Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

ΠΛΗΡΟΦΟΡΙΚΗ

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	Ευφυής Αλληλεπίδραση Java-Prolog
Ονοματεπώνυμο Φοιτητή	Μανωλάς Εμμανουήλ
Πατρώνυμο	Μηνάς Μανωλάς
Αριθμός Μητρώου	ΜΠΠΛ 08059
Επιβλέπων	Παναγιωτόπουλος Θεμιστοκλής, Καθηγητής

Ημερομηνία Παράδοσης Οκτώβριος 2012

Τριμελής Εξεταστική Επιτροπή		
(Υπογραφή)	(Υπογραφή)	(Υπογραφή)
Παναγιωτόπουλος Θεμιστοκλής	Φούντας Ευάγγελος	Παπαδάκης Ιωάννης
Καθηγητής	Καθηγητής	Καθηγητής

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Κατάλογος Προγραμμάτων

Περίληψη

Περίληψη

Στην παρούσα διπλωματική εργασία παρουσιάζεται η προσπάθεια απεικόνισης και κατανόησης ενός τρισδιάστατου περιβάλλοντος με τις σύγχρονες τεχνολογίες. Οι τεχνολογίες που χρησιμοποιήθηκαν κατά τη διάρκεια της εκπόνησης είναι η Java 3D, Prolog και φυσικά δεν θα μπορούσε να λειπει και το διαδίκτυο. Αφορά την συνεργασία των τεχνολογιών αυτών και επιδιώκει να παρουσιάσει ένα τρισδιάστατο περιβάλλον με μια γλώσσα Τεχνητής Νοημοσύνης. Τα εργαλεία που επιλέχθηκαν είναι μια Java 3D engine - Java Monkey Engine και η SICstus Prolog. Ο χρήστης μπορεί να δει το περιβάλλον ενός τρισδιάστατου χαρακτήρα και έχει τη δυνατότητα να βρίσκει παραγώγους μέσω της Prolog. Η εργασία σίγουρα επιδέχεται επέκταση και αυτό είναι ένα από τα θετικά.
Θα ήθελα να ευχαριστήσω τον καθηγήτη μου Θ.Παναγιωτόπουλο που μου έδωσε την ευκαιρία να ασχοληθώ με ένα θέμα που με ενδιέφερε.Θα ήθελα να ευχαριστήσω την οικογένεια μου, τη Νέλλη και όλους τους φίλους για την σημαντική και πολύτιμη υποστήριξη που μου παρείχαν κατά την διάρκεια της εργασίας αυτής.
Ευχαριστώ Πολύ.

My Diploma Thesis was an opportunity to try to undrestand and illustrate a 3D environment,using the contemporary technologies.The technologies used during the preparation of this project are Java 3D,Prolog and Internet,of course.It refers to the collaboration of these technologies and pursues to present a 3D environment through an Artificial Intelligence language.The chosen tools are a Java 3D engine- Java Monkey engine and SICstus Prolog.The user can see a 3D character and has opportunity to find the derivatives of an equation through Prolog.This essay can certainly be evolved,and this is a positive aspect. I would like to thank my professor Th. Panagiotopoulos for giving me the opportunity to cope with an issue that concerned me.I would like to thank my family,Nelly and all my friends for the significant support that they gave me during my project.
Thank you very much.

Κεφάλαιο 1

Τρισδιάστατο Περιβάλλον

1.1 Java 3D

Η Java στα γενικότερα πλαίσια ανάπτυξης και υποστήριξης νέων τεχνολογιών κινήθηκε προς τη δημιουργία επιπλέον βιβλιοθηκών για τον εμπλουτισμό των εφαρμογών που μπορούν να αναπτυχθούν με την τεχνολογία Java. Στα πλαίσια αυτά δημιούργησε μια οικογένεια βιβλιοθηκών που λέγεται Mediaproduct και περιέχει βιβλιοθήκες για δισδιάστατα γραφικά, ομιλία, animation, collaboration API και το Java 3D API. Οπότε ουσιαστικά το Java 3D αποτελεί ένα σύνολο εντολών υψηλού επιπέδου που βρίσκονται σε πλήρη συμφωνία και συνεργασία με τα υπόλοιπα πακέτα που διαθέτει η Java.

1.2 Τι είναι η Java3D

Η Java 3D είναι τμήμα της Java και αποτελεί ένα API (Application Programming Interface) για τη δημιουργία εφαρμογών και applet τρισδιάστατων γραφικών. Παρέχει στους προγραμματιστές εργαλεία υψηλού επιπέδου για τη δημιουργία και διαχείριση τρισδιάστατης γεωμετρίας καθώς και για την κατασκευή δομών που χρησιμοποιούνται για την φωτοαπόδοση (rendering) της γεωμετρίας αυτής. Η χρήση των παρεχόμενων εργαλείων δημιουργίας μπορεί να αξιοποιηθεί για την δημιουργία μεγάλων και σύνθετων εικονικών κόσμων, ενώ η Java 3D υποστηρίζει την αποδοτική απεικόνισή τους.

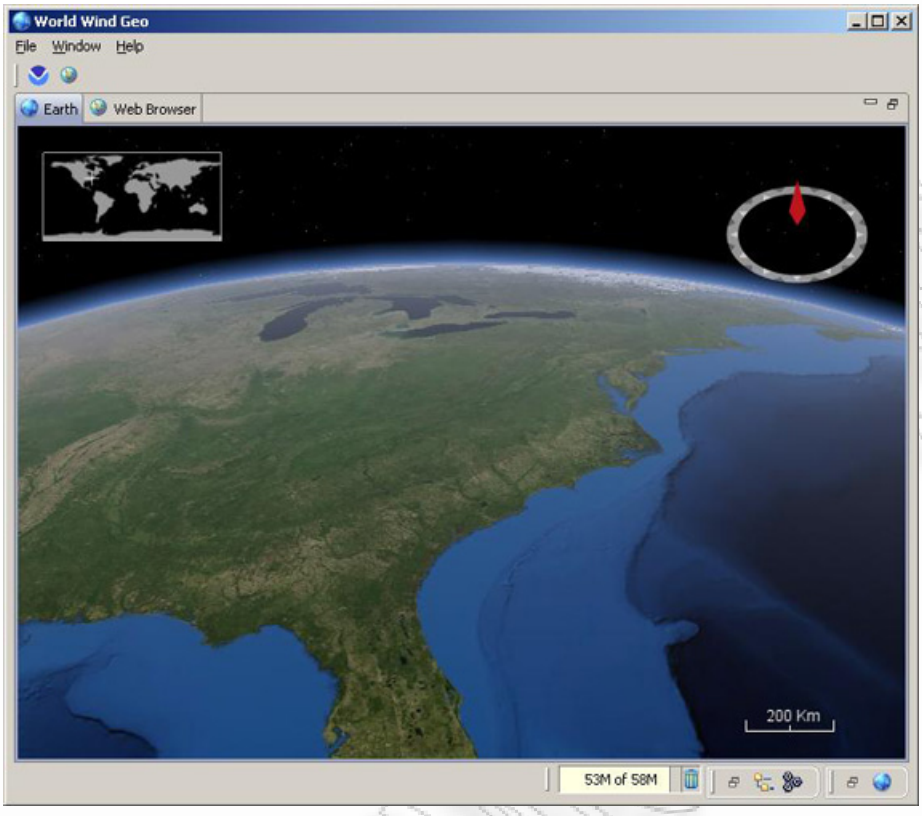
Η βιβλιοθήκη βασίζεται στην Java κληρονομώντας όλα τα σχετικά πλεονεκτήματα που παρέχει αυτή η γλώσσα. Ειδικότερα, η Java 3D αποτελεί τμήμα των JavaMedia suite APIs, καθιστώντας την διαθέσιμη για ένα μεγάλο πλήθος από πλατφόρμες. Επίσης μπορεί να χρησιμοποιηθεί σε δικτυακές εφαρμογές επειδή οι εφαρμογές και τα applets που αναπτύσσονται με το Java 3D API έχουν πρόσβαση σε όλο το σύνολο των κλάσεων της Java.

Η υλοποίηση του Java 3D API βασίζεται σε ιδέες υπαρχόντων APIs γραφικών καθώς και σε καινούριες τεχνολογίες. Οι χαμηλού επιπέδου δομές γραφικών του Java 3D συνδυάζουν τις καλύτερες ιδέες από τις τεχνολογίες Direct3D, OpenGL, QuickDraw3D και XGL. Επίσης οι υψηλού επιπέδου δομές συνδυάζουν τις καλύτερες ιδέες από αρκετά συστήματα γραφικής απεικόνισης σκηνών. Η Java 3D παρουσιάζει ορισμένες έννοιες που συνήθως δεν αποτελούν τμήμα ενός περιβάλλοντος γραφικών, όπως ο τρισδιάστατος χωρικός ήχος (3D spatial sound). Οι δυνατότητες του API αναφορικά με τον ήχο βοηθούν στη δημιουργία πιο ευέλικτων εφαρμογών αναφορικά με τις δυνατότητες που αντιλαμβάνεται ο χρήστης τους.

Ο σχεδιασμός του Java 3D έγινε με κύριο γνώμονα την καλύτερη απόδοση. Αρκετές σχεδιαστικές αποφάσεις έγιναν ώστε οι υλοποιήσεις του συγκεκριμένου API να παρέχουν υψηλό επίπεδο απόδοσης στους τελικούς χρήστες των εφαρμογών. Ειδικότερα, σε περιπτώσεις που απαιτούνταν σχεδιαστικές αποφάσεις οι επιλογές βασίστηκαν κυρίως στη ταχύτητα χρόνου εκτέλεσης. Άλλοι σημαντικοί στόχοι του Java 3D API[i] περιλαμβάνουν:

- Εκτενές σύνολο δυνατοτήτων για τη δημιουργία τρισδιάστατων κόσμων, ξεκαθαρίζοντας τα κρίσιμα από τα μη κρίσιμα χαρακτηριστικά. Όσες δυνατότητες μπορούν να υποστηριχθούν σε ανώτερο επίπεδο API δεν περιλαμβάνονται στη Java 3D.
- Σύνολο παραδειγμάτων αντικειμενοστραφούς προγραμματισμού υψηλού επιπέδου.

[i] https://java3d.dev.java.net/



Σχήμα 1.1: Java 3D

- Υποστήριξη runtime loaders. Με αυτό το τρόπο η Java 3D εξυπηρετεί ένα μεγάλο εύρος διαφορετικών file formats όπως συγκεκριμένου κατασκευαστή CAD formats, interchange formats και VRLM97[ii].

1.3 Περιγραφή

Το Java 3D API είναι αντικειμενοστραφές. Οι εφαρμογές δημιουργούν ξεχωριστά τμήματα γραφικών ως διακριτά αντικείμενα τα οποία διασυνδέονται σε μια δενδρική δομή που ονομάζεται γράφος σκηνής. Η εφαρμογή διαχειρίζεται αυτά τα αντικείμενα χρησιμοποιώντας τις προκαθορισμένες μεθόδους πρόσβασης, μετατροπής και διασύνδεσης συνδέσμων (accessor, mutator, node-linking methods). Βασική κατεύθυνση του Java 3D είναι να επικεντρώσει τις προγραμματιστικές προσπάθειες σε γεωμετρικά αντικείμενα αντί για τρίγωνα και στη σκηνή και τη σύνθεσή της αντί για την διαδικασία της αποδοτικής φωτοαπόδοσης της σκηνής.

Ουσιαστικά ο γράφος σκηνής που παράγεται είναι υπεύθυνος για την όλη λειτουργία της εφαρμογής. Μέσω του γράφου αυτού παρέχεται ένας ευέλικτος μηχανισμός για αναπαράσταση και φωτοαπόδοση της σκηνής. Ειδικότερα, ο γράφος περιέχει γεωμετρικά δεδομένα, τις πληροφορίες χαρακτηριστικών (attribute information) και πληροφορίες επισκόπησης που απαιτούνται για τη φωτοαπόδοση της σκηνής από συγκεκριμένη άποψη της κάμερας (point of view).

Το σύνολο των παρεχόμενων δυνατοτήτων της τεχνολογίας Java 3D περιλαμβάνει έλεγχο σχετικά με το σχήμα, το χρώμα και τη διαφάνεια ενός αντικειμένου. Επίσης διαδικασίες για τον έλεγχο εφέ και φωτισμού του background καθώς και εφέ περιβάλλοντος όπως ομίχλη. Για τα αντικείμενα επιτρέπεται επίσης έλεγχος σχετικά με μετακίνηση, περιστροφή και μεταβολή του μεγέθους του καθώς και μορφοποίηση στο χρόνο. Τέλος παρέχεται πλήρης έλεγχος μέσω διαδικασιών για την εκτέλεση συγκεκριμένου κώδικα και συγκεκριμένου ήχου όπως και έλεγχος του ήχου με το χρόνο.

Η τεχνολογία Java 3D παρέχει και ορισμένα επιπλέον στοιχεία, τα οποία ονομάζει συνδέσμους (nodes), για την αλληλεπίδραση με το χρήστη και την δημιουργία “αισθητήρων” για την αναγνώριση κίνησης ή άλλης μεταβολής στο χώρο της σκηνής. Ειδικότερα ορίζονται οι εξής τύποι αντικειμένων:

[ii] ISO/IEC 14772-1:1997

- **Behavior object**, το οποίο δέχεται καταστάσεις ενεργοποίησης και απενεργοποίησης εκτελώντας αντίστοιχο κύκλο ενεργειών. Η ακριβής δράση που πραγματοποιείται μπορεί να είναι οποιοδήποτε τμήμα κώδικα Java, επομένως και Java 3D. Οι διαθέσιμοι τύποι behavior object είναι Interpolator Behavior, Level-of-Detail Behavior και Billboard Behavior.
- **Sensor object**, αποτελεί ένα αισθητήρα ο οποίος ανιχνεύει ένα σύνολο εισόδων σχετικά με ορισμένες προδιαγραφές, εάν αυτές πληρούνται τότε ο αισθητήρας ενεργοποιείται ή απενεργοποιείται αντίστοιχα.

Το API της Java 3D υποστηρίζει τρεις διαφορετικούς τύπους φωτοαπόδοσης, ανάλογα με τις δυνατότητες που παρέχονται στους προγραμματιστές σε κάθε περίπτωση καθώς και σύμφωνα με τον απαιτούμενο χρόνο για την ολοκλήρωση της διαδικασίας φωτοαπόδοσης. Έτσι ορίζονται οι παρακάτω επιλογές rendering:

- **Immediate Mode**, το οποίο επιτρέπει μεγαλύτερη ευελιξία αλλά υστερεί σε χρόνο φωτοαπόδοσης. Στη συγκεκριμένη περίπτωση ο προγραμματιστής μπορεί να ορίσει επακριβώς γεωμετρία ή να χρησιμοποιήσει τμήματα από το γράφο σκηνής που θέλει να κάνει render. Η επιλογή αυτή δεν εκμεταλλεύεται καθόλου το δενδρικό γράφο σκηνής.
- **Retained Mode**, παρέχοντας παρόμοια ευελιξία στη διαχείριση και ελαφρώς καλύτερη ταχύτητα rendering. Παρέχεται πλήρης έλεγχος το γράφου σκηνής στο προγραμματιστή επιτρέποντας την επέκταση (create node) ή σμίκρυνση (delete node) του γράφου σκηνής παρατηρώντας ταυτόχρονα το αποτέλεσμα. Το API σε αυτή την περίπτωση κάνει επιπλέον βελτιστοποιήσεις στη σκηνή για να βελτιώσει την ταχύτητα φωτοαπόδοσης.
- **Compiled-Retained Mode**, στην περίπτωση αυτή εκτελούνται εκτενώς αλγόριθμοι για την μέγιστη βελτιστοποίηση του αποτελέσματος και την επιτάχυνση της φωτοαπόδοσης. Αυτό επιτυγχάνεται μεταγλωττίζοντας τμήμα ή ολόκληρο το γράφο σκηνής σε Java 3D.

1.4 Επισκόπηση Τεχνολογίας

Σύμφωνα με την παραπάνω περιγραφή είναι φανερό ότι σκοπός της τεχνολογίας Java 3D είναι να παρέχει ένα σύνολο βιβλιοθηκών υψηλού επιπέδου για τρισδιάστατα γραφικά. Η λογική ανάπτυξης του Java 3D είναι αρκετά ανοιχτή επιτρέποντας μεγάλο εύρος εφαρμογών που να αξιοποιούν τόσο της Java 3D όσο και τις γενικότερες Java βιβλιοθήκες. Επίσης η δυνατότητα για συνεργασία με άλλες τεχνολογίες και πρότυπα καθιστά ακόμα πιο εύκολη την ανάπτυξη εφαρμογών για τους προγραμματιστές, στην περίπτωση της υποστηριζόμενης τεχνολογίας ανήκει το πρότυπο VRML97. Όπως υποστηρίζεται στον ιστοτόπο της Java το API γραφικών μπορεί να χρησιμοποιηθεί σε εφαρμογές CAD, για δικτυακές εφαρμογές, για παιχνίδια, για Data Visualizations κ.α. Οι υλοποιήσεις που έχουν πραγματοποιηθεί με την τεχνολογία εκτείνονται περισσότερο προς εφαρμογές CAD, Data Visualizations και δικτυακές εφαρμογές. Η Sun Microsystems, η οποία βρίσκεται πίσω από την ανάπτυξη της Java, ήταν από τις πρώτες εταιρείες που υλοποίησαν 3D Desktop χρησιμοποιώντας Java και Java 3D για το Visualization. Παρά το γεγονός ότι υπάρχουν αρκετές χρήσεις Java 3D σε εφαρμογές, κυρίως visualizations επιστημονικών δεδομένων, στις περισσότερες περιπτώσεις τα προγράμματα αυτά είναι αρκετά παλιά.

1.5 VRML

Η τεχνολογία VRML, που αποτελεί ακρωνύμιο του Virtual Reality Modelling Language, αποτελεί την πρώτη προσπάθεια για δημιουργία ενός τρισδιάστατου προτύπου για το δίκτυο. Η όλη ιδέα ξεκίνησε στο πρώτο συνέδριο για το διαδίκτυο που πραγματοποιήθηκε στο κέντρο CERN της Γενεύης όπου εισήχθη η ιδέα της VRML ως προτύπου για τρισδιάστατο διαδίκτυο. Αυτές οι δηλώσεις λειτούργησαν ως καλό κίνητρο για τη δημιουργία μιας ομάδας εργασίας η οποία παρουσίασε τελικά τη VRML. Η πορεία εξέλιξης του προτύπου ξεκινάει από μια ομάδα μηχανικών και καλλιτεχνών που δημιουργούν την www-vrml mailing list. Αρχικά η ομάδα δημιουργεί το VRML 1 το οποίο παρουσιάζει περιορισμένες δυνατότητες για αλληλεπίδραση και κίνηση (animation), επιτρέποντας τη δημιουργία στατικών εικονικών κόσμων. Παρατηρώντας αυτές τις ελλείψεις οι προδιαγραφές αναθεωρούνται και δημιουργείται η δεύτερη έκδοσή του όπου και γίνεται πρότυπο το 1997 (ISO/IEC 14772-1:1997), έκτοτε καλείται VRML97. Με την VRML97 παρέχεται η δυνατότητα για καθορισμό τρισδιάστατων δυναμικών σκηνών όπου ο χρήστης μπορεί να

πλοηγηθεί μέσω ενός VRML browser. Οι εικονικές σκηνές που δημιουργούνται με VRML μπορούν να διανεμηθούν μέσω του διαδικτύου και κάθε χρήστης μπορεί να πλοηγηθεί με τη χρήση συγκεκριμένων VRML browsers, που συνήθως αποτελούν plug-in για web browsers. Η VRML είναι προσαρμοσμένη για δικτυακή χρήση επομένως επιτρέπει τη διασύνδεση διαφορετικών αρχείων τόσο τύπου VRML όσο και άλλα WWW formats μεταξύ τους μέσω URLs. Η βάση ανάπτυξης της VRML υπήρξε η HTML επομένως η γλώσσα ακολουθεί τις ίδιες ανοικτές αρχές ως προς τον ορισμό των αντικειμένων που διαχειρίζεται. Ειδικότερα οι αρχές που ακολουθήθηκαν κατά το σχεδιασμό της αφορούν:

- **Authorability**, παρέχοντας δυνατότητες τόσο για ανάπτυξη προγραμμάτων ικανών για δημιουργία, μορφοποίηση και συντήρηση αρχείων τύπου VRML όσο και για την δημιουργία προγραμμάτων μετάφρασης (translation) άλλων συνηθισμένων αρχείων τρισδιάστατων γραφικών σε αρχεία τύπου VRML.
- **Συνταξιμότητα (Composability)**, δίνοντας τις δυνατότητες για χρήση και συνδυασμό δυναμικών τρισδιάστατων αντικειμένων σε VRML κόσμους, επομένως προσφέροντας και τη δυνατότητα επανα-χρησιμοποίησης των αντικειμένων αυτών.
- **Επεκτασιμότητα (Extensibility)**, δυνατότητα για προσθήκη καινούργιων τύπων αντικειμένων που δεν προδιαγράφονται από τη VRML.
- Δυνατότητα ανάπτυξης σε μεγάλη ποικιλία διαφορετικών συστημάτων.
- **Απόδοση (Performance)**, δίνοντας έμφαση στη διαβαθμίσιμη, αυτόματη ρύθμιση της απόδοσης σε μεγάλο αριθμό υπολογιστικών συστημάτων.
- **Διαβαθμισιμότητα (Scalability)**, επιτρέποντας τη δημιουργία απεριόριστα μεγάλων δυναμικών τρισδιάστατων κόσμων.

Η VRML παρέχει ένα μοντέλο επεκτασιμότητας που επιτρέπει τον ορισμό νέων δυναμικών τρισδιάστατων αντικειμένων. Αυτό το μοντέλο μπορεί να αξιοποιηθεί από την κοινότητα ανάπτυξης εφαρμογών ώστε να δημιουργηθούν επεκτάσεις στο βασικό πρότυπο που εκτελούνται σε ποικιλία συστημάτων.

1.6 Πρότυπο VRML

Η VRML αποτελεί ένα τύπο αρχείου (file format) για την περιγραφή τρισδιάστατων αντικειμένων και κόσμων. Ο σχεδιασμός της έγινε με γνώμονα τη χρήση στο internet, σε intranets καθώς και σε τοπικά συστήματα. Ακολουθεί μια σύντομη περιγραφή των περιεχομένων ενός αρχείου τύπου VRML όπως αυτή δίνεται για το VRML97 standard. Κάθε αρχείο VRML αποτελεί ένα τρισδιάστατο χώρο ορισμένο στο χρόνο που περιέχει γραφικά και ακουστικά αντικείμενα που μπορούν να μορφοποιηθούν δυναμικά μέσα από μια ποικιλία μηχανισμών. Το πρότυπο είναι ανεξάρτητο από συγκεκριμένες υλοποιήσεις (π.χ. ανάλυση οθόνης και συσκευές εισόδου). Επίσης το πρότυπο απευθύνεται σε μεγάλο εύρος συσκευών και εφαρμογών, παρέχοντας μεγάλο εύρος μεταφράσεων και υλοποιήσεων των λειτουργιών που περιέχει. Κάθε VRML αρχείο:

- Ορίζει ένα σύστημα συντεταγμένων χώρου για όλα τα αντικείμενα που περιέχονται στο αρχείο, καθώς και τα αντικείμενα που γίνονται include στο αρχείο
- Ορίζει και συνθέτει ένα σύνολο από τρισδιάστατα και πολυμεσικά αντικείμενα
- Καθορίζει υπερσυνδέσμους προς άλλα αρχεία και εφαρμογές
- Μπορεί να ορίσει συμπεριφορές αντικειμένων

Ένα σημαντικό χαρακτηριστικό του VRML αρχείου είναι η δυνατότητα σύνδεσης αρχείων μέσω συμπερίληψης (διαδικασία inclusion) καθώς και συσχέτισης αρχείων μέσω υπερσυνδέσμων. Η ιεραρχική συμπερίληψη αρχείων επιτρέπει τη δημιουργία αυθαίρετα μεγάλων και δυναμικών κόσμων. Επομένως η VRML διασφαλίζει ότι κάθε αρχείο περιγράφεται πλήρως από τα αρχεία που περιέχει. Ένα άλλο σημαντικό χαρακτηριστικό της VRML είναι ότι σχεδιάστηκε ώστε να χρησιμοποιείται σε κατανεμημένα περιβάλλοντα όπως το διαδίκτυο. Υπάρχει μια ποικιλία αντικειμένων και μηχανισμών που περιλαμβάνει η γλώσσα και υποστηρίζουν πολλαπλά κατανεμημένα αρχεία όπως:

- In-lining άλλων αρχείων VRML, όπου ουσιαστικά ο τρισδιάστατος κόσμος που περιγράφει ένα αρχείο συμπεριλαμβάνεται στο τρισδιάστατο κόσμο του άλλου αρχείου.
- Διασύνδεση αρχείων μέσω υπερσυνδέσμου
- Υποστήριξη καθιερωμένων δικτυακών και ISO προτύπων τύπων αρχείου
- Ορισμός σύντομης σύνταξης.

Η λογική λειτουργίας του προτύπου βασίζεται στη πλέον διαδεδομένη γλώσσα HTML, δηλαδή σε κόμβους (nodes). Η όλη δημιουργία της τρισδιάστατης σκηνής αποτελεί το γράφο σκηνής (scene graph), που ουσιαστικά αποτελείται από μια δομή κόμβων που είτε είναι σειριακοί είτε εμφωλευμένοι ο ένας μέσα στον άλλο, συνήθως είναι ένας συνδυασμός αυτών των δύο. Ο βασικός περιορισμός σύνταξης αφορά στην έναρξη του αρχείου VRML όπου δηλώνεται ότι το συγκεκριμένο αρχείο αποτελεί αρχείο τύπου VRML(η αρχική γραμμή ενός VRML αρχείου είναι η VRML V2.0 utf8). Το πρότυπο εκτός από τον ορισμό του VRML file format ορίζει και τον τρόπο μετάφρασης των αρχείων VRML από browsers για το συγκεκριμένο format διευκολύνοντας και παρακινώντας την ανάπτυξη VRML browsers.

Κεφάλαιο 2

Εισαγωγή

2.1 Πρόλογος

Η εργασία αυτή στοχεύει στην δημιουργία ένας τρισδιάστατου κόσμου αλληλεπίδρασης με ιδιότητες φυσικής (βαρύτητα, συγκρούσεις, τριβή κτλ.) γραμμένο σε Java. Για να αναπτυχθεί μία τέτοιου είδους εφαρμογή εκτός από τον compiler της java ήταν απαραίτητη και η χρήση μιας 3D μηχανής, το Java Monkey Engine[i]. Για την εφαρμογή αυτή χρησιμοποιήθηκε η εκδόση **Java 1.6** και το IDE **NetBeans 6.9** [ii] Επίσης κάναμε εκτενή χρήση και άλλων βιβλιοθηκών όπως η **LWGJL** [iii], η **ODE** και η **odejava** στις οποίες θα αναφερθούμε πιο αναλυτικά στην συνέχεια.

