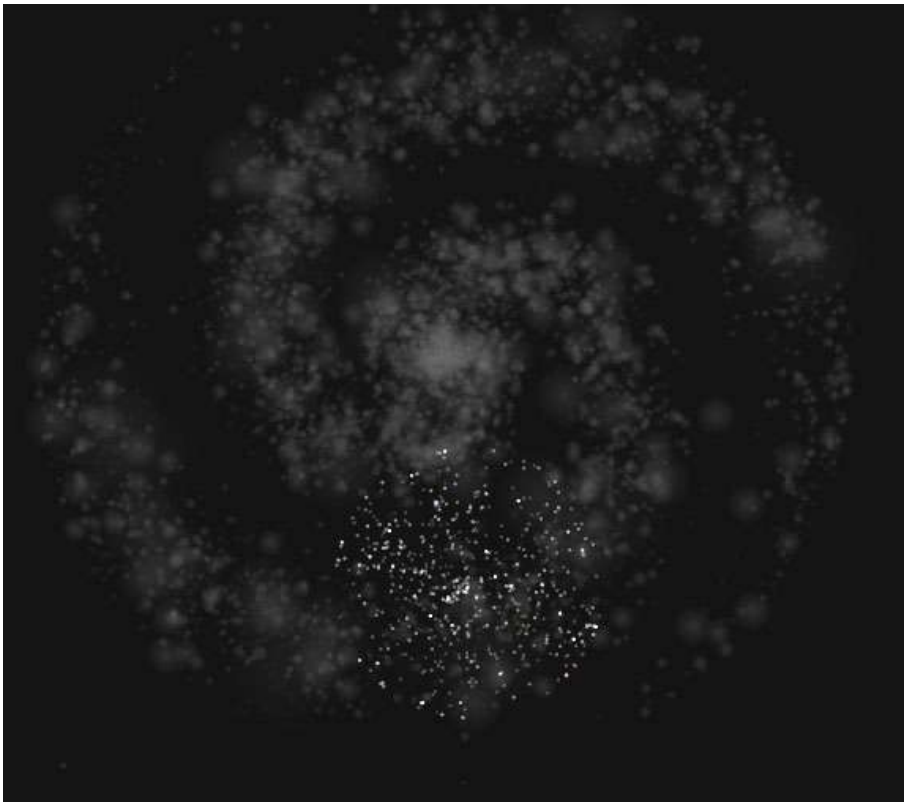


Figura 2.2: immagine presa da Celestia. In basso sono disegnate le stelle del catalogo Tycho.

Nei tempi passati, il limite all'osservazione è stato l'occhio umano; limite che si è spinto molto più in là con l'invenzione dei telescopi, dai primi di Galileo fino ai più potenti oggi a disposizione della scienza. L'osservazione dalla Terra ha sempre avuto la limitazione al solo campo visivo a causa dell'atmosfera e solo negli ultimi tempi, grazie ai satelliti, è stato possibile andare oltre e collocare il punto di osservazione al di fuori della sfera terrestre, procedendo così a rilevamenti anche ad altre frequenze (e.g. Infrarosso).
Nelle figure 2.2 e 2.3 è ben rappresentato il limite all'osservazione. La Via Lattea è disegnata come un alone grigio (in Figura 2.2 si può notare la sua forma a spirale). Le stelle conosciute ed osservate sono disegnate come puntini: si nota

chiaramente che la zona che occupano è relativamente piccola rispetto alla galassia. Il Sole si trova al centro di queste stelle.
Una stella può essere definita come un'enorme sfera autogravitante di gas caldissimo (principalmente idrogeno ed elio), che produce energia attraverso un processo di fusione nucleare e la riemette sotto forma di radiazione.
Le stelle si trovano a distanze immense dal nostro sistema solare, così, nonostante le loro dimensioni ci appaiono come piccoli puntini luminosi nel cielo.

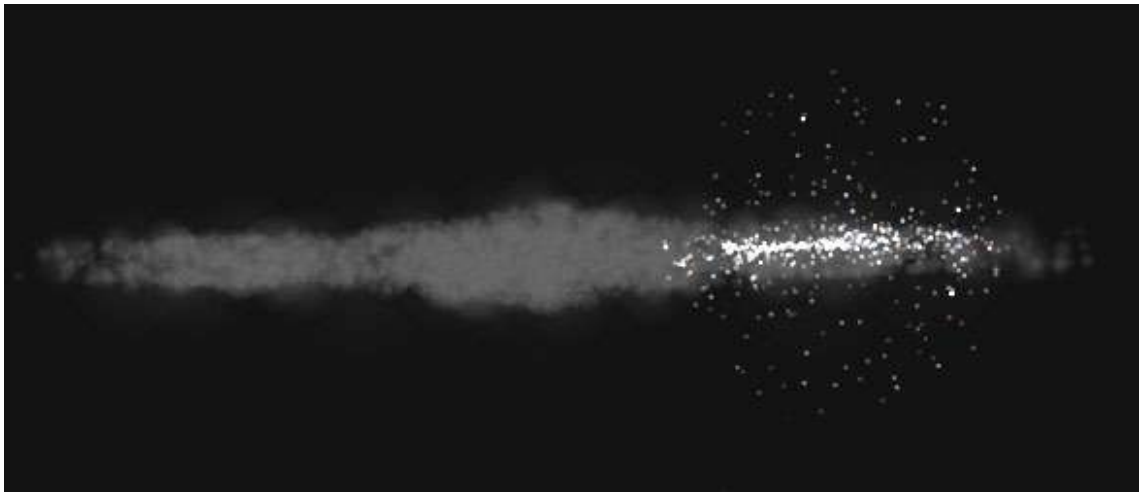


Figura 2.3: immagine presa da Celestia. A destra sono visibili le stelle del catalogo Tycho.

Le stelle ci appaiono sulla sfera celeste raggruppate in insiemi, detti costellazioni. A molte stelle gli astronomi hanno attribuito nomi propri, per lo più di origine greca, araba o latina. Altre vengono classificate con il nome della costellazione a cui appartengono e una lettera dell'alfabeto greco.
Vengono raccolte in cataloghi, da questi ne prendono il nome. I più moderni cataloghi, compilati con l'aiuto delle osservazioni di satelliti artificiali, contengono anche milioni di stelle.

2.2.1 Massa stellare

La massa di una stella può variare da circa un decimo a circa 100 volte la massa del Sole. Le dimensioni variano invece in un intervallo più ampio; il diametro di una stella è sempre piuttosto difficile da determinare, e può essere misurato solo per stelle vicine. Esso può variare da pochi Km per una nana bianca a cento mi-

lioni di Km per una supergigante rossa.

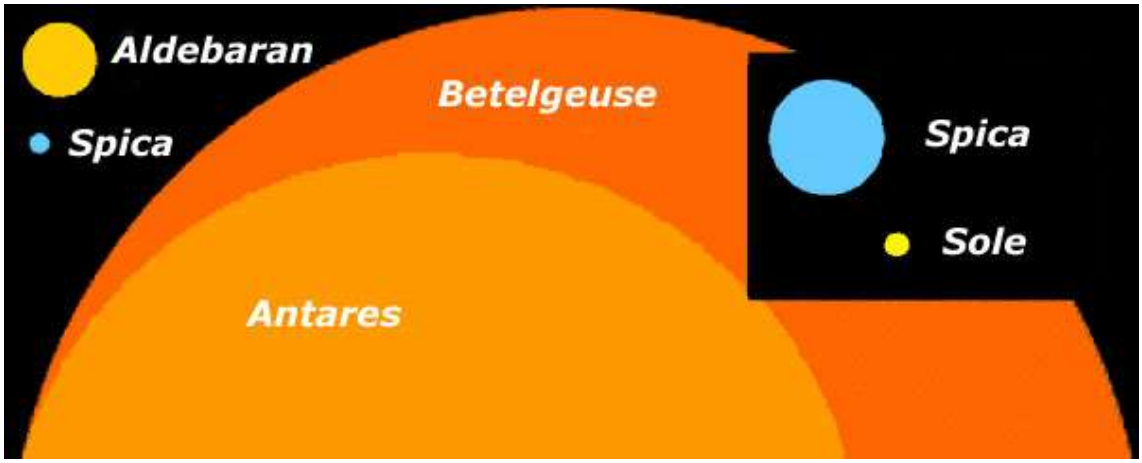


Figura 2.4: confronto tra le dimensioni di alcune stelle, tra cui il Sole.

2.2.2 Classi spettrali

Quando un fascio di luce bianca viene fatto passare attraverso un prisma o un altro mezzo dispersivo, viene scomposta nelle varie lunghezze d'onda che lo costituiscono, formando una striscia colorata: essa prende il nome di spettro della sorgente di luce.

Lo spettro della luce fornisce molte informazioni sulla composizione chimica della sorgente e sul suo stato fisico (temperatura, densità e grado di ionizzazione).

In astrofisica una stella viene caratterizzata da un "colore" e da una "temperatura superficiale" a seconda della forma del suo spettro.

Il colore è determinato dalla regione dello spettro nella quale l'intensità della luce è massima; le stelle hanno temperature superficiali di qualche migliaio o poche decine di migliaia di gradi, ed emettono la massima potenza nella regione ottica dello spettro. Il Sole emette al massimo di intensità nella regione gialla della banda ottica, perciò la sua temperatura superficiale è stata stabilita in 5780 gradi Kelvin.

Le classi spettrali sono: O, B, A, F, G, K, M, S, R, N. Si differenziano in base alla temperatura decrescente da superiore a 30 mila gradi Kelvin, corrispondente alla classe O, fino a 2-3 mila gradi Kelvin, delle classi M, S, R e N.

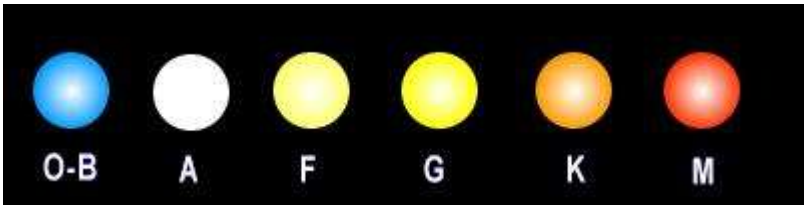


Figura 2.5: classi spettrali e relativi colori.

Ognuno di questi tipi spettrali è a sua volta suddiviso in sottoclassi, contrassegnate con numeri da 0 a 9 (per esempio il Sole è una stella di tipo spettrale G5).
A parità di temperatura superficiale e quindi di colore, le stelle possono avere una diversa luminosità. Gli astronomi hanno quindi introdotto anche alcune classi di luminosità per catalogarle.

2.2.3 Magnitudine

Con il termine magnitudine si intende la misura della quantità di luce che ci arriva da un corpo celeste. Questa quantità di luce dipende da molti fattori come la distanza dell'astro in questione, la sua grandezza, la sua temperatura, ecc.

Magnitudine apparente

Sappiamo che vi sono stelle che appaiono più luminose di altre. Per poter confrontare le stelle in base alla propria luminosità si deve utilizzare una scala. Salvo piccole estensioni, la scala che si utilizza oggi è la stessa introdotta dai Greci: si attribuisce il valore di magnitudine apparente 1 alla stella che in cielo appare più luminosa e 6 a quella più debole visibile ad occhio nudo. Con l'introduzione dei telescopi, che hanno permesso l'osservazione di stelle la cui luminosità è molto più debole, si sono aggiunti nuovi livelli alla scala di riferimento.
La dicitura “apparente” è dovuta al fatto che ci si riferisce alle luminosità delle stelle così come appaiono viste dalla superficie terrestre. Per questo motivo, è stata creata una scala non geocentrica, ricavata da una semplice trasformazione: la magnitudine assoluta.

Magnitudine assoluta

E' la misura della luminosità che avrebbero gli astri se fossero tutti alla distanza di 10 Parsec dalla Terra.
La relazione che lega la magnitudine apparente (m) a quella assoluta (M) è:
$M = m + 5 - 5 * \log(d)$
dove d è la distanza della stella in Parsec. Da questa relazione è possibile, conoscendo la distanza di una stella, determinare la sua magnitudine assoluta; viceversa, a partire dalla magnitudine assoluta si può ricavare la distanza.

Magnitudine bolometrica

Le magnitudini bolometriche tengono conto delle varie lunghezze d'onda, a differenza di quelle assolute che si basano solo sullo spettro visivo.
Ogni astro è caratterizzato dal suo indice di colore (B-V), un parametro che esprime la differenza tra la magnitudine bolometrica e visuale, indicando così il colore della stella.

2.2.4 Moti stellari

Anticamente si pensava che le stelle fossero fisse sulla volta celeste, mentre in realtà esse si spostano relativamente a noi, in conseguenza sia del loro moto di rotazione attorno al centro della nostra galassia, sia del moto stesso del Sole (e quindi del Sistema Solare).
La Galassia infatti si presenta come un corpo in rotazione: il nucleo ruota come un corpo solido (in cui la velocità angolare cresce al crescere della distanza); dal nucleo fino ad una distanza di poco inferiore a quella solare, la velocità resta costante intorno ai 220 km/s, mentre in seguito più ci si allontana dal centro, più la velocità diminuisce. Le stelle possiedono inoltre, come i pianeti, un moto di rotazione intorno al proprio asse.

2.2.5 Ammassi stellari

Le stelle si presentano spesso a gruppi detti ammassi stellari. Vi sono ammassi aperti, ammassi globulari e associazioni stellari. Queste stelle vengono tenute

insieme dalla forza di attrazione gravitazionale esercitata dalle une sulle altre. Un esempio di ammasso aperto visibile a occhio nudo è il gruppo delle Pleiadi.

Figura 2.6: l'ammasso aperto delle Pleiadi.

2.3 Coordinate stellari

2.3.1 Sistema equatoriale

E' un sistema di coordinate astronomiche che ha come direzione fondamentale l'asse del mondo (NS) e come piano fondamentale quello dell'equatore celeste. I poli del sistema sono i poli celesti nord e sud. I cerchi ausiliari si chiamano cerchi orari o meridiani celesti. I cerchi minori, paralleli all'equatore, si chiamano paralleli celesti. (immagine)

Le stelle sono individuate sulla sfera celeste da due grandezze che ne forniscono la posizione: l'ascensione retta e la declinazione. (Questo sistema di coordinate è piatto in quanto è una proiezione delle stelle sulla volta celeste, non dice nulla sulla loro distanza.)

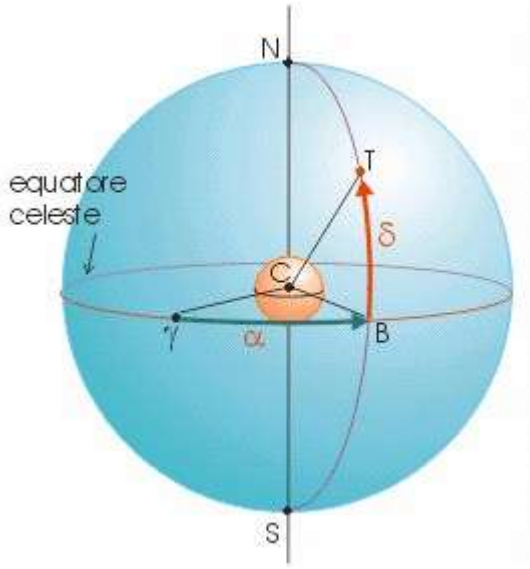


Figura 2.7: il sistema equatoriale.

Ascensione retta α *(RA)*: è l'ascissa sferica del sistema equatoriale. Essa rappresenta la distanza angolare tra il punto gamma γ (equinozio di primavera) e il meridiano celeste che passa per quel punto, misurata lungo l'equatore celeste. L'ascensione retta si misura in direzione est dal punto dell'equinozio, in ore da 0 a 24; ogni ora viene suddivisa in 60 minuti e ogni minuto in 60 secondi.

Declinazione δ *(Dec.)*: è l'ordinata sferica di questo sistema. Rappresenta la distanza angolare tra un punto della sfera celeste (il punto T) e l'equatore, misurata lungo il meridiano celeste che passa per tale punto. Si misura in gradi e frazio-

ni di grado con segno positivo verso il polo nord celeste e negativo verso il polo sud.

Siccome la Terra ruota, le coordinate si riferiscono alla mezzanotte del 21 marzo, cioè all'equinozio di primavera, il momento in cui l'eclittica del Sole interseca l'equatore celeste. Nella Figura 2.8 il punto gamma γ indica l'equinozio di primavera, il punto omega Ω l'equinozio d'autunno.

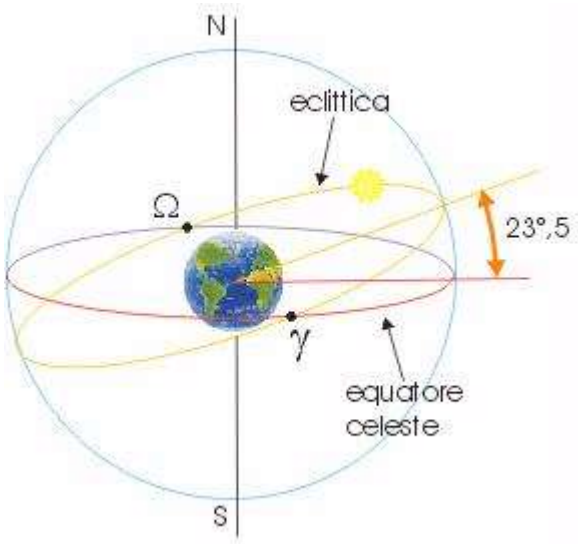


Figura 2.8: differenza di inclinazione tra l'equatore celeste (e terrestre) e l'eclittica del sole.

A causa dei moti delle stelle e della precessione della Terra, queste coordinate vengono calcolate ogni 50 anni. L'ultimo aggiornamento corrisponde all'anno 2000, quello precedente al 1950.

2.3.2 Sistema eclitticale

E' un sistema di coordinate astronomiche in cui il piano e la direzione fondamentale sono rispettivamente il piano dell'eclittica e la sua perpendicolare la quale individua, sulla sfera celeste, i poli dell'eclittica (boreale Pe con declinazione positiva e australe Pe' con declinazione negativa).

Le coordinate eclitticali sono:

Longitudine celeste o eclitticale (λ) di un punto (T) è la distanza angolare tra il punto gamma e il cerchio ausiliario che passa per quel punto; si misura da 0° a 360° lungo l'eclittica partendo dal punto gamma e procedendo in senso antiora-

rio (angolo γ OB).

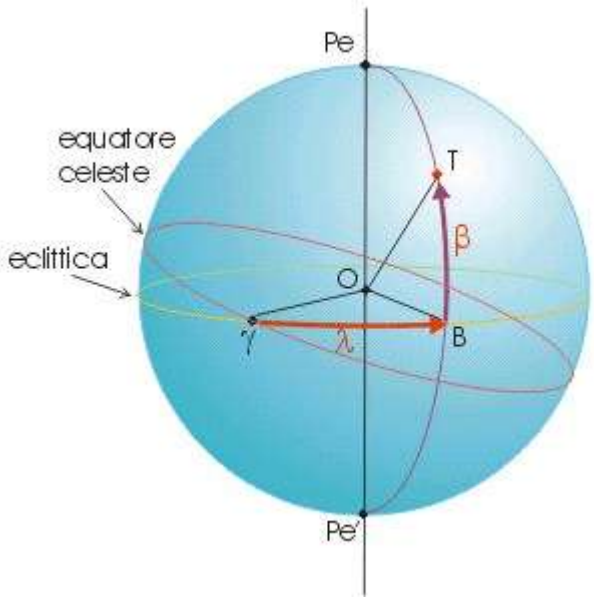


Figura 2.9: il sistema eclitticale.

Latitudine celeste o eclitticale (β) di un punto è la distanza angolare tra il punto (T) e il piano dell'eclittica, misurata lungo il cerchio ausiliario che passa per tale punto. Ha valore positivo (da 0° a +90°) nell'emisfero nord dell'eclittica e negativo (da 0° a -90°) in quello sud.
Il sistema eclitticale è importante per lo studio dei moti planetari. Tradizionalmente le dodici costellazioni attraversate dall'eclittica vengono chiamate fascia degli animali o zodiaco. Fin dall'antichità la fascia dello zodiaco veniva divisa in dodici parti uguali, a partire dal punto gamma.

2.3.3 Misura delle distanze

Le distanze delle stelle vengono misurate tramite la parallasse. Questa è il moto apparente di un oggetto relativamente vicino, rispetto ad altri molto più lontani, in seguito al moto dell'osservatore.
In altre parole, è l'angolo sotto cui viene osservato un oggetto lontano da due punti diversi, separati da una distanza nota. Conoscendo questo angolo si può ricavare la distanza dell'oggetto.

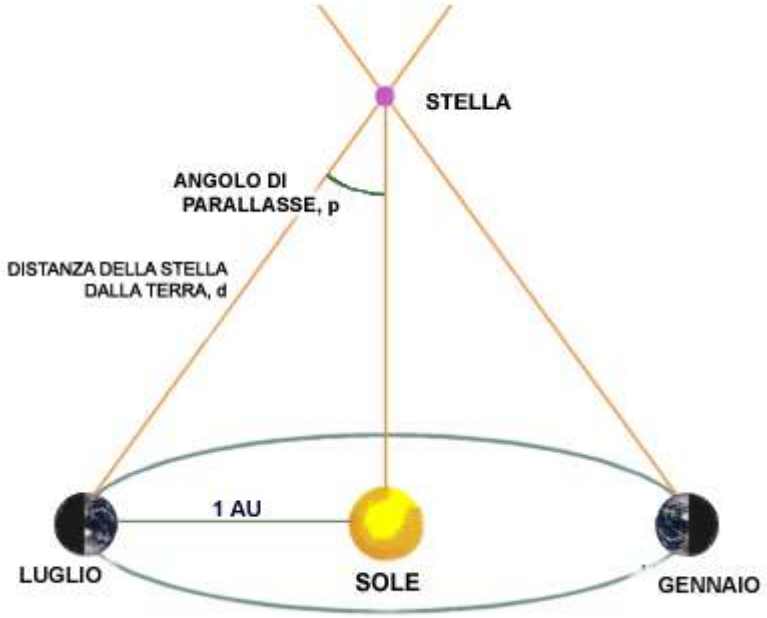


Figura 2.10: lo schema illustra il metodo con cui viene calcolata la parallasse

In astronomia, la parallasse corrisponde a metà dell'angolo di cui si è spostata una stella nel cielo, nel tempo in cui la Terra ha compiuto mezza orbita intorno al Sole.

2.3.4 Unità di misura delle distanze astronomiche

Per esprimere le enormi distanze tra i corpi celesti, sono state introdotte in Astronomia unità di misura specifiche.

Unità astronomica (U.A.): è usata in genere entro i limiti del Sistema Solare e corrisponde alla distanze media tra Terra e Sole, che è di circa 149.600.000 km.

Anno-luce (ly): è la distanza percorsa in un anno dalla luce, la quale si muove alla velocità di circa 300.000 km/s. Un anno-luce corrisponde, quindi, a una distanza di 9.463 miliardi di km.

Parsec (pc): significa Parallasse per secondo. Un parsec è la distanza di un punto dal quale un osservatore vedrebbe il semiasse maggiore dell'orbita terrestre, perpendicolarmente, sotto l'angolo di un secondo; tale distanza equivale a 3,26167 anni luce, e corrisponde perciò a 30.900 miliardi di km.

2.4 Cataloghi di stelle

La posizione delle stelle, definite dalle coordinate, ha reso possibile ordinarle in raccolte, dette cataloghi, nei quali sono definite anche le magnitudini, i tipi spettrali, ecc. Il primo elenco sistematico di stelle fu compilato da Ipparco (190 - 120 a.C.) nel 134 a.C.; in esso vi erano elencate 850 stelle fra le più luminose.
Nel suo catalogo Ipparco classificò ogni stella calcolandone la posizione, in base ad un sistema di longitudine e latitudine celesti, e ne stimò la luminosità secondo una scala di 'grandezze' o 'magnitudini' da lui stesso inventata.
Il catalogo di Ipparco venne ampliato tre secoli più tardi da Tolomeo (100 - 170 d.C.), l'ultimo grande astronomo greco antico, e incluso nella sua monumentale opera, l'Almagesto, che, a differenza delle opere di Ipparco andate perdute, è giunta fino a noi. Il catalogo tolemaico fornisce le coordinate eclittiche (latitudine e longitudine) delle stelle, a differenza dei cataloghi stellari moderni che riportano le coordinate equatoriali (ascensione retta e declinazione).

2.4.1 Hipparcos e Tycho

La missione Hipparcos (HIgh-Precision PARallax COllecting Satellite) è stata la prima missione spaziale dedicata all'Astrometria [Sito4]. Il satellite è stato ideato e costruito, sotto la supervisione dell'ESA. A bordo del satellite era installato un piccolo telescopio (di 29 cm di diametro) che aveva il compito di scansionare più volte la sfera celeste al fine di ottenere le posizioni, le parallassi (cioè le distanze) ed i moti propri di circa 120.000 stelle aventi una magnitudine minore della 13 e con una precisione da 2 a 4 milli-arcosecondi.
Hipparcos iniziò la sua missione scientifica il 26 Novembre 1989 e lavorò, anche se in maniera non continuativa, fino al 15 Agosto 1993.
Il programma di lavoro era diviso in due parti: l' esperimento Hipparcos, il cui obiettivo era di misurare i parametri astrometrici di circa 120.000 stelle con una precisione da 2 a 4 milli-arcosecondi (cioè con un'accuratezza paragonabile alla dimensione angolare della stellina metallica posta in cima alla Mole Antonelliana di Torino vista dalla Luna) e l' esperimento Tycho, la misura delle proprietà astrometriche e di fotometria in due colori di stelle ad una precisione legger-

mente inferiore.
Il Catalogo Hipparcos finale (120.000 stelle con risoluzione di 1 milliarcsec) e il Catalogo Tycho finale (più di due milioni di stelle con risoluzione di 20-30 milliarcsec e fotometria a 2 colori) furono completati nell'agosto del 1996, e pubblicati dall'ESA nel giugno del 1997.

2.4.2 Breve descrizione dei campi del catalogo Hipparcos

Il catalogo Hipparcos è molto dettagliato e contiene un totale di 78 campi [Sito5]. Di seguito, verranno passati brevemente in rassegna i dati più importanti che si possono ritrovare all'interno del catalogo (in particolare, l'attenzione è rivolta a quei campi che sono stati utilizzati per il progetto sviluppato).
I primi 2 campi del catalogo contengono informazioni generiche sui corpi celesti in questione. In particolare, il primo campo (H0) rappresenta il tipo di catalogo dal quale l'oggetto è stato derivato (H = Hipparcos, T = Tycho), mentre il secondo (H1) fornisce l'identificativo dell'oggetto, ossia il suo numero HIP.
I campi che vanno da H3 a H7 sono descrittori degli oggetti del catalogo. Le informazioni che si possono ritrovare sono l'indicatore dell'ascensione retta, espressa in h m s (campo H3), quello della declinazione, espressa in ° ' “ (gradi, primi, secondi) (campo H4) , la magnitudine V secondo il sistema fotometrico UBV di Johnson (H5), la variabilità o meno della magnitudine dell'oggetto (H6) e la sorgente tramite la quale è stato possibile derivare la magnitudine V (H7).
I campi successivi contengono i dati astrometrici degli oggetti: le coordinate equatoriali (ascensione retta e declinazione espresse in gradi), la parallasse trigonometrica, gli errori relativi ai vari dati (alle coordinate equatoriali, alla parallasse...) e altri campi che contengono coefficienti di correlazione tra dati diversi.
Infine i restanti campi, molto complessi e specifici, hanno a che fare con aspetti di fotometria e con i sistemi di stelle doppie o multiple.

2.5 Cenni sul sistema solare

2.5.1 Il Sole

Il Sole è una stella di medie dimensioni appartenente al gruppo spettrale G2V. Ha un diametro di 109 volte quello terrestre ed una massa ed un volume pari rispettivamente a 333000 e 1304000 volte quelli della Terra. La gravità è invece 28 volte maggiore di quella presente sul nostro pianeta.
Il Sole possiede anch'esso, come i pianeti, un moto di rotazione intorno al proprio asse, inclinato di 7° 15' sul piano dell'eclittica, con velocità angolare variabile secondo la latitudine; non ruota rigidamente ma presenta una rotazione differenziale, cioè più lenta ai poli e più veloce all'equatore.
All'equatore, il periodo di rotazione è di circa 25 giorni terrestri, mentre ai poli di 34 giorni.

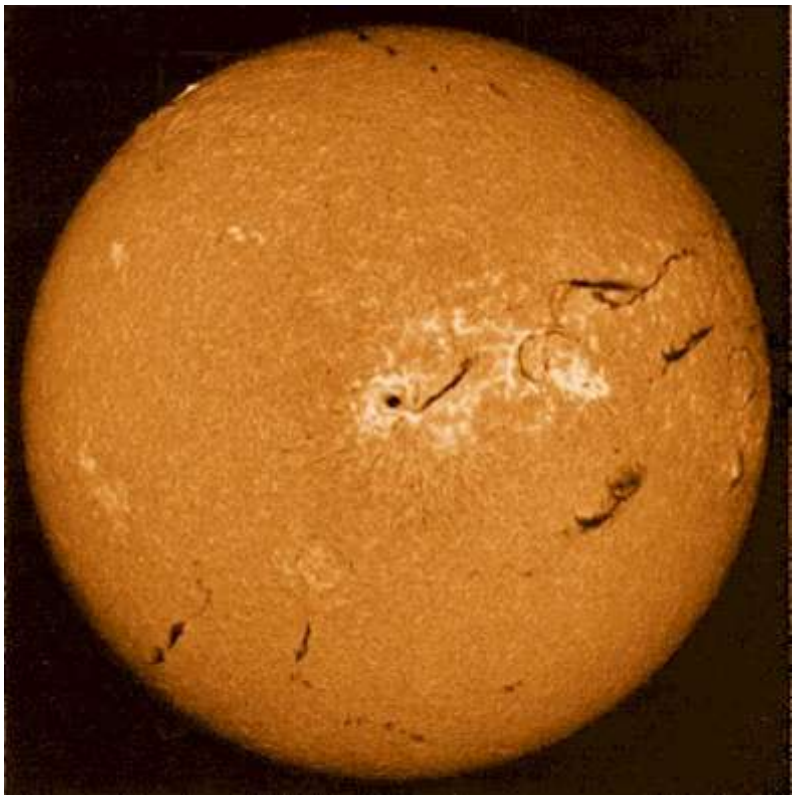


Figura 2.11: il nostro Sole. Nell'immagine sono visibili le macchie solari.

