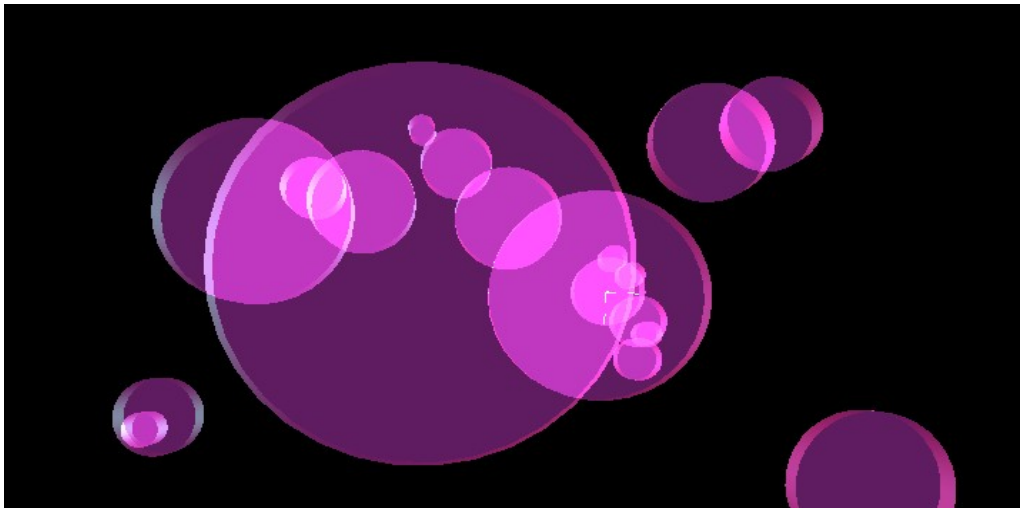
Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Σχολή Τεχνολογικών Εφαρμογών Τμήμα Εφαρμοσμένης Πληροφορικής & Πολυμέσων

Πτυχιακή Εργασία με Τίτλο:

Κατασκευή Τρισδιάστατων Χώρων σε Μορφότυπο OBJ



Τριγώνη Ειρήνη

Ηράκλειο, Μάρτιος 2010

Επιβλέπων Καθηγητής : Αθανάσιος Μαλάμος

Με την ολοκλήρωση της πτυχιακής μου εργασίας θα ήθελα να ευχαριστήσω όλους όσους με υποστήριξαν. Συγκεκριμένα, τον επόπτη – καθηγητή κύριο Αθανάσιο Μαλάμο για τις πολύτιμες συμβουλές του. Τους καθηγητές Eduardo Gutiérrez de Ravé και Francisco José Jiménez Hornero για την καθοδήγησή τους κατα την εκπόνηση της εργασίας. Θα ήθελα να απευθύνω τις ιδιαίτερες ευχαριστίες μου στους συναδέλφους και συμφοιτητές Antonio José Lázaro Muñoz, Μαρίνα Καμπίσιου, στους Ana Belén Ariza Villaverde, Nazareth Montilla και Fernando Sanchez García απο το εργαστήριο μηχανικών πληροφορικής για την υποστήριξή τους. Τέλος το τμήμα Εφαρμοσμένης Πληροφορικής & Πολυμέσων του Τεχνολογικού Εκπαιδευτικού Ιδρύματος Κρήτης για τη φιλοξενία του στα χρόνια της ακαδημαϊκής μου πορείας.

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΙΣΑΓΩΓΗ

Η αναφορά που ακολουθεί αποτελεί λεπτομερή καταγραφή των στόχων, της μεθοδολογίας, των προβλημάτων, των λύσεων αυτών και των αποτελεσμάτων της πτυχιακής εργασίας της συντάκτη αυτού του κειμένου και υποβάλλεται στο Τμήμα Εφαρμοσμένης Πληροφορικής & Πολυμέσων του Τ.Ε.Ι. Κρήτης στα πλαίσια της ολοκλήρωσης μέρους των υποχρεώσεων του σπουδαστή για την απόκτηση του τίτλου σπουδών του τμήματος.

Η ιδέα του θεματικού αντικειμένου της παρούσας πτυχιακής εργασίας γεννήθηκε στο Πανεπιστήμιο της Córdoba, Spain (Escuela Politécnica Superior de Córdoba, España) τον Σεπτέμβριο του 2008, στο τμήμα Γραφικών Μηχανικών και Μηχανικών Ανάπτυξης Πληροφοριακών Συστημάτων Χαρτογράφησης (Ingeniería Gráfica e Ingeniería y Sistemas de Información Cartográfica – UCO), όπου κλήθηκε η συντάκτης αυτής της πτυχιακής να υλοποιήσει μία εφαρμογή της οποίας τεχνικός στόχος θα ήταν να παράγει εύκολα και γρήγορα τρισδιάστατα αντικείμενα ή σύνολα αυτών σε ένα χώρο, σε τέτοια μορφή (.obj) ώστε να μπορεί να εισαχθεί σε άλλη εφαρμογή σαν δεδομένο η οποία παρουσίαζε το σύνολο των αντικειμένων σε πραγματικά τρισδιάστατη μορφή, αφού βέβαια ο χρήστης είχε στην κατοχή του τα ειδικά γυαλιά 3D. Υπο την επίβλεψη της εργασίας ήταν ο καθηγητής Eduardo Gutiérrez de Ravé, καθώς και ο καθηγητής Francisco José Jiménez Hornero.

Η εφαρμογή αυτή ονομάστηκε ObjCreator, γράφτηκε εξολοκλήρου σε java με τη χρήση ενός προγράμματος κειμένου. Για να μπορέσει να τρέξει σε οποιοδήποτε σύστημα, αρκεί η εγκατάσταση των java JRE και java3D. Παρακάτω στα επόμενα κεφάλαια καταγράφονται όλες οι απαραίτητες γνώσεις και λεπτομέρειες για την ανάπτυξη της εφαρμογής και για τα στοιχεία που χρησιμοποιήθηκαν.

ΚΕΦΑΛΑΙΟ1 - 3D COMPUTER GRAPHICS – ΤΡΙΣΔΙΑΣΤΑΤΑ ΓΡΑΦΙΚΑ ΥΠΟΛΟΓΙΣΤΗ

Τα γραφικά υπολογιστών (computer graphics) είναι ένας κλάδος της επιστήμης υπολογιστών που ασχολείται με τη θεωρία και την τεχνολογία σύνθεσης εικόνων σε ηλεκτρονικό υπολογιστή. Πρόκειται για γραφικά που χρησιμοποιούν τρισδιάστατη αναπαράσταση γεωμετρικών δεδομένων (συχνά Καρτεσιανών) τα οποία είναι αποθηκευμένα στον υπολογιστή για εκτελέσεις υπολογισμών και για αποδόσεις εικόνων σε δύο διαστάσεις. Αυτές οι εικόνες μπορεί να προορίζονται για υστερόχρονη έκθεση ή για real-time προβολή, ζωντανά. Το γεγονός οτι η απεικόνιση χρησιμοποιεί τρεις διαστάσεις τα καθιστά ιδιαίτερα ρεαλιστικά. Τέτοιου είδους γραφικά χρησιμοποιούνται συνήθως απο προγράμματα όπως παιχνίδια υπολογιστών, εικονικούς κόσμους. Τα τρισδιάστατα γραφικά βρίσκουν επίσης εφαρμογή στον κινηματογράφο, για τη δημιουργία σκηνών εικονικών κόσμων.

Τα τρισδιάστατα γραφικά ανάλογα με τον χρόνο στον οποίο γίνεται η απόδοσή τους(rendering) χωρίζονται σε στατικά και πραγματικού χρόνου. Τα στατικά γραφικά υπολογιστών αποτελούν αντικείμενα γραφικών τα οποία δεν αποδίδονται την στιγμή που εκτελούνται αλλά έχουν αποδοθεί μία φορά κατά τη δημιουργία τους. Παράδειγμα τέτοιων γραφικών είναι τα μικρά βίντεο, τα οποία εμφανίζονται σε διάφορα παιχνίδια, και τα οποία έχουν "γυριστεί" μια φορά και κάθε φορά που θα τα παρακολουθήσουμε παραμένουν ίδια. Για τη δημιουργία τους χρησιμοποιείται κάποιο πρόγραμμα δημιουργίας γραφικών και κίνησης (animation) όπως το 3dStudio Max, το Maya, το Lightwave, το Blender κτλ.

Τα γραφικά υπολογιστών πραγματικού χρόνου αποτελούν αντικείμενα γραφικών τα οποία αποδίδονται την στιγμή πυο εκτελούνται. Για παράδειγμα τα γραφικά που εμφανίζονται στην οθόνη ενός υπολογιστή, ο οποίος εκτελεί ένα παιχνίδι, ανήκουν συνήθως σε αυτήν την κατηγορία. Για τη δημιουργία τους απαιτείται κάποια μηχανή απόδοσης γραφικών (graphics rendering engine) πραγματικού χρόνου, όπως για παράδειγμα το Ogre3d, το Irrlich, το Crystal Space κτλ.

Η διαδικασία δημιουργίας τρισδιάστατων γραφικών στον υπολογιστή διαιρείται

διαδοχικά σε τρείς βασικές φάσεις.

(1). 3D modeling – το οποίο περιγράφει τη διαδικασία σχηματισμού ενός αντικειμένου.

(2). Layout and animation – που περιγράφει την κίνηση και την τοποθέτηση των αντικειμένων στην σκηνή.

(3). 3D rendering – το οποίο παράγει το ομοίωμα ενός αντικειμένου.

Αναφέρονται συχνά ως τρισδιάστατα μοντέλα (3D models). Ένα τρισδιάστατο μοντέλο είναι η μαθηματική αναπαράσταση οποιουδήποτε τρισδιάστατου αντικειμένου (είτε σταθερού είτε κινούμενου). Τεχνικά ένα μοντέλο γίνεται γραφικό μόνο όταν αναπαρασταθεί γραφικά. Μοντελοποίηση οπότε λέγεται η διαδικασία σχηματισμού του αντικειμένου. Οι πιο κοινές πηγές προέλευσης τρισδιάστατων μοντέλων είναι εκείνα που δημιουργήθηκαν σε υπολογιστή απο κάποιον καλλιτέχνη ή μηχανικό χρησιμοποιώντας κάποιο εργαλείο του 3D modeling, και εκείνα που σαρώθηκαν απο πραγματικά αντικείμενα του πραγματικού κόσμου και αναπαραστάθηκαν σε υπολογιστή.

Η διαδικασία του rendering γίνεται με δύο βασικές εργασίες. Transport (μεταφορά) , που ρυθμίζει το πόσο φως καθρεπτίζεται πού, και τις κατευθύνσεις που έχει αυτό, έτσι ώστε να φαίνεται πιο ρεαλιστικό, και Scattering (διασκορπισμός), που ρυθμίζει το πώς οι επιφάνειες αλληλεπιδρούν με το φωτισμό, η υφή της επιφάνειας δηλαδή. Πρέπει να σημειώσουμε οτι πριν τα αντικείμενα φτάσουν στη διαδικασία του rendering, πρέπει να τοποθετηθούν σε σκηνή, διότι αυτό καθορίζει τις σχέσεις των αντικειμένων στο χώρο. Τις θέσεις δηλαδή και τα μεγέθη τους.

1.1 ΔΙΕΠΑΦΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ ΕΦΑΡΜΟΓΩΝ ΤΡΙΣΔΙΑΣΤΑΤΩΝ ΓΡΑΦΙΚΩΝ – 3D GRAPHICS APPLICATION PROGRAMMING INTERFACES (APIs)

Διεπαφή ή διασύνδεση προγραμματισμού εφαρμογών ή API είναι ένα σύνολο

απο ρουτίνες, πρωτόκολλα, καθώς και εργαλεία για τη δημιουργία εφαρμογών λογισμικού. Ένα καλό API είναι πιο εύκολο να αναπτύξει ένα πρόγραμμα, παρέχοντας όλα τα δομικά στοιχεία. Ο προγραμματιστής τότε απλά συνδέει τα κομμάτια.

Τα περισσότερα λειτουργικά περιβάλλοντα, όπως το MS-Windows, παρέχουν ένα API, έτσι ώστε οι προγραμματιστές μπορούν να γράψουν εφαρμογές που να είναι συνάδουν με το περιβάλλον λειτουργίας. Παρόλο που τα APIs έχουν σχεδιαστεί για προγραμματιστές, είναι τελικά καλό για τους χρήστες, διότι εγγυάται ότι όλα τα προγράμματα που χρησιμοποιούν μια κοινή API θα έχουν παρόμοιες διεπαφές. Αυτό διευκολύνει τους χρήστες να μάθουν νέα προγράμματα.

Ένα API είναι μια αφηρημένη έννοια που ορίζει και περιγράφει μια διεπαφή για την αλληλεπίδραση με ένα σύνολο λειτουργιών που χρησιμοποιούνται από στοιχεία ενός συστήματος λογισμικού. Το λογισμικό που παρέχει τις λειτουργίες που περιγράφονται από ένα API λέγεται ότι είναι μια υλοποίηση του API.

1.1.1 Open Graphics Library (OpenGL)

Είναι ένα πρότυπο υλοποίησης βιβλιοθηκών σχεδίασης γραφικών. Εμπεριέχει δηλαδή το σύνολο των συναρτήσεων που πρέπει να υλοποιεί μια βιβλιοθήκη γραφικών προκειμένου να είναι συμβατή με αυτό. Το πρότυπο λοιπόν αυτό καθορίζει μια προγραμματιστική διεπιφάνεια (API). Είναι προφανές οτι δεν υπάρχει περιορισμός ως προς τη γλώσσα προγραμματισμού στην οποία θα υλοποιηθεί το πρότυπο της OpenGL.

Εφόσον με τον όρο OpenGL δεν αναφερόμαστε σε μια συγκεκριμένη βιβλιοθήκη αλλά σε ενα πρότυπο που ορίζει τη λειτουργικότητα μιας βιβλιοθήκης σχεδίασης, μπορούμε να ακολουθήσουμε τις ίδιες συμβάσεις σε όλες τις υλοποιήσεις του προτύπου (τις ίδιες εντολές). Αυτό σημαίνει ότι, εάν βασιστούμε στο πρότυπο της OpenGL, ο κώδικας που συντάσσουμε είναι ανεξάρτητος πλατφόρμας (platform independent) και μπορεί να εκτελεστεί σε ευρεία γκάμα περιβαλλόντων προγραμματισμού χωρίς ριζική τροποποίηση της δομής του.

Οι βιβλιοθήκες των περισσότερων νέων μεταγλωττιστών εμπεριέχουν ή υπάρχει δυνατότητα να ενσωματωθεί σε αυτούς μία υλοποίηση της OpenGL.

1.1.2 Mesa3D (OpenGL)

Ανοιχτού κώδικα βιβλιοθήκη που παρέχει μια υλοποίηση OpenGL για την απόδοση τρισδιάστατων γραφικών σε πολλαπλές πλατφόρμες. Αναπτύχθηκε αρχικά από τον Brian Paul το 1993.

1.1.3 VirtualGL

Ανοιχτού κώδικα πρόγραμμα που ανακατευθύνει τις εντολές 3D απόδοσης από Unix και Linux OpenGL εφαρμογές σε ένα υλικό 3D Accelerator σε έναν συγκεκριμένο server, και εμφανίζει την έξοδο διαδραστικά σε έναν thin client (ένας Η/Υ ή ένα πρόγραμμα Η/Υ που εξαρτάται κατά μεγάλο μέρος σε κάποιον server για να πραγματοποιήσει τους παραδοσιακούς υπολογιστικούς ρόλους του) , ο οποίος βρίσκεται κάπου αλλού μέσα στο δίκτυο.

1.1.4 Direct3D

Είναι μία προγραμματιστική διεπιφάνεια (Application Programming Interface ή API) από τη Microsoft για τα τρισδιάστατα γραφικά υπολογιστή. Είναι ένα συστατικό του DirectX, και έχει σχεδιαστεί για να παρέχει στα Windows εφαρμογές με δυνατή την άμεση πρόσβαση στο υλικό του υπολογιστή. Το Direct3D χρησιμοποιείται ευρέως κυρίως για τα ηλεκτρονικά παιχνίδια όπου η απόδοση είναι σημαντική (Xbox, Xbox 360) .

Επίσης επιτρέπει στις εφαρμογές να τρέχουν σε πλήρη οθόνη (FullScreen) αντί ενσωματωμένες σε ένα παράθυρο , αν και μπορούν να εξακολουθούν να εκτελούνται σε ένα παράθυρο εάν είναι προγραμματισμένα για τα εν λόγω χαρακτηριστικά.

Συγκριτικά με το OpenGL, το Direct3D είναι ένα ιδιόκτητο API που παρέχει λειτουργίες που αναπαριστούν τρισδιάστατα γραφικά, και χρησιμοποιεί την επιτάχυνση υλικού, εάν είναι διαθέσιμη στην κάρτα γραφικών. Σχεδιάστηκε από την Microsoft Corporation για χρήση στην πλατφόρμα των Windows.

Το OpenGL είναι ένα ανοιχτό πρότυπο API που παρέχει μια σειρά από λειτουργίες για την απόδοση 2D και 3D γραφικών και είναι διαθέσιμο στα περισσότερα σύγχρονα λειτουργικά συστήματα.

1.1.5 Java 3D API

Υπάρχουν 3 είδη API της προγραμματιστική γλώσσας java:

- Το επίσημο Java API, το οποίο συμπεριλαμβάνεται στο JDK ή στο JRE, σε μια από τις εκδόσεις της πλατφόρμας Java. Οι τρεις εκδόσεις είναι οι Java ME (Micro Edition), Java SE (Standard Edition), Java EE (Enterprise Edition).
- Προαιρετικά επίσημα APIs μπορούν να αποθηκευθούν χωριστά. Οι προδιαγραφές για αυτές τις API ορίζονται σύμφωνα με το JSR (Java Specification Request) και μερικές φορές κάποια από αυτά τα API περιλαμβάνονται αργότερα στον πυρήνα API της πλατφόρμας (παράδειγμα αυτού η Swing).
- Ανεπίσημη API που αναπτύχθηκε από τρίτους, αλλά δεν συνδέεται με κανέναν JSR.

1.1.6 RenderMan Interface Specification (RISpec)

Είναι ένα ανοιχτού κώδικα API ανεπτυγμένο από την Pixar Animation Studios για να περιγράψει τρισδιάστατες σκηνές και να τις μετατρέπει σε ψηφιακές φωτορεαλιστικές εικόνες. Περιλαμβάνει την RenderMan Shading Language.

1.1.7 Glide API

Είναι ένα ιδιόκτητο API τρισδιάστατων γραφικών ανεπτυγμένο από την 3Dfx Interactive για τις Voodoo Graphics 3D accelerator κάρτες γραφικών τους. Ήταν προορισμένο για gaming επιδόσεις, υποστηρίζοντας πρωταρχικά τη χαρτογράφηση γεωμετρίας και υφής, σε μορφές δεδομένων πανομοιότυπες με αυτές που χρησιμοποιούνται εσωτερικά στις κάρτες τους.

1.1.8 RenderWave (RW)

Είναι middleware[1] για computer και video παιχνίδια από την Criterion Software Limited. Είναι δια-πλατφορμικό, αφού μπορεί να τρέξει το ίδιο καλά σε εφαρμογές βασισμένες στα Windows όσο και σε Apple Mac OS, καθώς και σε πολλές κονσόλες video game όπως GameCube, Wii, Xbox, Xbox 360, PlayStation 2, PlayStation 3 και PlayStation Portable. Η RenderWave δεν είναι πλέον διαθέσιμη για αγορά, αλλά οι ήδη χρήστες αυτών των υλικών μπορούν ακόμη να τη χρησιμοποιούν.

1.1.9 QuickDraw 3D (QD3D)

Είναι ένα API τρισδιάστατων γραφικών ανεπτυγμένο από την Apple Inc., αρχικά το 1995 για τους υπολογιστές Macintosh, αλλά αργότερα σαν δια-πλατφορμικό σύστημα. Η QD3D παρείχε ένα API υψηλού επιπέδου με ένα πλούσιο σετ από στοιχεία 3D που ήταν γενικά πολύ πιο πλούσιο σε δυνατότητες αλλά και πιο εύκολο στην υλοποίηση από τα χαμηλότερου επιπέδου APIs όπως OpenGL ή Direct3D.

Υπάρχουν και άλλα υψηλού επιπέδου API που παρέχουν επιπρόσθετες λειτουργίες στα χαμηλότερου επιπέδου API. Αυτές οι βιβλιοθήκες περιλαμβάνουν τα : QSDK, Quesa, JSR 184 (M3G), VegaPrime by Presagis, Nvidia Scene Graph, VR – Vantage by VTMak, OpenGL Performer, OpenSceneGraph, OpenSG, OGRE, CrystalSpace, JmonkeyEngine, Irrlicht Engine, Hoops3D, UGS DirectModel, ClanLib.

1.2 ΑΡΧΕΙΑ ΑΠΟΘΗΚΕΥΣΗΣ ΤΡΙΣΔΙΑΣΤΑΤΩΝ ΜΟΝΤΕΛΩΝ

Ο τρόπος με τον οποίο περιγράφεται η γεωμετρία κάποιου αντικειμένου και οι επιπλέον πληροφορίες σχετικά με αυτή, όπως για παράδειγμα οι συντεταγμένες

1 Middleware είναι γενικός όρος για κάθε προγραμματισμό που ενώνει ή λειτουργεί σαν μεσάζων ανάμεσα σε δύο χωριστά και συχνά ήδη υπάρχοντα προγράματα. Συνήθως τα λογισμικά middleware παρέχουν υπηρεσίες επικοινωνίας, έτσι ώστε διαφορετικές εφαρμογές να επικοινωνούν.

που ορίζουν τη θέση της υφής σε σχέση με τη γεωμετρία, καθορίζονται από τη φόρμα του αρχείου αποθήκευσης της πληροφορίας αυτής.

Οι πιο ευρέως διαδεδομένοι τύποι αρχείων για την αποθήκευση τρισδιάστατων θεμάτων αριθμούν είναι γύρω στους 40, ενώ ο αριθμός τους συνεχώς αυξάνεται. Ένας γενικός διαχωρισμός που μπορεί να γίνει έτσι ώστε να απλοποιηθεί η διαδικασία επιλογής της πιο κατάλληλης λύσης, είναι αυτός μεταξύ των αρχείων όπου η πληροφορία αποθηκεύεται στη μορφή κειμένου και των αρχείων όπου η πληροφορία αποθηκεύεται σε δυαδική μορφή. Η αποθήκευση της πληροφορίας που αφορά την περιγραφή ενός θέματος σε τρεις διαστάσεις, όταν υλοποιείται σε μορφή κειμένου καταλαμβάνει πολύ περισσότερο αποθηκευτικό χώρο απ' ότι θα καταλάμβανε σε δυαδική μορφή, για το λόγο ότι οι αριθμοί που δηλώνουν τις τρισδιάστατες συντεταγμένες της πληροφορίας εκφράζονται με χαρακτήρες, δηλαδή κείμενο αναγνώσιμο από τον άνθρωπο, με συνέπεια ένας αριθμός να καταλαμβάνει τόσα byte αποθηκευτικού χώρου όσα είναι τα ψηφία του. Για παράδειγμα ο αριθμός 255 δυαδικά εκφράζεται με 1 Byte (8 bit – 11111111), ενώ όταν εκφράζεται σε μορφή κείμενου απαιτεί 3 byte (3 χαρακτήρες), καταλαμβάνοντας έτσι 3 φορές περισσότερο αποθηκευτικό χώρο.

Πέρα από τον παραπάνω βασικό διαχωρισμό, από τον οποίο εξαρτάται το μέγεθος του απαιτούμενου αποθηκευτικού χώρου, η επιλογή της φόρμας με την οποία θα υλοποιηθεί η αποθήκευση της ψηφιοποιημένης πληροφορίας είναι καθοριστική τόσο για τη βιωσιμότητα της πληροφορίας αυτής, όσο και για τη συμβατότητά της με τα διάφορα πακέτα λογισμικού. Επίσης, εξίσου καθοριστικός για την επιλογή του τύπου αρχείου αποθήκευσης, είναι και ο τρόπος με τον οποίο περιγράφεται η τρισδιάστατη γεωμετρία κάποιου θέματος, διότι η υποστήριξη κάποιων εξελιγμένων χαρακτηριστικών, όπως για παράδειγμα οι παραμετρικές επιφάνειες, μπορεί να μην συναντάται σε κάποιους τύπους αρχείων. Για τους παραπάνω λόγους, η αποκλειστική χρήση κάποιου μη διαδεδομένου τύπου αρχείου πρέπει να αποφεύγεται, αφού είναι πιθανό στο μέλλον να σταματήσει η υποστήριξή του και να εξαφανιστεί. Αντιθέτως, η μετατροπή και αποθήκευση της πληροφορίας σε περισσότερους του ενός κοινά

αποδεκτούς τύπους αρχείων και η περιοδική μετανάστευσή της σε πιο σύγχρονους, αποτελεί μια εγγυημένη λύση διατήρησής της.

1.2.1 Χαρακτηριστικά αρχείων αποθήκευσης τρισδιάστατων μοντέλων

Τα βασικά χαρακτηριστικά που πρέπει να λαμβάνονται υπόψη για την επιλογή του αρχείου αποθήκευσης του τελικού προϊόντος της τρισδιάστατης σάρωσης, για λόγους αρχειακής αποθήκευσης, είναι τα εξής:

- Δυαδική αποθήκευση της πληροφορίας για μείωση του όγκου των δεδομένων.
- Περιγραφή των τρισδιάστατων επιφανειών τουλάχιστον με πολύγωνα. Επιθυμητή είναι η υποστήριξη παραμετρικών επιφανειών, έτσι ώστε μέσω του κατάλληλου λογισμικού να επιτευχθεί περαιτέρω μείωση των δεδομένων.
- Τουλάχιστον από ένα ζεύγος συντεταγμένων για κάθε τρισδιάστατο σημείο της γεωμετρίας, για την εφαρμογή μιας η περισσότερων εικόνων υφής.
- Δυνατότητα αναγνώρισης από δημοφιλή εφαρμογές του είδους, όπως για παράδειγμα λογισμικό τρισδιάστατης μοντελοποίησης και κίνησης, φωτορεαλιστικής απόδοσης, μετατροπής σε άλλους τύπους αρχείων(format) και λοιπά.