2.2 Παρουσίαση του jME

Σχήμα 2.1: Java Monkey Engine

Το jME (jMonkey Engine) είναι μια open source μηχανή υψηλής απόδοσης γραφικών. Ένα μεγάλο μέρος της έμπνευσης για το jME προέρχεται από το βιβλίο 3D Game Engine Design του David Eberly. Το jME φτιάχτηκε για να καλύψει την έλλειψη μιας πλήρους μηχανής γραφικής αναπαράστασης γραμμένης σε java. Χρησιμοποιώντας ένα στρώμα αφαίρεσης, επιτρέπει σε οποιοδήποτε σύστημα απόδοσης (rendering system) να συνδεθεί. Αυτήν την περίοδο, υποστηρίζεται η LWJGL με προοπτική για την υποστήριξη της JOGL στο εγγύς μέλλον. Το jME δημιουργήθηκε από τον Mark Powell το 2003 ενώ ερευνούσε το rendering του OpenGL. Όταν ανακάλυψε το LWJGL αποφάσισε ότι η java (που ήταν η γλώσσα της επιλογής του) θα ήταν τέλεια για τα δικά του εργαλεία γραφικής παράστασης. Αυτά τα εργαλεία σύντομα εξελίχθηκαν σε μια πρωτόγονη μηχανή. Μετά την ανάγνωση του βιβλίου του David Eberly, 3D 3Game Engine Design πραγματοποίησε μια αρχιτεκτονική γραφικών παραστάσεων σκηνής. Τότε ήταν, που και το jME έγινε μέρος του λογισμικού της Sun Java.net Σύντομα προστέθηκαν πολλοί στην προσπάθεια για την ενίσχυση των δυνατοτήτων του jME. Από τότε τείνει να καλύψει πολλά προηγμένα και σύγχρονα χαρακτηριστικά γνωρίσματα γραφικής παράστασης και έχει εξελιχθεί πλέον σε μια σταθερή πλατφόρμα για την ανάπτυξη παιχνιδιών. Ο Joshua Slack προστέθηκε στο δυναμικό του JME στο τέλος του 2003 και έγινε βασικό μέλος και ένα αναπόσπαστο κομμάτι της ομάδας του jME.

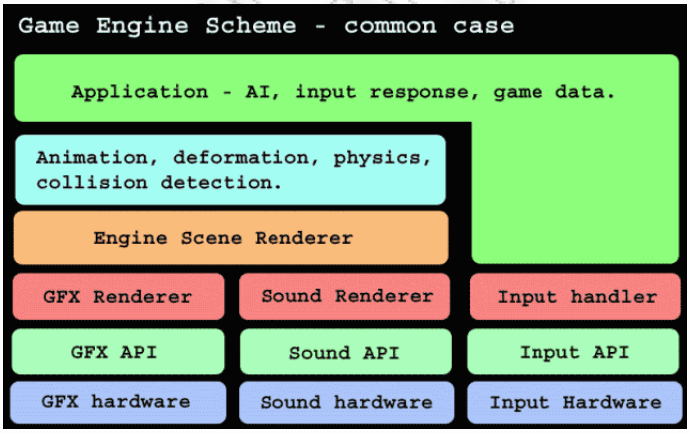
2.3 Μηχανές Παιχνιδιών

Τι είναι όμως μια μηχανή παιχνιδιών; Μια μηχανή παιχνιδιών είναι ο πυρήνας ενός τμήματος λογισμικού από ένα video game ή μιας οποιασδήποτε άλλης εφαρμογής αλληλεπίδρασης πραγματικού χρόνου. Παρέχει τεχνολογίες οι οποίες απλοποιούν την δημιουργία και ανάπτυξη ενός παιχνιδιού και συχνά επιτρέπει στις εφαρμογές μας, να έχουν την δυνατότητα να τρέχουν σε διαφορετικές πλατφόρμες, όπως οι κονσόλες παιχνιδιών ή τα λειτουργικά συστήματα των ηλεκτρονικών υπολογιστών Windows, Linux, Mac. Οι κύριες λειτουργίες που παρέχονται χαρακτηριστικά από μια μηχανή παιχνιδιών, είναι μια μηχανή

[i] http://www.jmonkeyengine.org
[ii] http://www.netbeans.org
[iii] http://www.lwjgl.org/

απόδοσης (renderer) για δύο ή τριών διαστάσεων γραφική αναπαράσταση, μια μηχανή φυσικής ή ανίχνευσης σύγκρουσης, ήχος, τεχνητή νοημοσύνη, διαχείριση μνήμης, scripting, animation, networking, streaming, και μια σκηνή γραφικής αναπαράστασης. Η βάση για κάθε μηχανή 3D βασίζεται στην μετατροπή των τριών διαστάσεων σε δυο. Η οθόνη του υπολογιστή μας είναι δύο διαστάσεων, πλάτους και ύψους. Με την βοήθεια πολλών μαθηματικών συναρτήσεων οι δημιουργοί των 3D μηχανών κατάφεραν να κάνουν αυτήν την μετατροπή εφικτή. Οι μηχανές αυτές εστιάζουν εκτός από το να μας δείχνουν πολύγωνα και στην ταχύτητα. Για να αντισταθμιστούν οι χιλιάδες μαθηματικές πράξεις που πρέπει να γίνονται, αν υπολογιστεί ότι κάτι δεν θα είναι ορατό από κάποιο συγκεκριμένο σημείο, οι μηχανές παραβλέπουν αυτούς τους υπολογισμούς. Η διαδικασία αυτή έχει πάρει διάφορες ονομασίες, μια από αυτές είναι η ”backface culling”. Μια φόρμουλα η οποία είναι ευρέως διαδεδομένη στις περισσότερες μηχανές, είναι να χρησιμοποιείται από τους δημιουργούς των μηχανών ένα άπειρο επίπεδο για να ορίσει την ορατότητα αντικειμένων. Αν η φόρμουλα επιστρέψει την τιμή -1 τότε η μπροστινή πλευρά των αντικειμένων δεν φαίνεται. Βλέπουμε λοιπόν ότι ακόμα και όταν παραβλέπονται αντικείμενα πάλι οι μηχανές χρησιμοποιούν τα μαθηματικά. Η φυσική είναι επίσης ζωτικής σημασίας στα παιχνίδια. Οι μηχανές παιχνιδιών μας επιτρέπουν να αλληλεπιδρούμε ζωντανά με τον κόσμο μας μέσω των μηχανών φυσικής τους. Έτσι δεν μπορούμε για παράδειγμα να περάσουμε μέσα από τους τοίχους ή αν πέσουμε από ένα ύψωμα θα συνεχίσουμε να πέφτουμε έως ότου φτάσουμε στο έδαφος. Η πτώση αυτή ανάλογα με το ύψος έχει και κάποιες συνέπειες. Όλοι αυτοί οι μαθηματικοί υπολογισμοί γίνονται σε πραγματικό χρόνο από το game engine. Τα game engines παρέχουν εκτός από τα επαναχρησιμοποιήσιμα τμήματα λογισμικού και ένα σύνολο από οπτικά εργαλεία ανάπτυξης. Αυτά τα 6 εργαλεία παρέχονται γενικά σε ένα ενσωματωμένο περιβάλλον ανάπτυξης, επιτρέποντας έτσι την απλουστευμένη και γρήγορη ανάπτυξη των παιχνιδιών. Κάποιες μηχανές παιχνιδιών παρέχουν μόνο δυνατότητες 3D αναπαράστασης σε πραγματικό χρόνο αντί για ένα ευρύ φάσμα λειτουργιών που απαιτείται από τα παιχνίδια. Αυτές οι μηχανές βασίζονται στον δημιουργό του κάθε παιχνιδιού στο να εφαρμόσει τις υπόλοιπες λειτουργίες ή να τις συγκεντρώσει από άλλα τμήματα ήδη υπάρχον παιχνιδιών. Αυτοί οι τύποι μηχανών αναφέρονται ως μηχανές γραφικής παράστασης (graphics engines), μηχανές αναπαράστασης (rendering engine) ή μηχανές 3Δ (3D engine), αντί για μηχανές παιχνιδιών (game engine). Οι σύγχρονες μηχανές παιχνιδιών ή γραφικής παράστασης παρέχουν μια σκηνή αναπαράστασης (scene graph) η οποία είναι μια αντικειμενοστραφής αντιπροσώπευση του τρισδιάστατου κόσμου παιχνιδιών που απλοποιεί συχνά το σχέδιο παιχνιδιών και μπορεί να χρησιμοποιηθεί για την καλύτερη απόδοση των απέραντων εικονικών κόσμων. Ο όρος “ μηχανή παιχνιδιών” προέκυψε στα μέσα της δεκαετίας του 90 ειδικά για τα 3D παιχνίδια όπως τα first person shooters (FPS).Παιχνίδια όπως το Doom και το Quake δεν δημιουργήθηκαν από το μηδέν. Κάποιοι προγραμματιστές είχαν χορηγήσει άδειες για σημαντικά τμήματα του λογισμικού και έτσι έπρεπε να σχεδιαστούν μόνο οι γραφικές σκηνές, οι χαρακτήρες , τα όπλα , τα επίπεδα , καθώς και τα περιεχόμενα και προτερήματα των παιχνιδιών

2.4 Τρισδιάστατες Μηχανές Απεικόνισης



Σχήμα 2.2: Γραφική Απεικόνιση Παιχνιδιών

Όπως έχουμε ήδη αναφέρει η μηχανή παιχνιδιών (ή 3D μηχανή) είναι ένα πολύπλοκο σύστημα, υπεύθυνο για τον σχηματισμό της εικόνας και του ήχου ενός παιχνιδιού που χειρίζεται τα δεδομένα που εισάγει ο χρήστης και παρέχει πηγές διαχείρισης, animation (απεικόνιση με κινούμενα σχέδια), φυσική και πολλά άλλα. Στο χαμηλότερο επίπεδο βρίσκεται το hardware. Το API το οποίο έχει πρόσβαση στο hardware βρίσκεται ένα επίπεδο πιο πάνω. Συχνά οι τρισδιάστατες μηχανές ή τα συστήματα αναπαράστασης σε μια μηχανή παιχνιδιών είναι χτισμένες πάνω σε ένα API γραφικών όπως το Direct3D ή το OpenGL που ελαφρύνουν κατά κάποιο τρόπο το λειτουργικό του GPU (Graphics Processing Unit) ή της τηλεοπτικής κάρτας. Οι χαμηλού επιπέδου βιβλιοθήκες όπως DirectX, OpenAL επίσης χρησιμοποιούνται συνήθως στα παιχνίδια καθώς

παρέχουν ανεξαρτησία hardware παρέχοντας πρόσβαση και σε άλλους υπολογιστές με διαφορετικά χαρακτηριστικά μονάδων εισόδου (ποντίκι, πληκτρολόγιο, joystick) , κάρτες δικτύου και ήχου. Η εισαγωγή δεδομένων γίνεται μέσω του DirectInput. Το GFX Renderer είναι υπεύθυνο για την απεικόνιση της τελικής σκηνής με όλα τα φαντασμαγορικά εφφέ. Το Sound Renderer έχει την ίδια θέση με τον ήχο και παίζει τους ήχους την σωστή στιγμή χρησιμοποιώντας τα σωστά εφφέ. Ο Input Handler συγκεντρώνει πληροφορίες και αλλαγές από το πληκτρολόγιο, το ποντίκι ή οποιεσδήποτε άλλες μονάδες εισόδου έχουμε και τις μετατρέπει σε τέτοια μορφή που να γίνονται αποδεκτές από την μηχανή. Είναι επίσης υπεύθυνο για να δίνει σωστές εντολές πίσω στις μονάδες (devices) ανάλογα με τις συνέπειες που ίσως υπάρχουν. Το engine scene renderer χρησιμοποιεί μεθόδους χαμηλότερου επιπέδου για να αποδώσει την σκηνή πάνω στην οθόνη και για να ακουστούν οι σωστοί ήχοι. Το επίπεδο πάνω από τον Renderer έχει πολλές λειτουργίες που συνεργάζονται μεταξύ τους. Συμπεριλαμβάνει animation και collision detection. Το Deformation χρησιμοποιεί τη φυσική για να προσδιορίσει τα σχήματα σε σχέση με τις δυνάμεις που ασκούνται. Μέσω της φυσικής μετριέται και υπολογίζεται η βαρύτητα, ο άνεμος κ.α. Σε αυτό το επίπεδο μπορεί να υπάρχουν και πολλές άλλες λειτουργίες οι οποίες εξαρτώνται από την εκάστοτε εφαρμογή. Πάνω από όλα αυτά βρίσκεται η εφαρμογή μας η οποία είναι υπεύθυνη για τα δεδομένα του παιχνιδιού, το AI (Artificial Intelligence), το GUI (Graphical User Interface) και επίσης τα δεδομένα που εισάγει ο χρήστης. Οι σύγχρονες μηχανές παιχνιδιών είναι μερικές από τις πιο σύνθετες εφαρμογές που έχουν γραφτεί χαρακτηρίζοντας συχνά πληθώρα από άριστα συντονισμένα συστήματα αλληλεπίδρασης για να εξασφαλίσουν μια πλήρη ελεγχόμενη εμπειρία για τους χρήστες. Η συνεχής βελτίωση των μηχανών παιχνιδιών έχει δημιουργήσει έναν ισχυρό διαχωρισμό μεταξύ του rendering, scripting,artwork και το level design. Τα παιχνίδια First Person Shooter παραμένουν οι κυρίαρχοι των μηχανών παιχνιδιών 3D. Το threading όσο περνάει ο καιρός παίρνει περισσότερη σημασία λόγο των σύγχρονων multi –core συστημάτων και των αυξημένων απαιτήσεών στην ρεαλιστικότητα. Τα threads χαρακτηριστικά περιλαμβάνουν το rendering, το streaming, τον ήχο και την φυσική. Σε κάθε γενιά των 3D accelerators βλέπουμε ομορφότερα και πιο ρεαλιστικά παιχνίδια με περισσότερες αλληλεπιδράσεις και πιο περίπλοκα μοντέλα

2.5 LWJGL - LightWeight Java Game Library

Η ελαφριά βιβλιοθήκη παιχνιδιών της Java (LWJGL) είναι μια λύση που στοχεύει άμεσα στους επαγγελματίες και ερασιτέχνες προγραμματιστές και τους επιτρέπει να δημιουργούν παιχνίδια εμπορικής ποιότητας γραμμένα σε java. Η LWJGL παρέχει πρόσβαση υψηλής απόδοσης σε crossplatform libraries (παίζουν όπου μπορεί να παίξει java) όπως η OpenGL (Open Graphics Library) και η OpenAL (Open Audio Library). Επιπλέον, η LWJGL παρέχει πρόσβαση σε controllers όπως Gamepads, Steering wheel και Joysticks όλα αυτά μέσα από ένα εύκολο και απλό API. Η LWJGL δεν καθιστά ιδιαίτερα εύκολο το γράψιμο των παιχνιδιών. Είναι πρωτίστως μια τεχνολογία η οποία επιτρέπει στους προγραμματιστές να αναπτύξουν πόρους οι οποίοι διαφορετικά είναι ή μη διαθέσιμοι ή κακώς σχεδιασμένοι στην υπάρχουσα java. Προσδοκάται ότι η LWJGL, μέσω της εξέλιξης και της επέκτασης της, θα αποτελέσει θεμέλιο για τις πληρέστερες βιβλιοθήκες παιχνιδιών ή “τις μηχανές παιχνιδιών” όπως είναι ευρύτερα γνωστό.

2.6 Open GL

Σχήμα 2.3: Open GL

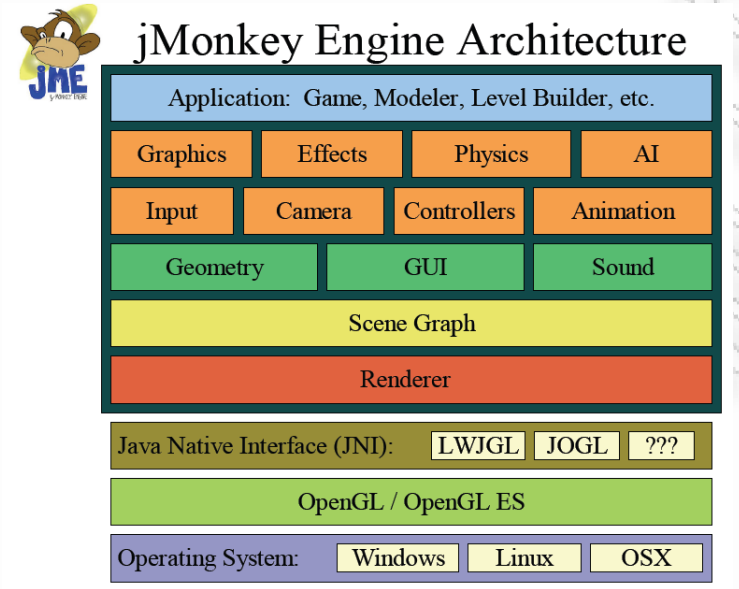
Η OpenGL είναι το αρχαιότερο περιβάλλον που καθορίζει ένα cross-language API για την ανάπτυξη φορητών, διαλογικών 2D και 3D εφαρμογών γραφικής παράστασης. Η OpenGL αναπτύχθηκε από την Silicon Graphics Inc. (SGI) το 1992 και είναι δημοφιλής στην βιομηχανία των video παιχνιδιών όπου ανταγωνίζεται το Direct3D της Microsoft. Το interface της αποτελείται πάνω από 250 διαφορετικές λειτουργίες που μπορούν να χρησιμοποιηθούν για να δημιουργήσουν σύνθετες τρισδιάστατες σκηνές από απλά σχήματα. Η OpenGL χρησιμοποιείται ευρέως στο CAD, την εικονική πραγματικότητα,την επιστημονική απεικόνιση, την απεικόνιση πληροφοριών, την προσομοίωση πτήσης και την τηλεοπτική ανάπτυξη παιχνιδιών. Στο πιο βασικό της επίπεδο η OpenGL είναι ένα έγγραφο το οποίο περιγράφει ένα σύνολο λειτουργιών και τις ακριβείς συμπεριφορές που πρέπει να εκτελούνται. Η OpenGL εξυπηρετεί δύο κύριους σκοπούς:

1. Κρύβει την πολυπλοκότητα διασύνδεσης με τους διαφορετικούς 3D acceleratos παρέχοντας ένα ενιαίο και ομοιόμορφο API στον προγραμματιστή.
2. Κρύβει τις διαφορετικές ικανότητες των πλατφόρμων υλικού, με την απαίτηση ότι όλες οι εφαρμογές πρέπει υποστηρίζουν το πλήρες σύνολο χαρακτηριστικών γνωρισμάτων της OpenGL

Βασική λειτουργία της openGL είναι να γίνουν αποδεκτά απλά στοιχεία όπως σημεία, γραμμές και πολύγωνα, και να το μετατρέπουν σε Pixels. Αυτό γίνεται με το graphics pipeline (τρόπος με τον οποίο γίνονται αποδεκτές εισαγωγές 3D δεδομένων) που είναι γνωστό ως OpenGL state machine. Οι περισσότερες εντολές στην OpenGL δίνουν στο graphics pipeline απλά σχήματα ή συντονίζουν πώς το pipeline θα επεξεργαστεί τα σχήματα αυτά.

2.7 Αρχιτεκτονική του JME

Τα επίπεδα αρχιτεκτονικής του jME απεικονίζονται στο 2.4 σχήμα.



Σχήμα 2.4: Αρχιτεκτονική του jME

Αν και όλα τα επίπεδα θα τα δούμε στην συνέχεια αναλυτικά, συνοπτικά μπορούμε να πούμε ότι στο πρώτο επίπεδο ορίζονται τα λειτουργικά συστήματα στα οποία μπορούν να τρέχουν οι εφαρμογές του jME. Ας μην ξεχνάμε ότι το jME είναι 100 Java και εξαρτάται από πλατφόρμα JNI (Java Native Interface). Έτσι τα λειτουργικά συστήματα τα οποία μπορούμε να χρησιμοποιήσουμε είναι τα Windows, Linux και OSX όπου μπορεί να τρέχει η LWJGL. Στο επόμενο στάδιο βρίσκονται τα OpenGL/ OpenGL ES. Όπως ήδη αναφέραμε η OpenGL είναι ένα cross–platform API για υψηλή απόδοση 2D και 3D γραφικών. Η OpenGL ES παρέχει προχωρημένες δυνατότητες για 2D/3D γραφικά, μεγάλη ποικιλία από κινητές συσκευές και συμπτυγμένες απεικονίσεις. Στο επόμενο επίπεδο βρίσκεται το **JNI**. Η τεχνική JNI (Java Native Interface) είναι η τοπική διασύνδεση προγραμμάτων της Java η οποία είναι μέρος του JDK. Η τεχνική JNI επιτρέπει τον κώδικα της Java, ο οποίος τρέχει με ένα φανταστικό μηχάνημα της Java VM (Virtual Machine), να χειρίζεται εφαρμογές και βιβλιοθήκες οι οποίες είναι γραμμένες σε άλλες γλώσσες, όπως είναι η C, C++ και assembly. Αμέσως πιο πάνω βρίσκεται ο Renderer. Οι ιδιότητες του Renderer είναι να μετατρέπει τα δεδομένα της σκηνής από τον πραγματικό κόσμο στο χώρο της οθόνης. Επίσης για να έχουμε καλύτερη απόδοση σε ταχύτητα, αφαιρεί από την διαδικασία του Rendering κομμάτια από την σκηνή παράστασης με τους ακόλουθους τρόπους :

- Culling : είναι η διαδικασία κατά την οποία διευκρινίζεται αν κάποιο από τα μέρη ενός αντικειμένου είναι ορατό από κάποιο συγκεκριμένο σημείο αναφοράς. Αν όχι τότε δεν θα περάσει από την διαδικασία του Render.
- Clipping : είναι η διαδικασία κατά την οποία χωρίζεται ένα αντικείμενο σε μικρότερα κομμάτια και τα κομματάκια τα οποία δεν είναι ορατά δεν περνάνε από rendering.

Επίσης ο Renderer είναι υπεύθυνος για την αναπαράσταση της μορφοποιημένης- τροποποιημένης σκηνής μας στην οθόνη. Ακολουθεί το scene graph. Το scene graph μπορούμε να το παρομοιάσουμε με ένα δέντρο όπου τα δεδομένα είναι κατηγοριοποιημένα σύμφωνα με μια ιεραρχική δομή. Η δομή αυτή περιέχει:

- Τους κόμβους γονέα (parent Nodes) που περιέχουν όσους κόμβους παιδιών θέλουμε.

- Τους κόμβους παιδιών (child Nodes) που έχουν έναν μόνο parent Node.
- Το rootNode που δεν έχει κανένα parent Node.
- Τα Leaf Nodes που περιέχουν γεωμετρικά δεδομένα
- Εσωτερικούς κόμβους για την διαχείριση ομάδων.

Πλεονεκτήματα scene graph:

1. απλοποιεί την διαχείριση γενικών χαρακτηριστικών π.χ. ορίζει φως για να φωτίσει ένα μόνο υποδέντρο της σκηνής
2. επιτρέπει την ομαδοποίηση αντικειμένων που βρίσκονται στην ίδια χωροταξική περιοχή π.χ. βοηθάει στην γρήγορη απαλοιφή περιοχών της σκηνής οι οποίες δεν θα είναι ορατές και έτσι δεν χρειάζονται να περάσουν από την διαδικασία του rendering.
3. διευκολύνει τον προσανατολισμό των ιεραρχικών μοντέλων όπως οι ανθρωποειδής χαρακτήρες π.χ. το κορμί καθορίζει την θέση του κεφαλιού, των χεριών και των ποδιών. Τα χέρια προσδιορίζουν την θέση της παλάμης κτλ.

Η απεικόνιση του scene graph σαν δέντρο προσφέρει:

1. Οι κόμβοι να διατηρούν σημαντικές πληροφορίες και πληροφορίες χώρου.
2. Μετασχηματισμούς, για την αλλαγή θέσης, προσανατολισμού και μεγέθους των αντικειμένων.
3. Bounding Volumes, χρησιμοποιούνται για την αφαίρεση κομματιών ιεραρχικά (hierarchal culling) και για τεστ συγκρούσεων.
4. Render State, χρησιμοποιείται για να στηθεί ο renderer ο οποίος θα απεικονίσει τα αντικείμενα.
5. Animation State, χρησιμοποιείται για να δείξει πως ο χρόνος επηρεάζει τα δεδομένα των κόμβων.

Στο αμέσως επόμενο στάδιο βρίσκεται η γεωμετρία, ο ήχος και το GUI. Η γεωμετρία προσδιορίζει τους Leaf κόμβους της σκηνής και περιέχουν γεωμετρικά δεδομένα που χρειάζονται για το Rendering των αντικειμένων. Επίσης διαχειρίζεται όλες τις πληροφορίες rendering συμπεριλαμβανομένων διαφορετικών καταστάσεων rendering, και δεδομένων των μοντέλων. Το GUI περιλαμβάνει τους Layout Managers. Οι ήχοι γίνονται render σε 3D χώρο και καθορίζονται από 2 interfaces. Το ISoundSystem που είναι υπεύθυνο για να παίζει την μουσική και το ISoundRenderer που διαχειρίζεται την σύνδεση του ήχου με την σκηνή αναπαράστασης καθώς και την διαχείριση των 3D ήχων. Στο επόμενο επίπεδο βρίσκονται αρκετές λειτουργίες οι οποίες αλληλεπιδρούν μεταξύ τους. Οι δυνατότητες των γραφικών είναι πάρα πολλές όπως να κάνουμε render ένα texture, να έχουμε multi-texturing υποστήριξη, να έχουμε την δυνατότητα φόρτωσης διάφορων τύπων μοντέλου, δημιουργία απλών σχημάτων (σημείου, γραμμής, κουτιού, σφαίρας κτλ), διαχείρισης καταστάσεων (όπως lights, culling, texturing, Zbuffer ,Materials), δυνατότητα για εφφέ (μέχρι στιγμής διαθέσιμα είναι το tinting και το particle system (ίχνη)). Από μονάδες εισόδου μέχρι στιγμής υποστηρίζονται το mouse και το πληκτρολόγιο. Η camera χρησιμοποιείται για να προβάλει τα αντικείμενα στην οθόνη του υπολογιστή. Είναι ένας συνδυασμός από την θέση του ματιού- τη θέση της cameras, το οπτικό πεδίο (view plane), το μέρος του οπτικού πεδίου που θα φαίνεται (view port) και το view frustrum. Οι controllers μεταλλάσσουν τους κόμβους και τα render states σε σχέση με το χρόνο. Στο τελευταίο στάδιο βρίσκονται τα πρότυπα, τα παιχνίδια και γενικά όλες οι εφαρμογές μας που δημιουργούμε.