Ciò dimostra, non essendo uguale la durata di rotazione di tutti i punti, che il Sole almeno in superficie non è solido e che la velocità di rotazione del Sole, ol-

tre che aumentare verso l'equatore, aumenta anche con l'altezza dei diversi strati della sua atmosfera; però nelle regioni molto alte la velocità non è uniforme.

2.5.2 I pianeti

Un pianeta è un corpo celeste che non brilla di luce propria e ruota intorno a una stella che lo illumina di luce riflessa. I pianeti del sistema solare, in ordine di distanza dal Sole, sono: Mercurio, Venere, Terra, Marte, Giove, Saturno, Urano, Nettuno e Plutone; percorrono orbite ellittiche intorno alla nostra stella e sono visibili in quanto ne riflettono la luce.
Spesso i pianeti sono accompagnati anche da altri piccoli corpi, definiti satelliti o lune, che ruotano attorno al compagno principale secondo orbite ellittiche e con il quale costituiscono un unico sistema orbitante attorno al Sole.

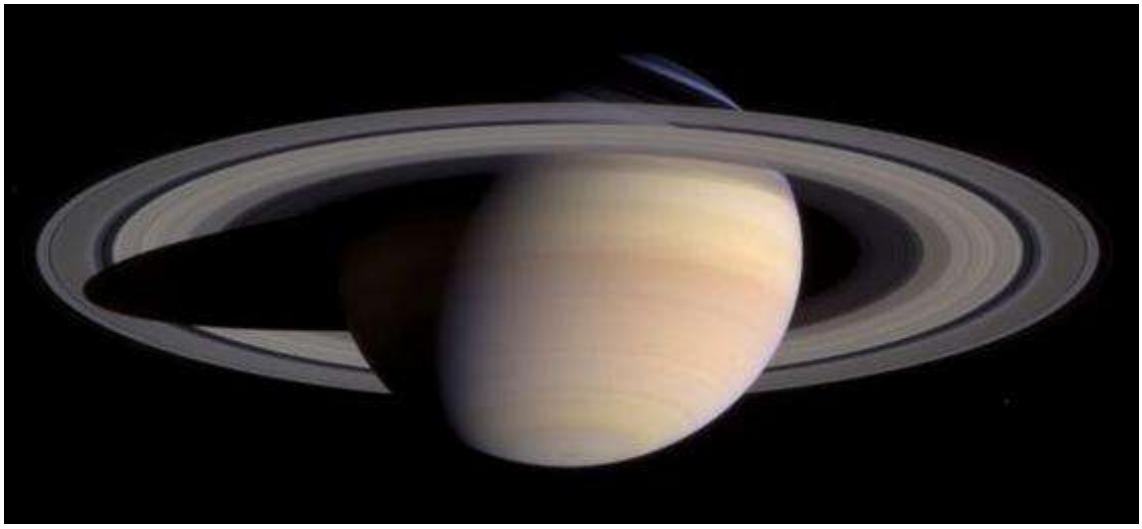


Figura 2.12: immagine di Saturno scattata dalla sonda Cassini nel settembre 2004.

Ogni pianeta, oltre a ruotare attorno al proprio asse, più o meno velocemente in senso antiorario, ad eccezione di Venere ed Urano che girano in senso contrario, compie un movimento di rivoluzione attorno al Sole secondo un'orbita di forma ellittica che, vista dal Nord del sistema solare, si svolge in un senso antiorario, per convenzione definito diretto.

2.5.3 Le orbite

Leggi di Keplero

Grazie a G. Keplero, oggi conosciamo i princìpi che regolano i moti dei pianeti che, sintetizzati nelle sue tre leggi affermano quanto segue:

1° - ogni pianeta percorre un'orbita ellittica, di cui il Sole occupa uno dei due fuochi;

2° - le aree descritte dalla retta Sole - pianeta sono proporzionali ai tempi impiegati a descriverle;

3° - il quadrato del periodo di rivoluzione è proporzionale al cubo del semiasse maggiore.

Dalla prima si ricava che la distanza dal Sole varia fra un massimo all'afelio, il punto più lontano, ad un minimo, al perielio, il punto più vicino; dalla seconda che la velocità orbitale di ogni pianeta è più alta al perielio e più bassa all'afelio; dalla terza invece che il tempo impiegato per percorrere un'orbita completa è direttamente proporzionale alla distanza media dal Sole.

VSOP87

La Teoria delle Variazioni Secolari VSOP87 consente di calcolare le effemeridi di 8 pianeti del sistema solare con precisioni che nel periodo 1900-2100 variano da 0.001" (Mercurio) a 0.100" (Saturno). I semplici algoritmi di calcolo permettono anche di determinare i vettori velocità e quindi l'evoluzione secolare degli elementi orbitali classici senza far ricorso all'integrazione numerica.

La teoria planetaria VSOP87 è stata sviluppata dal Bureau des Longitudes (BdL) di Parigi nel 1988 e fa riferimento a tutti i pianeti del sistema solare con l'esclusione di Plutone.

Il moto perturbato degli n-corpi viene risolto mediante l'utilizzo delle serie polinomiali di Poisson, costituite da un numero elevatissimo di termini periodici che si adattano egregiamente all'elaborazione elettronica.

3. LA TECNOLOGIA ULTRAPEG

Ultrapeg è una tecnologia di codifica e trasmissione dati per mondi digitali multidimensionali. Ultrapeg consente una elevata velocità di scaricamento e visualizzazione di ambienti digitali di qualunque estensione anche con connessioni di rete a bassa velocità.

3.1 Come funziona Ultrapeg

La sostanziale differenza rispetto alle tecniche normalmente usate per la realizzazione di ambienti tridimensionali interattivi risiede nella memorizzazione dei dati come concetti, invece che come collezioni di informazioni digitali.
Fino ad oggi la realizzazione di ambienti tridimensionali ha richiesto la fase di modellazione, nella quale artisti esperti di computergrafica letteralmente scolpivano ogni oggetto che si rendeva necessario. Tale tecnica è sempre stata come lavorare un blocco di creta grezza e trarne i modelli di volta in volta necessari.
Con Ultrapeg si passa alla definizione di una gigantesca 'scatola di costruzioni' dalla quale prendere ogni ‘mattoncino' che necessita per la realizzazione della scena. In tal modo l'autore non è costretto a scolpire ogni oggetto dell'ambiente ma lo può prendere dalla 'scatola di costruzioni'. Ogni oggetto eventualmente non presente dovrà essere realizzato ma sarà poi disponibile nella raccolta di base per chiunque altro ne necessiti in futuro; Ultrapeg si basa infatti su di una filosofia collaborativa. In tal modo si memorizza (e poi trasmette su Internet) solo l'elenco dei numeri di serie degli elementi della scena, non tutti i loro dati. Con questo sistema la dimensione dei dati di un ambiente tridimensionale si riduce enormemente, rendendone possibile la trasmissione in tempo reale su Internet. Tra i mattoncini disponibili si trovano non solo i modelli tridimensionali, ma anche tutti gli altri elementi necessari per la produzione di un titolo multi-

mediale come musiche di sottofondo, animazioni, descrizioni testuali degli elementi e così via.
La fase di sviluppo di nuovi contenuti è estremamente semplice: basta scegliere dalla raccolta di oggetti i singoli elementi che interessano e disporli sulla scena. Si crea così un elenco che a tutti gli effetti è un ambiente tridimensionale completo scaricabile da Internet da chiunque. I mattoncini elementari messi a disposizione da Ultrapeg possono essere singoli elementi o aggregati molto complessi. Si possono prendere e usare interi edifici (come la Mole Antonelliana o Palazzo Madama), loro parti (come porticati, arcate, portoni) o singoli elementi (come fregi, capitelli, semplici mattoni). Ogni elemento in Ultrapeg è 'intelligente', cioè in grado di adattarsi all'utilizzo. Non essendo una collezione fissa di dati ma bensì un programma vero e proprio, un porticato può, ad esempio, allungarsi o ridursi in modo da adattarsi al particolare uso che se ne vuol fare.
Un ambiente, essendo una serie di riferimenti agli elementi della raccolta di base, può migliorare nel tempo in modo automatico. Se un autore, infatti, decide di usare la Mole Antonelliana in un certo punto, memorizza il numero di serie dell'elemento e non i suoi dati effettivi. Se in un secondo momento il 'mattoncino' della Mole viene migliorato, automaticamente si migliorano anche tutti i canali che ad esso fanno riferimento, senza che gli autori debbano fare nulla.
Grazie a queste ed altre caratteristiche, Ultrapeg permetterà la creazione della vera televisione tridimensionale (ultravisione), nella quale gli utenti potranno seguire programmi simili a quelli attuali ma nei quali sarà possibile 'entrare' nei contenuti. L'Ultravisione permetterà inoltre il video-on-demand, cioè la fruizione di un qualsiasi titolo in qualunque momento, senza attendere che venga trasmesso. Ogni prodotto, infatti, sarà scaricabile e visualizzabile a richiesta in ogni momento, proprio come le pagine del World Wide Web.

3.2 Standard Dati

3.2.1 Le Tabulae

I dati in UltraPeg comprendono una molteplicità di categorie, per ognuna delle quali sono necessarie specifiche tipologie. Tutti i dati dei mondi di Ultramundum sono contenuti in Tabulae, le cui struttura ed evoluzione sono estremamente complesse.

Una Tabula è un'entità rappresentata da un blocco di dati binari presente sull'UltraServer che può essere richiesta dall'UltraPort e poi memorizzata su di un database locale onde evitare successive richieste.

Una Tabula è univocamente indicata dal proprio numero di serie, la cui dimensione in byte può crescere nel tempo onde accomodare un numero sempre crescente di Tabulae.

Ogni Tabula può contenere tre porzioni: Interfaccia, Procedurale e Dati.

L'Interfaccia contiene i dati necessari a definire come la Tabula si relaziona con il mondo esterno.

La porzione Procedurale, che può anche non essere presente per Tabulae di soli dati, contiene tutto il codice, scritto in linguaggio Carmina (nota), che consente alla Tabula di eseguire materialmente le proprie operazioni.

La parte Dati contiene tutte le informazioni che sono necessarie alla Tabula.

Le Tabulae sono gerarchiche: consentono di referenziare altre Tabulae per gli elementi inferiori. In tal modo, una Tabula che definisce un certo tipo di automobile può referenziare una Tabula per le ruote. La stessa Tabula viene referenziata da tutte le altre che contengono ruote al loro interno.

Le Tabulae sono astrazioni concettuali e inglobano archetipi di intere categorie di entità, potendo esprimere ogni singolo elemento specifico della categoria di cui si occupano mediante l'opportuna impostazione dei parametri passati all'Interfaccia.

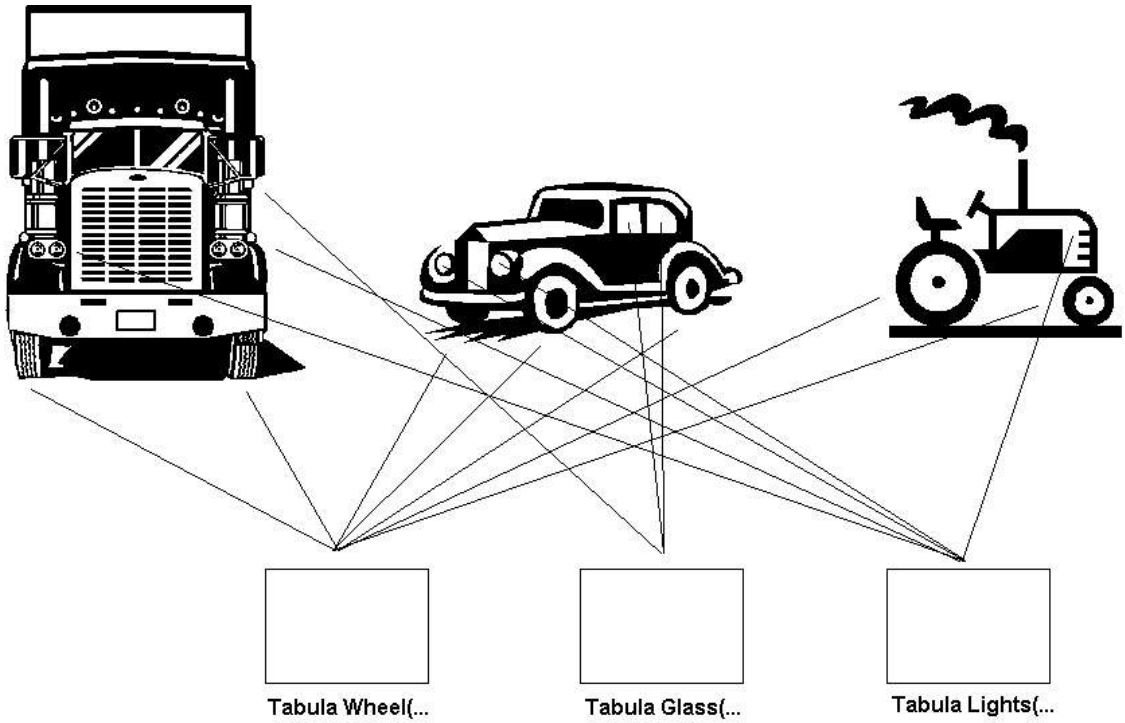


Figura 3.1: illustra la possibilità di riutilizzo delle Tabulae all'interno di altre che descrivono oggetti più complessi. In questo semplice esempio la Tabula ruota (wheel) viene richiamata nella Tabula camion, in quella auto e in quella trattore.

Ad esempio, la tabula Ferrari 550 consente di creare tutte le possibili copie di questa specifica automobile indicando all'interfaccia parametri come Colore, Tipo di cerchioni, Stato di usura e così via. Grande attenzione deve essere posta nella stesura delle Tabulae in modo da catturare al più alto livello possibile di astrazione la categoria che si va a definire.

3.2.2 Le Cellulae

Ogni Cellula rappresenta:

- un sistema di riferimento quadri-dimensionale, nel quale le informazioni che lo definiscono sono:
  - un centro (3 coordinate float);
  - un raggio (utile a mantenere l'insieme di cellule figlie) (1 float);
  - un orientamento completo a 3 assi (3 angoli float);
  - coordinate in metri;
- un modo per mantenere le equazioni del moto quadri-dimensionale (es. ter-

ra-sole) delle Cellulae figlie;

- un servizio attraverso cui poter ottenere informazioni su: quali periodi in cui si differenzia una certa Cellula (es. Cellula-terra => ...terziario, quaternario....), in quali luoghi;
- un involucro per tutte le Tabulae che contiene lo stato della/e Tabulae in essa contenute.

In particolare, all'interno delle Cellulae è presente un processo di calcolo della posizione (in metri) dell'ultranauta espresso in coordinate della cellula root. Infatti, grazie al fatto di aver memorizzato la posizione dell'utente in formato concettuale, viene percorso il path di Cellulae sino ad arrivare all'utente. Dopodichè, sapendo la posizione dell'utente rispetto alla cellula root, si procede con la generazione vera e propria.

3.3 Architettura di Ultraport

UltraPort indica la parte Client del sistema complessivo Client-Server che consente ad un Ultranauta l'esplorazione in ultravisione dei canali di Ultramundum.
In tale accezione, UltraPort può essere costituito, nel caso minimo, da un singolo software che viene eseguito su di un computer (con sistema operativo Windows, Linux, Mac OSX, ecc...) o da un opportuno decoder connesso ad un televisore (Xbox, PlayStation, ecc...).
In questo caso si parla di UltraPort monolitico.
Le varie funzionalità necessarie ad UltraPort sono suddivise in sottosistemi.

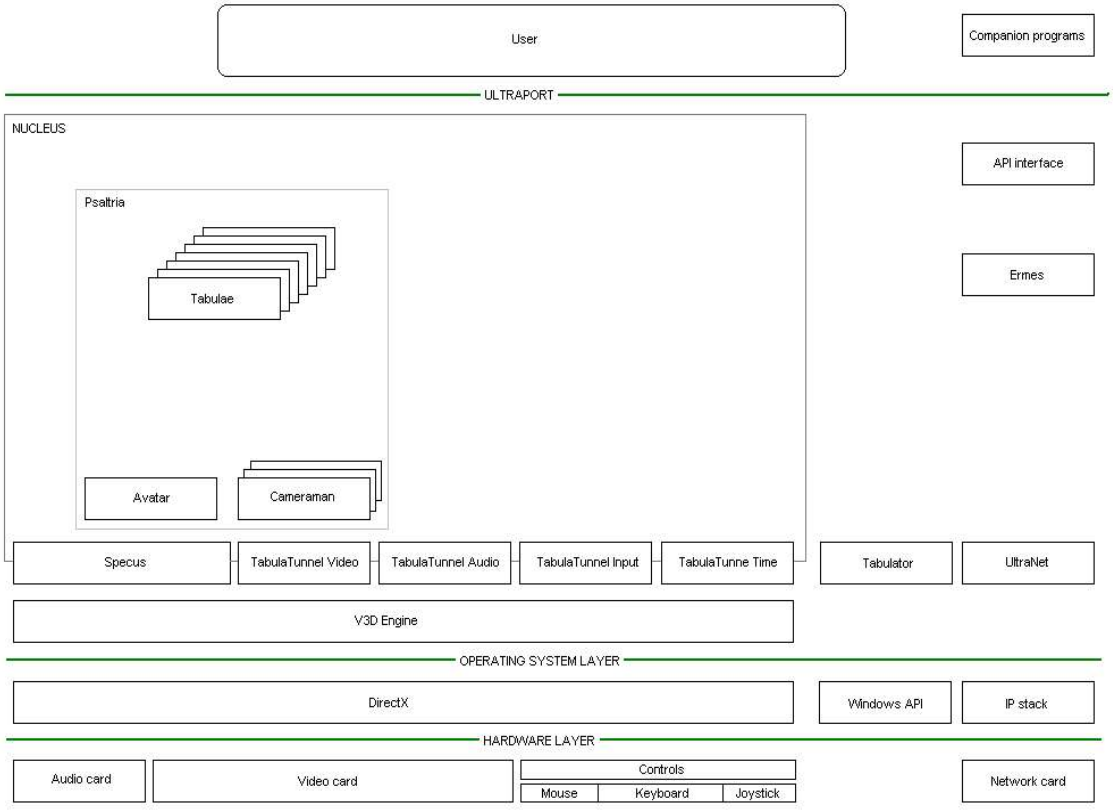


Figura 3.2: schema dell'architettura di Ultraport.

3.3.1 Il Nucleus

L'insieme delle Tabulae in essere su di uno specifico UltraPort prende il nome di Nucleus.
Nel Nucleus avviene la vera e propria elaborazione dell'evoluzione del mondo digitale in cui è immerso l'ultranauta.
Si compone essenzialmente di:

- Struttura dati (programmabile) rappresentata dalle Tabulae
- Tabula Avatar rappresentante l'ultranauta nel mondo digitale
- Tabulae tunnel per l'ingresso/uscita dati da/per il Nucleus
- Psaltria (Virtual machine) utile a gestire/interpretare in modo completo le tabulae (ivi compresa l'esecuzione della parte programmabile delle stesse)

Le Tabulae debbono comunicare intensamente tra di loro onde consentire l'evoluzione voluta del mondo digitale. Le comunicazioni tra Tabulae e tra le Tabulae e gli elementi del Nucleus sono governate dal sottosistema Ermes, cosicché il Nucleus possa anche essere distribuito su più computer.Finché non avviene uno scambio di messaggi, il Nucleus è in quiete e le Tabulae non sono in esecuzione. I sottosistemi di output sensoriale continuano a rigenerare l'immagine del mondo per l'utente (se ottimizzati, solo quando vi siano variazioni), ma nulla avviene.
Qualunque movimento dell'utente, o anche il semplice passaggio di un intervallo di tempo, produce la generazione di messaggi che attivano le operazioni del Nucleus per far evolvere il mondo digitale in modo corretto.
I comportamenti delle Tabulae possono essere definiti in termini di reazioni a messaggi (meccanismo Event-Action). Ogni Tabula non è attiva finché non riceve un messaggio da un'altra Tabula o dal Nucleus, dopodichè si attiva per elaborare una risposta allo stesso. Questo può portare alla generazione di altri messaggi e così via.

3.3.2 Specus

Lo Specus, nome derivato dal latino (Caverna) a citazione di Platone, è un modulo istanziato su tutti i computer che necessitano di un qualsiasi riferimento allo SceneGraph.
Lo SceneGraph rappresenta il grafo dei nodi che devono essere renderizzati. Ogni elemento della scena è rappresentato da tre nodi. Il primo descrive la "mesh", ossia una struttura dati che contiene i dettagli della mesh stessa (es. il numero dei vertici, il tipo dei vertici...). A questo nodo, ne vengono "attaccati" altri due: il primo è il nodo che rappresenta il materiale relativo alla mesh, il secondo definisce le primitive utilizzate, ossia i triangoli veri e propri, che saranno poi renderizzati nella scena. La costruzione dei nodi dello SceneGraph rispetta questa sequenza di passi. Il grafo dei nodi dello Specus ha origine nel nodo radice e si sviluppa "attaccando" i nodi successivi. Inoltre i nodi possono anche essere tolti dal grafo e distrutti per ottimizzare le risorse della macchina nella descrizione dell'ambiente.

L'interfaccia pubblica

Lo Specus ha un'interfaccia pubblica, accessibile mediante messaggi di Ermes, e un'interfaccia diretta, accessibile mediante chiamata ai suoi metodi.
La scrittura e modifica dello SceneGraph può avvenire sia tramite chiamata diretta ai metodi che tramite messaggi.
Ecco i principali messaggi che vengono inviati allo Specus per la gestione dei nodi:

- UM_EMESS_TYPE_CREATESGNODE, per la creazione di un nodo;
- UM_EMESS_TYPE_SETSGNODE, per impostare la descrizione di un nodo creato;
- UM_EMESS_TYPE_INITIALIZESGNODE, per inizializzare un nodo precedentemente creato e generare così tutte le variabili interne a partire dalla descrizione impostata;
- UM_EMESS_TYPE_ATTACHSGNODETOSGNODE, per "attaccare" il nodo creato ad un altro esistente per creare la struttura ad albero;
- altre funzioni per inserire e cancellare nodi, liberando così lo SceneGraph.

3.3.3 Sequenza di rendering

Il Nucleus è istanziato in una singola classe, fisicamente presente su un solo computer.
La sequenza di rendering avviene sulla base di una cadenza temporale definente il target di qualità che si vuole ottenere. Ad esempio, cinquanta fotogrammi al secondo. Una base tempi per il rendering è contenuta all'interno di ogni Specus ed avvia in parallelo due processi: la creazione del nuovo stato e il rendering dello stato precedente. Se uno dei due processi non è ancora terminato nell'istante di avvio del nuovo periodo, lo Specus conteggia il tempo necessario al termine. Indicatori specifici per i due processi sono inviati al Nucleus che può così porre in essere strategie di ottimizzazione sulla base di quanto tempo è stato risparmiato o di quanto ne è stato necessario in più rispetto alla cadenza temporale dell'obiettivo di qualità.
La gestione della base tempi è demandata al 3DEngine, che gestisce direttamente le sequenze di transform e rendering in multithreading. Tali sequenze sono temporizzate in modo da lasciare tempo di CPU per le funzioni del sistema operativo e degli altri moduli di UltraPort. Modifiche a tali temporizzazioni sono demandate al Nucleus, che riceve statistiche aggiornate in real time dal motore.
Ad ogni DeltaT di tempo il 3DEngine invia un messaggio interno allo Specus che lo invia con un messaggio Ermes locale all'Avatar. L'Avatar calcola la propria nuova posizione e frame d'animazione, generando messaggi di Ermes per l'interfaccia di manipolazione dello SceneGraph dello Specus.

3.3.4 Generazione iniziale

All'apertura di un canale si rende necessario generare la prima volta la scena.
Si utilizza una tecnica a doppia passata:
Prima passata:

- Il Nucleus richiama la prima cellula madre, con in mano la posizione dell'utente.
- La cellula madre si attiva per stabilire quale fra le Cellulae figlie andrà istanziata. Questo processo viene fatto in base a diversi parametri, tra cui: dimen-

sione, distanza dall'utente, coefficiente d'importanza.

- Per ogni Cellula figlia effettivamente istanziata si calcola il vettore di n elementi, rappresentanti i costi relativi alle risorse ritenute fondamentali per l'elaborazione (es. numero triangoli, dimensione texture, numero di istruzioni di linguaggio carmina eseguite, numero di canali audio, ecc...).
- Tale vettore deve essere retropropagato dalle Cellulae figlie alle Cellulae madri, in modo da garantire che ogni cellula madre possieda la somma dei costi indicati nei vettori delle Cellulae figlie.
- Si continua il processo ricorsivamente.
- Alla fine ogni Cellula madre saprà per ogni Cellula figlia effettivamente istanziata, di quali risorse ha bisogno per generarsi rispetto alla posizione ed alla distanza dall'ultranauta.
- Ad ogni Tabula viene passata la distanza e un coefficiente di importanza definito dall'autore.

Seconda passata:

- Il Nucleus, disponendo di un vettore di risorse massime spendibili, proveniente da un processo preliminare di analisi della configurazione hw/sw del computer utilizzato, procede con l'avvio di un processo ricorsivo di allocazione e consumo delle risorse usate, partendo dalla Cellula root.
- Ciascuna Cellula procederà ricorsivamente alla distribuzione delle risorse ancora spendibili alle Cellulae figlie istanziate, sino a raggiugere tutte le foglie della scena corrente.
- Ciascuna Cellula salverà dentro di sè le risorse che gli sono arrivate e che tocca distribuire alle Cellulae figlie. Tale info sarà utilizzata nella fase di realtime.
- Ciascuna Cellula si genererà in base alle risorse che gli sono state effettivamente assegnate.

3.4 Programmazione

3.4.1 Convenzione sui nomi

Al fine di uniformare la stesura del codice da parte di più programmatori, è ovviamente necessario stabilire alcune semplici convenzioni sulla denominazione dei vari elementi base costituenti un programma; nella fattispecie, per la programmazione ad oggetti, classi, strutture, variabili, costanti e funzioni. La maggior parte delle convenzioni si basa essenzialmente sull'uso di un particolare prefisso per i nomi degli elementi suddetti. Inoltre un buon uso nel scegliere i nomi veri e propri di quest'ultimi è quello di cominciare con la lettera maiuscola ciascun nome o parte dello stesso. Di seguito vengono forniti alcuni semplici esempi di quanto appena indicato: m_dwResolutionX, m_dwResolutionY, g_bIsInitialized, eccetera.

3.4.2 Nomi di variabili membro e globali

Parallelamente ai prefissi usati per indicare l'area di validità di un particolare dato, è necessario utilizzarne uno per far riconoscere immediatamente il tipo di una qualsiasi variabile o di una costante.

Tipo	**Prefisso**	**Tipo ridefinito**
void	v	UM_void
bool	b	UM_bool
char	s8	UM_s8
unsigned char	u8	UM_u8
int	n	UM_int
unsigned int	un	UM_uint
float	f32	UM_f32
double	f64	UM_f64

Nella tabella riportata sono inseriti i principali tipi di dati con il rispettivo prefis-

so e il nome del tipo ridefinito da utilizzarsi nei sorgenti Ultramundum in modo da evitare l'uso di nomi C che possono non avere implementazioni identiche sulle varie piattaforme.

3.4.3 Ulteriori prefissi di base

Parallelamente ai prefissi usati per indicare il tipo di una qualsiasi variabile o di una costante, ocorre utilizzarne altri da anteporre ad ogni altra entità che si possa incontrare nei programmi.
Di seguito viene riportata una tabella in cui sono inseriti i principali tipi di entità con il rispettivo prefisso.

Tipo entità	**Prefisso**	**Esempio**
Nomi classi	C	CStato
Nomi strutture	S	SWindData
Membro di una classe	m_	m_nElemento
Nomi di funzioni membro di una classe	UM_	UM_Salta(3)
Variabile globale	g_	g_nValore
Variabile statica	s_	s_nXCoord
Puntatore	p	ps32Counter
Puntatore a funzione	pfn	PftCalcola()
Puntatore FILE	pf	FILE* pfSource
Stringa Unicode con terminatore /0	sz	szNome

4. LE FASI DEL PROGETTO

4.1 Celestia

Celestia è un software open source sviluppato da Chris Laurel, un fisico americano specializzato in grafica tridimensionale. Questo programma è un “simulatore spaziale” che permette ad un osservatore di spostarsi nello spazio all'interno della nostra galassia da una stella a un'altra e all'interno del nostro sistema solare. Si possono quindi compiere “veri e propri” viaggi spaziali.

