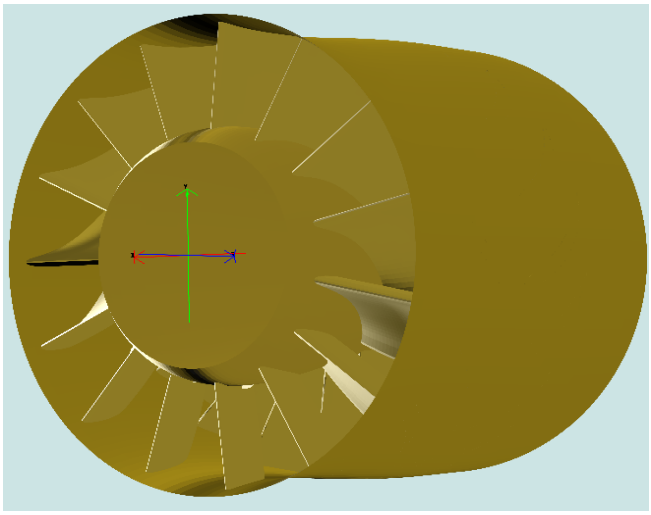
Πολυτεχνείο Κρήτης
Τμήμα Μηχανικών Παραγωγής & Διοίκησης

Ανάπτυξη λογισμικού για την τριδιάστατη απεικόνιση πολυβάθμιων πτερυγώσεων στροβιλομηχανών.



Διπλωματική εργασία
υπό
Θεολογίτη Αντώνη

Επιβλέπων: Ιωάννης Κ. Νικολός, Λέκτορας

Χανιά, Μάρτιος 2009

Περιεχόμενα

Ευρετήριο Εικόνων

Κεφάλαιο 1 - ΕΙΣΑΓΩΓΙΚΑ-ΑΝΑΛΥΣΗ ΤΗΣ ΛΕΙΤΟΥΡΓΙΑΣ ΚΑΙ ΤΩΝ ΧΑΡΑΚΤΗΡΙΣΤΙΚΩΝ ΤΗΣ JAVA ΚΑΙ ΤΗΣ ΒΙΒΛΙΟΘΗΚΗΣ JAVA 3D API.

Κεφάλαιο 2- ΑΝΑΛΥΣΗ ΤΗΣ ΜΕΘΟΔΟΛΟΓΙΑΣ ΚΑΤΑΣΚΕΥΗΣ ΤΟΥ ΤΡΙΣΔΙΑΣΤΑΤΟΥ T4T VIEWER.

Κεφάλαιο 3- ΑΝΑΛΥΣΗ ΚΩΔΙΚΑ ΤΟΥ VIEWER 3D-ΚΛΑΣΕΙΣ-ΣΥΝΑΡΤΗΣΕΙΣ-ΜΕΤΑΒΛΗΤΕΣ-ΑΠΟΤΕΛΕΣΜΑΤΑ

Κεφάλαιο 4- ΕΦΑΡΜΟΓΕΣ

Ευρετήριο Σχημάτων

Ευχαριστίες

Με το πέρας της παρούσας διπλωματικής εργασίας θα ήθελα να ευχαριστήσω τον Λέκτορα Νικολό Ιωάννη για την προθυμία, την υπομονή και την αμέριστη συμβολή του κατά την διεκπεραίωση της εργασίας. Επίσης θα ήθελα να ευχαριστήσω τα μέλη της τριμελούς επιτροπής Καθηγητή Κουϊκόγλου Βασίλειο και Επ. Καθηγητή Βλάσση Νικόλαο για την τιμή που μου κάνανε να συμμετέχουν στην εξέταση της παρούσας εργασίας.

Θα ήθελα επίσης να ευχαριστήσω τον υποψήφιο διδάκτορα κ. Σαρακηνό Σωτήριο και την κ. Κοίνη Γεωργία, που δημιούργησε το λογισμικό T4T που επεκτάθηκε με την παρούσα εργασία, για τις πολύτιμες γνώσεις που μου μετέφεραν, οι οποίες ήταν απαραίτητες για την ολοκλήρωση της διπλωματικής εργασίας μου.

Θα ήθελα ακόμη να ευχαριστήσω τους φίλους μου, με τους οποίους έζησα πολλές και ευχάριστες στιγμές σε όλη τη διάρκεια της φοιτητικής μου ζωής.

Τέλος, θέλω να πω ένα πολύ μεγάλο ευχαριστώ στην οικογένεια μου για την ηθική και υλική συμπαράστασή της όλα αυτά τα χρόνια.

Η παρούσα διπλωματική εργασία αφιερώνεται στη οικογένεια μου.

ΠΡΟΛΟΓΟΣ

Το αντικείμενο της Διπλωματικής Εργασίας είναι η ανάπτυξη κατάλληλου λογισμικού για την απεικόνιση (σε 3 διαστάσεις) πολυβάθμιων πτερυγώσεων στροβιλομηχανών. Η παρούσα εργασία αποτελεί συνέχεια της μεταπτυχιακής εργασίας της κ. Κοΐνη Γεωργίας, στην οποία παρουσιάζεται η ανάπτυξη της σχετικής μεθοδολογίας και του αντίστοιχου λογισμικού T4T (Tools for Turbomachinery), για την παραμετρική σχεδίαση τρισδιάστατων πολυβάθμιων πτερυγώσεων στροβιλομηχανών. Η επέκταση της μεταπτυχιακής εργασίας, μέσω της κατασκευής του συγκεκριμένου λογισμικού, κρίθηκε αναγκαία διότι παρέχεται πλέον η δυνατότητα στο χρήστη να έχει μια λεπτομερή τρισδιάστατη απεικόνιση της πτερύγωσης, που έχει σχεδιάσει στο στάδιο της παραμετρικής σχεδίασης, με αποτέλεσμα να έχει μια αληθοφανή άποψη της συγκεκριμένης γεωμετρίας. Ή γλώσσα προγραμματισμού που χρησιμοποιήθηκε για την κατασκευή του T4T Viewer 3D είναι η java. Η επιλογή της συγκεκριμένης γλώσσας προγραμματισμού κρίθηκε αναγκαία διότι αρχικά εξασφάλιζε την συμβατότητα των δύο εργασιών και κατά κύριο λόγο διότι η java προσφέρει αντικειμενοστραφή λογική, πολύ καλές βιβλιοθήκες γραφικών, ενώ είναι και ανεξάρτητη πλατφόρμας (λειτουργικού συστήματος). Ή βασική βιβλιοθήκη της java, που χρησιμοποιήθηκε σε αυτή την εργασία, είναι η JAVA 3D API (application programming interface). Για τα πλεονεκτήματα και για όλα τα χρήσιμα εργαλεία που παρέχει έχει ενσωματωθεί ιδιαίτερη αναφορά στο 1ο κεφάλαιο της εργασίας.

Η εργασία αναπτύσσεται σε 5 κεφάλαια, στα οποία γίνεται προσπάθεια να παρουσιαστεί στον αναγνώστη όλη η διαδικασία που ακολουθήσαμε. Επειδή πρόκειται για μια εργασία προγραμματιστικού περιεχομένου, θα χρησιμοποιούνται συχνά προγραμματιστικοί όροι, για τους οποίους θα δίδονται επεξηγήσεις με όσον το δυνατό πιο απλό τρόπο, ώστε να κατανοείται η ουσία και από κάποιον που δεν διαθέτει μεγάλη οικειότητα με το αντικείμενο του προγραμματισμού.

Στο 1ο κεφάλαιο της εργασίας γίνεται μια αναφορά στον ορισμό ενός application programming interface(API) ή αλλιώς διεπαφής προγραμματισμού εφαρμογών, ενώ παρουσιάζονται κάποια από τα ευρέως χρησιμοποιούμενα API σε διάφορες εφαρμογές και γίνεται μια ιδιαίτερη αναφορά στα σημαντικότερα από αυτά σήμερα. Επίσης γίνεται μια λεπτομερής ανάλυση στο πως λειτουργεί η γλώσσα προγραμματισμού java και παρουσιάζονται όλα τα συγκριτικά πλεονεκτήματα της σε σχέση με άλλες γλώσσες προγραμματισμού. Στην συνέχεια αναλύουμε την βιβλιοθήκη Java 3D API, που χρησιμοποιήθηκε στην εργασία αυτή. Παρουσιάζεται ο δόκιμος τρόπος με τον οποίο δομούνται τα προγράμματα που χρησιμοποιούν αυτή τη βιβλιοθήκη, όπως επίσης και ο τρόπος που κατασκευάζονται τριδιάστατα γραφικά, πως τα αντιλαμβάνεται ο υπολογιστής και πως τα αντιλαμβάνεται ο χρήστης.

Στο 2ο κεφάλαιο της εργασίας ασχολούμαστε με την ανάλυση όλης της διαδικασίας για την κατασκευή του Viewer 3D T4T. Αναλύοντας τις απαιτήσεις που υπήρχαν με την έναρξη της εργασίας αυτής, παρουσιάζεται ο τρόπος που δομήθηκε ο κατάλληλος κώδικας, ώστε να ικανοποιεί πλήρως τις απαιτήσεις αυτές. Δεν εισερχόμαστε στις λεπτομέρειες του κώδικα αλλά αναλύουμε την λογική και όλα τα βήματα που ακολουθήθηκαν ώστε να φτάσουμε στο επιθυμητό αποτέλεσμα.

Στο 3ο κεφάλαιο της εργασίας δίδονται λεπτομέρειες του κώδικα με έμφαση σε προγραμματιστικές εντολές ενώ αναλύονται συναρτήσεις και αλγόριθμοι που χρησιμοποιήθηκαν. Αυτό το κεφάλαιο αφορά κυρίως αναγνώστες που επιθυμούν να μάθουν ή είδη γνωρίζουν και επιθυμούν να εμπλουτίσουν την γνώση τους σε θέματα τρισδιάστατων γραφικών σε γλώσσα java και για το λόγο αυτό συμπεριλαμβάνονται αρκετές τεχνικές λεπτομέρειες.

Στο 4ο κεφάλαιο της εργασίας παρουσιάζουμε εικόνες από διάφορες πολυβάθμιες πτερυγώσεις στροβιλομηχανών, που προήλθαν από την παραμετρική σχεδίαση στο T4T. Επίσης παρουσιάζονται και εικόνες από την απεικόνιση των ίδιων στροβιλομηχανών αλλά με χρήση του λογισμικού CATIA, ώστε να γίνει σύγκριση των δυνατοτήτων τους.

Στο κεφάλαιο 5 προτείνουμε κάποιες μελλοντικές επεκτάσεις του Viewer3d T4T που θα τον καταστήσουν ακόμα πιο εύχρηστο και λειτουργικό.