2.8 Χαρακτηριστικά γνωρίσματα του jME

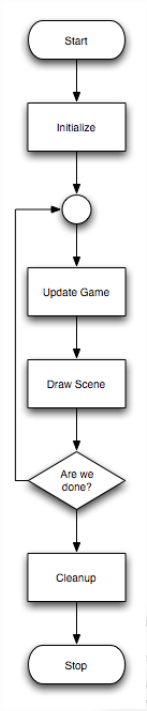
Το jMonkey Engine σχεδιάστηκε να είναι μια μηχανή γραφικών μεγάλης ταχύτητας σε πραγματικό χρόνο. Με τις προόδους που έγιναν τόσο στα graphics hardware όσο και στην γλώσσα προγραμματισμού της java , η ανάγκη για μια βιβλιοθήκη παιχνιδιών βασισμένη στην java ήταν πλέον προφανής. Το jME καλύπτει πολλές από αυτές τις ανάγκες παρέχοντας ένα σύστημα υψηλής απόδοσης (rendering system) γραφικών παραστάσεων σκηνής. Ο πυρήνας του JME είναι το scene graph. Το scene graph είναι η δομή δεδομένων που διατηρεί τα στοιχεία του κόσμου Οι σχέσεις μεταξύ των δεδομένων του

παιχνιδιού (γεωμετρίας , ήχου, φυσικής) διατηρούνται σε μια δομή δέντρου, όπου οι κόμβοι φύλλων αντιπροσωπεύουν τα στοιχεία του πυρήνα του παιχνιδιού. Αυτά τα βασικά στοιχεία επί το πλείστον είναι αυτά που φαίνονται στη σκηνή, ή που ακούγονται μέσω της κάρτας ήχου. Η οργάνωση του scene graph είναι πολύ σημαντική και εξαρτάται από την εκάστοτε εφαρμογή. Γενικά το scene graph αντιπροσωπεύει ένα μεγάλο αριθμό δεδομένων που έχει χωριστεί σε μικρότερα εύκολοσυντηρήσιμα κομμάτια. Αυτά τα κομμάτια ομαδοποιούνται με βάση κάποιο είδος σχέσης, συνηθέστερα από τη χωροταξική τους θέση. Αυτή η ομαδοποίηση επιτρέπει σε μεγάλα τμήματα δεδομένων του παιχνιδιών να αφαιρεθούν από την επεξεργασία εάν δεν απαιτούνται για να επιδείξουν την τρέχουσα σκηνή. Καθορίζοντας ότι ένα τμήμα του κόσμου δεν απαιτείται να υποβληθεί σε επεξεργασία, χρησιμοποιείται λιγότερη CPU και ξοδεύεται λιγότερος χρόνος GPU για την επεξεργασία των δεδομένων με αποτέλεσμα να βελτιώνεται η ταχύτητα του παιχνιδιού. Το scenegraph επιτρέπει την οργάνωση των στοιχείων του παιχνιδιών σε μια δομή δέντρων, όπου ένας κόμβος γονέων (parent node) μπορεί να περιέχει όσους κόμβων παιδιών (children nodes) θέλει. Αλλά, ένας κόμβος παιδιών περιέχει μόνο έναν κόμβο γονέα. Συνήθως, αυτοί οι κόμβοι οργανώνονται στο χώρο για να επιτρέπουν τη γρήγορη απόρριψη ολόκληρων υποδιαιρέσεων για επεξεργασία. Για παράδειγμα, έστω ότι έχουμε δημιουργήσει κάποια αντικείμενα τα οποία βρίσκονται μέσα σε ένα δωμάτιο. Όλα τα αντικείμενα μοιράζονται τον γονέα δωμάτιο, όλα τα δωμάτια μοιράζονται τον γονέα πάτωμα και όλα τα πατώματα έχουν τον γονέα κτίριο. Αν ο παίκτης είναι στο πρώτο δωμάτιο του πρώτου ορόφου τότε μπορούμε εύκολα και γρήγορα να απορρίψουμε τον κόμβο του δεύτερου ορόφου και αυτόματα να απορριφθούν και όλα τα δωμάτια που βρίσκονται στον όροφο αυτό, καθώς και όλα τα αντικείμενα που βρίσκονται μέσα σε αυτά τα δωμάτια. Μπορούμε έπειτα να επεξεργαστούμε το πάτωμα 1. Όλα τα δωμάτια που δεν είναι δωμάτιο 1 απορρίπτονται, καθώς και όλα τα αντικείμενα των δωματίων αυτών. Επεξεργαζόμαστε έπειτα το δωμάτιο 1 συμπεριλαμβανομένου όλων των αντικείμενων του. Η απόρριψη – αφαίρεση, μπορεί να σημαίνει ποικίλα πράγματα, αλλά ο σημαντικότερος στον προγραμματισμό γραφικής αναπαράστασης είναι το ξεδιάλεγμα των δεδομένων (culling of data). Το σύστημα κάμερας του jME χρησιμοποιεί το frustum culling για να αφαιρεί κομμάτια της σκηνής που δεν είναι ορατά. Αυτό επιτρέπει στις σύνθετες σκηνές να αποδίδονται γρήγορα, αφού στις περισσότερες περιπτώσεις το μεγαλύτερο μέρος της σκηνής δεν είναι ορατό τον περισσότερο χρόνο Ενώ η γραφική παράσταση σκηνής (scene graph) είναι ο πυρήνας των στοιχείων της γραφικής παράστασης του jME, τα εργαλεία εφαρμογής παρέχουν τα μέσα να δημιουργηθεί γρήγορα το πλαίσιο γραφικής παράστασης και να αρχίσει ένας κύριος βρόχος παιχνιδιών. Η δημιουργία του παραθύρου, το σύστημα εισαγωγής (input system), κάμερας και το σύστημα παιχνιδιών δεν είναι τίποτα περισσότερο από μια ή δύο κλήσεις της κάθε μεθόδου. Αυτό επιτρέπει στον προγραμματιστή να σταματήσει να χάνει το χρόνο του εξετάζοντας το σύστημα και να έχει περισσότερο χρόνο να ασχοληθεί με την εφαρμογή του. Ο χρήστης απαλλάσσεται ουσιαστικά από το rendering του scene graph. Το όφελος αυτό μας επιτρέπει την ανταλλαγή σε διαφορετικά συστήματα απόδοσης χωρίς να χρειάζεται να αλλαχτεί ούτε μια γραμμής κώδικα. Παραδείγματος χάριν, εάν τρέχουμε την εφαρμογή μας χρησιμοποιώντας την LWJGL μπορούμε να μεταπηδήσουμε εύκολα σε JOGL της sun. Το τελικό αποτέλεσμα θα διαφέρει ελάχιστα και δεν θα χρειάζεται να ξαναγραφτεί το πρόγραμμα μας. Εντούτοις, για να δημιουργηθεί ένα καλύτερο API, η εστίαση μέχρι τώρα είναι στην προσθήκη νέων χαρακτηριστικών γνωρισμάτων παρά τη δημιουργία ενός νέου συστήματος απόδοσης . Επομένως, η LWJGL είναι αυτήν την περίοδο το μόνο σύστημα renderer που υποστηρίζεται. (το JOGL έχει δημιουργηθεί αλλά δεν υποστηρίζεται μέχρι στιγμής)

Κεφάλαιο 3

Java Monkey Engine

3.1 Main Game Loop



Σχήμα 3.1: jME Properties

Στον πυρήνα κάθε παιχνιδιού υπάρχει ένας βρόχος. Ο βρόχος αυτός είναι ένα τμήμα κώδικα που εκτελείται συνέχεια καθ' όλη την διάρκεια του παιχνιδιού και είναι υπεύθυνος για να συντονίζει έναν κύκλο από ενημερώσεις και οι απεικονίσεις (update/draw). Όλες οι AbstactGame υποκλάσεις παίζουν ρόλο στην δημιουργία αυτού του βρόχου.Παρ΄ όλο που οι υποκλάσεις ποικίλουν όλες ακολουθούν την λογική που φαίνεται στο παρακάτω σχήμα. Για την δημιουργία των σχημάτων στης εφαρμογής μας χρειαζόμαστε μια main class που θα περιέχει την μέθοδο main. Το jme παρέχει την κλάση AbstractGame και τις υποκλάσεις της. Η AbstractGame παρέχει την main game loop. Αυτός ο βρόχος αποτελεί το στήριγμα της εφαρμογής μας γιατί αυτός είναι υπεύθυνος για την καθοδήγηση και ολοκλήρωση της εφαρμογής μας. Καλείται επαναλαμβανόμενα έως ότου η εφαρμογή μας τελειώσει (επανάληψη ανά frame) Το πώς θα διεξαχθεί το game loop εξαρτάται από ποιες υποκλάσεις του AbstractGame θα επιλέξουμε. Το game loop ξεκινάει όταν καλέσουμε την μέθοδο start

3.2 Properties Dialog

Το PropertiesDialog είναι ένα dialog το οποίο εμφανίζεται -αν το επιλέξουμε εμείς - όταν ξεκινάει η εφαρμογή μας και ζητά πληροφορίες για το πως θα τρέξει η εφαρμογή μας (display settings). Καλώντας την μέθοδο setDialogBehaviour Το dialog μας δίνει την δυνατότητα να επιλέξουμε την ανάλυση(resolution), το βάθος χρώματος(color depth), αν θα είναι full screen και τον ρυθμό ανανέωσης(refresh rate) και την δυνατότητα επιλογής API. Έπειτα αυτές οι ιδιότητες αποθηκεύονται σε ένα configuration file με την βοήθεια του PropertiesIO κάθε φορά που πάμε να εκτελέσουμε την εφαρμογή μας. Ανάλογα με το τι παραμέτρους θα δώσουμε στην setDialogBehaviour, το dialog θα εμφανίζεται πάντα, δεν θα εμφανίζεται ποτέ, ή θα εμφανίζεται όταν δεν υπάρχει το properties configuration file. Η AbstractGame δημιουργεί από μόνη της το dialog αυτό και εμείς μπορούμε να έχουμε πρόσβαση σε αυτήν απλά καλώντας την κλάση.

Ανάλυση παραμέτρων του Properties dialog:

- NEVER_ SHOW_ PROPS_ DIALOG

  Δεν θα εμφανιστεί ποτέ το παράθυρο επιλογών και έτσι δεν θα υπάρχει το properties file. Με αυτήν την επιλογή πρέπει να είμαστε σίγουροι το σύστημα μας έχει την δυνατότητα να τρέξει από default εφαρμογές με χαρακτηριστικά 640x480x60Hzx16bits fullscreen.

- FIRSTRUN_ OR_ NOCONFIGFILE_ SHOW_ PROPS_ DIALOG

  θα εμφανιστεί το παράθυρο επιλογών αν δεν υπάρχει το configuration file. Έτσι επιτρέπετε στον χρηστή να αλλάξει της παραμέτρους που θέλει μια φορά στην αρχή και να μην χρειάζεται να ξανα ασχοληθει με αυτό το θέμα. Ωστόσο αν ο χρηστής θελήσει για κάποιο λόγο αργότερα να αλλάξει τις παραμέτρους αυτές θα πρέπει να σβήσει το properties file έτσι ώστε να μην υπάρχει για να του ξαναζητηθεί από το σύστημα να τα ορίσει από την αρχή.

- ALWAYS_ SHOW_ PROPS_ DIALOG

  πάντα θα εμφανίζεται το παράθυρο επιλογών δίνοντας την δυνατότητα για εύκολη και γρήγορη αλλαγή των παραμέτρων.

Το PropertiesIO όπως είπαμε και προηγουμένως δημιουργεί ένα properties file και κρατάει αποθηκευμένα τις ρυθμίσεις της εφαρμογής μας. Εμείς δεν χρειάζεται να κάνουμε κάτι ιδιαίτερο αφού η AbstractGame ορίζει από μόνη της την κλάση PropertiesIO και αποθηκεύει το αρχείο αυτό ως properties.cfg στο τρέχον directory. Υπάρχουν αρκετοί τύποι παιχνιδιών που παρέχει η βιβλιοθήκη του JME. Εμείς θα αναφερθούμε στην BaseGame, SimpleGame που είναι και οι πιο βασικές. Η SimplePhysicsGame που είναι γραμμένη η εφαρμογή μας είναι υποκλάση της BaseGame.

```
AppSettings settings = new AppSettings(true);
settings.setRenderer(AppSettings.LWJGL_OPENGL3);
Application app = new Application();
app.setSettings(settings);
app.start();

/*****************************Several Settings*****************************/

setRenderer(AppSettings.LWJGL_OPENGL2)       //
setRenderer(AppSettings.LWJGL_OPENGL3)      //Switch Video Renderer   OpenGL 2

setAudioRenderer(AppSettings.LWJGL_OPENAL) //
setAudioRenderer(AppSettings.LWJGL_JOAL)    // Switch Audio Renderer  OpenAL

setBitsPerPixel(8)    //Set color depth.
                      //1 = black and white, 2 bit = gray,
                      //4 = 16 colors, 8 = 256 colors, 24 or 32 = "truecolor".

setFrequency(60)  // The screen frequency (refresh rate of the graphics card),
                  // usually 60 or 75 fps.    60 fps

setFramerate(60)  //How often per second the engine should try to refresh
                  //the frame. For the release, usually 60 fps. Can be lower
                  //(59-30) for FX-intensive games. Do not set to a
                  //higher value than the screen frequency. -1 (auto)

setFullscreen(true)   //Set this to true to make the display fill the whole
                      //screen; you need to provide a key that calls app.stop()
                      //to exit the fullscreen view gracefully (default: escape).
                      //Set it to false to play the game in a normal window of
                      //its own.  False (windowed)

setHeight(480),
setWidth(640)
setResolution(640,480)  //Two equivalent ways of setting the display resolution

setSamples(4)   //Set multisampling to 0 to switch antialiasing off
                //(harder edges, faster.)Set multisampling to 2 or 4 to activate
                //antialising (softer edges, may be slower.)Depending on your
                //graphic card, you may be able to go set multisampling to
                //8, 16, or 32 samples.




setVSync(true) //Set vertical syncing to true to time the frame buffer to
```

```
            //coincide with the refresh interval of the screen:
49          //Prevents page tearing, but slower; recommened for release.
            //Set to false to deactivate vertical syncing
            //(faster, but possible page tearing artifacts);
52          //can be deactivated during development.

    useInput(false)    //Respond to user input by mouse and keyboard.
55                     //Can be deactivated for use cases where you only
                       //display a 3D scene on the canvas without any interaction.

58  useJoysticks(true)    //Activate optional joystick support

    setSettingsDialogImage("/path/in/assets.png")     //A custom image to display
61                                                    //when the settings dialog
                                                      //is shown.    "/com/jme3/app/Monkey.png"

64  setTitle("MPPL08059-Game") //This string will be visible in the titlebar,
                               //unless the window is fullscreen.
```

Κώδικας 3.1: JME Settings

3.3 Base Game

Η BaseGame παρέχει την πιο βασική εφαρμογή της κλάσης AbstractGame. Αυτό που κάνει είναι και καθορίζει το main game loop και μόνο. Το loop είναι ταχύτατο και κάθε του επανάληψη εξαρτάται από το πόσο γρήγορα δουλεύουν οι CPU/GPU. Σε αυτήν την περίπτωση είμαστε υποχρεωμένοι να δημιουργήσουμε τις παρακάτω μεθόδους.

- InitSystem. Με την μεθοδο αυτή στήνουμε το display system.Τα στοιχεία που περιέχει(κάμερα, input system κτλ) δεν είναι στοιχεία του παιχνιδιού και εδώ γίνεται η αρχικοποίηση τους.
- InitGame. Είναι η μέθοδος όπου δημιουργούνται όλα τα δεδομένα του παιχνιδιού και των γραφικών.
- Update. Με την μέθοδο αυτή ανανεώνουμε τα δεδομένα σε κάθε παιχνίδι. Αυτό μπορεί να είναι ο timer του παιχνιδιού, μέθοδοι εισαγωγής δεδομένων, έλεγχος συγκρούσεων (collision checks). Οτιδήποτε μεταβάλλεται στην διάρκεια του χρόνου θα πρέπει να περνάει μέσα από την μέθοδο αυτή.
- Render. Η μέθοδος αυτή διαχειρίζεται το πώς θα εμφανιστεί γραφικά η σκηνή(scene).
- Reinit. Είναι η μέθοδος που χρησιμοποιούμε όταν το σύστημα μας χρειάζεται να ξαναδημιουργήσει κάτι (rebuild) π.χ. όταν αλλάζουμε τις ρυθμίσεις στο display.
- Cleanup. Η μέθοδος αυτή κλείνει τυχών ανοιχτές πήγες που έχουμε αφήσει έτσι ώστε να επιτρέψει στο σύστημα να επιλύσει το πρόβλημα ομαλά.

3.4 Simple Game

Ένας άλλος τύπος για γρήγορο και εύκολο σχεδιασμό παιχνιδιών είναι η SimpleGame. Η SimpleGame είναι υποκλάση της BaseGame και “κληρονομεί” όλες τις Abstract μεθόδους της BaseGame. Στην SimpleGame υπάρχει εξ’ ορισμού ένα root scene node (rootNode) όπου μπορούν να προσκολληθούν όλα τα δεδομένα του παιχνιδιού. Σε αυτόν τον γονικό κόμβο γίνετε αυτόματα render. Όλα τα υπόλοιπα αρχικοποιούνται από μόνα τους συμπεριλαμβανόμενου του συστήματος εισαγωγής δεδομένων (FirstPersonHandler), του timer, η απόδοση του : lighting, zbuffer, και WireFrameState (εξ ορισμού είναι ανενεργό). Επίσης περιέχει την μέθοδο simpleInitGame στην οποία ο χρηστής δημιουργεί όλα τα στοιχεία και τα δεδομένα που θέλει να έχει η εφαρμογή του. Όλα αυτά παρέχουν μια βασική πλατφόρμα έτσι ο χρηστής να μπορεί να πειραματιστεί με τα χαρακτηρίστηκα του JME χωρίς να χρειάζεται να ανησυχεί για το πώς θα στήσει μια πλήρης σε χαρακτηρίστηκα εφαρμογή. Αν θέλουμε μπορούμε να κάνουμε override την simpleUpdate για να έχουμε επιπρόσθετες απαιτήσεις ανανέωσης. Αυτό θα προσθέσει μόνο επιπλέον ιδιότητες στην update της SimpleGame χωρίς να καταργεί κάποια ήδη υπάρχουσα. Επίσης μπορούμε να κάνουμε override στην simpleRender ούτε και σε αυτήν την περίπτωση καταργείται κάτι που ήδη υπάρχει. Η SimpleGame δημιουργεί ένα σύστημα εισαγωγής δεδομένων με βάση τα παρακάτω πλήκτρα

Πλήκτρο	Ενέργεια
W	Move Forward
A	Move Left
S	Move Backward
D	Move Right
Up	Look Up
Down	Look Down
Left	Look Left
Right	Look right
T	Wireframe Mode on/off
P	Pause On/Off
L	Lights On/Off
C	Print Camera position
B	Bounding Volumes On/Off
N	Normmals On/Off
F1	Screenshot

Πίνακας 3.1: Πλήκτρα και Ενέργειες

Πρόσβαση στον InputHandler μπορούμε να έχουμε χρησιμοποιώντας την μεταβλητή input που παρέχει η SimpleGame. Η SimpleGame δημιουργεί μια στάνταρ camera στην θέση (0,0,25) , αριστερό διάνυσμα (-1,0,0), πάνω (0,1,0) και διεύθυνση (0,0,-1). Στην simpleGame αυτή η camera αναφέρεται ως cam και ο χειρισμός της γίνεται από την input. Ακόμα η SimpleGame μας παρέχει φως. Ένα φως με κατεύθυνση που βρίσκεται στην θέση (100,100,100). Το οποίο είναι προσκολλημένο σε ένα LightState που ονομάζεται και lightState και είναι προσκολλημένο στο rootNode. Ο φωτισμός αυτός θα επηρεάσει κάθε στοιχείο που έχει προστεθεί στην σκηνή εκτός και αν πατήσουμε το πηκτό L η καλέσουμε την lightState.setEnabled(false) όπου απενεργοποιείται.

Επιπλέον η SimpleGame δημιουργεί έναν timer για να παρακολουθεί την αναλογία των frame (frame rate) και τον χρόνο μεταξύ των frames. Ο χρόνος μεταξύ των frames tpf (time per frames) περνάει στις μεθόδους update και render και ειδοποιεί το σύστημα για την ταχύτητα του υπολογιστή. Όταν χρησιμοποιούμε την SimpleGame πάντα θα έχουμε στο κάτω μέρος της οθόνης μας ενημέρωση για το frame rate καθώς και για τα τρίγωνα και τις κορυφές που εμφανίζονται. Αυτά ελέγχονται μέσω του fpsNode και του fpsText όπου το δεύτερο ανανεώνεται αυτόματα από την μέθοδο update της SimpleGame. Ίσως όμως το πιο σημαντικό αντικείμενο που δημιουργεί η SimpleGame είναι το rootNode. Το rootNode είναι ένας κόμβος χώρου (spatial Node) όπου ορίζει το αρχικό επίπεδο (την ρίζα) της σκηνής που θα πρέπει να σχεδιαστεί. Προσθέτοντας και αλλά Spatial στο rootNode h SimpleGame θα χειριστεί το rendering και το update τους. Η SimpleGame παρέχει μεθόδους που μπορούν να γίνουν override και να έχουμε καλύτερο έλεγχο της εφαρμογής μας. Αυτές οι μέθοδοι είναι:

simpleUpdate: Παρέχει ένα τρόπο για να προσθέτουμε και άλλες λειτουργίες στην φάση αναβάθμισης του βρόχου. Για παράδειγμα αν περιστρέψουμε ένα αντικείμενο η αν ανιχνευτεί σύγκρουση μεταξύ αντικειμένων αυτά καλούνται μετά την update αλλά πριν από γίνει ανανέωση των γεωμετρικών δεδομένων της σκηνής.

simpleRender: Παρέχει τρόπους για να γίνεται αναπαράσταση δεδομένων τα οποία δεν περιέχονται στο δεδομένο rootNode. Για παράδειγμα αν κάνουμε rendering σε ένα texture αυτές τις κλήσης θα τις τοποθετήσουμε στην simpleRender. Καλούνται μετά την render αλλά πριν γίνει το rendering των στατιστικών ώστε να διασφαλίσουμε ακρίβεια στην αναπαράσταση.

DisplaySystem: Έχει δύο κυρίως δουλειές να δημιουργήσει ένα window και τον Renderer. Και τα δυο δημιουργούνται χρησιμοποιώντας την μέθοδο createWindow. Η μέθοδος αυτή παίρνει το μήκος και το πλάτος του παραθύρου (ανάλυση), και συχνότητα της οθόνης και την επιλογή για fullsreen η όχι. Αυτές οι παράμετροι χρησιμοποιούνται για την δημιουργία του παραθύρου και έπειτα την δημιουργία του Renderer που μας παρέχει η OpenGL. Σε αυτό το σημείο το σύστημα μας είναι έτοιμο για Rendering.Η εφαρμογή του SimpleGame δημιουργεί από μόνη της το DisplaySystem και εμείς δεν χρειάζεται να κάνουμε τίποτα.

```
/** Custom Keybinding: Map named actions to inputs. */
  private void initKeys() {
    // You can map one or several inputs to one named action
    inputManager.addMapping("Pause",  new KeyTrigger(KeyInput.KEY_P));
    inputManager.addMapping("Left",   new KeyTrigger(KeyInput.KEY_J));
    inputManager.addMapping("Right",  new KeyTrigger(KeyInput.KEY_K));
    inputManager.addMapping("Rotate", new KeyTrigger(KeyInput.KEY_SPACE),
                                      new MouseButtonTrigger(MouseInput.BUTTON_LEFT));
    // Add the names to the action listener.
    inputManager.addListener(actionListener, new String[]{"Pause"});
    inputManager.addListener(analogListener, new String[]{"Left", "Right", "Rotate"});

  }
```

```
  private ActionListener actionListener = new ActionListener() {
    public void onAction(String name, boolean keyPressed, float tpf) {
      if (name.equals("Pause") && !keyPressed) {
        isRunning = !isRunning;
      }
    }
  };

  private AnalogListener analogListener = new AnalogListener() {
    public void onAnalog(String name, float value, float tpf) {
      if (isRunning) {
        if (name.equals("Rotate")) {
          player.rotate(0, value*speed, 0);
        }
        if (name.equals("Right")) {
          Vector3f v = player.getLocalTranslation();
          player.setLocalTranslation(v.x + value*speed, v.y, v.z);
        }
        if (name.equals("Left")) {
          Vector3f v = player.getLocalTranslation();
          player.setLocalTranslation(v.x - value*speed, v.y, v.z);
        }
      } else {
        System.out.println("Press P to unpause.");
      }
    }
  };
```

Κώδικας 3.2: Custom Input Handlers

3.5 Υποστηριζόμενα Αρχεία jME

Η μηχανή jME υποστηρίζει τους παρακάτω τύπυς αρχείων που υποστηρίζουν 3d μοντέλα, ήχο και video.

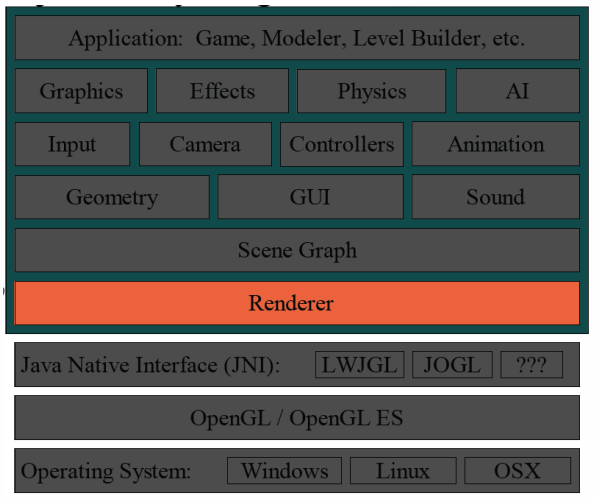
.j3o	Binary 3D model ή scene. Πρέπει να μετατραπούν όλα τα σε .j3o format. Κατά την διάρκεια της ανάπτυξης και όλων των φάσεων της ανάπτυξης του παιχνιδιού μπορεί ο χρήστης να φορτώσει αρχεία OgreXML/OBJ μοντέλα κατευθείαν.
.j3m	Προσαρμοσμένο Υλικό που μπορεί να φτιάξει ο χρήστης. και να τα προσαρμόσει στα 3D μοντέλα
.j3md	A Material definition. These are templates for advanced shader-based Materials Each custom .j3m Material is based on a .j3md Material Definition.

jME File Formats

.mesh.xml, .meshxml, .scene	Blender
.OBJ, .MTL	Wavefront, 3D model format
.JPG, .PNG, .GIF	Αρχεία Φωτογραφίας 2D
.DDS, .HDR, .PFM, .TGA	Αρχεία Texture
.fnt	Bitmap fonts
. wav , . ogg	Αρχεία ήχου
.OGV	Ogg Theora Video

jME external File Formats

3.6 Renderer



Σχήμα 3.2: Αρχιτεκτονική του jME -Renderer

Ο Renderer έχει την δυνατότητα να πάρει ένα screenshot με ότι υπάρχει εκείνη την στιγμή στην οθόνη. Αυτό ουσιαστικά έχει ως προϋπόθεση να ληφθούν τα δεδομένα από το τελευταίο frame που έχει γίνει render και να αποθηκευτούν σαν μια εικόνα png. Μια κλήση στη μέθοδο takeScreenShot είναι αρκετή ώστε να δημιουργηθεί το αρχείο png αποθηκεύοντας το στο τρέχον directory

3.7 Texture Renderer

Ο Texture Renderer παρέχει τα μέσα για να αποδοθεί (render) μια εξ' ολοκλήρου σκηνή σε κάλυμμα (texture) από ότι η σκηνή από μόνη της. Αυτό το Texture τώρα μπορεί να εφαρμοστεί πάνω σε οποιαδήποτε αντικείμενο όπως ένα οποιοδήποτε texture. Η χρησιμότητα του TextureRenderer είναι πανομοιότηπη με του Renderer. Υποστηρίζει την δική του κάμερα επιτρέποντας το σημείο οπτικής να είναι διαφορετικό από αυτό της κύριας σκηνής. Όπως στο Renderer η δημιουργία του TextureRenderer γίνετε μέσω της κλάσης DisplaySystem. Οι σκηνές τότε μπορούν να αποδοθούν σε ένα προμηθευόμενο Texture με μια κλήση στη μέθοδο render

3.8 RenderQueue

Το RenderQueue παρέχει μεθόδους για την οργάνωση των στοιχείων της σκηνής πριν την απεικόνιση τους. Το RenderQueue δεν είναι απαραίτητο και μπορεί να παραληφθεί. Παρ' όλα αυτά αργά η γρήγορα θα χρειαστεί να γίνει κάποια γεωμετρική ταξινόμηση για να αποφευχθούν τυχόν λάθη αναπαράστασης. Για παράδειγμα εξ΄ορισμού στην γεωμετρία γίνεται render με βάση τη θέση του αντικειμένου στο scene-graph. Αν υπάρχουν διαυγή-διάφανα αντικείμενα στην σκηνή και απεικονιστούν πριν τα αδιαφανή αντικείμενα που υποτίθεται ότι είναι από πίσω τους τότε αυτά τα αντικείμενα δεν θα είναι ορατά μέσα από τα διαφανή αντικείμενα. Εδώ είναι που χρειάζεται και η χρήση του RenderQueue. Το RenderQueue χρησιμοποιεί έναν αριθμό από διαφορετικούς "κάδους" για να κάνει ταξινόμηση των αντικειμένων με βάση της παρακάτω 25 ιδιότητες: αδιαφανή, διαφανή και ortho. Ο Renderer προσδιορίζει της σταθερές και για να ορίσουμε την σειρά (Queue) χρησιμοποιούμε την μέθοδο setRenderQueueMode. Π.χ. για να ορίσουμε ένα αντικείμενο obj σαν αδιαφανή θα γράφαμε:

```
obj.setRenderQueueMode(Renderer.QUEUE_TRANSPARENT);
```

Αφού μπουν τα αντικείμενα σε μια σειρά, μπαίνουν σε σειρά προτεραιότητας πριν απεικονιστούν. Η σειρά είναι η ακόλουθη :

1. Απεικονίζονται τα αδιαφανή αντικείμενα (από μπροστά προς τα πίσω).
2. Απεικονίζονται τα διαφανή αντικείμενα από πίσω προς τα μπροστά ζωγραφίζοντας το εσωτερικό του αντικειμένου.
3. Απεικονίζοντας τα διαφανή αντικείμενα (από πίσω προς τα μπροστά) σχεδιάζοντας το εξωτερικό των αντικειμένων.
4. Απεικόνιση αντικειμένων ortho.