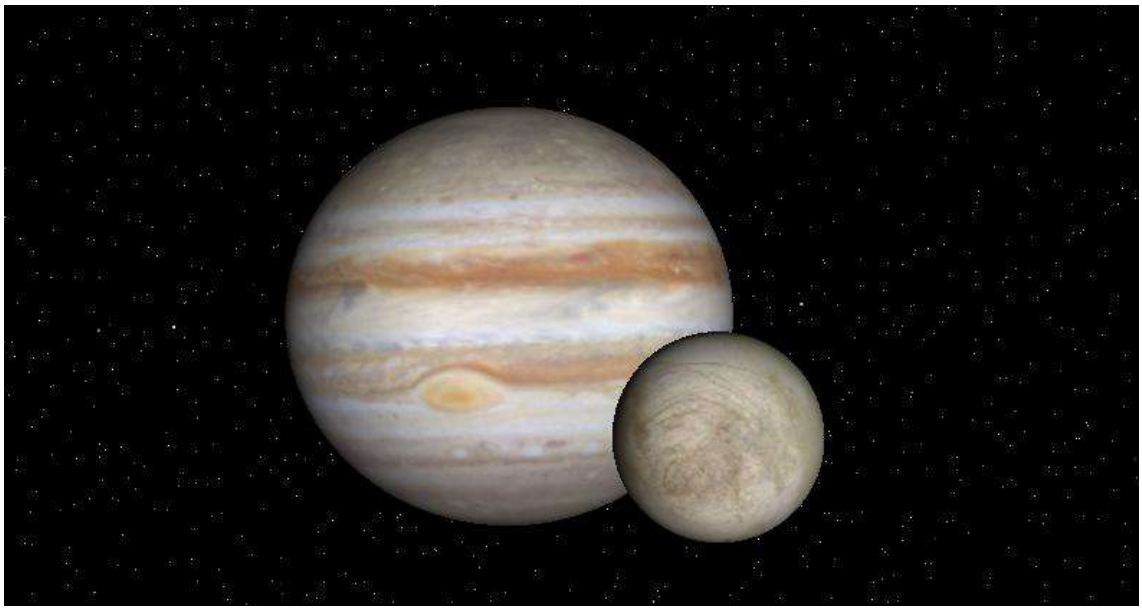


Figura 4.1: l'immagine catturata da Celestia, rappresenta Europa con Giove sullo sfondo.

Il software Celestia è multipiattaforma e ne esistono versioni per Windows, per Linux e per Mac OSX. Ciò vale sia per i pacchetti precompilati sia per i codici sorgenti.

Per apprendere conoscenze a noi non note, più che altro dal lato scientifico, è stato fatto del reverse engeenering sui codici di Celestia.

Celestia utilizza OpenGL per disegnare gli oggetti in tridimensionale. OpenGL è

un interfaccia software verso l'acceleratore grafico. L'interfaccia consiste in un set di parecchie centinaia di procedure e funzioni che permettono di specificare oggetti e operazioni implicate nel produrre immagini di oggetti tridimensionali. OpenGL può essere vista anche come una macchina a stati che controlla un set di operazioni per il disegno. Le chiamate alle funzioni OpenGL si trovano ospiti all'interno del codice C++.

Abbiamo compilato Celestia su piattaforma Linux, precisamente Slackware 9.1, sulla quale è stata interamente svolta la prima fase. Non è stato possibile utilizzare ambienti di sviluppo (quali ad esempio KDevelop), con i quali procedere al debug del programma in esecuzione. Abbiamo dovuto limitarci a capire il codice leggendolo o, eventualmente, apportando qualche modifica, ricompilandolo ed eseguendolo per vedere gli effetti del cambiamento.

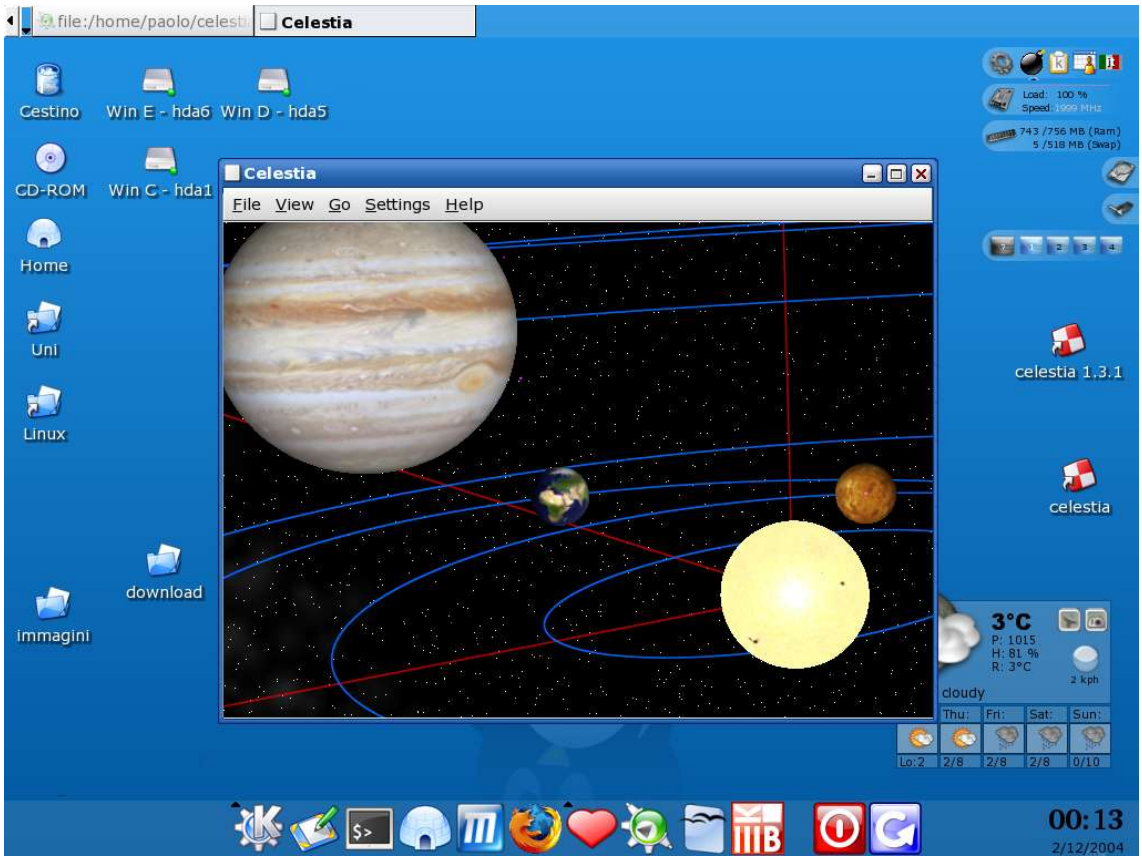


Figura 4.2: il sistema solare e, in rosso, gli assi (x, y, z) di riferimento.

La Figura 4.2 mostra un esempio di modifiche apportate a Celestia. In essa è visibile il sistema solare; per ingrandire i pianeti è bastato alterare il reale raggio

di ognuno di questi, che viene così renderizzato molto più grande. Sono stati disegnati da noi anche gli assi x, y, z. Ogni volta che Celestia renderizza un sistema planetario sposta il sistema di riferimento sulla stella principale di quel sistema. In Figura 4.3 è visibile un immenso poligono; un angolo di questo poligono è marcato con un rombo di colore rosso che indica la posizione del Sole.
L'obiettivo del nostro progetto è stato quello di implementare qualcosa di simile a Celestia utilizzando la tecnologia della Fondazione: leggere i dati dalle opportune sorgenti, importarli nel formato tabulare e visualizzarli.
I principali argomenti approfonditi sono stati:

- dove è possibile reperire cataloghi in formato facilmente leggibile;
- come utilizzare i dati presenti nei cataloghi;
- il calcolo della distanza e della posizione di ogni stella;
- il calcolo delle orbite dei pianeti;
- la gestione del far clipping plane e del depth buffer.

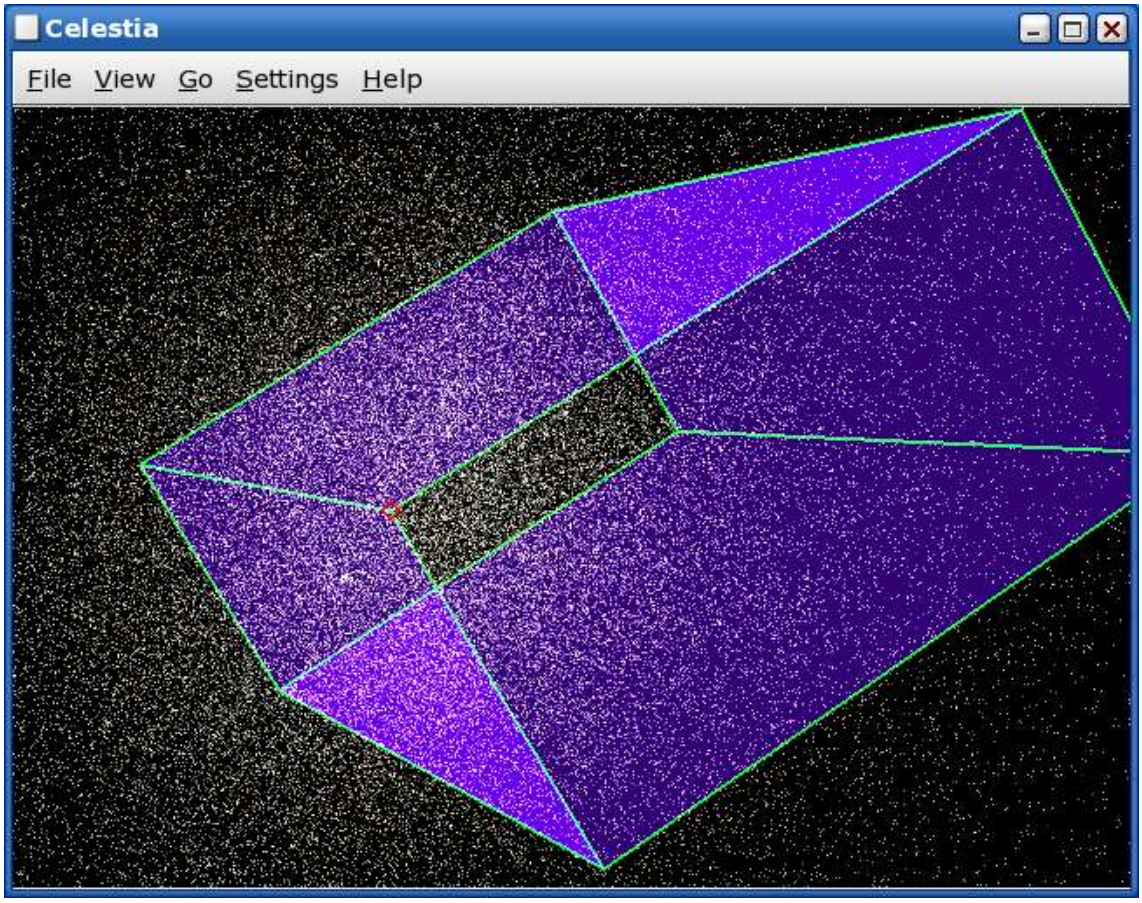


Figura 4.3: un immenso poligono nello spazio. Il rombo rosso in un angolo del cubo indica la posizione del Sole.

4.1.1 Catalogo delle stelle

Celestia memorizza i dati riguardanti le stelle, necessari per il loro posizionamento e renderizzazione (dati del catalogo Hipparcos o Tycho) in un file binario. Questo è stato ottenuto partendo dai cataloghi originali, con l'esclusione delle informazioni superflue o che possono essere indirettamente calcolate. Il database iniziale è costituito da 112522 stelle, prese per la maggior parte dal catalogo Hipparcos, più un aggiunta: il nostro Sole, che per questo viene indicato con il numero di catalogo 0 (zero). Con vari add-ons è possibile utilizzare anche il più corposo catalogo Tycho.
Il primo campo di questo catalogo è un `int` e contiene il numero di record (quindi di astri) presenti nel file, successivamente si trovano elencate tutte le occorrenze. Le informazioni tenute per ogni stella sono organizzate in questa forma e in quest'ordine:

```
4 byte int   : numero di catalogo HIP
4 byte int   : numero di catalogo HD
4 byte float : ascensione retta (RA)
4 byte float : declinazione (dec)
4 byte float : parallasse
2 byte int   : magnitudine apparente
2 byte int   : classe stellare
1 byte       : errore della parallasse
```

Il catalogo HD è un altro catalogo stellare molto usato; prende il nome dalle iniziali del suo autore: Henry Draper.
Siccome ci sono molte stelle nel database, è importante che la dimensione dei dati di ciascuna stella sia minima. Sono quindi memorizzate solo le informazioni essenziali. Altri dati, come il raggio, la temperatura, il colore, il periodo di rotazione, sono calcolati derivandoli dalle informazioni in possesso.

4.1.2 Calcolo della distanza e della posizione

La posizione di ogni una stella nel database è rappresentata da coordinate polari nel sistema equatoriale; quella che Celestia tiene in memoria centrale e usa durante l'esecuzione è rappresentata da coordinate cartesiane nel sistema eclittica-le, come unità di misura sono utilizzati gli anni luce. Di seguito è spiegato come

avviene la trasformazione tra questi due sistemi.
Quando il database viene letto in fase di avvio del programma, per ogni stella, RA, dec e parallasse sono convertiti in coordinate x, y, z e la magnitudine apparente in magnitudine assoluta. Per fare ciò si calcola prima la distanza della stella da noi. La formula utilizzata è la seguente:

```
distance = LY_PER_PARSEC / (parallax>0 ? parallax/1000 : 1e-6)
```

LY_PER_PARSEC rappresenta gli anni luce per parsec, ed equivale a 3.26167.
Se la parallasse è maggiore di 0, questa viene divisa per 1000, altrimenti viene utilizzato per il calcolo un numero molto grande, per far sì che la distanza risulti minima. Questo è il caso del Sole: in Celestia la posizione del nostro astro viene calcolata come quella di tutti gli altri, introducendo però un errore perché non risulta esattamente sull'origine degli assi (come dovrebbe) ma traslato di 0.00326167 anni luce sull'asse delle x.
Tuttavia questo errore è ininfluente in quanto la differenza è minima rispetto alle distanze stellari.
Per trasformare le informazioni sulle posizioni da coordinate polari (bidimensionali con l'indicazione della distanza) in un sistema di coordinate cartesiane (tridimensionale) vengono utilizzate le seguenti formule:

```
theta = ra / 24 * PI * 2 + PI;
phi = (dec / 90 - 1) * PI / 2;
x = cos(theta) * sin(phi) * distance;
y = cos(phi) * distance;
z = -sin(theta) * sin(phi) * distance;
```

A questo punto si sono ottenute delle coordinate cartesiane nel sistema equatoriale, cioè relative all'equatore celeste (che corrisponde anche all'equatore terrestre).
C'è bisogno però di un ulteriore passo, quello verso il sistema eclitticale, come indicato nel testo “Astronomy on the personal Computer” [1].
Le coordinate equatoriali e quelle eclittiche differiscono nel piano di riferimento dal quale vengono misurate. Nel primo caso il piano è perpendicolare all'asse terrestre, quindi parallelo all'equatore (cioè all'equatore celeste), mentre nel secondo è il piano dell'orbita della Terra. L'asse x-x' è comune ai due sistemi, ed è definito come la direzione dell'equinozio di primavera o del “Punto d'Ariete”.

Questa direzione è perpendicolare al polo nord dell'eclittica (l'asse z) e al polo nord celeste. L'angolo tra i due piani è di circa 23,5 gradi. E' necessaria perciò la rotazione di ogni punto della differenza di inclinazione tra i due piani.

```
angle = (23.4392911 / 180 * PI);
c = cos(angle);
s = sin(angle);
```

L'esatto valore dell'angolo (ε) in Figura 4.4 non è costante, ma decresce di 47" per secolo. Questo deriva da alterazioni nell'orbita della Terra come risultato di perturbazioni causate dagli altri pianeti e dalla Luna.
Il punto viene quindi moltiplicato per la matrice di rotazione:

```
//                {1, 0, 0}
// {x, y, z}   *  {0, c, -s}
//                {0, s, c}
```

Per ruotare o traslare oggetti in openGL, vengono sempre utilizzate matrici.
La relazione tra coordinate cartesiane equatoriali (x, y, z) e coordinate cartesiane eclittiche (x', y', z') di un punto è la seguente:

```
x' = x;
y' = (c * y) + (s * z);
z' = ((-s) * y) + (c * z);
```

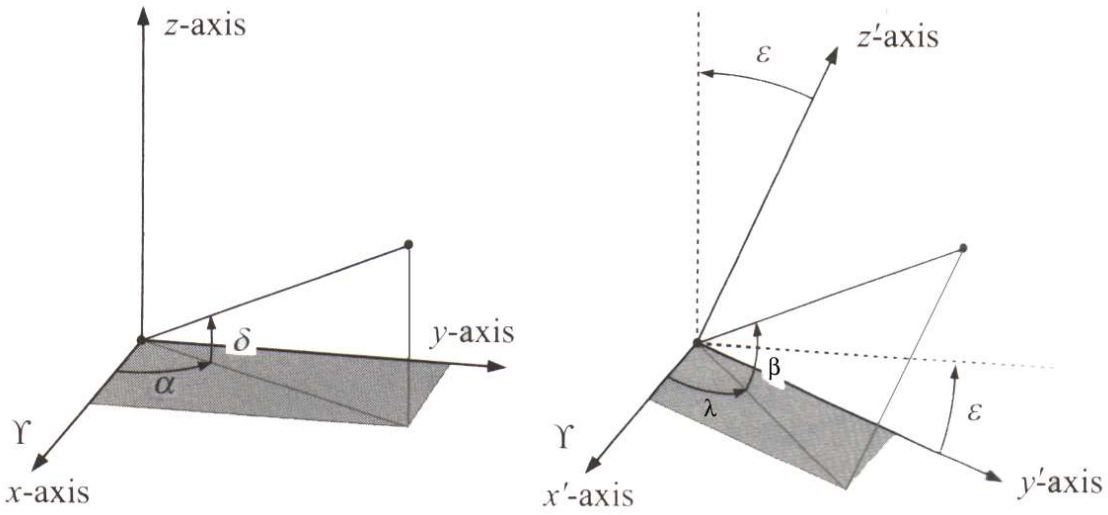


Figura 4.4: coordinate nel sistema equatoriale ed eclittico.

In Celestia ogni stella è memorizzata con coordinate assolute rispetto al sistema eclitticale.

4.1.3 Calcolo della magnitudine

Per determinare la magnitudine assoluta a partire da quella apparente è necessario sapere la distanza:

```
absoluteMag = (apparentMag / 256 + 5 - 5 * log10(distance / 3.26))
```

La magnitudine apparente andrebbe bene solo se non ci si dovesse mai spostare dalla Terra, ma in caso contrario è meglio rendere il tutto più generico usando la magnitudine assoluta.

4.1.4 Calcolo delle principali proprietà delle stelle

L'ultima informazione tenuta è la classe stellare; questa di per sé non è una caratteristica degli astri. E' semplicemente una variabile di tipo `unsigned short` (la cui dimensione è 16 bit), nella quale vengono tenute quattro proprietà in quest'ordine: il tipo di stella, la classe spettrale, la sottoclasse spettrale e la classe di luminosità. Ognuna di queste è rappresentata da un numero, il quale occupa quattro bit (con quattro bit ho la possibilità di avere valori compresi tra 0 e 15). Queste caratteristiche sono concatenate per poter essere memorizzate in un'unica variabile. In questo modo si ha un risparmio di spazio molto importante. Ad esempio per ottenere la classe spettrale bisogna fare uno shift di 8 bit sulla variabile e mettere in and bit a bit il risultato con un numero in formato binario che ha i quattro bit meno significativi a 1e tutti gli altri a 0. In questo modo vengono fatti emergere col valore precedente solo i quattro bit a sinistra, che rappresentano appunto la classe spettrale, mentre i restanti dodici più significativi sono settati a 0.

```
spectralClass = (stellarClass >> 8) & 0xf
```

Tutte le informazioni necessarie non tenute in memoria vengono calcolate a richiesta. Alcuni dei metodi utilizzati a questo fine sono di seguito riportati come esempio.

Colore

Il colore di una stella dipende dalla classe spettrale di questa, come si può vedere di seguito:

```
Color StellarClass::getApparentColor(SpectralClass spectralClass)
{
    switch (spectralClass)
    {
    case Spectral_O:
        return Color(0.7f, 0.8f, 1.0f);
    case Spectral_B:
        return Color(0.8f, 0.9f, 1.0f);
    case Spectral_A:
        return Color(1.0f, 1.0f, 1.0f);
    case Spectral_F:
        return Color(1.0f, 1.0f, 0.88f);
    case Spectral_G:
        return Color(1.0f, 1.0f, 0.75f);
    case Spectral_K:
        return Color(1.0f, 0.9f, 0.7f);
    case Spectral_M:
        return Color(1.0f, 0.7f, 0.7f);
    case Spectral_R:
    case Spectral_S:
    case Spectral_N:
        return Color(1.0f, 0.4f, 0.4f);
    case Spectral_L:
    case Spectral_T:
        return Color(0.75f, 0.2f, 0.2f);
    default:
        return Color(1.0f, 1.0f, 1.0f);
    }
}
```

Il metodo restituisce un colore nelle sue componenti RGB (Red, Green, Blue). Le classi `Spectral_O, Spectral_B,` eccetera... sono identificate da un `enum`.
Tutte le funzioni da qui in poi sono funzioni membri della classe `Star` e vengono quindi chiamate su un oggetto di tipo stella.
All'interno della classe `Star` sono presenti tavole di valori numerici che servono per il calcolo della temperatura, del periodo di rotazione, e della correzione dovuta alla magnitudine bolometrica. Per ognuna di queste tre caratteristiche, i dati sono tenuti in array e matrici. Esistono tanti array (temperatura) o tante matrici (le restanti due) quante sono le classi spettrali. A questi dati si accede a seconda della classe spettrale, della sottoclasse e della classe di luminosità della stella in questione.
I dati per le temperature delle stelle sono presi dal testo Carroll and Ostlie, Modern Astrophysics [3]. Quelle per i tipi non presenti sono interpolate.

Un esempio di questo è il codice seguente:

```
static float tempF[10] =
{
    7200, 7050, 6890, 6740, 6590, 6440, 6360, 6280, 6200, 6110
};
```

Le tavole con le correzioni per stimare la magnitudine bolometrica dalla magnitudine assoluta visuale sono matrici; ogni riga corrisponde a una classe di luminosità. I dati delle classi spettrali mancanti sono interpolati linearmente.

```
static float bmag_correctionO[3][10] =
{
    // Lum class V (main sequence)
    {   -4.75f, -4.75f, -4.75f, -4.75f, -4.45f,
        -4.40f, -3.93f, -3.68f, -3.54f, -3.33f  },
    // Lum class III
    {   -4.58f, -4.58f, -4.58f, -4.58f, -4.28f,
        -4.05f, -3.80f, -3.58f, -3.39f, -3.13f  },
    // Lum class I
    {   -4.41f, -4.41f, -4.41f, -4.41f, -4.17f,
        -3.87f, -3.74f, -3.48f, -3.35f, -3.18f  }
};
```

Le ultime tavole sono utilizzate per il periodo di rotazione. I dati possono non essere molto precisi per i periodi delle stelle giganti e supergiganti delle classi K e M, in quanto potrebbero essere considerevolmente più lenti, ma sono difficili da calcolare quando la velocità di rotazione è troppo lenta per interessare lo spettro.

```
static float rotperiod_A[3][10] =
{
   { 0.7f, 0.7f, 0.6f, 0.6f, 0.5f, 0.5f, 0.5f, 0.6f, 0.6f, 0.7f },
   { 2.5f, 2.3f, 2.1f, 1.9f, 1.7f, 1.6f, 1.6f, 1.7f, 1.7f, 1.8f },
   { 75.0f, 77.0f, 80.0f, 82.0f, 85.0f, 87.0f, 95.0f, 104.0f,
     115.0f, 125.0f },
};
```

Temperatura

Il valore della temperatura del Sole è attualmente 5780 gradi Kelvin, ma le tavole listano la temperatura di una stella della classe G2V come 5860 gradi. Verrà usato il secondo valore, così il suo raggio è calcolato correttamente. L'alta presenza di metalli nel Sole è probabilmente la causa della discrepanza.
La temperatura è restituita in gradi Kelvin.

```
float Star::getTemperature()
{
    if (stellarClass.getStarType() == NormalStar)
    {
        spectralSubClass = stellarClass.getSpectralSubclass();
        switch (stellarClass.getSpectralClass())
        {
        case Spectral_O:
            return tempO[spectralSubClass];
        case Spectral_B:
            return tempB[spectralSubClass];
        ...
        case Spectral_T:
            return tempT[spectralSubClass];
        default:
            return -1;
        }
    }
    else if (stellarClass.getStarType() == WhiteDwarf)
        return 10000;
    else if (stellarClass.getStarType() == NeutronStar)
        return 10000;
    else if (stellarClass.getStarType() == BlackHole)
        return 0;
    else
        return -1;
}
```

Se è una normale stella, a seconda delle classe spettrale si accede al determinato array contenete le temperature. L'indice del vettore è identificato dalla sottoclasse. Nel caso di nane bianche, stelle di neutroni o buchi neri viene restituito un valore fisso.

Raggio

La funzione `getRadius()` utilizza la legge di Stefan-Boltzmann per calcolare il raggio.

```
float Star::getRadius()
{
      // Calculate the luminosity of the star from the bolometric
    float solarBMag = SOLAR_ABSMAG + bmag_correctionG[0][2];
    float bmag = getBolometricMagnitude();
    float luminosity = (float) exp((solarBMag - bmag) / LN_MAG);

      // Use the Stefan-Boltzmann law to estimate the radius of a
      // star from surface temperature and luminosity
    return SOLAR_RADIUS * (float) sqrt(luminosity) *
        square(SOLAR_TEMPERATURE / getTemperature());
}
```

Prima di applicare questa legge, la funzione deve reperire alcune informazioni, tra le quali la magnitudine bolometrica, utilizzata per calcolare la luminosità

della stella e la sua temperatura. La magnitudine bolometrica viene calcolata dall'apposita funzione `getBolometricMagnitude()`, omessa per motivi di spazio. E' una stima derivata dalla magnitudine assoluta visuale più un valore di correzione. Il raggio calcolato è espresso in kilometri.

Periodo di rotazione

Anche le stelle, oltre ai pianeti, possiedono un moto di rotazione; il tempo impiegato a compiere un intero giro su se stesse è conteggiato in giorni terrestri. Per le stelle normali si usano i dati presenti nelle tavole sopra menzionate, memorizzati in matrici. Per trovare il valore della correzione, si rendono necessari due indici utilizzati sulla matrice appropriata, che è identificata dalla classe spettrale. Il primo indice, che indica la riga, è basato sulla classe di luminosità; il secondo, che identifica la colonna, si basa sulla sottoclasse spettrale.
Per le nane bianche e le stelle di neutroni vengono assegnati dei valori fissi, pari a mezz'ora per le prime e a un secondo per le seconde. Per i buchi neri non è possibile asserire nulla, a causa delle scarse conoscenze in campo astronomico.

```
float Star::getRotationPeriod()
{
   float period = 1.0f;

   if (stellarClass.getStarType() == NormalStar)
   {
       specSubClass = stellarClass.getSpectralSubclass();
       lumIndex = 0;

       switch (stellarClass.getLuminosityClass())
       {
           case Lum_Ia0:
               lumIndex = 2;
               break;
           ...
           case Lum_V:
           case Lum_VI:
               lumIndex = 0;
               break;
       }

       switch (stellarClass.getSpectralClass())
       {
           case Spectral_O:
               period = rotperiod_O[lumIndex][specSubClass];
               break;
           case Spectral_B:
               period = rotperiod_B[lumIndex][specSubClass];
               break;
           ...
```

```
            case Spectral_L:
            case Spectral_T:
                period = 0.2f;
                break;
        }
    }
    else if (stellarClass.getStarType() == WhiteDwarf)
        // Assign white dwarfs a rotation period of half an hour;
        // very rough, as white rotation rates vary a lot.
        return 1.0f / 48.0f;

    else if (stellarClass.getStarType() == NeutronStar)
        // Let all neutron stars have a rotation period of one second
        return 1.0f / 86400.0f;

    return period;
}
```

Altre proprietà

La classe `Astro` mette a disposizione una serie di funzioni per la conversione di valori tra diverse unità di misura, di altre proprietà (e.g. la luminosità) da informazioni conosciute.
La funzione nell'esempio restituisce il numero di volte che la stella è più luminosa rispetto al Sole. Come parametro richiede la magnitudine assoluta.

```
float Astro::absMagToLum(float absMag)
{
    return exp((SOLAR_ABSMAG - absMag) / LN_MAG);
}
```

4.1.5 Gestione del depth buffer

Tanto più lontano è un oggetto dalla camera (dall'osservatore), tanto più piccolo apparirà nell'immagine sullo schermo. Ciò accade perché il volume di osservazione per la proiezione della prospettiva è un frustum, cioè una piramide troncata la cui parte superiore è tagliata da un piano parallelo alla sua base [9].
Gli oggetti che fanno parte del volume di osservazione sono proiettati verso l'apice della piramide, in cui si trova il punto di vista. Gli oggetti più vicini all'osservatore appaiono più grandi perché occupano uno spazio proporzionalmente maggiore del volume di osservazione, rispetto a quelli più lontani. Questo metodo di proiezione è simile al modo in cui il nostro occhio (o una macchina fotografica) lavora.