Σε όλα τα κεφάλαια γίνεται αναφορά στην βιβλιογραφία είτε ονομαστικά είτε με τον κωδικό [αριθμός] όπου ο αριθμός αντιστοιχεί στον αριθμό της κατάταξης της βιβλιογραφίας.

1

ΕΙΣΑΓΩΓΙΚΑ-ΑΝΑΛΥΣΗ ΤΗΣ ΛΕΙΤΟΥΡΓΙΑΣ ΚΑΙ ΤΩΝ ΧΑΡΑΚΤΗΡΙΣΤΙΚΩΝ ΤΗΣ JAVA ΚΑΙ ΤΗΣ ΒΙΒΛΙΟΘΗΚΗΣ JAVA 3D API.

1.1 ΕΙΣΑΓΩΓΗ

Ξεκινώντας την ανάλυση της εργασίας είναι επιθυμητό να αναλυθούν πρώτα κάποιοι γενικοί όροι αλλά και να γίνει μια ιστορική αναδρομή και ανάλυση στην ανάπτυξη των γραφικών. Θα αναλυθεί η σημασία του API (application programming interface) και θα παρουσιαστούν διάφορες εφαρμογές του, κάνοντας και μια σχετική ανάλυση σε κάποιες από αυτές.

Ο όρος «εικονική πραγματικότητα» (Virtual Reality - VR) προτάθηκε αρχικά από τον Jaron Lanier, ιδρυτή του VPL Research (1989). Άλλοι παρόμοιοι όροι περιλαμβάνουν 'τεχνητή πραγματικότητα' (Myron Krueger, δεκαετία 1970), 'κυβερνοχώρος' (William Gibson, 1984), και πιο πρόσφατα, 'εικονικοί κόσμοι' και 'εικονικά περιβάλλοντα' (δεκαετία 1990) [Beier, 2001].

Η εικονική πραγματικότητα (ΕΠ) παρουσιάζει στον χρήστη ένα χώρο εργασίας που του δίνει την αίσθηση ότι η πληροφορία που παρουσιάζεται από τον Η/Υ συμπεριφέρεται όπως τα αντικείμενα του πραγματικού κόσμου **[14].** Η οθόνη του Η/Υ δεν αποτελεί πλέον ένα παράθυρο του κόσμου. Ο χρήστης αισθάνεται ότι βρίσκεται «μέσα» στον Η/Υ. Μπορεί ν' αλληλεπιδράσει με τα στοιχεία του εικονικού κόσμου, να μετακινηθεί μέσα σ' αυτόν και να τον αλλάξει. Πρόκειται για έναν «εικονικό κόσμο», έναν κόσμο δηλ. χωρίς υλική σύσταση, μια τεχνητή τρισδιάστατη απεικόνιση, που δημιουργείται μέσω των τεχνολογιών τρισδιάστατων γραφικών, κίνησης και εξομοίωσης ενός ισχυρού ηλεκτρονικού υπολογιστή και που επιτρέπει στον χρήστη να αλληλεπιδρά μ' αυτόν τον εικονικό κόσμο μέσω πράξεων, κινήσεων και εκτιμήσεων, που μοιάζουν με τις καθημερινές του ενέργειες στο πραγματικό του περιβάλλον. Με άλλα λόγια, η εικονική πραγματικότητα είναι μια αλληλεπίδραση (interface) ανθρώπου-μηχανής, που βιώνεται από τον άνθρωπο με τρόπο φυσικό και ενστικτώδη.

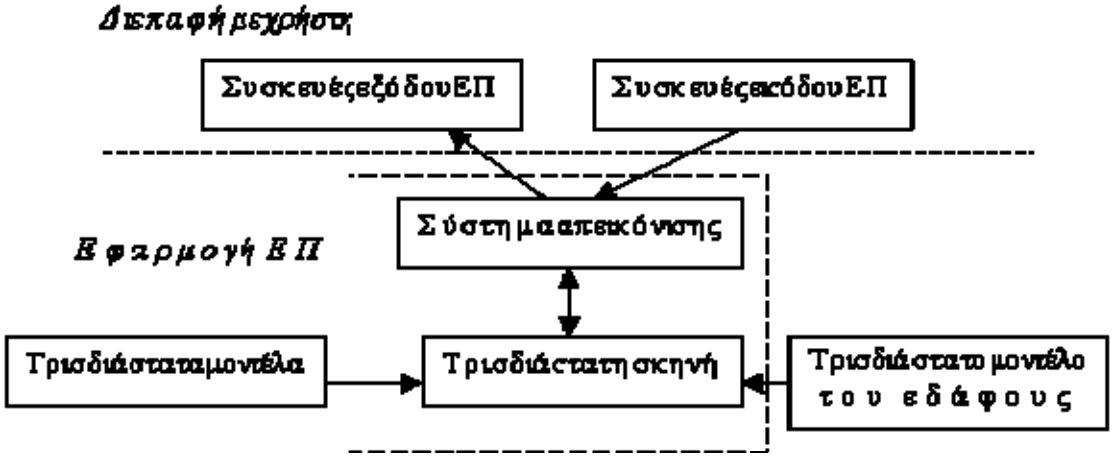
Το κύριο πλεονέκτημα από τη χρήση αλληλεπίδρασης ΕΠ με τον Η/Υ είναι ότι ο χρήστης μπορεί να αλληλεπιδρά με το σύστημα χρησιμοποιώντας ενέργειες και κινήσεις που μοιάζουν με τις καθημερινές του δραστηριότητες **[14]**. Οι άνθρωποι είμαστε πολύ καλοί στην αναγνώριση προτύπων, και στην αλληλεπίδραση με τριδιάστατα αντικείμενα του πραγματικού χώρου, πράγμα το οποίο μπορεί ν' αποδειχθεί πολύ χρήσιμο σε περιπτώσεις

που χρειάζεται να ληφθούν αποφάσεις από πολύπλοκα πληροφοριακά συστήματα όπως αυτά που περιγράφουμε παρακάτω.

Αρχικά ο όρος εικονική πραγματικότητα σήμαινε τη δημιουργία εικόνων από εικονικά περιβάλλοντα και την αναπαράστασή τους στο χρήστη χρησιμοποιώντας ειδικές συσκευές απεικόνισης. Ο χρήστης φορούσε ένα κράνος ΕΠ και μπορούσε να δει, όχι τον πραγματικό κόσμο, αλλά έναν εικονικό κόσμο που παρήγαγε ο Η/Υ. Σήμερα, όπως θα δούμε, υπάρχουν πολλές τεχνολογίες με τις οποίες ο χρήστης μπορεί να δει και να αισθανθεί έναν εικονικό κόσμο (με το κράνος ΕΠ να είναι ο πιο διάσημος από αυτούς).

Η ιστορία της εικονικής πραγματικότητας ξεκινά πίσω στα 1966, όταν ο Ivan Sutherland δημιούργησε το πρώτο (μονοσκοπικό) κράνος ΕΠ (η «Δαμόκλειος Σπάθη») [Beier 2001, Χαρίτος & Μαρτάκος, 1999]. Το 1970, ο Myron Kreuger κατασκεύασε το πρώτο σύστημα προβολικής ΕΠ, το VIDEOPLACE, όπου ο χρήστης μπορούσε να δει μια σκιά του εαυτού του μέσα σ' αυτό. Από κει και πέρα, οι εφαρμογές άρχισαν να πληθαίνουν, με την Boeing να είναι η πρώτη που δημιούργησε την Ενισχυμένη (Augmented) Πραγματικότητα (όπου ο χρήστης βλέπει τον πραγματικό κόσμο ενισχυμένο με εικονική πραγματικότητα)· εξελιγμένα στερεοσκοπικά κράνη ΕΠ για στρατιωτικές εφαρμογές κ.λπ. Το 1983, ο Zimmerman κατασκευάζει το πρώτο γάντι ΕΠ στα εργαστήρια της VPL.

Ένα σύστημα εικονικής πραγματικότητας (ΕΠ) αποτελείται από τα συστατικά που φαίνονται στο Σχήμα 1.1[Χαρίτος & Μαρτάκος, 1999] :



Σχήμα 1.1. *Συστατικά στοιχεία ενός συστήματος εικονικής πραγματικότητας* **[14]**

- *Σύστημα απεικόνισης (viewer) / τρισδιάστατη σκηνή*: Αυτά τα δυο στοιχεία συνδέονται στενά αφού η επιλογή του τρισδιάστατου περιβάλλοντος απεικόνισης ως 3D viewer υποδηλώνει μια τρισδιάστατη υλοποίηση του σκηνικού (3D scene). Η τρισδιάστατη σκηνή λαμβάνει συνεισφορές από ένα τρισδιάστατο μοντέλο του εδάφους και τρισδιάστατες απεικονίσεις των αντικειμένων του πραγματικού κόσμου. Και τα δυο μαζί αποτελούν την τρισδιάστατη μηχανή απεικόνισης (3D player engine).
- *Μοντέλο εδάφους*: μια γεωγραφική βάση δεδομένων του εδάφους σε τρισδιάστατη μορφή
- *Τρισδιάστατα μοντέλα του πραγματικού κόσμου,*
- *Συσκευές εισόδου ΕΠ,*
- *Συσκευές εξόδου ή απεικόνισης ΕΠ,*

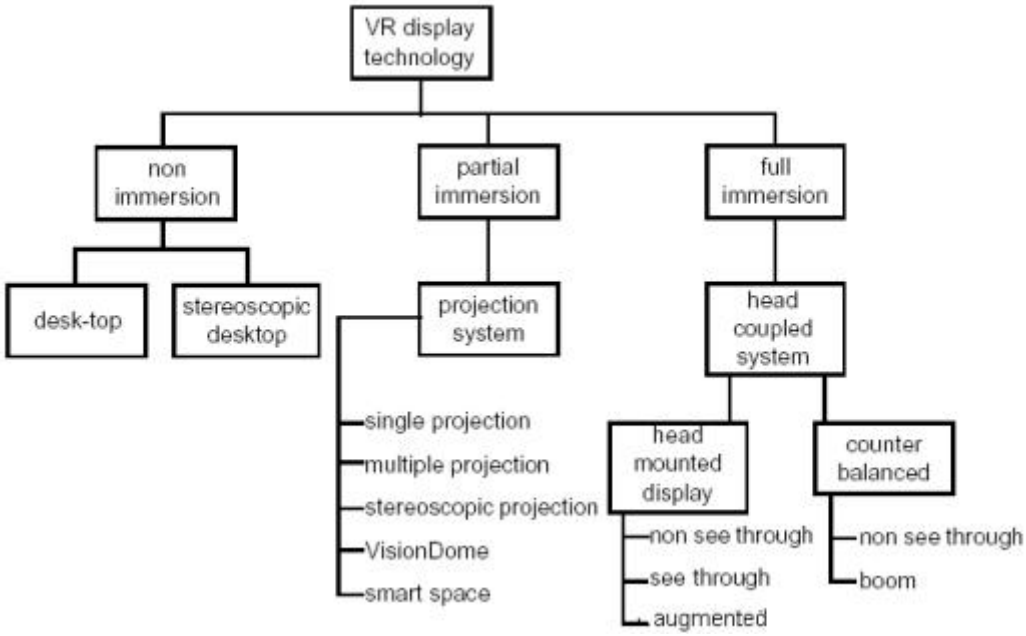
Οι χρήστες βλέπουν έναν τρισδιάστατο εικονικό κόσμο στις συσκευές εξόδου εικονικής πραγματικότητας και μπορούν ν' αλληλεπιδράσουν μ' αυτόν μέσω συσκευών εισόδου εικονικής πραγματικότητας. Ένα σύστημα απεικόνισης (viewer) περιέχει μια τρισδιάστατη

σκηνή, η οποία αποτελείται από τρισδιάστατα μοντέλα και (πιθανώς) από ένα μοντέλο του περιβάλλοντος, που καθοδηγεί τις συσκευές εισόδου και εξόδου. Η τρισδιάστατη σκηνή είναι μια δυναμική δομή δεδομένων, η οποία περιέχει όλη την πληροφορία που η εφαρμογή εικονικής πραγματικότητας πρόκειται να δείξει στο χρήστη. Τα τρισδιάστατα μοντέλα περιγράφουν τις κλάσεις των ορατών αντικειμένων της τρισδιάστατης σκηνής. Το μοντέλο του περιβάλλοντος περιγράφει το τοπίο σε τρισδιάστατη μορφή και η μηχανή απεικόνισης το απεικονίζει.