Η σειρά που ακολουθείται είναι σημαντική για τους παρακάτω λόγους :

- Απεικονίζοντας πρώτα τα αδιαφανή αντικείμενα διασφαλίζουμε ότι όλα τα αδιαφανή αντικείμενα θα είναι ορατά πίσω από διαφανή αντικείμενα. Απεικονίζοντας από πίσω προς τα μπροστά επιτρέπει στην OpenGL να βελτιστοποιήσει την απεικόνιση παραλείποντας pixels που είναι “υπερφορτωμένα” (occluded)

- Απεικονίζοντας τα διαφανή αντικείμενα από πίσω προς τα μπροστά διασφαλίζει ότι όλα τα αντικείμενα θα είναι ορατά πίσω από ένα διάφανο αντικείμενο

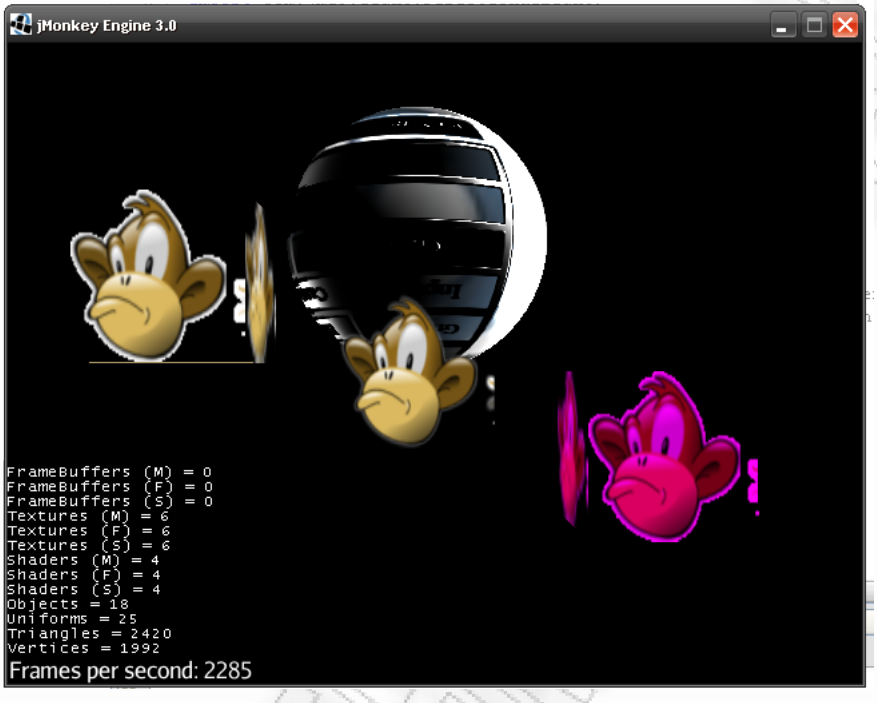
- Απεικονίζοντας τα διαφανή αντικείμενα 2 φορές (μια αποδίδοντας το εσωτερικό και μια το εξωτερικό μέρος) επιτρέπει σε περίπλοκα αντικείμενα να φαίνονται σαν να είναι πραγματικά διάφανα.

- Τα ortho αντικείμενα σχεδιάζονται τελευταία γιατί είναι αντικείμενα που τοποθετούνται στην οθόνη (π.χ. HUD) και όχι στην σκηνή και πρέπει πάντα να βρίσκονται μπροστά από κάθε αντικείμενο σκηνής

```
/* How to make objects transparent, or let colors "leak" through partially
 * transparent textures. How to make bumpy and shiny surfaces.  */

public void simpleInitApp() {

  /** A simple textured cube -- in good MIP map quality. */
  Box boxshape1 = new Box(new Vector3f(-3f,1.1f,0f), 1f,1f,1f);
  Geometry cube = new Geometry("My Textured Box", boxshape1);
  Material mat_stl = new Material(assetManager, "Common/MatDefs/Misc/Unshaded.j3md");
  Texture tex_ml = assetManager.loadTexture("Interface/Logo/Monkey.jpg");
  mat_stl.setTexture("ColorMap", tex_ml);
  cube.setMaterial(mat_stl);
  rootNode.attachChild(cube);

  /** A translucent/transparent texture. */
  Box boxshape31 = new Box(new Vector3f(0f,0f,0f), 1f,1f,0.01f);
  Geometry seethrough = new Geometry("see-through box", boxshape31);
  Material mat_t1t = new Material(assetManager, "Common/MatDefs/Misc/Unshaded.j3md");
  mat_t1t.setTexture("ColorMap",
      assetManager.loadTexture("Textures/ColoredTex/Monkey.png"));
  mat_t1t.getAdditionalRenderState().setBlendMode(BlendMode.Alpha); // activate transparency
  seethrough.setMaterial(mat_t1t);
  seethrough.setQueueBucket(Bucket.Transparent);
  rootNode.attachChild(seethrough);

  /** A cube with base color "leaking" through a partially transparent texture */
  Box boxshape4 = new Box(new Vector3f(3f,-1f,0f), 1f,1f,1f);
  Geometry cube_leak = new Geometry("Leak-through color cube", boxshape4);
  Material mat_tl = new Material(assetManager, "Common/MatDefs/Misc/Unshaded.j3md");
  mat_tl.setTexture("ColorMap",
      assetManager.loadTexture("Textures/ColoredTex/Monkey.png"));
  mat_tl.setColor("Color", new ColorRGBA(1f,0f,1f, 1f)); // purple
  cube_leak.setMaterial(mat_tl);
  rootNode.attachChild(cube_leak);

  /** A bumpy rock with a shiny light effect */
  Sphere rock = new Sphere(32,32, 2f);
  Geometry shiny_rock = new Geometry("Shiny rock", rock);
  rock.setTextureMode(Sphere.TextureMode.Projected); // better quality on spheres
  TangentBinormalGenerator.generate(rock);           // for lighting effect
  Material mat_lit = new Material(assetManager, "Common/MatDefs/Light/Lighting.j3md");
  mat_lit.setTexture("DiffuseMap",
      assetManager.loadTexture("Textures/Terrain/Pond/Pond.png"));
  mat_lit.setTexture("NormalMap",
      assetManager.loadTexture("Textures/Terrain/Pond/Pond_normal.png"));
  mat_lit.setFloat("Shininess", 5f); // [0,128]
  shiny_rock.setMaterial(mat_lit);
  shiny_rock.setLocalTranslation(0,2,-2); // Move it a bit
  shiny_rock.rotate(1.6f, 0, 0);          // Rotate it a bit
  rootNode.attachChild(shiny_rock);

  /** Must add a light to make the lit object visible! */
```

```
    DirectionalLight sun = new DirectionalLight();
    sun.setDirection(new Vector3f(1,0,-2).normalizeLocal());
55  sun.setColor(ColorRGBA.White);
    rootNode.addLight(sun);}}
```

Κώδικας 3.3: Material Edititng

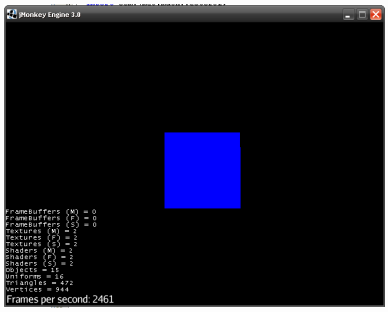


Σχήμα 3.3: Τροποποιήση των Υλικών στο JME

3.9 Βασικά Σχήματα

Το JME παρέχει μέσα για την δημιουργία βασικών σχημάτων. Έτσι μας επιτρέπει να βάλουμε πολύ εύκολα και γρήγορα 3D σχήματα στην σκηνή μας. Κάποια από τα βασικά σχήματα είναι:

Box



Σχήμα 3.4: Βασικά Σχήματα-Box

Υπάρχουν 2 τρόποι για να δημιουργήσουμε ένα κουτί

A) με min και max points
Vector3f min = new Vector3f (-5,-5,-5);
Vector3f max == new Vector3f (5,5,5);
Box b = new Box("box",min,max);

B) ορίζοντας κέντρο και πλευρές
Box c = new Box("box" , new Vextor3f(-10,-10,-10),1000,1,1000);
Όπου Vextor3f(-10,-10,-10) είναι η γωνία του κουτιού και τα επόμενα τρία ορίσματα που ακολουθούν είναι το μήκος, το ύψος και το πλάτος αντίστοιχα.

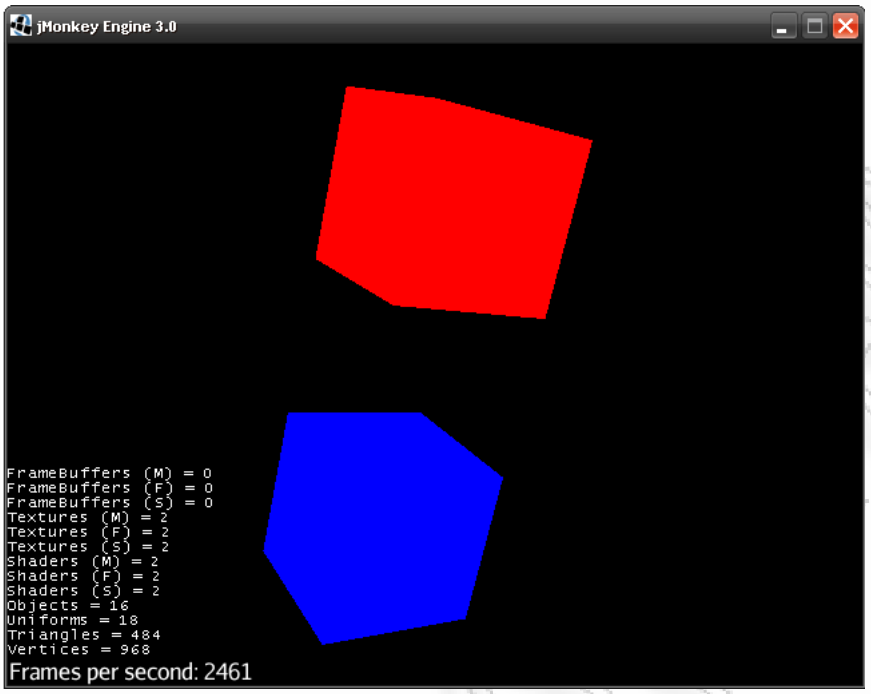
```
1     public void simpleInitApp() {
          Box b = new Box(Vector3f.ZERO, 1, 1, 1);
          Geometry geom = new Geometry("Box", b);
4         Material mat = new Material(assetManager, "Common/MatDefs/Misc/Unshaded.j3md");
          mat.setColor("Color", ColorRGBA.Blue);
          geom.setMaterial(mat);
7         rootNode.attachChild(geom);
      }
```

Κώδικας 3.4: Box

Όπως επίσης μπορούμε να τοποθετήσουμε δυο κύβους στο παράδειγμα μας αλλάζοντας το χρώμα και την θέση τους ωστε να φαίνονται στην σκηνή.

```
1  /* How to use nodes as handles to manipulate objects in the scene graph.
    * You can rotate, translate, and scale objects by manipulating their parent nodes.
    * The Root Node is special: Only what is attached to the Root Node appears in the scene. */
4      public void simpleInitApp() {
           // create a blue box at coordinates (1,-1,1)
           Box box1 = new Box( new Vector3f(1,-1,1), 1,1,1);
7          Geometry blue = new Geometry("Box", box1);
           Material mat1 = new Material(assetManager, "Common/MatDefs/Misc/Unshaded.j3md");
           mat1.setColor("Color", ColorRGBA.Blue);
10         blue.setMaterial(mat1);
           // create a red box straight above the blue one at (1,3,1)
           Box box2 = new Box( new Vector3f(1,3,1), 1,1,1);
13         Geometry red = new Geometry("Box", box2);
           Material mat2 = new Material(assetManager, "Common/MatDefs/Misc/Unshaded.j3md");
           mat2.setColor("Color", ColorRGBA.Red);
16         red.setMaterial(mat2);
           // create a pivot node at (0,0,0) and attach it to root
           Node pivot = new Node("pivot");
19         rootNode.attachChild(pivot);
           // attach the two boxes to the *pivot* node!
           pivot.attachChild(blue);
22         pivot.attachChild(red);
           // rotate pivot node: Both boxes have rotated!
           pivot.rotate( 0.4f , 0.4f , 0.0f );
25     }
}
```

Κώδικας 3.5: 2 Boxes



Σχήμα 3.5: Βασικά Σχήματα(Box)-Δυο Κύβοι με διαφορετικό χρώμα

Cylinder

Ένας άπειρος κύλινδρος ορίζεται ως ένα σύνολο σημείων και μία σταθερή απόσταση από μία γραμμή. Ένας πεπερασμένος κύλινδρος είναι ένα υποσύνολο του άπειρου κυλίνδρου όπου το μήκος του είναι λιγότερο από ένα ορισμένο σημείο. Στο JME ο κύλινδρος είναι πάντα ένας πεπερασμένος κύλινδρος. Η δημιουργία ενός κυλίνδρου απαιτεί το ύψος και την ακτίνα. Ο αριθμός των τμημάτων που αποτελούν το ακτινωτό των κυλίνδρου καθώς και το ύψος παρέχονται από μας. Όσο μεγαλύτερος ο αριθμός των τμημάτων τόσο πιο λεπτομερής θα είναι και ο κύλινδρος. Ο κύλινδρος προσανατολίζεται πάντα ξαπλώνοντας κατά μήκος του άξονα Z.

Cylinder c = new Cylinder(“Cyl” ,10,10,5,20);

10 είναι τα τμήματα για το ύψος και το ακτινωτό.
5 είναι η ακτίνα άρα ο κύλινδρος θα είναι διαμέτρου 10.
20 είναι το ύψος.

```
1     public void simpleInitApp() {
          Cylinder v = new Cylinder(10, 10, 1, 2);
          Geometry geom = new Geometry("Cyl", c);
4         Material mat = new Material(assetManager, "Common/MatDefs/Misc/Unshaded.j3md");
          mat.setColor("Color", ColorRGBA.Red);
          geom.setMaterial(mat);
7         rootNode.attachChild(geom);
      }
}
```

Κώδικας 3.6: Cylinder

Sphere

Μια σφαίρα καθορίζεται από ένα σύνολο σημείων που ισαπέχουν από ένα κεντρικό σημείο. Στο jme η σφαίρα καθορίζεται από ένα κεντρικό σημείο, μια ακτίνα και έναν αριθμό δειγμάτων για τον άξονα Z και τον ακτινωτό άξονα. Οσο μεγαλύτερος ο αριθμός δειγμάτων (ή αριθμός τριγώνων) τόσο καλύτερη η ανάλυση της σφαίρας,
Δημιουργία σφαίρας

Sphere s = new Sphere(“Sphere”,63,50,25);

```
    public void simpleInitApp() {
        Sphere s = new Sphere(63,50,25, true, true);
        Geometry geom = new Geometry("Box", s);
        Material mat = new Material(assetManager, "Common/MatDefs/Misc/Unshaded.j3md");
        mat.setColor("Color", ColorRGBA.Blue);
        geom.setMaterial(mat);
        rootNode.attachChild(geom);
    }
}
```

Κώδικας 3.7: Sphere

3.10 Μαθηματικά στο jME

Τα μαθηματικά παίζουν σημαντικό ρόλο και στο jme γιατί αναμφισβήτητα τα 3D γραφικά απαιτούν αρκετές γνώσεις μαθηματικών. Ωστόσο το jme έχει την δυνατότητα και απαλλάσσει τον χρήστη από πολλές λεπτομέρειες καλύπτοντας ένα μεγάλο μέρος των μαθηματικών "υψηλού επιπέδου" Θεμελιώδεις τύποι αναλαμβάνουν την αντιπροσώπευση σύνθετων μαθηματικών εννοιών και την λειτουργία τους χωρίς να χρειάζεται από μας κάποια ιδιαίτερη εμβάθυνση. Τα διανύσματα αποτελούν στοιχειώδες τύπο σε 3D περιβάλλον και χρησιμοποιούνται κατά κόρον. Οι πίνακες είναι επίσης ένα βασικό στοιχείο για την απεικόνιση γραμμικών συστημάτων. Το Quaternion είναι ίσως το πιο περίπλοκο από τους βασικούς τύπους και στο jme χρησιμοποιείται για την περιστροφή αντικειμένων. Τα Vectors (διανύσματα) χρησιμοποιούνται για να αντιπροσωπεύσουν κάποιο σημείο ή διεύθυνση ενώ οι πίνακες χρησιμοποιούνται συνήθως για περιστροφές σε συνδυασμό με το quaternion . Η κλάση Vector3f είναι ίσως η πιο πολυχρησιμοποιημένη κλάση στο jme.

3.11 Quaternions

Το jme χρησιμοποιεί τα quaternions επειδή επιτρέπουν τις συμπαγείς αντιπροσωπεύσεις των περιστροφών ή αντίστοιχα τους προσανατολισμούς στον τρισδιάστατο χώρο. Με τέσσερις μόνο float τιμές μπορούμε να αναπαραστήσουμε τον προσανατολισμό ενός αντικειμένου όταν ένας πίνακας περιστροφής θα απαιτούσε 9. Ενώ είναι αρκετά δύσκολο να κατανοηθεί πλήρως η έννοια του quaternion υπάρχει ένας μεγάλος αριθμός πιστικών μεθόδων που μας επιπρέπει να τις χρησιμοποιούμε χωρίς να είναι απαραίτητο να κατανοούμε όλα τα μαθηματικά που κρύβονται από πίσω. Βασικά αυτές οι μέθοδοι δεν περιλαμβάνουν τίποτα περισσότερο από το να θέτουν τις τιμές των quaternion x,y,z,w χρησιμοποιώντας άλλα μέσα για της περιστροφές.

3.12 Angle Axis

Μπορούμε εάν θέλουμε να ορίσουμε την περιστροφή μας ως προς ένα ζευγάρι άξονα γωνίας (Angle Axis) δηλαδή καθορίζουμε έναν άξονα περιστροφής και την γωνία περιστροφής γύρο από αυτόν τον άξονα. Το quaternion ορίζει μια μέθοδο την fromAngleAxis και την fromAngleNormalAxis ώστε να δημιουργήσει ένα quaternion από αυτό το ζευγάρι. Μπορούμε επίσης να προκαλέσουμε περιστροφή από ένα ήδη υπάρχων quaternion χρησιμοποιώντας την μέθοδο toAngleAxis.
Παράδειγμα περιστροφής χρησιμοποιώντας την fromAngleAxis:
περιστροφή στον άξονα Y κατά π

Vector3f axis = new Vector3f(0,1,0);
Float angle = 3,14f;
s.getLocalRotation().fromAngleAxis(angle,axis);

3.13 Three Angles

Μπορούμε επίσης να περιστρέψουμε ένα αντικείμενο καθορίζοντας τρεις γωνίες. Οι γωνίες αντιπροσωπεύουν την περιστροφή του κάθε άξονα. Παίρνοντας σαν όρισμα έναν πίνακα τριών

στοιχείων float ορίζουν την γωνία, όπου το πρώτο στοιχείο είναι το x, το δεύτερο το y και το τρίτο το z. Η μέθοδος που παρέχεται από το quaternion είναι η fromAnlges και μπορούμε επίσης να γεμίσουμε έναν πίνακα χρησιμοποιώντας την toAngles.

Παράδειγμα περιστροφής χρησιμοποιώντας την fromAngles:
περιστροφή 1 rad στον x, 3 στον y και 0 στον z
```
float[] angles = 1, 3, 0;
s.getLocalRotation().fromAngles(angles);
```

3.14 Three Axes

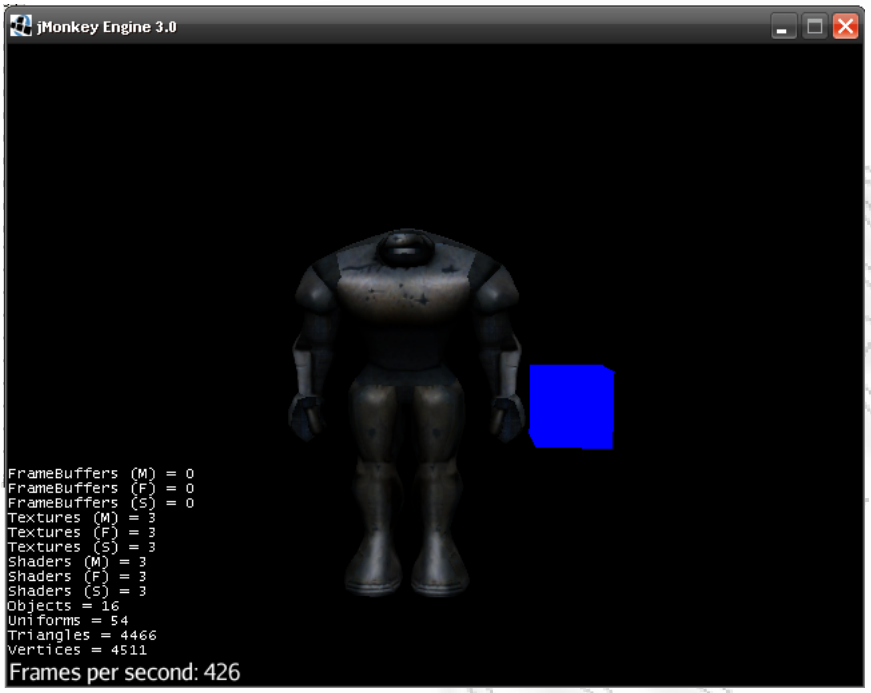
Εάν έχουμε 3 άξονες για να καθορίζουν την περιστροφή μας τότε οι άξονες θα καθορίζουν τον αριστερό άξονα, τον επάνω άξονα και τον κατευθυντικό άξονα αντίστοιχα. Σε αυτήν την περίπτωση μπορούμε να χρησιμοποιήσουμε την fromAxes για να δημιουργήσουμε το quaternion. Πρέπει να σημειώσουμε ότι σε αυτήν την περίπτωση θα δημιουργηθεί ένας καινούργιος πίνακας όπου μετά θα είναι άχρηστος, γι΄αυτό και αυτή η μέθοδος δεν είναι καλό να χρησιμοποιείται εάν είναι να κληθεί πολλές φορές.

3.15 Visibility Determination

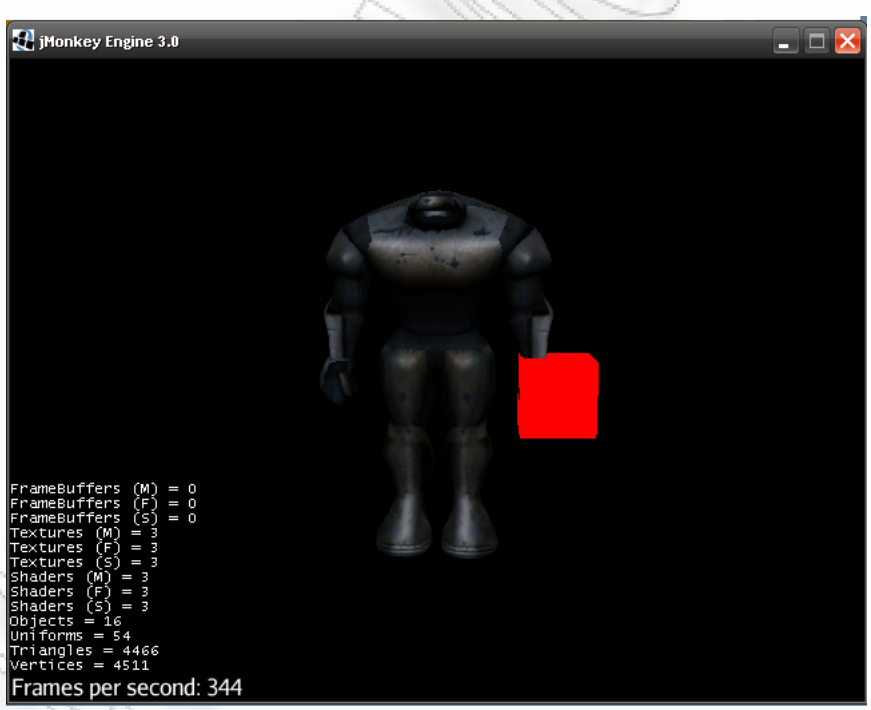
Ο προσδιορισμός την εμβέλειας των ορατών στοιχείων ασχολείται με την ελαχιστοποίηση των δεδομένωντων στοιχείων που στέλνονται στην κάρτα γραφικών για να αποδοθούν στην οθόνη. Συγκεκριμένα δεν θέλουμε να στέλνουμε στοιχεία και δεδομένα τα οποία δεν πρόκειται να εμφανιστούν. Τα στοιχεία τα οποία δεν στέλνονται στην κάρτα γραφικών χαρακτηρίζονται ως culled. Η αρχική εστίαση αυτού του τμήματος είναι το frustrum Culling που βασίζεται στο οπτικό frustrum της κάμερας.Ουσιαστικά αυτό το frustrum δημιουργεί έξι τυποποιημένα επίπεδα οπτικής. Εξετάζονται τα Bounding Volumes των αντικειμένων για να ελεγχθεί εάν τα όρια τους αυτά βρίσκονται εντός εμβέλειας του frustrum. Εάν σε οποιοδήποτε σημείο η οριοθέτηση του αντικειμένου είναι έξω από το επίπεδο που ορίσαμε, πετιέται έξω και δεν επεξεργάζονται πλέον ώστε να αποδοθούν στην σκηνή. Αυτό περιλαμβάνει και τυχόν “παιδία” τα οποία διαχειρίζονται αυτό επιτρέπει γρήγορα culling για μεγάλα τμήματα σκηνής.

3.16 Collision Calculations

Το jme παρέχει ένα εκτενές σύστημα ανίχνευσης συγκρούσεων. Χρησιμοποιώντας το Bounding Volumes μπορούν να γίνει ελεγχος για το εάν δυο αντικείμενα συγκρούονται η όχι.