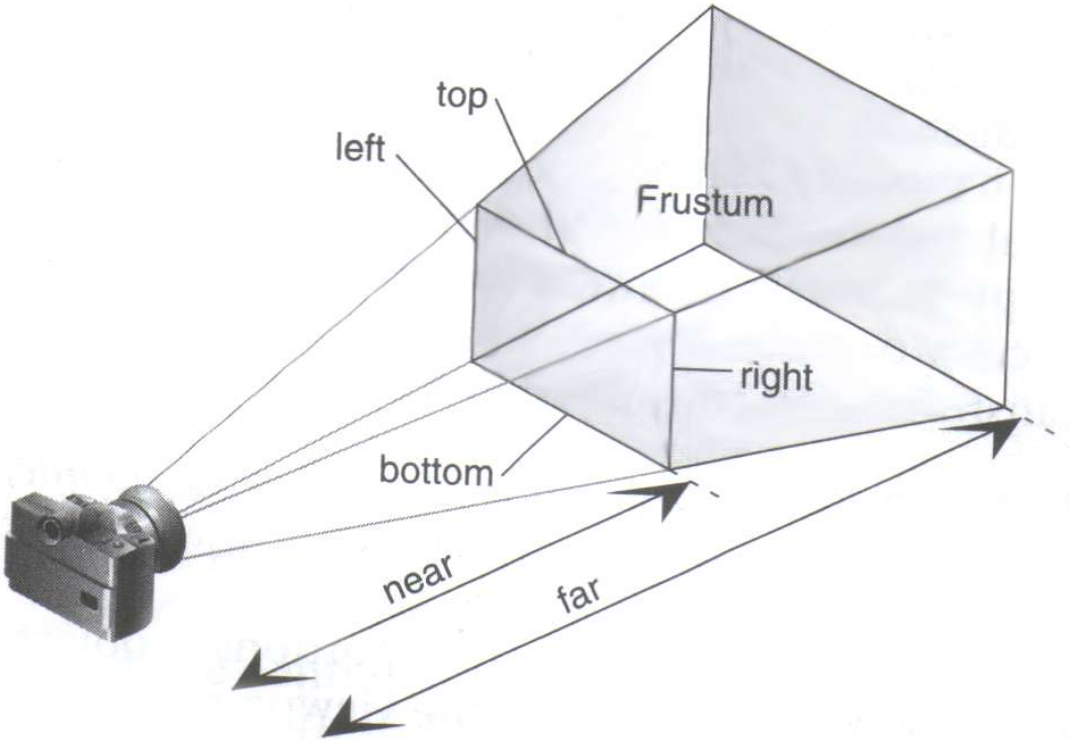


Figura 4.5: le due frecce indicano il near e il far clipping plane.

La base del frustum più grande e lontana dall'osservatore è chiamata far clipping plane; qualsiasi oggetto situato oltre questo piano non viene disegnato. Quella più piccola e vicina prende il nome di near clipping plane; per questa, al contrario, qualsiasi oggetto più vicino (quindi compreso tra l'osservatore e il piano) non viene renderizzato. Il sistema disegna quindi solamente tutto ciò che risiede all'interno del frustum.

Per ogni pixel sullo schermo, il depth buffer tiene traccia della distanza tra il punto di vista e l'oggetto che occupa il pixel. Il depth buffer è usato per eliminare le superfici nascoste; ogni nuovo pixel è disegnato solo se il corrispondente oggetto è più vicino dell'oggetto precedente. In questo modo, dopo che l'intera scena è stata renderizzata, rimangono solo gli oggetti non oscurati da altri [11].

Il valore della profondità è rapportato tra 0 (sul near clipping plane) e 1 (sul far clipping plane). Ogni valore corrisponde a un intervallo, nel quale viene inserito il pixel con quella profondità. Il range è quindi spalmato tra i due piani e se questi sono troppo lontani (e.g. diverse migliaia di anni luce, come nel nostro caso), l'accuratezza del depth buffer diminuisce enormemente, poiché gli intervalli tra

un valore e l'altro si allargano.

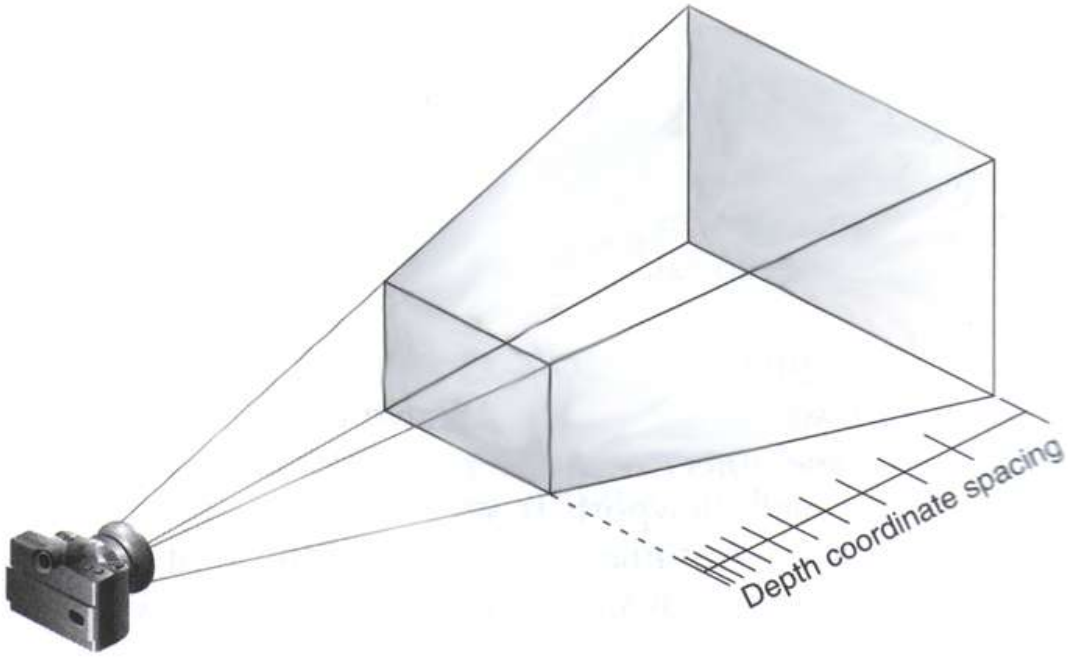


Figura 4.6: il range di intervalli del depth buffer.

Se due oggetti vicini tra loro cadono nello stesso intervallo (perché non distanti sufficientemente per andare in diversi intervalli), é renderizzato sopra quello disegnato dopo e non quello effettivamente più vicino.
In Celestia e in De Astris si tratta appunto di disegnare oggetti con distanze enormi tra loro; bisogna perciò settare il far clipping plane alla distanza di migliaia di anni luce (per non tagliare fuori le stelle). Ad esempio, renderizzando un pianeta con un suo satellite, questi potrebbero cadere nello stesso intervallo del depth buffer e verrebbe quindi disegnato sopra quello renderizzato per ultimo e non il più vicino.
Celestia risolve questo problema utilizzando l'algoritmo “del pittore” nel quale gli oggetti vengono disegnati in ordine dal più lontano al più vicino.
Prima di tutte, vengono disegnate le stelle lontane (rappresentate con punti più o meno luminosi) con il depth buffer disabilitato e non viene eseguito il controllo sulla distanza dei pixel. Anche se due punti risultano sovrapposti non viene causato nessun errore di visualizzazione.

In seguito vengono inseriti in una lista tutti gli oggetti vicini (pianeti, satelliti, comete, ecc...); la lista viene poi ordinata in modo decrescente rispetto alla distanza. Gli oggetti sono disegnati, col depth buffer abilitato, secondo l'ordine della lista.

4.1.6 Conclusioni

Il file binario che Celestia utilizza come database di stelle, è molto comodo da usare, in quanto sono già filtrate le informazioni strettamente necessarie, evitando dati non utili ai nostri fini che comporterebbero solo un aumento del volume delle informazioni da trattare senza apportare benefici.
Al termine di questa fase abbiamo scelto, per l'implementazione del progetto "De Astris", di utilizzare proprio questo catalogo stellare; più precisamente si è adottato il file binario contenente il catalogo Tycho, che è costituito da 2.072.783 stelle e che comprende al suo interno anche le stelle del catalogo Hipparcos.

4.2 Scrittura della documentazione

Dopo il periodo di reverse engeneering su Celestia, è seguita una fase di ricerca e di studio sui meccanismi che stanno alla base dell'universo e sui fondamenti dell'astronomia e dell'astrofisica. Dal momento che la nostra conoscenza dell'universo in campo scientifico era inizialmente piuttosto superficiale, si è dovuto dedicare molto tempo a ricercare materiale utile ai fini dell'apprendimento e della conoscenza in questo campo. Era infatti indispensabile imparare le basi dell'astronomia e dell'astrofisica per poter portare avanti il nostro progetto in modo adeguato. Le ricerche sono state effettuate principalmente su Internet e su testi dedicati agli argomenti in questione.

Durante questa fase di studio, abbiamo prodotto una documentazione di oltre 300 pagine [Sito2], il tomo "De Astris", che andrà ad arricchire l'"Enchiridion", una biblioteca multimediale sviluppata e aggiornata dalla Fondazione. Un estratto di questa documentazione è presentato nel capitolo 2 come breve relazione introduttiva ai temi trattati.

Gli argomenti sui quali ci si è concentrati hanno spaziato dalla struttura generale dell'universo fino ai singoli pianeti nel particolare.

I primi due capitoli della documentazione sono introduttivi e delineano il panorama all'interno del quale si colloca l'approfondimento successivo dell'universo. In particolare, il primo capitolo fornisce nozioni di astronomia indispensabili nella trattazione degli argomenti successivi. Oltre alle definizioni e alle grandezze utilizzate in ambito scientifico, uno spazio rilevante è occupato dai sistemi di coordinate astronomiche utilizzati per individuare nell'universo le galassie, le stelle, i pianeti e tutte le altre entità.

Il secondo capitolo, invece, analizza aspetti già tecnici che hanno a che fare con la suddivisione spaziale dell'universo, ai fini della sua implementazione in tre dimensioni. Questa sezione, dal titolo "Rappresentazione cellulare", stabilisce a grandi linee quale può essere la scomposizione dell'universo in livelli successivi dal più generale al più particolare, cioè partendo dal tutto fino ad arrivare alla Terra, ai suoi continenti ed, eventualmente, alle singole città del nostro pianeta. Questa analisi è stata condotta in modo superficiale e non approfondito. Infatti,

nel momento in cui abbiamo prodotto la documentazione, l'organizzazione da dare all'universo era ancora da discutere e da stabilire.

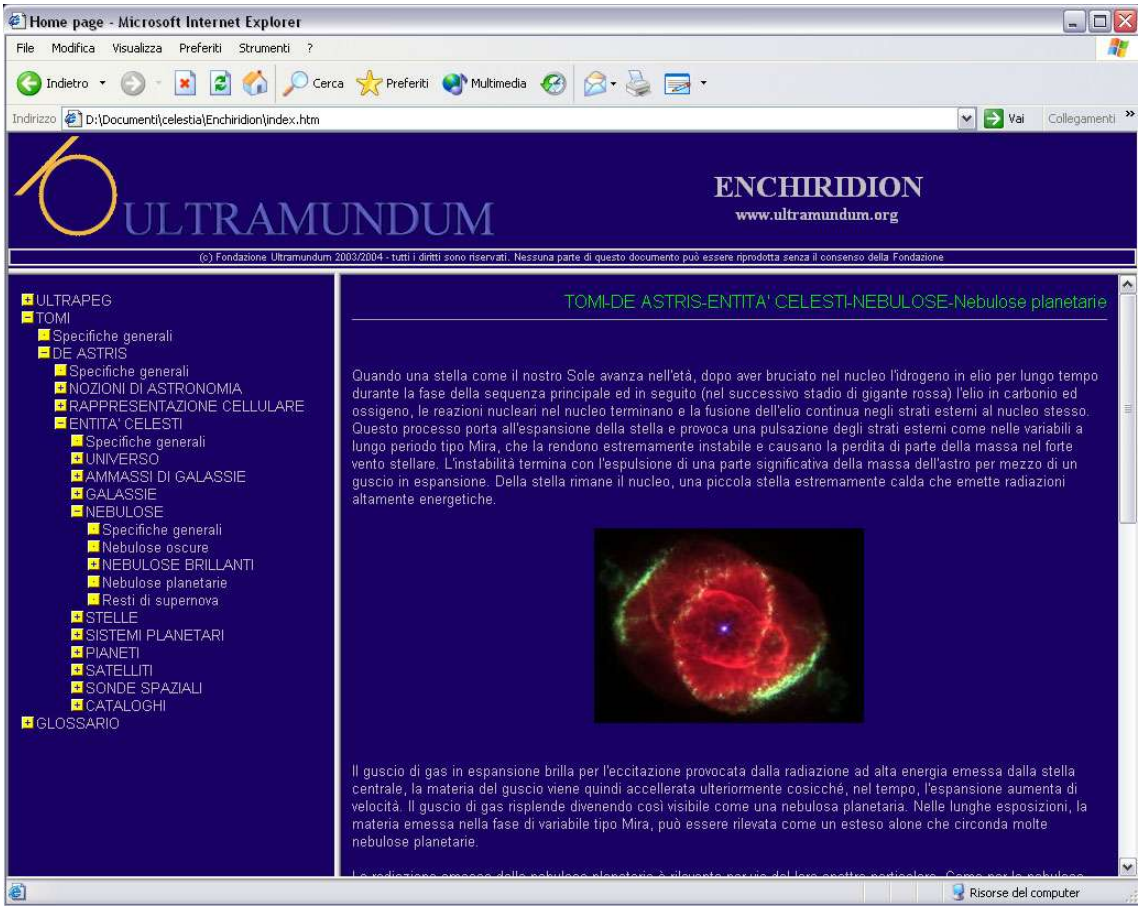


Figura 4.7: screenshot della documentazione del Tomo De Astris, inserito nell'Enchiridion della Fondazione Ultramundum.

Il capitolo successivo, che rappresenta il cuore del tomo "De Astris", è la sezione "Entità celesti". In essa sono stati sviluppati e approfonditi tutti gli argomenti e tutti i livelli accennati in precedenza, con un approccio dal generale al particolare.

La prima parte riguarda l'analisi dell'universo: le diverse teorie a proposito della sua origine, i meccanismi e i principi che lo governano, il suo destino e, per finire, la sua struttura e organizzazione.

La parte successiva è relativa alle galassie, con i loro moti, la classificazione e l'organizzazione spaziale, e agli ammassi e superammassi di galassie. Un'altra sezione è dedicata alle nebulose, anch'esse suddivise e classificate in base alle loro caratteristiche.

Il capitolo successivo analizza le stelle e le loro peculiarità: le classi spettrali, la magnitudine, la formazione, il ciclo di vita, il destino, i moti. Questa parte è stata sviluppata in dettaglio, dal momento che riguarda l'argomento fondamentale del progetto che abbiamo elaborato.
Un'altra sezione curata nei particolari è quella relativa ai sistemi planetari, con l'analisi del sistema solare e dei sistemi extrasolari. In questa parte, vengono analizzate le componenti che caratterizzano i sistemi planetari, vengono studiati i moti dei corpi celesti e vengono calcolate le orbite dei pianeti attraverso le leggi di Keplero e la Teoria delle Variazioni Secolari VSOP87.
Nei capitoli successivi, vengono approfonditi gli argomenti relativi ai pianeti del sistema solare e ai satelliti, naturali e artificiali, e alle sonde ruotanti intorno ad essi.
L'ultima parte è relativa ai cataloghi utilizzati dalla comunità scientifica per individuare le entità celesti all'interno dell'universo. I cataloghi sono suddivisi in cataloghi stellari e cataloghi di deep sky object (che contengono gli oggetti del cielo profondo, cioè tutto ciò che sta al di fuori delle stelle). Questo capitolo è stato approfondito nel dettaglio dal momento che l'analisi dei cataloghi è stata fondamentale per il nostro progetto. Infatti, l'obiettivo base del progetto è quello di importare stelle da database (cataloghi) e riprodurle in tridimensionale.
La documentazione scientifica prodotta in questa fase è stata resa di pubblico dominio ed è disponibile presso il sito della fondazione [Sito2].
Al termine di questa fase di ricerca esclusivamente teorica, abbiamo cominciato ad analizzare e ad utilizzare il software Ultraport, di proprietà della Fondazione, per iniziare a produrre le prime Tabulae di stelle e a visualizzarle nello spazio tridimensionale.

4.3 L'importatore

In questa fase è stata affrontata l'implementazione di un importatore, un modulo dell'"Invector", che permettesse di leggere i dati relativi alle stelle a partire dalle informazioni presenti sui cataloghi stellari utilizzati nella comunità scientifica. La Fondazione ha messo a nostra disposizione un esempio di importatore di base, che è in grado di supportare moduli diversi a seconda delle esigenze e dei dati che si vogliono importare. Questa versione di base, ancora mai usata nella realtà, è stata da noi ampliata durante lo svolgimento del progetto per meglio adattarlo alle reali esigenze di flessibilità nella creazione delle Tabulae.
L'Invector si compone in due distinte parti: l' interfaccia grafica e il modulo che implementa le funzioni per l'importazione dei dati.

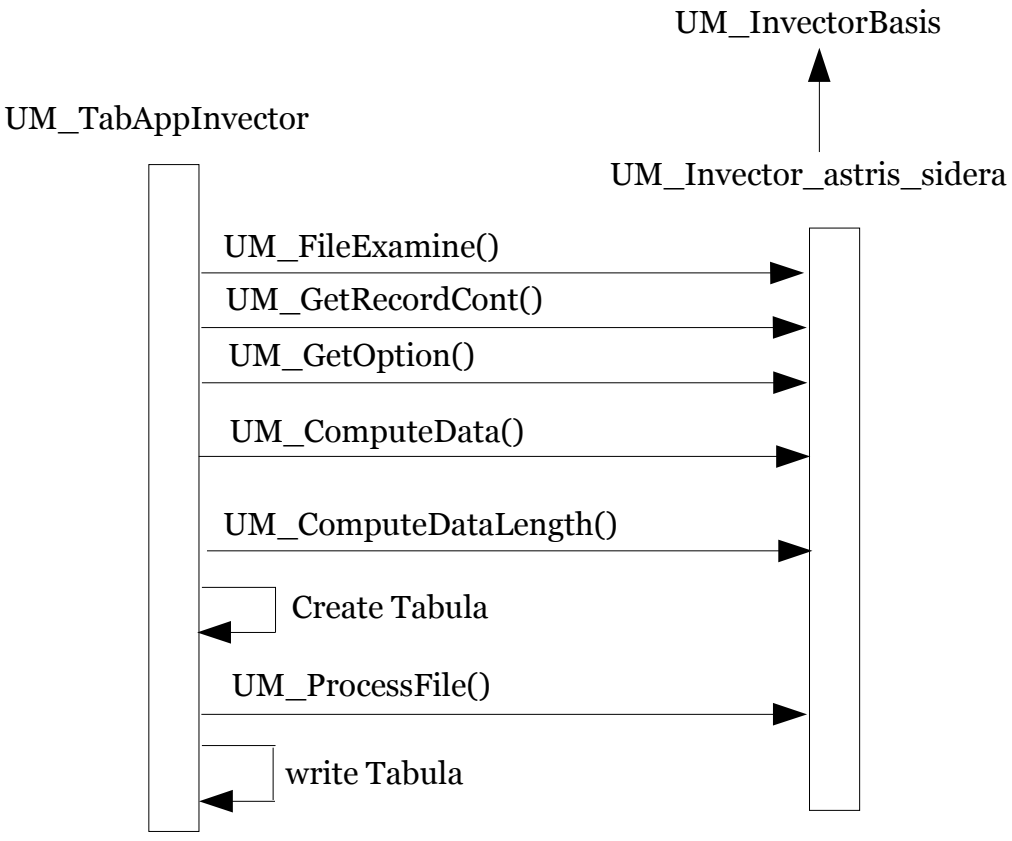


Figura 4.8: il diagramma rappresenta le principali chiamate dell'interfaccia grafica verso il modulo dell'Invector.

L'interfaccia grafica è una procedura che guida l'utente durante il processo di importazione dei dati e di creazione delle Tabulae; è realizzata dalla classe `UM_TabAppInvector`.

La classe `UM_InvectorBa`sis definisce tutte le funzioni virtuali da implementare se si vuole creare un importatore compatibile con le funzioni chiamate da `UM_TabAppInvector`. Il modulo implementato da noi prende il nome di `UM_Invector_astris_sidera`.

Il diagramma in Figura 4.8 illustra la sequenza dei passi e delle chiamate significative del processo di importazione.

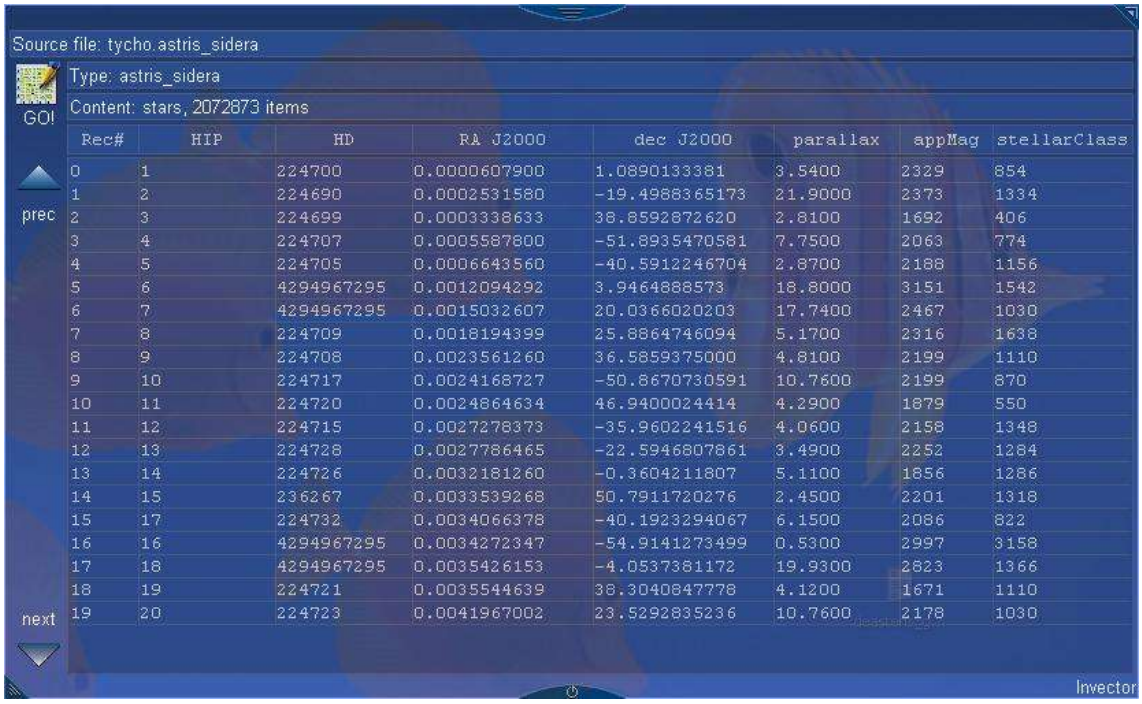
Di seguito viene spiegata la sequenza di passi svolti nel processamento; molte delle funzioni chiamate sono state aggiunte da noi, in quanto necessarie ma non presenti nella versione di base. L' Invector viene avviato dall'ambiente Ultraport; quella sottostante, la Figura 4.9 è un'immagine dell'interfaccia grafica per attivare tutte le applicazioni che compongono l'ambiente.

Figura 4.9: Adiutor, una interfaccia che permette di attivare tutte le applicazioni componenti di Ultraport

Una volta iniziata l'esecuzione del programma, deve essere selezionato il file sorgente che contiene i dati da importare. Questo file rappresenta il database che viene utilizzato per ricavare i dati da inserire nelle Tabulae.

Abbiamo precedentemente detto che l'Invector supporta moduli diversi di im-

portatori a seconda della tipologia dei dati da processare. `UM_TabAppInvector`, è in grado di risalire, partendo dal file selezionato, al tipo di dati contenuti e quindi al modulo al quale demandare il lavoro di processamento.
Una volta scelto il file, viene esaminato dalla funzione `UM_FileExamine()`, la quale oltre a leggere il numero di elementi presenti, deve preoccuparsi di rilevare tutte le informazioni necessarie per eventuali inizializzazioni.
Successivamente i dati presenti nel database vengono letti e stampati a video sotto forma di tabelle, per dare visione all'utente di cosa contiene il file che ha precedentemente selezionato. Nella prima pagina vengono mostrati i primi venti record, ma è possibile scorrerli tutti, se si vuole, mediante l'apposito bottone.



Rec#	HIP	HD	RA J2000	dec J2000	parallax	appMag	stellarClass
0	1	224700	0.0000607900	1.0890133381	3.5400	2329	854
1	2	224690	0.0002531580	-19.4988365173	21.9000	2373	1334
2	3	224699	0.0003338633	38.8592872620	2.8100	1692	406
3	4	224707	0.0005587800	-51.8935470581	7.7500	2063	774
4	5	224705	0.0006643560	-40.5912246704	2.8700	2188	1156
5	6	4294967295	0.0012094292	3.9464888573	18.8000	3151	1542
6	7	4294967295	0.0015032607	20.0366020203	17.7400	2467	1030
7	8	224709	0.0018194399	25.8864746094	5.1700	2316	1638
8	9	224708	0.0023561260	36.5859375000	4.8100	2199	1110
9	10	224717	0.0024168727	-50.8670730591	10.7600	2199	870
10	11	224720	0.0024864634	46.9400024414	4.2900	1879	550
11	12	224715	0.0027278373	-35.9602241516	4.0600	2158	1348
12	13	224728	0.0027786465	-22.5946807861	3.4900	2252	1284
13	14	224726	0.0032181260	-0.3604211807	5.1100	1856	1286
14	15	236267	0.0033539268	50.7911720276	2.4500	2201	1318
15	17	224732	0.0034066378	-40.1923294067	6.1500	2086	822
16	16	4294967295	0.0034272347	-54.9141273499	0.5300	2997	3158
17	18	4294967295	0.0035426153	-4.0537381172	19.9300	2823	1366
18	19	224721	0.0035544639	38.3040847778	4.1200	1671	1110
19	20	224723	0.0041967002	23.5292835236	10.7600	2178	1030

Figura 4.10: visualizzazione dei primi 20 record presenti nel database. In questo caso si tratta di stelle. Da sinistra a destra sono presenti i campi: N° del record, N° HIP, N° HD, ascensione retta, declinazione, parallasse, magnitudine apparente e classe stellare.

Proseguendo nella procedura, all'utente possono essere presentate più possibili scelte su come trattare i dati posseduti. Ciò a causa del fatto che gli stessi dati possono servire per fini diversi.
Dopo aver scelto l'operazione desiderata, sul modulo viene chiamata la funzione `UM_ComputeData()`, nel caso sia necessaria una computazione dei dati prima

della scrittura.
A questo punto `UM_TabAppInvector` chiede al modulo la grandezza della Tabula da creare, dopodiché la crea e chiama la funzione `UM_ProcesssFile()` che si occupa di copiare dentro questa i dati che ritiene opportuni. Infine la Tabula viene scritta nel Tabularium. Questi ultimi passi vengono ripetuti fino ad aver completato la lettura e la scrittura di tutti i dati. Per dare un feedback all'utente di ciò che sta accadendo, viene visualizzata una nuova pagina che fornisce un'indicazione sullo stato di avanzamento nella computazione, con una barra di progresso che indica la percentuale di avanzamento.

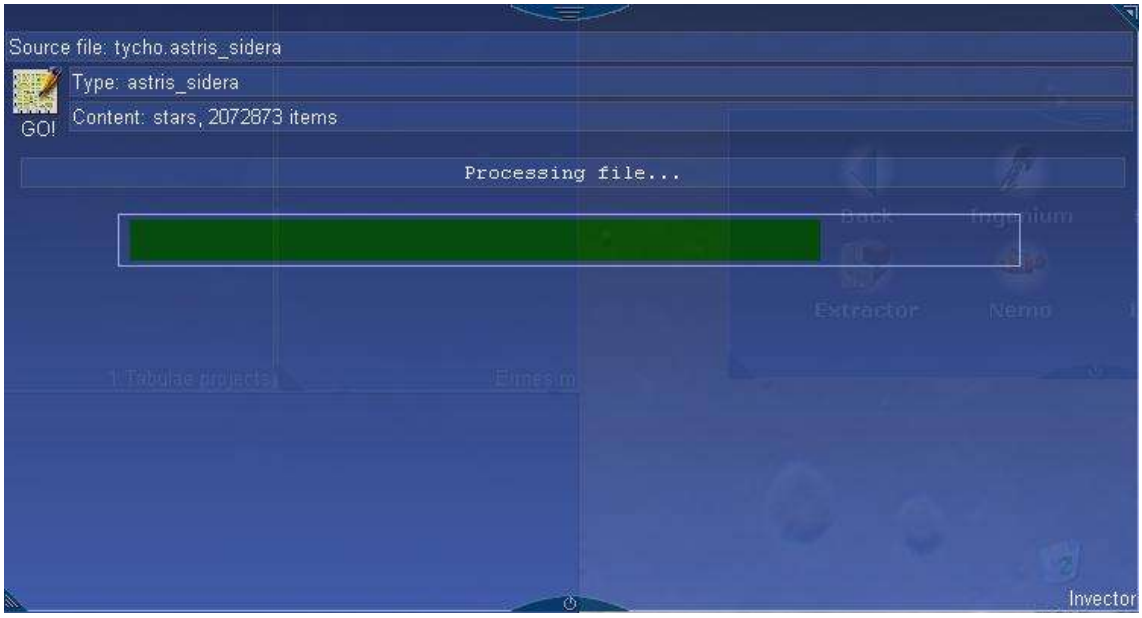


Figura 4.11: la barra di barra di avanzamento indica il progresso nella scrittura delle Tabulae.

Quando il processo è terminato, viene prodotta una pagina di report, nella quale sono visualizzate informazioni riassuntive sull'operazione appena conclusa, come il numero di Tabulae create, lo spazio totale su disco occupato.
Oltre a ciò viene fatta una statistica sulla distribuzione del numero di Tabulae per determinate fasce di spazio occupato, come si può vedere nella Figura 4.12.