Όσον αφορά τη χρήση των δεδομένων αυτών σε εφαρμογές ειδικού τύπου, όπως είναι για παράδειγμα η προβολή μέσω του διαδικτύου και η πραγματικού χρόνου, διαδραστική παρουσίασή της, συνιστάται η επιπλέον μετατροπή της σε μορφές αρχείων περισσότερο κατάλληλες για τις εφαρμογές αυτές. Τα βασικά κριτήρια επιλογής αρχείου αποθήκευσης που πρέπει να λαμβάνονται υπόψη για εφαρμογές τέτοιου ίδιους, εν μέρη περιγράφτηκαν παραπάνω. Παρόλα αυτά, οι ιδιαίτερες απαιτήσεις της τρισδιάστατης απεικόνισης πραγματικού χρόνου και η πιθανή μεταφορά της πληροφορία μέσω του διαδικτύου, ή κάποιου άλλου αργού δίαυλου επικοινωνίας, προτρέπουν στη χρήση ειδικών τύπων αρχείων με τα εξής ιδιαίτερα χαρακτηριστικά:

- Επίπεδο λεπτομέρειας (LOD-Level of Detail) συνεχόμενο ή προκαθορισμένο. Το χαρακτηριστικό αυτό αφορά την τρισδιάστατη απεικόνιση του θέματος σε πραγματικό χρόνο και σχετίζεται με την ποσότητα των πολύγονων που περιγράφουν το θέμα (λεπτομέρεια), κάθε φορά που αυτό αποδίδεται γραφικά στην οθόνη του Η/Υ, ανάλογα με την απόσταση του αντικειμένου από τη θέση της ιδεατής κάμερας και κατά συνέπεια της επιφάνειας που αυτό καταλαμβάνει στην οθόνη του Η/Υ. Έτσι, όσο πιο μικρή είναι η επιφάνεια που καταλαμβάνει το θέμα στην οθόνη του υπολογιστή, τόσο λιγότερη είναι η οπτική πληροφορία που μπορεί να αποδοθεί σε αυτή και συνεπώς τόσο πιο απαρατήρητη είναι η χρήση λιγότερων πολύγωνων για την οπτική απόδοσή του θέματος. Η τεχνική αυτή έχει σαν αποτέλεσμα την ταχύτερη οπτική απόδοση της τρισδιάστατης σκηνής και εφαρμόζεται σε περιπτώσεις ιδεατών περιπάτων.
- Βαθμιαίο φόρτωμα και απεικόνιση της πληροφορίας. Αφορά την έκφραση της πληροφορίας με τέτοιο τρόπο ώστε κατά τη μεταφόρτωσή της, όσο χρονοβόρα και αν είναι αυτή, να καθίσταται εφικτή η απεικόνιση ολόκληρου του θέματος ακόμα και από τα πρώτα στάδια της διαδικασίας. Με την τεχνική αυτή, η απόδοση του θέματος στην αρχή είναι χονδρική και με το χρόνο, καθώς γίνονται διαθέσιμα περισσότερα δεδομένα, αποκτά σταδιακά όλο και περισσότερη λεπτομέρεια.
- Συμπίεση και κωδικοποίηση της πληροφορίας για περαιτέρω μείωση του όγκου που καταλαμβάνει και συνεπώς του όγκου που πρέπει να μεταφερθεί μέσω του διαδικτύου.
- Μηχανισμούς διασφάλισης της πνευματικής ιδιοκτησίας των δεδομένων, μέσω υδατογραφίας ή άλλων τεχνικών, και περιορισμού της χρήσης τους μόνο για κάποιο συγκεκριμένο σκοπό (π.χ. μόνο για τη θέαση μέσω κάποιου συγκεκριμένου λογισμικού).

1.2.2 Μορφότυπα αρχείων αναπαράστασης τρισδιάστατων αντικειμένων και χαρακτηριστικά

Τα αρχεία OBJ της Wavefront αποθηκεύονται σε μορφή κειμένου (ASCII) ενώ εκφράζει την πληροφορία χρησιμοποιώντας τρίγωνα και παραμετρικές επιφάνειες. Περιέχει επιπλέον συντεταγμένες υφής και συναντά μεγάλη διάδοση, και μάλιστα στην παράγραφο 4.3 παραθέτονται τα περισσότερα λογισμικά που υποστηρίζουν αυτή τη μορφή. Δεν έχει μεγάλο επίπεδο λεπτομέρειας, βαθμιαίο φόρτωμα ούτε κανέναν βαθμό συμπίεσης. Επίσης δεν θεωρείται ασφαλές, καθώς είναι σε μορφή κειμένου και ο οποιοσδήποτε μπορεί να το επεξεργαστεί. Ακολουθεί μία λίστα όπου περιλαμβάνει όλα τα αρχεία που υποστηρίζουν τρισδιάστατα γραφικά, καθώς και κάποια χαρακτηριστικά τους.

Από το πρόγραμμα 3D Studio μπορούμε να εξάγουμε αρχεία .3ds και .asc. Το πρώτο αποθηκεύει το αντικείμενο σε δυαδική μορφή, χρησιμοποιεί τρίγωνα για να εκφράσει την τρισδιάστατη πληροφορία και χρησιμοποιεί επίσης επιπλέον συντεταγμένες υφής. Έχει μεγάλη διάδοση και μέτρια ασφάλεια. Το δεύτερο είδος αρχείου (.asc) περιέχει κείμενο, είναι δηλαδή γραμμένο σε μορφή ASCII, και χρησιμοποιεί και αυτό τρίγωνα για να αναπαραστήσει τις τρισδιάστατες επιφάνειες. Και αυτό περιλαμβάνει επιπλέον συντεταγμένες υφής. Έχει και αυτό μεγάλη διάδοση και ανύπαρκτη ασφάλεια, καθώς μπορεί ο οποιοσδήποτε να ανοίξει το αρχείο με επεξεργασία κειμένου και να το πειράξει.

Δύο αρχεία του 3D Studio Max είναι τα .ase και .max. Και τα δύο αναπαριστούν τις τρισδιάστατες επιφάνειες χρησιμοποιώντας τρίγωνα και παραμετρικές επιφάνειες, ενώ έχουν και επιπλέον συντεταγμένες υφής και φυσικά έχουν μεγάλη διάδοση. Το πρώτο, το .ase είναι σε μορφή ASCII, άρα είναι σε μορφή κειμένου, και μπορεί να διαβαστεί από το χρήστη, άρα και να υφισταθεί επεξεργασία από αυτόν. Το αρχείο με κατάληξη .max είναι γραμμένο σε δυαδική μορφή άρα και περισσότερο ασφαλές.

Άλλα αρχεία είναι τα .fbx (Alias FBX), .cob (Caligari True Space), .c4d (Cinema 4D), iob (Imagine). Όλα τα παραπάνω αποθηκεύονται με δυαδικό τρόπο, εκφράζουν την τρισδιάστατη πληροφορία χρησιμοποιώντας τρίγωνα και κάποια και με χρήση παραμετρικών επιφανειών. Μεγαλύτερη διάδοση μάλλον θα λέγαμε ότι έχει το πρώτο (.fbx) το οποίο έχει και μία μέτρια ασφάλεια.

Τα αρχεία .dfx και .dxb (AutoCAD), .x (DirectX), .fact (Electric Image), .iges και .igs (Initial Graphics Exchange Specific Format G-code) αποθηκεύονται όλα και δυαδικά και σε κείμενο και όλα εκφράζουν την τρισδιάστατη πληροφορία χρησιμοποιώντας τρίγωνα και παραμετρικές επιφάνειες. Από αυτά ωστόσο, μόνο τα .x και .fact χρησιμοποιούν και επιπλέον συντεταγμένες για την υφή. Μεγαλύτερη διάδοση από αυτά έχουν φυσικά τα αρχεία του AutoCAD, και του DirectX.

Κάποια αρχεία που αποθηκεύονται σε μορφή κειμένου είναι τα .geo (AOFF), .off (Object File Format), .ma (Maya ASCII), .raw (POV-Ray RAW Triangle Format), .pts (Points File), .rax (Extended raw triangles), .xsi (Softimage), .objf (Stripe), .vrml και .wrl (Virtual Reality Modelling Language), .xgl (XGL) και φυσικά το .obj (WaveFront). Από αυτά αξίζει να αναφερθούμε περισσότερο στα .vrml και .wrl που έχουν και αρκετά μεγάλη διάδοση. Χρησιμοποιεί τρίγωνα και παραμετρικές επιφάνειες για να αναπαραστήσει τρισδιάστατες επιφάνειες. Περιλαμβάνει επιπλέον συντεταγμένες υφής και έχει και ένα κάποιο επίπεδο λεπτομέρειας. Επιτρέπει το βαθμιαίο φόρτωμα και μπορεί να συμπιεστεί μέσω λογισμικού συμπίεσης σε gzip.

Αρχεία που αποθηκεύονται σε δυαδική μορφή είναι τα .lwo και .lw (LightWave3D), .mb (Maya ASCII), .mts (Metacreations Metastream), .ndo (Nendo), .ngn και .ngw (NGRain), .flt (OpenFlight), .iv (Open Inventor), .pro (Power Render Object), .3dm (Rhinoceros), .w3d (Shockwave 3D), .u3d (Universal 3D), .vpp (Viewpoint). Από αυτά αξίζει να αναφερθούμε λίγο παραπάνω στο .mts που έχει καλό επίπεδο λεπτομέρειας, βαθμιαίο φόρτωμα και συμπίεση, στο .flt που έχει κι αυτό καλό επίπεδο λεπτομέρειας και είναι περισσότερο διαδεδομένο από το προηγούμενο, το .vpp που δεν είναι πολύ διαδεδομένο αλλά έχει καλό επίπεδο λεπτομέρειας και βαθμιαίο φόρτωμα, καθώς και μέτριο επίπεδο ασφάλειας. Τέλος το .u3d που ούτε κι αυτό έχει ιδιαίτερα μεγάλη διάδοση ακόμη, αλλά αναμένεται να χρησιμοποιείται περισσότερο στο μέλλον. Αναπαριστά τις τρισδιάστατες επιφάνειες χρησιμοποιώντας τρίγωνα, χρησιμοποιεί επιπλέον συντεταγμένες υφής και έχει

καλό επίπεδο λεπτομέρειας, βαθμιαίο φόρτωμα και συμπίεση.

Τέλος ας αναφέρουμε και το .x3d (X3D) που αποθηκεύεται σε δυαδική και σε μορφή κειμένου, χρησιμοποιεί τρίγωνα και παραμετρικές επιφάνειες για να εκφράσει τρισδιάστατα αντικείμενα, επιτρέπει επιπλέον συντεταγμένες υφής. Έχει κάποια υποτυπώδη επίπεδα λεπτομέρειας, επιτρέπει το βαθμιαίο φόρτωμα, και την συμπίεση. Πέρα απο τα είδη αρχείων που αναφέραμε υπάρχουν και άλλα τα οποία δεν συναντούν μεγάλη διάδοση.

ΚΕΦΑΛΑΙΟ2 - JAVA Η ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

η java είναι μία γλώσσα προγραμματισμού αρχικά αναπτυγμένη απο τον James Gosling στη Sun Microsystems, η οποία ανακοινώθηκε το 1995 ως βασική συνιστώσα της Java Platform της εταιρείας. Οι αρχικοί Java compilers (μεταγλωττιστές), virtual machines (εικονικές μηχανές), και class libraries (βιβλιοθήκες κλάσεων) αναπτύχθηκαν απο τη Sun απο το 1995 και έπειτα. Η Sun ανακοίνωσε την πρώτη έκδοση Java 1.0 το 1995. Υποσχόταν το γνωστό σαν σλόγκαν “Write Once, Run Anywhere”. Σύντομα, η Sun έκανε διαθέσιμες τις περισσότερες εφαρμογές της Java σε χαμηλό κόστος, διαχωρίζει τα Software Development Kit (SDK) - εργαλεία ανάπτυξης λογισμικού και Run-time Environment (JRE) – περιβάλλον εκτέλεσης λογισμικού. Στις 13 Νοεμβρίου 2006, η Sun ανακοινώνει οτι μεγάλο μέρος της Java γίνεται δωρεάν και open source υπό τους όρους του GNU General Public License (GPL).

2.1 ΒΑΣΙΚΑ ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΤΗΣ ΓΛΩΣΣΑΣ

Η γλώσσα αντλεί μεγάλο μέρος του συντακτικού της απο την C και την C++, αλλά έχει πιο απλό object model, πολυ πιο έντονο αντικειμενοστραφή (object oriented) χαρακτήρα σε σχέση με την C++ και χαρακτηρίζεται για την απλότητα της, καθώς λείπουν διάφορα κομμάτια της C όπως οι δείκτες, οι δομές και οι ενώσεις. Object Model είναι η συλλογή αντικειμένων και κλάσεων μέσω των οποίων ένα πρόγραμμα μπορεί να εξετάσει και να χειριστεί κάποια συγκεκριμένα κομμάτια του κόσμου του. Με άλλα λόγια αυτή είναι η αντικειμενοστραφής διεπαφή ενός συστήματος.

Η Java όπως και η C, χρησιμοποιεί τις κλάσεις (classes) για να οργανώσει τον κώδικα σε λογικές ενότητες. Κατα το χρόνο εκτέλεσης, το πρόγραμμα δημιουργεί απο τις κλάσεις αντικείμενα. Τα αντικείμενα αυτά έχουν δύο συνιστώσες: τα πεδία και τις μεθόδους. Τα πεδία περιγράφουν τί είναι το

αντικείμενο, ενώ οι μέθοδοι περιγράφουν τί κάνει το αντικείμενο.

Η μεταγλώττιση (compile) του πηγαίου προγράμματος έχει σαν αποτέλεσμα την παραγωγή ενός ειδικού κώδικα, ο οποίος ονομάζεται bytecode. Ο κώδικας αυτός ενώ μοιάζει με τη γλώσσα μηχανής, μπορεί να εκτελεστεί απο οποιοδήποτε λειτουργικό σύστημα που διαθέτει ερμηνευτή της Java.

Σημαντικό χαρακτηριστικό της είναι η ασφάλεια εκτέλεσης του κώδικα σε δίκτυο, ενώ χρησιμοποιεί επίσης έναν ισχυρό μηχανισμό για τον έλεγχο αναμενόμενων και μή αναμενόμενων σφαλμάτων (exception handling).

Υποστηρίζει multithreading (πολυνημάτωση), δηλαδή μπορεί να περιλαμβάνει πολλές ξεχωριστές διαδικασίες, οι οποίες να εκτελούνται συνεχώς και ανεξάρτητα η μία απο την άλλη.

Μπορεί να δημιουργήσει γρήγορο κώδικα, καθώς οι εταιρείες έχουν αναπτύξει ειδικούς compilers που μετατρέπουν το bytecode σε κώδικα γλώσσας μηχανής ο οποίος ανταγωνίζεται σε ταχύτητα τον κώδικα της C++ ή άλλων γλωσσών. Οι compilers αυτοί ονομάζονται “Just in time compilers” (JIT).

Παρέχει libraries (βιβλιοθήκες κώδικα) για διάφορες χρήσεις, όπως δημιουργία γραφικών στο διαδίκτυο, χειρισμό σχεσιακών βάσεων δεδομένων, δημιουργία εφαρμογών πελάτη – εξυπηρετητή, κλήση απομακρυσμένων αντικειμένων, κτλ.

Ένα απο τα βασικά πλεονεκτήματα της java έναντι των περισσότερων άλλων γλωσσών είναι η ανεξαρτησία του λειτουργικού συστήματος και πλατφόρμας. Τα προγράμματα που είναι γραμμένα σε java τρέχουν ακριβώς το ίδιο σε όλα τα λειτουργικά συστήματα χωρίς να χρειαστεί να γίνει μεταγλώττιση (compiling) ή να αλλάξει ο πηγαίος κώδικας για καθένα είναι διαφορετικό λειτουργικό σύστημα.

Για να επιτευχθεί όμως αυτό χρειαζόταν κάποιος τρόπος έτσι ώστε τα προγράμματα γραμμένα σε java να μπορούν να είναι “κατανοητά” απο κάθε υπολογιστή ανεξάρτητα του είδους επεξεργαστή αλλά και του λειτουργικού συστήματος (Windows, Unix, Linux, Mac-OS). Ο λόγος είναι οτι κάθε κεντρική μονάδα επεξεργασίας κατανοεί διαφορετικό κώδικα μηχανής. Η λύση δόθηκε με την ανάπτυξη της Εικονική Μηχανής (ΕΜ) ή Virtual Machine (VM).

2.2 Η ΕΙΚΟΝΙΚΗ ΜΗΧΑΝΗ ΤΗΣ JAVA

Αφού γραφεί κάποιο πρόγραμμα σε Java τότε μεταγλωττίζεται μέσω του εργαλείου javac, το οποίο παράγει έναν αριθμό απο αρχεία .class που περιέχουν bytecode. Για την εκτέλεση λοιπόν μίας εφαρμογής το Java Virtual Machine που πρέπει να είναι εγκατεστημένο στο σύστημα, θα αναλάβει να διαβάσει τα αρχεία .class και να τα μεταφράσει σε γλώσσα και εντολές μηχανής (assembly) που υποστηρίζει το συγκεκριμένο λειτουργικό σύστημα και επεξεργαστής, έτσι ώστε να εκτελεστεί. Χωρίς αυτό δε θα ήταν δυνατή η εκτέλεση λογισμικού γραμμένου σε Java. Η Java Virtual Machine είναι λογισμικό εξαρτημένο απο την πλατφόρμα, δηλαδή για κάθε είδος λειτουργικού συστήματος και αρχιτεκτονικής επεξεργαστή υπάρχει διαφορετική έκδοσή του. Έτσι υπάρχουν διαφορετικές JVM για Windows, Linux, Unix, Macintosh, κινητά τηλέφωνα, παιχνιδομηχανές κλπ.

Οτιδήποτε θέλει να κάνει ο ο προγραμματιστής ο χρήστης γίνεται μέσω της εικονικής μηχανής. Αυτο βοηθάει στο να υπάρχει μεγαλύτερη ασφάλεια στο σύστημα γιατί η εικονική μηχανή είναι υπεύθυνη για την επικοινωνία χρήστη-υπολογιστή. Ο προγραμματιστής δεν μπορεί να γράψει κώδικα ο οποίος θα έχει καταστροφικά αποτελέσματα για τον υπολογιστή γιατί η εικονική μηχανή θα το ανιχνεύσει και δε θα επιτρέψει να εκτελεστεί. Απο την άλλη μεριά ούτε ο χρήστης μπορεί να κατεβάσει “κακό” κώδικα απο το δίκτυο και να τον εκτελέσει. Αυτό είναι ιδιαίτερα χρήσιμο για μεγάλα κατανεμημένα συστήματα οπου πολλοί χρήστες χρησιμοποιούν το ίδιο πρόγραμμα συγχρόνως.

2.3 GARBAGE COLLECTOR (ΣΥΛΛΕΚΤΗΣ ΑΠΟΡΡΙΜΑΤΩΝ)

Είναι μία ακόμη ιδέα που βρίσκεται πίσω απο τη Java. Συλλογή απορριμμάτων είναι μια κοινή ονομασία που χρησιμοποιείται στον τομέα της πληροφορικής για να δηλώσει την ελευθέρωση τμημάτων μνήμης απο δεδομένα που δε χρειάζονται και δε χρησιμοποιούνται άλλο. Αυτή η απελευθέρωση μνήμης στη Java είναι αυτόματη και γίνεται μέσω του συλλέκτη απορριμμάτων. Υπεύθυνη

για αυτό είναι και πάλι εικονική μηχανή (Virtual Machine) η οποία μόλις “καταλάβει” οτι ο σωρός (heap) της μνήμης (στη Java η συντριπτική πλειοψηφία των αντικειμένων αποθηκεύονται στο σωρό σε αντίθεση με τη C++ όπου αποθηκεύονται κυρίως στη στοίβα - stack) κοντεύει να γεμίσει, ενεργοποιεί το συλλέκτη απορριμμάτων. Έτσι ο προγραμματιστής δεν χρειάζεται να ανησυχεί για το πότε και αν θα ελευθερώσει ένα συγκεκριμένο τμήμα της μνήμης, ούτε και για δείκτες (pointers) που αναφέρονται σε άδειο χώρο μνήμης.

2.4 JAVA 3D

Είναι μια top-down προσέγγιση για τη δημιουργία τρισδιάστατων αλληλεπιδραστικών προγραμμάτων που μπορούν να εκτελεστούν σε έναν web browser ή αυτόνομα ως εφαρμογές. Υλοποιημένο πάνω στη γλώσσα προγραμματισμού Java, χρησιμοποιεί μια ιεραρχία scene graph ως βασική δομή. Υποστηρίζονται διάφορες γεωμετρικές παραστάσεις, animation/interaction, φωτισμός, υφή , collision detection και ήχος.
Η πρώτη του έκδοση ανακοινώθηκε τον Δεκέμβριο του 1998.
Σε σύγκριση με άλλες λύσεις, η Java 3D δεν είναι μόνο ένα περίβλημα γύρω απο τα API γραφικά, είναι μία διεπαφή που ενσωματώνει τον προγραμματισμό γραφικών χρησιμοποιώντας ένα πραγματικό αντικειμενοστραφές σχέδιο.
Η Java 3D και το documentation είναι διαθέσιμα στο διαδίκτυο ξεχωριστά , και όχι μέρος του Java Development Kit.

2.4.1 Χαρακτηριστικά της Java3D

- πολυνηματική δομή σκηνής γραφήματος.
- Ανεξάρτητου πλατφόρμας.
- Αναπαράσταση πραγματικού χρόνου, χρησιμοποιήσιμο τόσο για visualization όσο και για gaming.
- Υποστήριξη για retained, compiled-retained, και immediate mode

rendering.

- Περιλαμβάνει hardware accelerated JOGL, OpenGL και Direct3D renderers (ανάλογα την πλατφόρμα).
- Εξελιγμένη αναπαράσταση μοντέλου βασισμένο στην εικονική πραγματικότητα με υποστήριξη στερεοσκοπικής απόδοσης και πολύπλοκων ρυθμίσεων προβολής.
- Εγγενής υποστήριξη για head-mounted προβολή.
- CAVE (πολλαπλές προβολές στην οθόνη).
- χωρικός ήχος 3D.
- Προγραμματιζόμενες σκιάσεις, υποστηρίζοντας GLSL και CG.
- Stencil Buffer.
- Importers για εισαγωγή των περισσότερων mainstream φορμάτ όπως 3DS, OBJ, VRML, X3D, NWN, και FLT.

Για την ανάπτυξη της παρακάτω εφαρμογής απαιτείται εγκατάσταση του λογισμικού της Java στον υπολογιστή, εξίσου το ίδιο απαιτείται εγκατάστασή της σε κάθε υπολογιστή που θέλουμε να προβάλουμε την εφαρμογή. Χρησιμοποιήθηκαν οι εξής εκδόσεις: Java3D 1.5.2 , jdk1.6.0_12 , jre1.6.0_12 . Η ανάπτυξη του κώδικα της εφαρμογής έγινε μέσα απο το πρόγραμμα Notepad++ το οποίο κυκλοφορεί ελεύθερα και χωρίς χρέωση μέσω του διαδικτύου. [http://notepad-plus.sourceforge.net/uk/site.htm]. Το λογισμικό που χρησιμοποιήθηκε για την προβολή των εξερχόμενων αρχείων *.obj είναι το GLC_Player 2.1.0 το οποίο κυκλοφορεί επίσης ελεύθερα στο διαδίκτυο μέσα απο την ιστοσελίδα http://www.glc-player.net/download.php . Και τέλος τα κουμπιά της διεπαφής δημιουργήθηκαν εξολοκλήρου με το Adobe Photoshop CS2 version 9.0.2.

2.5 ΕΚΤΕΛΕΣΙΜΑ ΑΡΧΕΙΑ .JAR

Ένα αρχείο .jar ορίζεται σαν ένα μορφότυπο συνάθροισης πολλών αρχείων σε ένα. Οι μηχανικοί λογισμικού γενικά χρησιμοποιούν αρχεία .jar για να διανέμουν

εφαρμογές java ή βιβλιοθήκες, σε μορφή κλάσεων, συσχετισμένων metadata και πηγών (κειμένων, εικόνων, κτλ.). Τα αρχεία .jar έχουν βασιστεί στο μορφότυπο αρχείου .zip. Ένας χρήστης μπορεί να εξάγει ή να μετατρέψει αρχεία .jar με την αντίστοιχη εντολή που δουλεύει με ένα εγκατεστημένο JDK. Τα αρχεία .jar έχουν προαιρετικά αρχεία manifest τα οποία αποφασίζουν πώς θα χρησιμοποιείται το αρχείο .jar. Εκείνα που στοχεύουν να εκτελούνται σαν ανεξάρτητες εφαρμογές, θα πρέπει να δηλώνουν σε αυτό το αρχείο την κεντρική κλάση τους, όπως στην συγκεκριμένη εφαρμογή εμείς δηλώσαμε την κλάση ObjCreatorV1.java. Συγκεκριμένα το αρχείο manifest.txt που βρίσκεται στον ίδιο φάκελο με τις κλάσεις περιέχει το εξής.

Main-Class: ObjCreatorV1

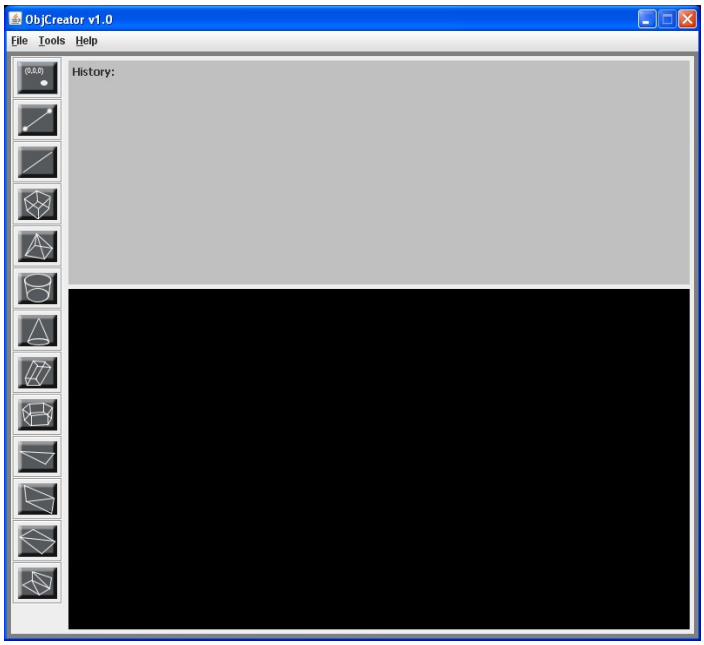
Για να τρέξουν και να συμπεριληφθούν όλες οι πηγές που χρησιμοποιήθηκαν σωστά, η εντολή που χρησιμοποιήθηκε για να δημιουργήσει το εκτελέσιμο αρχείο μας ήταν η παρακάτω.

*jar cvfm ObjCreator.jar manifest.txt *.class *.txt *.png java.jpg*

Όπου cvfm είναι εντολές που ζητούν την δημιουργία νέου αρχείου (c) με λεπτομερή έξοδο, οπου εμείς καθορίζουμε το όνομα του αρχείου και παρέχουμε πληροφορίες τύπου manifest απο ένα συγκεκριμένο αρχείο. ObjCreator.jar είναι το όνομα του αρχείου που θα δημιουργηθεί απο αυτήν την εντολή. Manifest.txt είναι το αρχείο που αναφέραμε και παραπάνω, ενώ τα υπόλοιπα συμβολίζουν όλα τα αρχεία που επιθυμούμε να προσθέσουμε στο .jar αρχείο μας. Δηλαδή όλα τα .class αρχεία, όλα τα .txt αρχεία, όλα τα .png αρχεία που είναι τα κουμπιά της εφαρμογής, και ένα αρχείο εικόνας.

ΚΕΦΑΛΑΙΟ 3 - Η ΕΦΑΡΜΟΓΗ OBJECT CREATOR v1.0

Πρόκειται για μία εφαρμογή που σκοπός της είναι να διευκολύνει τον χρήστη στη δημιουργία πολύπλοκων αρχείων σε μορφότυπο .obj, μία υλοποίηση που θα απαιτούσε περισσότερες γνώσεις, καθώς ο πιο συνηθισμένος τρόπος παραγωγής αρχείων obj είναι μέσω του λογισμικού 3D Studio Max. Το μόνο που χρειάζεται να γνωρίζει ο χρήστης, είναι η εγκατάσταση των αρχείων που χρειάζονται για να "τρέξει" μία εφαρμογή Java – Java3D στον υπολογιστή.
Ο χρήστης συναντώντας την εφαρμογή, προσανατολίζεται γρήγορα, συνηθίζει εύκολα τα σύμβολα και την ερμηνεία τους, καθώς η διεπαφή είναι απλή και χωρίς πολύπλοκα και περιττά κομμάτια.