Σχήμα 3.6: Collision-Πριν τη Σύγκρουση:Μπλε Χρώμα



Σχήμα 3.7: Collision-Μετα τη Σύγκρουση:Κόκκινο Χρώμα

```
    public void simpleInitApp() {
        bulletAppState = new BulletAppState();
        stateManager.attach(bulletAppState);
        Material mat = new Material(getAssetManager(), "Common/MatDefs/Misc/WireColor.j3md");
        mat.setColor("m_Color", ColorRGBA.Red);
        Material mat2 = new Material(getAssetManager(), "Common/MatDefs/Misc/WireColor.j3md");
        mat2.setColor("m_Color", ColorRGBA.Magenta);

        // Add a physics sphere to the world
        PhysicsNode physicsSphere=new PhysicsNode(new SphereCollisionShape(1),1);
```

```
        physicsSphere.setLocalTranslation(new Vector3f(3,6,0));
        physicsSphere.attachDebugShape(assetManager);
        rootNode.attachChild(physicsSphere);
        getPhysicsSpace().add(physicsSphere);

        // Add a physics sphere to the world using the collision shape from sphere one
        PhysicsNode physicsSphere2=new PhysicsNode(physicsSphere.getCollisionShape(),1);
        physicsSphere2.setLocalTranslation(new Vector3f(4,8,0));
        physicsSphere2.attachDebugShape(mat2);
        physicsSphere2.addCollideWithGroup(PhysicsNode.COLLISION_GROUP_02);
        rootNode.attachChild(physicsSphere2);
        getPhysicsSpace().add(physicsSphere2);

        // an obstacle mesh, does not move (mass=0)
        PhysicsNode node2=new PhysicsNode(new MeshCollisionShape(new Sphere(16,16,1.2f)),0);
        node2.setLocalTranslation(new Vector3f(2.5f,-4,0f));
        node2.attachDebugShape(mat);
        node2.setCollisionGroup(PhysicsNode.COLLISION_GROUP_02);
        node2.setCollideWithGroups(PhysicsNode.COLLISION_GROUP_02);
        rootNode.attachChild(node2);
        getPhysicsSpace().add(node2);

        // the floor, does not move (mass=0)
        PhysicsNode node3=new PhysicsNode(new MeshCollisionShape(new
            Box(Vector3f.ZERO,100f,0.2f,100f)),0);
        node3.setLocalTranslation(new Vector3f(0f,-6,0f));
        node3.attachDebugShape(assetManager);
        rootNode.attachChild(node3);
        getPhysicsSpace().add(node3);
    }

    private PhysicsSpace getPhysicsSpace(){
        return bulletAppState.getPhysicsSpace();
    }

    @Override
    public void simpleUpdate(float tpf) {
        //TODO: add update code
    }

    @Override
    public void simpleRender(RenderManager rm) {
        //TODO: add render code
    }
```

Κώδικας 3.8: Collision

Κεφάλαιο 4

jME Advanced

4.1 Camera

Το Jme χρησιμοποιεί την κάμερα για να περιγράψει διάφορους βασικούς όρους σχετικά με το πώς γίνεται render στο scenegraph. Όπως και σε μια πραγματική φωτογραφική μηχανή η θέση και ο φακός καθορίζουν πόσο και ποιο μέρος του πραγματικού κόσμου θα καταγραφεί για να αποδοθεί αργότερα. Έτσι και εδώ η κάμερα του jme καθορίζει πόσος από τον εικονικό μας κόσμο θα αποδοθεί και θα επιδειχθεί στο χρήστη. Την κάμερα μπορούμε να την χειριστούμε όπως οποιοδήποτε άλλο αντικείμενο πάνω στην σκηνή. Αυτό απλά απαιτεί να κάνουμε χρήση του CameraNode. Η τοποθεσία και οι ρυθμίσεις τις κάμερας μας καθορίζουν το λεγόμενο View Frustrum το οποίο είναι μια λογική περιγραφή του συνολικού όγκου του χώρου που είναι ορατός στον χρηστή. Εάν παίρναμε μια πυραμίδα, κόβαμε το πάνω μέρος της και κοιτάζαμε την πυραμίδα από την κορυφή προς την βάση αυτό που βλέπουμε είναι αυτό που ορίζεται ως view frustrum. Η νέα διαμόρφωση κορυφής η οποία δημιουργείται αντιπροσωπεύει την οθόνη του χρήστη. Η πυραμίδα επεκτείνεται προς τα έξω διευρύνοντας το τοπικό πεδίο. Οτιδήποτε βρίσκεται εντός της πυραμίδας γίνεται render ενώ οτιδήποτε βρίσκεται εκτός αγνοείται. Αυτό δίνει την δυνατότητα στο Jme να χειρίζεται σύνθετα scenegraph με πολλά αντικείμενα διατηρώντας ωστόσο υψηλό Frame Per Second (fps). Οι ρυθμίσεις της κάμερας μας θα καθορίσουν ποιο θα είναι το view Frustrum, ο προσανατολισμός της στον κόσμο και ποιες οι ιδιότητες των πλευρών της π.χ. οι πλευρές εκτείνονται με τέτοιο τρόπο που να δημιουργούν μια μεγάλη γωνία δίνοντας στον χρήστη ένα πολύ ευρύ οπτικό πεδίο ή μένουν σχετικά κοντά δημιουργώντας την αίσθηση ενός τούνελ στην σκηνή; Επιπλέον η κάμερα επιτρέπει στο χρήστη να ελέγχει ποιο είναι το frustrum και ποιος ο προσανατολισμός του. Στο jme δεν θα πρέπει ποτέ να δημιουργούμε άμεσα μια κάμερα έτσι ώστε να αποφύγουμε τυχόν εξαρτήσειςαπό κάποιο συγκεκριμένο API. Εναλλακτικά ζητάμε μια κάμερα από τον Renderer. Η μόνη παράμετρος που απαιτείται είναι η ανάλυση που θα αποδοθεί η εικόνας μας ώστε να έχουμε ακριβή απόδοση.

4.2 View Frustrum

Στο jme το frustrum μέσα στην κλάση της κάμερας αποτελείται από έξι επίπεδα (top, bottom, front,back,left,right). Υπάρχουν δυο τρόποι για να ορίσουμε το View Frustrum.

1. public void setFrustum(float near, float far, float left, float right, float top, float bottom);
   χρησιμοποιώντας την setFrustrum ο χρήστης καθορίζει τα έξη επίπεδα οπτικής του frustrum. Αυτά τα επίπεδα ορίζονται ως η απόσταση από το σημείο του ματιού.
   Για παράδειγμα το :
   cam.setFrustum(1.0f, 1000.0f, -0.55f, 0.55f, 0.4125f, -0.4125f);
   ορίζει το κοντινό επίπεδο ως μια μονάδα από το σημείο του ματιού, το μακρινό επίπεδο ως 1000 μονάδες, το left και right ως 0.55 (κάνοντας έτσι το κοντινό επίπεδο να έχει πλάτος 1,1) και το top και botton από 0,4125 (κάνοντας το κοντινό επίπεδο ύψους 0,85)

2. public void setFrustumPerspective(float fovY, float aspect, float near, float far);
   όπου το foy(Y) ορίζει την γωνία οπτικής στην κατεύθυνση Y. Το aspect καθορίζει την σχέση μεταξύ του ύψους και του πλάτους του παραθύρου και το near και far καθορίζουν το κοντινό και το μακρινό επίπεδο.

Για παράδειγμα
cam.setFrustumPerspective(45.0f,(float) display.getWidth() / (float) display.getHeight(), 1, 1000);

4.3 Scenegraph

Μια αναπαράσταση σκηνής (scenegraph) αποτελείται από δύο τύπους κόμβων. Τους εσωτερικούς (Internal Nodes) και τους φυλλώδες (Leaf Nodes). Οι εσωτερικοί κομβόι αναφέρονται και ως Nodes και οι LeafNodes αναφέρονται ως Geormetry (γεωμετρίες). Η διαφορά των δύο αυτών είναι ότι τα Nodes μπορούν να περιέχουν παιδία (π.χ. άλλο κόμβο ή άλλη γεωμετρία) ενώ οι γεωμετρίες δεν μπορούν να έχουν παιδιά, ενώ και οι δυο τύποι περιέχουν σημαντικές πληροφορίες για τον εαυτό τους (transforms,Boounding Volumes, renderState και controllers).

4.4 Spatial Scenes

Το Spatial καθορίζει την βασική κλάση για όλα τα στοιχεία στο scenegraph. Έχουμε δύο κατηγορίες Spatial τα Geometry και Nodes. Οι πληροφορίες του scenegraph δημιουργούνται σε 2 φάσεις – περάσματα. Στο πρώτο πέρασμα που γίνεται από πάνω προς τα κάτω γίνονται όλες οι μετατροπές (transforms) και στο δεύτερο πέρασμα πουυ γίνεται από κάτω προς τα πάνω γίνονται οι οριοθετήσεις.

4.5 Camera Node

Η CameraNode μας επιτρέπει να χρησιμοποιήσουμε ο αντικείμενο Camera σαν στοιχείο του scenegaph. Το όφελος από αυτό είναι ότι μπορούμε να προσκολλήσουμε την Camera μας σε οποιοδήποτε αντικείμενος την σκηνής. Ο προσανατολισμός της CameraNode καθορίζει και τον προσανατολισμό της camera. Κατά την κλήση της UpdateWorldData η CameraNode θα εξασφαλίσει την ανανέωση του οπτικού πεδίου της κάμερας. Κατά συνέπεια αν καλέσουμε την updateGeometricState από την σκηνή πάνω στην οποία είναι προσκολλημένη η CameraNode θα ανανεωθεί το οπτικό πεδίο.

4.6 Model Bound

Τα όρια ενός μοντέλου καθορίζονται από το Boundong Volume που περιέχει τα γεωμετρικά δεδομένα προέλευσης του. Τα όρια αυτά δεν μεταφράζονται σε χώρος τον κόσμο μας αλλά κρατιούνται στο χώρο γεωμετρίας. Αυτά τα όρια χρησιμοποιούνται για να μετρήσουμε τα όρια των gometry και κατ’ επέκταση τα όρια των κόμβων.

4.7 Bounding Volumes

Το Bounding Volumes χρησιμοποιείται για την αφαίρεση περιττών αντικειμένων πριν το render της σκηνής και καθορίζει εάν ένα αντικείμενο βρίσκεται μέσα στο οπτικό πεδίο μιας κοινότητας. Για σύνθετα αντικείμενα όπως ένας ανθρώπινος οργανισμός είναι πολύ δύσκολο να καθοριστεί υπολογίστηκα εάν αυτό το αντικείμενο βρίσκεται μέσα στο οπτικό που μας ενδιαφέρει δεδομένου ότι όλες οι δοκιμές θα έπρεπε να γίνουν για κάθε πολύγωνο που υπάρχει στο μοντέλο. Έτσι ένας αόρατος απλός μαθηματικός όγκος όπως μια σφαίρα ή ένας κύβος τοποθετείται γύρο από το πρότυπο με τέτοιο τρόπο ώστε να περιλαμβάνει όλη την γεωμετρία του σύνθετου προτύπου. Όταν λείπουν γίνεται η διαδικασία του culling στο scenegraph αυτό το απλό αντικείμενο εξετάζεται για το αν θα είναι ορατό ή όχι πριν σταλεί για rendering. Εάν το απλό αυτό αντικείμενο αποτύχει το τεστ (δεν βρίσκεται μέσα στο οπτικό πεδίο της κάμερας) τότε το σύνθετο μοντέλο και όλα του τα παιδιά απορρίπτονται από την διαδικασία του render κερδίζοντας έτσι δύναμη στον επεξεργαστή

4.8 CullState

Καθορίζει ένα RenderState [i] που χειρίζεται την πρόσοψη ενός συγκεκριμένου τριγώνου. Υπάρχουν τρεις επιλογές : CS_FRONT, CS_BACK, CS_NONE. Η πιο συνηθισμένη χρήση του CullState είναι η CS_BACK (αφαίρεση του πίσω μέρους των αντικειμένων). Με αυτόν τον τρόπο επιτρέπει στον Renderer να μην υπολογίζει τα τρίγωνα τα οποία δεν βλέπουν προς την κάμερα αυτό μπορεί να έχει σαν αποτέλεσμα σημαντική αύξηση της ταχύτητας. Το CullState χειρίζεται όπως οποιαδήποτε άλλο RenderSate και μπορεί να τοποθετηθεί οπουδήποτε στην σκηνή μας. Εξ' ορισμού καμία πλευρά δεν απορρίπτεται.

4.9 Lightstate

Παρέχει τα μέσα ώστε να μπορεί να προστεθεί φωτισμός σε μια γραφική σκηνή. Ενα lightState μπορεί να περιέχει καθόλου ή πολλά φώτα. Το LigthState μπορούμε να το χειριστούμε καλύτερα χρησιμοποιώντας το μέσο ενός LightNode. Με το LightNode μπορούμε αν χειριστούμε το φως σαν να ήταν αντικείμενο της σκηνής.

4.10 Buffer State

Το ZBufferState βοηθάει στο rendering. Αυτό που κάνει είναι να κρατάει πληροφορίες για το κάθε pixel π.χ. το χρώμα, το πόσο μακριά βρίσκεται αυτό το pixel καθώς και αν μπορεί να το κάνει override ή όχι. Το τελευταίο εξαρτάται από το τι ορίσματα θα του δώσουμε εμείς.

- CF_NEVER - το πρώτο pixel που τοποθετείται να μην καλυφθεί από κανένα επόμενο.
- CF_LESS – έχουμε αντικατάσταση pixel αν το καινούργιο pixel βρίσκεται πιο κοντά (σε βάθος) από τα τρέχον.
- CF_EQUAL – γίνεται αντικατάσταση pixel αν το καινούργιο pixel βρίσκεται στο ίδιο βάθος με το ήδη υπάρχον.
- CF_LEQUAL - γίνεται αντικατάσταση pixel αν το καινούργιο pixel βρίσκεται σε μικρότερο ή ίσο βάθος από το ήδη υπάρχον.
- CF_GREATER - γίνεται αντικατάσταση pixel αν το καινούργιο pixel βρίσκεται πιο μακριά από το τρέχον pixel.
- CF_NOTEQUAL - γίνεται αντικατάσταση pixel αν το καινούργιο pixel δεν βρίσκεται στο ίδιο βάθος από το ίση υπάρχον.
- CF_GEQUAL - γίνεται αντικατάσταση pixel αν το καινούργιο pixel βρίσκεται σε ίσο η μεγαλύτερο βάθος από το ήδη υπάρχον.
- CF_ALWAYS – πάντα το καινούργιο pixel αντικαθιστά τα ήδη υπάρχον.

[i] Ενώ η γεωμετρία καθορίζει τα δεδομένα που πρόκειται να γίνουν render το RenderState καθορίζει την εμφάνιση των δεδομένων.

Κεφάλαιο 5

Java Physics

5.1 jME Physics

Το σύστημα του jME physics παρέχει ένα interface ανάμεσα στο jME και στην ODE (Open Dynamic Engine). Τοποθετείται πάνω από μια ελαφρός τροποποιημένη έκδοση της odejava και παρέχει τρόπους για να δημιουργηθεί πολύ εύκολα ένας κόσμος φυσικής και να προστεθούν σε αυτόν αντικείμενα. Μπορούμε να έχουμε κάτι πολύ απλό όπως ένα κουτί να εκτελεί μια ελεύθερη πτώση έως κάτι πολύ σύνθετο π.χ. ένα αυτοκίνητο ή έναν άνθρωπο. Η αρχική ιδέα ήταν του Nicolaas de Bruyn που έβαλε τις βάσεις ενώ αργότερα προστέθηκαν και οι Per Thulin και Ahmed Al-Hindawi σε μια προσπάθεια για ανάπτυξη και πρόσθεση νέων χαρακτηριστικών.

5.2 ODE

Είναι μια open source βιβλιοθήκη φυσικής γραμμένη σε C. Με άλλα λόγια είναι μια ελεύθερη βιβλιοθήκη για μιμήσεις αυστηρά δυναμικής σωμάτων. Για παράδειγμα επίγειων οχημάτων, πλασμάτων με πόδια καθώς και κίνηση αντικειμένων σε εικονικές πραγματικότητες. Η ODE είναι γρήγορη, εύκαμπτη, ανεξάρτητη πλατφόρμας με προηγμένες ενώσεις (joints), επαφές με τριβή, ανίχνευση συγκρούσεων κ.α. Εν ολίγοις είναι μια βιβλιοθήκη κατάλληλη για φυσικές προσομοιώσεις.

5.3 ODE Java

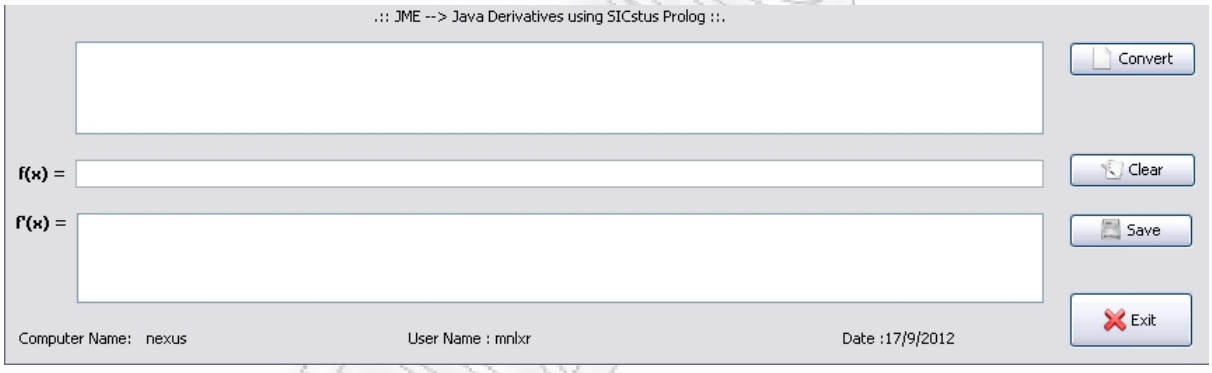
Είναι μια open dynamic Engine αποκλειστικά για java. Το API παρουσιάζει πρόσβαση, και χαμηλού επιπέδου στην βιβλιοθήκη της ODE και πρόσβαση υψηλού επιπέδου στα αντικείμενα της odejava. Καθώς η βιβλιοθήκη είναι γραμμένη σε C δεν είναι αντικειμενοστραφής και γι' αυτό το λόγο χρειάστηκε να δημιουργηθούν τα αντικείμενα υψηλότερου επιπέδου. Η odejava μπορεί να χρησιμοποιηθεί σε οποιαδήποτε εφαρμογή γιατί είναι μια ανεξάρτητη βιβλιοθήκη. Η πιο κοινή χρήση της odejava είναι σε 3D εφαρμογές.

Κεφάλαιο 6

Prolog Derivative

6.1 Prolog Derivative GUI

Η πρώτη επαφή του χρήστη είναι το γραφικό περιβάλλον για την εύρεση της παραγώγου. Υπάρχει μενου με τις βασικές λειτουργίες όπου ο χρήστης μπορεί να αποθηκεύσει τα αποτελέσματα(σε μορφή *.txt) και να τα επεξεργαστεί, να κάνει την μετατροπή και να καθαρίσει το πεδίο που δίνει τις συναρτήσεις (Μήκος – 4, Βάθος - 2). Στην είσοδο (Text Field) δίνει ο χρήστης την συνάρτηση που θέλει για να βρεθεί η παράγωγος.



Σχήμα 6.1: Prolog Derivative - GUI

Έστω η συνάρτηση:

$$f(x) = x^2 \tag{6.1}$$

Η παράγωγος της σχέσης 6.1 είναι :

$$f'(x) = 2 * x^1 \tag{6.2}$$

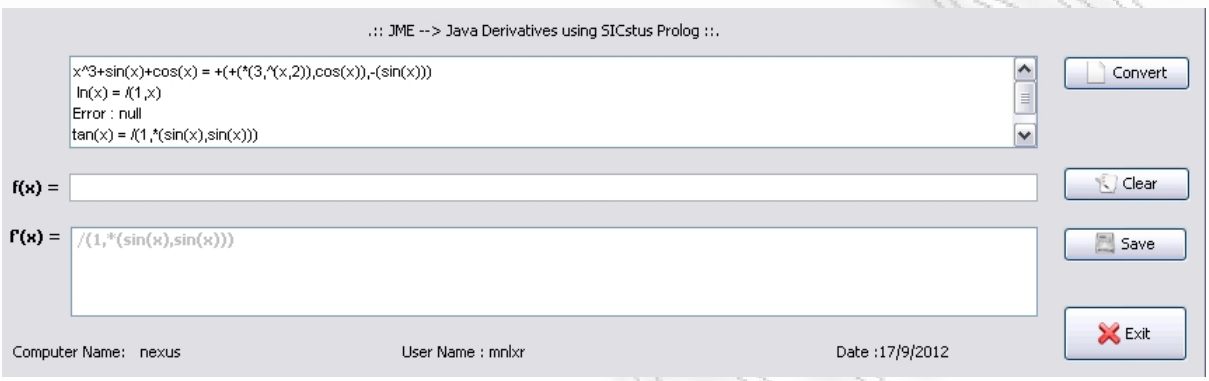
Η prolog μας δίνει απότελεσμα:

$$f'(x) = *(2,^(x,1)) \tag{6.3}$$

Μας δίνει την παράγωγο της 6.1 σε προθεματική γραφή(πολωνική). Η προτεραιότητα των τελεστών της Prolog είναι :

```
:- op( 1200, xfx, [ :-, --> ]).
:- op( 1200, fx, [ :-, ?- ]).
:- op( 1150, fx, [ mode, public, dynamic, volatile,
discontiguous, multifile, block,
meta_predicate,initialization ]).
:- op( 1100, xfy, [ ; ]).
:- op( 1050, xfy, [ -> ]).
:- op( 1000, xfy, [ ',' ]).
:- op( 900, fy, [ \+, spy, nospy ]).
```

```
:- op( 700, xfx, [ =, \=, is, =.., ==, \==, @<, @>, @=<, @>=,
=:=, =\=, <, >, =<, >= ]).
:- op( 550, xfy, [ : ]).
:- op( 500, yfx, [ +, -, \, /\, \/ ]).
:- op( 400, yfx, [ *, /, //, mod, rem, <<, >> ]).
:- op( 200, xfx, [ ** ]).
:- op( 200, xfy, [ ^ ]).
:- op( 200, fy, [ +, -, \ ]).
```



Σχήμα 6.2: Prolog Derivative - Εισαγωγή συναρτήσεων

Όποτε έχουμε από την εξαγόμενη συνάρτηση 6.3 $\rightarrow f'(x) = *(2,(x,1))$

$f(x) = x^n \rightarrow f'(x) = n * x^n - 1$
$df(x^N, N * x^M) : -MisN - 1, !$
Ανίστοιχα για την

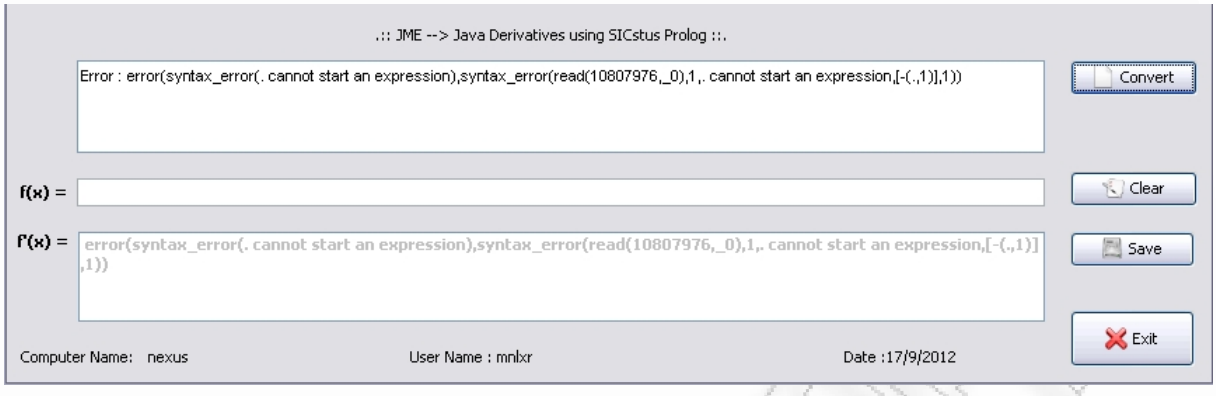
$$f(x) = x * x \tag{6.4}$$

έχουμε $f'(x) = 1 * x + x * 1$ Δίνοντας σαν παράμετρο στην είσοδο του Prolog-Deriv GUI (από εδώ και πέρα θα το λέμε GUI) στο TextField παίρνουμε

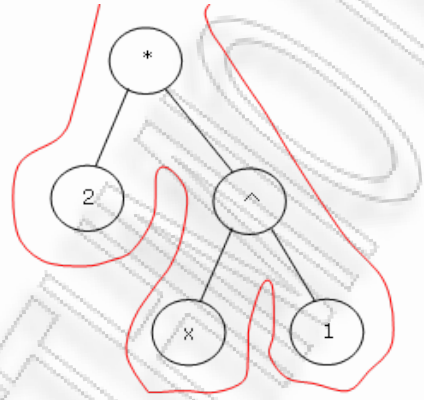
$$fp1'(x) = +(*(1,x),*(x,1)) \tag{6.5}$$

Η αντιστοίχία σε δυαδικό δένδρο είναι (Εδω πρέπει να αναφέρουμε ότι η αντιστοιχία ανάμεσα στην παράγωγο και το δυαδικό δένδρο είναι **αμφιμονοσήμαντη**):

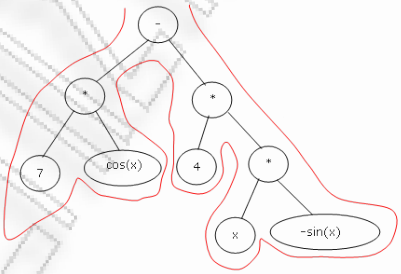
Έστω η συνάρτηση :$f(x) = 3 * sin(x) + 4 * x * cos(x)$.
Έχουμε $fp2'(x) = +(+(*(0,sin(x)),*(3,cos(x))),+(*(+(*(0,x),*(4,1)),cos(x)),*(*(4,x),-(sin(x)))))$
$fp2'(x) = 0 * sin(x) + 3 * cos(x) + ((0 * x + 4 * 1) * cos(x) + 4 * x * -(sin(x)))$
$fp2'(x) = 0 + 3 * cos(x) + 4 * cos(x) - 4 * x * sin(x)$
$fp2'(x) = 7 * cos(x) - 4 * x * sin(x)$
Η prolog λειτουγεί με το πρώτυπο ISO/IEC 13211-1 (PROLOG: Part 1—General Core) και λειτουργει εσωτερικά προθεματικά όπως είπαμε και παράπανω. Το αντίστοιχο δυαδικό δένδρο είναι στην **??**.
Όταν δώσουμε λάθος τιμή στην είσοδο ή κενή τιμή τότε ο prolog server στέλνει μύνημα λάθους. Τις υπόλοιπες μεταβλητές τις θεωρεί σαν την παράγωγο ένος σταθερού αριθμού δηλ. Μηδέν . Το GUI μας δίνει :
Error : error(syntax$_error(.cannotstartan$
expression),syntax$_error(read(15705832_0), 1, .$
cannot start an expression,[-(.,1)],1))
sadsadsad = 0



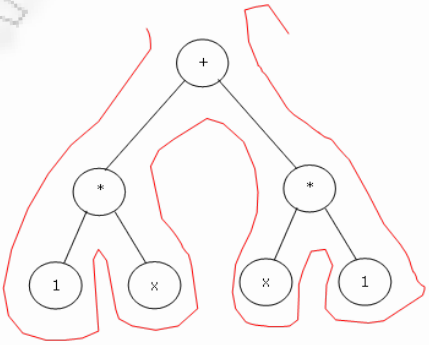
Σχήμα 6.3: Prolog Derivative - Εισαγωγή λάθος δεδομένων



(α)



(β)



(γ)

(δ) Προθεματική Διάσχιση Δυαδικών Δένδρων

Κεφάλαιο 7

Αποτελέσματα

7.1 Πορεία και Τελικό Συμπέρασμα

Στο ξεκίνημα της διπλωματικής εργασίας κλήθηκα να επιλέξω ανάμεσα σε δεκάδες μηχανές τρισδιάστατης απεικόνισης που κυκλοφορούν στο διαδίκτυο. Τα βασικά κριτήρια που έπρεπε να πληρεί η μηχανή ήταν να είναι σε Java και να είναι ανοιχτού κώδικα.
Επιλέχθηκε η Java Monkey Engine, η οποία χρησιμοποιεί OpenGl, βλ. κεφ 2.6 γραφικά, OpenAL(Open Audio Library) και OpenCL(Open Computing Language) για τον έλεγχο περιφερειακών συσκευών όπως ποντικιά, joystics, κα. Επιπλέον χρησιμοποιείται για την ανάπτυξη ενός ολοκληρωμένου παιχνιδιού σε desktop και σε mobile συσκευές. Οι βιβλιοθήκες αυτές είναι κάτω από την Lightweight Java Game Library **(LWJGL)** βλ. κεφ. 2.5.[i]. Ο κύριος λόγος επιλογής της JME είναι ότι στηρίζεται στην Java 3D, είναι ανοιχτού κώδικα(BSD Licence)[ii] και παρέχεται υποστήριξη από την κοινότητα που υπάρχει.