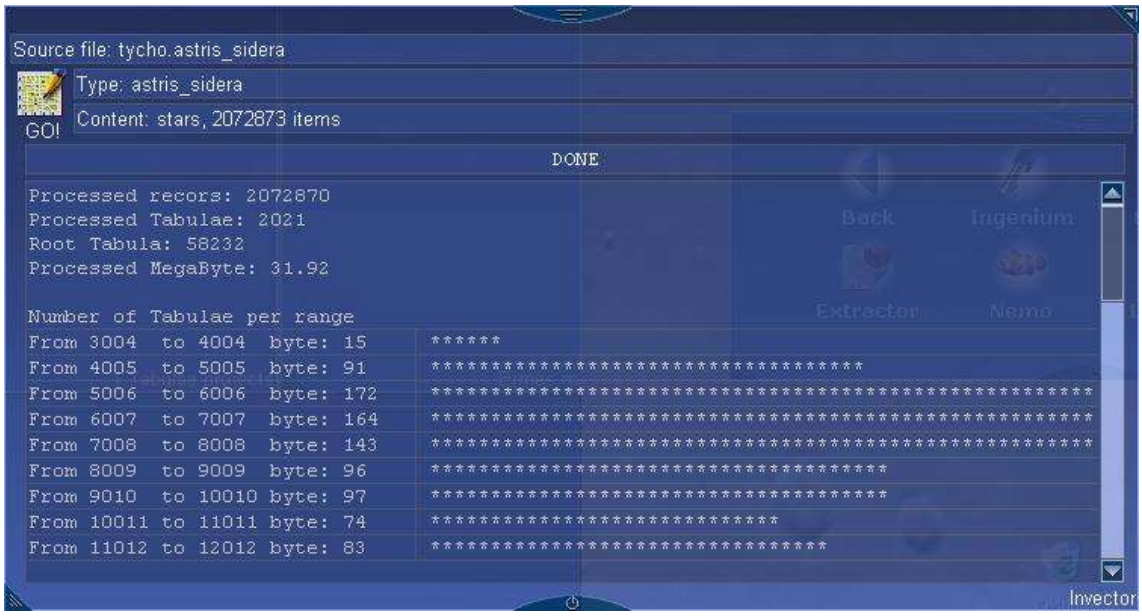


Figura 4.12: il report indica informazioni sul processamento appena finito, oltre a una statistica sulla distribuzione delle dimensioni delle Tabulae create. In questo caso riporta informazioni sull'importazione del catalogo stellare di Tycho.

4.4 Il primo importatore

Pertanto si è iniziato a scrivere il modulo dell'Invector, chiamato “UM_Invector_astris_sidera”, che dovrà creare le Tabulae contenenti tutte le informazioni delle quali necessitano le stelle.
Dopo aver scelto i due cataloghi stellari da utilizzare, i quali sono database di dati in forma binaria, si sono implementate le funzioni che questo modulo eredita da `UM_InvectorBasis` e che gli permettono di essere compatibile con le chiamate dell'interfaccia grafica `UM_TabAppInvector`.
I primi record del file vengono letti e inseriti in tabelle in tabelle contenenti 20 record ciascuna, come spiegato nel paragrafo precedente.
In questo modo è possibile visualizzare, scorrendo le pagine, l’archivio completo di stelle e i rispettivi dati.
I dati vengono visualizzati “grezzi”, cioè nello stesso modo in cui si trovano nel file binario, senza computazioni per trasformarli in quelli che servono realmente. In particolare, vengono importati: il n° catalogo HIP, il n° catalogo HD, l'ascensione retta (RA), la declinazione (dec), la parallasse, la magnitudine apparente e la classe stellare.
Scegliendo di proseguire, l'utente può scegliere quale operazione attivare sui dati in questione.
Ogni modulo di Invector fornisce opzioni diverse per processare i dati. Nel caso dell'”UM_Invector_astris_sidera”, l'unica azione consentita, in quanto non servono processamenti differenti, è quella che va sotto il nome di "Produce stars Tabulae", che si occupa di creare le Tabulae.
La prima versione dell’”UM_Invector_astris_sidera”, più rudimentale e provvisoria, si occupava di creare le Tabulae inserendo le stelle in maniera sequenziale, nello stesso ordine in cui venivano lette.
In particolare, gli astri erano caricati da file, impacchettati all’interno di strutture dati (`struct` del C++) che contenevano tutte le informazioni e infine raccolti a formare Tabulae di dati.

Le struct di stelle incapsulavano le seguenti variabili:

```
- int hipCatalog     :  numero del catalogo HIP
- int hdCatalog      :  numero del catalogo HD
- float x            :  valore della coordinata spaziale x
- float y            :  valore della coordinata spaziale y
- float z            :  valore della coordinata spaziale z
- float appMag       :  il valore della magnitudine apparente
- short stellarClass :  numero che corrisponde alla classe stellare
```

Le Tabulae consistevano pertanto in una sequenza di `struct` ed erano caratterizzate da un header che aveva al suo interno metainformazioni riguardanti la Tabula stessa: il suo numero di codice identificativo, il suo tipo e la sua dimensione in byte.
Come prima approssimazione, ogni Tabula conteneva un numero di stelle pari a 100. Considerando che ogni stella occupava uno spazio di 26 byte, le dimensioni di ogni Tabula, escludendo l'header, erano quindi di 2600 byte l'una.
Nel caso del nostro Invector, venivano inseriti nel report i seguenti dati: il numero di stelle processate e il numero di Tabulae create, che ammontava a 1126.
Sebbene l'idea in questa fase fosse soltanto quella di testare la scrittura di alcune Tabulae e la rappresentazione di queste nello spazio tridimensionale, si è potuto già intuire che l'organizzazione nella scrittura delle Tabulae avrebbe dovuto attraversare una fase di discussione per valutarne l'effettiva efficienza e una successiva di pianificazione, per mettere in atto le scelte fatte.
C'è da premettere inoltre che, dal momento che l'ambiente Ultraport è ancora in fase di sviluppo, l'implementazione del progetto "De Astris" è stata la prima occasione nella quale le Tabulae venivano generate col meccanismo dell'importatore. Per questo motivo, non ci si è potuti adeguare a standard pre-esistenti ed è stato necessario un periodo di testing a tal proposito.
Inoltre fino a quel momento non si era ancora trattato il problema della suddivisione spaziale dell'universo, dell'organizzazione spaziale delle Tabulae al suo interno e delle possibili ottimizzazioni per ridurre le dimensioni occupate in memoria.

4.5 La prima visualizzazione

La fase successiva ha avuto come unico scopo quello di iniziare ad avere familiarità con il motore grafico; si è proceduto pertanto a creare una funzione che, aprendo e leggendo le Tabulae appena scritte, disegnasse delle stelle. A questo fine, si è utilizzata un'altra applicazione messa a disposizione dalla Fondazione Ultramundum, il Nucleus. Questa parte è stata implementata provvisoriamente e sperimentalmente nel Nucleus, in quanto Ultraport è ancora in fase di sviluppo e al momento dell'implementazione non erano ancora pronti gli ambienti propri nei quali inserire questo codice.
All'interno del Nucleus, infatti, è possibile creare vertici, triangoli e, più in generale, poligoni che, una volta inseriti in nodi che vengono "attaccati" allo Scene Graph, verranno poi renderizzati. I nodi sono strutture dati che contengono le primitive grafiche e il materiale dello stesso.
In questo caso, la visualizzazione delle stelle è stata poco curata e molto pratica, dal momento in cui l'unico obiettivo è stato quello di iniziare a visualizzare per la prima volta le Tabulae in tre dimensioni.
La prima versione del Nucleus si occupava di aprire e leggere le Tabulae, di scorrere le stelle presenti al loro interno, creare vertici in corrispondenza delle coordinate x, y, z di ogni astro e infine, disegnare triangoli che corrispondevano alle stelle presenti nell'universo.
In un primo momento, le stelle sono state rappresentate sotto forma di triangoli nei quali la posizione esatta era associata al vertice alto del triangolo. La rappresentazione a triangoli presentava, tuttavia, il problema del "culling": infatti quando un poligono viene disegnato, gli viene assegnata dal sistema una normale alla superficie. La normale punta, secondo la regola della "mano destra", nella direzione del pollice quando i vertici vengono dati nell'ordine in cui ruotano le dita. Per default il poligono è visibile solo dalla parte verso cui è indirizzata la normale, mentre se lo si guarda dall'altro lato esso non appare, in quanto non viene disegnato. Pertanto, con questa rappresentazione le stelle erano visibili solamente di fronte, mentre volgendo lo sguardo all'indietro sparivano tutte.
Per ovviare a questo problema, si è scelta una rappresentazione a cubi, nei quali

la posizione reale della stella corrispondeva al centro del cubo. Il cubo possiede sei facce, quindi in ogni momento ce n'è sempre una rivolta verso l'osservatore. Ovviamente il numero di triangoli da disegnare per questa rappresentazione è cresciuto notevolmente. Infatti, per disegnare un cubo sono necessari due triangoli per ogni faccia; ogni stella è così costituita da dodici triangoli.
Proprio per questo motivo, si è notato un notevole calo di prestazioni e di velocità nella rappresentazione tridimensionale delle stelle. Passando infatti da uno a dodici triangoli disegnati per ogni stella, il tempo necessario per visualizzarle nello spazio è cresciuto, così come la fluidità della navigazione tra le stelle presenti nell'universo.
Di seguito sono riportate alcune immagini che documentano tutte le varie prove di visualizzazione fatte in questa fase.

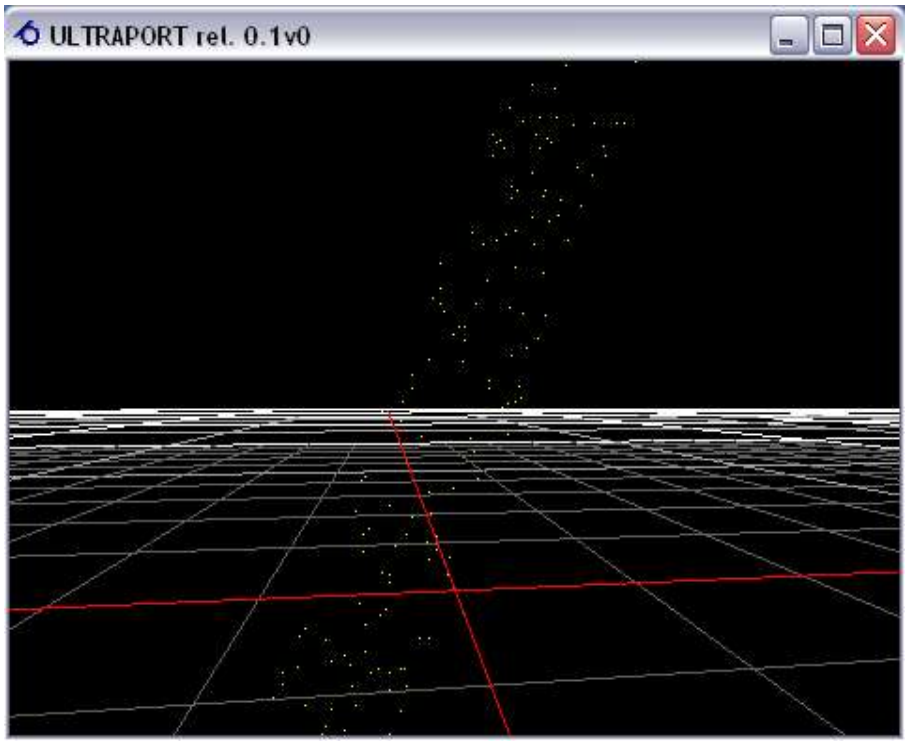


Figura 4.13: rappresentazione delle stelle contenute in una singola Tabula.

Le stelle nei cataloghi Hipparcos e Tycho sono ordinate in modo crescente per ascensione retta. Queste sono inserite nelle Tabulae sequenzialmente, nell'ordine in cui si trovano nel catalogo. Pertanto ogni Tabula rappresenta uno spicchio

di spazio, cioè un intervallo di RA all'interno del quale sono situate le stelle.
Nella Figura 4.13 sono disegnati soltanto i dati contenuti in una Tabula. Ciò per velocizzare la renderizzazione: disegnare i dati di 1126 Tabulae comportava un quarto d'ora di tempo. I puntini visibili nell'immagine sono i cubi alla reale distanza in anni luce delle stelle che rappresentano. I cubi a tali lontananze prendono la forma di piccoli punti. Agli occhi dell'osservatore che si muove nello spazio con una velocità nell'ordine dei km/h, le stelle appaiono fisse sulla volta celeste, è impossibile raggiungerle; una visualizzazione dello spazio simile a quello che vediamo noi dalla Terra.

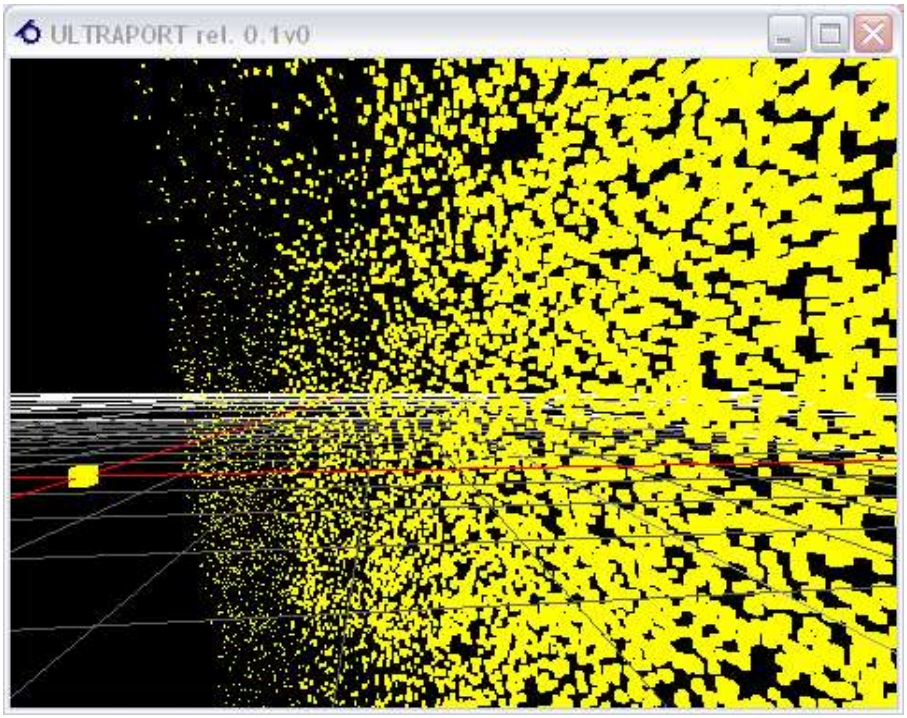


Figura 4.14: la stella visibile più a sinistra, al centro degli assi, è il Sole. In questa visualizzazione, l'unità di misura usata per le distanze è stata quella dei centimetri.

Nella Figura 4.14 è possibile vedere il Sole, posto a sinistra dell'immagine; è il cubo sull'origine degli assi. Si può notare la stratificazione delle stelle nello spazio allontanandosi dal Sole. Ciò succede perché la lunghezza dello spigolo di ogni cubo è funzione della distanza dal nostro astro. Questa volta le distanze interstellari sono espresse in centimetri, per questo gli astri risultano così ammassati. E' perciò possibile "navigare" tra gli astri, compiendo un "viaggio spaziale".

Per rendere più realistica la rappresentazione dei cubi, si è deciso di sfumare due spigoli verso l'arancio. Questo ha contribuito ad aumentare l'effetto tridimensionale delle stelle nella visualizzazione grafica.

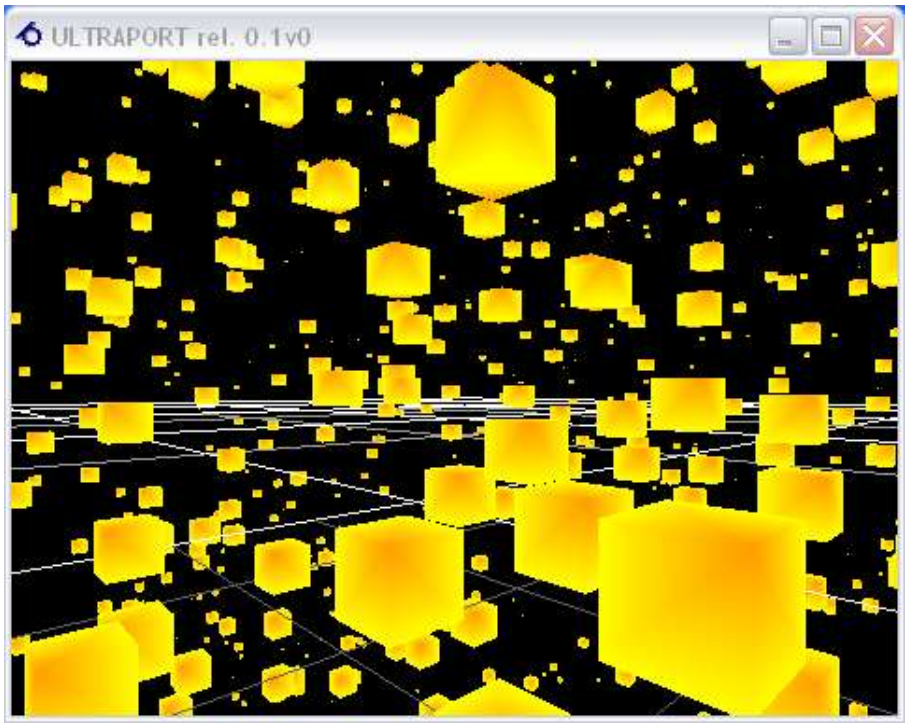


Figura 4.15: le stelle hanno dimensioni enormi in rapporto alla loro distanza.

Non tutti gli errori vengono per nuocere, infatti in seguito ad una errata impostazione della dimensione dei cubi (spropositata) si è avuta la visione rappresentata nelle figure 4.15 e 4.16.
Vicino all'origine, gli astri sono così densi da sovrapporsi, creando uno spettacolare effetto quando ci si sposta tra questi, ma che attraverso le immagini viene risaltato solo in piccola parte.
Per rendere navigabile l'universo, cercando però di non allontanarci troppo dalla realtà si è passati a una rappresentazione utilizzando come unità di misura per le distanze i metri.

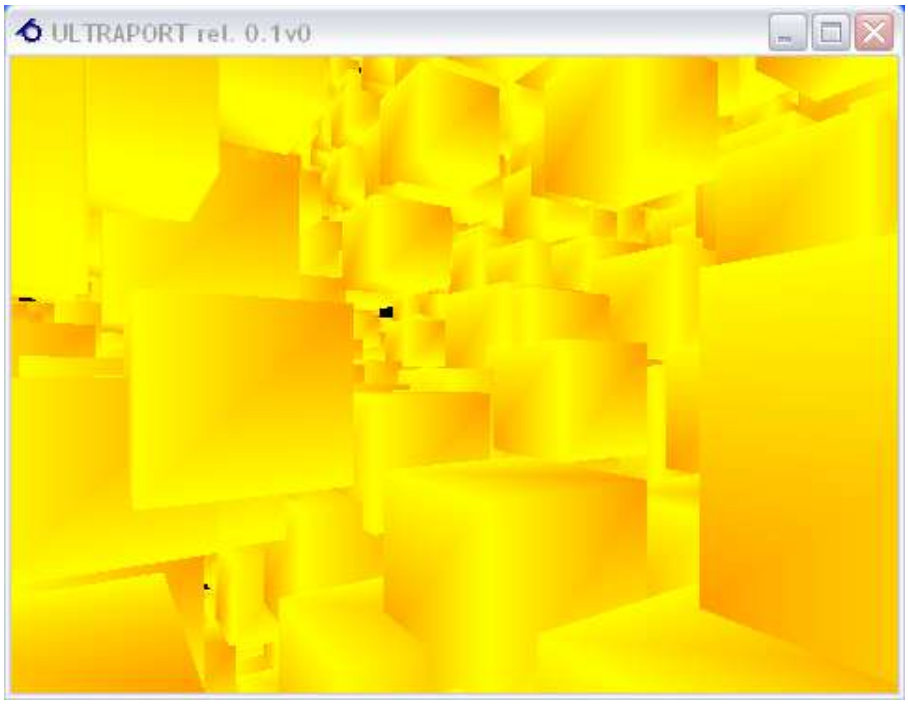


Figura 4.16: si è completamente sommersi dalle stelle.

In Figura 4.17 si può notare il nostro Sole situato nell'origine; gli assi sono raffigurati in rosso. Le stelle vicine sono visibili come grossi cubi. Allontanando il punto di osservazione, a causa della distanza, i cubi sono percepiti di dimensioni sempre minori, fino a diventare puntini.

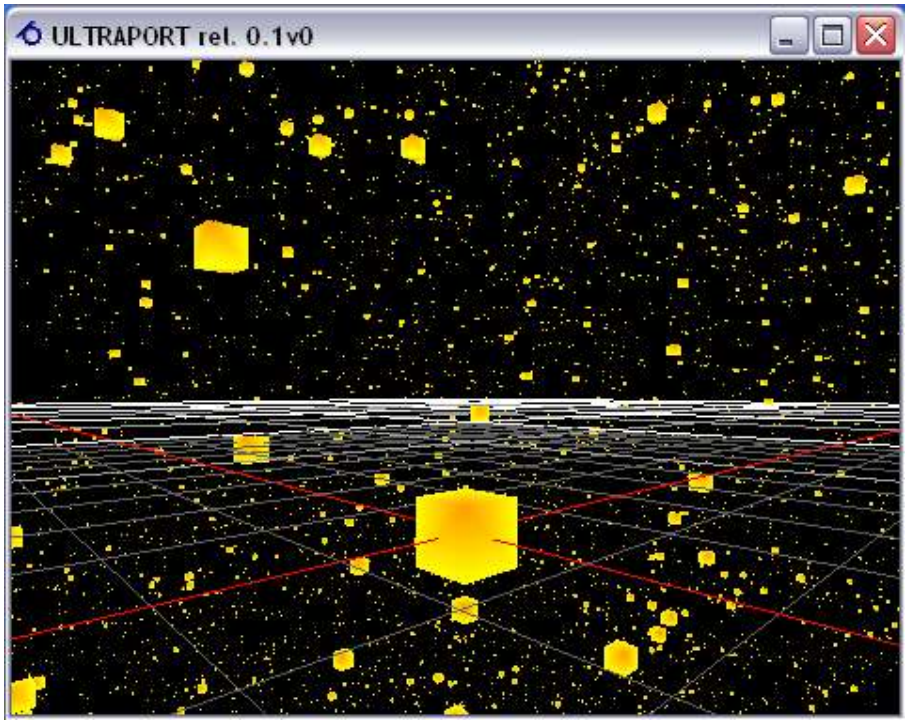


Figura 4.17: Da notare, al centro degli assi disegnati in rosso, il Sole.

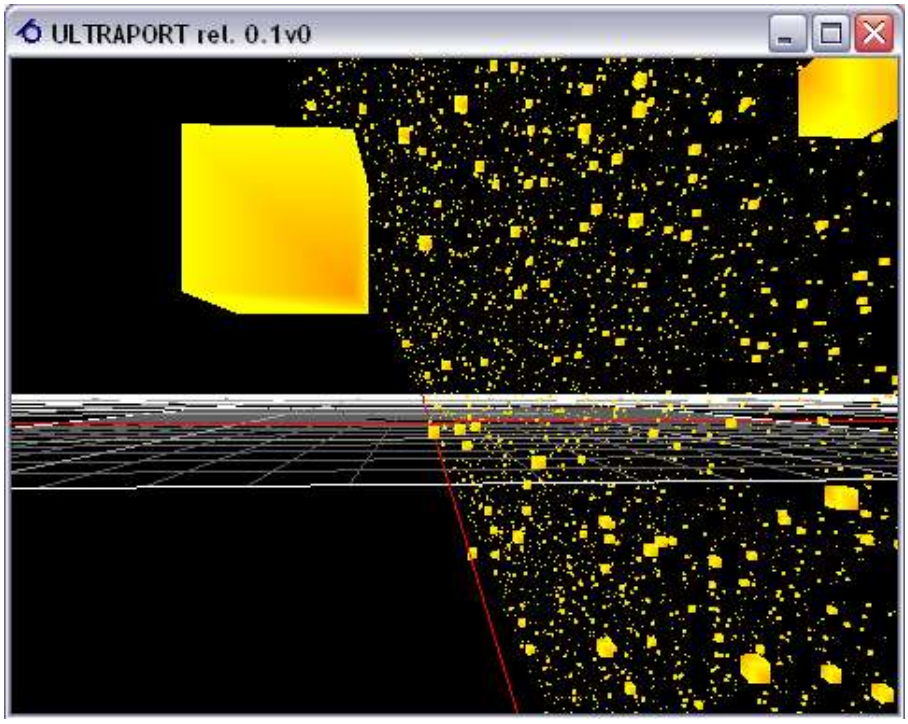


Figura 4.18: in primo piano, una stella sul limite dello “spicchio”.

La rappresentazione “a spicchi” delle Tabulae è evidente in Figura 4.18, nella quale sono state visualizzate all'incirca la metà delle Tabulae del catalogo. Per questo motivo, le stelle inserite nello spazio sono quelle che hanno una ascensione retta compresa tra 0° e 180°.

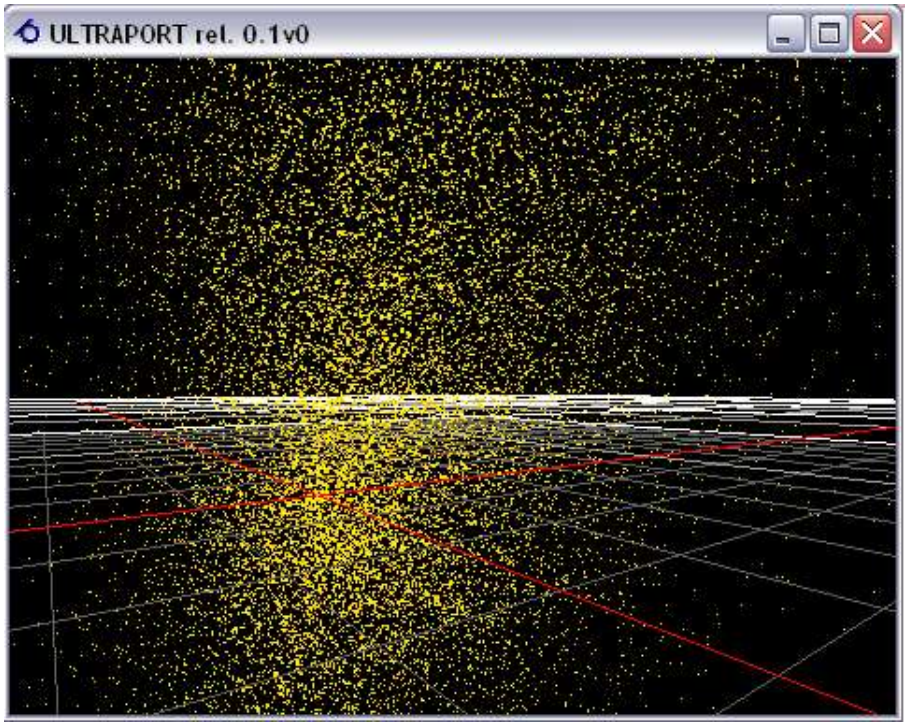


Figura 4.19: una visione più reale dell'universo.

E' stato sufficiente ridurre le dimensioni delle stelle per ottenere un universo molto più realistico e bello da navigare, come si può notare in Figura 4.19

4.5.1 Array di vertici e di indici

Ci sono aspetti riguardanti limitazioni imposte dall'hardware, che abbiamo dovuto prendere in considerazione durante queste visualizzazioni. Infatti, a livello di renderizzazione, c'è un numero di vertici o di triangoli (un triangolo corrisponde a tre vertici) massimo che possono essere passati all'acceleratore grafico in una sola chiamata. Questo numero, normalmente compreso tra i 30 e i 40 mila vertici, non deve essere superato per evitare che i triangoli non vengano disegnati. E' per questo motivo che si è cominciato a valutare quale potesse essere l'oggetto grafico migliore da disegnare per rappresentare le stelle. Questa discussione, tuttavia, è stata rimandata alla fase successiva, cioè a quella vera e propria di implementazione.

Ogni vertice può essere disegnato singolarmente chiamando la funzione apposita, tuttavia quando si tratta di un numero così alto, ciò farebbe decadere le prestazioni. Quindi si utilizza una tecnica differente: i vertici vengono messi in un array e poi trattati tutti quanti assieme.

Per disegnare un triangolo servono tre vertici, per un cubo sono necessari dodici triangoli, quindi trentasei vertici; ma un cubo è composto solamente da otto vertici. Per ottimizzare ulteriormente si utilizzano più array: uno per i vertici e uno per gli indici. Nel primo array vengono messi solamente gli otto vertici necessari per dare forma al cubo, ma più in generale gli n vertici che compongono l'insieme dei poligoni. Nel secondo si assegnano gli indici ai vertici del primo vettore; l'ordine nel quale sono memorizzati corrisponde a quello con cui verranno renderizzati. L'acceleratore grafico disegna i poligoni seguendo l'array degli indici; per ognuno di essi accede al vettore dei vertici alla posizione indicata dall'indice. Questo metodo permette quindi di riutilizzare più volte gli stessi vertici per creare triangoli diversi.