Ανάλογα με τη συσκευή οπτικής απεικόνισης που χρησιμοποιείται, μπορούμε να κατατάξουμε τις εικονικές πραγματικότητες στις ακόλουθες κατηγορίες **[14]**:

- *Εμβυθισμένη ΕΠ*, όταν ο χρήστης εμβυθίζεται στο περιβάλλον μέσω ενός ειδικού κράνους (Head Mounted Display - HMD ή BOOM)
- *Επιτραπέζια ΕΠ,* όταν χρησιμοποιείται απλά μια μονοσκοπική ή στερεοσκοπική οθόνη και η τρισδιάστατη απεικόνιση επιτυγχάνεται μέσω ειδικών γυαλιών
- *Προβολική ΕΠ,* όταν η απεικόνιση γίνεται μέσω μονοσκοπικής ή στερεοσκοπικής προβολής από πολλαπλές οθόνες που κυκλώνουν το χρήστη, και τέλος
- *Κατοπτρικοί κόσμοι,* όπου ο χρήστης βλέπει κάποια απεικόνιση του εαυτού του μέσα στο εικονικό περιβάλλον, με την οποία αλληλεπιδρά σε πραγματικό χρόνο.

Η παραπάνω κατηγοριοποίηση αντιστοιχίζεται στην ταξινόμηση που φαίνεται στο Σχήμα 1.2. Η εμβυθισμένη ΕΠ αντιστοιχεί στην πλήρη εμβύθιση (full immersion), η επιτραπέζια ΕΠ στην μη εμβύθιση (non immersion) και η προβολική ΕΠ στη μερική εμβύθιση (partial immersion).



Σχήμα 1.2. *Κατηγοριοποίηση των συσκευών απεικόνισης εξόδου [Traill et al ., 1997]* **[14]**

Τα μοναδικά χαρακτηριστικά της εμβυθισμένης ΕΠ περιγράφονται περιληπτικά ακολούθως [Beier, 2001]:

- Θέαση, η οποία γίνεται με την κίνηση του κεφαλιού, παρέχει μια φυσική διεπαφή για πλοήγηση στον τρισδιάστατο χώρο και επιτρέπει δυνατότητες όπως κοίταγμα τριγύρω, περίπατος, ακόμα και αεροπορική πορεία (fly-through) στα εικονικά περιβάλλοντα.
- Στερεοσκοπική θέαση, που αυξάνει την αίσθηση του βάθους και του χώρου.
- Ο εικονικός κόσμος αναπαρίσταται σε πλήρη αναλογία και συσχετίζεται με τις ανθρώπινες αναλογίες.

- Ρεαλιστικές αλληλεπιδράσεις με εικονικά αντικείμενα μέσω γαντιών και παρόμοιων συσκευών επιτρέπουν στον χειρισμό και τον έλεγχο των εικονικών κόσμων.
- Η πειστική αυταπάτη της πλήρους εμβύθισης στον εικονικό κόσμο μπορεί να αυξηθεί με ακουστικές, απτικές και άλλες μη οπτικές τεχνολογίες.
- Δικτυακές εφαρμογές, που επιτρέπουν διαμοιραζόμενα εικονικά περιβάλλοντα.

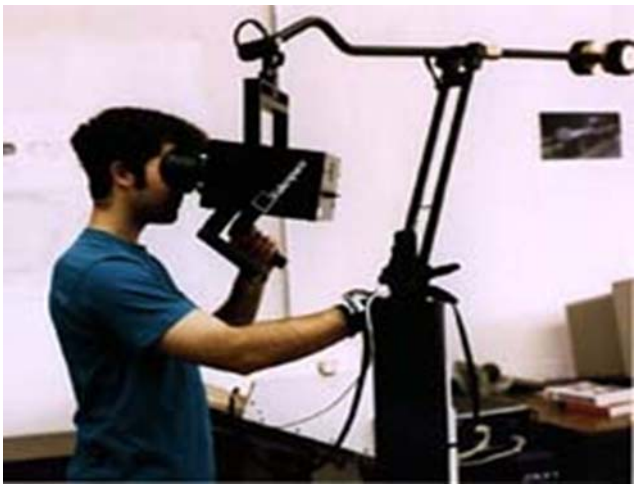
Την αίσθηση αυτή δίνουν ειδικές συσκευές hardware εικονικής πραγματικότητας όπως:

Συσκευές εξόδου ΕΠ:

- *Κράνη ΕΠ (Head Mounted Displays),* τα οποία διαθέτουν δυο μικροσκοπικές στερεοσκοπικές οθόνες (μια για κάθε μάτι), που προβάλλουν τις κινούμενες εικόνες του εικονικού περιβάλλοντος. Ο χρήστης αισθάνεται να «εμβυθίζεται» στο εικονικό περιβάλλον. Η παραίσθηση αυτή λέγεται «τηλεπαρουσία» και επηρεάζεται από πολλούς αισθητήρες κίνησης (motion trackers) που συλλέγουν τις κινήσεις του χρήστη και ανάλογα προσαρμόζουν την απεικόνιση των οθονών σε πραγματικό χρόνο. Έτσι ο χρήστης μπορεί να εξερευνήσει τον κόσμο εικονικής πραγματικότητας, αλλάζοντας οπτικές γωνίες, βασισμένος στην περιστροφή του κεφαλιού.

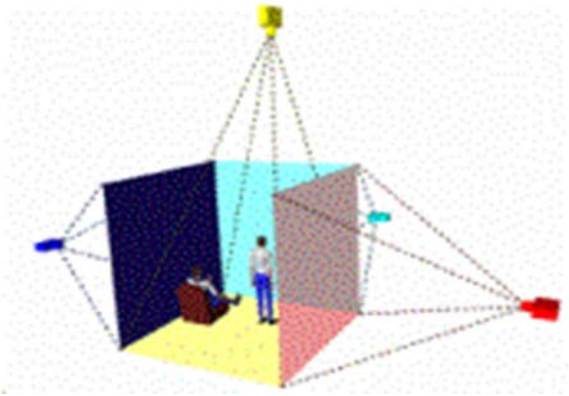
Εικόνα 1.1. Κράνος εικονικής πραγματικότητας [14]

- *Η πανκατευθυντική διοπτρική οθόνη (Binocular Omni-directional monitor - BOOM),* όπου οι οθόνες και το οπτικό σύστημα τοποθετούνται σ' ένα κουτί, το οποίο τοποθετείται σ' ένα βραχίονα πολλαπλών συνδέσμων. Ο χρήστης βλέπει τον εικονικό κόσμο κοιτώντας μέσα στο κουτί και μπορεί να καθοδηγήσει το κουτί σε οποιαδήποτε θέση μέσα στον όγκο λειτουργίας της συσκευής. Οι αισθητήρες κίνησης βρίσκονται στους συνδέσμους του βραχίονα, που κρατάει το κουτί.



Εικόνα 1.2. Συσκευή BOOM [14]

- Το *Σύστημα Αυτόματου Εικονικού Περιβάλλοντος Σπηλαίου (Cave Automatic Virtual Environment - CAVE)* παρέχει την ψευδαίσθηση της εμβύθισης, με το να προβάλλει στερεοσκοπικές εικόνες στους τείχους και στο δάπεδο ενός κυβικού δωματίου. Μια ομάδα ατόμων, η οποία φοράει τρισδιάστατα γυαλιά, μπορεί να μετακινηθεί ελεύθερα στο CAVE, ενώ αισθητήρες κίνησης συνεχώς αναπροσαρμόζουν τη στερεοσκοπική προβολή του διευθύνοντος ατόμου.



Εικόνα 1.3. Αρχή της λειτουργίας του Cave [14]

- Τρισδιάστατα γυαλιά (LCD shutter glasses), τα οποία χρησιμοποιούνται συνήθως με μονοσκοπικές αλλά και στερεοσκοπικές οθόνες και παρέχουν την αίσθηση του βάθους στις δισδιάστατες οθόνες.

Εικόνα 1.4.Τριδιάστατα γυαλιά [14]

Συσκευές εισόδου ΕΠ:

- Γάντια, που είναι εξοπλισμένα με συσκευές αφής ή/και "force-feedback", που δίνουν την αίσθηση της αφής στον χρήστη, ώστε να μπορεί να σηκώσει και να μετακινήσει αντικείμενα στο εικονικό περιβάλλον.

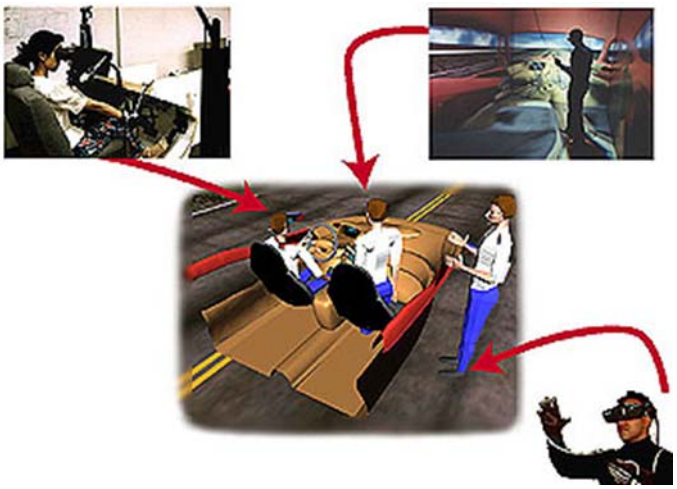
Εικόνα 1.5.Γαντι εικονικής πραγματικότητας [14]

- Συσκευές, που χρησιμοποιούνται για την πλοήγησή μας στον τρισδιάστατο χώρο και την επιλογή 3-σδιάστατων αντικειμένων, περιλαμβάνουν: Τρισδιάστατο ποντίκι (spacemouse), μπίλια (spaceball), ραβδί, χειριστήριο (joystick) κ.ά.) και τρισδιάστατη μπίλια – spaceball.