Η επιλογή της Prolog [iii] σαν μία δηλωτική γλώσσα προγραμματισμού στην οποία τόσο οι δομές δεδομένων της όσο και ο μηχανισμός λειτουργίας της βασίζονται στο προτασιακό και κατηγορηματικό λογισμό. Μία από τις βασικές διαφορές της Prolog από άλλες γλώσσες προγραμματισμού όπως η C, Java, Basic, Fortran κτλ. είναι ότι όταν γράφουμε ένα πρόγραμμα σε Prolog, δηλώνουμε όχι τη σειρά των πεπερασμένων βημάτων που πρέπει να ακολουθήσουμε για να εκτελέστει επιτυχώς το πρόγραμμα, αλλά τους στόχους που θέλουμε να πετύχουμε καθώς και τη γνώση που έχουμε σχετικά με την επίτευξη των στόχων αυτών.Η Prolog έχει την δυνατότητα μέσω του μηχανισμού ελέγχου που διαθέτει, να βρίσκει τον τρόπο επίτευξης των στόχων αυτών κάνοντας χρήση της γνώσης που έχουμε εισάγει.
Στην παρούσα εργασία εισάγουμε γεγονότα και κανόνες για το πως μπορούμε να βρούμε την παράγωγο μιας συνάρτησης, δηλαδή να εισάγουμε γνώση σχετική με το χώρο του προβλήματός μας. Η Prolog δίνει απάντησεις στον χαρακτήρα της τρισδιάστατης μηχανής. Ο χαρακτήρας κάνει ερωτήσεις προς την Prolog για τις παραγώγους. Δηλ. πρέπει η Prolog να δώσει απάντηση στο ποια είναι η παράγωγος της κάθε συνάρτησης που ρωτάει ο χαρακτήρας.

Η ενασχόλησή μου με την διπλωματική στα πλαίσια του Μεταπτυχιακού Προγράμματος ήταν πολύ ενδιαφέρουσα και επικοδομητική. Είμαι σίγουρος ότι θα αποτελέσει πολύ σημαντικό εφόδιο στην περαιτέρω πορεία μου.

[i] http://www.lwjgl.org/
[ii] http://en.wikipedia.org/wiki/Template:BSD
[iii] http://www.sics.se/

Κεφάλαιο 8

Πηγαίος Κώδικας

8.1 Autorun

Σχήμα 8.1: Main GUI

```
1  package mppl08059;
2
3  import java.awt.Color;
4  import java.awt.Dimension;
5  import java.awt.Toolkit;
6  import java.awt.event.ActionEvent;
7  import java.awt.event.KeyEvent;
8  import java.util.Enumeration;
9  import java.util.Properties;
10 import javax.swing.AbstractAction;
11 import javax.swing.ActionMap;
12 import javax.swing.InputMap;
13 import javax.swing.JComponent;
14 import javax.swing.JOptionPane;
15 import javax.swing.KeyStroke;
16
17
18 public class Mppl08059_Autorun extends javax.swing.JFrame {
19     /** A return status code - returned if Cancel button has been pressed */
```

```
    public static final int RET_CANCEL = 0;
    /** A return status code - returned if OK button has been pressed */
    public static final int RET_OK = 1;
    /** Creates new form Mppl08059_Autorun */
    String titlos = "ΜΠΠΛ08059 - ΠΜΣΠΛΗΡΟΦΟΡΙΚΗ ";
    public Mppl08059_Autorun() {
        initComponents();
        // Close the dialog when Esc is pressed
        String cancelName = "cancel";
        InputMap inputMap =
            getRootPane().getInputMap(JComponent.WHEN_ANCESTOR_OF_FOCUSED_COMPONENT);
        inputMap.put(KeyStroke.getKeyStroke(KeyEvent.VK_ESCAPE, 0), cancelName);
        ActionMap actionMap = getRootPane().getActionMap();
        actionMap.put(cancelName, new AbstractAction() {

            public void actionPerformed(ActionEvent e) {
                doClose(RET_CANCEL);
            }
        });
    }


private void okButtonActionPerformed(java.awt.event.ActionEvent evt)
    {//GEN-FIRST:event_okButtonActionPerformed
        doClose(RET_OK);
 Mppl08059_Autorun mini = new Mppl08059_Autorun();
mini.setExtendedState(Mppl08059_Autorun.ICONIFIED);
        try
    {
Runtime.getRuntime().exec("rundll32 url.dll,FileProtocolHandler "
        + "mppl08059/Mppl08059.exe");
    }
    catch(Exception e){
      System.out.println("error==="+e.getMessage());
      e.printStackTrace();
    }

}//GEN-LAST:event_okButtonActionPerformed

private void cancelButtonActionPerformed(java.awt.event.ActionEvent evt) {
        doClose(RET_CANCEL);
        System.exit(0);
}//GEN-LAST:event_cancelButtonActionPerformed

private void sisctusImageMouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    try
    {
Runtime.getRuntime().exec("rundll32 url.dll,FileProtocolHandler "
        + "apps/Sicstus/InstallSICStus.exe");
    }
    catch(Exception e){
      System.out.println("error==="+e.getMessage());
      e.printStackTrace();
    }
}//GEN-LAST:event_sisctusImageMouseClicked

private void javaImage1MouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
        try
    {
Runtime.getRuntime().exec("rundll32 url.dll,FileProtocolHandler "
        + "apps/jre-7u7-windows-i586.exe");
    }
    catch(Exception e){
      System.out.println("error==="+e.getMessage());
      e.printStackTrace();
    }
}//GEN-LAST:event_javaImage1MouseClicked

private void sisctusImage1MouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
            try
```

```
91      {
Runtime.getRuntime().exec("rundll32 url.dll,FileProtocolHandler "
            + "apps/AdbeRdr1013_en_US.exe");
94      }
        catch(Exception e){
          System.out.println("error==="+e.getMessage());
97        e.printStackTrace();
        }
} //GEN-LAST:event_sisctusImage1MouseClicked
100
private void prologserverMouseClicked(java.awt.event.MouseEvent evt) {
        // TODO add your handling code here:
103     if (true) {
try
        {
106 Runtime.getRuntime().exec("rundll32 url.dll,FileProtocolHandler "
            + "mppl08059/PathRunPrologServerFirst.bat");
Mppl08059_Autorun mini = new Mppl08059_Autorun();
109 mini.setExtendedState(Mppl08059_Autorun.ICONIFIED);
prologserver.setForeground(Color.black);
prologserver.setText("1. Prolog Server Started...");
112 okButton.setEnabled(true);
jarOpen.setForeground(Color.black);
        }
115     catch(Exception e){
          System.out.println("error==="+e.getMessage());
          e.printStackTrace();
118     }
        }

121 } //GEN-LAST:event_prologserverMouseClicked

private void jarOpenMouseClicked(java.awt.event.MouseEvent evt) {
124     // TODO add your handling code here:
        String c1 = "1. Start prolog Server First",
               c2 = "1. Prolog Server Started...";
127     Object t1=c1.toString(),
               t2=c2.toString();
        if (prologserver.getText().equals(t1)) {
130 JOptionPane.showMessageDialog(null, "Start Prolog Server First",
            titlos, JOptionPane.ERROR_MESSAGE);
        } else if(prologserver.getText().equals(t2)) {
133   try
        {
Mppl08059_Autorun mini = new Mppl08059_Autorun();
136 mini.setExtendedState(Mppl08059_Autorun.ICONIFIED);
Runtime.getRuntime().exec("rundll32 url.dll,FileProtocolHandler "
            + "mppl08059/Mppl08059.jar");
139 jarOpen.setForeground(Color.black);
        }
        catch(Exception e){
142       System.out.println("error==="+e.getMessage());
          e.printStackTrace();
        }
145     }

} //GEN-LAST:event_jarOpenMouseClicked
148
private void jarJmeMouseClicked(java.awt.event.MouseEvent evt) {
// TODO add your handling code here:
151     if (true) {
try
        {
154 Runtime.getRuntime().exec("rundll32 url.dll,FileProtocolHandler "
            + "mppl08059/Mppl08059_1.jar");
Mppl08059_Autorun mini = new Mppl08059_Autorun();
157 mini.setExtendedState(Mppl08059_Autorun.ICONIFIED);
jarJme.setForeground(Color.black);
        }
160     catch(Exception e){
          System.out.println("error==="+e.getMessage());
          e.printStackTrace();
163     }
```

```
    }

} //GEN-LAST:event_jarJmeMouseClicked

private void pdfdocMouseClicked(java.awt.event.MouseEvent evt) {
// TODO add your handling code here:
     try
    {
Runtime.getRuntime().exec("rundll32 url.dll,FileProtocolHandler "
        + "docs/mppl08059.pdf");
Mppl08059_Autorun mini = new Mppl08059_Autorun();
mini.setExtendedState(Mppl08059_Autorun.ICONIFIED);
pdfdoc.setForeground(Color.black);
    }
    catch(Exception e){
      System.out.println("error==="+e.getMessage());
      e.printStackTrace();
    }
} //GEN-LAST:event_pdfdocMouseClicked

private void pdfslideMouseClicked(java.awt.event.MouseEvent evt) {
// TODO add your handling code here:
     try
    {
Runtime.getRuntime().exec("rundll32 url.dll,FileProtocolHandler "
        + "docs/mppl08059_slides.pdf");
Mppl08059_Autorun mini = new Mppl08059_Autorun();
mini.setExtendedState(Mppl08059_Autorun.ICONIFIED);
pdfslide.setForeground(Color.black);
    }
    catch(Exception e){
      System.out.println("error==="+e.getMessage());
      e.printStackTrace();
    }
} //GEN-LAST:event_pdfslideMouseClicked

    public int getReturnStatus() {
        int returnStatus = 0;
        return returnStatus;
    }
    private void doClose(int retStatus) {
        int returnStatus;
        returnStatus = retStatus;
        setVisible(false);
        dispose();
    }


    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
         javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Windows".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(Mppl08059_Autorun.class.getName()).
         log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(Mppl08059_Autorun.class.getName()).
         log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(Mppl08059_Autorun.class.getName()).
         log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(Mppl08059_Autorun.class.getName()).
         log(java.util.logging.Level.SEVERE, null, ex);
        }
```

```
        // </editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {

            public void run() {
                new Mppl08059_Autorun().setVisible(true);
            }
        });
    }
    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton cancelButton;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JPanel jPanel4;
    private javax.swing.JPanel jPanel5;
    private javax.swing.JPanel jPanel6;
    private javax.swing.JPanel jPanel7;
    private javax.swing.JLabel jarJme;
    private javax.swing.JLabel jarOpen;
    private javax.swing.JLabel javaImage1;
    private javax.swing.JScrollPane javaP;
    private javax.swing.JTextArea jtextAreaSettings;
    private javax.swing.JButton okButton;
    private javax.swing.JLabel pdfdoc;
    private javax.swing.JLabel pdfslide;
    private javax.swing.JLabel prologserver;
    private javax.swing.JLabel sisctusImage;
    private javax.swing.JLabel sisctusImage1;
    // End of variables declaration//GEN-END:variables
}
```

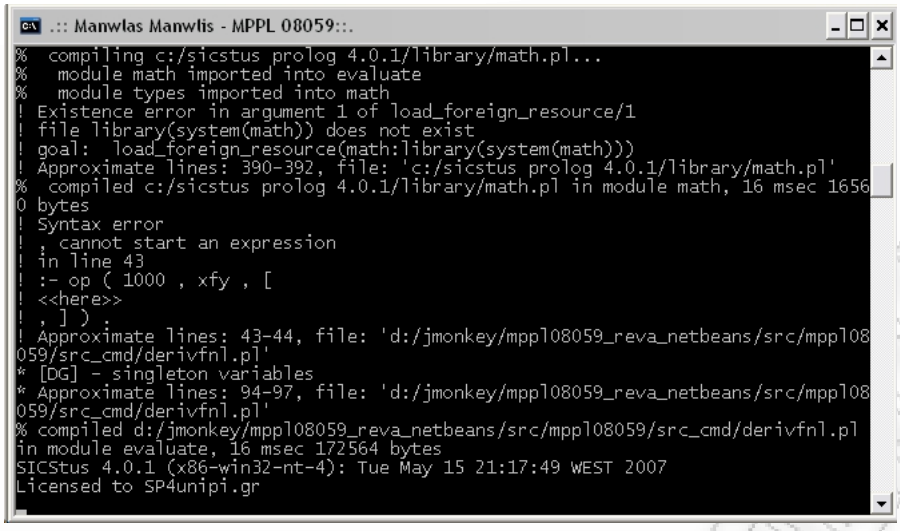
Κώδικας 8.1: Main GUI

8.2 Prolog - Derivative

Το αρχείο Prolog που μας δίνει τις απαντήσεις για τις παραγώγους είναι το:

```
:- module(evaluate,[main/0,my_predicate/2]).
:- use_module(library(prologbeans)).
:- use_module(library(codesio),[read_from_codes/2]).
//Math.pl
:- use_module(library('math')).

/*Register acceptable queries and start the server (using default port)*/

main:-
    register_query(evaluate(C,P), my_predicate(C, P)),
    start.
/*
In this case we know that we have received a list of characters
that needs to be converted into an expression!
*/
my_predicate(Chars, P) :-
    read_from_codes(Chars, X),
    df(X,P).

:- op( 1200, xfx, [ :-, --> ]).
:- op( 1200, fx, [ :-, ?- ]).
:- op( 1150, fx, [ mode, public, dynamic, volatile, discontiguous,
```

```
multifile, block, meta_predicate,
initialization ]).
:- op( 1100, xfy, [ ; ]).
:- op( 1050, xfy, [ -> ]).
:- op( 1000, xfy, [ ',' ]).
:- op( 900, fy, [ \+, spy, nospy ]).
:- op( 700, xfx, [ =, \=, is, =.., ==, \==, @<, @>, @=<, @>=,
=:=, =\=, <, >, =<, >= ]).
:- op( 550, xfy, [ : ]).
:- op( 500, yfx, [ +, -, \, /\, \/ ]).
:- op( 300, yfx, [ *, /, //, mod, rem, <<, >> ]).
:- op( 200, xfx, [ ** ]).
:- op( 200, xfy, [ ^ ]).
:- op( 200, fy, [ +, -, \ ]).

/* df(F,DF) */
//f(x)=n --> f'(x)=0
        df(F,0) :- F\=x, atomic(F), !.
//f(x)=x --> f'(x)=1
        df(x,1) :- !.
//f(x)=1/x --> f'(x)=-1/x^2
        df(1/x,(-((1)/(x*x)))) :- !.
//f(x)=x^n --> f'(x)=n*x^n-1
        df(x^N,N*x^M) :- M is N-1, !.
//f(x)=e^x --> f'(x)=e^x
        df(exp(x),exp(x)) :- !.
//f(x)=e^(-x) --> f'(x)=-e^(-x)
        df(exp(-x),-exp(-x)) :- !.
//f(x)=sqrt(x) --> f'(x)=1/2*sqrt(x)
        df(sqrt(x),(1/(2*sqrt(x)))) :-!.
//f(x)=x^x --> f'(x)=x^x(ln(x)+1)
        df(x^x,((x^x)*((ln(x))+1))) :- !.
//f(x)=ln(x) --> f'(x)=1/x
        df(ln(x),1/x) :- !.
//f(x)=sin(x) --> f'(x)=cos(x)
        df(sin(x),cos(x)) :- !.
//f(x)=cos(x) --> f'(x)=-sin(x)
        df(cos(x),-sin(x)) :- !.
//f(x)=tan(x) --> f'(x)=1/sin^2(x)
        df(tan(x),(1/(sin(x)*sin(x)))) :- !.
//f(x)=arcsin(x) --> f'(x)=1/sqrt(1-x^2)
        df(arcsin(x),(1/(sqrt(1-(x^2))))) :-!.
//f(x)=arccos(x) --> f'(x)=-(1/sqrt(1-x^2))
        df(arccos(x),-(1/(sqrt(1-(x^2))))) :- !.

//f(x)+g(x) --> f'(x)+g'(x)
        df(F+G,DF+DG) :-
        df(F,DF), df(G,DG).
//f(x)-g(x) --> f'(x)-g'(x)
        df(F-G,DF-DG) :-
        df(F,DF), df(G,DG).
//f(x)*g(x) --> f'(x)*g(x)+f(x)*g'(x)
        df(F*G,DF*G+F*DG) :-
        df(F,DF), df(G,DG).
//f(x)/g(x) --> (f'(x)*g(x)-f(x)*g'(x))/(g(x))^2
        df(F/G,(DF*G-G*DF/G*G)) :-
        df(F,DF), df(G,DG).

// This will be called if we build a run-time system
user:runtime_entry(start) :-
    main.
```



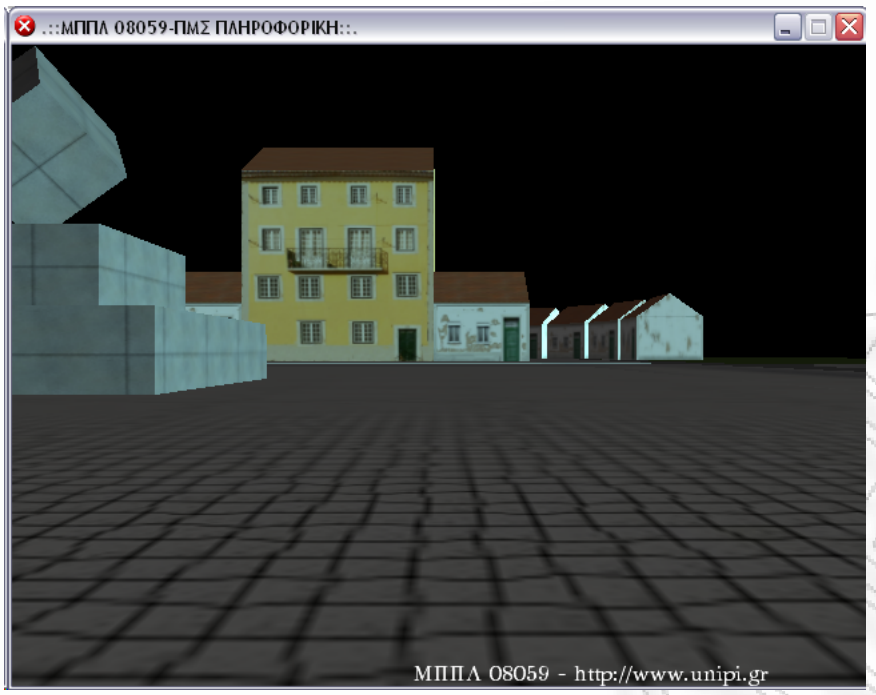
Σχήμα 8.2: Prolog Server

8.3 Batch File

Η εργασία απαιτεί πρώτα να τρέχει ο Prolog Server. Κάνουμε consult το αρχείο derivFnl, που έχουμε στο κεφ. 8.2, στη SICstus Prolog :

```
title .:: Manwlas Manwlis - MPPL 08059::.
color 07
if not "%minimized%"=="" goto :minimized
set minimized=true
start /min cmd /C "%~dpnx0"
goto :EOF
:minimized
sicstus -l derivFnl --goal "main."
```

8.4 Java Monkey Engine Code



Σχήμα 8.3: JME - Town

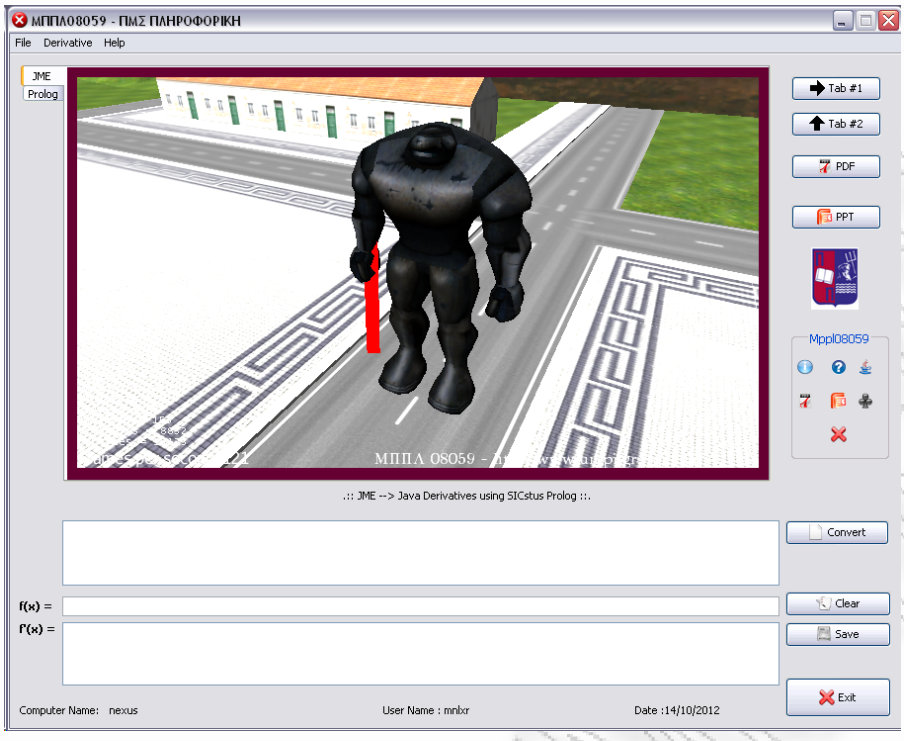
```
package mppl08059.NelLand;

import com.jme3.animation.*;
import com.jme3.app.SimpleApplication;
import com.jme3.font.BitmapText;
import com.jme3.input.KeyInput;
import com.jme3.input.controls.ActionListener;
import com.jme3.input.controls.KeyTrigger;
import com.jme3.light.DirectionalLight;
import com.jme3.math.ColorRGBA;
import com.jme3.math.Quaternion;
import com.jme3.math.Vector3f;
import com.jme3.scene.Geometry;
import com.jme3.scene.Node;
import com.jme3.scene.Spatial;
import com.jme3.scene.shape.Box;

public class TestOgreAnim extends SimpleApplication
        implements AnimEventListener, ActionListener {

    private AnimChannel channel;
    private AnimControl control;
    private Geometry geom;
    String unipiUrl="http://www.unipi.gr",
       StudentCode="ΜΠΠΛ 08059",
       SpceChar=" – ";

    public static void main(String[] args) {
        TestOgreAnim app = new TestOgreAnim();
        app.start();
    }

    @Override
    public void simpleInitApp() {
        flyCam.setMoveSpeed(10f);
        cam.setLocation(new Vector3f(6.4013605f, 7.488437f, 12.843031f));
        cam.setRotation(new Quaternion(-0.060740203f, 0.93925786f, -0.2398315f, -0.2378785f));

        DirectionalLight dl = new DirectionalLight();
        dl.setDirection(new Vector3f(-0.1f, -0.7f, -1).normalizeLocal());
        dl.setColor(new ColorRGBA(1f, 1f, 1f, 1.0f));
```

```
        rootNode.addLight(dl);

        Spatial model = (Spatial) assetManager.loadModel("Models/Oto.mesh.xml");
        model.center();

        control = model.getControl(AnimControl.class);
        control.addListener(this);
        channel = control.createChannel();

        for (String anim : control.getAnimationNames())
            System.out.println(anim);

        channel.setAnim("stand");
        geom = (Geometry)((Node)model).getChild(0);
        SkeletonControl skeletonControl = model.getControl(SkeletonControl.class);

        Box b = new Box(.25f,3f,.25f);
        Geometry item = new Geometry("Item", b);
        item.move(0, 1.5f, 0);
        item.setMaterial(assetManager.loadMaterial("Materials/RedColor.j3m"));
        Node n = skeletonControl.getAttachmentsNode("hand.right");
        n.attachChild(item);

        rootNode.attachChild(model);

        inputManager.addListener(this, "Attack");
        inputManager.addMapping("Attack", new KeyTrigger(KeyInput.KEY_SPACE));


        //Use Custom Font For Text Title
guiNode.detachAllChildren();
guiFont = assetManager.loadFont("/Textures/mppl080591.fnt");
BitmapText TitleText = new BitmapText(guiFont, false);
TitleText.setSize(guiFont.getCharSet().getRenderedSize());
TitleText.setText(StudentCode + SpceChar + unipiUrl);
TitleText.setLocalTranslation(300, TitleText.getLineHeight(), 200);
guiNode.attachChild(TitleText);


    }

    @Override
    public void simpleUpdate(float tpf) {
        super.simpleUpdate(tpf);

                        geom.getMesh().createCollisionData();

    }


    @Override
    public void onAnimCycleDone(AnimControl control, AnimChannel channel, String animName) {
        if (animName.equals("Dodge")){
            channel.setAnim("stand", 0.50f);
            channel.setLoopMode(LoopMode.DontLoop);
            channel.setSpeed(1f);
        }
    }

    @Override
    public void onAnimChange(AnimControl control, AnimChannel channel, String animName) {
    }

    @Override
    public void onAction(String binding, boolean value, float tpf) {
        if (binding.equals("Attack") && value){
            if (!channel.getAnimationName().equals("Dodge")){
                channel.setAnim("Dodge", 0.50f);
                channel.setLoopMode(LoopMode.Cycle);
                channel.setSpeed(0.10f);
            }
        }
```

```
        }
116
}
```