Questa prima versione del Nucleus, che è stata molto utile in fase di testing delle Tabulae, ha messo anche in mostra aspetti importanti che dovevano essere discussi e trattati prima di implementare la versione definitiva dell'Invector e del

Nucleus. Sebbene la trattazione sequenziale delle Tabulae e delle stelle fosse parsa inefficiente fin dall'inizio, le prestazioni che il sistema ha offerto in fase di renderizzazione ha tolto ogni dubbio. Infatti il tempo necessario perché fossero visualizzate tutte le stelle del catalogo Hipparcos (112522 stelle in 1126 tabulae) era troppo elevato, così come la navigazione tridimensionale attraverso i cubi era lenta e a scatti. Proprio per questo motivo, si è incominciato a discutere a proposito della struttura dell'universo e dell'organizzazione delle Tabulae secondo criteri più efficienti.

4.6 Suddivisione spaziale dell'universo

Le Tabulae vanno create in modo dinamico, non statico come fatto in precedenza. La prima versione dell'Invector va quindi modificata in modo tale che le Tabulae non siano costruite prendendo sequenzialmente le stelle dal database, ma applicando un'euristica alla ricerca. I criteri secondo cui quest'euristica applica la selezione sono la posizione spaziale delle stelle all'interno della galassia e la loro magnitudine assoluta.
Durante l'esecuzione del programma gli astri non dovranno essere visualizzati tutti in ogni momento, ma solo quelli visibili all'osservatore, dipendenti dalla sua posizione. Quelli troppo distanti in relazione alla loro luminosità non devono essere presi in considerazione. Ciò è anche realistico, in quanto l'occhio umano, e anche i telescopi, hanno dei limiti. Ha anche l'effetto di rendere invisibile una grossa fetta di universo, con un notevole miglioramento delle prestazioni; questo è il metodo utilizzato anche in Celestia.
La creazione dinamica delle Tabulae secondo i criteri sopra esposti si è rivelata vantaggiosa anche dal momento che permette estensioni future del catalogo stellare senza troppo pesanti ripercussioni sulle prestazioni del sistema e sull'efficienza delle funzioni che creano le Tabulae.
Un altro aspetto importante doveva essere valutato in vista dell'organizzazione da dare all'universo. Infatti, poiché le Tabulae sono finalizzate principalmente al trasferimento via Internet, è stata fissata per ogni Tabula una dimensione massima, che nel nostro caso è di 65536 byte comprensivi dell'header e dei dati, che non può essere superata. Questo aspetto ha valore anche dal punto di vista dell'efficienza del sistema, poiché l'universo e i corpi celesti al suo interno vengono gestiti “a moduli”; questo rappresenta una delle differenze fondamentali rispetto a Celestia, che utilizza l'insieme degli astri come un unico blocco.
Prima di approvare la scelta definitiva sull'organizzazione dell'universo, abbiamo attraversato una fase di studio e di analisi statistica per valutare e rendersi conto della distribuzione spaziale delle stelle presenti nei cataloghi Hipparcos e Tycho.

4.6.1 Analisi statistica

Per giungere a una suddivisione spaziale efficiente, capace di soddisfare le esigenze, si è iniziata un'analisi dei cataloghi Hipparcos e Tycho. L'obiettivo è capire meglio la distribuzione delle stelle nello spazio, capirne la concentrazione a una determinata distanza dal Sole (e quindi dalla Terra) e la loro quantità in una determinata area o in una fascia. Questa ricerca statistica si è resa indispensabile per trovare la struttura dati più adeguata in grado di memorizzarle. Ovviamente la concentrazione è massima nelle vicinanze del Sole e decresce uniformemente man mano che ci si sposta verso l'esterno. Questo fenomeno è dovuto al fatto che con l'allontanarsi dalla Terra cresce la difficoltà di osservazione e quindi il numero di oggetti osservati.

Si è realizzato un semplice programma in C che legge il catalogo di stelle, per ognuna di esse calcola la distanza in base alla parallasse e le raggruppa in fasce concentriche di grandezza fissata.

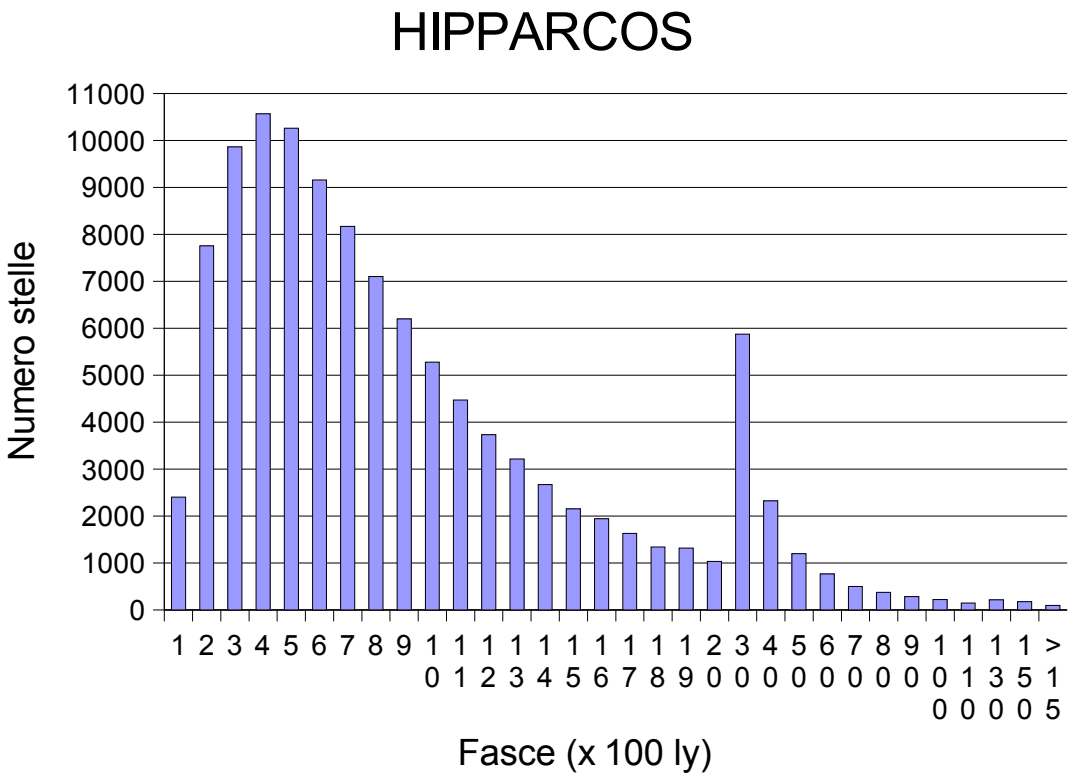


Figura 4.20: stella a distanza massima: 16308,35 anni luce. Le distanze sono espresse in anni luce (ly).

Nei diagrammi nelle figure 4.20 e 4.21 è rappresentato il confronto tra i cataloghi di Hipparcos e di Tycho nella distribuzione spaziale delle stelle.

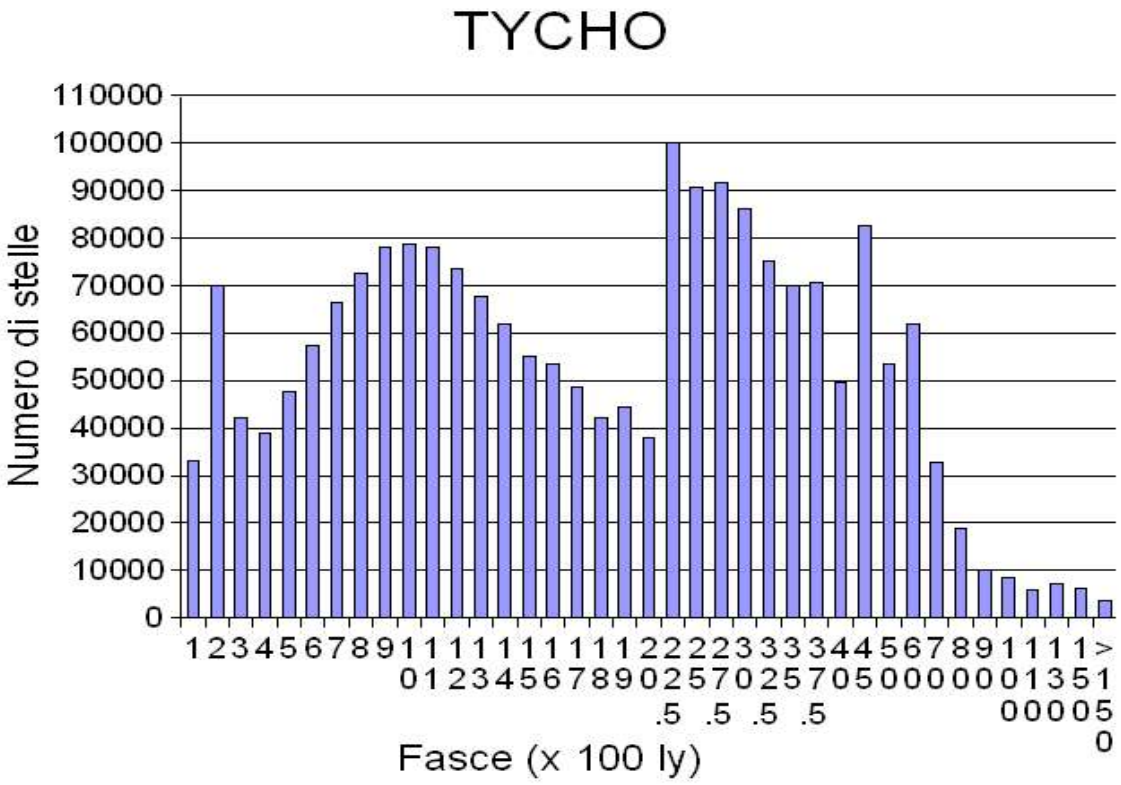


Figura 4.21: stella a distanza massima: 29651,55 anni luce. Le distanze sono espresse in anni luce (ly).

In questa analisi statistica, lo spazio è stato suddiviso in sfere concentriche.
Come si può notare, nel catalogo di Hipparcos la concentrazione diminuisce linearmente con l'allontanarsi dal centro. Nel secondo caso invece, data la completezza del catalogo, decresce meno velocemente.
Bisogna tenere conto del fatto che, anche se più fasce hanno la stessa larghezza (e.g. 100 anni luce), quelle a maggiore distanza risultano più vaste di quelle vicine, perché occupano un volume più grande. Quindi la diminuzione della densità è in realtà più rapida di quella indicata dai grafici, nei quali non è tenuto conto del volume.
Nelle due immagini si può notare (spostandosi verso destra), una improvvisa impennata. Ciò è dovuto al fatto che, per motivi di spazio, la grandezza degli intervalli cresce all'aumentare della distanza dal sole. Pertanto, le prime fasce han-

no ampiezza 100 anni luce, quelle successive hanno ampiezza di 250 o 1000 anni luce.
Per conoscere la reale densità delle stelle presenti nel catalogo Tycho, cioè quanto densamente è occupata ogni fascia, si è fatta un'ulteriore ricerca che ha portato a realizzare il seguente grafico:

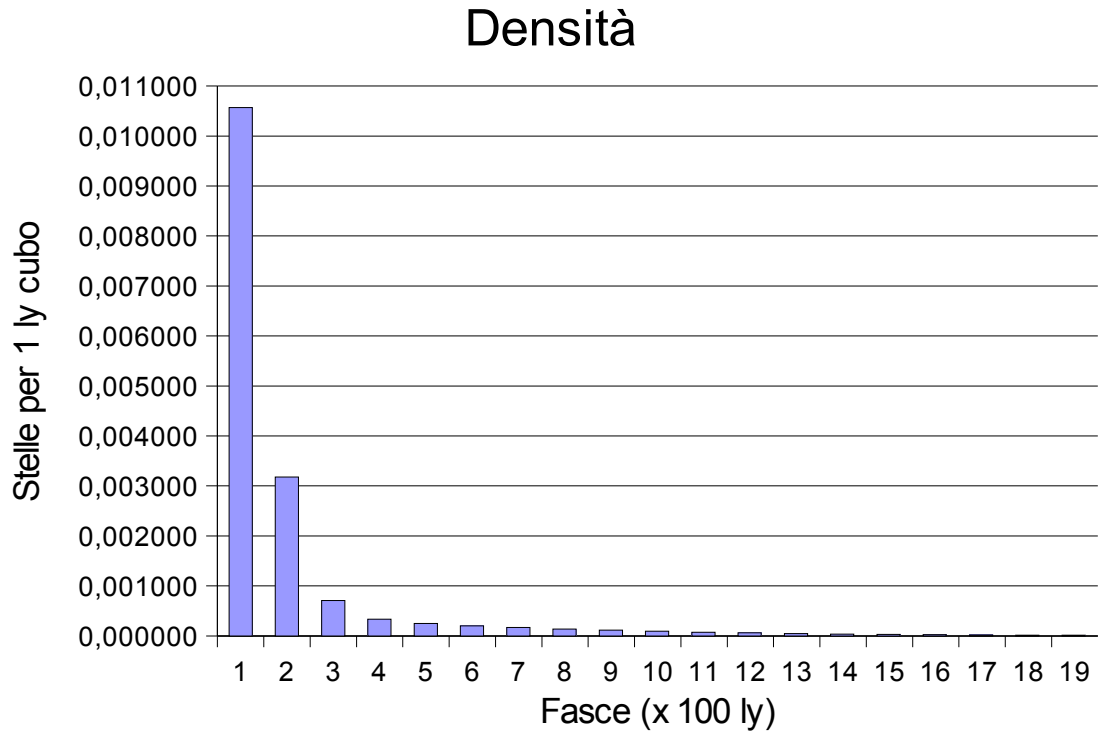


Figura 4.22: le distanze sono espresse in anni luce (ly).

Si nota come, ad esempio, nella prima fascia, da 0 a 100 anni luce di distanza dal sole, ci siano circa 0,01 stelle per un volume cubico di 1 anno luce di spigolo. Si può vedere che la concentrazione decresce molto rapidamente allontanandosi dal sole. Il grafico non va oltre i 1900 anni luce di distanza in quanto la densità per quei valori è prossima allo 0. I dati sono presi dal catalogo di Tycho.
L'obiettivo del progetto è la rappresentazione del catalogo Tycho; nonostante ciò, verrà spesso utilizzato anche il catalogo Hipparcos, che grazie alle sue più ridotte dimensioni permette di velocizzare simulazioni ed esecuzione di prove.
E' quindi necessario trovare una struttura che sia efficiente indipendentemente dal numero di stelle da memorizzare.

4.6.2 Scelta finale

Le possibili modalità di organizzazione spaziale che sono state poste al vaglio in questa fase sono state essenzialmente tre: una struttura “a sfere concentriche”, un'altra utilizzando una griglia di cubi della stessa dimensione e, infine, una struttura ad albero di tipo octree.
La prima modalità si è rivelata troppo geocentrica e pertanto poco efficiente nel momento in cui un osservatore avesse dovuto allontanarsi dalla Terra per navigare nello spazio circostante; la seconda possibilità, invece, ha un limite nell'ottimizzazione in quanto la grandezza dei cubi non è proporzionale alla densità delle stelle. Come si evince dall'analisi statistica condotta in precedenza, con la scelta di questa struttura si rischierebbe di avere cubi molto densi di stelle in prossimità del sole e altri vuoti nelle zone più distanti da esso.
Per tutta questa serie di motivi, la scelta finale è caduta su un'organizzazione dell'universo con una struttura ad albero, in cui ogni nodo o foglia rappresenta un cubo che corrisponde a una porzione di spazio. Nella letteratura un albero di questo tipo viene detto octree, in cui l'oggetto di partenza viene scomposto in celle.

4.6.3 Scomposizione ed enumerazione spaziale delle celle

Si consideri un oggetto comune, per esempio una tazzina di caffè, e si immagini di scomporla in pezzi separati, cosicché ogni pezzo della scomposizione finale sia più semplice da descrivere della tazzina di partenza. E' possibile effettuare la scomposizione di una o più parti (il manico, il fondo, ...) fino a raggiungere una predeterminata approssimazione di descrivibilità [5].
Il procedimento descritto prende il nome di "scomposizione in celle": ogni solido può essere rappresentato come la somma o l'unione di un insieme di celle nelle quali esso è diviso. Tale metodo di rappresentazione è utile in tutti i casi in cui l'oggetto intero può non essere semplice da rappresentare, ma esserlo attraverso una sua scomposizione. Esistono molti modi per scomporre un solido in celle componenti; nessuno di essi è unico, ma tutti non sono ambigui.
L'"enumerazione per occupazione spaziale" rappresenta un caso particolare del-

la scomposizione in celle, in cui queste ultime sono cubiche nella forma e collocate in una griglia spaziale fissata; se si diminuiscono sempre più le dimensioni del cubo, tale metodo approssima la rappresentazione di un solido come insieme di punti contigui nello spazio. La definizione di un solido attraverso l'enumerazione per occupazione spaziale richiede un metodo efficace di rappresentazione dell'insieme delle celle cubiche; uno tra i più semplici consiste nell'elencare le coordinate dei centri delle celle; un oggetto solido è pertanto visto come un insieme di celle adiacenti, la cui dimensione determina la massima risoluzione del modello.

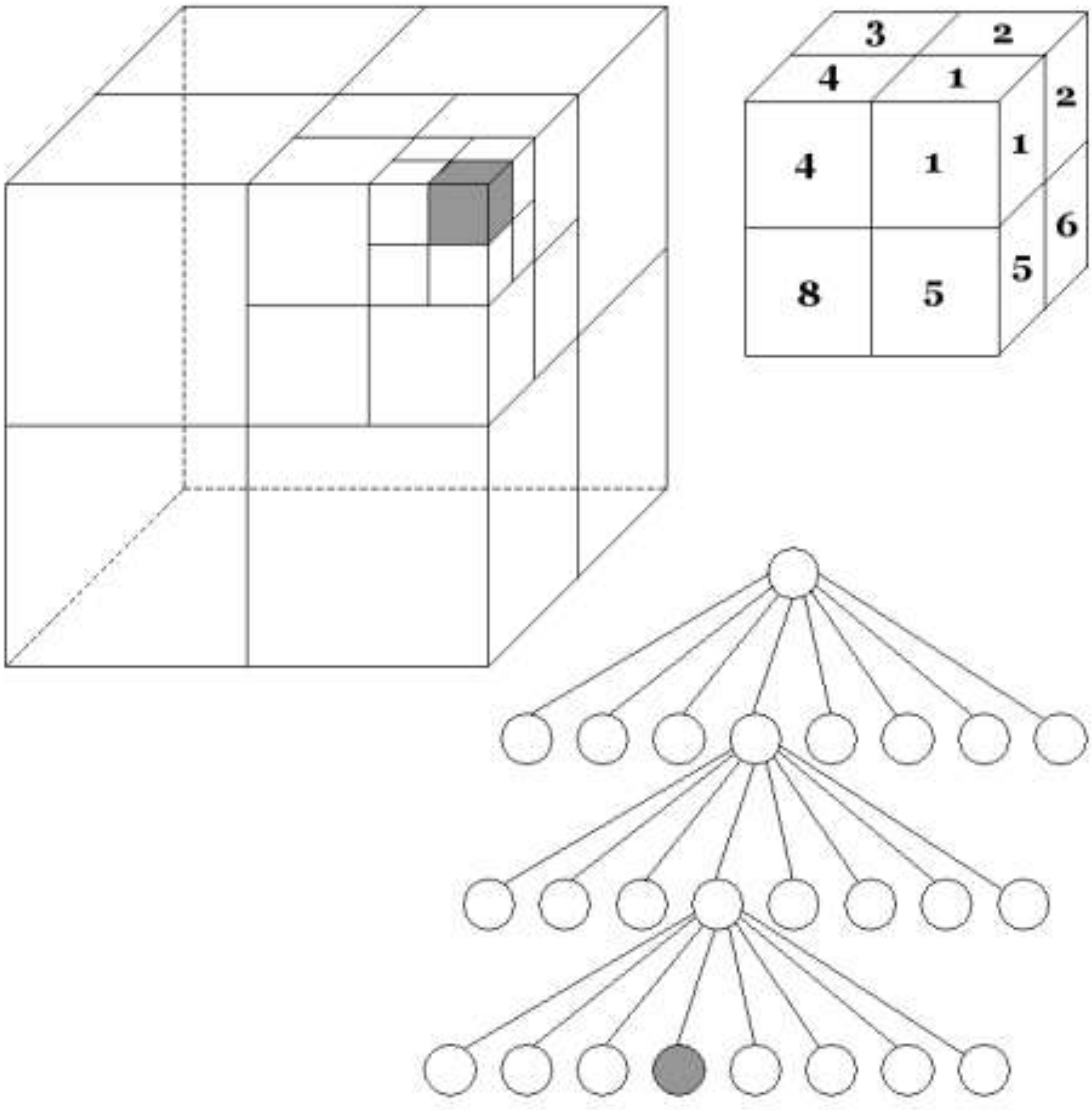


Figura 4.23: rappresentazione dello spazio mediante un octree.

Il metodo più efficiente per utilizzare l'enumerazione per occupazione spaziale viene suggerito dai "quadtree" e dai loro analoghi tridimensionali "octree".
La rappresentazione mediante quadtree di un oggetto bidimensionale si basa su una suddivisione ricorsiva di un quadrato in quadranti; ogni nodo rappresenta una regione quadrata sul piano. In un quadtree, ogni nodo possiede quattro discendenti.
La codifica octree costituisce un'estensione in tre dimensioni della codifica quadtree. Questo metodo utilizza una struttura gerarchica ad albero ottagonario, corrispondente agli octree, per rappresentare i solidi e si serve di algoritmi che crescono solo linearmente con la complessità dell'oggetto. La codifica octree di un modello è analoga a quella quadtree: una regione cubica viene suddivisa ricorsivamente in ottanti o in otto regioni cubiche. Ogni nodo di un octree, che non sia un nodo foglia, possiede otto discendenti.
Ipotizziamo di sovrapporre a un cubo un generico oggetto tridimensionale; se tale oggetto non copre uniformemente il cubo, quest'ultimo viene suddiviso in otto ottanti uguali e se tutti risultano completamente pieni o vuoti, non è necessaria una ulteriore suddivisione. Se un ottante è parzialmente pieno, lo si suddivide ancora in ottanti e si prosegue la suddivisione degli ottanti parzialmente pieni sino a quando le regioni risultanti non siano o completamente piene o completamente vuote, o fino a quando non sia stato raggiunto un predeterminato livello di risoluzione; in tal caso gli ottanti ancora parzialmente pieni possono essere arbitrariamente considerati pieni o vuoti, a seconda della convenzione assunta.
Il nodo radice dell'octree rappresenta l'intero vettore (oggetto) ed è spesso chiamato l'elemento universo. I nodi foglia costituiscono le regioni che non richiedono ulteriori suddivisioni e hanno dimensioni standard e posizioni esprimibili come potenze di 2. Il numero di suddivisioni in sequenza da un dato nodo all'elemento universo, determina il livello del nodo nell'octree e la dimensione della regione rappresentata dal nodo. Se l'altezza dell'albero è n, la massima dimensione potenziale del vettore corrisponde a 2^n x 2^n x 2^n.

4.6.4 Conclusioni

Considerando che la stella più lontana dista 29651 anni luce dal sole, lo spazio da rappresentare è rinchiuso in un cubo che abbiamo deciso avere dimensioni di 65536 anni luce (2^{16}) per lato. Questa grandezza, essendo una potenza di due, è comoda in quanto è possibile suddividere in ottanti, continuando ad avere le grandezze come valori interi. Il cubo è centrato sul Sole, che corrisponde anche all'origine degli assi. In realtà il nostro astro non dovrebbe essere situato esattamente nell'origine (sulla quale dovrebbe invece trovarsi la Terra), ma questa piccola differenza viene ignorata per avere una maggiore semplicità. Le distanze astronomiche sono tali da rendere nullo questo scostamento.

L'idea è quella di ordinare le stelle in modo decrescente in base alla magnitudine. Il cubo radice viene riempito con le stelle più luminose, quando viene raggiunta una certa dimensione di memoria occupata, viene suddiviso in ottanti e si procede al completamento di questi. Il processo viene ripetuto ricorsivamente fino ad aver inserito tutte le stelle.

Questa suddivisione spaziale delle stelle, in base anche alla loro magnitudine, è parsa in assoluto la scelta più efficiente. Infatti, anche a livello di renderizzazione, non verranno processati e rappresentati tutti i cubi con le stelle in essi presenti, ma solo i cubi effettivamente appartenenti al campo visivo e al punto di vista dell'osservatore, cioè solamente ciò che si trova nel frustum.

Ogni nodo contiene stelle più luminose di quelle dei nodi figli ma meno di quelle del padre. Se non si vuole superare un certo limite di magnitudine oltre il quale le stelle non sono più visibili all'osservatore, la ricerca degli astri da visualizzare può fermarsi a un certo livello senza dover scorrere l'intero albero.

A partire da queste conclusione sulla suddivisione spaziale dell'universo e sull'organizzazione delle Tabulae, si è passati alla modifica dell'Invector per conformarlo ai requisiti che sono stati messi in luce dall'analisi svolta in precedenza.

4.7 Il secondo importatore

4.7.1 Lettura delle stelle

Nella trattazione che segue, saranno implementati i principi appena discussi nel paragrafo precedente sulla suddivisione spaziale dell'universo.
La totalità delle stelle è contenuta nel cubo di dimensione di 65536 anni luce di spigolo con centro sul Sole.
Come nell'Invector precedente i dati sono letti da file e su questi vengono applicate le stesse trasformazioni necessarie a renderli utili ai nostri fini. Questa procedura viene eseguita dalla funzione `UM_ComputeData()`.
I dati sono incapsulati in `struct` di tipo `UM_S_Star`, e i riferimenti a queste sono inseriti in un array di puntatori. Si sono utilizzati array di puntatori a strutture, piuttosto che direttamente array di strutture, per rendere più veloce e leggera la loro gestione.
Questo array viene ordinato in modo crescente per valore di magnitudine assoluta, e quindi in modo decrescente di luminosità: ricordiamo che un minore valore di magnitudine corrisponde a una maggiore luminosità. Per l'ordinamento si è utilizzato l'algoritmo del quick sort [12]; a causa dell'elevato numero di elementi (più di due milioni), algoritmi come il selection sort o l'insertion sort, di complessità O(n^2), sarebbero risultati inefficienti. Il quick sort, che invece ha complessità O(n log n), completa l'ordinamento in tempi brevi.
C'è da premettere che le stelle, per come si trovano nel catalogo, sono ordinate in modo crescente secondo l'ascensione retta; quindi senza ordinamenti particolari (anche parziali) secondo la luminosità. Con questa constatazione abbiamo la sicurezza che il quick sort non degeneri mai nel caso peggiore, con complessità quadratica.
A questo punto bisogna fare una considerazione: tre stelle sono memorizzate con un valore di parallasse uguale a zero e questo causa una distanza dalla Terra di più di tre milioni di anni luce.
Ciò è stato valutato da noi come una imprecisione presente nel catalogo, in quanto queste tre stelle sarebbero collocate fortemente al di fuori del confine della Via Lattea, addirittura più lontane delle galassie prossime alla nostra. Le

tre stelle vengono quindi tolte dall'array.
Come già detto nel precedente paragrafo verrà costruito un octree, in cui ogni nodo, non solo le foglie, contiene un certo numero di stelle situate nella zona di spazio che rappresenta. In ognuno di questi vengono memorizzati i dati che verranno poi scritti in Tabulae. A ogni nodo corrisponderà perciò una Tabula.
I nodi (e le foglie) dell'albero sono implementati tramite la `struct UM_S_TabulaNode` (il nume della struttura ricorda il fatto che rappresenta una futura Tabula). Questa struttura contiene l'array di stelle e il numero di queste. In più si mantiene traccia degli eventuali nodi figli, mediante otto puntatori; quando verranno scritti in Tabulae, al posto del puntatore ci sarà il numero di Tabula figlia. Per ogni figlio è indicata anche la luminosità massima tra le stelle in esso presenti.
La dimensione massima stabilita per un nodo è di 20 kilobyte, ma come si vedrà in seguito questo limite potrà essere superato in casi eccezionali.
La costruzione della struttura ad octree è imputata alla funzione ricorsiva `UM_CreateTabulaeTree()`. Il metodo restituisce il puntatore al nodo radice dell'albero che viene mantenuto in una variabile d'istanza come proprietà del modulo.

4.7.2 Creazione dell'octree

L'algoritmo si fonda sull'idea di inserire stelle nel nodo fino a saturarlo; se ne rimangono ancora si suddivide il cubo associato al nodo in otto ottanti e su ognuno di questi è chiamato ricorsivamente il procedimento.
La funzione riceve in input i seguenti parametri: le coordinate dei piani che delimitano il cubo di spazio in cui l'euristica deve compiere la ricerca (`minX, maxX, minY, maxY, minZ, maxZ`); un array di puntatori alle stelle presenti in questa zona di universo; il numero di astri contenuti nell'array; il livello di profondità delle chiamate ricorsive della funzione.
Per prima cosa si controlla la condizione di terminazione. Se il vettore ricevuto da parametro è nullo (quindi non contiene stelle) viene restituito `NULL`; si chiude così un ramo dell'albero.

In caso contrario si alloca lo spazio necessario per il nodo:

```
UM_S_TabulaNode* pSTabulaNode =
            (UM_S_TabulaNode*)UM_Sys_Malloc(sizeof
(UM_S_TabulaNode));
```

La funzione di libreria `malloc` è stata ridefinita nell'ambiente Ultraport come `UM_Sys_Malloc()`.

Si procede con la copia dei riferimenti alle stelle nell'array presente nel nodo fino a che la somma delle dimensioni occupate da queste non supera i venti kilobyte.