Εικόνα 1.6.Τρισδιάστατο ποντίκι (αριστερά) - spacemouse (κέντρο) - τρισδιάστατη μπίλια (δεξιά) [14]

Συνεργασιακά ή διαμοιραζόμενα εικονικά περιβάλλοντα

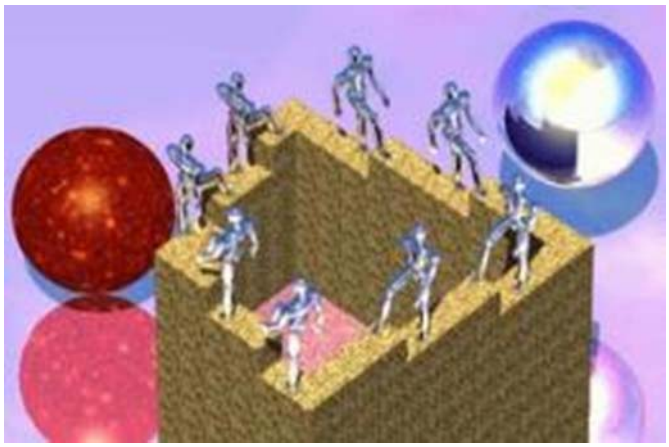
Στην Εικόνα 1.7, τρεις δικτυωμένοι χρήστες, που βρίσκονται σε διαφορετικές τοποθεσίες, συναντιόνται στον εικονικό κόσμο, χρησιμοποιώντας μια συσκευή BOOM, ένα σύστημα CAVE, και ένα HMD. Και οι τρεις χρήστες βλέπουν τον ίδιο εικονικό κόσμο από τη δική τους οπτική γωνία. Κάθε χρήστης αναπαρίσταται στους άλλους συμμετέχοντες ως ένας εικονικός άνθρωπος (ενσάρκωση - avatar). Όλοι οι χρήστες μπορούν να δουν και να αλληλεπιδράσουν με τους άλλους χρήστες και με τον εικονικό κόσμο σαν ομάδα.



Εικόνα 1.7. Παράδειγμα διαμοιραζόμενου εικονικού περιβάλλοντος [14]

Εικονική πραγματικότητα και Internet - Η γλώσσα VRML

Η εικονική πραγματικότητα είναι δυνατή και μέσω του Internet μέσω της γλώσσας VRML, που μας δίνει τη δυνατότητα να παρακολουθήσουμε εικονικούς κόσμους μέσα από τον φυλλομετρητή μας. Αποτελεί προσθήκη στη γλώσσα HTML (τη γλώσσα στην οποία γράφονται οι ιστοσελίδες) και επιτρέπει την απεικόνιση τριδιάστατων κόσμων με ενσωματωμένους υπερδεσμούς (hyperlinks) στο Διαδίκτυο. Οι ιστοσελίδες μετατρέπονται σε ιστο-χώρους (home spaces). Η αλληλεπίδραση με μια σελίδα VRML γίνεται με τη χρήση ενός ποντικιού σε μια οθόνη και επομένως είναι μη εμβυθισμένη.



Εικόνα 1.8. Παράδειγμα ιστοσελίδας VRML [14]

Εφαρμογές Εικονικής Πραγματικότητας

Οι εφαρμογές εικονικής πραγματικότητας μεταφέρουν το χρήστη σε ένα εικονικό περιβάλλον, που έχει κατασκευαστεί εξ' ολοκλήρου από ηλεκτρονικό υπολογιστή και που μπορεί να εξομοιώσει την πραγματικότητα μέσα από τη χρήση ειδικών συσκευών. Η ΕΠ αναμένεται να επαναδιαμορφώσει τη διεπαφή ανθρώπου και πληροφορικής τεχνολογίας, προσφέροντας νέους τρόπους επικοινωνίας και πληροφόρησης, απεικόνισης, και τη δημιουργική έκφραση ιδεών. Εκτός από το χώρο του θεάματος και των βιντεοπαιχνιδιών, η εικονική πραγματικότητα βρίσκει εφαρμογές σε πολλούς κλάδους της επιστήμης [CHIP]:

- Στη Χημεία, προσφέρει τρισδιάστατη απεικόνιση των χημικών ενώσεων και αντιδράσεων, προσφέροντας μια αίσθηση ασφαλείας στους χημικούς.
- Στην Ιατρική, όπου η εικονική πραγματικότητα έχει κάνει θαύματα, με αποτέλεσμα πολλές κακώσεις και αρρώστιες να αντιμετωπίζονται άμεσα και αποτελεσματικά. Οι εφαρμογές εικονικής πραγματικότητας στην ιατρική εντοπίζονται σε τρεις τομείς: εγχείριση ανοικτής καρδιάς, ενδοσκοπισμός και ραδιοχειρουργική. Σε συνδυασμό με την τηλεϊατρική είναι δυνατή η ανταλλαγή δεδομένων μεταξύ ιατρών που βρίσκονται σε μεγάλη απόσταση μεταξύ τους.
- Στην Αρχαιολογία, όπου η εικονική πραγματικότητα και η ενισχυμένη (augmented) πραγματικότητα δίνει τη δυνατότητα αναπαράστασης ολόκληρων μνημείων και πόλεων της ιστορίας. Παράδειγμα τέτοιας εφαρμογής είναι η αναπαράσταση του Παρθενώνα και της αρχαίας Αθήνας κατά τον «Χρυσό Αιώνα» του Περικλή.
- Τέλος, η εικονική πραγματικότητα εισάγεται σιγά-σιγά και σε στρατιωτικές εφαρμογές, για εκπαίδευση, για εκτίμηση σχεδίασης (virtual prototyping), αρχιτεκτονική προεπισκόπιση, εργονομικές μελέτες, εξομοίωση συναρμολογημένων ακολουθιών και εργασιών συντήρησης, βοήθεια για τους ανάπηρους, διασκέδαση και πολλά άλλα.

1.2 NETBEANS IDE 5.5.1

Το πρόγραμμα το οποίο επιλέχθηκε για την ανάπτυξη του κώδικα σε java είναι το **NetBeans IDE 5.5.1.** Το NetBeans **[11]** είναι ένα επιτυχημένο ερευνητικό έργο ανοιχτής πηγής (open source) με μεγάλο αριθμό χρηστών, μια αναπτυσσόμενη κοινωνία, που περιλαμβάνει κοντά στους 100 (και πλέον!) συνεργάτες παγκοσμίως. Η Sun Microsystems ίδρυσε το ερευνητικό έργο ανοιχτής πηγής NetBeans τον Ιούνιο του 2000 και συνεχίζει να είναι ο κύριος ανάδοχος.

Σήμερα δύο ερευνητικά έργα υπάρχουν: Το NetBeans IDE και το NetBeans Platform.

Το NetBeans IDE είναι ένα περιβαλλοντικό ανάπτυγμα IDE - ένα εργαλείο για τους προγραμματιστές για να γράψουν, να κάνουν compile, debug και να αναπτύξουν προγράμματα. Είναι γραμμένο σε Java - αλλά μπορεί να υποστηρίξει όλες τις γλώσσες προγραμματισμού. Υπάρχει επίσης ένας μεγάλος αριθμός υπομονάδων (modules), που βοηθάνε στην επέκταση της λειτουργικότητας του NetBeans IDE. Το NetBeans IDE είναι ένα ελεύθερο προϊόν δίχως περιορισμούς στον τρόπο χρησιμοποίησής του.

Διαθέσιμο επίσης είναι το NetBeans Platform. Είναι ένα εκτατό θεμέλιο αποτελούμενο από υπομονάδες (modular) που χρησιμοποιείται σαν βάση λογισμικού για τη δημιουργία μεγάλων επιτραπέζιων (desktop) εφαρμογών. Οι ISV συνεργάτες διαθέτουν προσθήκες, επιπρόσθετα προγράμματα (plug-ins) που εύκολα συνενώνονται στην πλατφόρμα και μπορούν επίσης να χρησιμοποιηθούν για την ανάπτυξη άλλων εργαλείων και λύσεων.

Και τα δύο τα προϊόντα είναι ανοιχτής πηγής (open source) και ελεύθερα για εμπορική ή μη χρήση. Ο κώδικας πηγής (source code) είναι διαθέσιμος για επαναχρησιμοποίηση κάτω από το Common Development and Distribution License (CDDL).

1.3 API (Application Programming Interface)

API : Application programming interface **[11]** ή διεπαφή προγραμματισμού εφαρμογών ή εν συντομία API, καλείτε η διεπαφή των προγραμματιστικών διαδικασιών που ένα λειτουργικό σύστημα, βιβλιοθήκη ή εφαρμογή παρέχει, προκειμένου να επιτρέπει να γίνονται προς αυτό αιτήσεις από άλλα προγράμματα ή / και ανταλλαγή δεδομένων. Υπάρχουν 2 μορφές API:

- Το language-depended API, το οποίο είναι γραμμένο σε μία συγκεκριμένη γλώσσα προγραμματισμού. Τα προγράμματα πρέπει να κάνουν χρήση της συγκεκριμένης γλώσσας προκειμένου να μπορούν να αξιοποιήσουν το API.
- Το language-independed API, το οποίο είναι γραμμένο με τέτοιο τρόπο ώστε να μπορεί να χρησιμοποιηθεί από προγράμματα που χρησιμοποιούν διαφορετικές γλώσσες προγραμματισμού.

Ένας από τους βασικούς σκοπούς μίας διεπαφής είναι να ορίζει και να διατυπώνει το σύνολο των λειτουργιών-υπηρεσιών που μπορεί να παρέχει μια βιβλιοθήκη ή ένα λειτουργικό σύστημα σε άλλα προγράμματα, χωρίς να επιτρέπει πρόσβαση στον κώδικα που υλοποιεί αυτές τις υπηρεσίες. Η διεπαφή, ένα "συμβόλαιο κλήσης" μεταξύ καλούντος και καλούμενου, διαχωρίζει την προγραμματιστική υλοποίηση κάποιων υπηρεσιών από τη χρήση τους.