Εικόνα 3.1 : στιγμιότυπο της διεπαφής της εφαρμογής

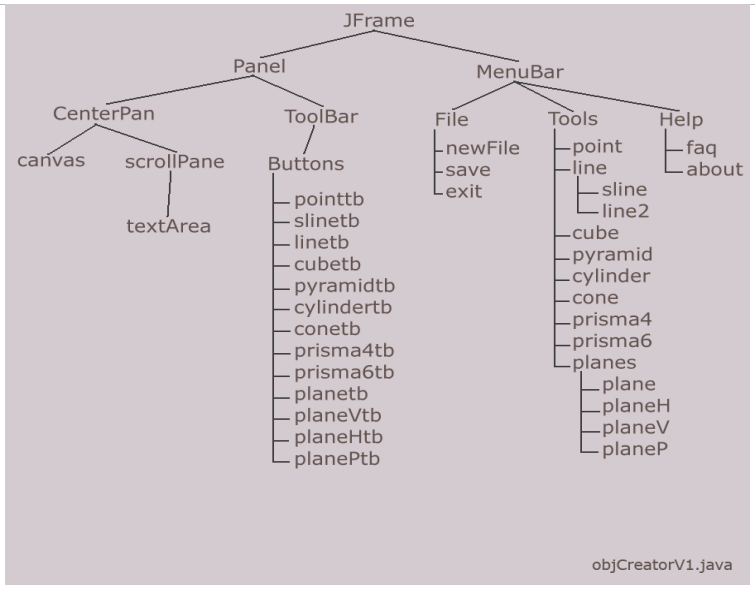
Ο χρήστης επιλέγει ένα ένα τα αντικείμενα απο το μενού στην δεξιά πλευρά της εφαρμογής. Πατώντας ένα κουμπί απο αυτά, μπορεί να ρυθμίσει το μέγεθος του αντικειμένου, και τις συντεταγμένες στις οποίες επιθυμεί να εμφανιστεί αυτό.

Υπάρχουν φυσικά, και προεπιλεγμένες ρυθμίσεις για κάθε αντικείμενο. Στη συνέχεια, το αντικείμενο αυτό εμφανίζεται με τις ρυθμίσεις που έχει επιλέξει ο χρήστης, στη σκηνή της εφαρμογής, στη δεξιά και κάτω πλευρά. Χρησιμοποιώντας το ποντίκι μπορεί τότε ο χρήστης να επιστρέψει το αντικείμενο, να πλησιάσει ή να απομακρυνθεί απο αυτό, ή να μετακινήσει το αντικείμενο προς όλες τις κατευθύνσεις. Μπορεί επίσης, να προσθέσει όποια αντικείμενα θέλει, απο όσες φορές επιθυμεί το καθένα. Στην δεξιά και πάνω πλευρά της διεπαφής μας, υπάρχει ένα πλαίσιο μέσα στο οποίο εμφανίζεται το ιστορικό του κάθε συνόλου αντικειμένων. Εκεί θα βλέπει ο χρήστης να καταγράφεται κάθε αντικείμενο που προστίθεται, καθώς και πιθανά προβλήματα που μπορεί να προκύψουν κατά τη διάρκεια της εκτέλεσης των εντολών. Όταν ο χρήστης ολοκληρώσει το αρχείο και είναι ικανοποιημένος απο το αποτέλεσμα, μπορεί να το αποθηκεύσει με τη μορφή obj όπου εκείνος επιθυμεί, ενώ στη συνέχεια μπορεί να ξεκινήσει ένα νέο “project” εάν το επιθυμεί.

Στη συνέχεια μπορούμε να αναλύσουμε κάθε μέρος της εφαρμογής ξεχωριστά, δίνοντας περισσότερες λεπτομέρειες στο προγραμματιστικό τμήμα της εφαρμογής. Θα ξεκινήσουμε φυσικά με το αρχείο που υποστηρίζει τη διεπαφή μας και καλεί όλα τα επιμέρους αρχεία.

3.1 ObjCreatorV1.java

Πρόκειται λοιπόν για το “κεντρικό” αρχείο, αυτό που υποστηρίζει το μεγαλύτερο τμήμα της διεπαφής, το κεντρικό παράθυρο δηλαδή, χρησιμοποιεί τις περισσότερες βιβλιοθήκες και κάνει τις περισσότερες διεργασίες. Ακολουθεί ένα διάγραμμα όπου παρουσιάζει τα τμήματα απο τα οποία αποτελείται.



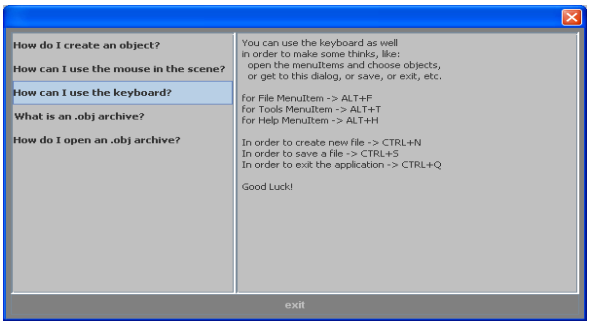
Σχεδιάγραμμα 3.1: τα στοιχεία απο τη βιβλιοθήκη swing της java που χρησιμοποιούνται στην εφαρμογή

Όπως βλέπουμε και στο σχήμα η διεπαφή χωρίζεται σε δύο μέρη. Το πάνω μέρος (MenuBar) και το κάτω μέρος (Panel). Τ

3.1.1 Το MenuBar, η μπάρα εργαλείων

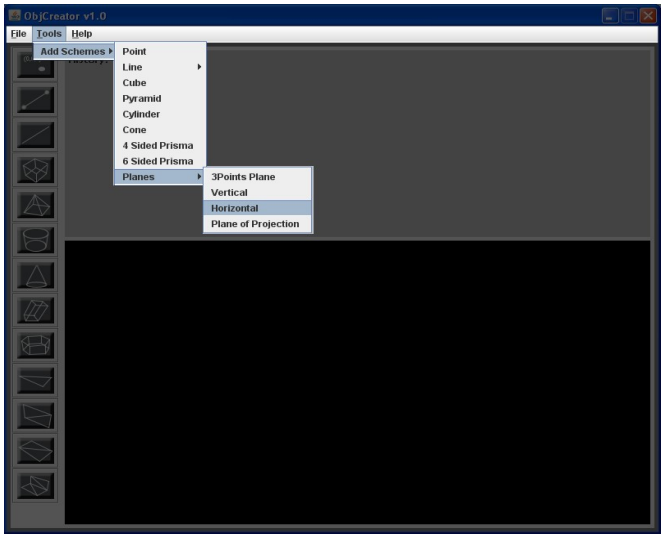
ο menubar περιέχει τα εξής όπως βλέπουμε και στο διάγραμμα: Στο Tools όλα τα κουμπιά που απαιτούνται για τη δημιουργία όλων των αντικειμένων, Στο File τις βασικές επιλογές λήξης ή έναρξης μίας νέας εργασίας, δηλαδή, new File – όπου αδειάζει τη σκηνή απο προηγούμενα αντικείμενα, και το ιστορικό απο τις προηγούμενες καταχωρήσεις αντικειμένων, καθώς και το προσωρινό αρχείο temp.obj στο οποίο αποθηκεύονται προσωρινά όλα τα αντικείμενα που δημιουργούμε. Το Save – που μας επιτρέπει να σώσουμε τ σύνολο των αντικειμένων που έχουμε φτιάξει ως τώρα. Στην πραγματικότητα αυτό το κουμπί καλεί το αρχείο Save.java το οποίο είναι ένας JFileChooser ο οποίος ζητάει απο τον χρήστη ένα όνομα για το αρχείο .obj στο οποίο θα αποθηκεύσει την εργασία – σύνολο αντικειμένων που έφτιαξε, καθώς και το που θέλει να αποθηκευτεί. Εσωτερικά ο JFileChooser δημιουργεί ένα αρχείο .obj, του δίνει το όνομα που επιθυμεί ο χρήστης, και αντιγράφει μέσα σε αυτό όλα όσα υπάρχουν εκείνη τη

στιγμή μέσα στο προσωρινό αρχείο temp.obj στο οποίο αναφερθήκαμε και νωρίτερα. Το Help περιέχει το πάντα χρήσιμο Frequently Asked Questions (FAQ) – Συχνές Ερωτήσεις χρηστών, το οποίο καλεί το αρχείο Faq.java και το About το οποί απλά αναφέρει τις εκδόσεις της java στις οποίες δούλεψα το πρόγραμμα.



Εικόνα 3.2: Στιγμιότυπο απο το παράθυρο βοήθειας της εφαρμογής.

Σε αυτό το σχήμα βλέπουμε ένα στιγμιότυπο του Faq παραθύρου μας, το οποίο εμφανίζεται όταν πατάμε το αντίστοιχο κουμπί απο το MenuBar. Το παράθυρο αυτό είναι πάντα εκεί απλά όταν ο χρήστης πατά το κουμπί exit το παράθυρο απλά κρύβεται. Είναι φτιαγμένο απο ένα JSplitPane το οποίο στην αριστερή πλευρά έχει ένα JList του οποίου τα στοιχεία όταν επιλέγουμε καλούνται αντίστοιχα κείμενα απο αρχεία .txt και εμφανίζονται στην δεξιά πλευρά του JSplitPane. Αυτό σημαίνει οτι για κάθε ερώτηση απο τη δεξιά πλευρά υπάρχει μία αντίστοιχη απάντηση σε ένα συγκεκριμένο αρχείο κειμένου. Στο επόμενο σχήμα βλέπουμε το MenuBar.



Εικόνα 3.3: Οι πιθανές επιλογές του Tools Button

3.1.2 Το κεντρικό πάνελ της εφαρμογής

Περνώντας στο κάτω μέρος της διεπαφής ή στο αριστερό “κλαδί” του δέντρου, υπάρχει το Panel, το οποίο περιλαμβάνει τα εξής δύο υποσυστατικά. Το centerPan και το ToolBar. Το centerPan περιλαμβάνει τα αντικείμενα στο κάτω και δεξιά μέρος. Τον καμβά – σκηνή και το textArea που παίζει το ρόλο του ιστορικού. Εκεί καταγράφονται όλα τα νέα αντικείμενα που προστίθενται στη σκηνή, καθώς και το μέγεθός τους και το σημείο στο οποίο τοποθετούνται. Στο History δεν υπάρχει λόγος να παραμείνουμε αφού πρόκειται ίσως για το πιο απλό τμήμα της διεπαφής. Είναι απλά ένα textArea στο οποίο προσθέτονται κομμάτια κειμένου - strings κάθε φορά που ένα αντικείμενο προστίθεται στη σκηνή. Θα μείνουμε όμως λίγο παραπάνω στη σκηνή που χρησιμοποιεί τις βιβλιοθήκες της Java3D.

Η σκηνή αυτή δημιουργείται με μια εντολή της μορφής

```
GraphicsConfiguration config =
SimpleUniverse.getPreferredConfiguration();
canvas = new Canvas3D(config);
```

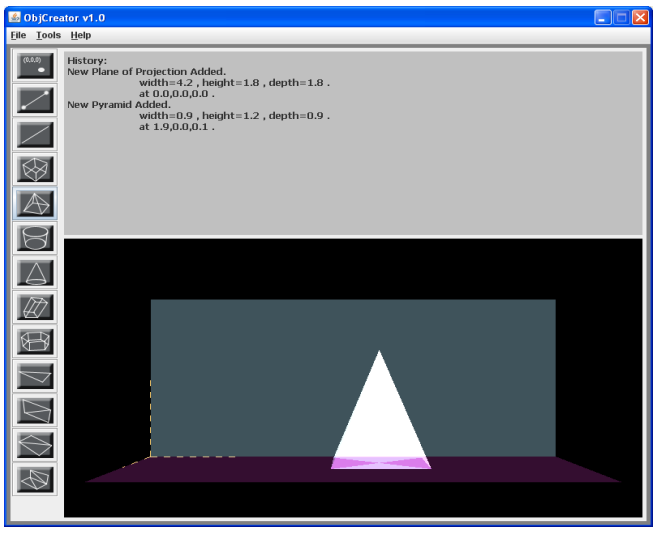
οπου η σκηνή στην πραγματικότητα είναι η το αντικείμενο canvas. Κάθε περιοχή οπου μπορούν να αναπαρασταθούν τρισδιάστατα γραφικά ονομάζεται canvas3d. Πρόκειται για ένα ορθογώνιο που περιέχει μία άποψη των αντικειμένων που βρίσκονται στο universe. Τοποθετούμε τον canvas μέσα σε ένα frame, και τότε δημιουργούμε ένα universe που θα εμφανιστεί μέσα στο canvas. Η σειρά λοιπόν με την οποία γράφουμε τις εντολές είναι κάπως έτσι :

```
SimpleUniverse universe = new SimpleUniverse();
BranchGroup group = new BranchGroup();
group.addChild(new ColorCube(0.3));
universe.getViewingPlatform().setNominalViewingTransform();
universe.addBranchGraph(group);
```

Δηλαδή δημιουργούμε ένα virtual universe για να βάλουμε το scene, έπειτα δημιουργούμε μια δομή δεδομένων για να συμπεριλάβουμε το σύνολο των αντικειμένων, και προσθέτουμε ένα αντικείμενο στο σύνολο. Εδώ βάλαμε έναν κύβο χρωμάτων με μήκος πλευράς 0.3. Τοποθετούμε τον viewer έτσι ώστε να κοιτά το αντικείμενο, και τέλος προσθέτουμε το σύνολο αντικειμένων στο universe. Το canvas3d εκμεταλλεύεται στο έπακρον τις δυνατότητες της κάρτας γραφικών του υπολογιστή μας για να αυξήσει την απόδοσή του. Αυτό δυστυχώς σημαίνει οτι δεν συνδυάζεται και πολύ καλά με τα εργαλεία διεπαφής χρήστη που περιέχει η swing της Sun. Αυτά τα συστατικά λέγονται LightWeight. Τα LightWeight συστατικά μπορεί να κρυφτούν απο έναν Canvas3D ακόμα και αν τα έχουμε τοποθετήσει μπροστά απο αυτόν. Στο συγκεκριμένο πρόγραμμα καταφέραμε να συνδυάσουμε LightWeight και HeavyWeight συστατικά στο ίδιο παράθυρο με το να τα κρατάω σε χωριστούς containers. Μία άλλη λύση θα ήταν να τοποθετήσουμε τον Canvas σε pop-up παράθυρο, που όμως δεν θα ήταν πολύ χρηστικό.

Γενικά οι τοποθεσίες περιγράφονται και εδώ χρησιμοποιώντας xyz συντεταγμένες. Όσο αυξάνονται οι συντεταγμένες του άξονα x΄x μετακινούμαστε προς τα δεξιά, όσο αυξάνονται οι συντεταγμένες του y'y μετακινούμαστε προς τα πάνω, ενώ όσο αυξάνονται οι συντεταγμένες του z'z μετακινούμαστε προς τα έξω απο την οθόνη. Όλες οι αποστάσεις μετριούνται σε μέτρα (m). Για να τοποθετήσουμε τα αντικείμενά μας στη σκηνή , ξεκινάμε απο το σημείο (0,0,0)

και έπειτα μετακινούμε τα αντικείμενά μας όπου επιθυμούμε. Η μετακίνηση των αντικειμένων ονομάζεται “Transformation”, γι' αυτό και οι κλάσεις που χρησιμοποιούμε είναι οι TransformGroup και Transform3D. Προσθέτουμε το αντικείμενο και την Transform3D σε ένα TransformGroup πριν προσθέσουμε το TransformGroup στη σκηνή. Αυτό γίνεται μέσω των επιλογών που εμφανίζονται στο παράθυρο πριν δημιουργηθεί και εμφανιστεί το αντικείμενό μας στη σκηνή. Πρέπει να πούμε επίσης οτι μπορεί και ο χρήστης να μετακινήσει το σύνολο των αντικειμένων με 3 επιπλέον κινήσεις που μπορεί να κάνει με τη βοήθεια του ποντικιού. Πριν όμως δούμε περισσότερα για αυτό ας δούμε και τον όρο appearance που επηρεάζει τον τρόπο εμφάνισης των αντικειμένων μας. Μπορεί να αλλάξει το χρώμα, την υφή, το ποσοστό αντανάκλασης του φωτός, ή μπορούμε να τους δώσουμε συγκεκριμένη ταπετσαρία. Ο πιο απλός τρόπος είναι να καθορίσουμε το χρώμα, τον τρόπο σκίασης ίσως και ποσοστό διαφάνειας. Έχοντας επεξεργαστεί την εμφάνιση των αντικειμένων, ας επιστρέψουμε στις επιπλέον κινήσεις που τροφοδοτεί το ποντίκι - πρέπει να σημειώσουμε με τις παρακάτω κινήσεις δεν μετακινούνται τα αντικείμενα αλλά η κάμερα, και μας δίνει την αίσθηση οτι κινούνται τα αντικείμενα. Οι πρόσθετες κινήσεις λοιπόν είναι οι εξής:

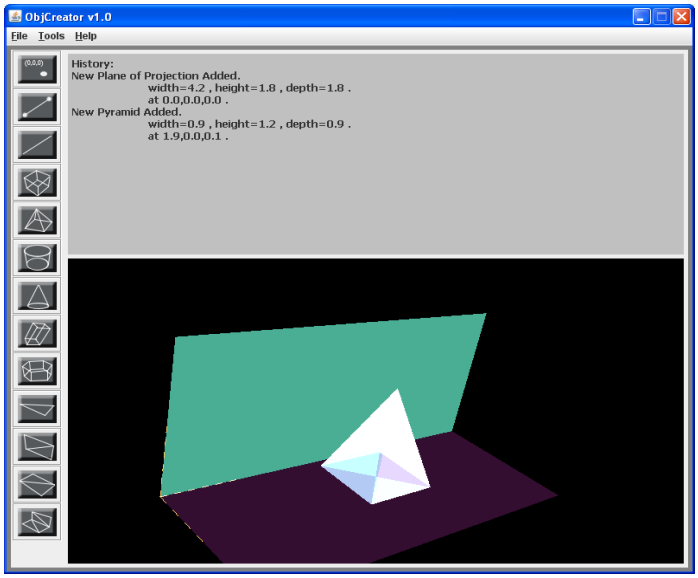


Εικόνα 3.4: Στιγμιότυπο απο μία εργασία της εφαρμογής

Ας υποθέσουμε οτι τα αντικείμενα που έχουμε εισάγει είναι το επίπεδο προβολής και μία πυραμίδα.

1. Rotation - Περιστροφή

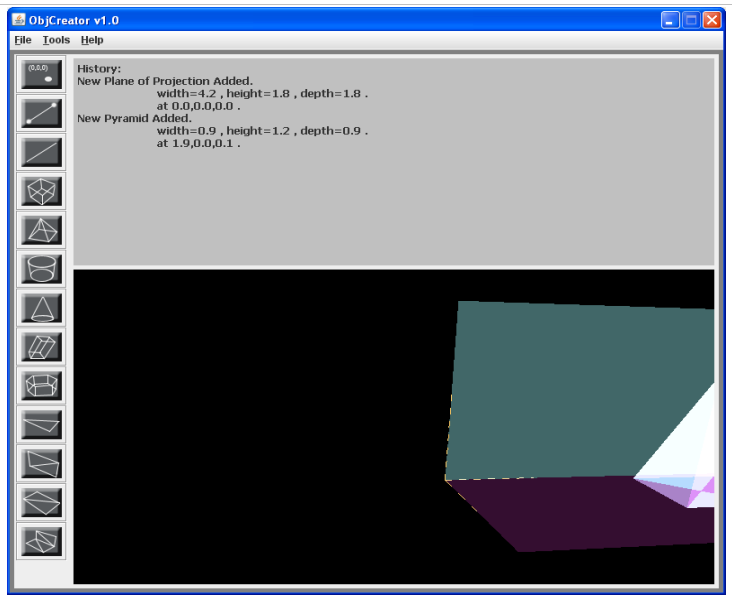
Πατώντας το αριστερό κουμπί στο ποντίκι και κρατώντας το πατημένο πάνω στη σκηνή ενώ ταυτόχρονα το μετακινούμε προς διάφορες κατευθύνσεις εκτελούνται κινήσεις περιστροφής της κάμερας. Ένα στιγμιότυπο είναι το παρακάτω. Αυτό πραγματοποιείται με τη χρήση της συνάρτησης MouseRotate().



Εικόνα 3.5: Στιγμιότυπο απο την εφαρμογή

2. Translation – Μετακίνηση

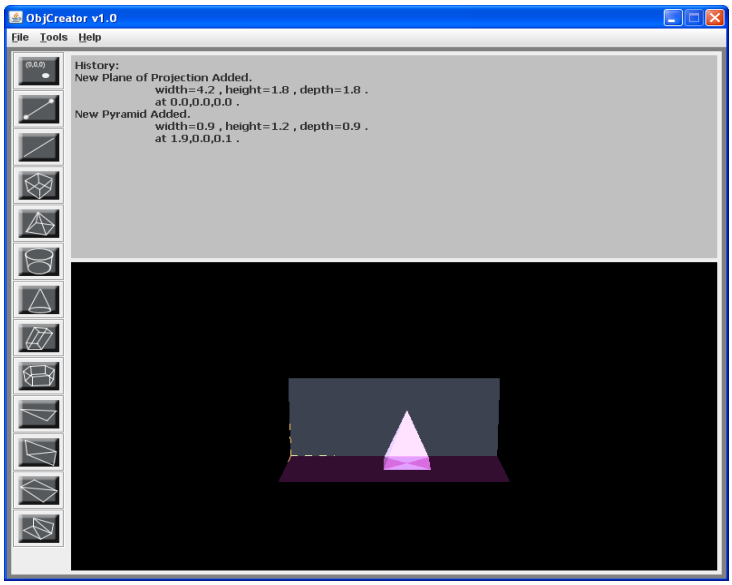
Πατώντας και κρατώντας πατημένο το δεξί κουμπί του ποντικιού, και ταυτόχρονα μετακινώντας το προς άλλες κατευθύνσεις, παρατηρούμε μετακίνηση του συνόλου των αντικειμένων. Ακολουθεί στιγμιότυπο απο ένα τέτοιο παράδειγμα. Αυτό πραγματοποιείται με τη χρήση της συνάρτησης MouseTranslate().



Εικόνα 3.6: Στιγμιότυπο απο την εφαρμογή

3. Zoom – Μεγέθυνση / Σμίκρυνση

Πατώντας και κρατώντας πατημένο το μεσαίο κουμπί του ποντικιού, ή τη ροδέλα για όσα ποντίκια έχουν, ενώ ταυτόχρονα μετακινώντας το προς διάφορες κατευθύνσεις θα δούμε το το σύνολο των αντικειμένων μας να μικραίνει ή να μεγαλώνει. Αυτό υλοποιείται με τη συνάρτηση MouseZoom(). Ένα στιγμιότυπο είναι το παρακάτω.

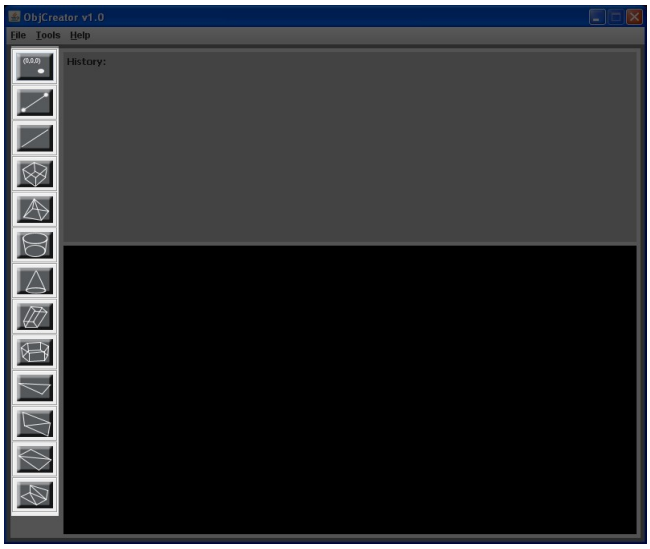


Εικόνα 3.7: Στιγμιότυπο απο μια εργασία της εφαρμογής.

Τελευταίο αλλά πολύ σημαντικό κομμάτι της διεπαφής της εφαρμογής μας είναι η μπάρα εργαλείων ή toolbar που στην πραγματικότητα με απλά λόγια είναι μία λίστα απο κουμπιά τα οποία είναι πιο εύκολα στη χρήση. Η μπάρα με τα κουμπιά βρίσκεται στη δεξιά πλευρά της διεπαφής, και καταλαμβάνει όλη την πλευρά απο πάνω ως κάτω. Περιλαμβάνει τα εξής αντικείμενα:

- σημείο
- ευθύγραμμο τμήμα ορισμένο απο δύο σημεία
- ημιευθεία ορισμένη απο ένα σημείο, κλίση σε σχέση με τον άξονα των χ' και κλίση σε σχέση με τον άξονα των y'.
- κύβος ορισμένος απο μία πλευρά
- πυραμίδα ορισμένη απο πλάτος, ύψος και βάθος
- κύλινδρος ορισμένος απο ακτίνα και ύψος
- κώνος ορισμένος απο ακτίνα και ύψος
- παραλληλεπίπεδο ορισμένο απο πλάτος, ύψος, βάθος και απόκλιση.
- οκτάεδρο πρίσμα ορισμένο απο πλάτος, ύψος και βάθος.
- επίπεδο ορισμένο απο τρία σημεία

- κάθετο επίπεδο ορισμένο απο πλάτος και ύψος
- οριζόντιο επίπεδο ορισμένο απο πλάτος και βάθος
- και πλάνο προβολής ορισμένο απο πλάτος, ύψος, βάθος.



Εικόνα 3.8: Η μπάρα εργαλείων της εφαρμογής.

Πατώντας σε κάποιο κουμπί του toolbar ο χρήστης ενεργοποιεί έναν action listener ο οποίος ξεκινά τη διαδικασία δημιουργίας κάποιου αντικειμένου. Αρχικά καλεί ένα νέο αρχείο Java να εκτελεστεί, το οποίο θα εμφανίσει στην οθόνη μας ένα παράθυρο το οποίο θα ρωτά τον χρήστη αν επιθυμεί να χρησιμοποιήσει τις default - προεπιλεγμένες τιμές που αφορούν το μέγεθος του αντικειμένου, και τις συντεταγμένες στις οποίες επιθυμεί να το τοποθετήσει. Οι παραπάνω μεταβλητές είναι global, άρα οι μεταβλητές αλλάζουν τιμή αυτόματα. Πατώντας ο χρήστης το κουμπί “ok”, καλεί ένα νέο αρχείο java, που είναι υπεύθυνο για τη δημιουργία του αντικειμένου και εμφάνισή του στη σκηνή. Αυτό το νέο αρχείο παίρνει τις τιμές των εκάστοτε μεταβλητών, και δημιουργεί κάθε φορά ξεχωριστό πίνακα τιμών, ανάλογα το σχήμα, ο οποίος περιέχει τις συντεταγμένες κάθε σημείου που θα χρησιμοποιήσει το νέο αντικείμενο.