Κώδικας 8.2: Java Monkey Engine-Town

8.5 Main Form



Σχήμα 8.4: Java GUI-tab0

```
package mppl08059;

import com.jme3.app.Application;
import com.jme3.app.SimpleApplication;
import com.jme3.system.AppSettings;
import com.jme3.system.JmeCanvasContext;
import com.jme3.util.JmeFormatter;
import java.awt.*;
import java.awt.geom.Rectangle2D;
import java.io.*;
import java.net.InetAddress;
import java.util.*;
import java.util.List;
import java.util.concurrent.Callable;
import java.util.logging.ConsoleHandler;
import java.util.logging.Handler;
import javax.swing.*;
import mppl08059.NelLand.TestOgreAnim;
import mppl08059.classes.AboutMppl08059;
import mppl08059.classes.Mppl08059Info;
import mppl08059.classes.Mppl08059ShortCuts;
import org.apache.log4j.*;
import se.sics.prologbeans.Bindings;
import se.sics.prologbeans.PBTerm;
import se.sics.prologbeans.PrologSession;
import se.sics.prologbeans.QueryAnswer;
/**
 *
 * @author Manolas Manolis – MPPL08059
 */
public class Mppl08059_Main extends javax.swing.JFrame //implements ActionListener
{
    private static Logger logger = Logger.getLogger(Mppl08059_Main.class.getName());
    private static final Properties properties = new Properties();
    private String filenameTxt, filename;
```

```
36      static SplashScreen mySplash;                   // instantiated by JVM we use it to get
            graphics
        static Graphics2D splashGraphics;               // graphics context for overlay of the
            splash image
        static Rectangle2D.Double splashTextArea;       // area where we draw the text
39      static Rectangle2D.Double splashProgressArea;   // area where we draw the progress bar
        static Font font;                               // used to draw our text
        String emfanisiMetal = "javax.swing.plaf.metal.MetalLookAndFeel";
42      String emfanisiWindowsXP = "com.sun.java.swing.plaf.windows.WindowsLookAndFeel";
        String emfanisiMotif = "com.sun.java.swing.plaf.motif.MotifLookAndFeel";
        String gtkmotif = "com.sun.java.swing.plaf.gtk.GTKLookAndFeel";
45      private static JmeCanvasContext context;
        private static Application app;
        private static Container currentPanel;
48      private static final String appClass = "mppl08059.NelLand.TestOgreAnim";
        //private static JTabbedPane tabbedPane;
        //private static GLCanvas gLCanvas1;
51      FileOutputStream out; // declare a file output object
        PrintStream p; // declare a print stream object
        boolean nai = true;
54      boolean oxi = false;
        String titlos = "ΜΠΠΛ08059 – ΠΜΣΠΛΗΡΟΦΟΡΙΚΗ ";
        String unipiurl = "http://www.cs.unipi.gr";
57  /*********************************************/
    //======== PrologSession ========
    /*
60       * PrologSession handles the connection with the Prolog Server. Currently
         * only synchronous connections with the server are supported.
         */
63      private PrologSession sessionmppl08059 = new PrologSession();

    /**
66   * Creates new form Mppl08059
     */
        public Mppl08059_Main() throws java.io.IOException {
69          initComponents();
            //Open JFrame in Full Screen
    //        Toolkit tk = Toolkit.getDefaultToolkit();
72  //        Dimension d = tk.getScreenSize(); //Gets the user screen size
    //        int scrnHigh = d.height; //Pulls out the High
    //        int scrnWide = d.width; //Pulls out the Wide
75  //        setSize(scrnWide, scrnHigh);//sets screen to FULL size
    //        setLocation(0, 0);
    //        logger.info("CLASS –Screen Resoloution-> W:"+d.width+" H:"+d.height);
78          //End of Open JFrame in Full Screen
            logger.info("CLASS –Init Components");
            if ((Integer.getInteger("se.sics.prologbeans.debug", 0)).intValue() != 0) {
81              logger.info("CLASS –se.sics.prologbeans.debug");
                /**
                 * Sets the timeout in milliseconds before the connection to the
84               * Prolog server is reset (when a query is not answered). Setting
                 * the timeout to 0 will disable timeouts for this prolog session.
                 * Default is 2000 milliseconds.
                 *
87               */
                sessionmppl08059.setTimeout(0);
90              logger.info("CLASS –Prolog–Session Timeout");
            }
            /*
93           * Connects to the Prolog server. By default executeQuery will
             * automatically connect to the server when called.
             */
96          sessionmppl08059.connect();
            logger.info("CLASS –Prolog Server Started–Connect");
    //sessionmppl08059.disconnect();
99  //logger.info("Prolog Server Not Started");

        }//End of Mppl08059
102


            //currentPanel=canvasPanel1;
105         //canvas1.setVisible(true);
            //createCanvas(appClass);
```

```
        //startApp();
108     Calendar cal = new GregorianCalendar();
        int month = cal.get(Calendar.MONTH);
        int year = cal.get(Calendar.YEAR);
111     int day = cal.get(Calendar.DAY_OF_MONTH);
        javax.swing.JLabel dateLabel = new javax.swing.JLabel();
        String imero=day + "/" + (month + 1) + "/" + year;
114     String resultImero=day+""+(month+1)+""+year;
        // Code adding the component to the parent container - not shown here
        jLabel1.setText("Date :"+imero);
117
        java.awt.Dimension screenSize = java.awt.Toolkit.getDefaultToolkit().getScreenSize();
        setBounds((screenSize.width-975)/2, (screenSize.height-874)/2, 975, 874);
120 }// </editor-fold>//GEN-END:initComponents

    private void exitMenuItemActionPerformed(java.awt.event.ActionEvent evt)
        {//GEN-FIRST:event_exitMenuItemActionPerformed
123     app.stop(nai);
        System.exit(0);
        logger.info("CLASS -app stop");
126     logger.info("CLASS -Exit Application");
    }//GEN-LAST:event_exitMenuItemActionPerformed

129 private void contentsMenuItemActionPerformed(java.awt.event.ActionEvent evt)
        {//GEN-FIRST:event_contentsMenuItemActionPerformed
// TODO add your handling code here:
        Runtime run = Runtime.getRuntime();
132     try {
            Process child = Runtime.getRuntime().exec("rundll32 url.dll,"
                    + "FileProtocolHandler help/mppl08059.chm");
135     } catch (Exception ex) {
            System.out.println(ex.getMessage());
        }
138     logger.info("CLASS -Help Mppl08059 Application");
    }//GEN-LAST:event_contentsMenuItemActionPerformed

141 private void aboutMenuItemActionPerformed(java.awt.event.ActionEvent evt)
        {//GEN-FIRST:event_aboutMenuItemActionPerformed
        AboutMppl08059 about = new AboutMppl08059(this, true);
        about.setVisible(nai);
144     requestFocus();
        logger.info("CLASS -About...");
    }//GEN-LAST:event_aboutMenuItemActionPerformed
147
    private void systMItemActionPerformed(java.awt.event.ActionEvent evt)
        {//GEN-FIRST:event_systMItemActionPerformed
        Mppl08059Info si = new Mppl08059Info();
150     si.setVisible(nai);
        requestFocus();
        logger.info("CLASS -System Informations");
153 }//GEN-LAST:event_systMItemActionPerformed

    private void exitFormBtActionPerformed(java.awt.event.ActionEvent evt)
        {//GEN-FIRST:event_exitFormBtActionPerformed
156 //        app.stop(nai);
        System.exit(0);         // TODO add your handling code here:
        logger.info("CLASS -Exit Application");
159 }//GEN-LAST:event_exitFormBtActionPerformed

    private void shortcutMIActionPerformed(java.awt.event.ActionEvent evt)
        {//GEN-FIRST:event_shortcutMIActionPerformed
162     Mppl08059ShortCuts scuts = new Mppl08059ShortCuts(this, nai);
        scuts.setVisible(nai);
        requestFocus();
165     logger.info("CLASS -Windows ShortCuts");
    }//GEN-LAST:event_shortcutMIActionPerformed

168 private void pdf_MIActionPerformed(java.awt.event.ActionEvent evt)
        {//GEN-FIRST:event_pdf_MIActionPerformed
        // TODO add your handling code here:
        try {
171
            File pdfFile = new File("docs/mppl08059.pdf");
```

```
            if (pdfFile.exists()) {

                if (Desktop.isDesktopSupported()) {
                    Desktop.getDesktop().open(pdfFile);
                } else {
                    outPutprolog.setText("Awt Desktop is not supported!");
                }

            } else {
                outPutprolog.setText("File is not exists!");
            }

            outPutprolog.setText("Done");
            logger.info("CLASS -index PDF open");
        } catch (Exception ex222) {
        }
    }//GEN-LAST:event_pdf_MIActionPerformed

    private void ppt_MIActionPerformed(java.awt.event.ActionEvent evt)
        {//GEN-FIRST:event_ppt_MIActionPerformed
        // TODO add your handling code here:
        try {
            File pdfFile = new File("docs/mppl08059_slides.pdf");
            if (pdfFile.exists()) {
                if (Desktop.isDesktopSupported()) {
                    Desktop.getDesktop().open(pdfFile);
                } else {
                    outPutprolog.setText("Awt Desktop is not supported!");
                }
            } else {
                outPutprolog.setText("File is not exists!");
            }
            outPutprolog.setText("Done");
            logger.info("CLASS -ppt PDF open");
        } catch (Exception ex222) {
        }
    }//GEN-LAST:event_ppt_MIActionPerformed

private void tab0_MIActionPerformed(java.awt.event.ActionEvent evt)
    {//GEN-FIRST:event_tab0_MIActionPerformed
    //currentPanel = canvasPanel1;
    tabbedPane.setSelectedIndex(0);

}//GEN-LAST:event_tab0_MIActionPerformed

private void clearButtonActionPerformed(java.awt.event.ActionEvent evt)
    {//GEN-FIRST:event_clearButtonActionPerformed
// TODO add your handling code here:
    inputProlog.setText("");
    textProlog.setText("");
    logger.info("CLASS -Prolog-clear text");

}//GEN-LAST:event_clearButtonActionPerformed

private void clearMIActionPerformed(java.awt.event.ActionEvent evt)
    {//GEN-FIRST:event_clearMIActionPerformed
// TODO add your handling code here:
    inputProlog.setText("");
    textProlog.setText("");
    logger.info("CLASS -Prolog-clear text");
}//GEN-LAST:event_clearMIActionPerformed

private void evaluateActionPerformed(java.awt.event.ActionEvent evt)
    {//GEN-FIRST:event_evaluateActionPerformed
// TODO add your handling code here:
    try {
//Bindings bindings = new Bindings();
//se.sics.prologbeans.QueryAnswer answer = new se.sics.prologbeans.QueryAnswer();
//PBTerm result = new PBTerm();
        /**
         * ****************************************
         * Binding method() : Adds the specified variable binding Pairnei tin
         * eisodo apo to TextField Returns the value for the specified variable
```

```
         * or null if the variable is not bound.
         ****************************************
         */
        Bindings bindings = new Bindings() {
        }.bind("E", inputProlog.getText() + '.');
        logger.info("CLASS -Prolog-bindings: " + bindings);
        /**
         * ************************************************************************
         * executeQuery() : Sends a query to the Prolog server and waits for the
         * answer before returning the QueryAnswer. Bindings are variable
         * bindings for the given query and will ensure that the values are
         * stuffed correctly. Parameters: query - the query to send to the
         * prolog server The characters in the query are restricted to
         * ISO-8859-1. bindings - the variable bindings to use in the query
         * Returns: the answer from the prolog server Throws:
         * java.io.IOException - if an error occurs. A possible cause is a
         * timeout. IllegalCharacterSetException
         ************************************************************************
         */
//QueryAnswer answer = session.executeQuery("paragogos(E,R)", bindings);
        QueryAnswer answer = sessionmppl08059.executeQuery("evaluate(E,R)", bindings);
        logger.info("CLASS -Prolog-query answer: " + answer);
        /**
         * ************************************************************************
         * getValue(): Returns the value for the specified variable or null if
         * the variable is not bound. Parameters: name - the name of the
         * variable Returns: the value of the variable as a Term or null if the
         * variable is not bound
         ************************************************************************
         */
//px. 2+3 auto epistrefei
        PBTerm result = answer.getValue("R");
        logger.info("CLASS -Prolog-pbterm: " + result);
        /**
         * ************************************************************************
         * The Java code above first sets up the GUI with a text area for
         * showing results, a text field for entering expressions, and a button
         * for requesting an evaluation (the constructor Manwlas()). It will
         * also add itself as ActionListener on both the text field and the
         * button. The method actionPerformed(ActionEvent event) will be called
         * whenever the user has pressed Enter or clicked on the button.
         * actionPerformed first binds the variable E to the value of the text
         * field, and then sends the query to the Prolog server with
         * session.executeQuery("paragogos(E,R)", bindings);. If everything goes
         * well, the Prolog server will return an answer (bound to R), which
         * will be appended to the text area.
         ************************************************************************
         */
        // Ean i eisodos einai diafori tou midenos tote vgale apotelesma
        if (result != null) //if(result1 != null)
        {
            textProlog.append(inputProlog.getText() + " = " + result + '\n');
            textProlog1.append(inputProlog.getText() + " = " + result + '\n');
            inputProlog.setText("");
            outPutprolog.setText(" " + result + '\n');
            logger.info("CLASS -Prolog-text null: " + result);
            //TestOgreAnim.main(args);
        } //Alliws tipwse to lathos pou exoume
        else {
            textProlog.append("Error : " + answer.getError() + '\n');
            textProlog1.append("Error : " + answer.getError() + '\n');
            outPutprolog.setText(" " + answer.getError() + '\n');
        }
    } catch (Exception e) {
        textProlog.append("Error when querying Prolog Server: " + e.getMessage() + '\n');
        textProlog1.append("Error when querying Prolog Server: " + e.getMessage() + '\n');
        outPutprolog.setText(" " + e.getMessage() + '\n');
        logger.info("CLASS -Prolog-error: " + e.getMessage());
    }
}//GEN-LAST:event_evaluateActionPerformed

private void evaluateMIActionPerformed(java.awt.event.ActionEvent evt)
    {//GEN-FIRST:event_evaluateMIActionPerformed
```

```
// TODO add your handling code here:
    try {
//Bindings bindings = new Bindings();
//se.sics.prologbeans.QueryAnswer answer = new se.sics.prologbeans.QueryAnswer();
//PBTerm result = new PBTerm();
        /**
         * ****************************************
         * Binding method() : Adds the specified variable binding Pairnei tin
         * eisodo apo to TextField Returns the value for the specified variable
         * or null if the variable is not bound.
         ******************************************
         */
        Bindings bindings = new Bindings() {
        }.bind("E", inputProlog.getText() + '.');
        logger.info("CLASS -Prolog-bindings: " + bindings);
        /**
         * ************************************************************************
         * executeQuery() : Sends a query to the Prolog server and waits for the
         * answer before returning the QueryAnswer. Bindings are variable
         * bindings for the given query and will ensure that the values are
         * stuffed correctly. Parameters: query - the query to send to the
         * prolog server The characters in the query are restricted to
         * ISO-8859-1. bindings - the variable bindings to use in the query
         * Returns: the answer from the prolog server Throws:
         * java.io.IOException - if an error occurs. A possible cause is a
         * timeout. IllegalCharacterSetException
         ************************************************************************
         */
        //QueryAnswer answer = session.executeQuery("paragogos(E,R)", bindings);
        QueryAnswer answer = sessionmppl08059.executeQuery("evaluate(E,R)", bindings);
        logger.info("CLASS -Prolog-query answer: " + answer);
        /**
         * ************************************************************************
         * getValue(): Returns the value for the specified variable or null if
         * the variable is not bound. Parameters: name - the name of the
         * variable Returns: the value of the variable as a Term or null if the
         * variable is not bound
         ************************************************************************
         */
        //px. 2+3 auto epistrefei
        PBTerm result = answer.getValue("R");
        logger.info("CLASS -Prolog-pbterm: " + result);
        /**
         * ************************************************************************
         * The Java code above first sets up the GUI with a text area for
         * showing results, a text field for entering expressions, and a button
         * for requesting an evaluation (the constructor Manwlas()). It will
         * also add itself as ActionListener on both the text field and the
         * button. The method actionPerformed(ActionEvent event) will be called
         * whenever the user has pressed Enter or clicked on the button.
         * actionPerformed first binds the variable E to the value of the text
         * field, and then sends the query to the Prolog server with
         * session.executeQuery("paragogos(E,R)", bindings);. If everything goes
         * well, the Prolog server will return an answer (bound to R), which
         * will be appended to the text area.
         ************************************************************************
         */
        // Ean i eisodos einai diafori tou midenos tote vgale apotelesma
        if (result != null) //if(result1 != null)
        {
            textProlog.append(inputProlog.getText() + " = " + result + '\n');
            textProlog1.append(inputProlog.getText() + " = " + result + '\n');
            inputProlog.setText(" ");
            outPutprolog.setText(" " + result + '\n');
            logger.info("CLASS -Prolog-text null: " + result);
        } //Alliws tipwse to lathos pou exoume
        else {
            textProlog.append("Error : " + answer.getError() + '\n');
            textProlog1.append("Error : " + answer.getError() + '\n');
            outPutprolog.setText(" " + answer.getError() + '\n');
        }
    } catch (Exception e) {
        textProlog.append("Error when querying Prolog Server: " + e.getMessage() + '\n');
```

```
            textProlog1.append("Error when querying Prolog Server: " + e.getMessage() + '\n');
387         outPutprolog.setText(" " + e.getMessage() + '\n');
            logger.info("CLASS -Prolog-error: " + e.getMessage());
        }
390 }//GEN-LAST:event_evaluateMIActionPerformed

    private void saveMIActionPerformed(java.awt.event.ActionEvent evt)
        {//GEN-FIRST:event_saveMIActionPerformed
393 // TODO add your handling code here:
        FileDialog fd = new FileDialog(new Frame(), "Save File...", FileDialog.SAVE);
        fd.setLocation(50, 50);
396     fd.setFile("Mppl08059_derivative.txt");
        fd.show();
        if (fd.getFile() != null) {
399         filename = fd.getDirectory() + fd.getFile();
            fd.setTitle(filename);
            try {
402             FileWriter fw = new FileWriter(filename);
                DataOutputStream d = new DataOutputStream(new FileOutputStream(filename));
                String line = textProlog.getText();
405             BufferedWriter bw = new BufferedWriter(fw);
                BufferedReader br = new BufferedReader(new StringReader(line+
                        "/*****************************/\n"));
408             textProlog.write(fw);

                while ((line = br.readLine()) != null) {
411                 d.close();
                    outPutprolog.setText("File has been Saved\n");
                }
414         } catch (IOException ess) {
                outPutprolog.setText("File has not been Saved\n");
            }
417         textProlog.requestFocus();
            textProlog1.requestFocus();
        }
420 }//GEN-LAST:event_saveMIActionPerformed

    private void saveButtonActionPerformed(java.awt.event.ActionEvent evt)
        {//GEN-FIRST:event_saveButtonActionPerformed
423 // TODO add your handling code here:
        if (textProlog  != null & outPutprolog != null) {
            FileDialog fd = new FileDialog(new Frame(), "Save File...", FileDialog.SAVE);
426         fd.setLocation(200, 200);
            fd.setFile("Mppl08059_derivative.txt");
            fd.show();
429         if (fd.getFile() != null) {
                filename = fd.getDirectory() + fd.getFile();
                fd.setTitle(filename);
432             try {
                    FileWriter fw = new FileWriter(filename);
                    DataOutputStream d = new DataOutputStream(new FileOutputStream(filename));
435                 String line = textProlog.getText();
                    BufferedWriter bw = new BufferedWriter(fw);
                    BufferedReader br = new BufferedReader(new StringReader(line));
438                 textProlog.write(fw);
                    logger.info("Prolog-File Save");
                    while ((line = br.readLine()) != null) {
441                     d.close();
                        outPutprolog.setText("File has been Saved\n");
                    }
444                 } catch (IOException ess) {
                    outPutprolog.setText("File has not been Saved\n");
                }
447             textProlog.requestFocus();
                textProlog1.requestFocus();
            }
450     } else {
            outPutprolog.setText("Nothing to Save\n");
        }
453 }//GEN-LAST:event_saveButtonActionPerformed

    private void tab1ActionPerformed(java.awt.event.ActionEvent evt)
        {//GEN-FIRST:event_tab1ActionPerformed
```

```
456         // TODO add your handling code here:
            tabbedPane.setSelectedIndex(1);
        }//GEN-LAST:event_tab1ActionPerformed
459
        private void ppt_BtActionPerformed(java.awt.event.ActionEvent evt)
            {//GEN-FIRST:event_ppt_BtActionPerformed
            // TODO add your handling code here:
462         try {

                File pdfFile = new File("docs/mppl08059_slides.pdf");
465             if (pdfFile.exists()) {
                    if (Desktop.isDesktopSupported()) {
                        Desktop.getDesktop().open(pdfFile);
468                 } else {
                        outPutprolog.setText("Awt Desktop is not supported!");
                    }
471             } else {
                    outPutprolog.setText("File is not exists!");
                }
474             outPutprolog.setText("Done");
                logger.info("CLASS -PDF open");
            } catch (Exception ex222) {
477         }
        }//GEN-LAST:event_ppt_BtActionPerformed

480     private void diplwmaPDFActionPerformed(java.awt.event.ActionEvent evt)
            {//GEN-FIRST:event_diplwmaPDFActionPerformed
            // TODO add your handling code here:
            try {
483             File pdfFile = new File("docs/mppl08059.pdf");
                if (pdfFile.exists()) {
                    if (Desktop.isDesktopSupported()) {
486                     Desktop.getDesktop().open(pdfFile);
                    } else {
                        outPutprolog.setText("Awt Desktop is not supported!");
489                 }
                } else {
                    outPutprolog.setText("File is not exists!");
492             }
                System.out.println("Done");
                logger.info("CLASS -PDF open");
495
            } catch (Exception ex222) {
            }
498     }//GEN-LAST:event_diplwmaPDFActionPerformed

        private void windowClosed(java.awt.event.WindowEvent evt) {//GEN-FIRST:event_windowClosed
501         // TODO add your handling code here:
            app.stop();
            System.exit(0);
504         logger.info("CLASS -Exit App: "+evt);
        }//GEN-LAST:event_windowClosed

507     private void tab1_Mlact(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_tab1_Mlact
            // TODO add your handling code here:
            tabbedPane.setSelectedIndex(1);
510 //          currentPanel.remove(canvas);
//          app.stop(true);
//
513 //          createCanvas(appClass);
//          currentPanel.add(canvas);
//          pack();
516 //          startApp();
        }//GEN-LAST:event_tab1_Mlact

519 private void tab0ActionPerformed(java.awt.event.ActionEvent evt)
        {//GEN-FIRST:event_tab0ActionPerformed
//  TODO add your handling code here:
    tabbedPane.setSelectedIndex(0);
522 }//GEN-LAST:event_tab0ActionPerformed

    void batch_queries() {
525     logger.info("CLASS -batch queries");
```

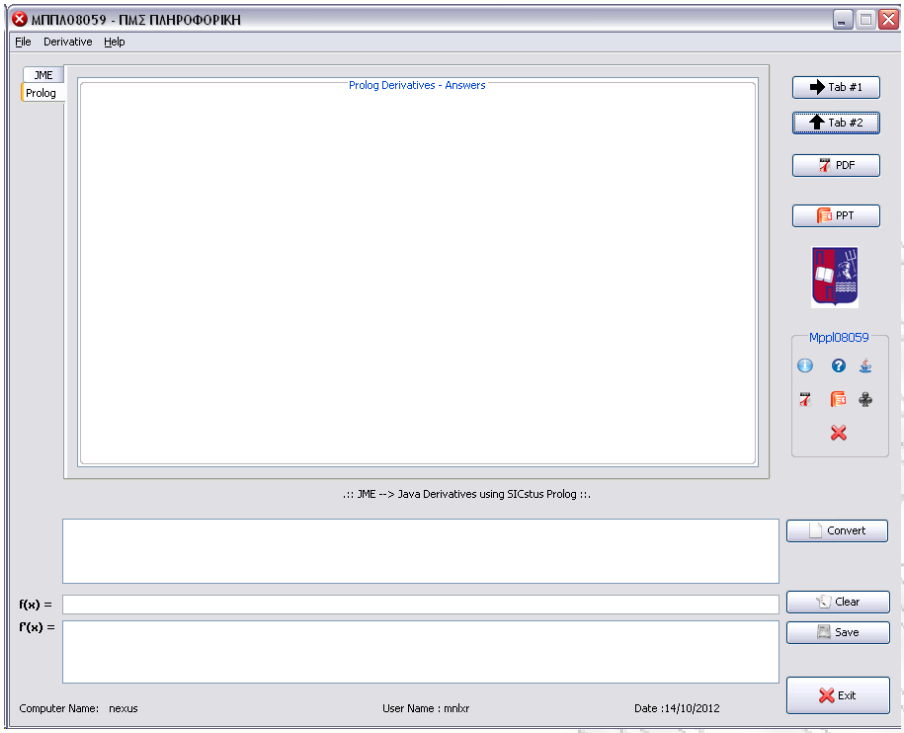
```
        final Runnable calculate = new Runnable() {

            @Override
            public void run() {
                evaluate.doClick();
                logger.info("CLASS –evaluate: " + evaluate);
            }
        };
        new Thread() {

            @Override
            public void run() {
                try {
// [PM] 4.0.4 The Java on Louis does not allow passing the strings as varargs
                    List ql = Arrays.asList(new String[]{" ", "4+6", "11–5", "3*7", "9/3"});
                    logger.info("CLASS –Prolog–List: " + ql);
                    /* Iterators allow the caller to remove elements from the underlying
                     * collection during the iteration with well–defined semantics.
                     */
                    for (Iterator i = ql.iterator(); i.hasNext();) {
                        String q = (String) i.next();
                        inputProlog.setText(q);
                        SwingUtilities.invokeAndWait(calculate);
                        logger.info(q);
                    }

                    System.exit(0);
                } catch (Exception ex) {
                    textProlog.append("Error when clicking Convert button: " +
                        ex.getMessage() + '\n');
                    outPutprolog.setText(" " + ex.getMessage() + '\n');
                }
            }
        }.start();
        logger.info("CLASS –End of batch queries");
    }//End of batch queries

    public static void createCanvas(String appClass) {
        AppSettings settings = new AppSettings(true);
        settings.setWidth(640);
        settings.setHeight(480);
        //settings.containsValue(app);
        settings.setSettingsDialogImage("/Interface/infoStart.png");
        settings.setFullscreen(false);
        settings.setRenderer(AppSettings.LWJGL_OPENGL3);
        settings.setStereo3D(false);
        settings.setFrequency(60);
        settings.setFullscreen(false);
        //settings.setAudioRenderer(appClass);
        settings.setTitle(" ");

        try {
            Class<? extends Application> clazz =
                    (Class<? extends Application>)
                    Class.forName(appClass);
            logger.info("CLASS –"+clazz);
            app = clazz.newInstance();
            logger.info("CLASS –New Instance");
        } catch (ClassNotFoundException ex) {
            ex.printStackTrace();
            logger.info(ex);
        } catch (InstantiationException ex) {
            ex.printStackTrace();
            logger.info(ex);
        } catch (IllegalAccessException ex) {
            logger.info(ex);
        }
        app.setPauseOnLostFocus(false);
        app.setSettings(settings);
        app.createCanvas();
        app.startCanvas();

        context = (JmeCanvasContext) app.getContext();
```

```
        canvas = context.getCanvas();
        canvas.setSize(settings.getWidth(), settings.getHeight());
600     logger.info("CLASS -" + settings);
        logger.info("CLASS -CreateCanvas");
    }//End of createCanvas
603
    public static void startApp() {
        app.startCanvas();
606     app.enqueue(new Callable<Void>() {

            public Void call() {
609             if (app instanceof SimpleApplication) {
                    SimpleApplication simpleApp = (SimpleApplication) app;
                    simpleApp.getFlyByCamera().setDragToRotate(true);
612             }
                return null;
            }
615     });
        logger.info("CLASS -StartApp");
    }
618
    public static void ison() {
        //canvasPanel1.setLayout(new FlowLayout());
621     currentPanel = canvasPanel1;
        logger.info("CLASS -ison");
    }
624
    private static void appInit() {
        for (int i = 1; i <= 10; i++) {
627         int pctDone = i * 100;
            splashText("Loading : #" + i);
            splashProgress(pctDone);
630         try {
                Thread.sleep(250);
            } catch (InterruptedException ex) {
633             // ignore it
            }
        }
636 }

    public static void splashProgress(int pct) {
639     if (mySplash != null && mySplash.isVisible()) {

// Note: 3 colors are used here to demonstrate steps
642 // erase the old one
            splashGraphics.setPaint(Color.white);
            splashGraphics.fill(splashProgressArea);
645
// draw an outline
            splashGraphics.setPaint(Color.white);
648         splashGraphics.draw(splashProgressArea);

// Calculate the width corresponding to the correct percentage
651         int x = (int) splashProgressArea.getMinX();
            int y = (int) splashProgressArea.getMinY();
            int wid = (int) splashProgressArea.getWidth();
654         int hgt = (int) splashProgressArea.getHeight();

            int doneWidth = Math.round(pct * wid / 1000.f);
657         doneWidth = Math.max(0, Math.min(doneWidth, wid - 1));  // limit 0-width

// fill the done part one pixel smaller than the outline
660         splashGraphics.setPaint(Color.RED);
            splashGraphics.fillRect(x, y + 1, doneWidth, hgt - 1);

663 // make sure it's displayed
            mySplash.update();
        }
666 }

    public static void splashText(String str) {
669     if (mySplash != null && mySplash.isVisible()) {   // important to check here so no
            other methods need to know if there
```

```
// really is a Splash being displayed

// erase the last status text
            splashGraphics.setPaint(Color.WHITE);
            splashGraphics.fill(splashTextArea);

// draw the text
            splashGraphics.setPaint(Color.BLACK);
            splashGraphics.drawString(str, (int) (splashTextArea.getX() + 10),
                    (int) (splashTextArea.getY() + 15));
// make sure it's displayed
            mySplash.update();
        }
    }

    private static void splashInit() {
        mySplash = SplashScreen.getSplashScreen();
        if (mySplash != null) {    // if there are any problems displaying the splash this
            will be null
            Dimension ssDim = mySplash.getSize();
            int height = ssDim.height;
            int width = ssDim.width;
// stake out some area for our status information
            splashTextArea = new Rectangle2D.Double(width * 0.35, height * 0.7, width * .4,
                32.);
            splashProgressArea = new Rectangle2D.Double(width * .35, height * .75, width *
                .4, 12);

// create the Graphics environment for drawing status info
            splashGraphics = mySplash.createGraphics();
            font = new Font("Dialog", Font.PLAIN, 14);
            splashGraphics.setFont(font);

// initialize the status info
            splashText("Starting");
            splashProgress(0);
        }
    }

    public static void LogMppl08059() {
        long time = System.currentTimeMillis();
        long logTime = System.currentTimeMillis() - time;
        String pattern = "Milliseconds since program start: r pattern += "Classname of caller: pattern +=
            "Date in ISO8601 format: pattern += "Location of log event: pattern += "Message: PatternLayout layout = new
            PatternLayout(pattern;
        ConsoleAppender appender = new ConsoleAppender(layout);
        logger.info("\n ********************************************** \n");
        logger.info("\n Mppl08059-Unipi \n .::2011-2012::. \n");
        logger.info("\n ********************************************** \n");
        logger.info("main method called..");
        logger.info(pattern);

    }

    @SuppressWarnings("static-access")
    public static void preJME() {
        JmeFormatter formatter = new JmeFormatter();
        logger.info("CLASS -Formater-preJME");
        Handler consoleHandler = new ConsoleHandler();
        logger.info("CLASS -Handler-preJME");
        consoleHandler.setFormatter(formatter);
        logger.info("CLASS -Console Handler-preJME");
        //logger.getLogger("").removeHandler(logger.getLogger("").getHandlers()[0]);
        //logger.getLogger("").addHandler(consoleHandler);
    }


public static void main(String args[]) throws IOException //throws java.io.IOException,
    InterruptedException
{
/*********************************************/
    LogMppl08059();
    logger.info("MAIN -LogMppl08059");
/*********************************************/
```

```
738     splashInit();
        logger.info("MAIN –Splash Init");
/*********************************************/
741     appInit();
        logger.info("MAIN"+" –appInit");
/*********************************************/
744     if (mySplash != null) {
            mySplash.close();
            logger.info("MAIN"+" –Splash Finished");}
747 /*********************************************/
        preJME();
        logger.info("MAIN"+" –pre JME");
750 /*********************************************/
        try {
            Thread.sleep(250);
753         logger.info("MAIN"+" –Thread Sleep 250");
        } catch (InterruptedException ex) {}
/*********************************************/
756     try {
         createCanvas(appClass);
         logger.info("MAIN – appCls: "+args, null);
759     } catch (Exception e) {}
/*********************************************/
        try {
            Thread.sleep(100);
762         logger.info("MAIN"+" –Thread Sleep");
        } catch (InterruptedException ex) {
765     }
/*********************************************/

768 SwingUtilities.invokeLater(new Runnable()
//java.awt.EventQueue.invokeLater(new Runnable()
        {
771             public void run() {
                logger.info("RUN –" + this);
                JPopupMenu.setDefaultLightWeightPopupEnabled(false);
774             ison();
                logger.info("RUN –ison: " + this);
                //currentPanel.add(canvas);
777             canvas.setVisible(true);
                //pack();
                startApp();
780             logger.info("RUN –startApp: " + this);
                logger.info("RUN –Canvas Visible: "+canvas.isVisible());
                logger.info("RUN –Canvas Enable: "+canvas.isEnabled());
783             BasicConfigurator.configure();
                logger.info("RUN –BasicConfigurator");
                PropertyConfigurator.configure("src/log4j.properties");
786             try {
                    UIManager.setLookAndFeel("com.sun.java.swing.plaf."
                            + "windows.WindowsLookAndFeel");
789             } catch (UnsupportedLookAndFeelException e) {
                } catch (ClassNotFoundException e) {
                } catch (InstantiationException e) {
792             } catch (IllegalAccessException e) {
                }
                try {
795                 new Mppl08059_Main().setVisible(true);
                } catch (IOException ex) {
                }
798         }//End of Run
        }//End of Runnable
                );
801 }//End of Main

 }//EOF
```