Παραδείγματος χάριν το ταχυδρομείο παρέχει την υπηρεσία της αποστολής γραμμάτων. Οι κανόνες οι οποίοι πρέπει να ακολουθηθούν για την υποβολή ενός αιτήματος αποστολής (φορμά διεύθυνσης παραλαβής, γραμματόσημο κ.λπ.) είναι καλώς ορισμένοι, αλλά το πώς θα υλοποιηθεί στην πράξη αυτό το αίτημα αφορά έναν ολόκληρο μηχανισμό υπαλλήλων εν πολλοίς αθέατο στον χρήστη της υπηρεσίας. Στο εν λόγω παράδειγμα διεπαφή είναι οι υπηρεσίες που παρέχονται στους πελάτες, οι οποίες συνήθως είναι γραμμένες σε ένα φυλλάδιο, τη διεπαφή του ταχυδρομείου προς τους χρήστες του.

Έτσι για παράδειγμα το λειτουργικό σύστημα Windows έχει τη δική του διεπαφή (κλήσεις συστήματος), το φορμά της οποίας διατίθεται δωρεάν από την κατασκευάστρια εταιρεία Microsoft, και η οποία περιγράφει τους τρόπους αξιοποίησης από προγράμματα χρήστη του συνόλου των υπηρεσιών που παρέχει το λειτουργικό. Το τμήμα του λειτουργικού συστήματος το οποίο υλοποιεί τις υπηρεσίες που περιγράφονται στη διεπαφή, συνήθως στον πυρήνα του, λέμε ότι είναι η *υλοποίηση της διεπαφής*.

ΠΑΡΑΔΕΙΓΜΑΤΑ ΠΡΟΓΡΑΜΜΑΤΩΝ ΠΟΥ ΕΙΝΑΙ API:

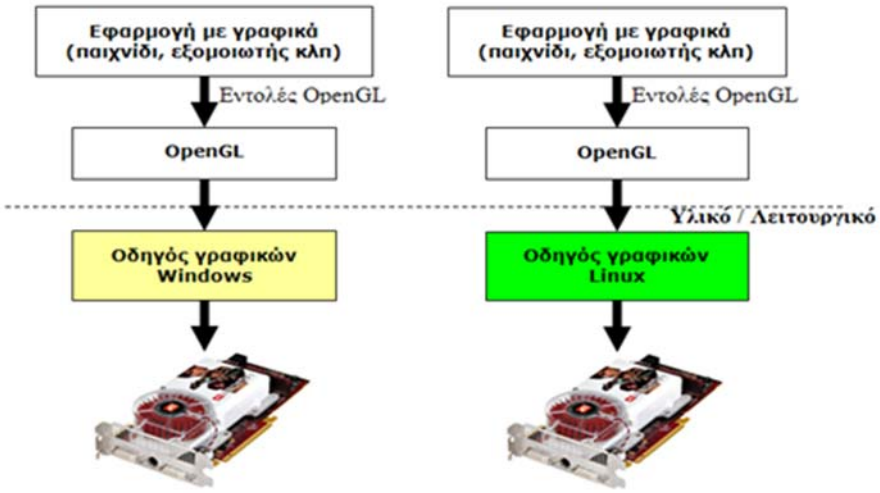
- The PC BIOS call interface
- Comedi Data Acquisition API
- Single UNIX Specification (SUS)
- Windows API
- The various Java Platform Edition APIs (Micro, Standard, Enterprise)
- ASPI for SCSI device interfacing
- Carbon and Cocoa for the Macintosh OS
- iPhone API
- OpenGL cross-platform 3D graphics API
- DirectX for Microsoft Windows
- Simple DirectMedia Layer (SDL)

- Google Maps API
- MediaWiki API
- YouTube API
- PayPal Payment Pro
- Facebook API
- Drupal API (Drupal)

1.4 API OpenGL-API DirectX

Δύο από τις σημαντικότερες παραπάνω API εφαρμογές αποτελεί το **OpenGL**(open graphics library) και το **DirectX** της Microsoft. **[8]**

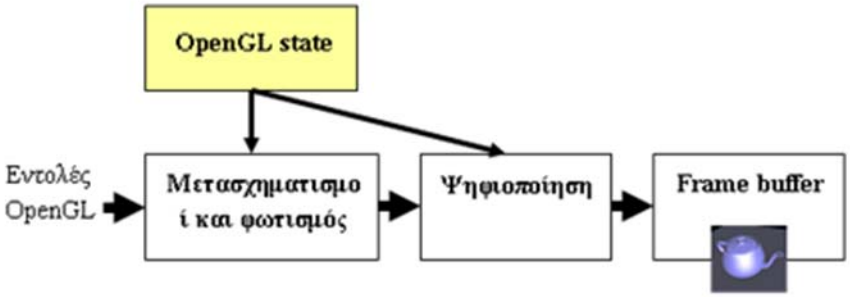
- Το **OpenGL** είναι ένα API για την επικοινωνία με την κάρτα γραφικών. Το OpenGL είναι ένα σύνολο εντολών που μας επιτρέπει την δημιουργία τριδιάστατων γραφικών. Δεν είναι γλώσσα προγραμματισμού. Αποτελείται από περίπου 300 διαφορετικές εντολές και δρα σαν ένα ενδιάμεσο στρώμα ανάμεσα στην εφαρμογή και στην κάρτα γραφικών, που θα αναλάβει να απεικονίσει τα γραφικά στην οθόνη, κρύβοντας λεπτομέρειες υλοποίησης του υλικού και των οδηγών του.



Σχήμα 1.3. *Λειτουργικός ρόλος της OpenGL.* **[8]**

Το OpenGL είναι σχεδιασμένο ώστε να είναι ανεξάρτητο από το υλικό (hardware) και από λειτουργικό σύστημα - παραθυρικό περιβάλλον. Μπορεί να χρησιμοποιηθεί σχεδόν σε όλες τις «γνωστές» γλώσσες προγραμματισμού (C, C++, JAVA, Visual Basic, Delphi). Παρέχει ένα σύνολο εντολών για την επικοινωνία με το υλικό και υποστηρίζεται από όλες τις εταιρείες κατασκευής καρτών γραφικών. Είναι το τρέχον standard παρά τις προσπάθειες της Microsoft με το DirectX. Για αυτό το λόγο χρησιμοποιείται ευρέως σε πολλές εφαρμογές και παιχνίδια. Το OpenGL είναι μια *μηχανή καταστάσεων* (state machine) με την έννοια ότι είμαστε διαρκώς σε μια κατάσταση μέχρι με μια εντολή να μεταβούμε σε κάποια άλλη. Για παράδειγμα, όταν σε μια εφαρμογή χρωματίζουμε αντικείμενα, μπορούμε να θέσουμε το χρώμα σε κόκκινο, κίτρινο, … και θα χρησιμοποιούμε το ίδιο χρώμα συνέχεια, μέχρι να ορίσουμε ένα καινούριο χρώμα ή να κλείσουμε την εφαρμογή. Εφόσον λοιπόν η εφαρμογή

καθορίσει το περιβάλλον που θα χρησιμοποιήσει το OpenGL για να απεικονίσει το μοντέλο μας (χρώματα, υφές, πηγές φωτός, κάμερα κλ.π.), περνάει στο πρώτο στάδιο κατά το οποίο θα μετασχηματίσει και θα φωτίσει (transform and lighting) τα σημεία (vertices) του μοντέλου. Στη συνέχεια το OpenGL περνά στο στάδιο της ψηφιοποίησης, το οποίο λαμβάνει όλες τις πληροφορίες και την γεωμετρία από το προηγούμενο στάδιο (του μετασχηματισμού) και παράγει την τελική, ψηφιακή, εικόνα. Η εικόνα αντιγράφεται στην μνήμη του frame buffer και φτάνει έτσι στην οθόνη του υπολογιστή.



Ας δούμε σχηματικά στο σχήμα 1.4 τα βήματα που κάνει το OpenGL από τη στιγμή που πάρει τις εντολές γραφικών από την εφαρμογή μας μέχρι την στιγμή δημιουργίας της τελικής εικόνας (rendered image).

Σχήμα 1.4. *Στάδια λειτουργίας του OpenGL* **[8]**

.

- **DirectX, [3]** παλαιότερα γνωστό ως το παιχνίδι SDK, είναι ένας όρος που δίνεται για μια συλλογή API(s) που χρησιμοποιούνται και υποστηρίζονται από προγράμματα που σχετίζονται με πολυμέσα, ειδικότερα για προγραμματισμό παιχνιδιών και video σε πλατφόρμες Microsoft. Με τον όρο DirectX εννοούμε μία σειρά από πρότυπα, καινοτομίες και τεχνολογίες, που αφορούν στην απεικόνιση γραφικών μέσα από τα Windows (για την ακρίβεια, περιλαμβάνουν και άλλα στοιχεία πλην των γραφικών), και ειδικότερα στα εργαλεία που παρέχει η Microsoft στους προγραμματιστές, προκειμένου να αναπτύξουν εφαρμογές και παιχνίδια, που θα εκμεταλλεύονται αυτές τις δυνατότητες. Γι' αυτό και πολλές φορές θα δει κανείς το DirectX να αναφέρεται ως API (Application Programming Interface). Όλα τα API, που αποτελούν το πακέτο, ξεκινούν την ονομασία τους με την λέξη Direct, όπως το Direct3D, DirectDraw, DirectMusic, DirectSound, DirectPlay. DirectX είναι ο γενικός όρος που δόθηκε για όλα αυτά τα API. Τα μέρη του DirectX αναλύονται στη συνέχεια:

- **Direct3D. [3]** Αυτό επιτρέπει την εμφάνιση των κινούμενων γραφικών τριών διαστάσεων στην οθόνη του υπολογιστή. Το Direct3D έχει σχεδιαστεί ώστε να παρέχει μια ισχυρή σύνδεση ανάμεσα στην κάρτα γραφικών του υπολογιστή και στα προγράμματα λογισμικού, που μπορούν να αποδίδουν αντικείμενα τριών διαστάσεων (3-D). Όσο ταχύτερα επεξεργάζεται ο υπολογιστής τα κινούμενα γραφικά τριών διαστάσεων, τόσο πιο ρεαλιστικά θα φαίνονται τα αντικείμενα 3-D, ο φωτισμός και η κίνηση στην οθόνη.

- **DirectDraw. [3]** Αυτό βοηθά στη δημιουργία οπτικών εφέ δύο διαστάσεων (2-D). Η κάρτα γραφικών του υπολογιστή καθώς και πολλά προγράμματα λογισμικού χρησιμοποιούν το

DirectDraw για τη μεταξύ τους επικοινωνία πριν την αποστολή της ολοκληρωμένης οπτικής εικόνας στην οθόνη. Τα παιχνίδια υπολογιστή, τα πακέτα γραφικών 2-D και οι δυνατότητες συστήματος των Windows, χρησιμοποιούν όλα το DirectDraw.