Στη συνέχεια, αποθηκεύει στο τέλος ενός αρχείου κειμένου (temp.txt), τις

μεταβλητές τοποθετημένες με τέτοιο τρόπο ώστε να είναι συμβατές με την μορφή αρχείων .obj (θα εξηγήσουμε παρακάτω πως συντάσσονται αυτά τα αρχεία). Τοποθετήσαμε τις μεταβλητές του αντικειμένου αυτού στο τέλος του αρχείου διότι υποθετικά υπήρχε και άλλο αντικείμενο πριν απο αυτό, το οποίο θα θέλαμε να εμφανιστεί στην ίδια σκηνή. Είναι γενικά πολύπλοκος ο τρόπος που συντάχθηκε το αρχείο .obj, διότι οι συντεταγμένες πρέπει να εμφανίζονται σε συγκεκριμένη σειρά, έτσι ώστε τα αντικείμενα να εμφανίζονται σωστά. Κάπου εδώ δημιουργήθηκε ένα πρόβλημα με τις συντεταγμένες, καθώς στους αρχικούς πίνακες, και στην java3 χρησιμοποιούμε και αρνητικές τιμές, καθώς ναι μεν το σημείο (0,0,0) σημαίνει την αρχή του πλάνου μας, άλλα αυτό δεν μας εμποδίζει να μετακινήσουμε κάποιο αντικείμενο κάτω, πίσω ή πιο αριστερά απο το πλάνο προβολής. Δυστυχώς τα αρχεία .obj όπως θα δούμε και αργότερα δεν υποστηρίζουν αρνητικές τιμές στις συντεταγμένες. Θα δούμε γιατί. Το πρόβλημα λύθηκε μετακινώντας το κέντρο στο πραγματικό σημείο (50,50,50). Αυτό βέβαια δεν το καταλαβαίνει πουθενά ο χρήστης, ούτε φαίνεται κάπου, καθώς η αλλαγή των τιμών γίνεται λίγο πριν αποθηκευθούν στο αρχείο κειμένου.

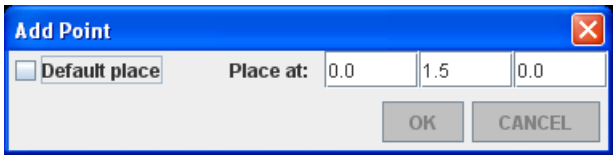
Μετά την ολοκλήρωση αποθήκευσης των συντεταγμένων, ολοκληρώνεται η κλήση αυτού του αρχείου, εκτυπώνεται στο history τμήμα της διεπαφής οτι δημιουργήθηκε το αντικείμενο με επιτυχία, και ολοκληρώνεται και η κλήση του πρώτου αρχείου. Τώρα επιστρέφουμε στο κεντρικό αρχείο java οπου δημιουργείται πια το αντικείμενο με τη βοήθεια της βιβλιοθήκης **j3d**. Ακολουθεί ένα παράδειγμα δημιουργίας αντικειμένων. Να σημειώσουμε εδώ οτι τα κουμπιά αυτά δημιουργήθηκαν ειδικά για αυτήν την εφαρμογή με τη βοήθεια του προγράμματος Adobe Photoshop CS2, έτσι ώστε ο χρήστης να αντιλαμβάνεται πιο εύκολα για ποιο σχήμα πρόκειται.

3.1.3 <u>Παράδειγμα:</u>

Δημιουργία ενός σημείου.

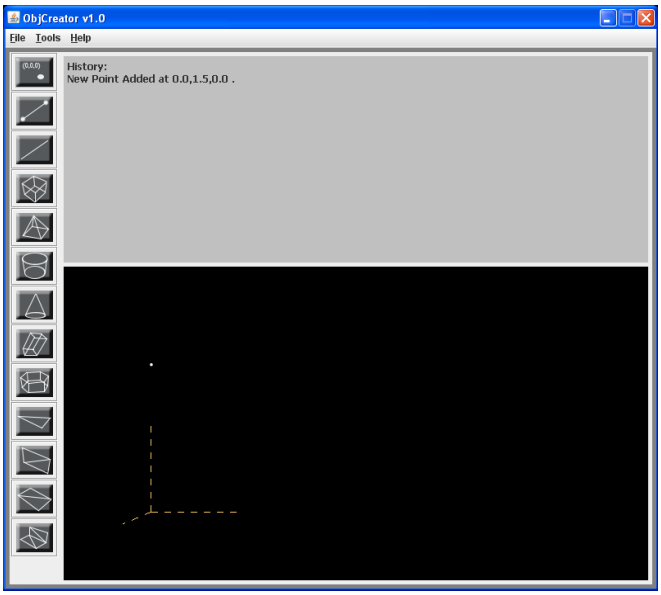
Κάνοντας κλικ στο κουμπί με το σημείο που υπάρχει στο toolbar, εμφανίζεται το

παρακάτω παράθυρο που μας δίνει τη δυνατότητα να τοποθετήσουμε το σημείο όπου επιθυμούμε, η και να το αφήσουμε στην προεπιλογή.



Εικόνα 3.9: Το παράθυρο Add Point

Κάνοντας κλικ στο “ok:” στη συνέχεια, οι τιμές που δώσαμε αποθηκεύονται, ή μένουν ίδιες αν δεν τις αλλάξαμε, το παράθυρο που βλέπουμε κρύβεται, και καλείται το επόμενο αρχείο java που ο χρήστης δεν αντιλαμβάνεται (Point.java). Αυτο το αρχείο είναι φυσικά διαφορετικό για κάθε αντικείμενο και εκτελεί ίσως μία απο τις πιο σημαντικές λειτουργίες του προγράμματος. Παίρνει τις τιμές που χρειάζεται το κάθε αντικείμενο για να δημιουργηθεί κάνει τους αντίστοιχους πίνακες όποτε αυτό χρειάζεται και αποθηκεύει το νέο αντικείμενο στο τελικό αρχείο με κατάληξη obj. Θα αναφερθούμε στα αρχεία αυτά πιο λεπτομερώς σε άλλο κεφάλαιο. Όταν ολοκληρωθεί η αποθήκευση του αντικειμένου στο αρχείο, επιστρέφουμε στο κεντρικό μας παράθυρο πάλι, όπου το κουμπάκι που είχαμε πατήσει στο toolbar έχει ξεκινήσει μία νέα διαδικασία εμφάνισης του νέου μας αντικειμένου στη σκηνή, στον καμβά. Όσο προσθέτουμε αντικείμενα στο αρχείο μας, θα τα βλέπουμε να προστίθενται και στη σκηνή. Μάλιστα πρόκειται για αναπαράσταση όσων υπάρχουν και στο αρχείο μας. Δηλαδή αν αλλάξουμε μέγεθος σε κάποιο αντικείμενο, ή αν αλλάξουμε το σημείο οπου θα εμφανιστεί, στη σκηνή θα δούμε αυτό που ζητήσαμε εμείς να γίνει και στο αρχείο. Ταυτόχρονα στο χώρο πάνω απο τη σκηνή βλέπουμε το ιστορικό. Τί έχουμε προσθέσει στη σκηνή, σε ποιο σημείο με ποιο μέγεθος κτλ.

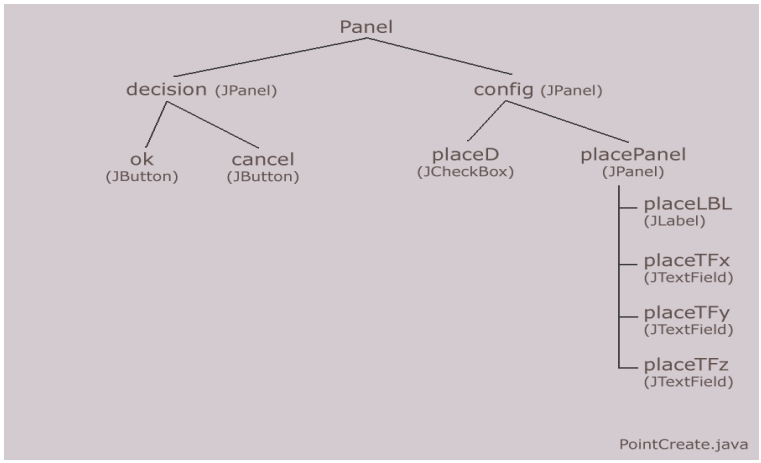


Εικόνα 3.10: Στιγμιότυπο απο μια εργασία στην εφαρμογή.

3.2 ΔΗΜΙΟΥΡΓΙΑ ΣΗΜΕΙΟΥ

Κάθε φορά που πατάμε το πρώτο κουμπί στην μπάρα εργαλείων ή όποτε πατάμε την επιλογή Tools → Add Schemes → Point καλούνται τα αρχεία PointCreate.java και Point.java. Τότε, εμφανίζεται ένα νέο παράθυρο που μέχρι εκείνη τη στιγμή ήταν κρυμμένο και ονομάζεται Add Point (βλέπε εικόνα 3.9). Εδώ μας δίνεται η επιλογή των ρυθμίσεων του αντικειμένου.

Στο συγκεκριμένο παράθυρο μπορούμε απλά να επιλέξουμε το που θα εμφανιστεί. Στο παρακάτω σχήμα βλέπουμε τη μορφή με την οποία είναι δομημένο το παράθυρο.

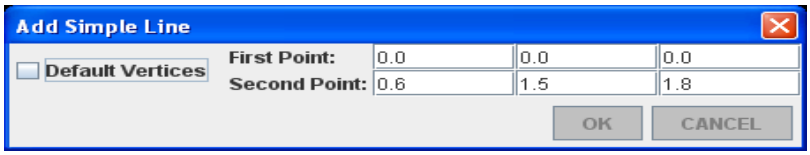


Σχεδιάγραμμα 3.2: τα στοιχεια του παραθύρου προσθήκης του σημείου

Όπως βλέπουμε είναι όντως πολύ απλό. Αποτελείται απο μερικά JTextFields που αποθηκεύουν τις νέες τιμές των συντεταγμένων. Αν ο χρήστης επιλέξει το κουμπί ακύρωσης, τότε δεν πραγματοποιείται καμία εργασία. Το παράθυρο ξανακρύβεται χωρίς να εκτελέσει τίποτα. Αν ο χρήστης επιλέξει το ok τότε καλείται το επόμενο αρχείο που ονομάζεται Point.java. Εκεί δημιουργούνται οι πίνακες που θα αποθηκευθούν στο αρχείο .obj οι οποίοι όπως είπαμε προηγουμένως παίρνουν τις τιμές τους απο global τιμές, δηλαδή οι τιμές αυτών των μεταβλητών φαίνονται και αλλάζουν απο κάθε αρχείο. Αφού αποθηκευθούν τα απαραίτητα στοιχεία στο αρχείο με φορμάτ .obj τότε το αρχείο αυτό κλείνει και γυρίζουμε πίσω στο κεντρικό μας αρχείο ObjCreatorV1.java οπου εκεί το πρόγραμμα εκτυπώνει στο πεδίο History τα απαραίτητα στοιχεία (ποιο αντικείμενο προστέθηκε, σε ποιες συντεταγμένες και σε τί μέγεθος), ενώ στη συνέχεια σχεδιάζει το αντικείμενο αυτό στον καμβά. (Για να είμαστε ακριβείς, το πρώτο όπως και όλα τα επόμενα αντικείμενα που προστίθενται στο εκάστοτε project σχεδιάζουν επίσης και την αρχή των αξόνων. Δηλαδή τρεις γραμμούλες που αντιπροσωπεύουν τους 3 άξονες, άρα είναι και κάθετες μεταξύ τους. Αυτό βέβαια δεν προστίθεται στο τελικό .obj αρχείο.)

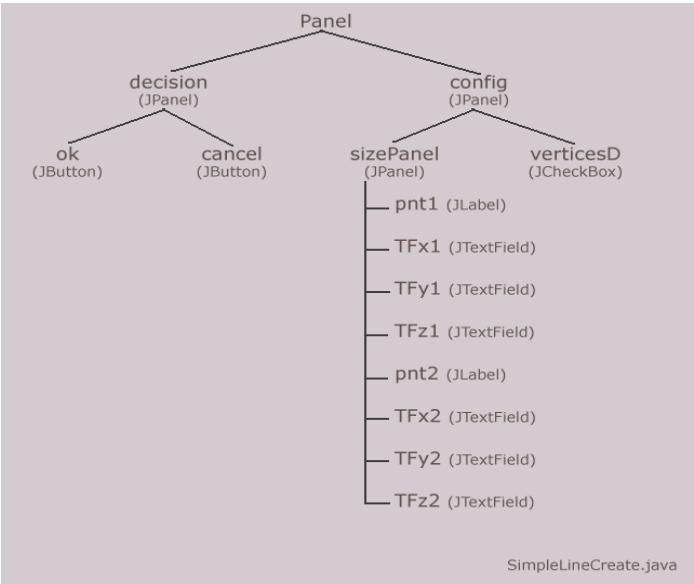
3.3 ΔΗΜΙΟΥΡΓΙΑ ΓΡΑΜΜΗΣ ΟΡΙΣΜΕΝΗΣ ΑΠΟ ΔΥΟ ΣΗΜΕΙΑ

Το αρχείο SimpleLineCreate.java ξεκινά τη λειτουργία του μόλις ο χρήστης πατήσει το δεύτερο κουμπί στη λίστα, που δημιουργεί μία απλή γραμμή η οποία χαρακτηρίζεται απο δύο σημεία στο χώρο. Τα δύο άκρα της. Ο δεύτερος τρόπος να επιλέξουμε την απλή γραμμή είναι η εξής: Tools → Add Schemes →Line → Simple Line.



Εικόνα 3.11: Το παράθυρο για την προσθήκη της απλής γραμμής.

Όπως θα δείτε και στο σχεδιάγραμμα παρακάτω, ούτε και αυτό το αρχείο περιέχει κάτι πολύπλοκο προγραμματιστικά.



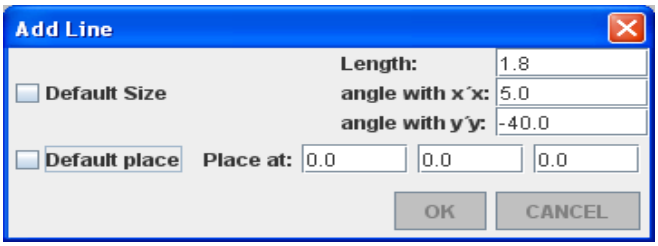
Σχεδιάγραμμα 3.3: τα στοιχεία που χρησιμοποιήθηκαν για το παράθυρο δημιουργίας της απλής γραμμής

Περιέχει έξι JtextFileds τα οποία παίρνουν τις τιμές των δύο σημείων όταν αυτά

είναι άλλα απο τα προεπιλεγμένα. Εδώ να πούμε οτι σε όλα τα αντίστοιχα παράθυρα υπάρχει ένα JCheckBox, στο συγκεκριμένο αρχείο έχει το όνομα verticesD, και αυτό γιατί ελέγχει το αν θέλουμε οι συντεταγμένες (vertices) να είναι οι προεπιλεγμένες (default) ή άλλες της επιλογής μας. Όταν είναι τσεκαρισμένο αυτό το JCheckBox όλο το sizePanel είναι κρυμμένο, αφού δεν ενδιαφέρεται ο χρήστης να το αλλάξει. Όταν είναι μή τσεκαρισμένο μπορεί ο χρήστης να δεί όλα τα πεδία, JLabel και JTextField, με τις προεπιλεγμένες τιμές τις οποίες μπορεί να αλλάξει. Αν τελικά επιλέξει ο χρήστης να πατήσει το άκυρο (cancel), πριν κρυφτεί το παράθυρο αυτό, γίνεται μια μεταβλητή false – ψευδής, έτσι ώστε γυρνώντας στο κεντρικό πρόγραμμα να μην εκτελεστεί καμία εντολή σχεδίασης. Αν ο χρήστης πατήσει το ok, τότε εκείνη η ίδια μεταβλητή γίνεται true – αληθής, και καλείται το αρχείο SimpleLine.java το οποίο αναλαμβάνει να δημιουργήσει τον πίνακα με τα απαραίτητα στοιχεία, και να τον εκτυπώσει συντακτικά σωστά στο τέλος του αρχείου temp.obj. Αφού ολοκληρωθεί και αυτή η διαδικασία κλείνει το αρχείο αυτό και επιστρέφουμε στο αρχικό αρχείο οπου τώρα σχεδιάζεται η γραμμή στη σκηνή. Αφού σχεδιαστεί το πρόγραμμα επιστρέφει σε κατάσταση αναμονής μέχρι ο χρήστης να πατήσει το επόμενο κουμπί.

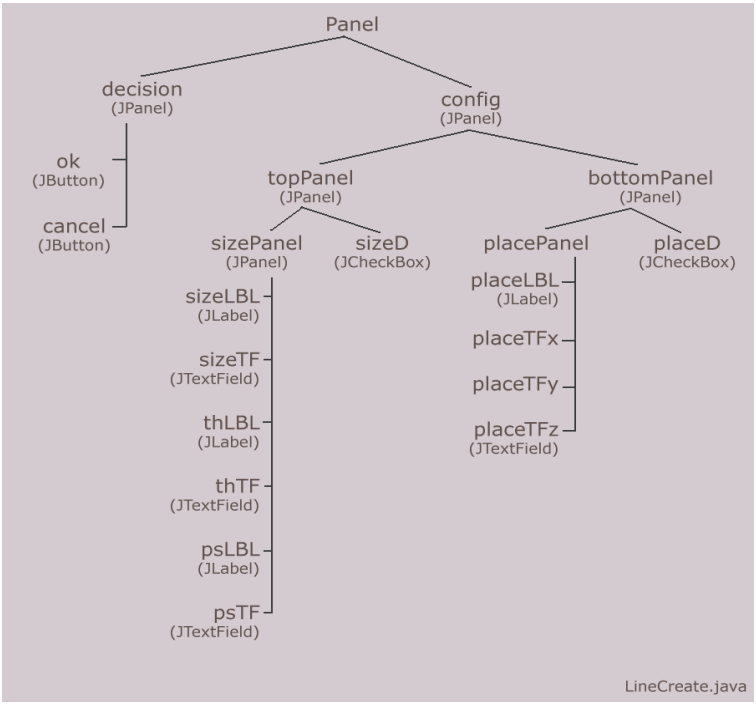
3.4 ΔΗΜΙΟΥΡΓΙΑ ΓΡΑΜΜΗΣ ΟΡΙΣΜΕΝΗΣ ΑΠΟ ΜΗΚΟΣ ΚΑΙ ΔΥΟ ΓΩΝΙΕΣ

Το αρχείο LineCreate.java καλείται όποτε ο χρήστης πατήσει το τρίτο απο πάνω κουμπί στη μπάρα εργαλείων, ή όποτε μεταβεί στο Tools → Add Schemes →Line → Line Plus. Το κουμπί αυτό δημιουργεί τελικά μία γραμμή (στην πραγματικότητα ένα ευθύγραμμο τμήμα αν χρησιμοποιήσουμε τους μαθηματικούς όρους) , η οποία χαρακτηρίζεται απο τρεις τιμές που ορίζουν το μήκος της και την κατεύθυνση της, καθώς και τρεις ακόμη τιμές που ορίζουν το σημείο στο οποίο ακουμπά το ένα της άκρο. Στην προεπιλογή αυτό είναι η αρχή των αξόνων.



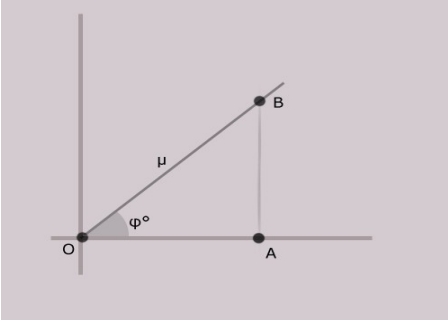
Εικόνα 3.12: Το παράθυρο για την προσθήκη της γραμμής δεδομένου του μήκους και δύο γωνιών.

Σχετικά με το μήκος της γραμμής δεν υπάρχει κάτι το οποίο αξίζει αναφοράς. Ίσως όμως πρέπει να αναφερθούμε στο πώς καθορίζουμε την κατεύθυνση της ευθείας με τις άλλες δυο τιμές. Όπως φαίνεται λοιπόν στο παραπάνω παράθυρο η μία τιμή είναι η γωνία που θέλουμε να σχηματίζει η ευθεία μας με τον άξονα x'x, και η δεύτερη η γωνία που θέλουμε να σχηματίζει η ευθεία μας με τον άξονα y'y, και οι δύο τιμές σε μοίρες (°).



Σχεδιάγραμμα 3.4: τα στοιχεία της swing που χρησιμοποιήθηκαν για το παράθυρο δημιουργίας της γραμμής που ορίζεται απο ένα μήκος και δύο γωνίες.

Ας ξεκινήσουμε με τη μαθηματική θεωρία σχετικά. Κοιτάζοντας το παρακάτω σχέδιο (σχεδιάγραμμα 3.5) μπορούμε να συμπεράνουμε τα εξής:



Σχεδιάγραμμα 3.5

$$OB^2=OA^2+AB^2$$

(Πυθαγόρειο θεώρημα)

$$\sin\varphi^\circ = AB / \mu$$

$$\cos\varphi^\circ = OA / \mu$$

Τώρα, ας ονομάσουμε θ° την γωνία που χαρακτηρίζει την ευθεία μας με τον άξονα των x'x, και την γωνία που χαρακτηρίζει την ευθεία με τον άξονα των y'y ας την ονομάσουμε ψ°. Εύκολα μπορούμε να δούμε οτι στην περίπτωση της γωνίας θ° μας κάνουν και οι δύο προηγούμενες σχέσεις. Αφού γνωρίζουμε την γωνία και το μήκος (μ). Άρα κρατάμε όποιο απο τα δύο θέλουμε.

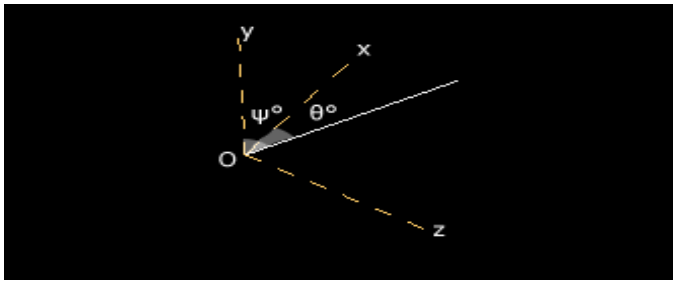
$$\cos\theta^\circ = OA / \mu \rightarrow \mathbf{OA = \mu * \cos\theta^\circ}$$

$$\sin\theta^\circ = AB / \mu \rightarrow \mathbf{AB = \mu * \sin\theta^\circ}$$

Άρα αφου μπορούμε να βρούμε το ΟΑ δηλαδή τη συντεταγμένη του άξονα x'x και το ΑΒ που είναι η συντεταγμένη y, αρκεί να βρούμε άλλη μία πλευρά, συγκεκριμένα αρκεί να βρούμε τη συντεταγμένη z, για να έχουμε και τις 3 συντεταγμένες του δεύτερου σημείου. Για να λύσουμε τη γωνία ψ°, αρκεί να φανταστούμε ένα αντίστοιχο τρίγωνο ΟΒΓ. Όπου ΟΒ = μ, και $\mu^2=O\Gamma^2+B\Gamma^2$. Όπου ΟΓ κομμάτι του άξονα y'y, και ΒΓ η τρίτη πλευρά. Εδώ θα ισχύει:

$$\cos\psi^\circ = O\Gamma / \mu \rightarrow \mathbf{O\Gamma = \mu * \cos\psi^\circ}$$

sin ψ° = ΒΓ / μ



Σχεδιάγραμμα 3.6

Στη συγκεκριμένη εφαρμογή επιλέξαμε να χρησιμοποιήσουμε τις έντονα γραμμένες σχέσεις. Τώρα λοιπόν, έχοντας τα παραπάνω βλέπουμε οτι έχουμε τα ΟΑ, ΑΒ, ΟΓ . Το οποίο σημαίνει οτι αν κρατήσουμε για πρώτο σημείο της ευθείας το κέντρο των αξόνων (0,0,0) και σαν δεύτερο σημείο το (ΟΑ, ΑΒ, ΟΓ) , και αν τα ενώσουμε αυτά τα δύο, θα έχουμε την ευθεία που ψάχνουμε. Συγκεκριμένα το αντίστοιχο κομμάτι του προγράμματος είναι κάπως έτσι:

```
...sidex = (length * Math.cos(th));
sidey = (length * Math.sin(th));
sidez = (length * Math.cos(ps));
double[] vertstable = { 0.0+atX, 0.0+atY, 0.0+atZ,
                        sidex+atX, -sidey+atY, -sidez+atZ};...
```

Είμαστε ακόμα βέβαια στο αρχείο LineCreate.java, το οποίο είναι απλά υπεύθυνο να αποθηκεύσει τις τιμές των μεταβλητών. Με την εξής εντολή διαβάζονται οι καινούριες τιμές που δίνονται στα πεδία κειμένου.

```
Global.linelength = Double.parseDouble(sizeTF.getText());
```

Η συγκεκριμένη μεταβλητή (*linelength*) είναι το μήκος της γραμμής αυτής. Είναι Global γιατί έχουμε κρατήσει ένα αρχείο με όλες τις μεταβλητές όλων των αντικειμένων σε ένα αρχείο που ονομάζεται Global.java. Έτσι έχουμε επιτρέψει στο πρόγραμμα να έχει πρόσβαση σε όλες τις μεταβλητές που χρειάζεται. Δηλαδή τα μεγέθη, τις τοποθεσίες που θα εμφανίζονται τα αντικείμενα, καθώς και εκείνες τις Boolean μεταβλητές που γίνονται προσωρινά true ή false ανάλογα με το αν πραγματοποιείται ένα αντικείμενο ή οχι. Άρα, Αφου διαβάσει

και αποθηκεύσει τις νέες τιμές των μεταβλητών, αν βέβαια ο χρήστης πατήσει το ok όπως έχουμε ξαναπεί, τότε μεταφερόμαστε στο αρχείο Line.java όπου δημιουργούνται οι απαραίτητοι πίνακες και εκτυπώνονται τα απαραίτητα στοιχεία στο αρχείο temp.obj. Επιστρέφοντας στο κεντρικό παράθυρο, βλέπουμε να εμφανίζεται στη σκηνή και η νέα ευθεία, καθώς και στο πεδίο History τα στοιχεία της ευθείας που προστέθηκε. Στη συνέχεια το πρόγραμμα επιστρέφει σε κατάσταση αναμονής.

3.5 ΔΗΜΙΟΥΡΓΙΑ ΚΥΒΟΥ

Το CubeCreate.java που ονομάζεται έτσι ασχολείται όπως φαίνεται με τη δημιουργία ενός κύβου. Εμφανίζεται όταν πατάμε το τέταρτο κουμπάκι ή μεταβαίνοντας στο Tools → Add Schemes → Cube.

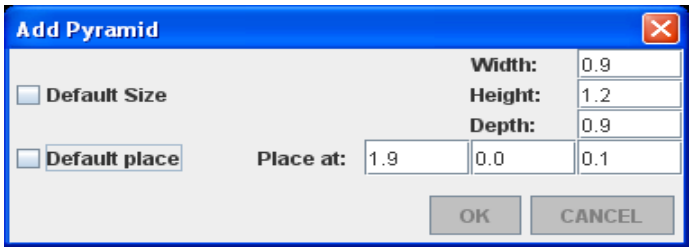
Εικόνα 3.13: Το παράθυρο προσθήκης του κύβου.

Εδώ οι μεταβλητές που χρειάζονται είναι τέσσερις. Το μήκος της πλευράς του κύβου, και οι συντεταγμένες του σημείου στο οποίο θα εμφανιστεί. Το σημείο στο οποίο θα επιλέξουμε να εμφανιστεί, θα εφάπτεται στην αριστερά πίσω και κάτω γωνία του κύβου. Όπως και στις προηγούμενες περιπτώσεις, αφου ο χρήστης πατήσει ok, μεταφερόμαστε στο αρχείο Cube.java, το οποίο θα τοποθετήσει τον κύβο στο αρχείο temp.obj. Αφού επιστρέφουμε στο κεντρικό αρχείο βλέπουμε οτι ο κύβος έχει ήδη σχεδιαστεί στη σκηνή. Αυτο γίνεται με τη χρήση της συνάρτησης **public Box(float xdim, float ydim, float zdim, Appearance ap)** οπου xdim, ydim, zdim οι διαστάσεις του κουτιού. Για μας είναι

και τα τρία ίδια, εφόσον θέλουμε έναν κύβο. Appearance είδαμε νωρίτερα, είναι μια συνάρτηση που επηρεάζει την εξωτερική εμφάνιση των αντικειμένων μας. Αφού ολοκληρώνεται ο σχεδιασμός του κύβου στη σκηνή και προστεθούν τα στοιχεία του κύβου στο πεδίο History, το πρόγραμμα επιστρέφει σε κατάσταση αναμονής μέχρι να τεθεί σε λειτουργία το επόμενο αντικείμενο.