La posizione delle stelle, da assoluta rispetto all'intero universo, viene resa relativa rispetto al centro del cubo nel quale esse si trovano. A ogni coordinata viene sottratto il valore del centro dell'ottante . Per gli ottanti con coordinate negative la sottrazione di un numero negativo diventa la somma di un numero positivo. `ppSidera` è l'array di puntatori alle stelle.

```
ppSidera[k]->x = ppSidera[k]->x - (minX + ((maxX - minX)/2));
ppSidera[k]->y = ppSidera[k]->y - (minY + ((maxY - minY)/2));
ppSidera[k]->z = ppSidera[k]->z - (minZ + ((maxZ - minZ)/2));
```

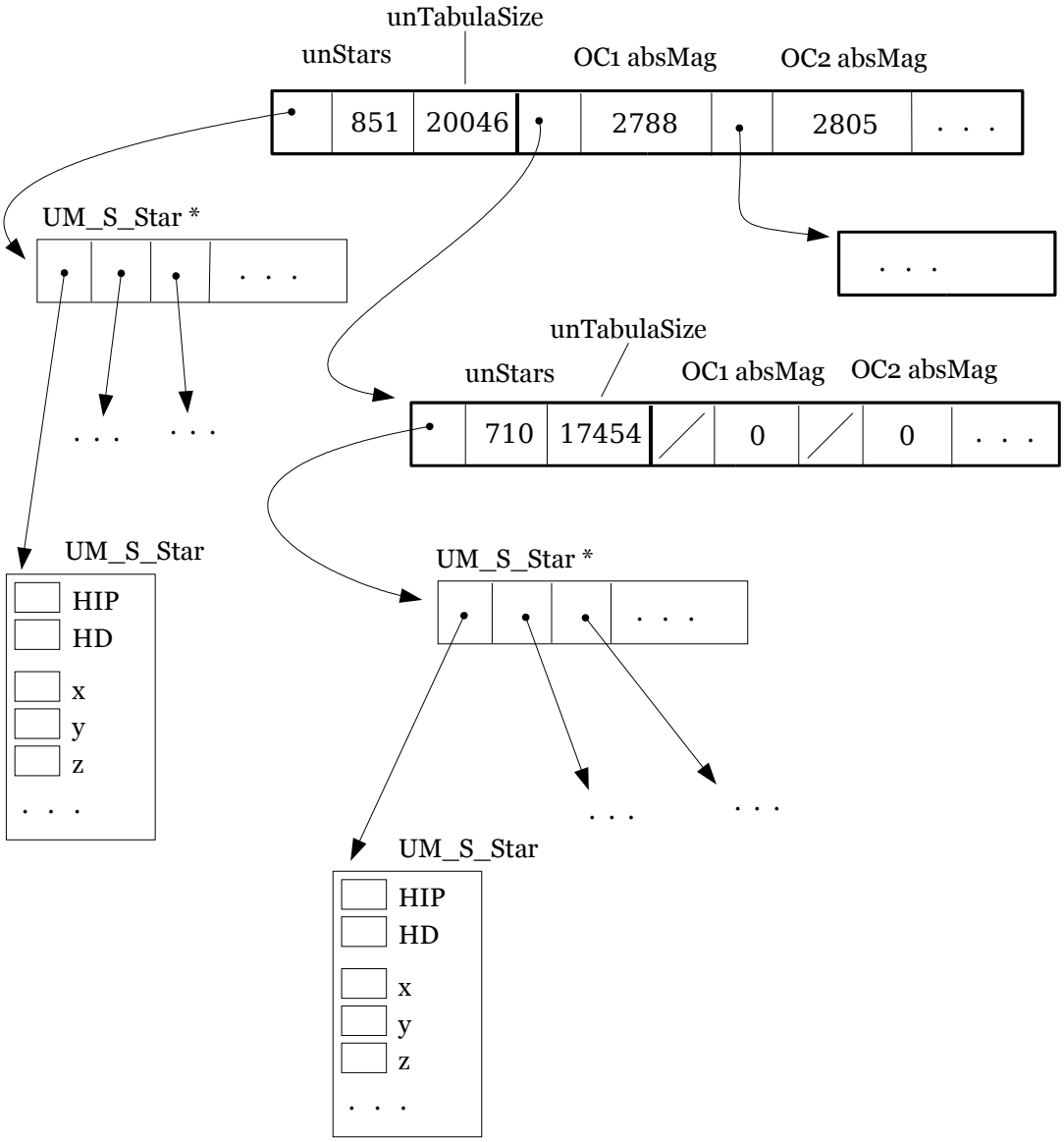
Nel caso si esauriscano gli astri prima di riempire completamente il nodo viene ritornato il riferimento allo stesso; ciò vuol dire che questo nodo è una foglia.

Se al contrario lo spazio di memoria a disposizione del nodo non è sufficiente, prima di effettuare il passo ricorsivo si controlla se è possibile esaurire tutte le stelle rimanenti con un sovraccarico del nodo, in modo da troncare qui il ramo dell'octree. Il sovraccarico che si è disposti a sopportare è del 100% della dimensione di partenza. La grandezza totale potrebbe giungere nel caso peggiore fino a quaranta kilobyte. Se risulta possibile troncare il ramo con il sovraccarico, si aggiungono nel nodo le stelle rimanenti e si ritorna al chiamante il puntatore al nodo stesso.

Nel caso medio, in cui le stelle rimanenti risultino comunque troppe, si rende necessaria la suddivisione del cubo associato al nodo in otto ottanti sui quali verrà ripetuto ricorsivamente il procedimento.

Per migliorare l'efficienza dell'algoritmo, l'array con le stelle rimanenti viene suddiviso in otto sotto-array, uno per ogni nodo figlio. Ognuno di questi nuovi array contiene le stelle dell'ottante al quale è associato.

Dal momento che la distribuzione delle stelle nello spazio non è sempre uniforme, dopo la suddivisione si può verificare che ci siano ottanti con molte stelle e altri con un esiguo numero. In tal caso, per evitare la creazione di Tabulae molto piccole, se la dimensione totale degli astri contenuti in ognuno degli otto sottoarray è inferiore a un certo limite (fissato al 25% rispetto alla dimensione massima di un nodo), le stelle in questione vengono inglobate nel nodo padre. In questo modo, non vengono create Tabulae con una dimensione inferiore ai 4 o 5 kilobyte, mentre la dimensione massima nel caso peggiore (cioè il caso in cui tutti gli ottanti sfruttano la percentuale di sovraccarico) diventa di 60 kilobyte.



Nel codice sottostante è illustrata la chiamata ricorsiva sul primo ottante.

```
pSTabulaNode->pSOC1 = UM_CreateTabulaeTree(
                        vppSOC[0],vuNStars[0], unLevel,
                        minX + (CUBE_SIZE / pow(2, unLevel)), maxX,
                        minY + (CUBE_SIZE / pow(2, unLevel)), maxY,
                        minZ + (CUBE_SIZE / pow(2, unLevel)), maxZ);
```

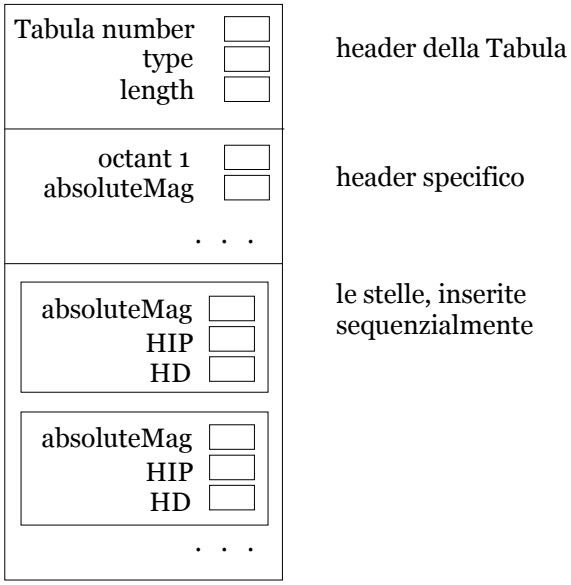
`pSTabulaNode->pSOC1` è il puntatore alla struttura del nodo figlio che rappresenta il primo ottante, `vppSOC[0]` è uno degli otto sotto-array; `vuNStars[0]` è il numero di stelle presenti nel precedente array e `unLevel` è il numero di livelli di profondità.
Le dimensioni di ogni ottante sono determinate a partire dalle dimensioni del cubo padre. Consideriamo come primo ottante quello in cui le coordinate x, y e z sono tutte positive. Quindi i limiti massimi (`maxX, maxY, maxZ`) rimangono tali, mentre vengono aumentati i limiti inferiori. Il valore dell'aumento è calcolato rispetto al livello di profondità.
Se nell'ottante sono presenti stelle e quindi la funzione non restituisce un valore nullo, si memorizza il valore massimo della magnitudine assoluta tra le stelle di quell'ottante. Corrisponde alla magnitudine della prima stella presente nel nodo, in quanto ricordiamo che l'array che le contiene è ordinato.

```
if (pSTabulaNode->pSOC1 != NULL)
  pSTabulaNode->magOC1 = pSTabulaNode->pSOC1->ppSData[0]->absMag;
```

4.7.3 Struttura delle Tabulae

L'octree rispecchia la gerarchia con cui verranno create le Tabulae. La struttura di una Tabula è così formata:

- l'header, standard per ogni Tabula, contenente metainformazioni (il proprio numero identificativo, il tipo, la lunghezza);
- un header specifico che memorizza:
  - gli otto riferimenti alle Tabulae figlie, tenuti tramite il loro numero identificativo;
  - accoppiato con ogni riferimento, è presente il valore di magnitudine assoluta della stella più luminosa fra quelle presenti nella Tabula figlia riferita
- l'insieme di stelle, ognuna delle quali ha la struttura già specificata.



Lo schema rende meglio l'idea di come sia strutturata ogni Tabula.
La struttura gerarchica che nell'albero è trattata con puntatori a nodi, nelle Tabulae viene gestita con i numeri univoci delle stesse.
Per quanto riguarda le magnitudini assolute, esse vengono memorizzate per migliorare l'efficienza nella ricerca delle stelle da visualizzare secondo il criterio di visibilità all'osservatore. Tale proprietà è determinata dal rapporto tra la luminosità della stella con la sua distanza dal punto di vista.
Conoscendo la magnitudine massima di ogni Tabula figlia e il limite di magnitudine sotto il quale non devo più visualizzare le stelle (perché troppo poco luminose), posso sapere in anticipo se devo proseguire nella lettura delle Tabulae figlie o interrompere la ricerca, risparmiando la lettura del sottoalbero.
I vantaggi conseguiti da questa scelta sono due: un risparmio di risorse computazionali, in quanto non vengono processate Tabulae contenenti stelle non visibili all'osservatore; ricordando che le Tabulae devono essere trasferite su Internet, si evita che vengano scaricate dalla rete Tabulae inutili.

4.7.4 Scrittura delle Tabulae

L'`UM_TabAppInvector` si occupa di creare nuove Tabulae; ogni volta che ne deve creare una, prima di procedere, deve conoscere la dimensione che occuperà. Quest'ultima viene richiesta al modulo corrente tramite il metodo `UM_Com-`

puteDataLength(). Tale funzione restituisce la grandezza del nodo associato alla Tabula.
La Tabula madre deve conoscere i numeri delle Tabulae figlie; siccome il numero di una nuova Tabula viene assegnato dal sistema al momento della creazione, è necessario partire nella scrittura delle Tabulae dai nodi foglia. Al momento della creazione e della scrittura del nodo figlio sulla Tabula che rappresenta, il numero di quest'ultima viene memorizzato nel nodo padre.
Un algoritmo ricorsivo, implementato nella funzione UM_ComputeDataLengthTree() e invocato dalla funzione UM_ComputeDataLength(), visita l'albero in profondità scegliendo la foglia da scrivere. Dopo averla trovata, rimuove il nodo dall'albero e ne restituisce la dimensione.

```
UM_int UM_Invector_astris_sidera::UM_ComputeDataLengthTree(
                                        UM_S_TabulaNode* pSTabulaNode)
{
      if ((*pSTabulaNode).pSQ1 != NULL)
      {
            m_pnTabulaNumber = &(*pSTabulaNode).nTabulaQ1;
            return UM_ComputeDataLengthTree((*pSTabulaNode).pSQ1);
      }
      if ((*pSTabulaNode).pSQ2 != NULL)
      {
            m_pnTabulaNumber = &(*pSTabulaNode).nTabulaQ2;
            return UM_ComputeDataLengthTree((*pSTabulaNode).pSQ2);
      }

     ...

      if ((*pSTabulaNode).pSQ8 != NULL)
      {
            m_pnTabulaNumber = &(*pSTabulaNode).nTabulaQ8;
            return UM_ComputeDataLengthTree((*pSTabulaNode).pSQ8);
      }

            // Save the node to be written in the Tabula
      m_pSTNode = pSTabulaNode;

            // Return the size of the tabula
      return (sizeof(UM_S_StarTabHeader) +
                        (*pSTabulaNode).unStars * sizeof(UM_S_Star));
}
```

La dimensione è calcolata come l'header specifico più la somma della grandezza di ogni stella.
Dopo aver allocato le risorse necessarie, UM_TabAppInvector demanda il riempimento della Tabula al modulo corrente attraverso il metodo UM_ProcessFile(). Questa funzione riceve come parametro il puntatore all'area di me-

moria in cui andrà a scrivere i dati e la grandezza dello spazio riservato.
Le operazioni compiute sono fondamentalmente due: la scrittura dell'header specifico e un ciclo (mostrato di seguito) nel quale vengono copiate sequenzialmente le stelle presenti nel nodo. Per ogni stella copiata, viene aggiornato lo spazio libero rimanente e viene fatto avanzare il puntatore all'area di memoria in cui scrivere il prossimo astro.

```
      // write the data of stars
for (i=0; i < (*m_pSTNode).unStars; ++i)
{
      UM_MemCpy(*m_ppDataPos, *punSpace, m_pSTNode->pSData[i],
                                          sizeof(struct UM_S_Star));
      *punSpace -= sizeof(struct UM_S_Star);
      *m_ppDataPos += sizeof(struct UM_S_Star);
      m_unCurrFileRecord++;
}
```

La funzione restituisce la percentuale di avanzamento, calcolata come rapporto fra le stelle processate fino a quel momento e quelle totali.
Come ultima operazione, `UM_TabAppInvector` scrive la Tabula su periferica.
Questo insieme di passi (creazione, riempimento e scrittura) viene iterato fino a quando non si sono esauriti i nodi dell'albero.

4.7.5 Analisi statistiche sulla creazione delle Tabulae

Prima di terminare la sua esecuzione, `UM_TabAppInvector` riporta a video un insieme di informazioni sul processo appena terminato. Nell'ultima versione dell'Invector, i dati che abbiamo ottenuto nell'importare il catalogo Tycho sono stati i seguenti:

- numero totale di stelle processate: 2.072.780;
- numero di Tabulae create: 4115;
- numero identificativo della Tabula radice: 56112;
- dimensione totale delle Tabulae scritte: 56,64 MegaByte.

Con questo nuovo importatore, le Tabulae non hanno una grandezza fissa come in precedenza, ma la loro dimensione può variare da un minimo di 4-5 ad un massimo di 60 kilobyte. Per monitorare come le loro dimensioni si distribuiscono su questo intervallo, si è aggiunta una nuova funzionalità: un grafico a istogrammi. L'intero intervallo è stato suddiviso in 50 fasce; per ognuna di queste,

una barra indica il numero di Tabulae create, la cui dimensione cade in quel range.
L'analisi sui risultati ottenuti ha messo in luce qualche limite nella scrittura delle Tabulae. Infatti la dimensione totale occupata dalle Tabulae create, che era pari a 56 Megabyte, è parsa eccessiva. Inoltre il maggior numero di Tabulae erano di piccole dimensioni. Si è pensato pertanto di ottimizzare i dati relativi alle stelle per renderle meno spaziose e ottenere pertanto meno Tabulae, ma più dense di astri.

4.7.6 Compressione dei dati stellari

Analizzando tutti i campi presenti nelle strutture dati che definiscono le stelle, si è valutato quali potessero essere ottimizzati o tolti per risparmiare lo spazio da loro occupato e ottenere così che lo stesso numero di stelle possa essere contenuto in meno Tabulae.
(Ricordiamo che le Tabulae vengono riempite fino a una certa dimensione di memoria, non in base a un numero precisato di stelle).
Con la compressione dei dati, ogni struttura stella avrà una grandezza variabile, determinata dai valori eventualmente assenti o da quelli indicati con una precisione minore.

Compressione del campo HD

Il primo campo sul quale è stata posta l'attenzione è quello che definisce il numero del catalogo HD delle stelle. Nel database da noi utilizzato, le stelle sono identificate attraverso due numeri di catalogo: quello HIP, generalmente il più riconosciuto all'interno della comunità scientifica, e quello HD, meno standard del precedente. Inoltre bisogna premettere che, mentre il valore del catalogo HIP è presente in tutte le stelle del database (sia del catalogo Hipparcos sia di quello Tycho), il numero di catalogo HD non compare in tutti gli astri, ma solo in alcuni (che per il catalogo Tycho rappresentano una minoranza).
Pertanto, in un primo momento si era ventilata l'ipotesi di escludere il valore HD per risparmiare 4 byte (il numero HD è contenuto in un `int`). Questa scelta non pareva penalizzante per il progetto, considerando il fatto che il numero HD

era un dato ridondante e non indispensabile. In seguito, tuttavia, si è deciso di conservare questo numero, ma ottimizzando la sua memorizzazione. Infatti, il valore del catalogo HD è stato conservato solo nelle stelle che lo possedevano, mentre per le altre è stato risparmiato. Se questa ottimizzazione non ha portato grandi vantaggi utilizzando il catalogo Hipparcos (poiché quasi tutti gli astri in esso presenti conservano il numero HD), i benefici si sono apprezzati soprattutto nel processamento del catalogo Tycho, nel quale molte stelle sono state compresse.

Compressione delle coordinate spaziali

Una ulteriore modifica effettuata per ridurre la dimensione in memoria occupata dalle stelle è stata quella di ottimizzare la memorizzazione delle sue coordinate spaziali. Prima di questa analisi, le coordinate x, y e z delle stelle erano tenute in variabili di tipo `float` (32 bit per ogni coordinata).

Si è valutato il fatto di poter "forzare" tali valori in variabili `short` (16 bit) o addirittura `byte` (8 bit), se questo avesse comportato solo una piccola perdita di precisione.

L'idea di fondo è quella di suddividere lo spazio all'interno del cubo, nel quale la stella è situata, in un certo numero di strati (coincidenti con i possibili valori rappresentabili, cioè 65536 per gli `short` e 256 per i `byte`), ottenere la posizione della stella in relazione a uno degli strati, rapportare lo strato alla dimensione totale del cubo e, infine, valutare se l'errore ottenuto a causa della trasformazione fosse accettabile o meno.

Nel nostro caso, l'errore di misurazione tollerato è stato fissato in 0,01 anni luce.

In seguito, è presente l'estratto di codice che prova a memorizzare le coordinate all'interno di variabili di tipo `byte`.

Le variabili x, y e z sono le coordinate in `float` della stella; `maxX`, `minX`, `maxY`, `minY`, `maxZ`, `minZ` sono le coordinate che definiscono lo spazio rappresentato dal cubo in cui la stella è posta; MAX_ACCEPTED_ERROR è la costante che definisce l'errore di misurazione tollerato: in questo caso, vale 0.01; `pSidera` rappresenta il puntatore alla stella che stiamo considerando.

```
f32CubeSize = fabs(maxX - minX);

byteX = (byte)((x * 127) / (f32CubeSize/2));
byteY = (byte)((y * 127) / (f32CubeSize/2));
byteZ = (byte)((z * 127) / (f32CubeSize/2));

if ((x - ((byteX * (f32CubeSize/2))/127) < MAX_ACCEPTED_ERROR)
    && (y - ((byteY * (f32CubeSize/2))/127) < MAX_ACCEPTED_ERROR)
    && (z - ((byteZ * (f32CubeSize/2))/127) < MAX_ACCEPTED_ERROR))
{
    ...

    pSidera->x = byteX;
    pSidera->y = byteY;
    pSidera->z = byteZ;

    ...
}
```

Se l'errore per una qualsiasi coordinata é maggiore della soglia prevista si prova la memorizzazione in variabili di tipo short. Il codice per questo secondo tentativo coincide con quello del primo, a differenza del tipo delle variabili e del numero di strati rappresentabili.
Se anche questa prova fallisce, le coordinate rimangono sotto forma di variabili floating point.

Compressione della magnitudine assoluta

La variabile `unAbsMag` è di tipo `float` (32 bit). Siccome per la magnitudine assoluta non servono più di due decimali e l'intervallo di valori nel quale è definita è sicuramente compreso tra -30,00 e +30,00, essa può essere contenuta in una variabile di tipo `short`. Infatti tra i due limiti dell'intervallo (-30,00 e +30,00), sono rappresentabili 6000 numeri con due decimali.
Il valore letto da file, trasformato in magnitudine assoluta, viene reso intero con una moltiplicazione per 100. A questo risultato viene sommato 3000 per renderlo sicuramente positivo (o al massimo pari a 0 se la magnitudine della stella è -30,00). A questo punto ci si rende conto che il valore ottenuto può essere rappresentabile addirittura utilizzando solo 13 bit invece dei 16 a disposizione.

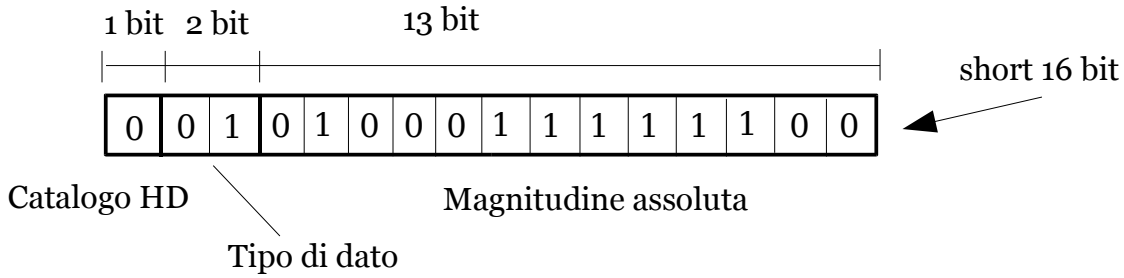
```
m_CFile.UM_Read(&appMag, sizeof(appMag));
f32AbsMag = (appMag / 256.0 + 5 - 5 * log10(f64Distance / 3.26));
unAbsMag = UM_FloatToInt(f32AbsMag * 100);
unAbsMag += 3000;
```

Come rendere operative le compressioni

Le modifiche apportate ai dati alleggerivano notevolmente lo spazio occupato dalle stelle, ma c'era da considerare un aspetto non trascurabile. All'interno delle strutture dati che definiscono le stelle, bisognava inserire in qualche modo l'informazione legata alla presenza o meno del catalogo HD per quella stella e al formato nel quale le coordinate erano espresse. Senza queste informazioni aggiuntive, sarebbe risultato impossibile risalire al tipo di compressione adottato per ciascuna stella.

Si è esclusa immediatamente l'idea di inserire nuove variabili che rappresentassero queste informazioni, dal momento che lo spazio occupato da questi campi aggiuntivi avrebbe vanificato e annullato i guadagni ottenuti precedentemente. Pertanto si è deciso di sfruttare la variabile `unAbsMag`, perché come detto precedentemente, per questo valore si utilizzano solo 13 bit dei 16 disponibili. In questi 3 bit, i 3 più significativi, si incapsulano le informazioni sul catalogo HD e sul tipo di codifica delle coordinate.

In particolare, il bit più significativo è stato utilizzato per indicare la presenza o meno del numero di catalogo HD (vale 1 in presenza del valore, 0 altrimenti); i due bit successivi, che possono rappresentare in tutto 4 valori, sono stati invece sfruttati per segnalare il tipo delle coordinate spaziali: 00 indica che le variabili x, y e z sono memorizzate in float; 01 corrisponde a variabili di tipo short; 10 a variabili byte.



Questa immagine è un esempio di come vengono memorizzate la magnitudine assoluta con le informazioni aggiuntive sulla compressione dei campi. La stella in questione non ha il campo HD (il primo bit è a zero) e le sue coordinate sono espresse in variabili di tipo short (il secondo e terzo bit valgono 01). La magnitu-

dine assoluta di questo astro vale -7.00, che in seguito alle trasformazioni viene memorizzata come 2300, cioè 100011111100 in binario.
Lo spazio occupato dalla suddetta stella, che prima della compressione ammontava a 26 byte, ora è pari a solamente 14 byte; il tutto senza perdita di informazioni.

Cosa comporta la compressione

La modifica della composizione dei dati ha portato a un parziale cambiamento del codice, sia nella costruzione dell'octree, sia nel calcolo della grandezza del nodo (e quindi della Tabula), sia nella scrittura della Tabula.
Nella costruzione dell'albero, sia per l'overload dell'intero nodo sia per quello degli ottanti, è necessario calcolare la grandezza di ogni stella per capire quante ce ne stanno nella percentuale di overload, dal momento che ora la dimensione di ogni astro è variabile.
Inoltre è stato aggiunto del codice per impostare la variabile `unAbsMag` secondo le specifiche e per ricavare da essa le informazioni sulla compressione delle stelle. Questo viene effettuato tramite operazioni logiche di 'and' e di 'or' bit a bit. Nel codice sottostante è verificata la presenza del numero del catalogo HD. Se non c'è, il primo bit più significativo del campo `unAbsMag` della struttura `pSidera` (la stella) viene settato a 0; altrimenti quest'ultimo viene settato a 1 e la dimensione del valore HD viene aggiunta alla dimensione totale della stella.

```
    // check if there is the HD catalog number
if (pSidera->unHDCatalog == HD_NOT_VALID)
{
        // set the first bit of ppSidera[k]->unAbsMag to 0
    pSidera->unAbsMag &= ~0x8000;    // ~0x8000 == 0 11 111111111111
}
else
{
        // add the size of HD catalog number
    pSidera->unStarSize += sizeof(UM_uint);
        // set the first bit of ppSidera[k]->unAbsMag to 1
    pSidera->unAbsMag |= 0x8000;     // 0x8000 == 1 00 000000000000
}
```

Ovviamente, nella fase di copiatura delle stelle (descritta sopra), quando si sceglie il tipo di variabile, bisogna anche tenere traccia della dimensione che occupa e delle modifiche da effettuare sulla variabile `unAbsMag`.

Per il tipo `byte` tramite un or logico, viene forzato il secondo bit più significativo al valore di 1.

```
    // add the size of x, y, z coordinates in byte
pSidera->unStarSize += (sizeof(UM_s8) * 3);
    // set the second bit of ppSidera[k]->unAbsMag to 1
pSidera->unAbsMag |= 0x4000;        //0 10 000000000000
```

Come si vede gli `short` occupano 6 byte (2 byte per 3 variabili) e in questo caso viene settato il terzo bit più significativo a 1:

```
    // add the size of x, y, z coordinates in short
pSidera->unStarSize += (sizeof(UM_s16) * 3);
    // set the third bit of ppSidera[k]->unAbsMag to 1
pSidera->unAbsMag |= 0x2000;        //0 01 000000000000
```

Nel caso in cui l'errore sia troppo grande si aumenta solamente la dimensione occupata di 12 byte, in quanto i bit interessati di `unAbsMag` valgono già 0.

```
    // let float
pSidera->unStarSize += (sizeof(UM_f32) * 3);
```

La funzione `UM_ComputeDataLength()` restituisce la somma della grandezza di ogni stella presente nel nodo.
A differenza di prima, la funzione `UM_ProcessFile()` non si limita a copiare strutture di grandezza fissa ma, per ogni stella presente nel nodo, copia campo per campo, valutando la grandezza di quelli variabili.

```
for (i=0; i < m_pSTNode->unStars; ++i)
{
    copy the stellar class

    copy HIP Catalog number

           // 1 xx 000000000000
    if (0x8000 == (m_pSTNode->ppSData[i]->unAbsMag & 0x8000))
    {
           copy HD Catalog number
    }

          // x 00 000000000000
    if (0x0 == (m_pSTNode->ppSData[i]->unAbsMag & 0x6000))
    {
            using float coordinates, copy x, y, z coordinates
    }
          // x 01 000000000000
    else if (0x2000 == (m_pSTNode->ppSData[i]->unAbsMag & 0x6000))
    {
        using short coordinates, copy x, y, z coordinates
    }
```

```
            // x 10 000000000000
        else
        {
            using byte coordinates, copy x, y, z coordinates
        }

        copy the absolute magnitude
    }
```

Miglioramenti ottenuti

In seguito a questa operazione di compressione, si è notata una notevole diminuzione del numero di Tabulae ottenute, che è sceso da 4115 a 2021.
Inoltre c'è stato un consistente ridimensionamento dello spazio occupato dalle Tabulae in memoria. Infatti, se in un primo momento la grandezza era di oltre 56,64 Megabyte, dopo la compressione essa si limitava a 31,92 Megabyte, con un risparmio superiore al 40%.
Supponendo di voler accettare un errore di un decimo di anno luce (0,1 ly) invece di un centesimo (0,01 ly), utilizzato in precedenza, il numero di Tabulae create è sceso a 1745 e la dimensione di queste a 28,9 Megabyte; tuttavia il risparmio ottenuto è stato considerato insufficiente per compensare l'aumento dell'errore di una unità di grandezza (anche se, considerando le distanze astrali, l'errore di 0,1 ly non ha conseguenze rilevanti nella visualizzazione sullo schermo di un computer).

4.8 La seconda visualizzazione

Prima di questa fase, il motore grafico è stato modificato, a seconda delle nostre esigenze e richieste, per adattarsi alla gestione del depth buffer. E' stata aggiunta la possibilità di disattivare il depth buffer, per fare in modo che tutte le stelle siano disegnate sullo sfondo senza controlli relativi alla profondità.