- **DirectSound. [3]** Αυτό ενισχύει την απόδοση των ηχητικών εφέ στον υπολογιστή και επιτρέπει τη δημιουργία διακριτικών εφέ στη μίξη και στην αναπαραγωγή του ήχου. Παρέχει μια σύνδεση μεταξύ των προγραμμάτων λογισμικού και του υλικού στον υπολογιστή. Το DirectSound παρέχει στα προγράμματα λογισμικού πολυμέσων, όπως τα παιχνίδια και οι ταινίες, επιτάχυνση υλικού, δυνατότητες μίξης και πρόσβαση στην κάρτα ήχου.

Παρακάτω παρουσιάζονται 2 εικόνες που χρησιμοποιούν το DirectX. Οι 2 εικόνες είναι θεματικά οι ίδιες αλλά διαφέρουν ως προς την έκδοση του DirectX που χρησιμοποιούν. Είναι εμφανής η εξέλιξη των γραφικών με το πέρασμα του χρόνου.

Εικόνα 1.9. Γραφικά από DirectX 9 [3]

Εικόνα 1.10. Γραφικά από DirectX 10 [3]

1.5 Εισαγωγή στη Java

Η Java **[11]** αυτή τη στιγμή αντιπροσωπεύει την τελευταία γενιά των γλωσσών προγραμματισμού. Λόγω του ότι τα πάντα πλέον αρχίζουν να περιστρέφονται γύρω από το διαδίκτυο με τη μορφή υπηρεσιών, η Java είναι ένα πολύ εύχρηστο εργαλείο για την ανάπτυξη εφαρμογών που συνδυάζουν την ευκολία υλοποίησης, τη δυνατότητα να τρέχουν σε πολλά διαφορετικά περιβάλλοντα, να μην χρειάζεται ο χρήστης να κάνει καμία εγκατάσταση στον υπολογιστή του προκειμένου να τρέξει κάποια εφαρμογή (π.χ. τα applets δεν προαπαιτούν να είναι εγκατεστημένο κάτι, τρέχουν μέσα από τον ίδιο τον browser), τη δυνατότητα να είναι δικτυακές κ.λπ.

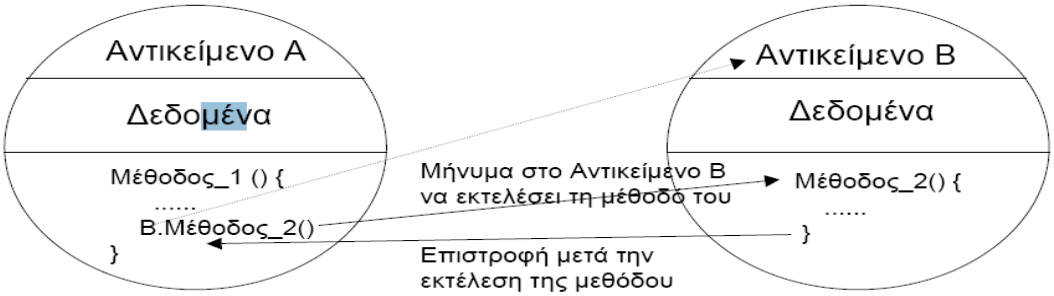
Επιπλέον η ίδια η Sun έχει καταφέρει να κρατήσει τη Java ελεύθερη, και δίνει μάλιστα και τον κώδικά της σε όποιον το ζητήσει. Ακόμη έχουν γίνει πάρα πολλές προσπάθειες για ανάπτυξη πακέτων και APIs που κάνουν σχεδόν οτιδήποτε, από ήχο και βάσεις δεδομένων μέχρι κλήση και εκτέλεση απομακρυσμένου κώδικα (μέσω RMI και CORBA), ενώ υπάρχει πάρα πολύ μεγάλη υποστήριξη πάνω σε αυτά τα θέματα.

Η java είναι μια γλώσσα προγραμματισμού που ενσωματώνει αντικειμενοστραφή (object-oriented) λογική. Σύμφωνα με την συγκεκριμένη λογική το πεδίο εφαρμογής θεωρείται ως ένα σύνολο αντικειμένων οργανωμένων σε ιεραρχία(ες), τα οποία επικοινωνούν μεταξύ τους με μηνύματα. Συνεπτυγμένα τα χαρακτηριστικά της Java είναι : **[15]**

- Αντικειμενοστραφής (object oriented)
- Ανεξάρτητη Πλατφόρμας (Platform independent)
- Γλώσσα (δια)δικτύου (Internet language)
- Δυναμική (Dynamic)
- Πολυνηματική (Multi-threaded)

Οι αντικειμενοστραφείς γλώσσες καλούνται και οντοκεντρικά μοντέλα προγραμματισμού. Στην εικόνα 1.11 παρουσιάζεται σχηματικά ένα οντοκεντρικκό μοντέλο προγραμματισμού.

Οντοκεντρικό Μοντέλο Προγραμματισμού



Εικόνα 1.11. Οντοκεντρικό μοντέλο προγραμματισμού [15]

Ο οντοκεντρικός προγραμματισμός χαρακτηρίζεται από τις εξής ιδιότητες για τα αντικείμενα που δημιουργεί: **[15]**

- Τα αντικείμενα κατηγοριοποιούνται σε **κλάσεις**. Οι κλάσεις αποτελούν τύπους δεδομένων οι οποίοι μπορούν να δημιουργούν ένα η περισσότερα αντικείμενα.
- Αντικείμενα της ίδια κλάσης έχουν τα ίδιες ιδιότητες και την ίδια συμπεριφορά.
- Κάθε αντικείμενο ανήκει σε μία κλάση.
- Οι κλάσεις ορίζονται από τον προγραμματιστή και αποτελούν ουσιαστικά ένα νέο τύπο δεδομένων, ενώ τα αντικείμενα που εμπεριέχουν αποτελούν τις μεταβλητές αυτού του τύπου.

Ένα από τα χαρακτηριστικά γνωρίσματα των οντοκεντρικών γλωσσών, όπως είναι η java, είναι ότι παρέχουν τη δυνατότητα να ορίζουμε ειδικεύσεις κλάσεων. Οι κλάσεις που είναι ειδικεύσεις άλλων κλάσεων ονομάζονται υποκλάσεις ή κλάσεις-τέκνα. Οι κλάσεις που είναι γενικεύσεις άλλων κλάσεων ονομάζονται υπερκλάσεις ή κλάσεις-γονείς ή βασικές κλάσεις. Η σχέση από μία γονέα-κλάση προς μία κλάση-παιδί ονομάζεται σχέση **κληρονομικότητας.** Οι υποκλάσεις, ανάλογα με τον τύπο της κληρονομικότητας, έχουν συνήθως όλες τα μέλη δεδομένων και τις μεθόδους των κλάσεων-γονέων τους. Επιπλέον ορίζουν νέες μεθόδους η τροποποιούν κατάλληλα με νέο κώδικα τις μεθόδους που κληρονομούν από τους 'γονείς' τους. Το παράδοξο με την σχέση κληρονομικότητας είναι ότι η υπερκλάση έχει ουσιαστικά ένα υποσύνολο των μεθόδων της υποκλάσης.

Η βασική ιδέα με την χρήση της κληρονομικότητας είναι να έχουν τα προγράμματα μας την δυνατότητα:

1. Να επαναχρησιμοποιούμε κώδικα ο οποίος είναι κοινός για 2 ή περισσότερες κλάσεις.
2. Να δηλώνουμε την διεπαφή μίας η περισσοτέρων κλάσεων με την μορφή δημόσιων μεθόδων που κληρονομούνται από μία ή περισσότερες κοινές υπερκλάσεις.

Ένα σημείο που πρέπει να δώσουμε ιδιαίτερη έμφαση είναι το ποιες μέθοδοι και γενικότερα μέλη του προγράμματος είναι κληρονομήσιμα, δηλαδή τα δικαιώματα προσπέλασης στην κληρονομικότητα. Συγκεκριμένα:

- Οι υποκλάσεις έχουν πρόσβαση σε μέλη που έχουν δηλωθεί public ή protected στη βασική κλάση.

- Οι υποκλάσεις έχουν πρόσβαση σε μέλη τα οποία δεν έχουν δήλωση προσπέλασης αλλά ανήκουν στο ίδιο πακέτο με την βασική κλάση.
- Στην εικόνα 1.12 παρουσιάζεται την λογική της προσπέλασης στην κληρονομικότητα.

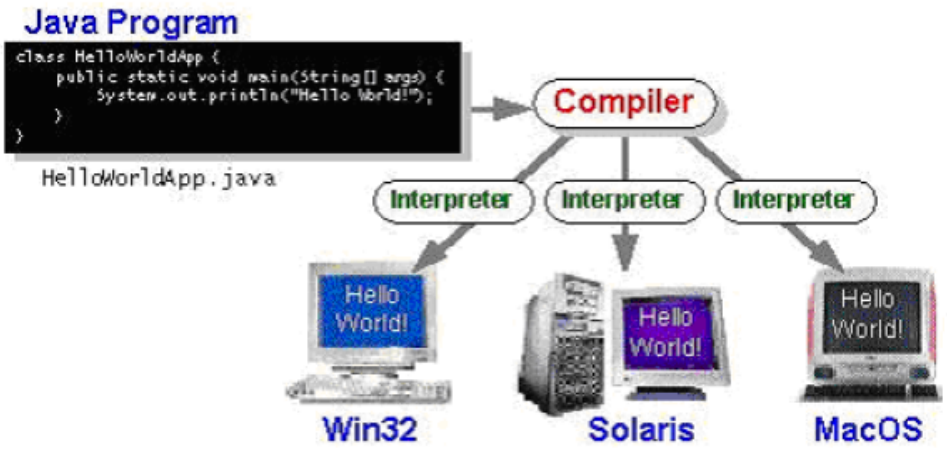
Προσπέλαση

Δυνατότητα Προσπέλασης Μεταβλητής	public	protected	Χωρίς Προσδιοριστή (package)	private
Στην ίδια κλάση	Ναι	Ναι	Ναι	Ναι
Από κλάση του ίδιου πακέτου	Ναι	Ναι	Ναι	Όχι
Από μια υποκλάση του ιδίου πακέτου	Ναι	Ναι	Ναι	Όχι
Από μια υποκλάση εκτός του πακέτου	Ναι	Ναι	Όχι	Όχι
Από οποιαδήποτε κλάση	Ναι	Όχι	Όχι	Όχι

Εικόνα 1.12.Προσπέλαση στην κληρονομικότητα [15]