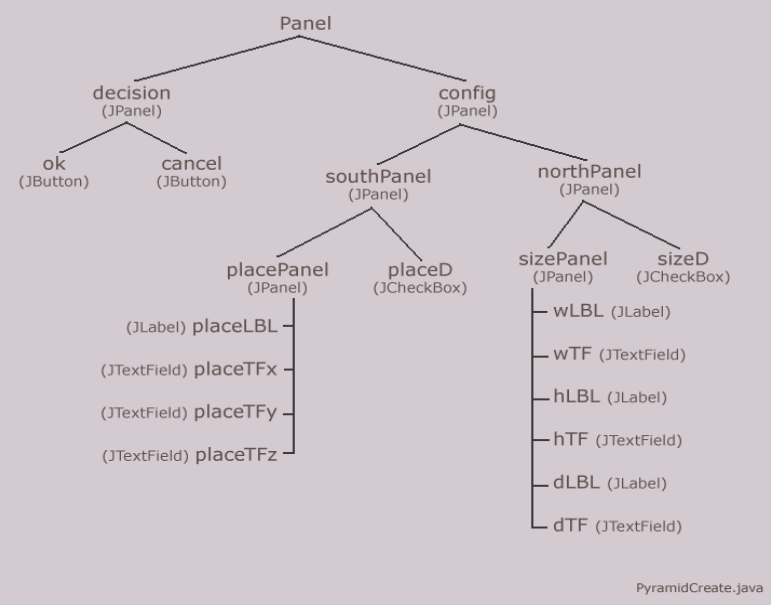
3.6 ΔΗΜΙΟΥΡΓΙΑ ΠΥΡΑΜΙΔΑΣ

Πρόκειται για το αρχείο PyramidCreate.java που δημιουργεί μια πυραμίδα. Ο χρήστης ανοίγει το παρακάτω παράθυρο πατώντας το πέμπτο κουμπί απο πάνω προς τα κάτω στην μπάρα εργαλείων που έχει σαν εικόνα μία αναπαράσταση της πυραμίδας, ή πηγαίνοντας μέσω του MenyBar Tools → Add Schemes → Pyramid.



Εικόνα 3.14: Το παράθυρο προσθήκης της πυραμίδας.

Η βάση της πυραμίδας είναι παραλληλόγραμμο, οπότε για να το σχεδιάσουμε χρειαζόμαστε σαν δεδομένα το πλάτος και το βάθος, ενώ άλλο δεδομένο είναι φυσικά το ύψος της πυραμίδας, και τέλος το σημείο στο οποίο θα εμφανιστεί. Αυτό το σημείο θα είναι η γωνία κάτω μέσα αριστερά και πάλι όπως και στον κύβο.



Σχεδιάγραμμα 3.7: τα στοιχεία της swing που συντάσσουν το παράθυρο προσθήκης πυραμίδας

Αφού αποθηκευθούν οι νέες τιμές των μεταβλητών τότε ξεκινά να διαβάζεται το αρχείο Pyramid.java που αναλαμβάνει να μεταφέρει τις νέες τιμές στο αρχείο temp.obj σε τέτοια μορφή ώστε να σχηματίζουν μία πυραμίδα. Αφού υλοποιηθεί και αυτό, γυρνάμε στο κεντρικό παράθυρο οπου θα έχει ήδη προστεθεί η πυραμίδα στον καμβά, στο σημείο (x, y, z) που ζητήσαμε. Η συνάρτηση την οποία χρησιμοποιήσαμε για να μετατρέψουμε έναν πίνακα με αριθμούς σε πυραμίδα είναι η εξής: public **TriangleStripArray** (int vertexCount, int vertexFormat, int stripVertexCounts). Όπου vertexCount το πλήθος των στοιχείων, vertexFormat είναι μία μάσκα η οποία προσδιορίζει τις συνιστώσες που περιέχονται σε κάθε σημείο. Ο stripVertexCounts είναι ένας πίνακας που καθορίζει τον αριθμό του πλήθους των συντεταγμένων για κάθε ξεχωριστή λωρίδα. Το μήκος αυτού του πίνακα ορίζει το πλήθος των ταινιών. Το άθροισμα των στοιχείων αυτού του πίνακα καθορίζει το συνολικό αριθμό των συντεταγμένων που χρησιμοποιούνται. Αμέσως μετά εκτελείται η *pyramid1.setCoordinates(0, facesPyr);* η οποία δίνει τις συντεταγμένες σε

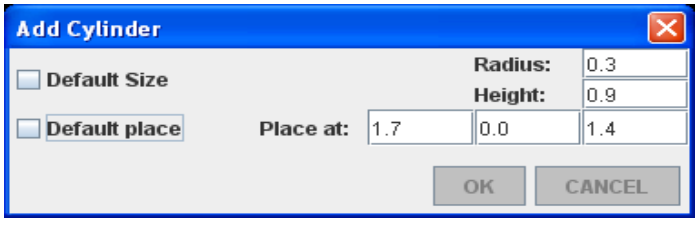
πίνακα, με συγκεκριμένη σειρά όμως, τέτοια ώστε να εμφανιστούν σωστά οι πλευρές της πυραμίδας. Στη συνέχεια προσθέτουμε στο αντικείμενο την εμφάνιση (appearance), η οποία στη συγκεκριμένη εφαρμογή έχουμε επιλέξει να έχει την ίδια εμφάνιση με όλα τα άλλα αντικείμενα. Επόμενη εντολή είναι η TransformGroup η οποία τοποθετεί το αντικείμενο στο σημείο που επέλεξε ο χρήστης.

TransformGroup pyramidtg = createTG(Global.pyramidX/3, Global.pyramidY/3, Global.pyramidZ/3);

Όπως βλέπουμε όλες οι συντεταγμένες έχουν διαιρεθεί με το τρία. Αυτό έχει γίνει σε όλα τα αντικείμενα. Ο λόγος είναι οτι οι τιμές που έχουν δοθεί είναι κατάλληλες για τα εξερχόμενα αρχεία μας, αλλά είναι πολύ μεγάλες και δημιουργούν μεγάλες αποστάσεις όταν χρησιμοποιούνται στην java3D. Εφόσον πρόκειται για αναπαράσταση αυτό δεν είναι πρόβλημα, διότι είναι μεν ακριβής, τα αντικείμενα δεν έχουν διαφορετική θέση απο οτι στο εξερχόμενο αρχείο. Απλά έχουν άλλες τιμές οι συντεταγμένες. Μια λεπτομέρεια που σε αυτήν την εφαρμογή δεν επηρεάζει και ούτε ενδιαφέρει το χρήστη. Το επόμενο αντικείμενο στη λίστα είναι ο κύλινδρος.

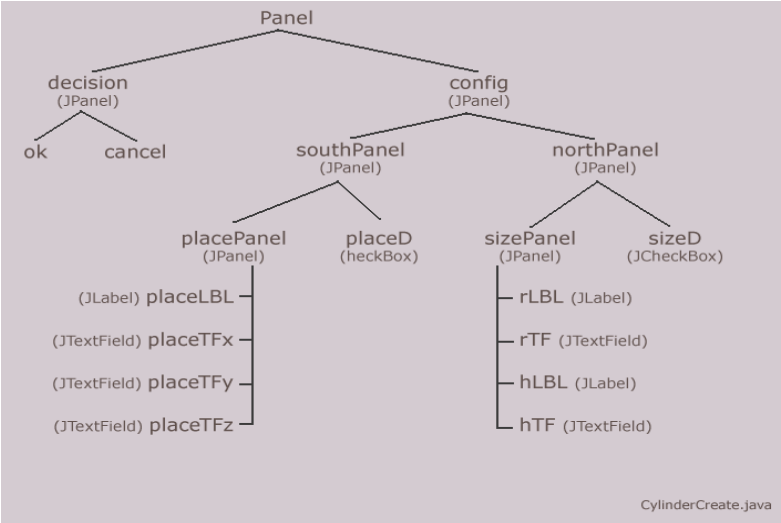
3.7 ΔΗΜΙΟΥΡΓΙΑ ΚΥΛΙΝΔΡΟΥ

Πρόκειται για έναν κλασικό κύλινδρο λοιπόν, με μεταβλητές τιμές το ύψος και την ακτίνα του, καθώς και το σημείο στο οποίο θα εμφανιστεί. Το παράθυρο CylinderCreate.java με τις επιλογές αυτές εμφανίζεται πατώντας στο αντίστοιχο κουμπάκι με τον κύλινδρο που βρίσκεται στην μπάρα εργαλείων ή μεταβαίνοντας στο Tools → Add Schemes → Cylinder και ονομάζεται add Cylinder.



Εικόνα 3.15: Το παράθυρο προσθήκης του κυλίνδρου

Οι συντεταγμένες του σημείου που δίνουμε για να τοποθετηθεί το αντικείμενο αναφέρονται στο κέντρο της βάσης του κυλίνδρου.



Σχεδιάγραμμα 3.8: τα στοιχεία swing που περιέχονται στο παράθυρο προσθήκης κυλίνδρου

Απο τη στιγμή που αυτό το αρχείο παίρνει τις απαραίτητες τιμές και τις αποθηκεύει ξεκινά να εκτελείται το επόμενο αρχείο για να συνεχίσει την εργασία, το αρχείο Cylinder.java θα αναλάβει να μας βρει όλα τα σημεία που χρειαζόμαστε για να φτιαχτεί ο κύλινδρος. Θα ήταν αρκετά κουραστικό να φτιάξουμε και να υπολογίσουμε στο χέρι και τα 1154 (!) σημεία που χρειάζονται για να σχηματιστεί. Άρα χρησιμοποιώντας τη συνάρτηση *Cylinder cylinder1 = new **Cylinder**(radius, height, primflags, 35, 15, app)* που είναι μία συνάρτηση της Java3D, δημιουργήσαμε έναν κύλινδρο του οποίου τα σημεία δεν τα γνωρίζουμε ακόμη μεν, αλλά μπορούμε να τα μάθουμε μέσω της εντολής *int verts = cylinder1.getNumVertices();* που μας δίνει το πλήθος τους αρχικά , και στη συνέχεια ψάχνοντας σε κάθε ένα απο τα τρία μέρη στα οποία η Java3D χωρίζει τον κύλινδρο. Σε κορυφή, σώμα και βάση. Ας δούμε απο κοντά τις

εντολές. Ακολουθούν μόνο όσες αφορούν την κορυφή του κυλίνδρου. (Ομοίως υπάρχουν και οι εντολές για τα δύο άλλα μέρη του) :

```
GeometryArray geomTop =
(GeometryArray)cylinder1.getShape(Cylinder.TOP).getGeometry();
int verticesTop = geomTop.getValidVertexCount();
```

Μόλις αποθηκεύσαμε στο verticesTop το πλήθος των σημείων για αυτό το μέρος του κυλίνδρου. Στην επόμενη εντολή δημιουργούμε έναν προσωρινό πίνακα τριών θέσεων, τον vec[ ] και έναν πίνακα coordsTop[ ] στον οποίο θα αποθηκευθούν οι συντεταγμένες όλων των σημείων, δηλαδή το σύνολο των σημείων επί τρία που είναι οι συντεταγμένες για το κάθε σημείο.

```
float[] coordsTop = new float[verticesTop*3];
float[] vec = new float[3];
```

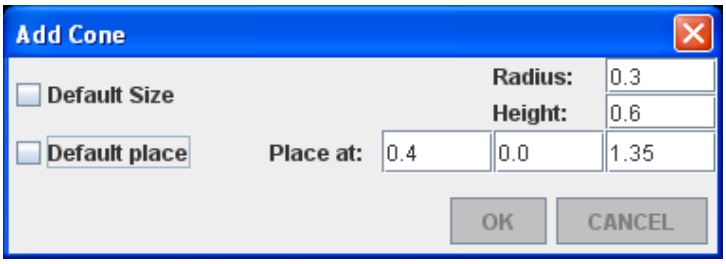
Στις επόμενες γραμμές η βασική εντολή είναι η geomTop.getCoordinate(i, vec) που απο εδώ καταλαβαίνουμε οτι το i δείχνει αύξων αριθμό σημείου, και στο vec[ ] αποθηκεύονται οι 3 συντεταγμένες του σημείου αυτού. Ένας τρόπος να αποθηκεύσουμε όλες τις συντεταγμένες σε έναν μονοδιάστατο πίνακα είναι ο εξής:

```
for (int i=0; i<verticesTop; i++){
geomTop.getCoordinate(i, vec);
coordsTop[3*i] = vec[0]+atX;
coordsTop[3*i + 1] = vec[1]+atY+h;
coordsTop[3*i + 2] = vec[2]+atZ;
}
```

Στην προτελευταία εντολή βλέπουμε οτι σε όλα τα σημεία η συντεταγμένη y έχει αυξηθεί κατα h που είναι το μισό του ύψους. Αυτό έχει γίνει γιατί ο κύλινδρος κατασκευάζεται σε διαφορετικό ύψος απο αυτό που τον θέλουμε. Στην πραγματικότητα κατασκευάζεται το μισό πάνω απο το επίπεδο σχεδιασμού και το άλλο μισό κάτω. Οπότε με αυτές τις εντολές όλα τα σημεία του κυλίνδρου βρίσκονται απο το επίπεδο y=0 και πάνω. Στη συνέχεια γίνεται η ίδια διαδικασία για τα άλλα δύο μέρη του κυλίνδρου, και τέλος εκτυπώνονται οι απαραίτητοι πίνακες στο αρχείο temp.obj. Επιστρέφοντας στο κεντρικό παράθυρο ο κύλινδρος έχει ήδη προστεθεί στη σκηνή και η εφαρμογή βρίσκεται πάλι σε κατάσταση αναμονής.

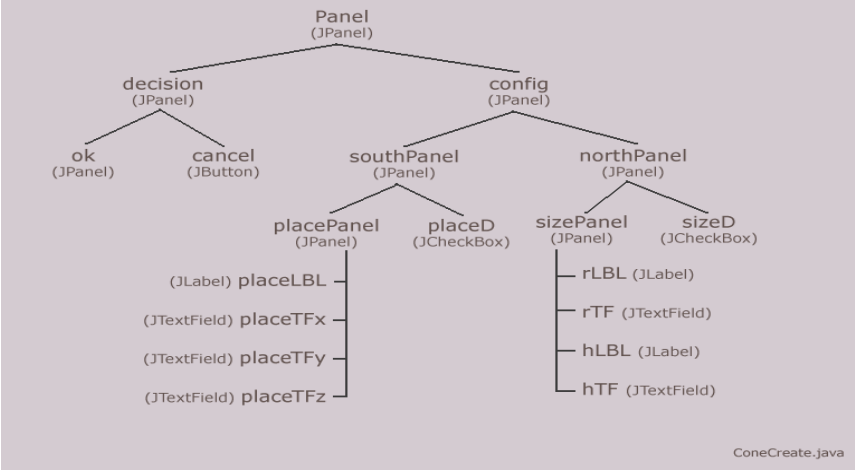
3.8 ΔΗΜΙΟΥΡΓΙΑ ΚΩΝΟΥ

Όταν ο χρήστης πατήσει το κουμπί που αναπαριστά τον κώνο στην μπάρα εργαλείων (ToolBar) ή μεταβεί μέσω του MenuBar στο Tools → Add Schemes → Cone , τότε εμφανίζεται στην οθόνη το εξής παράθυρο Add Cone που αποθηκεύει τις τιμές με τις οποίες θα δημιουργήσουμε τον κώνο. Είτε χρησιμοποιώντας τις προεπιλεγμένες τιμές είτε επιλέγοντας νέες και τοποθετώντας αυτές στα πεδία κειμένου του παρακάτω παραθύρου.



Εικόνα 3.16: Το παράθυρο προσθήκης του κώνου.

Ακολουθεί και ένα σχεδιάγραμμα της δομής του παραθύρου Add Cone (σχεδιάγραμμα 3.9).



Σχεδιάγραμμα 3.9: τα στοιχεία swing που δομούν το παράθυρο δημιουργίας κώνου

Άρα απο ότι μπορούμε να δούμε, οι τιμές τις οποίες χρειαζόμαστε για να κατασκευάσουμε τον κώνο είναι η ακτίνα του κύκλου της βάσης και το ύψος που επιθυμούμε να έχει. Επιπλέον όπως πάντα μπορεί ο χρήστης να δώσει τις συντεταγμένες στις οποίες θα προτιμούσε να το κατασκευάσουμε. Αφού αποθηκευθούν οι τιμές όπως πάντα, το ConeCreate.java καλεί το αρχείο Cone.java το οποίο είναι υπεύθυνο να φτιάξει τη δομή του αρχείου .obj έτσι ώστε να προστεθεί ο κώνος στο μέχρι στιγμής σύνολο των αντικειμένων. Οπότε όπως έχουμε ήδη αναφέρει νωρίτερα, για κάθε αντικείμενο υπάρχουν δύο αρχεία .java. Το πρώτο εμφανίζει το παράθυρο οπου αποθηκεύουμε τις τιμές των μεταβλητών, ενώ το δεύτερο ξεκινά τη λειτουργία του μόλις ο χρήστης πατήσει το ok. Ταυτόχρονα κρύβεται το παράθυρο αυτο, και το δεύτερο αρχείο που αναλαμβάνει τώρα δράση συνθέτει το κείμενο που θα γραφτεί στο αρχείο .obj για να εμφανισθεί σωστά το αντικείμενο. Όταν ολοκληρωθεί και αυτό, τότε επιστρέφουμε πάντα στο κεντρικό παράθυρο οπου δημιουργείται το αντικείμενο στην java3d και προστίθεται στη σκηνή. Αργότερα θα μιλήσουμε σε άλλο κεφάλαιο για τη μορφή των αντικειμένων μέσα στα αρχεία .obj . Προς το παρόν ασχολούμαστε με όσα αφορούν την java. Η δημιουργία του κώνου στην java γίνεται με την εντολή

Cone cone1 = new Cone(Global.coneRadius/3, Global.coneHeight/3, primflags, 35, 15, app);

ενώ όπως και στον κύλινδρο, για να φτιάξουμε μόνοι μας στο χέρι τους πίνακες με τις συντεταγμένες των σημείων θα έπαιρνε πάρα πολύ χρόνο (αποτελείται απο 74 σημεία στη συγκεκριμένη περίπτωση). Οπότε μία λύση είναι και πάλι να αφήσουμε τη Java3D να δημιουργήσει το αντικείμενο δίνοντάς της ακτίνα και ύψος, και έπειτα με κάποιες εντολές να πάρουμε σε έναν πίνακα όλες τις συντεταγμένες των σημείων. Όπως είδαμε προηγουμένως, ο κύλινδρος χωρίζεται σε τρία τμήματα, κορυφή – σώμα – βάση . Ο κώνος χωρίζεται σε δύο. Κυρίως σώμα (body) και βάση (κάλυμμα / τάπα – cap). Παρακάτω δίνεται ένα τμήμα του κώδικα που δείχνει τη δημιουργία του κώνου, καθώς και τις εντολές που χρειάζονται για να πάρουμε τις συντεταγμένες του κυρίως σώματος του κώνου (body). Ομοίως με τον ίδιο τρόπο παίρνουμε τις συντεταγμένες απο τη

βάση του κώνου (cap).

```
Cone cone2 = new Cone(radius, height, primflags, 35, 15, app);
float h = height/2;
```

Σημείωση: χρησιμοποιούμε σε κάποια σημεία τη μεταβλητή h η οποία έχει τη μισή τιμή του ύψους. Το κέντρο του κώνου εξ' ορισμού δημιουργείται και εφάπτεται στο κέντρο του σύμπαντος ή όπου αλλού το ζητήσουμε εμείς. Στη δική μας περίπτωση επιθυμούμε το σημείο που επιλέγουμε να εμφανιστεί ο κώνος να μήν αναφέρεται στο κέντρο του κώνου, αλλά στο κέντρο της βάσης του. Για να καταφέρουμε κάτι τέτοιο, απλά προσθέσαμε σε όλες τις συντεταγμένες y το μισό του ύψους του κώνου (h), με αποτέλεσμα να φαίνεται ολόκληρος πάνω στο επίπεδο προβολής (διαφορετικά ο μισός κώνος, απο τη μέση και κάτω δεν θα φαινόταν, ή αν φαινόταν θα ήταν ο μισός πάνω απο το πλάνο προβολής και ο άλλος μισός κάτω απο αυτό).

```
cone2.getShape(Cone.BODY).getGeometry().setCapability(Geometry.ALLOW_INTERSECT);
//////////////create the cone and store vertices in table//////////////////
GeometryArray geomBody =
(GeometryArray)cone2.getShape(Cone.BODY).getGeometry();
int verticesBody = geomBody.getValidVertexCount();
int verts = cone2.getNumVertices();
float[] coordsBody = new float[verticesBody*3];
float[] vec = new float[3];
for (int i=0; i<verticesBody; i++){
geomBody.getCoordinate(i, vec);
coordsBody[3*i] = vec[0]+atX;
coordsBody[3*i + 1] = vec[1]+atY+h;
coordsBody[3*i + 2] = vec[2]+atZ;
} …...
```

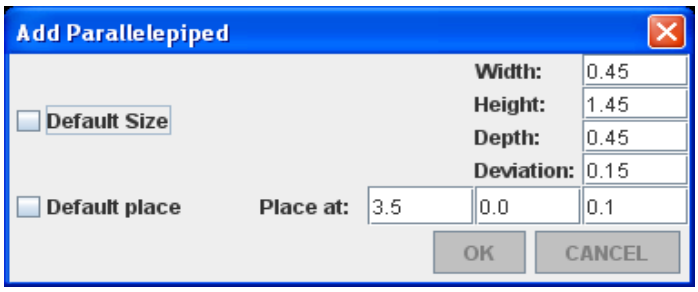
Όπως ακριβώς δηλαδή έγινε και στην περίπτωση του κυλίνδρου. Στη συνέχεια εκτελούνται και οι αντίστοιχες εντολές για τη βάση του κώνου. Αμέσως μετά ακολουθούν οι εντολές οπου ανοίγουν το αρχείο κειμένου όπου θα αποθηκευθούν τα δεδομένα που χρειάζονται τα αρχεία .obj για να δείξουν έναν κώνο. Περισσότερες λεπτομέρειες θα δούμε και αργότερα. Επόμενο αντικείμενο είναι το παραλληλεπίπεδο.

3.9 ΔΗΜΙΟΥΡΓΙΑ ΠΑΡΑΛΛΗΛΕΠΙΠΕΔΟΥ ΜΕ ΚΛΗΣΗ

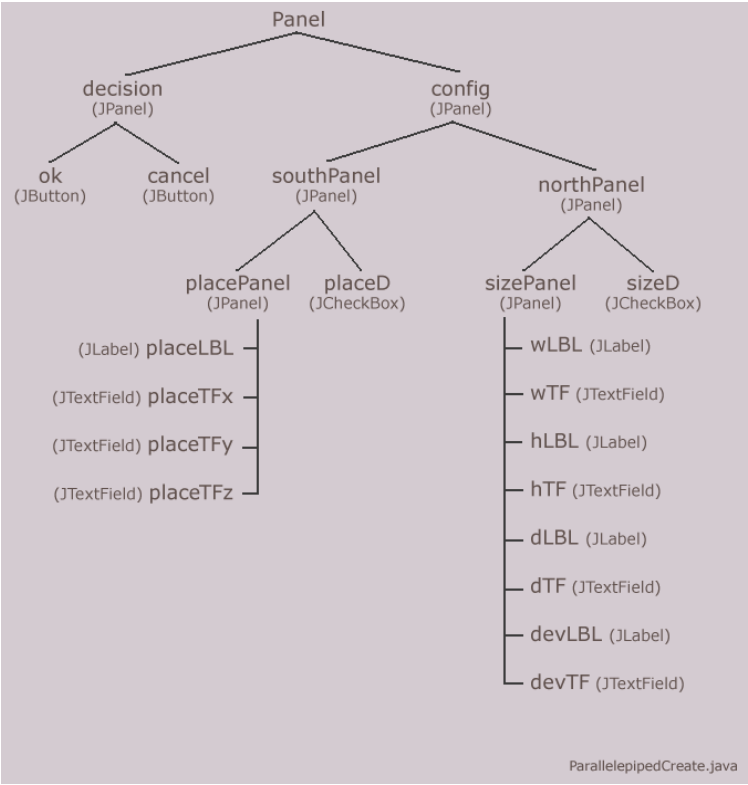
Πρόκειται για ένα αντικείμενο που όλες οι απέναντι πλευρές είναι παράλληλες, για παράδειγμα ένα παραλληλεπίπεδο είναι και ο κύβος. Μπορούμε να δημιουργήσουμε ένα παραλληλεπίπεδο πατώντας είτε το αντίστοιχο κουμπί είτε πηγαίνοντας μέσω του MenuBar στο Tools → Add Schemes → 4Sided Prisma οπου ενεργοποιείται το αρχείο ParallelepipedCreate.java. Οι τιμές που όπως βλέπουμε στο παρακάτω στιγμιότυπο του προγράμματος ζητούνται για να φτιάξουμε ένα παραλληλεπίπεδο είναι φυσικά το πλάτος, το ύψος, το βάθος, και η κλίση που επιθυμούμε να έχει (deviation). Η τιμή της κλίσης δίνεται σε μέτρα (m) και εκφράζει την απόσταση που θα έχει μία πάνω κορυφή απο την ίδια κάτω κορυφή στον άξονα των x. Δηλαδή αν η κλίση είναι 0 (μηδέν) τότε θα έχουμε ένα απλό ορθογώνιο.

Εικόνα 3.17: Στιγμιότυπο απο σκηνή με τρία παραλληλεπίπεδα με διαφορετικές κλίσεις.

Στην παραπάνω εικόνα βλέπουμε τρία παραλληλεπίπεδα με διαφορετικές κλίσεις (0.25 , 1.0 και 1.75).



Εικόνα 3.18: Το παράθυρο προσθήκης του παραλληλεπίπεδου.