4.8.1 Lo scene graph

Lo Specus mantiene una struttura interna con tutti i nodi delle entità del mondo digitale, detta scene graph.
Lo scene graph è basato sul concetto di grafo della scena: un grafo diretto aciclico che organizza e memorizza tutti i dati necessari per visualizzare la scena 3D.
I modelli 3D sono le foglie del grafo, i nodi intermedi sono trasformazioni che controllano la posizione e l'orientazione e selettori di nodi. Il rendering viene effettuato semplicemente visitando il grafo. E' possibile implementare in maniera efficiente ed elegante numerose ottimizzazioni, come l'esclusione dei modelli 3D che non possono essere visti dall'utente.
In alcuni casi, il grafo della scena contiene anche il "comportamento" degli oggetti.
Le Tabulae creano oggetti nel mondo digitale tramite la costruzione di nodi dello scene graph.
Lo scene graph ha una struttura dinamica che varia nel tempo, ma mantiene una parte invariante costituente la struttura di base, costruita all'avvio dal Nucleus.
Su tale struttura di base viene costruito lo scene graph del singolo canale con tutte le entità che vi si trovano. Ogni entità del canale deve essere creata al di sotto di uno dei due nodi di raggruppamento:

- FAR ENTITIES: raccoglie tutte le entità poste a grande distanza dall'osservatore (stelle, montagne...).
- LOCAL ENTITIES: raccoglie tutte le entità comprese nella sfera locale dell'osservatore.

Ogni parte del canale ultravisivo può essere cancellata a partire dal nodo di raggruppamento, invariante, quando si deve attivare un nuovo canale.

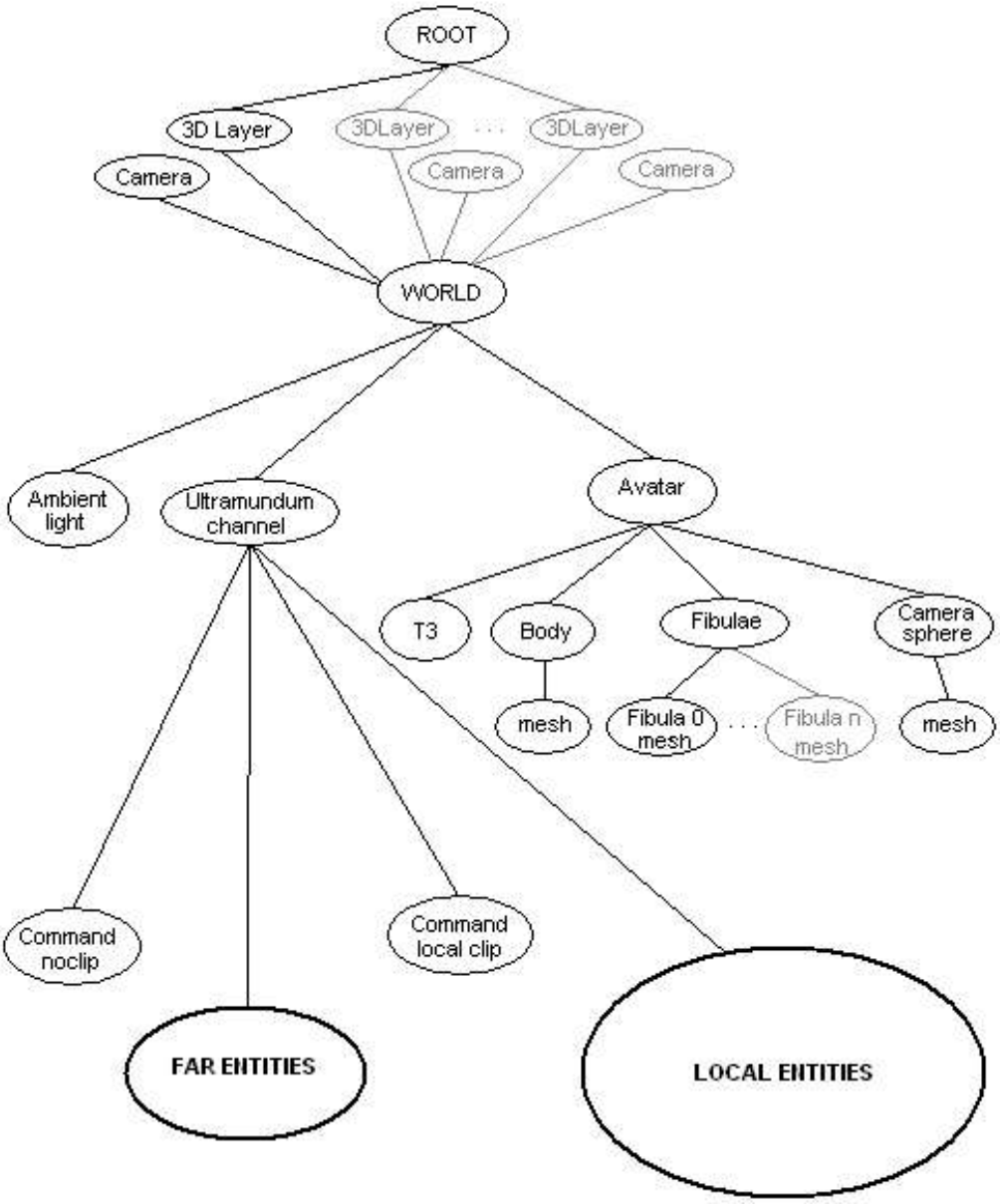


Figura 4.24: lo scene graph.

4.8.2 Le stelle come far entities

Il lavoro cospicuo compiuto in precedenza per la suddivisione spaziale dell'universo ha facilitato notevolmente le operazione da svolgere in questa fase.
Questa parte è stata inserita provvisoriamente nel Nucleus, in quanto non è ancora completamente pronta la struttura adatta. In esso è contenuto il codice per la lettura della Tabulae e la visualizzazione delle stelle nello spazio tridimensio-

nale. Rispetto alla prima versione del Nucleus, si è notato un notevole miglioramento delle prestazioni, determinato dalla modularizzazione dei dati, ma anche dal fatto che la primitiva utilizzata per la visualizzazione degli astri è stata quella dei puntini. Questi puntini sono inseriti in nodi dello scene graph che vengono attaccati alle FAR ENTITIES, disegnati quindi senza il controllo sul buffer di profondità. Gestire punti risulta più veloce che gestire cubi, anche la navigazione risulta quindi più fluida.

Il discorso della scomposizione in moduli ha avuto però come effetto quello di rendere necessarie delle modifiche da apportare al codice per la visualizzazione delle stelle. Le Tabulae non dovevano più essere lette in modo sequenziale, renderizzando le stelle al loro interno; in questo modo, infatti, ogni beneficio apportato dalla struttura ad albero delle Tabulae sarebbe stato reso vano, oltre al fatto che le coordinate con cui sono stati memorizzati gli astri sono relative al centro della Tabula a cui appartengono. I dati dovevano essere letti attraverso una visita in profondità dell'albero e, in seguito, renderizzati. Grazie a ciò, è anche possibile fissare un limite massimo di profondità o di luminosità oltre il quale gli astri non vengono visualizzati. La visita in profondità viene realizzata attraverso un algoritmo ricorsivo. La funzione che lo implementa e che ha reso possibile questo metodo di visualizzazione si chiama `UM_ReadStarsTabulaeTree()`. Essa prende in input il numero della Tabula radice dell'albero, le coordinate del cubo che delimitano l'area in cui si trovano le stelle appartenenti a quella Tabula e l'array di vertici che deve essere riempito dalla funzione stessa. Quest'ultima effettua tutte le operazione necessarie per decomprimere i dati, per analizzare ad una ad una le stelle della Tabula e per inserire i vertici corrispondenti a ogni stella nell'array. Infine richiama ricorsivamente il metodo su tutte le Tabulae figlie. In questo modo, l'algoritmo ricorsivo permette di visitare tutto l'albero delle Tabulae.

A questo punto, l'ultima operazione compiuta per visualizzare le stelle è stata quella di inserire nello scene graph i nodi che contengono le coordinate dei punti degli astri e il modo in cui devono essere disegnati. Tali nodi, a causa dei limiti imposti dall'hardware, non possono contenere più di 30 o 40 mila vertici, limite oltre il quale le prestazioni del sistema nella visualizzazione decadrebbero note-

volmente.
Nella magnitudine assoluta sono settate tutte le opzioni con cui sono memorizzate le stelle. Ad esempio prima di leggere le coordinate bisogna estrarre il secondo e il terzo bit più significativi della magnitudine assoluta. In questi due bit è settato il tipo (float, short o byte) con cui i valori di x, y, z erano stati memorizzati.
Nel momento in cui le coordinate di ogni vertice (e quindi di ogni stella), da relative alla Tabula vengono rapportate ad assolute rispetto al sistema, deve essere specificato anche il colore con cui verrà disegnato.
Per determinare il colore si estrae dalla classe stellare la classe spettrale, contenuta in quattro bit, dal quarto all'ottavo bit più significativi.

```
spectralClass = (unStellarClass >> 8 & 0xf)   // 0000 1111 0000 0000
```

A seconda della classe spettrale della stella corrisponde un colore, che normalmente varia tra il bianco e il giallo ma a volte tendente verso l'arancione.

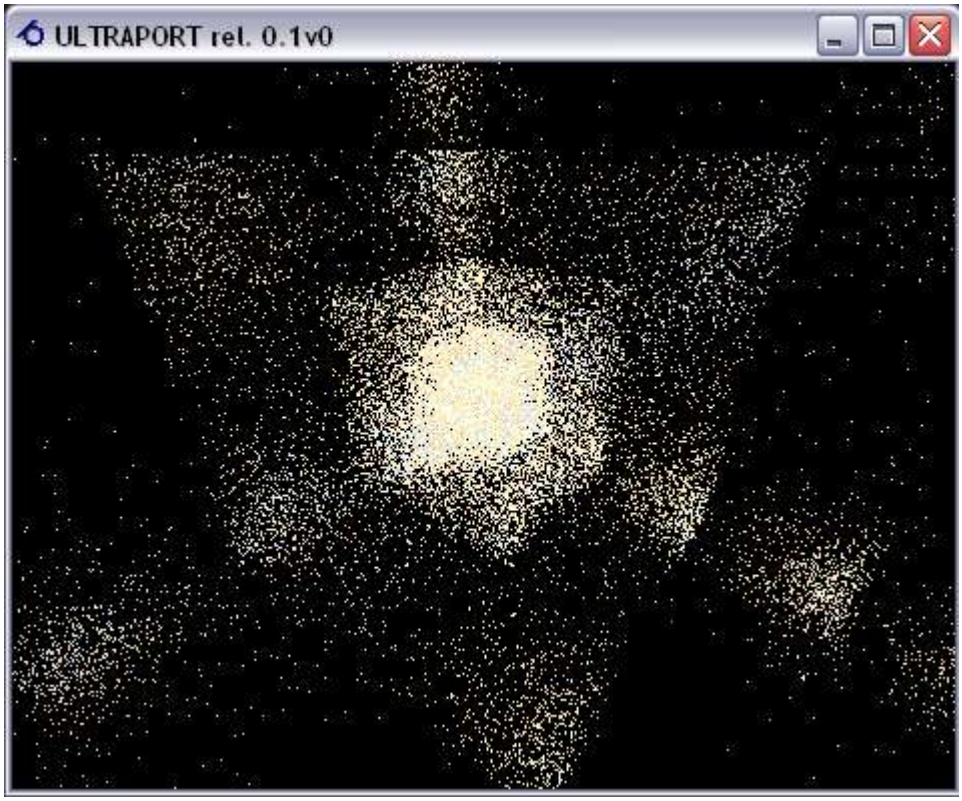


Figura 4.25: si note che le Tabulae rappresentano dei cubi di spazio.

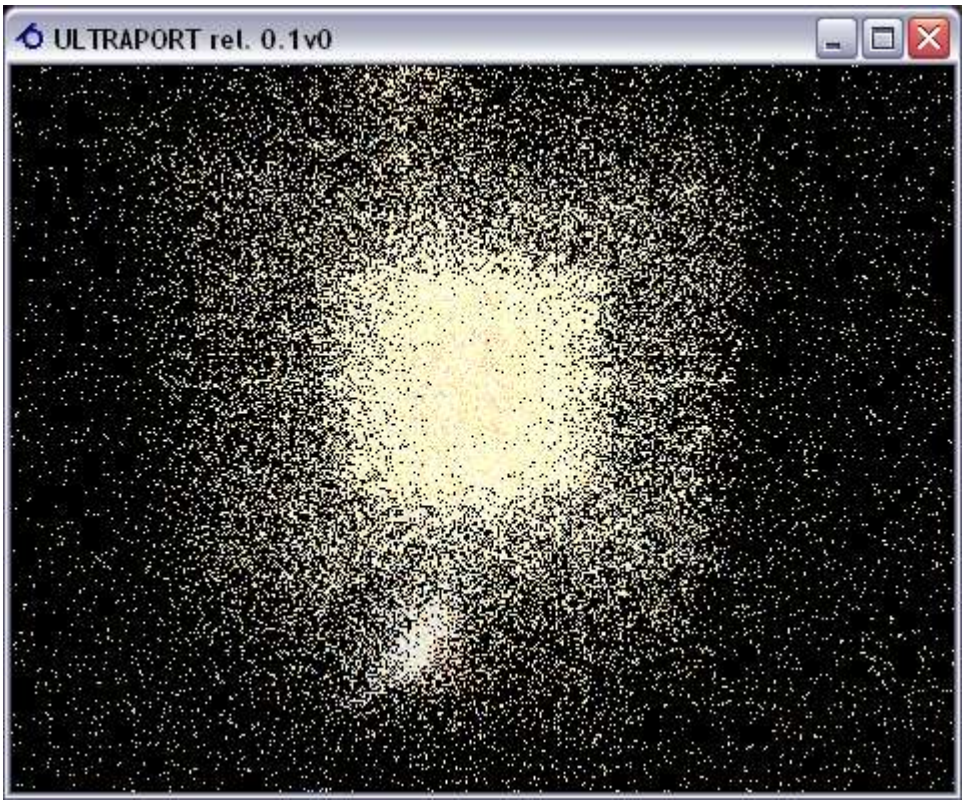


Figura 4.26: le Tabulae che rappresentano il centro dello spazio conosciuto.

Anche questa volta il risultato corretto non è stato raggiunto al primo tentativo: in Figura 4.25 e 4.26 la rappresentazione risente di errori nella conversione delle coordinate da relative ad assolute. Si nota in modo evidente che ogni Tabula rappresenta un cubo di spazio.

La seconda di queste figure mostra il centro dello spazio conosciuto; al centro di quel cubo molto denso c'è il nostro Sole.

E' possibile apprezzare i diversi colori degli astri: la maggior parte è gialla o bianca, ma ne sono visibili anche un discreto numero di tendenti all'arancione e qualche sporadica tendente all'azzurro.

Corretto l'errore nella conversione delle coordinate l'universo ha preso una forma più reale (vedere Figure 4.27, 4.28 e 4.29). Risulta possibile quindi esplorare le zone conosciute muovendosi a velocità superiori a quelle della luce.

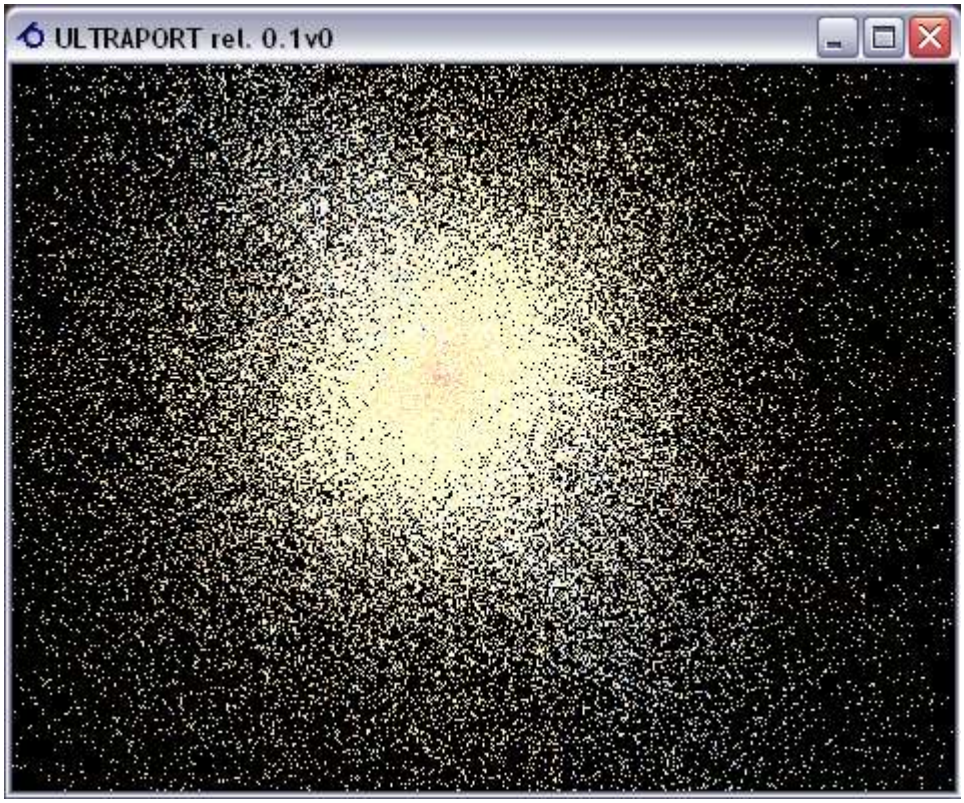


Figura 4.27: le 112522 stelle appartenenti al catalogo di Hipparcos.

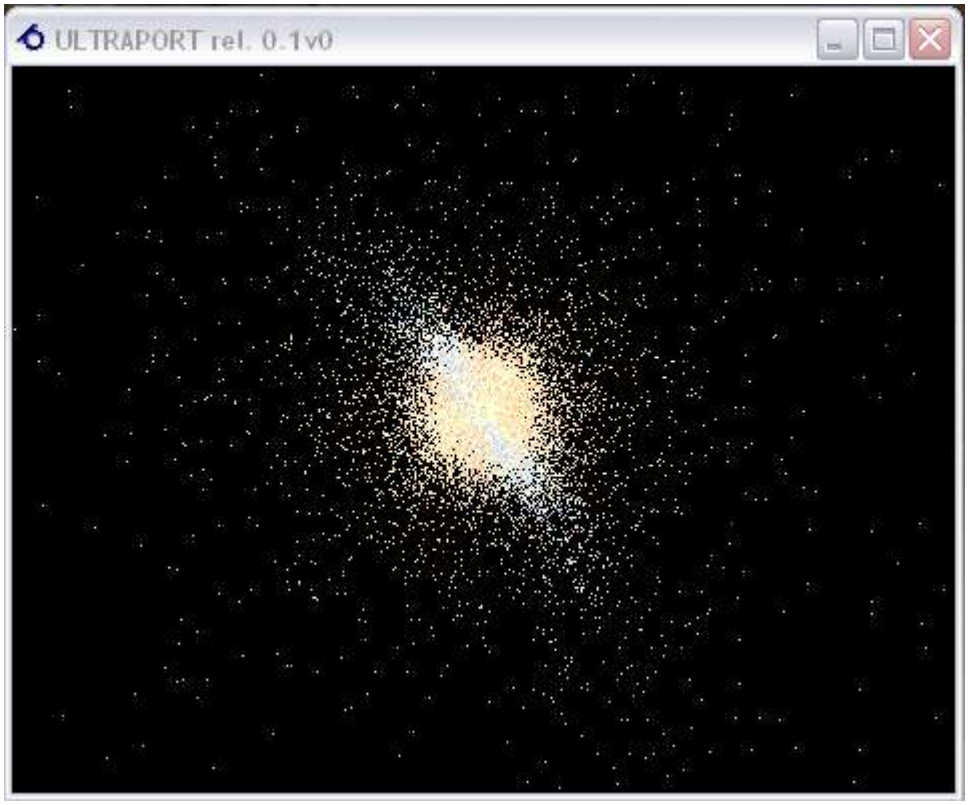


Figura 4.28: le stelle più luminose, di colore bianco-azzurrino, sono situate sul piano galattico.

In Figura 4.28 si nota in maniera evidente il piano galattico. In esso si trovano le stelle più luminose, con una temperatura più elevata. Il colore con cui vengono disegnate tende quindi verso l'azzurro. Gli astri situati nell'alone galattico tendono invece verso il giallo-arancio, causato dal fatto che hanno temperature meno elevate.

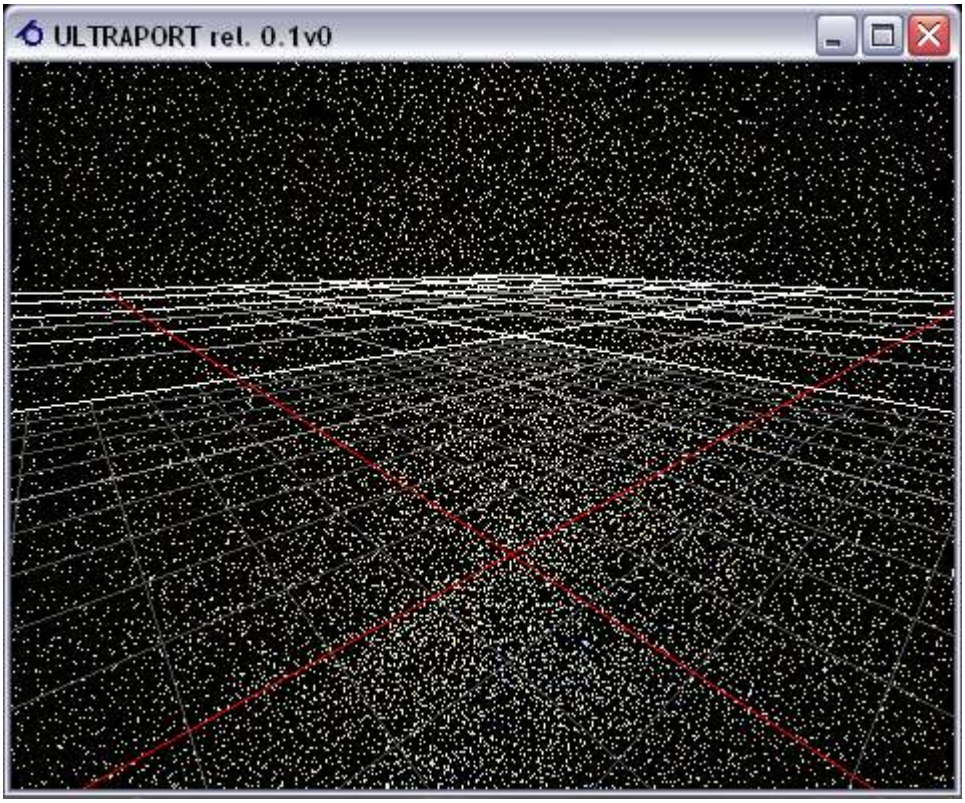


Figura 4.29: lo spazio visto dall'origine degli assi.

Il risultato finale di questo processo è stata la rappresentazione definitiva del nostro planetario tridimensionale navigabile.

5. CONCLUSIONI

Lo svolgimento dello stage ha richiesto lo studio preliminare di argomenti a noi non noti, nonché l'utilizzo di linguaggi e tecniche informatiche non ancora affrontati nel piano di studi della laurea triennale, in quanto previsti per la laurea specialistica.
Il progetto ha avuto carattere interdisciplinare, dal momento che ha visto fortemente coinvolti argomenti non connessi all'informatica, come l'astronomia. Questi concetti sono stati integrati tramite uno studio approfondito che ha occupato una parte rilevante dell'intero periodo di tirocinio.
Anche nell'ambito informatico, abbiamo dovuto lavorare per raggiungere una certa padronanza su argomenti, in particolare relativi alla grafica tridimensionale, incontrati per la prima volta nella nostra carriera universitaria.
Il tempo impiegato per lo svolgimento dello stage è stato superiore a quello previsto, anche per i motivi sopra citati; tuttavia, questo è andato a nostro vantaggio in quanto il maggiore tempo dedicato ci ha permesso di fare più esperienza, soprattutto in un ambito informatico in forte sviluppo come quello della grafica tridimensionale.
Abbiamo speso queste energie pensando di poter sfruttare l'accrescimento culturale nell'ambito di un proseguimento degli studi e comunque come cultura personale.
L'esperienza dello stage è stata interessante perché ci ha fatto maturare dal punto di vista lavorativo; il fatto di affrontare un progetto così complesso e vasto ha reso necessaria una nuova metodologia di approccio ai problemi; infatti, la fase di pianificazione e di discussione teorica dei concetti da realizzare ha richiesto un tempo maggiore rispetto all'implementazione del codice.
Questo nuovo approccio sarà indubbiamente utile nelle future esperienze lavorative.

Possibili sviluppi futuri

Il progetto “De Astris”, ambizioso fin da quando è stato concepito, è stato pensato per uno sviluppo a lungo periodo e il nostro stage ne ha realizzato la prima parte. L'idea del planetario non si limita infatti alla visualizzazione delle stelle, ma prevede nel tempo il raggiungimento di livelli di dettaglio sempre più precisi dell'universo.

I moduli che saranno implementati in futuro comprenderanno la modellazione del sistema solare, onde consentire l'esplorazione a livello planetario.

Non appena il sistema solare sarà realizzato, i singoli pianeti, a partire dalla Terra, saranno codificati.

Si prospetta un'interessante espansione del progetto nel comprendere il modello del pianeta Marte e della Luna, in grado di consentire l'esplorazione libera dallo spazio fino al livello del suolo, senza soluzione di continuità.

La tecnologia UltraPeg consente la definizione di modelli delle sonde spaziali, da collocarsi nel luogo esatto corrispondente alla loro posizione nel mondo fisico, onde rendere possibile la visita virtuale ai luoghi più interessanti della storia della conquista dello spazio. Il sito dell'atterraggio dell'Apollo 11, la missione che ha visto il primo uomo sulla Luna, sarà il primo ad essere realizzato e proposto al pubblico via Internet.

Il progetto riveste un grande interesse didattico e divulgativo, che non si arresta alla prima realizzazione, il planetario virtuale, ma si potrà estendere ad ogni sorta di realizzazioni che, in ossequio ai principi della Fondazione Ultramundum, potranno giovarsi del materiale realizzato.

Possibili applicazioni saranno: film d'animazione, anche ambientati sulla Terra (che potranno includere un cielo scientificamente corretto), videogiochi, simulazioni didattiche di argomento spaziale ed astronautico, simulazioni e visualizzazione scientifiche e/o commerciali di missioni in essere o in progetto, ad esempio la ipotizzanda base lunare.

BIBLIOGRAFIA

[1]
Oliver Montenbruck, Thomas Pfleger
“Astronomy on the Personal Computer”
Springer, 2003

[2]
Jean Meeus
“Astronomical Algorithms”
Willmann-Bell, 1991

[3]
Bradley Carroll, Dale Ostlie
“Modern Astrophysics”
Addison Wesley, 1996

[4]
G. Romano
“Introduzione all'Astronomia”
Muzzio, 1985

[5]
Michael E. Mortenson
"Modelli geometrici in computer graphics"
McGraw-Hill, 1989

[6]
J. Foley, A. Van Dam, S. Feiner, J. Hughes, R. Phillips
“Introduction to Computer Graphics”
Addison Wesley, 1997

[7]
Donald Hearn, Pauline Baker
"Computer Graphics"
Prentice Hall, 1997

[8]
Tomas Akenine-Moller, Eric Haines
"Real-Time Rendering"
A K Peters
Natick Massachusetts, 2002

[9]
Mason Woo, Jackie Neider, Tom Davis, Dave Shreiner
"OpenGL Programming Guide
The official guide to learning OpenGL"
Addison-Wesley, 2000

[10]
David H. Eberly
"3D Game Engine Design
A Pratical Approach to Real-Time Computer Graphics"
Morgan Kauffmann, 2001

[11]
Richard S. Wright, Michael Sweet
"OpenGL SuperBible"
Waite Group Press, 2000

[12]
Alan Bertossi
"Strutture, Algoritmi, Complessità"
Ecig, 1993

SITOGRAFIA

[Sito1]
Sito della Fondazione Ultramundum
www.ultramundum.org

[Sito2]
Documentazione scientifica, il tomo De Astris
http://www.ultramundum.org/italy/ulita.htm

[Sito3]
Portale dell'ESA (European Space Agency)
www.esa.int/

[Sito4]
The Hipparcos Space Astrometry Mission
http://astro.estec.esa.nl/Hipparcos/

[Sito5]
Interrogazioni on-line dei cataloghi Hipparcos e Tycho
http://astro.estec.esa.nl/Hipparcos/HIPcatalogueSearch.html

[Sito6]
NASA
www.nasa.gov/home/index.html

[Sito7]
NASA Scientific Visualization Studio
http://svs.gsfc.nasa.gov/stories/

[Sito8]
Hubble Space Telescope
http://hubblesite.org/

[Sito9]
Homepage del sito di Celestia
www.shatters.net/celestia/

[Sito10]
Biblioteca on-line di cataloghi astronomici
http://vizier.u-strasbg.fr/viz-bin/VizieR

[Sito11]
Il Catalogo Messier
www.seds.org/messier/

[Sito12]
Il New General Catalog
www.seds.org/~spider/ngc/ngc.html

[Sito13]
Il progetto NGC/IC
www.ngcic.org

[Sito14]
Professor Stephen Hawking's web pages
www.hawking.org.uk

[Sito15]
Strasbourg Astronomical Data Center
http://cdsweb.u-strasbg.fr/cats/Cats.htx

[Sito16]
Sloan Digital Sky Survey
http://www.sdss.org/background/index.html

[Sito17]
Appunti di astronomia
www.astrosurf.com/cosmoweb/

[Sito18]
University of Illinois, Department of Astronomy
www.astro.uiuc.edu/~kaler/sow/sowlist.html

[Sito19]
Domande e risposte sull'astronomia
www.vialattea.net/esperti/astro.html

[Sito20]
The NASA Astrophysics Data System
http://adsabs.harvard.edu/ads_abstracts.html