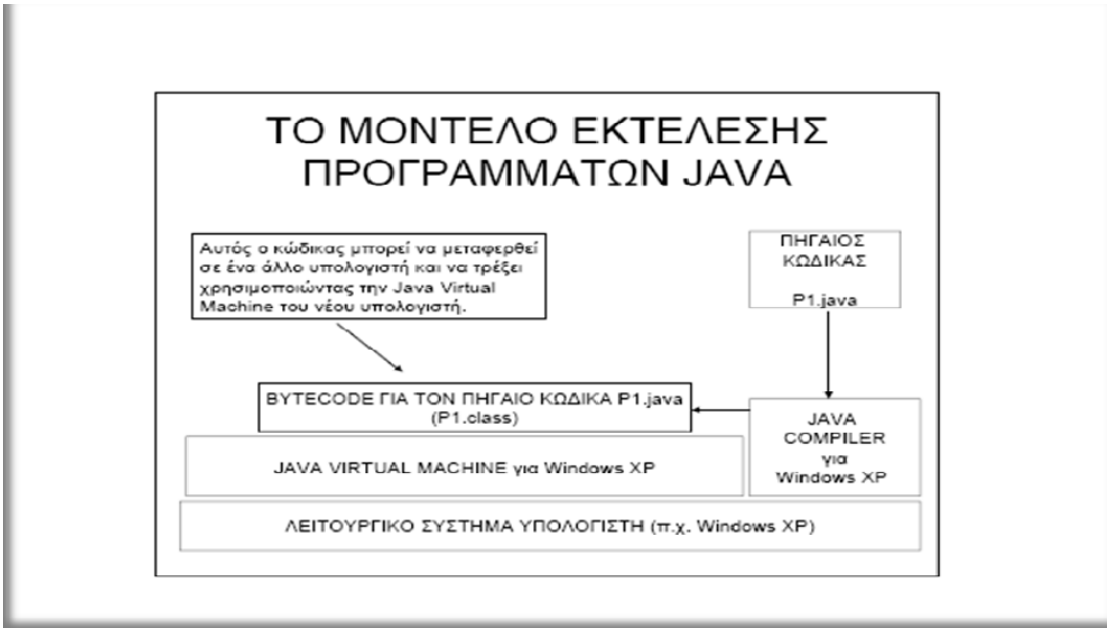
Η μεταγλώττιση του προγράμματος από γλώσσα κώδικα σε γλώσσα μηχανής ονομάζεται compiling. Στην java το compiling(μεταγλώττιση) χωρίζεται σε δύο μέρη. Στο πρώτο μέρος γίνεται έλεγχος συντακτικών λαθών και παράγεται ένας Byte code ο οποίος είναι κοινός ανεξαρτήτως σε ποιο μηχάνημα ή λειτουργικό σύστημα έγινε το compiling. Στο δεύτερο μέρος γίνεται η μετάφραση του πιο πάνω Byte code σε εκτελέσιμο κώδικα μηχανή. Αυτό εξαρτάται από το είδος της μηχανής και του λειτουργικού που υπάρχει. Για τον λόγο αυτό όταν γίνεται εγκατάσταση της java σε ένα υπολογιστή εγκαθίσταται μαζί με τον Compiler και μια Java Virtual Machine ή Εικονική Μηχανή Java (JVM). Αυτή παίρνει τον Byte code, βλέπει τι είδους μηχανή υπάρχει από κάτω και παράγει τον αντίστοιχο εκτελέσιμο κώδικα. Άρα μπορούμε να κάνουμε μεταγλώττιση σε ένα μηχάνημα π.χ. με λειτουργικό σύστημα Solaris και να τρέχουμε τον byte code σε ένα μηχάνημα με λειτουργικό σύστημα Windows. Τα χαρακτηριστικά της JVM είναι τα εξής: **[5]**

- Τα προγράμματα Java εκτελούνται στην JVM και όχι στον πραγματικό Η\Υ.
- Τα προγράμματα Java γράφονται για την JVM.
- Η JVM λειτουργεί σαν ένα firewall μεταξύ Η\Υ και προγραμμάτων.



Εικόνα 1.13. Μεταγλώττιση κώδικα java σε διαφορετικά λειτουργικά συστήματα [15]

Ένα χαρακτηριστικό μοντέλο εκτέλεσης προγραμμάτων απεικονίζεται στο Σχήμα 1.5.



Σχήμα 1.5. *Μοντέλο εκτέλεσης προγραμμάτων java* **[15]**

Κάποιες βασικές παρατηρήσεις για το μοντέλο εκτέλεσης της Java είναι οι εξής:

- Η Java Virtual Machine (JVM) εξαρτάται από το λειτουργικό σύστημα του Η\Υ στον οποίο έχει εγκατασταθεί.
- Ο συμβολομεταφραστής της Java (javac) εξαρτάται και αυτός από το λειτουργικό σύστημα του Η\Υ στον οποίο έχει εγκατασταθεί.
- Όλοι οι μεταγλωττιστές (ανεξάρτητα του λειτουργικού συστήματος) παράγουν τον ίδιο byte code για το ίδιο πηγαίο πρόγραμμα. Επίσης όλες οι JVM (ανεξάρτητα του λειτουργικού συστήματος στο οποίο τρέχουν) εκτελούν με τον ίδιο τρόπο ένα πρόγραμμα σε Java byte code.
- Όταν μεταφέρουμε έναν byte code(π.χ. από το διαδίκτυο) σε έναν υπολογιστή με διαφορετικό λειτουργικό σύστημα, που έχουν όμως μια JVM εγκατεστημένη, το πρόγραμμα θα εκτελεστεί επιτυχώς ανεξαρτήτως λειτουργικού συστήματος.
- Το τίμημα για το παραπάνω πλεονέκτημα που προσφέρει η Java είναι ότι απαιτείτε μια παραπάνω λειτουργία μετατροπής (από byte code σε γλώσσα μηχανής συστήματος) σε σχέση με γλώσσες όπως η C/C++, με αποτέλεσμα η ταχύτητα εκτέλεσης των προγραμμάτων να είναι μικρότερη.

1.6 ΤΙ ΕΙΝΑΙ Η JAVA 3D API;

Η βιβλιοθήκη Java 3D API **[1]** αποτελεί ένα περιβάλλον εργασίας (interface) για την δημιουργία προγραμμάτων σε java, με σκοπό την απεικόνισή τους και την αλληλεπίδραση τους με τριδιάστατα γραφικά. Αποτελεί την επέκταση της Java 2 JDK, βιβλιοθήκης που παρέχει τα κατάλληλα εργαλεία για διδιάστατα γραφικά. Το εν λόγω API (application programming interface) προσφέρει μια γκάμα από κατάλληλης δομής αρχικά προγράμματα **(constructors)** τα οποία διευκολύνουν την δημιουργία και τον χειρισμό 3D γεωμετριών, καθώς και δομές προγραμμάτων για την απεικόνισή των αντίστοιχων γεωμετριών. Η βιβλιοθήκη Java 3D προσφέρει όλες τις αναγκαίες συναρτήσεις για την κατασκευή εικόνων, οπτικών αναπαραστάσεων(visualization), τεχνικών που δημιουργούν την ψευδαίσθηση της κίνησης αντικειμένων(animation) και αλληλεπιδρώντα προγράμματα τρισδιάστατων γραφικών. Όλες οι γεωμετρίες που κατασκευάζονται τοποθετούνται για την απεικόνιση τους σε ένα τρισδιάστατο εικονικό χώρο που καλείται **virtual universe**. Το API έχει σχεδιαστεί έτσι ώστε να προσφέρει μεγάλη ευελιξία στο να δημιουργούμε τέτοιους χώρους με ακρίβεια, παρέχοντας ένα ευρύ πεδίο από μεγέθη, από μακροσκοπικά έως υποατομικά. Παρόλο τον αρκετά μεγάλο όγκο λεπτομερών πληροφοριών που προσφέρει το API είναι αρκετά εύχρηστο, καθώς οι λεπτομέρειες για τη σωστή απεικόνιση εκτελούνται αυτόματα από την ίδια την βιβλιοθήκη.

1.7 ΔΟΜΗ ΠΡΟΓΡΑΜΜΑΤΩΝ JAVA 3D.

Όπως αναφέρθηκε προηγουμένως η Java είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού. Κάθε πρόγραμμα αποτελεί ένα σύνολο αντικειμένων συγκεκριμένων ιδιοτήτων. Οι ιδιότητες των αντικειμένων καθορίζονται σε ξεχωριστές για κάθε αντικείμενο δομές προγραμμάτων, που καλούνται κλάσεις. Οι κλάσεις αποτελούν τη βασική δομή όλων των προγραμμάτων, που χρησιμοποιούν την γλώσσα Java και περιέχουν μεθόδους, συναρτήσεις και άλλα εργαλεία για τον πλήρη καθορισμό των ιδιοτήτων του αντικειμένου

το οποίο δημιουργούν. Όταν καθοριστούν οι ιδιότητες των αντικειμένων μέσα στις αντίστοιχες κλάσεις τα αντικείμενα τοποθετούνται σε μια δομή προγράμματος που καλείται **scene graph [1]**. Η δομή scene graph είναι το σημείο του προγράμματος όπου κατασκευάζεται ο τριδιάστατος εικονικός χώρος (virtual universe) και τα 'σκηνικά' που εμπεριέχει. Αποτελείται από μια ταξινόμηση αντικειμένων σε δενδρική δομή, η οποία καθορίζει σαφώς τον τρόπο απεικόνισης του χώρου αλλά και το περιεχόμενο του χώρου virtual universe, δηλαδή αποτελείται από αντικείμενα που προσδιορίζουν τη γεωμετρία, τον φωτισμό, τα ηχητικά εφέ, τη θέση και τον προσανατολισμό καθώς και την εμφάνιση όλων των αντικειμένων που τοποθετούνται στον virtual χώρο. Το API παρέχει πάνω από 100 προκαθορισμένες κλάσεις, οι οποίες παρουσιάζονται στο `javax.media.j3d` πακέτο της βιβλιοθήκης. Αυτές οι προκαθορισμένες κλάσεις καλούνται κλάσεις πυρήνα (core classes). Υπάρχουν αρκετά τέτοια πακέτα με προκαθορισμένες κλάσεις, τα σημαντικότερα των οποίων παρουσιάζονται στη συνέχεια: **[1]**

- `com.sun.j3d.utils`. **[1]** Είναι ένα πακέτο που περιέχει βοηθητικές κλάσεις (utility classes). Αποτελεί ουσιαστικά μια απαραίτητη προσθήκη στις κλάσεις πυρήνα, οι οποίες μας παρέχουν μόνο τις άκρως απαραίτητες κλάσεις για την κατασκευή του προγράμματος μας. Με τις βοηθητικές κλάσεις έχουμε μια σειρά από κατάλληλα εργαλεία, που κάνουν την κατασκευή του προγράμματος πιο εύκολη και πιο ουσιαστική. Τέτοιες βοηθητικές κλάσεις είναι οι κλάσεις γεωμετρίας (geometry classes), οι κλάσεις αποθήκευσης πληροφοριών (content loaders), καθώς και βοηθητικές κλάσεις για την κατασκευή του scene graph. Αποτέλεσμα της χρήσης των κλάσεων αυτών είναι και η αρκετά μεγάλη μείωση του όγκου του προγράμματος.