Σχεδιάγραμμα 3.10: τα στοιχεία που δομούν το παράθυρο Add Parallelepiped

Μετά την αποθήκευση των τιμών απο το αρχείο ParallelepipedCreate.java, καλείται το αρχείο Parallelepiped.java να εκτελεστεί. Αυτό είναι υπεύθυνο να συνθέσει τους πίνακες που του ζητάμε εμείς χειροκίνητα να το κάνει, και να αποθηκεύσει τις τιμές σωστά στο αρχείο temp.obj όπου τοποθετούνται προσωρινά όλα τα αντικείμενα. Η μορφή του πίνακα που περιέχει όλες τις συντεταγμένες είνα κάπως έτσι

```
float[] vERTs ={ 0.0f+atX, 0.0f+atY, depth+atZ,          //bottom
                 width+atX, 0.0f+atY, depth+atZ,
                 width+atX, 0.0f+atY, 0.0f+atZ,
                 0.0f+atX, 0.0f+atY, 0.0f+atZ,
                 0.0f+atX, 0.0f+atY, depth+atZ,          //front
                 width+atX, 0.0f+atY, depth+atZ,
                 wU2+atX, height+atY, depth+atZ,
```

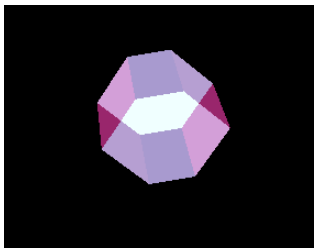
```
wU1+atX, height+atY, depth+atZ,
wU2+atX, height+atY, depth+atZ,      //right
width+atX, 0.0f+atY, depth+atZ,
width+atX, 0.0f+atY, 0.0f+atZ,
wU2+atX, height+atY, 0.0f+atZ,
width+atX, 0.0f+atY, 0.0f+atZ,       //back
wU2+atX, height+atY, 0.0f+atZ,
wU1+atX, height+atY, 0.0f+atZ,
0.0f+atX, 0.0f+atY, 0.0f+atZ,
0.0f+atX, 0.0f+atY, depth+atZ,       //left
0.0f+atX, 0.0f+atY, 0.0f+atZ,
wU1+atX, height+atY, 0.0f+atZ,
wU1+atX, height+atY, depth+atZ,
wU2+atX, height+atY, depth+atZ,      //top
wU2+atX, height+atY, 0.0f+atZ,
wU1+atX, height+atY, 0.0f+atZ,
wU1+atX, height+atY, depth+atZ
};
```

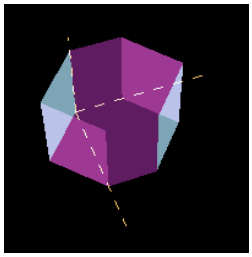
Όπου wU1=deviation και wU2=width+deviation.

Στα δεξιά σε μορφή σχολίων βλέπουμε την πλευρά στην οποία αντιστοιχούν αυτές οι τέσσερις μεταβλητές. Δεν αρκεί δυστυχώς απλά να ορίσουμε οκτώ σημεία, πρέπει να δηλώσουμε πλευρές στην πραγματικότητα, δηλαδή τέσσερα σημεία για την κάθε πλευρά, και μάλιστα πρέπει να τα παρουσιάσουμε με τη σωστή σειρά έτσι ώστε να εμφανιστούν σωστά στη σκηνή. Όπως παρατηρούμε ο σωστός τρόπος για να εμφανιστεί ένα αντικείμενο είναι να δηλώνουμε τις πλευρές με αντίθετη κατεύθυνση απο αυτήν του ρολογιού. Στο παραπάνω κομμάτι κώδικα βλέπουμε οτι το παραλληλεπίπεδο έχει δηλωθεί με την εξής σειρά. Κάτω, μπροστά, δεξιά, πίσω, αριστερά, και πάνω. Αφού αποθηκευθούν αυτά τα στοιχεία και τοποθετηθούν στο προσωρινό αρχείο κειμένου με κατάληξη .obj, τότε επιστρέφουμε στο κεντρικό παράθυρο οπου σχεδιάζεται το παραλληλόγραμμο και η εφαρμογή επιστρέφει σε κατάσταση αναμονής.

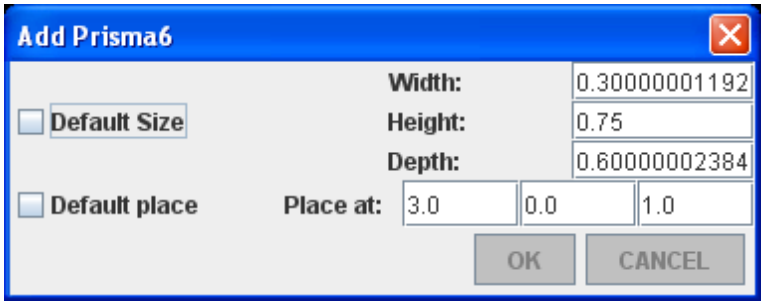
3.10 ΔΗΜΙΟΥΡΓΙΑ ΟΚΤΑΠΛΕΥΡΟΥ ΠΡΙΣΜΑΤΟΣ

Το πρίσμα είναι ένα αντικείμενο με έξι πλευρές αν είναι ανοιχτό πάνω και κάτω. Στη δική μας περίπτωση είναι αντικείμενο με οκτώ πλευρές και δώδεκα σημεία. Για να ανοίξουμε το παράθυρο Add Prisma6 μπορούμε είτε μέσω της μπάρας εργαλείων πατώντας το αντίστοιχο κουμπί, ή διαφορετικά μέσω του MenuBar στο Tools → Add Schemes → 6Sided Prisma οπου ενεργοποιείται το αρχείο Prisma6Create.java.



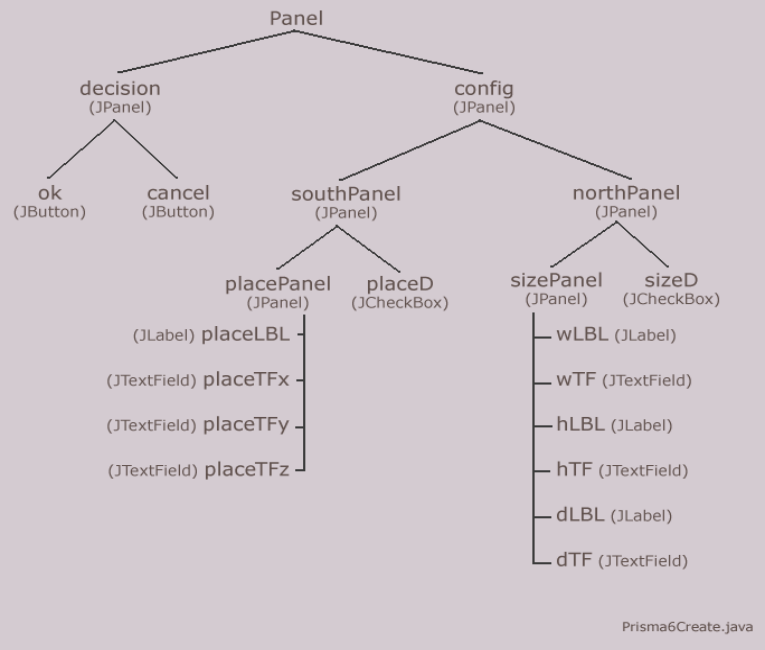


Εικόνες 3.19, 3.20: Στιγμιότυπα απο πρίσμα



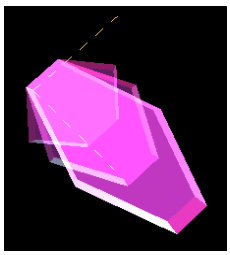
Εικόνα 3.21: Το παράθυρο προσθήκης του πρίσματος

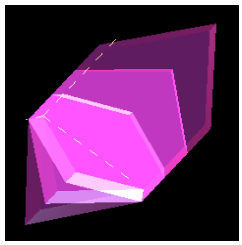
Οι μεταβλητές οι οποίες είναι απαραίτητες για να κατασκευαστεί ένα πρίσμα είναι το πλάτος, ύψος, και το βάθος. Καθώς επίσης φυσικά και οι συντεταγμένες του σημείου στο οποίο επιθυμούμε να το τοποθετήσουμε, το οποίο σημείο αναφέρεται στο πίσω κάτω δεξιά σημείο του πρίσματος όπως βλέπουμε στη δεύτερη εικόνα – στιγμιότυπο του προγράμματος.



Σχεδιάγραμμα 3.11: τα στοιχεία που συντάσσουν το παράθυρο δημιουργίας του πρίσματος

Το ύψος είναι προφανές οτι αναφέρεται στην απόσταση απο ένα σημείο στη βάση έως το αντίστοιχο σημείο στην κορυφή του πρίσματος. Το πλάτος αναφέρεται στο μέγεθος της κάθε πλευράς που εφάπτεται στο υποθετικό έδαφος και είναι στα πλάγια, ενώ το βάθος αναφέρεται στην απόσταση που έχουν οι δυο παράλληλες μεταξύ τους πλευρές οι οποίες είναι κάθετες στο υποθετικό έδαφος και οι οποίες είναι η μπροστά και πίσω πλευρά όπως τις βλέπει ο χρήστης.



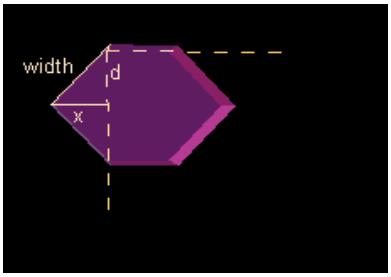


Εικόνα 3.22 (αριστερά): Στιγμιότυπο απο σκηνή με τρια πρίσματα διαφορετικού βάθους. Εικόνα 3.23 (δεξιά): Στιγμιότυπο απο σκηνή με τρια πρίσματα διαφορετικού πλάτους.

Στην πρώτη εικόνα βλέπουμε τρία πρίσματα με διαφορετικό βάθος, ενώ στη δεύτερη τρία πρίσματα με διαφορετικό πλάτος. Και εδώ όπως και στο παραλληλεπίπεδο οι συντεταγμένες υπολογίζονται στο χέρι και τοποθετούνται σε πίνακα. Ο κώδικας που χρησιμοποιείται είναι κάπως έτσι.

```
double height = Global.prisma6Height/3;        double h = height/2;
double depth = Global.prisma6Depth/3;          double d = depth/2;
double width = Global.prisma6Width/3;          double w = width/2;
double x1 = Math.pow(width, 2.0);
double x2 = Math.pow(d, 2.0);
double x3 = Math.abs(x1 - x2);
double x = Math.sqrt(x3);                      // because x = sqrt(w^2 – (depth/2)^2)
```

Κάπου εδώ να κάνουμε μια παρένθεση να εξηγήσουμε τις παραπάνω εντολές. Όπως βλέπουμε στο παρακάτω σχήμα για να βρούμε την κορυφή αριστερά θα



Εικόνα 3.24

πρέπει να υπολογίσουμε τη μεταβλητή x η οποία θα χρησιμοποιηθεί στη συντεταγμένη του άξονα x'x. Width είναι το πλάτος που δίνει ο χρήστης, και d είναι το μισό του βάθους που δίνει ο χρήστης (depth/2). Οπότε χρησιμοποιώντας το Πυθαγόρειο θεώρημα βρίσκουμε οτι $\mathbf{width^2 = d^2 + x^2}$.

Άρα $x = \sqrt{width^2 - d^2}$. Οπότε x1 ονομάστηκε η πράξη $width^2$, x2 ονομάστηκε η πράξη d^2, και x3 ονομάστηκε η αφαίρεση αυτών. Άρα στην επόμενη εντολή το x βρέθηκε απλά κάνοντας τη ρίζα του x3. Παρακάτω συνεχίζεται ο κώδικας και βλέπουμε τη δημιουργία του πίνακα των συντεταγμένων.

```
double[] verts = {0.0, 0.0, depth,          //bottom
                  width, 0.0, depth,
```

```
(width+x), 0.0, d,
width, 0.0, 0.0,
0.0, 0.0, 0.0,
-(x), 0.0, d,                    //eof bottom
0.0, 0.0, depth,                 //left-front
0.0, height, depth,
-(x), height, d,
-(x), 0.0, d,                    //eof left-front
0.0, 0.0, depth,                 //front
width, 0.0, depth,
width, height, depth,
0.0, height, depth,              //eof front
.....
};
```

Και πάλι έπρεπε να τοποθετήσουμε στον πίνακα κάθε πλευρά χωριστά, και με τη σωστή σειρά για να εμφανιστεί το αντικείμενο σωστά. Στη δεξιά πλευρά βλέπουμε σε μορφή σχολίων πότε ξεκινούν και πότε τελειώνουν οι συντεταγμένες της κάθε πλευράς. Έχουμε μεταφέρει εδώ μόνο τρείς πλευρές αλλά ομοίως φτιάχνονται και οι υπόλοιπες. Στη συνέχεια ακολουθούν οι εξής εντολές που παίρνουν τον παραπάνω πίνακα και τον μετατρέπουν σε πρίσμα.

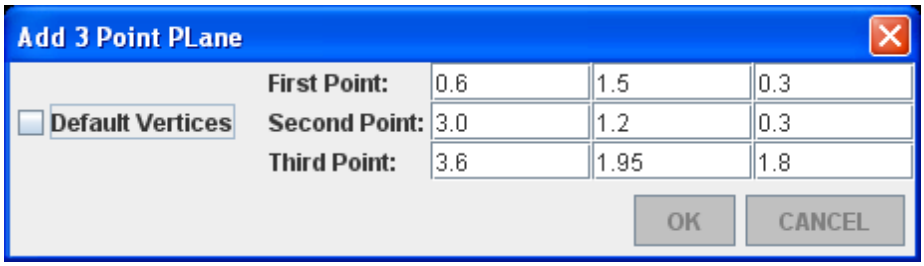
```
int[] stripCounts= {6, 4, 4, 4, 4, 4, 4, 6};
GeometryInfo gi = new GeometryInfo(GeometryInfo.POLYGON_ARRAY);
gi.setStripCounts(stripCounts);
gi.setCoordinates(verts);
gi.recomputeIndices();
new NormalGenerator().generateNormals(gi);
gi.recomputeIndices();
new Stripifier().stripify(gi);
gi.recomputeIndices();
GeometryArray ga = gi.getGeometryArray();
ga.setCapability(Geometry.ALLOW_INTERSECT);
Appearance app = createAppearance();
Shape3D prisma=new Shape3D(gi.getGeometryArray(), app);
TransformGroup prisma6tg = createTG((float)Global.prisma6X/3,
        (float(Global.prisma6Y/3), (float)(Global.prisma6Z/3));
prisma6tg.addChild(prisma);
```

```
prisma6Group.addChild(prisma6tg);
prisma6Group.compile();
scene.addChild(prisma6Group);
```

Μετά την τελευταία εντολή το πρίσμα έχει προστεθεί στη σκηνή. Επόμενο αντικείμενο είναι η επιφάνεια ορισμένη απο τρία σημεία.

3.11 ΔΗΜΙΟΥΡΓΙΑ ΠΛΑΝΟΥ ΤΡΙΩΝ ΣΗΜΕΙΩΝ

Πρόκειται για ένα απλό σύνολο τριών σημείων. Ο χρήστης έχει τη δυνατότητα να δώσει όποια τρία σημεία επιθυμεί στο εξής παράθυρο Add 3 Point Plane το οποίο εμφανίζεται είτε πατώντας στο αντίστοιχο κουμπί στην μπάρα εργαλείων στην αριστερή πλευρά της εφαρμογής, είτε πηγαίνοντας στο MenuBar και επιλέγοντας Tools → Add Schemes → Planes → 3Points Plane οπότε και ενεργοποιείται το αρχείο PlaneCreate.java.



Εικόνα 3.25: Το παράθυρο προσθήκης του πλάνου τριών σημείων

Όταν ο χρήστης πατήσει ok οι τιμές αποθηκεύονται στο αρχείο Global.java απο οπου μπορούμε να τις ανασύρουμε ανα πάσα στιγμή. Μόλις αποθηκευθούν οι τιμές, αρχίζει να εκτελείται το αρχείο Plane.java το οποίο αναλαμβάνει απλά να μεταφέρει τις συντεταγμένες στο προσωρινό αρχείο temp.obj. Κάπου εδώ να δούμε κάποιες λεπτομέρειες όλων των αρχείων που μεταφέρουν δεδομένα σε αρχεία κειμένου. Καταρχήν, όλα αυτά τα αρχεία πρέπει να εισάγουν τις βιβλιοθήκες java.io.*, δηλαδή διάφορες βιβλιοθήκες που μπορούν να εισάγουν ή να εξάγουν δεδομένα σε ή απο αρχεία java. Έπειτα πρέπει να δηλωθούν μέσα στην κλάση οι εξής μεταβλητές

```
File temp;
FileWriter out;
FileInputStream fin;
BufferedInputStream bis;
BufferedReader dis;
```

όπου temp είναι το αρχείο temp.obj. Ένα παράδειγμα οπου παίρνουμε δεδομένα απο ένα αρχείο κειμένου είναι το παρακάτω. Στο παράδειγμα το σενάριο είναι να ανοίξουμε το αρχείο και να μετρήσουμε πόσες συντεταγμένες έχει μέσα το αρχείο μέχρι στιγμής. Ο κώδικας για την εκτέλεση αυτού του σεναρίου είναι κάπως έτσι.

```
try{
fin = new FileInputStream(temp);
bis = new BufferedInputStream(fin);
dis = new BufferedReader(new InputStreamReader(bis));
inLine=dis.readLine();
while(inLine != null) {
        if(inLine.startsWith("v ")){
                countOfVertices++;
        }
        inLine=dis.readLine();
}
fin.close();
bis.close();
dis.close();
}catch(IOException ioe){
        System.out.println(ioe.getMessage());
}
System.out.println("there are "+countOfVertices+" vertices");
```

Το κομμάτι κώδικα που μόλις είδαμε όντως χρησιμοποιείται και υπάρχει και λόγος που γίνεται αυτό όμως δεν θα το μάθουμε ακόμη, αλλά όταν αναλύσουμε τα αρχεία obj και το πώς συντάσσονται. Ένα άλλο μικρό κομμάτι κώδικα όπου δείχνει το πώς τοποθετούμε δεδομένα σε κάποιο αρχείο temp είναι το παρακάτω.

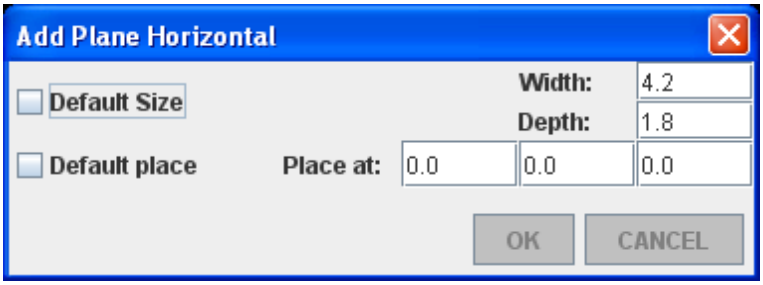
```
Try{
out = new FileWriter(temp, true);
out.write("\n\n# object Plano3D\n");
```

```
out.write("\n");
out.flush();
out.close();
}catch(java.io.IOException ioe) {
System.out.println(ioe.getMessage());
}
```

Η εντολή new FileWriter(temp, true) επισημαίνει οτι το αρχείο που θέλουμε να τοποθετήσουμε τα δεδομένα είναι το temp και οτι θέλουμε να γράψουμε στο τέλος του αρχείου, χωρίς να πειράξουμε τα προηγούμενα δεδομένα, να κάνουμε append δηλαδή.

3.12 ΔΗΜΙΟΥΡΓΙΑ ΟΡΙΖΟΝΤΙΑΣ ΠΑΡΑΛΛΗΛΟΓΡΑΜΜΗΣ ΕΠΙΦΑΝΕΙΑΣ

Πρόκειται για το την οριζόντια επιφάνεια που χρησιμέυει σαν πάτωμα στη σκηνή ή στο πλάνο προβολής. Αυτή η επιφάνεια ορίζεται απο πλάτος και βάθος.



Εικόνα 3.26: το παράθυρο προσθήκης του οριζόντιου επιπέδου

Όπως και στα άλλα αντικείμενα μπορούμε να ανοίξουμε αυτό το παράθυρο είτε πατώντας στο αντίστοιχο κουμπί στην μπάρα εργαλείων, είτε επιλέγοντας απο το MenuBar Tools → Add Schemes → Planes → Horizontal. Αφού το αρχείο PlaneHCreate.java αποθηκεύει τις τιμές όπως πάντα ξεκινά να τρέχει το αρχείο PlaneH.java το οποίο κατασκευάζει τον απαραίτητο πίνακα με τις συντεταγμένες και τον τυπώνει στο αρχείο tenp.obj. Όταν επιστρέφουμε στο κεντρικό παράθυρο βλέπουμε την επιφάνεια κατασκευασμένη και έτοιμη στη σκηνή.

```
planeHGroup = new BranchGroup();
planeHGroup.setCapability(BranchGroup.ALLOW_CHILDREN_EXTEND);
planeHGroup.setCapability(BranchGroup.ALLOW_CHILDREN_WRITE);
planeHGroup.setCapability(BranchGroup.ALLOW_DETACH);
int countPH=12;
int vertsPH=countPH/3;
float[] vertstablePH =  { 0.0f, 0.0f, 0.0f,
                          0.0f, 0.0f, Global.planeHDepth/3,
                          Global.planeHWidth/3, 0.0f, Global.planeHDepth/3,
                          Global.planeHWidth/3, 0.0f, 0.0f};
Point3f[] plPH = new Point3f[vertsPH];
int j=0;
int p=0;
try{
while( p<countPH) {
        plPH[j++] = new Point3f(vertstablePH[p++], vertstablePH[p++], vertstablePH[p+
+]);
}
}catch(Exception e){
        System.out.println(e.getMessage());
}
QuadArray polygon2 = new QuadArray (vertsPH, QuadArray.COORDINATES |
QuadArray.NORMALS);
polygon2.setCoordinates(0, plPH);
Vector3f polygonNormal = new Vector3f(0f,0.3f,-0.5f);
polygonNormal.normalize();
polygon2.setNormal(0,polygonNormal);
polygon2.setNormal(1,polygonNormal);
polygon2.setNormal(2,polygonNormal);
polygon2.setNormal(3,polygonNormal);
Shape3D planeH = new Shape3D(polygon2, app);
TransformGroup planeHtg = createTG(Global.planeHX/3, Global.planeHY/3,
Global.planeHZ/3);
planeHtg.addChild(planeH);
planeHGroup.addChild(planeHtg);
```

Άρα αρχικά δημιουργούμε ένα BranchGroup και επιλέγουμε τις ιδιότητες που θέλουμε έχει. Έπειτα δημιουργούμε έναν πίνακα με τις συντεταγμένες που

θέλουμε (vertstablePH), ο οποίο δυστυχώς δεν μας είναι και πολύ χρήσιμος αν δεν τον μετατρέψουμε σε πίνακα Point3f[ ], του οποίου τα στοιχεία είναι σημεία. Άρα στις αμέσως επόμενες εντολές μεταφέρουμε τις συντεταγμένες στον πίνακα plPH[ ]. Έπειτα με τη χρήση της συνάρτησης

new QuadArray (vertsPH, QuadArray.COORDINATES | QuadArray.NORMALS)

δημιουργούμε το πολύγωνο, και τελικά δημιουργούμε το planeH, το μετακινούμε στο σημείο που επιθυμεί ο χρήστης με τη συνάρτηση TransformGroup, η οποία προσδιορίζει μία κίνηση κάθε φορά. Κάπως έτσι κατασκευάζουμε και το επόμενο αντικείμενο που είναι μία κάθετη επιφάνεια.

3.13 ΔΗΜΙΟΥΡΓΙΑ ΚΑΘΕΤΗΣ ΠΑΡΑΛΛΗΛΟΓΡΑΜΜΗΣ ΕΠΙΦΑΝΕΙΑΣ

Το επόμενο αντικείμενο είναι η κάθετη επιφάνεια που χρησιμοποιείται και στην επιφάνεια προβολής. Είναι ένα απλό παραλληλόγραμμο που ορίζεται απο πλάτος και ύψος, και φυσικά ο χρήστης αν θέλει μπορεί να επιλέξει που το θέλει να εμφανιστεί με σημείο αναφοράς το κάτω αριστερά σημείο του παραλληλογράμμου.

Εικόνα 3.27: το παράθυρο προσθήκης του κάθετου επιπέδου

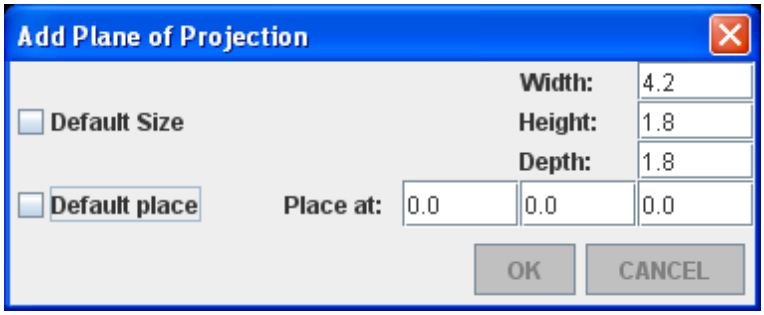
Για την ακρίβεια όλες οι εντολές είναι σχεδόν ίδιες με το προηγούμενο αντικείμενο απλώς οι τιμές που παίρνει το πρόγραμμα τοποθετούνται σε διαφορετικές θέσεις του πίνακα.

```
float[] vertstablePV = {  0.0f, 0.0f, 0.0f,
                          0.0f, Global.planeVHeight/3, 0.0f,
                          Global.planeVWidth/3, Global.planeVHeight/3, 0.0f,
                          Global.planeVWidth/3, 0.0f, 0.0f
                          };
```

Εκεί είναι οι μόνες διαφορές. Για να κατασκευάσουμε αυτήν την κάθετη επιφάνεια πρέπει να πατήσουμε το αντίστοιχο κουμπί απο την μπάρα εργαλείων, ή να το βρούμε μέσω του Menubar όπως και προηγουμένως, πηγαίνοντας στο Tools → Add Schemes → Planes → Vertical. Το τελευταίο αντικείμενο είναι ο συνδυασμός των δύο παραπάνω. Η επιφάνεια προβολής, που αποτελείται απο δύο ίδια παραλληλόγραμμα που τέμνονται κάθετα.

3.14 ΔΗΜΙΟΥΡΓΙΑ ΕΠΙΠΕΔΟΥ ΠΡΟΒΟΛΗΣ – ΔΥΟ ΚΑΘΕΤΕΣ ΜΕΤΑΞΥ ΤΟΥΣ ΠΑΡΑΛΛΗΛΟΓΡΑΜΜΕΣ ΕΠΙΦΑΝΕΙΕΣ

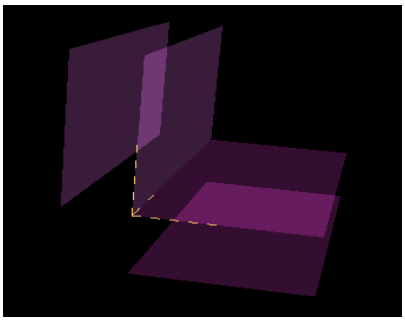
Φτάσαμε στο τελευταίο αντικείμενο της λίστας, το οποίο είναι ένας συνδυασμός δύο άλλων πολύ απλών αντικειμένων. Ορίζεται απο τρείς τιμές, πλάτος, ύψος και βάθος. Επίσης ο χρήστης μπορεί να δώσει το σημείο στο οποίο θέλει να την τοποθετήσει πάντα μέσα απο το παράθυρο που ενεργοποιεί το αρχείο PlanePCreate.java.



Εικόνα 3.28: Το παράθυρο προσθήκης του επιπέδου προβολής

Για να ενεργοποιηθεί το παράθυρο που βλέπουμε πρέπει ο χρήστης να επιλέξει

το τελευταίο κουμπί της μπάρας εργαλείων ή να πάει στο Menubar και να επιλέξει Tools → Add Schemes → Planes → Plane of Projection. Σε αυτήν την περίπτωση, ισχύουν όσα ισχύουν και για τα άλλα αντικείμενα, και μάλιστα υπάρχουν και πάλι δυο χωριστοί πίνακες ένας για κάθε μία επιφάνεια απο τις δύο απλά έχουν δύο κοινά σημεία.