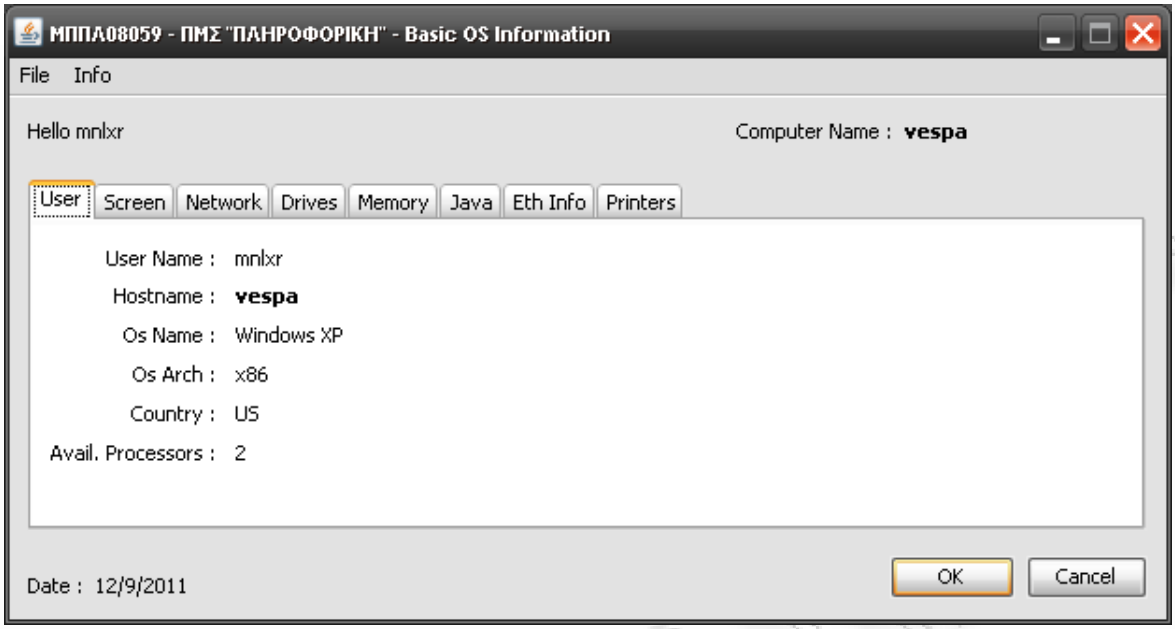
Κώδικας 8.3: Main Form



Σχήμα 8.5: Java GUI-tab1

8.5.1 System Info

```
1
4   private Object stdoutStream;
    /** A return status code - returned if Cancel button has been pressed */
    public static final int RET_CANCEL = 0;
    /** A return status code - returned if OK button has been pressed */
7   public static final int RET_OK = 1;
    private int returnStatus;
    /** Creates new form Mppl08059Info */
10  public Mppl08059Info() {
        initComponents();
        String cancelName = "cancel";
13      javax.swing.InputMap inputMap = getRootPane().getInputMap
      (JComponent.WHEN_ANCESTOR_OF_FOCUSED_COMPONENT);
        inputMap.put(KeyStroke.getKeyStroke(KeyEvent.VK_ESCAPE, 0), cancelName);
16      javax.swing.ActionMap actionMap = getRootPane().getActionMap();
        actionMap.put(cancelName, new AbstractAction() {

19          public void actionPerformed(ActionEvent e) {
                doClose(RET_CANCEL);
            } });}
22      setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
        setTitle("ΜΠΠΛ08059 - ΠΜΣΠΛΗΡΟΦΟΡΙΚΗ \"\" - Basic OS Information");
        jLabel23.setText("User Name :");
25      jLabel39.setText(System.getProperty("user.name"));
        jLabel35.setText("Hostname :");
        try{
28          InetAddress addr = InetAddress.getLocalHost();
            // Get IP Address
            byte[] ipAddr = addr.getAddress();
31          // Get hostname
            String hostname = addr.getHostName();
            jLabel40.setText(hostname);
34      }
        catch(Exception e) {
            e.printStackTrace();}
```



Σχήμα 8.6: System Info

```
37
        jLabel36.setText("Os Name :");
        jLabel41.setText(System.getProperty("os.name"));
40
        jLabel37.setText("Os Arch :");
        jLabel42.setText(System.getProperty("os.arch"));
43
        jLabel38.setText("Country :");
        jLabel43.setText(System.getProperty("user.country"));
46
        jLabel4.setText("Avail. Processors :");
        jLabel5.setText(""+Runtime.getRuntime().availableProcessors()); );
49
        tabberMenu.addTab("User", userP);
        jLabel12.setText("Screen Resolution :");
52      Toolkit toolkit =  Toolkit.getDefaultToolkit ();
        Dimension dim = toolkit.getScreenSize();
        jLabel31.setText(dim.width+" x "+dim.height+" pixels");
55      GraphicsEnvironment ge = GraphicsEnvironment.getLocalGraphicsEnvironment();
        GraphicsDevice[] gs = ge.getScreenDevices();
        for (int i = 0; i < gs.length; i++) {
58          DisplayMode dm = gs[i].getDisplayMode();
            int refreshRate = dm.getRefreshRate();
            if (refreshRate == DisplayMode.REFRESH_RATE_UNKNOWN) { }
61          int bitDepth = dm.getBitDepth();
            int numColors = (int) Math.pow(2, bitDepth);
            jLabel32.setText(bitDepth+" bit");}
64
        jLabel30.setText("Number of Screens :");
        jLabel29.setText("Number of Colors :");
67      GraphicsEnvironment ge1 = GraphicsEnvironment.getLocalGraphicsEnvironment();
        GraphicsDevice[] gs1 = ge1.getScreenDevices();

70      for (int i1 = 0; i1 < gs1.length; i1++) {
            DisplayMode dm1 = gs1[i1].getDisplayMode();
            int refreshRate = dm1.getRefreshRate();
73          if (refreshRate == DisplayMode.REFRESH_RATE_UNKNOWN) {}
            int bitDepth1 = dm1.getBitDepth();
            int numColors1 = (int) Math.pow(2, bitDepth1);
76          jLabel33.setText(numColors1+" Colors");}
        GraphicsEnvironment ge2 = GraphicsEnvironment.getLocalGraphicsEnvironment();

79    try {GraphicsDevice[] gs2 = ge2.getScreenDevices();
            int numScreens = gs2.length;
```

```
            jLabel34.setText(numScreens+" ");
82        }

        catch (HeadlessException e2) { }
85        jLabel28.setText("Bit Depth :"););
          tabberMenu.addTab("Screen", screenP);
          jPanel1.setBackground(new java.awt.Color(255, 255, 255));
88
          try{
              InetAddress thisIp = InetAddress.getLocalHost();
91            jLabel3.setHorizontalAlignment(javax.swing.SwingConstants.LEFT);
              jLabel3.setText(thisIp.getHostAddress()); }

94        catch(Exception e) {e.printStackTrace(); }
          jLabel6.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
          jLabel6.setText("IP address :");
97
          try{
              InetAddress addr = InetAddress.getLocalHost();
100           byte[] ipAddr = addr.getAddress();
              String hostname = addr.getHostName();
              jLabel7.setHorizontalAlignment(javax.swing.SwingConstants.LEFT);
103           jLabel7.setText(hostname);}

          catch(Exception e) {e.printStackTrace();}
106       jLabel8.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
          jLabel8.setText("Hostname :");
          jLabel222.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
109       jLabel222.setText("MAC Address :");

          try{
112           String str1 = "";
              InetAddress address = InetAddress.getLocalHost();
              DecimalFormat twoPlaces = new DecimalFormat("0.00");
115           NetworkInterface ni = NetworkInterface.getByInetAddress(address);
              if (ni != null) {
                  byte[] mac = ni.getHardwareAddress();
118               if (mac != null) {
                      for (int i = 0; i < mac.length; i++)
                      {
121                   str1 += String.format("02XjLabel9.setText(""+str1;}}
             else {jLabel3.setText("Network Interface for the specified address is not
                 found.");}}}

124

        catch (UnknownHostException e) {
              e.printStackTrace();
127       }

        catch (SocketException e) {
130           e.printStackTrace();}
          jLabel10.setText("Eth name :");
          int netPrefix=0;
133
        try{
              InetAddress localHost = Inet4Address.getLocalHost();
136           NetworkInterface networkInterface = NetworkInterface.getByInetAddress(localHost);
              networkInterface.getInterfaceAddresses().get(0).getNetworkPrefixLength();
              int shiftby = (1<<31);
139           for (int i=netPrefix-1; i>0; i--) {shiftby = (shiftby >> 1);}
              String maskString = Integer.toString((shiftby >> 24) & 255) + "."
           + Integer.toString((shiftby >> 16) & 255) + "." + Integer.toString((shiftby >> 8) &
                255)
           + "." + Integer.toString(shiftby & 255);
142           networkInterface.getDisplayName();
              jLabel11.setText(""+networkInterface.getDisplayName()););}
145
          catch(IOException ex){}

148       tabberMenu.addTab("Network", jPanel1);
          drives.setViewportView(jTextArea1);
          FileSystemView fsv11 = FileSystemView.getFileSystemView();
151       File[] roots11 = fsv11.getRoots();
```

```
        for (int i = 0; i < roots11.length; i++)

154     {jTextArea1.setText("Root : " + roots11[i]+" \n\n"); }
        jTextArea1.append("Home directory : " + fsv11.getHomeDirectory()+" \n");
        File[] f11 = File.listRoots();
157
        for (int i = 0; i < f11.length; i++)
        {
160         jTextArea1.append("\nDrive : " + f11[i]+"\n");
            jTextArea1.append("Display name : " + fsv11.getSystemDisplayName(f11[i])+"\n");
            jTextArea1.append("Is drive : " + fsv11.isDrive(f11[i])+"\n");
163         jTextArea1.append("Is floppy : " + fsv11.isFloppyDrive(f11[i])+"\n");
            jTextArea1.append("Readable : " + f11[i].canRead()+"\n");
            jTextArea1.append("Writable : " + f11[i].canWrite()+"\n");
166         jTextArea1.append("Total space : " + (f11[i].getTotalSpace()/1024/1024)+" MB \n");
            jTextArea1.append("Usable space : " + (f11[i].getUsableSpace()/1024/1024)+" MB
                \n");
        }
169
        tabberMenu.addTab("Drives", drives);
        Runtime runtime = Runtime.getRuntime();
172     long maxMemory = runtime.maxMemory();
        long allocatedMemory = runtime.totalMemory();
        long freeMemory = runtime.freeMemory();
175     jTextArea4.setText("Memory Alocated by Java VM :\n\n");
        jTextArea4.append("Free Memory : " + freeMemory / 1024/1024+" MB \n");
        jTextArea4.append("Allocated Memory: " + allocatedMemory / 1024/1024+" MB \n");
178     jTextArea4.append("Max Memory: " + maxMemory /1024/1024+" MB \n");
        jTextArea4.append("Total Free Memory: "+
        (freeMemory + (maxMemory – allocatedMemory)) / 1024/1024+" MB \n");
181     tabberMenu.addTab("Memory", memo);
       Properties p = System.getProperties();
        Enumeration keys = p.keys();
184
    while (keys.hasMoreElements()) {
        String key = (String)keys.nextElement();
187     String value = (String)p.get(key);
        jTextArea3.append("\n"+key+": "+value+"\n"); }
      tabberMenu.addTab("Java", javaP);
190   jScrollPane1.setViewportView(jNetworkInfo);

      try{
193       String str12 = "";
          InetAddress address22 = InetAddress.getLocalHost();
          DecimalFormat twoPlaces22 = new DecimalFormat("0.00");
196

          setJMenuBar(jMenuBar1););
199       getRootPane().setDefaultButton(okButton);
          pack();
          java.awt.Dimension screenSize =
              java.awt.Toolkit.getDefaultToolkit().getScreenSize();
202       java.awt.Dimension dialogSize = getSize();
          setLocation((screenSize.width–dialogSize.width)/2,
       (screenSize.height–dialogSize.height)/2);}
205   public void actionPerformed(java.awt.event.ActionEvent evt) {

 private void okButtonActionPerformed(java.awt.event.ActionEvent evt) {
208     doClose(RET_OK);}

private void cancelButtonActionPerformed(java.awt.event.ActionEvent evt) {
211     doClose(RET_CANCEL);}

public int getReturnStatus() {
214     return returnStatus;}

private void doClose(int retStatus) {
217     returnStatus = retStatus;
        setVisible(false);
        dispose();}
220
public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
```

```
223
        public void run() {
                try {
226       UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndFeel");
      }

229   catch (Exception e) {
        e.printStackTrace();}
              new Mppl08059Info().setVisible(true);}});}}
```

Κώδικας 8.4: System Information

8.5.2 Windows Shortcuts

```
package mppl08059.classes;

3 import java.awt.event.*;
import javax.swing.*;

6 public class Mppl08059ShortCuts extends javax.swing.JDialog {
    /** A return status code - returned if Cancel button has been pressed */
    public static final int RET_CANCEL = 0;
9   /** A return status code - returned if OK button has been pressed */
    public static final int RET_OK = 1;
    /** Creates new form AboutMppl08059 */
12  public Mppl08059ShortCuts(java.awt.Frame parent, boolean modal) {
        super(parent, modal);
        initComponents();
15      // Close the dialog when Esc is pressed
        String cancelName = "cancel";
        InputMap inputMap =
            getRootPane().getInputMap(JComponent.WHEN_ANCESTOR_OF_FOCUSED_COMPONENT);
18      inputMap.put(KeyStroke.getKeyStroke(KeyEvent.VK_ESCAPE, 0), cancelName);
        ActionMap actionMap = getRootPane().getActionMap();
        actionMap.put(cancelName, new AbstractAction() {
21          public void actionPerformed(ActionEvent e) {
                doClose(RET_CANCEL);}});}
    public int getReturnStatus() {
24      return returnStatus;
    }
    private void initComponents() {
27
        javax.swing.JButton okButton = new javax.swing.JButton();
        javax.swing.JButton cancelButton = new javax.swing.JButton();
30      javax.swing.JScrollPane jScrollPane1 = new javax.swing.JScrollPane();
        javax.swing.JTextPane jTextPane1 = new javax.swing.JTextPane();
        javax.swing.JLabel jLabel1 = new javax.swing.JLabel();
33      setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
        setTitle("ΜΠΠΛ08059 - ΠΜΣΠΛΗΡΟΦΟΡΙΚΗ \"\" - Windows Shortcuts");
        setResizable(false);
36      addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent evt) {
                closeDialog(evt);
39          }});
        okButton.setText("OK");
        okButton.addActionListener(new java.awt.event.ActionListener() {
42          public void actionPerformed(java.awt.event.ActionEvent evt) {
                okButtonActionPerformed(evt);
            }
45      });
        cancelButton.setText("Cancel");
        cancelButton.addActionListener(new java.awt.event.ActionListener() {
48          public void actionPerformed(java.awt.event.ActionEvent evt) {
                cancelButtonActionPerformed(evt);
            }});
51      jTextPane1.setContentType("text/html");
        jTextPane1.setEditable(false);
        jTextPane1.setFont(new java.awt.Font("Tahoma", 0, 9));
54      jTextPane1.setText("<html></html>");
        jScrollPane1.setViewportView(jTextPane1);
        jLabel1.setFont(new java.awt.Font("Tahoma", 1, 14));
```

```
57          jLabel1.setText("Windows Shortcuts");
            javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
            getContentPane().setLayout(layout); );
60          layout.linkSize(javax.swing.SwingConstants.HORIZONTAL, new java.awt.Component[]
                {cancelButton, okButton});
            layout.setVerticalGroup();
            getRootPane().setDefaultButton(okButton);
63          pack();
            java.awt.Dimension screenSize = java.awt.Toolkit.getDefaultToolkit().getScreenSize();
            java.awt.Dimension dialogSize = getSize();
66          setLocation((screenSize.width-dialogSize.width)/2,(screenSize.height-dialogSize.height)/2);
        }
        private void okButtonActionPerformed(java.awt.event.ActionEvent evt) {
69          doClose(RET_CANCEL);
        }
          private void closeDialog(java.awt.event.WindowEvent evt) {
72          doClose(RET_CANCEL);
        }
            private void doClose(int retStatus) {
75          returnStatus = retStatus;
            setVisible(false);
            dispose();
78      }
        public static void main(String args[]) {
            java.awt.EventQueue.invokeLater(new Runnable() {
81              public void run() {
                    Mppl08059ShortCuts dialog = new Mppl08059ShortCuts(new javax.swing.JFrame(),
                        true);
                    dialog.addWindowListener(new java.awt.event.WindowAdapter() {
84                      public void windowClosing(java.awt.event.WindowEvent e) {
                            System.exit(0);
                        }});
87                  dialog.setVisible(true);
                } }); }
     private int returnStatus = RET_CANCEL;}
```

Κώδικας 8.5: Windows ShortCuts

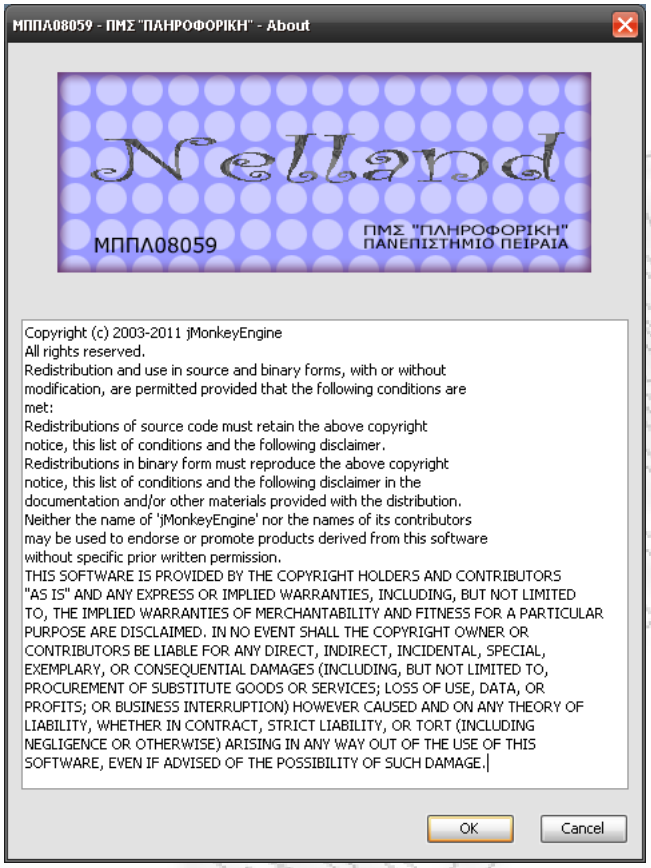
Σχήμα 8.7: Windows Shortcuts

8.5.3 About

```
import java.awt.event.ActionEvent;
import java.awt.event.KeyEvent;
import javax.swing.*;

public class AboutMppl08059 extends javax.swing.JDialog {

    /** A return status code – returned if Cancel button has been pressed */
    public static final int RET_CANCEL = 0;
    /** A return status code – returned if OK button has been pressed */
    public static final int RET_OK = 1;

    /** Creates new form AboutMppl08059 */
    public AboutMppl08059(java.awt.Frame parent, boolean modal) {
        super(parent, modal);
        initComponents();

        // Close the dialog when Esc is pressed
        String cancelName = "cancel";
        InputMap inputMap =
            getRootPane().getInputMap(JComponent.WHEN_ANCESTOR_OF_FOCUSED_COMPONENT);
        inputMap.put(KeyStroke.getKeyStroke(KeyEvent.VK_ESCAPE, 0), cancelName);
        ActionMap actionMap = getRootPane().getActionMap();
        actionMap.put(cancelName, new AbstractAction() {

            public void actionPerformed(ActionEvent e) {
                doClose(RET_CANCEL);
            }
        });
    }
    /** @return the return status of this dialog – one of RET_OK or RET_CANCEL */
    public int getReturnStatus() {
        return returnStatus;
    }

    private void initComponents() {

        javax.swing.JButton okButton = new javax.swing.JButton();
        javax.swing.JButton cancelButton = new javax.swing.JButton();
        javax.swing.JScrollPane jScrollPane1 = new javax.swing.JScrollPane();
        javax.swing.JTextArea jTextArea1 = new javax.swing.JTextArea();
        javax.swing.JPanel jPanel1 = new javax.swing.JPanel();
        javax.swing.JLabel jLabel1 = new javax.swing.JLabel();

        setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
        setTitle("ΜΠΠΛ08059 – ΠΜΣΠΛΗΡΟΦΟΡΙΚΗ \"\" – About");
        setResizable(false);
        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent evt) {
                closeDialog(evt);
            }
        });

        okButton.setText("OK");
        okButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                okButtonActionPerformed(evt);
            }
        });

        cancelButton.setText("Cancel");
        cancelButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                cancelButtonActionPerformed(evt);
            }
        });

        jTextArea1.setColumns(20);
        jTextArea1.setFont(new java.awt.Font("Tahoma", 0, 11)); // NOI18N
        jTextArea1.setLineWrap(true);
        jTextArea1.setRows(5);
        jTextArea1.setText("");
        jTextArea1.setWrapStyleWord(true);
        jScrollPane1.setViewportView(jTextArea1);
```

```
73
            jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
            jLabel1.setIcon(new
                javax.swing.ImageIcon(getClass().getResource("/mppl08059/images/infoStart.png")));
                // NOI18N
76
            );
79
            getRootPane().setDefaultButton(okButton);
82          pack();
            java.awt.Dimension screenSize = java.awt.Toolkit.getDefaultToolkit().getScreenSize();
            java.awt.Dimension dialogSize = getSize();
85          setLocation((screenSize.width-dialogSize.width)/2,(screenSize.height-dialogSize.height)/2);
        }
88      private void okButtonActionPerformed(java.awt.event.ActionEvent evt) {
            doClose(RET_OK);
        }
91
        private void cancelButtonActionPerformed(java.awt.event.ActionEvent evt) {
            doClose(RET_CANCEL);
94      }

        /** Closes the dialog */
97      private void closeDialog(java.awt.event.WindowEvent evt) {
            doClose(RET_CANCEL);
        }
100
        private void doClose(int retStatus) {
            returnStatus = retStatus;
103         setVisible(false);
            dispose();
        }
106
        public static void main(String args[]) {
            java.awt.EventQueue.invokeLater(new Runnable() {
109
                public void run() {
                    AboutMppl08059 dialog = new AboutMppl08059(new javax.swing.JFrame(), true);
112                 dialog.addWindowListener(new java.awt.event.WindowAdapter() {

                        @Override
115                     public void windowClosing(java.awt.event.WindowEvent e) {
                            System.exit(0);
                        }
118                 });
                    dialog.setVisible(true);
                }
121         });
        }
        // Variables declaration - do not modify//GEN-BEGIN:variables
124     // End of variables declaration//GEN-END:variables
        private int returnStatus = RET_CANCEL;
}
```

Κώδικας 8.6: About



Σχήμα 8.8: About

Βιβλιογραφία

[1] Leslie Lamport, *LaTeX: A Document Preparation System*. Addison Wesley, Massachusetts, 2nd Edition, 1994

[2] Andrew Davison, Pro Java 6 3D Game Development:Java 3D, JOGL, JInput, and JOAL APIs. Apress, 2007

[3] Jim X. Chen,Edward J. Wegman, Foundations of 3D Graphics Programming Using JOGL and Java3D. Springer, 2006

[4] John Zukowski, The Definitive Guide to Java Swing. Apress, 2008

[5] Wikipedia LaTeX
http://en.wikibooks.org/wiki/LaTeX/

[6] Java :
http://www.java.com

[7] Java Monkey Engine :
http://www.jmonkeyengine.com
http://www.jmonkeyengine.org

[8] Lightweight Java Game Library (LWJGL) :
http://www.lwjgl.org/

[9] SICstus Prolog Homepage:
http://www.sics.se/sicstus

[10] Πανεπιστήμιο Πειραιά :
http://www.unipi.gr

[11] NetBeans IDE :
http://www.netbeans.org

[12] JBullet Physics :
http://bulletphysics.com