Εικόνα 3.29

Στην εικόνα βλέπουμε την επιφάνεια προβολής στην προεπιλεγμένη θέση της, καθώς επίσης και τις δύο κάθετες επιφάνειες απο τις οποίες αποτελείται λιγο πιο απομακρυσμένες. Όταν ο χρήστης δώσει τα δεδομένα και πατήσει ok ενεργοποιείται όλος ο μηχανισμός που έχουμε αναφέρει. Αρχικά αποθηκεύονται οι τιμές στο αρχείο Global.java, ενώ στη συνέχεια εκτελείται το αρχείο PlaneP.java το οποίο δημιουργεί τους πίνακες συντεταγμένων και τους τοποθετεί με τη σωστή μορφή στο αρχείο temp.obj. Στη συνέχεια επιστρέφουμε στο κεντρικό παράθυρο οπου το αντικείμενο έχει ήδη προστεθεί στη σκηνή και οι τιμές με τις οποίες κατασκευάστηκε τοποθετήθηκαν στο History, που είναι ένα textArea, ένα πεδίο κειμένου δηλαδή, που δείχνει το ιστορικό της συγκεκριμένης εργασίας. Ποια αντικείμενα έχουν προστεθεί, πού και με τί μεγέθη.

Στο επόμενο κεφάλαιο θα δούμε επιτέλους τί είναι ένα αρχείο obj, πώς συντάσσεται, και όλες τις μικρές λεπτομέρειες που αφορούν την εφαρμογή.

ΚΕΦΑΛΑΙΟ 4 - ΜΟΡΦΗ ΑΡΧΕΙΟΥ OBJ

Είναι μορφή αρχείου κειμένου (text file format), και άρα μπορεί κάποιος αν το επιθυμεί να το επεξεργαστεί σε πρόγραμμα επεξεργασίας κειμένου. Ορίζει γεωμετρικά σχήματα και αναπτύχθηκε πρώτα απο την Wavefront Technologies για na βελτιώσει το πακέτο προηγμένης απεικόνισης animation. Η μορφή του αρχείου είναι ανοιχτή, άρα δωρεάν και ελεύθερο στην κυκλοφορία, και έχει εγκριθεί άπο πωλητές άλλων εφαρμογών τρισδιάστατων γραφικών. Το μεγαλύτερο μέρος της μορφής του αρχείου είναι διεθνώς αποδεκτές. Σε αυτήν την έκδοση, το OBJ αρχείο υποστηρίζει πολυγωνικά αλλά και αντικείμενα ελευθέρου σχήματος. Η πολυγωνική γεωμετρία χρησιμοποιεί σημεία, γραμμές, και πρόσωπα για να καθορίσει αντικείμενα, ενώ υποστηρίζει ελεύθερης μορφής αντικείμενα χρησιμοποιώντας καμπύλες και επιφάνειες.

Το μορφότυπο OBJ είναι αρχείο απλής μορφής δεδομένων που αναπαριστά τρισδιάστατη γεωμετρία – ονοματικά, τη θέση της κάθε κορυφής, την υφή των επιφανειών, τα faces των αντικειμένων που ορίζουν ένα πολύγωνο σαν μία λίστα απο συντεταγμένες, απο συντεταγμένες υφής και συντεταγμένες των normals.

Για να καταλάβουμε λίγο περισσότερο τα normals, ένα επίπεδο πολύγωνο τοποθετημένο σε ένα χώρο τρισδιάστατων συντεταγμένων πρέπει αναγκαστικά να έχει έναν προσανατολισμό, κοιτά προς κάποια συγκεκριμένη κατεύθυνση. Μία υποθετική ακτίνα, κάθετη σε μία επιφάνεια του πολυγώνου, η οποία ξεπροβάλει δείχνοντας προς τα έξω απο αυτήν, ονομάζεται normal ενός πολυγώνου. Αφού θα υπάρχουν πάντα δύο normals, ένα σε κάθε πλευρά της επιφάνειας, δείχνοντας προς αντίθετες κατευθύνσεις, η επιλογή της πλευράς απο την οποία σχεδιάζεται το normal, καθορίζει το μπροστινό μέρος ή “πρόσωπο” του πολυγώνου. Στα τρισδιάστατα γραφικά υπολογιστή (3D computer graphics), σε αντιδιαστολή με τον πραγματικό κόσμο, είναι συνηθισμένο για ένα πολύγωνο να έχει μόνο ένα “πρόσωπο” ή πλευρά, και επομένως μόνο ένα normal. Αυτό οφείλεται στο γεγονός οτι τα πολύγωνα χρησιμοποιούνται συνήθως για να δημιουργήσουν ένα κλειστό πλέγμα που

αντιπροσωπεύει την επιφάνεια ενός τρισδιάστατου αντικειμένου και συνεπώς η πίσω πλευρά του πολυγώνου είναι κρυμμένη εντός του αντικειμένου. Για να γλιτώσουμε χρόνο απόδοσης του αντικειμένου (render time), τα πολύγωνα διατηρούνται μονής όψης και το normal εξέχει μόνο απο την εκτεθειμένη όψη. Ωστόσο, μερικές φορές είναι απαραίτητο να δημιουργηθούν δύο όψεων πολύγωνα που έχουν normals που εξέχουν και απο τις δύο όψεις, και οι οποίες ως εκ τούτου μπορούν να σχεδιαστούν και απο τις δύο όψεις καθώς οι διαφορετικές όψεις εμφανίζονται κατα τη διάρκεια της πορείας ενός animation. Η μπροστινή πλευρά ενός πολυγώνου είναι συνήθως εγκατεστημένη σε ένα Model file οργανωμένα στη σειρά με την οποία παρατίθενται οι κορυφές του πολυγώνου, δεξιόστροφα (clockwise) ή αριστερόστροφα (counterclockwise) γύρω απο την μπροστινή πλευρά.

Τα normals μπορούν να συνδεθούν όχι μόνο με τις επίπεδες επιφάνειες των πολυγώνων, αλλά επίσης και με μεμονωμένα σημεία που απαρτίζουν τις κορυφές οπου πολύγωνα συναντιούνται στην επιφάνεια ενός μοντέλου. Αυτή η τεχνική χρησιμοποιείται στην αναπαράσταση για τη δημιουργία της εμφάνισης κυρτών επιφανειών αντί για επίπεδες πλευρές. Τέτοια normals κορυφών μπορούν να ανατεθούν απευθείας στο αρχείο του μοντέλου, αλλά συνήθως υπολογίζονται κατα τη διάρκεια της αναπαράστασης με τον υπολογισμό του μέσου όρου των γειτονικών πολυγώνων.

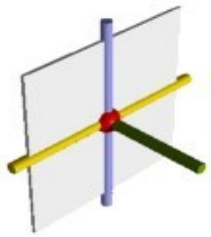
Εικόνα 4.1

Για να καταλάβουμε καλύτερα τα normals ας δούμε τα παρακάτω σχέδια. Στην πρώτη εικόνα βλέπουμε απλά το σύστημα συντεταγμένων μας με τους τρείς άξονες x, y και z.Αν κοιτάξουμε λίγο καλύτερα την εικόνα θα δούμε οτι είναι

φωτισμένοι απο ένα και μόνο φώς το οποίο έρχεται απο μπροστά και δείχνει προς τον άξονα των z. Η μπροστινή άκρη του άξονα των z είναι φωτισμένη, καθώς επίσης και οι μπροστινές πλευρές των αξόνων x και y.

Ο πιο απλός τρόπος φωτισμού είναι αμιγώς κατευθυντήριος. Δεν απλώνεται τριγύρω ούτε προέρχεται απο ένα σημείο. Είναι παραπλήσιο στο φως του ήλιου. (Ο ήλιος είναι σε τόσο μεγάλη απόσταση απο εμάς, που όλες οι ακτίνες του πέφτουν παράλληλα η μία στην άλλη.)

Αν τοποθετήσουμε ένα λευκό επίπεδο με το σημείο z = 0 έτσι ώστε να περνάει μέσα απο τους άξονες x και y.



Εικόνα 4.2

Ο άξονας z τώρα είναι το normal του επιπέδου. Το normal καθορίζει την επίδραση του φωτός στη επιφάνεια του πολυγώνου. Στην δεύτερη εικόνα, η κατεύθυνση του φωτός είναι ακριβώς αντίθετη απο την κατεύθυνση που έχει το normal του πολυγώνου. Αυτό παράγει την μεγαλύτερη ένταση φωτός στην επιφάνεια.

Ένα αρχείο OBJ επίσης υποστηρίζει εξομαλυντικούς παράγοντες για τις κυρτές επιφάνειες των αντικειμένων, καθώς επίσης και τη δυνατότητα ονομασίας ομάδων (groups) απο πολύγωνα. Υποστηρίζει επίσης υλικά παραπέμποντας σε ένα εξωτερικό αρχείο υλικού επονομαζόμενο MTL.

Ακολουθούν κάποιοι χαρακτήρες που χρησιμοποιούμε για να δηλώσουμε δεδομένα των κορυφών. Όπως (v) για τις γεωμετρικές συντεταγμένες (geometric vertices), το (vt) για τις συντεταγμένες υφής (texture vertices), που δεν έχουμε χρησιμοποιήσει και πολύ στην συγκεκριμένη εφαρμογή. Το (vn) που συμβολίζει τις συντεταγμένες των normals (vertex normals), και (f) για τα

πρόσωπα (faces) των αντικειμένων. Αυτά κυρίως χρησιμοποιούμε απο τα αρχεία OBJ, χωρίς αυτό να σημαίνει οτι είναι οι μοναδικές δυνατότητες που παρέχει αυτή η μορφή αρχείου.

Παρακάτω ακολουθεί ένα παράδειγμα αρχείου OBJ. Το συγκεκριμένο περιλαμβάνει έναν απλό κύβο με μήκος πλευράς 0.6 . Ας εξηγήσουμε λοιπόν, με λίγα λόγια αρχικά, αυτά που υπάρχουν πάντα σε ένα αρχείο OBJ, και έπειτα πρέπει να εξηγήσουμε για κάθε αντικείμενο ξεχωριστά.

```
1  # object Cubo
2
3  #vertices
4  v 0.0 0.0 0.0
5  v 0.0 0.6 0.0
6  v 0.6 0.6 0.0
7  v 0.6 0.0 0.0
8  v 0.0 0.0 0.6
9  v 0.0 0.6 0.6
10 v 0.6 0.6 0.6
11 v 0.6 0.0 0.6
12  #8 vertices
13
14 #texture vertices
15 vt 0.1 0.1 0.1
16 vt 0.1 0.1 0.1...
17 ...
18 vt 0.1 0.1 0.1
19 #8 texture vertices
20
21 #vertices normals
22 vn 0.0 0.0 0.0
23 vn 0.0 0.0 0.0...
24 ...
25 vn 0.0 0.0 0.0
26  #8 vertices normals
27
28 #faces section
29 f 1//1 2//2 3//3 4//4
30 f 5//5 6//6 7//7 8//8
31 f 1//1 2//2 6//6 5//5
32 f 8//8 7//7 3//3 4//4
33 f 6//6 2//2 3//3 7//7
34 f 5//5 1//1 4//4 8//8
35
```

Εικόνα 4.3: Κομμάτι κώδικα που εξάγει η εφαρμογή

Καταρχήν όσες γραμμές ξεκινούν με τον χαρακτήρα δίεση (#) ή είναι κενές δεν τις λαμβάνει υπόψη ένα πρόγραμμα που αναπαριστά OBJ. Άρα οι γραμμές με τη δίεση είναι σχόλια. Συνήθως βάζουμε σχόλια στην αρχή του αρχείου για να εξηγούμε τι αντικείμενο ακολουθεί, καθώς επίσης και πρίν κάθε τομέα δεδομένων όπως βλέπουμε στην εικόνα.

4.1 ΣΥΝΤΑΞΗ ΤΟΥ ΑΡΧΕΙΟΥ

Η πρώτη ομάδα δεδομένων είναι οι συντεταγμένες (z, y, z) του αντικειμένου. Πρόκειται για την εντολή

v x y z

Παρατηρούμε οτι όλες οι γραμμές που περιέχουν συντεταγμένες ξεκινούν όλες με τον χαρακτήρα v (Vertices). Είναι δήλωση για πολυγωνικά αντικείμενα αλλά και ελεύθερης μορφής. Προσδιορίζει μία γεωμετρική κορυφή και τις τρείς συντεταγμένες της. Οι λογικές καμπύλες και επιφάνειες απαιτούν και μία τέταρτη ομογενή συντεταγμένη, ονομαζόμενη βάρος (weight).

Σε όλο το αρχείο λοιπόν, όπου υπάρχει ο χαρακτήρας v στην αρχή, το πρόγραμμα θεωρεί οτι είναι συνέχεια των συντεταγμένων. Πράγμα που σημαίνει οτι μπορούμε απλά να προσθέσουμε κι άλλο αντικείμενο με πάρα πολύ απλό τρόπο. Απλά επικολλώντας στο τέλος του αρχείου τα νέα δεδομένα. Έτσι, και είναι ομαδοποιημένα τα δεδομένα του κάθε αντικειμένου, και ταυτόχρονα προστίθενται νέα αντικείμενα στην ίδια σκηνή. Επιπρόσθετα, οι συντεταγμένες αριθμούνται με την σειρά με την οποία τις γράφουμε. Άρα, αν αυτές είναι οι πρώτες οκτώ συντεταγμένες, έχουν η κάθε μία τον αριθμό τους και με αυτό χαρακτηρίζονται πια. Δηλαδή η πρώτη είναι η 1, η επόμενη 2, κ.ο.κ. Όταν προστεθεί το επόμενο αντικείμενο, η πρώτη του συντεταγμένη θα είναι η 9 και πάει λέγοντας. Επίσης το ίδιο ισχύει και για άλλες ομάδες συντεταγμένων. Όπως για συντεταγμένες υφής. Η πρώτη ονομάζεται 1, και οι υπόλοιπες

ακολουθούν με τη σειρά που είναι γραμμένες, ακόμα και όταν παρεμβάλλονται και άλλα δεδομένα ενδιάμεσα. Επιπρόσθετα, εκτός απο το να μετράμε συντεταγμένες απο την πρώτη και κάτω, απο την αρχή δηλαδή, μπορούμε να μετράμε τις συντεταγμένες αντίστροφα απο κάποιο στοιχεία στη λίστα και πάνω. Όταν μετράμε έτσι , απο κάτω προς τα πάνω, οι αναφορικοί αριθμοί είναι αρνητικοί. Δηλαδή η αναφορά του αριθμού -1υποδεικνύει την ακριβώς προηγούμενη συντεταγμένη του συγκεκριμένου στοιχείου. Η αναφορά του -2 υποδεικνύει τη συντεταγμένη πριν την προηγούμενη του ίδιου στοιχείου πάντα. Δηλαδή 2 γραμμές πάνω. Τώρα μπορούμε να εξηγήσουμε και τη σχετική αναφορά σε συντεταγμένες.

Τα αρχεία OBJ, λόγω της δομής της λίστας, είναι σε θέση να αναφέρουν κορυφές, normals, κτλ είτε με την απόλυτη θέση τους στη λίστα συντεταγμένων, ή χρησιμοποιώντας σχετικές τιμές, δηλαδή με αρνητικούς δείκτες που μετρούν αντίστροφα. Αυτή είναι μιά ιδιότητα η οποία επηρέασε τον τρόπο που έχει γραφτεί ο κώδικας, με την εξής έννοια. Εξ' αρχής, μιας και χρησιμοποιούμε το σύστημα συντεταγμένων των τριών αξόνων, και αφού για να προσανατολιστούμε φυσικά και χρησιμοποιούμε την αρχή των αξόνων, άρα υπάρχει πάντα μεγάλη πιθανότητα ο χρήστης να μετακινήσει το αντικείμενο σε σημείο με έστω και μία αρνητική συντεταγμένη, το οποίο όπως μόλις είδαμε, μπορεί να δημιουργήσει μεγάλο πρόβλημα στην αναπαράσταση των αντικειμένων.

Η Java3D όπως είδαμε δουλεύει πολύ καλά και με τις αρνητικές τιμές στον πίνακα συντεταγμένων, καθώς η αρχή των αξόνων βρίσκεται στο κέντρο του “σύμπαντος” αν μπορούμε να το πούμε έτσι.

Το σκεπτικό αλλάζει πολύ στην περίπτωση του κώδικα του αρχείο OBJ, οπου όταν ένα πρόγραμμα που διαβάζει αυτά τα αρχεία διαβάσει αρνητική τιμή στον πίνακα συντεταγμένων, δεν το θεωρεί συντεταγμένη, αλλά σχετική τιμή. Οπότε αν η προηγούμενη τιμή είναι (25, 1, 1) και η επόμενη είναι (-1, 2, 2) τότε το πρόγραμμα θα εμφανίζει το σημείο αυτό στις συντεταγμένες (25, 2, 2), δηλαδή το (-1) αντιστοιχεί στην προηγούμενη τιμή. Αν ήταν (-2) τότε θα έπαιρνε την τιμή που θα είχε το ακόμα προηγούμενο σημείο. Μόνο οι θετικές τιμές θεωρούνται

συντεταγμένες. Αυτο σημαίνει δηλαδή οτι η αρχή των αξόνων στη περίπτωση του κώδικα OBJ βρίσκεται στην αρχή του “σύμπαντος”, δεν υπάρχει συντεταγμένη πριν. Οπότε για να αποφύγουμε τα μπερδέματα, έπρεπε με κάποιο τρόπο να αποφύγουμε να χρησιμοποιήσουμε αρνητικές συντεταγμένες. Και για αυτό το λόγο μετακινήσαμε την αρχή των αξόνων στο σημείο (50, 50 , 50) με αποτέλεσμα να είναι απίθανο (αλλά οχι αδύνατο) να παραμορφωθεί οποιοδήποτε αντικείμενο. Αυτή τη ρύθμιση δεν την αντιλαμβάνεται πουθενά ο χρήστης που δίνει τις τιμές κανονικά, και απλά αθόρυβα σε κάποια στιγμή λίγο πριν αποθηκευθούν οι συντεταγμένες στο αρχείο, προστίθεται σε κάθε τιμή η τιμή 50. Ωστόσο δεν υποστηρίζουν όλα τα λογισμικά αυτήν την τελευταία προσέγγιση με τις σχετικές τιμές με το αρνητικό πρόσημο, και αντίστροφα κάποια λογισμικά εγγενώς γράφουν μόνο στην τελευταία μορφή (λόγω της ευκαιρίας του να προσαρτήσει τα στοιχεία χωρίς να χρειάζεται να υπολογίσει εκ νέου τα αντισταθμιστικά των κορυφών, κτλ), που οδηγεί σε περιστασιακές ασυμβατότητες.

Η δεύτερη ομάδα δεδομένων είναι οι συντεταγμένες της υφής. Πρόκειται για την εντολή

vt u v w

Επίσης δήλωση πολυγωνικού αλλά και ελεύθερης μορφής αντικειμένου. Προσδιορίζει μία κορυφή υφής και τις συντεταγμένες της. Μία μονοδιάστατη υφή απαιτεί μόνο την πρώτη συντεταγμένη υφής (u). Μία δισδιάστατη υφή απαιτεί τις δύο πρώτες συντεταγμένες (u και v) για να περιγραφεί, και μία τρισδιάστατη υφή απαιτεί και τις τρείς συντεταγμένες. Η πρώτη (u) είναι η τιμή για την οριζόντια κατεύθυνση της υφής. Η δεύτερη (v) είναι η τιμή για την κάθετη κατεύθυνση της υφής, και η προεπιλογή είναι στο 0 (μηδέν).Η τρίτη (w) είναι η τιμή που προσδιορίζει το βάθος της υφής, και η προεπιλεγμένη τιμή της είναι επίσης 0 (μηδέν).

Η τρίτη ομάδα δεδομένων είναι οι συντεταγμένες των normals. Η αντίστοιχη εντολή είναι η εξής

vn i j k

Και αυτή η εντολή χρησιμοποιείται και για πολυγωνικά αλλά και για αντικείμενα ελεύθερης μορφής. Προσδιορίζει ένα διάνυσμα με συνιστώσες i, j και k. Τα

normals κορυφών επηρεάζουν την ομαλότητα της σκίασης και την αναπαράσταση της γεωμετρίας. Για τα πολύγωνα, τα normals κορυφών χρησιμοποιούνται στη θέση των πραγματικών προσώπων των normals. Για τις επιφάνειες, τα normals κορυφών παρεμβάλλονται πάνω απο την επιφάνεια και αντικαθιστούν τα πραγματικά normal της επιφάνειας. Όταν υπάρχουν normals κορυφών , υπερισχύουν των smoothing groups (εξομαλυντικών groups).

4.2 ΑΝΑΦΟΡΑ ΟΜΑΔΩΝ ΚΟΡΥΦΩΝ

Κάποια στοιχεία, όπως τα πρόσωπα και οι επιφάνειες, ίσως έχουν ένα τρίστιχο αριθμών που αναφέρονται σε δεδομένα κορυφών. Αυτοί οι αριθμοί είναι οι αναφορικοί αριθμοί για μία γεωμετρική κορυφή, μία κορυφή υφής, και μία κορυφή normal. Πρόκειται για την τέταρτη ομάδα δεδομένων, οπου κάθε εντολή ξεκινά με f (faces). Αυτοί οι αναφορικοί αριθμοί πρέπει να είναι στη σειρά και να χωρίζονται απο κάθετο (/ -slash), ενώ μέσα σε κάθε τρίστιχο δεν υπάρχουν κενά ανάμεσα σε αριθμούς και σε κάθετους. Σε κάθε σειρά μπορεί να υπάρχουν πάνω απο ένα τρίστιχο απο γεωμετρική κορυφή/κορυφή υφής/κορυφή normal. Ακολουθεί ένα παράδειγμα face για ένα αντικείμενο με τέσσερις πλευρές.

f 1/1/1 2/2/2 3/3/3 4/4/4

Αν χρησιμοποιούσαμε τα v για την γεωμετρική κορυφή, vt για την κορυφή υφής και vn για την normal κορυφή, τότε η εντολή θα ήταν κάπως έτσι

f v/vt/vn v/vt/vn v/vt/vn v/vt/vn

Εαν υπάρχουν μόνο γεωμετρικές κορυφές και κορυφές normals ενώ δεν υπάρχουν κορυφές υφής για ένα πρόσωπο στοιχείου, τότε μπορούμε να χρησιμοποιήσουμε απλά δύο καθέτους (//) χωρίς να βάλουμε αριθμό ανάμεσα. Όπως έχουμε κάνει και στον κύβο που είναι στο παράδειγμα.

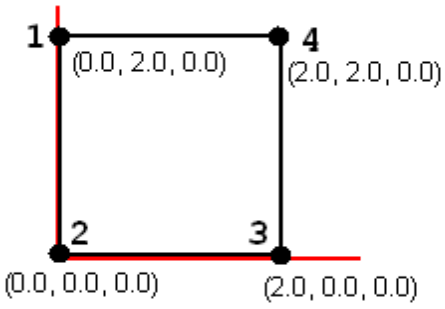
f 1//1 2//2 3//3 4//4

Όταν γράφουμε αυτά τα τρίστιχα πρέπει να είμαστε προσεκτικοί και συνεπείς στον τρόπο που αναφερόμαστε στα δεδομένα κορυφών. Για παράδειγμα, θεωρείται λάθος να δώσουμε κορυφές υφής για κάποιες συντεταγμένες, αλλά όχι για όλες. Για να εξηγήσουμε πώς ακριβώς δουλεύει το τμήμα των faces, θα παραθέσουμε ένα μικρό παράδειγμα αρχικά, για ένα απλό τετράγωνο. Για

λόγους ευκολίας θα γράψουμε μόνο τις συντεταγμένες και τα faces όπως πρέπει να εμφανίζονται.

```
v   0.000000   2.000000   0.000000
v   0.000000   0.000000   0.000000
v   2.000000   0.000000   0.000000
v   2.000000   2.000000   0.000000
f   1   2   3   4
```

Αυτό το τετράγωνο λοιπόν έχει μήκος πλευράς 2 μονάδες μέτρησης και έχει θετική κατεύθυνση (κοιτάζει προς την κάμερα). Ας προσέξουμε οτι η σειρά που τοποθετούνται τα σημεία πάνω στο τετράγωνο είναι αντίστροφη απο τη φορά του ρολογιού (αριστερόστροφα). Αυτή η σειρά τοποθέτησης προσδιορίζει οτι το αντικείμενο κοιτάζει μπροστά.



Εικόνα 4.4

Αφού πήραμε μία ιδέα, τώρα μπορούμε να εξηγήσουμε λίγο τι μπορεί να δούμε στα αρχεία .obj που εξάγει η εφαρμογή. Θα τα περιγράψουμε και πάλι με τη σειρά που εμφανίζονται στην μπάρα εργαλείων, η οποία καμία σχέση δεν έχει με τη σειρά δυσκολίας και πολυπλοκότητας.

4.2.1 Το σημείο

Αυτό το διαπιστώνουμε κιόλας απο την αρχή, απο το πρώτο αντικείμενο, το σημείο. Φυσικά δεν γινόταν να βάλουμε απλά μία συντεταγμένη, γιατί δεν θα ήταν ορατό στο μάτι σε πολλά προγράμματα αναπαράστασης αρχείων .obj,

οπότε η αναγκαστική επιλογή ήταν να χρησιμοποιήσουμε μία σφαίρα μεγέθους 0.01. Η εντολή που χρησιμοποιήσαμε στην δημιουργία αντικειμένου είναι η

Sphere sphere1 = new Sphere(0.01f, primflags, 30, app);

και απο εκεί πήραμε τις συντεταγμένες των σημείων που κατασκεύασε η java για εμάς, τα οποία σημεία ήταν 640 στο πλήθος. Αυτό εξηγεί μάλλον γιατί δεν το κάναμε χειροκίνητα, δηλαδή δίνοντας συντεταγμένες. Αφού τις πήραμε (με την εντολή getCoordinate(i, vec)) και τις τοποθετήσαμε σε έναν πίνακα, τις εκτυπώσαμε με την ίδια σειρά στο αρχείο temp.obj. Στη συνέχεια βάλαμε τυχαία texture vertices και vertices normals, ενώ το άλλο και ακόμα πιο σημαντικό μέρος είναι το τελευταίο μέρος που καθορίζουμε την ακολουθία των σημείων, εκεί οπου δηλαδή ενώνουμε τα σημεία και καθορίζουμε το σχήμα του αντικειμένου. Ακολουθούν οι δύο πρώτες και οι δύο τελευταίες σειρές στη δήλωση των faces

#faces section
f 1//1 320//320 321//321 2//2
f 2//2 321//321 322//322 3//3
…...
f 638//638 318//318 319//319 639//639
f 639//639 319//319 320//320 1//1

Όπως βλέπουμε, ακόμα κι εδώ έπρεπε να κατασκευάσουμε έναν βρόγχο έτσι ώστε να μην χρειαστεί να κάνουμε όλους αυτούς τους συνδυασμούς στο χέρι. Να σημειώσουμε κάτι πολύ σημαντικό εδώ. Η πρώτη σειρά συντεταγμένων, ονομάστηκε 1, επειδή το σημείο ήταν το πρώτο αντικείμενο που κατασκευάστηκε και προστέθηκε στο αρχείο temp.obj. Αν είχαμε προσθέσει κι άλλο αντικείμενο πριν, τότε το 1 δεν θα ανήκε στα σημεία της σφαίρας. Οπότε δεν αρκεί να βρούμε πώς θα συνδέσουμε τα σημεία της σφαίρας. Πρέπει αναγκαστικά να έχουμε βρεί τρόπο να ξεχωρίζουμε ποιές είναι οι συντεταγμένες που χρησιμοποιεί η σφαίρα. Αυτό γίνεται με τον εξής τρόπο. Κάθε φορά πριν προσθέσουμε ένα νέο αντικείμενο στο αρχείο, μετράμε το πλήθος των συντεταγμένων που βρίσκονται ήδη μέσα σε αυτό.

try{
fin = new FileInputStream(temp);
bis = new BufferedInputStream(fin);

```
dis = new BufferedReader(new InputStreamReader(bis));
inLine=dis.readLine();
while(inLine != null) {
if(inLine.startsWith("v ")){
countOfVertices++;
}
inLine=dis.readLine();
}
fin.close();
bis.close();
dis.close();
}catch(IOException ioe){
System.out.println(ioe.getMessage());
}
System.out.println("there are "+countOfVertices+" vertices");
```

Άρα πριν αρχίσουμε να τυπώνουμε συντεταγμένες σημείων στο αρχείο temp.obj ήδη γνωρίζουμε πόσα σημεία έχουν δηλωθεί. Αυτή η πληροφορία είναι άκρως σημαντική για να φτιάξουμε τα faces. Αυτά συνδυάζονται με προηγούμενες παρατηρήσεις που κάναμε. Είπαμε λοιπόν προηγουμένως οτι μπορούμε να αναφερθούμε σε σημεία αναφέροντας την θέση τους. Και οτι η αρίθμηση συνεχίζεται ακόμα και αν παρεμβάλλονται και άλλες πληροφορίες ενδιάμεσα. Θα είναι πιο εύκολο να κατανοήσουμε όλα αυτά προχωρώντας σε επόμενα αντικείμενα.

4.2.2 Η απλή γραμμή ορισμένη απο δύο σημεία

Στην απλή γραμμή για παράδειγμα, που ορίζεται απο δύο σημεία. Τα προγράμματα αναπαράστασης αρχείων .obj χρειάζονται 110 σημεία για να προσδιορίσουν μία ευθεία. Στη πραγματικότητα και 2 μόνο αρκούν, αλλά αν δεν βάλουμε 110 δεν την αναγνωρίζουν σαν ευθεία. Οπότε αυτό που κάναμε ήταν να δώσουμε τις ίδιες συντεταγμένες 109 φορές, τις συντεταγμένες του πρώτου σημείου δηλαδή, και σαν σημείο 110 βάλαμε τις συντεταγμένες του δεύτερου σημείου. Τώρα, επειδή την ευθεία την προσθέσαμε μετά το σημείο, η πρώτη σειρά συντεταγμένων της ευθείας έχει τον αριθμό 641. Πάντα λοιπόν, σε κάθε

αρχείο χρησιμοποιούμε την μεταβλητή countOfVertices.

#faces section
f 641//641 642//642 641//641
f 642//642 641//641 642//642
.....
f 749//749 748//748 749//749
f 750//750 749//749 750//750

Άρα για να τυπώσουμε αυτές τις τιμές κατασκευάσαμε έναν βρόγχο χρησιμοποιώντας πάντα τη μεταβλητή (countOfVertices+i) όπου i ήταν τα σημεία της ευθείας, απο 1 έως 110. Κάπως έτσι λειτούργησε σε όλα τα αντικείμενα.

4.2.3 Η ευθεία ορισμένη απο μήκος και δύο γωνίες

Η ευθεία ορισμένη απο μήκος και δύο γωνίες είναι ακριβώς το ίδιο καθώς όλοι οι υπολογισμοί γίνονται πριν αποθηκευθούν οι συντεταγμένες σε πίνακα. Άρα πάλι έχουμε να κάνουμε με 110 σημεία, οπου τα πρώτα 109 είναι ίδια.

#faces section
f 751//751 752//752 751//751
f 752//752 751//751 752//752
...
f 859//859 858//858 859//859
f 860//860 859//859 860//860

Εδώ βλέπουμε τα faces της ευθείας η οποία είναι το τρίτο αντικείμενο στο αρχείο, και τα σημεία της έχουν τους αριθμούς απο 751 έως 860 όπως βλέπουμε παραπάνω. Προηγούνται το σημείο (640 σειρές συντεταγμένων) και η απλή ευθεία (110 σειρές συντεταγμένων), άρα σύνολο προηγούμενων σειρών συντεταγμένων 750.

4.2.4 Ο κύβος

Ακολουθεί ο κύβος, οποίος αποτελείται απο 8 σημεία που υπολογίζουμε εμείς έχοντας το μήκος της κάθε πλευράς και το σημείο στο οποίο ακουμπά η πίσω και κάτω αριστερά κορυφή του κύβου. Τα faces είναι εξίσου απλά και εύκολα και

θα είναι κάπως έτσι

#faces section
f 1//1 2//2 4//4 3//3
f 5//5 6//6 8//8 7//7
f 1//1 2//2 6//6 5//5
f 8//8 7//7 3//3 4//4
f 6//6 2//2 4//4 8//8
f 5//5 7//7 3//3 1//1

Στη συγκεκριμένη περίπτωση βέβαια, οπου ο κύβος είναι το τέταρτο αντικείμενο, αν βάζαμε τα παραπάνω faces, θα είχαμε πρόβλημα. Χρησιμοποιώντας πάλι την μεταβλητή countOfVertices αποκτούμε το παρακάτω αποτέλεσμα.

#faces section
f 861//861 862//862 864//864 863//863
f 865//865 866//866 868//868 867//867
f 861//861 862//862 866//866 865//865
f 868//868 867//867 863//863 864//864
f 866//866 862//862 864//864 868//868
f 865//865 867//867 863//863 861//861

Οι αριθμοί έχουν τοποθετηθεί στα faces με τέτοιο τρόπο ώστε το πρόγραμμα που αναπαριστά τα αντικείμενα να γνωρίζει ποια είναι η μπροστινή πλευρά του αντικειμένου. Μετά την προσθήκη του κύβου έχουμε συνολικά (640 + 110 + 110 + 8 =) 868 σειρές συντεταγμένων.

4.2.5 Η πυραμίδα

Η πυραμίδα αποτελείται απο 5 κορυφές, οι οποίες υπολογίζονται πάλι στο χέρι αφου γνωρίζουμε φυσικά ύψος και πλάτος βάσης, καθώς και τιο σημείο στο οποίο θα ακουμπά η κάτω αριστερα κορυφή της πυραμίδας. Αφού αποθηκευθούν οι συντεταγμένες σε πίνακα, εκτυπώνονται στο αρχείο temp.obj πάντα με τη χρήση της μεταβλητής countOfVertices.

#faces section
f 869//869 870//870 871//871 869//869
f 869//869 871//871 872//872 869//869
f 869//869 872//872 873//873 869//869

f 869//869 873//873 870//870 869//869

f 870//870 871//871 872//872 873//873

f 872//872 873//873 870//870 871//871

Βλέπουμε οτι οι συντεταγμένες της πυραμίδας έχουν τις τιμές απο 869 έως 871 αφού έχουν τοποθετηθεί μετά απο τα παραπάνω 4 αντικείμενα. Αν είχαμε κατασκευάσει μόνο την πυραμίδα τα faces θα ήταν ως εξής.

#faces section
f 1//1 2//2 3//3 1//1
f 1//1 3//3 4//4 1//1
f 1//1 4//4 5//5 1//1
f 1//1 5//5 2//2 1//1

f 2//2 3//3 4//4 5//5
f 4//4 5//5 2//2 3//3

Οι τέσσερις πρώτες σειρές συνδέουν την κορυφή της πυραμίδας (1) με τα υπόλοιπα 4 σημεία, κατασκευάζοντας τις 4 πλευρές της, ενώ οι 2 επόμενες σειρές συνδέουν τα σημεία της βάσης της πυραμίδας. Για να μην μπερδευόμαστε άλλο με την μεταβλητή countOfVertices, και για να καταλαβαίνουμε καλύτερα τα faces, τα επόμενα αντικείμενα θα τα κατασκευάζουμε μόνα τους, σε νέα projects αντικειμένων.

4.2.6 Ο κύλινδρος

Ακολουθεί ο κύλινδρος με τα 1.154 σημεία του. Για άλλη μία φορά ζητήσαμε απο την java να κατασκευάσει έναν κύλινδρο και έπειτα πήραμε τις συντεταγμένες απο αυτόν τον έτοιμο κύλινδρο γαι να φτιάξουμε τον πίνακα συντεταγμένων που τοποθετήσαμε στο αρχείο temp.obj. Σε αυτό το αντικείμενο ήταν πιο δύσκολο να βρούμε τρόπο να συνδέσουμε τα σημεία με βρόγχο. Τελικά κατασκευάστηκαν τρείς διαφορετικοί βρόγχοι για κάθε κομμάτι απο τα τρία απο τα οποία αποτελείται ο κύλινδρος. Σώμα, βάση και καπέλο/καπάκι.

```
183     out.write("\n\n");
184     //faces section
185     out.write("#faces section\n");
186     j=countOfVertices+2;
187     while(j < countOfVertices+verticesTop) {
188         out.write("\nf "+(countOfVertices+1)+"//"+(countOfVertices+1));
189         out.write(" "+(j)+"//"+(j));
190         out.write(" "+(j+1)+"//"+(j+1));
191         out.write(" "+(countOfVertices+1)+"//"+(countOfVertices+1));
192         j++;
193     }j = countOfVertices+verticesTop+1;
194     while(j< (countOfVertices+verticesTop+verticesBody-1)){
195         out.write("\nf "+(j-1)+"//"+(j-1));
196         out.write(" "+(j)+"//"+(j));
197         out.write(" "+(j+1)+"//"+(j+1));
198         out.write(" "+(j+2)+"//"+(j+2));
199         j++;
200     }j = countOfVertices+verticesTop+verticesBody+2;
```

Εικόνα 4.5: στιγμιότυπο κώδικα java απο την εφαρμογή

Στην παραπάνω εικόνα βλέπουμε τους βρόγχους που τυπώνουν στο αρχείο τα faces για το καπάκι και το σώμα του κυλίνδρου. Όπως γίνεται για το καπάκι ομοίως γίνεται και για την βάση.

#faces section
f 1//1 2//2 3//3 1//1
f 1//1 3//3 4//4 1//1
f 1//1 4//4 5//5 1//1
f 1//1 5//5 6//6 1//1
f 1//1 6//6 7//7 1//1
f 1//1 7//7 8//8 1//1
f 1//1 8//8 9//9 1//1
f 1//1 9//9 10//10 1//1
f 1//1 10//10 11//11 1//1
f 1//1 11//11 12//12 1//1
....
f 1118//1118 1151//1151 1152//1152 1118//1118
f 1118//1118 1152//1152 1153//1153 1118//1118
f 1118//1118 1153//1153 1154//1154 1118//1118

4.2.7 Ο κώνος

Επόμενο αντικείμενο είναι ο κώνος, ο οποίος αποτελείται απο 74 σημεία. Για άλλη μία φορά και τελευταία ζητάμε απο την java να κατασκευάσει για εμάς έναν κώνο έτσι ώστε να πάρουμε έτοιμες τις συντεταγμένες των σημείων απο την java. Αρκεί φυσικά να ορίσουμε ακτίνα της βάσης, ύψος του κώνου καθώς και το σημείο στο οποίο θέλουμε να βρίσκεται το κέντρο της βάσης του κώνου. Για να φτιάξουμε τα faces αρκούσε να παρατηρήσουμε οτι στις συντεταγμένες που μας έδωσε η java, το πρώτο σημείο ήταν η κορυφή του κώνου ενώ το σημείο με αριθμό 38 ήταν το κέντρο της βάσης. Άρα αρκούσε να συνδέσουμε όλα τα υπόλοιπα σημεία με αυτά τα δύο.

```
#faces section
f 1//1 1//1 2//2 38//38
f 1//1 2//2 3//3 38//38
...
f 1//1 72//72 73//73 38//38
f 1//1 73//73 74//74 38//38
```

4.2.8 Το παραλληλεπίπεδο

Το παραλληλεπίπεδο είναι ένα αντικείμενο με 24 σημεία. Στην πραγματικότητα πρόκειται για 8 διαφορετικά σημεία, τα οποία υπολογίζουμε εμείς στο χέρι αρκεί να έχουμε το ύψος, το πλάτος, το βάθος και την απόκλιση που θέλουμε να έχει η πάνω πλευρά απο την κάτω στον άξονα x'x. Τα faces υπολογίστηκαν επίσης στο χέρι, αφού ήταν μόνο 6.

```
#faces section
f 1//1 2//2 3//3 4//4
f 5//5 6//6 7//7 8//8
f 9//9 10//10 11//11 12//12
f 13//13 14//14 15//15 16//16
f 17//17 18//18 19//19 20//20
f 21//21 22//22 23//23 24//24
```

Τα παραπάνω faces φυσικά ισχύουν μόνο στην περίπτωση οπου το παραλληλεπίπεδο κατασκευάστηκε πρώτο. Γι' αυτό βλέπουμε οτι τα faces χρησιμοποιούν τις κορυφές 1 έως 24 για να σχηματίσουν το αντικείμενο. Αν υπήρχαν και άλλα αντικείμενα πριν απο αυτό, τα faces θα χρησιμοποιούσαν τις κορυφές (countOfVertices+1) έως (countOfVertices+24), οπου countOfVertices

το πλήθος των προηγούμενων κορυφών.

4.2.9 Το πρίσμα των 8 πλευρών

Ακολουθεί το πρίσμα, το ένατο αντικείμενο στην μπάρα εργαλείων, το οποίο αποτελείται απο 12 σημεία τα οποία βρίσκουμε εμείς μετά απο υπολογισμούς αρκεί να έχουμε τα βάθος, ύψος, πλάτος και το σημείο στο οποίο θα ακουμπά η πίσω-κάτω-αριστερά κορυφή του πρίσματος. Δυστυχώς ή ευτυχώς, η java παίρνει χωριστά τις μεταβλητές για κάθε πλευρά απο τις 8 του πρίσματος οπότε κάπως έτσι φτάνουμε να έχουμε 108 σειρές συντεταγμένων στο αρχείο .obj. Η τοποθέτηση των συντεταγμένων έγινε και πάλι χειροκίνητα, η τουλάχιστον το μεγαλύτερο κομμάτι της. Ακολουθεί ένα στιγμιότυπο (printscreen) απο το αντίστοιχο σημείο του κώδικα που γίνεται η τύπωση των συντεταγμένων στο αρχείο temp.obj. Στην εικόνα βλέπουμε στο πρώτο if, που ελέγχει αν το i ισούται με 1, οπου i εδώ είναι η κορυφή, να τυπώνει στο αρχείο τις συντεταγμένες της κάτω πλευράς, της βάσης δηλαδή, έως ότου το i γίνει 6. Όπως βλέπουμε χρειάστηκαν τρείς γραμμές συντεταγμένων για να κατασκευαστεί η κάτω πλευρά. Ακολουθεί ένας βρόγχος για i απο 7 έως 30 οπου είναι οι κορυφές που σχηματίζουν τις 6 πλαϊνές πλευρές του πρίσματος. Το κομμάτι που λείπει απο την εικόνα είναι αυτό που ελέγχει αν το i ισούται με 31, και τυπώνει στο αρχείο τις τρείς σειρές που θα κατασκευάσουν την πάνω πλευρά του πρίσματος. Συνολικά, στο αρχείο τυπώθηκαν δώδεκα γραμμές απο faces.

```
148     //faces section
149     int i=1;
150     while(i<vertsSUM-6){
151         if(i==1){
152             out.write("f");
153                 out.write(" "+(countOfVertices+i)+"//"+(countOfVertices+i++));
154                 out.write(" "+(countOfVertices+i)+"//"+(countOfVertices+i++));
155                 out.write(" "+(countOfVertices+i)+"//"+(countOfVertices+i));
156                 out.write(" "+(countOfVertices+i-1)+"//"+(countOfVertices+i-1));
157             out.write("\nf");
158                 out.write(" "+(countOfVertices+i)+"//"+(countOfVertices+i++));
159                 out.write(" "+(countOfVertices+i)+"//"+(countOfVertices+i++));
160                 out.write(" "+(countOfVertices+i)+"//"+(countOfVertices+i));
161                 out.write(" "+(countOfVertices+i-1)+"//"+(countOfVertices+i-1));
162             out.write("\nf");
163                 out.write(" "+(countOfVertices+i-1)+"//"+(countOfVertices+i-1));
164                 out.write(" "+(countOfVertices+i)+"//"+(countOfVertices+i++));
165                 out.write(" "+(countOfVertices+i)+"//"+(countOfVertices+i++));
166                 out.write(" "+(countOfVertices+i-2)+"//"+(countOfVertices+i-2));
167             out.write("\n");
168         }
169         if((i>6) && (i<31)){
170             out.write("f");
171                 out.write(" "+(countOfVertices+i)+"//"+(countOfVertices+i++));
172                 out.write(" "+(countOfVertices+i)+"//"+(countOfVertices+i++));
173                 out.write(" "+(countOfVertices+i)+"//"+(countOfVertices+i++));
174                 out.write(" "+(countOfVertices+i)+"//"+(countOfVertices+i++));
175             out.write("\n");
176         }
```

Εικόνα 4.6: στιγμιότυπο κώδικα java απο την εφαρμογή

4.2.10 Η επιφάνεια τριών σημείων

Ακολουθεί το δέκατο αντικείμενο, μια επιφάνεια που ορίζεται απο τρία σημεία, τα οποία παίρνουμε όπως είναι και τα τοποθετούμε στον πίνακα συντεταγμένων, και απο κει τα εκτυπώνουμε στο αρχείο temp.obj. Είναι το πιο σύντομο αντικείμενο, καθώς αποτελείται απο τρείς σειρές συντεταγμένων, και μία σειρά faces.

#vertices

v 50.6 51.5 50.3

v 53.0 51.2 50.3

v 53.6 51.95 51.8

#faces section

f 1//1 2//2 3//3

4.2.11 Η κάθετη επιφάνεια προβολής

Επόμενο αντικείμενο είναι η κάθετη επιφάνεια απο την επιφάνεια προβολής, η οποία ορίζεται απο πλάτος και ύψος, καθώς επίσης και απο το σημείο στο οποίο θα ακουμπάει η κάτω αριστερή κορυφή της. Μερικά προγράμματα αναπαράστασης αρχείων OBJ χρειάζονται 25 σημεία για να θεωρήσουν μια επιφάνεια επιφάνεια προβολής. Απο αυτά μόνο τα 4 χρειαζόμαστε εμείς, τα άλλα 21 μπορεί να δείχνουν ας πούμε όλα στο πρώτο σημείο, σε μία απο τις κορυφές. Ο κώδικας που τυπώνει τις γραμμές είναι απλός και περιλαμβάνει απλά έναν έλεγχο έτσι ώστε να τοποθετεί τις σωστές συντεταγμένες στις σειρές 1, 5, 20 και 25, ενώ σε όλες τις υπόλοιπες να βάζει ας πούμε τις συντεταγμένες του πρώτου σημείου. Τα faces είναι μόνο 2 σειρές, φυσικά γραμμένες χωρίς βρόγχο, πολύ απλές και αρκετές για να ενώσουν τις κορυφές σωστά.

4.2.12 Η οριζόντια επιφάνεια προβολής

Η οριζόντια επιφάνεια απο την επιφάνεια προβολής είναι ένα απλό παραλληλόγραμμο ορισμένο απο βάθος και πλάτος. Αυτό το αντικείμενο είναι όμοιο με το προηγούμενο, την κάθετη επιφάνεια, απλώς αλλάζουν 2 απο τα 4 σημεία του αντικειμένου. Το κείμενο που τυπώνεται στο αρχείο temp.obj για να κατασκευαστεί η επιφάνεια θα είναι πάλι, οι 25 συντεταγμένες απο τις οποίες μόνο οι 4 μας ενδιαφέρουν πραγματικά, τα texture vertices, τα vertices normals που είναι τοποθετημένα στοιχειωδώς και όμοια παντού, απλά και μόνο γιατι είναι απαραίτητα σε κάποια προγράμματα προσομοίωσης αρχείων .obj, και τέλος τα faces, που είναι υπεύθυνα να καθοδηγήσουν το πρόγραμμα να ενώσει τις κορυφές που του δόθηκαν. Όπως είπαμε και πρίν, στο συγκεκριμένο αντικείμενο χρησιμοποιούνται δύο σειρές faces μόνο.

#faces section
f 1//1 5//5 25//25 21//21
f 21//21 1//1 5//5 25//25

4.2.13 Η επιφάνεια προβολής

Το επόμενο και τελευταίο αντικείμενο είναι η επιφάνεια προβολής, που ορίζεται απο βάθος, ύψος, πλάτος και φυσικά το σημείο στο οποίο θα ακουμπάει η κάτω

αριστερά και πίσω κορυφή της επιφάνειας προβολής. Πρόκειται στην πραγματικότητα για τα δύο προηγούμενα αντικείμενα τοποθετημένα μαζί στη σκηνή. Μία κάθετη και μία οριζόντια επιφάνεια οι οποίες έχουν δύο σημεία κοινά και άρα τέμνονται κάθετα. Συνεπώς, το αντικείμενο αυτό θα έχει 50 σειρές συντεταγμένων να τυπώσει στο αρχείο .obj (απο τις οποίες και πάλι μόνο οι 8 θα έχουν σημασία), και μόνο 4 faces.

4.3 Λογισμικά που υποστηρίζουν το OBJ

- 3DPaintBrush (http://www.3dpaintbrush.com/)
- 3D Studio Max (http://usa.autodesk.com/adsk/servlet/pc/index?siteID=123112&id=13567410)
- Art of Illusion (http://aoi.sourceforge.net/)
- Autodesk Mudbox (http://usa.autodesk.com/adsk/servlet/pc/index?id=13565063&siteID=123112)
- Ayam (http://ayam.sourceforge.net/)
- Blender (http://www.blender.org/)
- CADdoctor (http://caddoctor.net/)
- Carrara
- Cheetah3D (http://www.cheetah3d.com/)
- Cinema 4D (http://en.wikipedia.org/wiki/Cinema_4D)
- CityEngine (http://www.procedural.com/cityengine/)
- Curvy3D (http://www.curvy3d.com/)
- DAZ Studio (http://www.daz3d.com/)
- EnSight (CEI) (http://www.ensight.com/)
- Feature Manipulation Engine (http://en.wikipedia.org/wiki/Feature_Manipulation_Engine)
- GLC Player (http://www.glc-player.net/)
- Google Skethup (http://sketchup.google.com/)
- Hexagon (http://www.daz3d.com/i.x/software/hexagon)

- Irrlicht Engine (http://irrlicht.sourceforge.net/)
- Lightwave (http://www.newtek.com/lightwave/)
- Mathematica (http://www.wolfram.com/)
- Maya (http://usa.autodesk.com/adsk/servlet/pc/index?siteID=123112&id=13577897)
- modo (http://en.wikipedia.org/wiki/Modo_%28software%29)
- MeshLab (http://meshlab.sourceforge.net/)
- Misfit Model 3D (http://www.misfitcode.com/misfitmodel3d/)
- Open Cobalt (http://www.duke.edu/~julian/Cobalt/Home.html)
- Open CTM
- Poser (http://en.wikipedia.org/wiki/Poser)
- Rhinoceros 3D (http://www.rhino3d.com/)
- SideFX Houdini (http://www.sidefx.com/)
- Softimage XSI (http://usa.autodesk.com/adsk/servlet/pc/index?siteID=123112&id=13571168)
- Sweet Home 3D (http://www.sweethome3d.eu/index.jsp)
- TransMagic (http://www.transmagic.com/)
- VisIt (https://wci.llnl.gov/codes/visit/)
- VXL (through the contrib/brl/bbas/imesh sub-library)
- Wings 3D (http://www.wings3d.com/)
- X-Plane Plane Maker (http://www.x-plane.com/)
- Zbrush (http://www.pixologic.com/home.php)

ΣΥΝΟΨΗ

Ο αρχικός στόχος, η κατασκευή δηλαδή μιας εφαρμογής η οποία θα διευκόλυνε το χρήστη στη δημιουργία ενός συνόλου αντικειμένων σε μορφότυπο .obj, επετεύχθη. Και με το παραπάνω φυσικά αφου μπορεί κάνοντας μόνο κάποια κλικ να δημιουργήσει τεράστια κείμενα τύπου OBJ σε λίγα μόνο λεπτά. Τα προβλήματα που συναντήθηκαν επιλύθηκαν, με αποτέλεσμα η εφαρμογή να μην εμφανίζει ποτέ λάθη εκτέλεσης.

Βελτιώσεις που θα μπορούσαν να μελετηθούν στο μέλλον θα ήταν καταρχήν η επεξεργασία υφής και επιφάνειας ή χρώματος του κάθε αντικειμένου, κάτι το οποίο δεν έχει προβλεφθεί στη συγκεκριμένη εφαρμογή.

Ένα άλλο πολύ σημαντικό θέμα μετά τη δημιουργία κάποιων αντικειμένων και την εμφάνισή τους στη σκηνή είναι το εξής. Για την ώρα, ο χρήστης μπορεί να επιλέξει το σύνολο των αντικειμένων και να το περιστρέψει ή μετακινήσει. Στην πραγματικότητα όμως μετακινεί την κάμερα. Άρα μία πολύ σημαντική βελτίωση θα ήταν να μπορεί ο χρήστης ακόμα και μετά την δημιουργία του αντικειμένου, να επιλέξει ένα συγκεκριμένο αντικείμενο απο τη σκηνή και να το μετακινήσει ή να το επεξεργαστεί.

Άλλη μία βελτίωση θα ήταν να προσθέσουμε και άλλα αντικείμενα πιο πολύπλοκα ίσως, καμπύλες γραμμές και επιφάνειες. Επίσης μια λειτουργία undo θα ήταν πολύ χρήσιμη όταν αλλάζουμε γνώμη για ένα αντικείμενο που προσθέσαμε, έτσι ώστε με μία κίνηση να γυρίσουμε στην προηγούμενη κατάσταση, αντί να καθαρίσουμε τη σκηνή και να ξεκινήσουμε απο την αρχή. Σε γενικές γραμμές είναι μια εφαρμογή αρκετά επεξεργασμένη και ικανοποιητική με κάποιες προοπτικές βελτίωσης.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. http://en.wikipedia.org/wiki/3D_computer_graphics
2. http://www.webreference.com/3d/glossary/
3. http://en.wikipedia.org/wiki/Java_(programming_language)
4. http://java.sun.com/javase/6/docs/api/
5. http://forums.sun.com/index.jspa
6. http://leepoint.net/notes-java/index.html
7. http://zetcode.com/tutorials/javaswingtutorial/
8. http://www.engin.umd.umich.edu/CIS/course.des/cis400/java/java.html
9. http://el.wikipedia.org/wiki/Java
10. http://java.sun.com/developer/onlineTraining/java3d/
11. http://en.wikipedia.org/wiki/Java_3d
12. http://www.java3d.org/
13. http://java.sun.com/javase/technologies/desktop/java3d/forDevelopers/J3D_1_3_API/j3dapi/index.html
14. https://java3d.dev.java.net/
15. http://portal.acm.org/citation.cfm?id=369340.369288
16. http://java.sun.com/developer/onlineTraining/java3d/
17. http://en.wikipedia.org/wiki/Obj
18. http://www.martinreddy.net/gfx/3d/OBJ.spec
19. http://www.eg-models.de/formats/Format_Obj.html
20. http://www.royriggs.com/obj.html
21. http://people.sc.fsu.edu/~burkardt/data/obj/obj.html
22. http://en.wikipedia.org/wiki/3D_computer_graphics
23. http://www.webreference.com/3d/glossary/
24. http://en.wikipedia.org/wiki/Cartesian_coordinate_system#Three-dimensional_coordinate_system
25. http://en.wikipedia.org/wiki/3D_computer_graphics_software
26. http://en.wikipedia.org/wiki/JAR_%28file_format%29

27. http://www.ipet.gr/digitech2/
28. http://www.medialab.ntua.gr/education/ComputerGraphics/OpenGL_Lectures/01-Introduction.pdf
29. http://en.wikipedia.org/wiki/Direct3D
30. http://en.wikipedia.org/wiki/Opengl
31. http://www.renderware.com/
32. http://en.wikipedia.org/wiki/List_of_3D_graphics_APIs