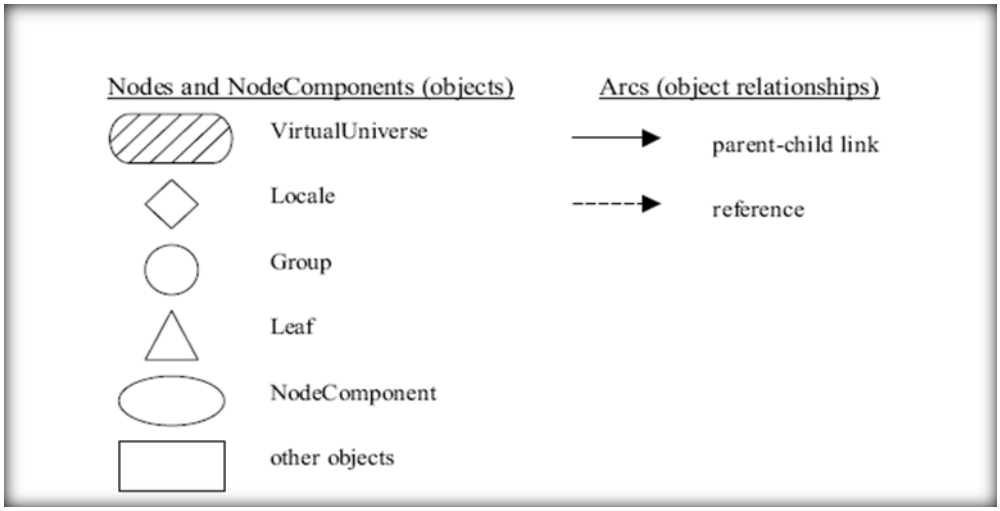
- `java.awt`. **[1]** Το πακέτο αυτό προσδιορίζει το AWT (abstract window toolkit), το οποίο παρέχει τα κατάλληλα εργαλεία για την κατασκευή παραθύρων για την απεικόνιση των 3D αντικειμένων. Για αυτό το πακέτο θα γίνει εκτενής ανάλυση στο κεφάλαιο 2, όπου θα αναπτυχθεί η μεθοδολογία κατασκευής του Viewer 3D καθώς αποτελεί βασικό κομμάτι της όλης εφαρμογής.

- `javax.vecmath`. **[1]** Μας παρέχει κλάσεις που περιέχουν μαθηματικές συναρτήσεις, πίνακες και άλλες μαθηματικές πληροφορίες.

Μία συνηθισμένη δομή scene graph είναι αυτή που απαρτίζεται από αντικείμενα που καλούμε κομβικά στοιχεία η κόμβους (**nodes**) **[1]**, τα χαρακτηριστικά των κόμβων **NodeComponents** καθώς και την σχέση μεταξύ των κόμβων, την οποία καλούμε **arcs**. Τα nodes είναι υποδείξεις σε συγκεκριμένες κλάσεις, που είτε έχουμε κατασκευάσει εμείς είτε προϋπάρχουν σε κάποιο από τα πακέτα της βιβλιοθήκης Java 3D. Οι μορφές των nodes που συναντούμε είναι 2, τα group nodes τα οποία έχουν μεταφορικό ρόλο και στα οποία είτε αποθηκεύονται άλλα group nodes είτε αποθηκεύονται τα leaf nodes και τα leaf nodes που καθορίζουν το είδος του αντικειμένου (π.χ. shape3d). Τα NodeComponents είναι αυτά που καθορίζουν την γεωμετρία και όλα τα χαρακτηριστικά τής εμφάνισης των αντικειμένων. Οι σχέσεις μεταξύ των nodes συναντώνται σε δύο μορφές. Η πρώτη και πιο συνηθισμένη είναι αυτή που καλείται σχέση **γονέα-παιδιού [β-1]**. Κάθε group node μπορεί να έχει όσο μεγάλο αριθμό παιδιών θέλουμε αλλά μόνο έναν πατέρα. Κάθε leaf node μπορεί να έχει μόνο έναν πατέρα αλλά κανένα παιδί. Το άλλο είδος σχέσης καλείται σχέση αναφοράς (reference relationship), η οποία συνδέει τα NodeComponets με τα leaf nodes μέσα στη δομή του scene graph, δηλαδή συνδέονται τα χαρακτηριστικά που προσδιορίστηκαν, δημιουργώντας διαφορετικά NodeComponets με το αντικείμενο που δημιουργούν τα leaf nodes, προσδίδοντας έτσι στο αντικείμενο τα συγκεκριμένα χαρακτηριστικά.

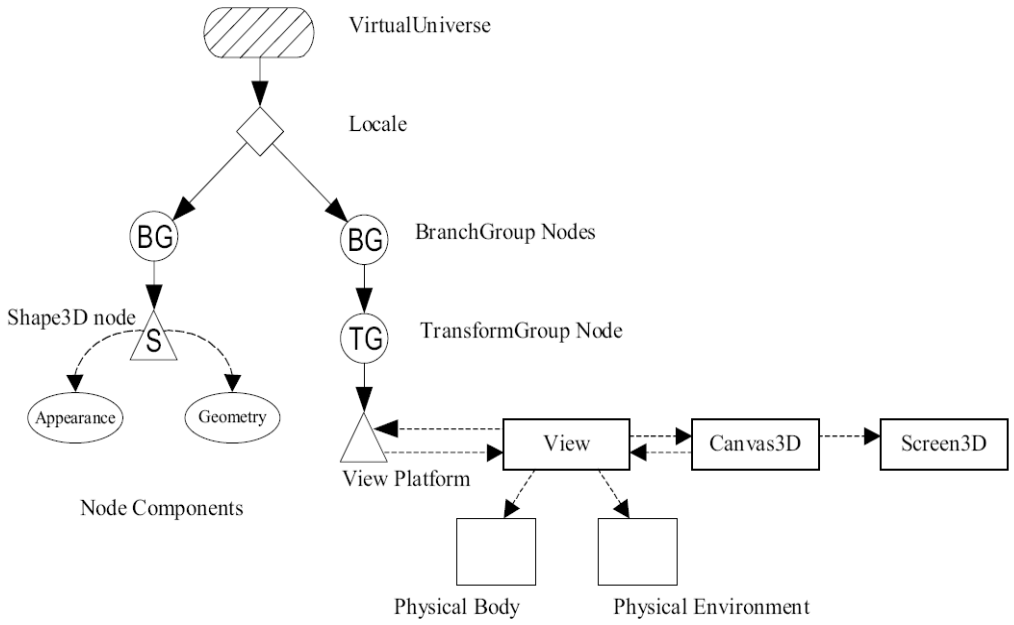
Όπως αναφέρθηκε παραπάνω η δομή του scene graph είναι δενδρικής μορφής. Σε μία δενδρική δομή υπάρχει μόνο μία ρίζα(αρχή) την οποία καταλαμβάνει μόνο ένα node το

οποίο καλείται αρχικό node(root node). Τα υπόλοιπα nodes είναι προσπελάσιμα ξεκινώντας από το αρχικό node και ακολουθώντας τη σχέση μεταξύ των nodes, που έχει καθοριστεί στο πρόγραμμα. Συμβολικά θα μπορούσαμε να το φανταστούμε σαν τη ρίζα ενός δέντρου (root node) που, μέσω των κλαδιών του (arcs), επικοινωνεί με τα φύλλα του (leaf nodes), που διέπονται από συγκεκριμένα χαρακτηριστικά (NodeComponets). Κάθε 'μονοπάτι' προς ένα φύλλο είναι μοναδικό, δηλαδή υπάρχει μόνο μια διαδρομή για να καταλήξουμε σε ένα leaf node, το οποίο μονοπάτι καλούμε scene graph path. Ακολουθώντας ένα τέτοιο μονοπάτι καθορίζουμε λεπτομερώς όλες τις αναφορικές πληροφορίες του leaf node στο οποίο καταλήγει, όπως θέση, κατεύθυνση και μέγεθος του αντικειμένου. Στο σχήμα 1.6 παρουσιάζεται η σχηματική αντιστοιχία που συνηθίζεται για τα nodes και τα arcs τον εικονικό χώρο και όλα τα υπόλοιπα αντικείμενα, ενώ στο σχήμα 1.7 παρουσιάζεται η δομή ενός προγράμματος scene graph.



Σχήμα 1.6. *Αντιστοίχιση σχήματος-στοιχείων του scene graph* **[1]**

Κάθε σύμβολο που παρουσιάζεται αριστερά του σχήματος 1.6 αντιστοιχεί σε ένα και μόνο node που φαίνεται στα δεξιά, όταν χρησιμοποιείται σε διαγράμματα όπως αυτό του σχήματος 1.7. Το βέλος με την συμπαγή γραμμή και μύτη συμβολίζει την σχέση γονέως-παιδιού ενώ το βέλος με την διακεκομμένη γραμμή συμβολίζει την σχέση αναφοράς.



Σχήμα 1.7. *Δομή scene graph* **[1]**

Όπως παρατηρούμε στο σχήμα 1.7, κάθε πρόγραμμα που κατασκευάζουμε μπορεί να έχει μόνον έναν τριδιάστατο εικονικό χώρο(virtual universe), στον οποίο μπορούν να τοποθετηθούν πολλές διαφορετικές σκηνές απεικόνισης αντικειμένων (locales). Συνηθίζεται όμως να χρησιμοποιείτε μόνο μία σκηνή. Μπορούμε να φανταστούμε αυτή τη σκηνή σαν ένα οροθέσιο που μας προσδιορίζει που θα τοποθετούνται τα αντικείμενα που θέλουμε να απεικονίσουμε στο virtual universe. Η σκηνή αυτή μπορεί να αποτελέσει τον γονέα για πολλά αρχικά nodes (root nodes). Το αρχικό node(root node) είναι πάντα ένα group node το οποίο καλείται **BranchGroup node**.Το περιεχόμενο του προγράμματος που ακολουθεί το BranchGroup node λέγεται **content branch graph [1]** και είναι το σημείο του προγράμματος που καθορίζεται οτιδήποτε θα απεικονίζεται κατά την εκτέλεση του προγράμματος στο virtual universe δηλαδή η γεωμετρία των αντικειμένων, η εμφάνιση, η συμπεριφορά, η θέση, ο φωτισμός και τα ηχητικά εφέ. Αυτό αποτελεί το πρώτο βασικό σκέλος της κατασκευής ενός scene graph προγράμματος. Το δεύτερο βασικό σκέλος αποτελεί η κατασκευή του **view branch graph [1],** στο οποίο προσδιορίζεται ο τρόπος με τον οποίο θα απεικονίζονται τα στοιχεία του virtual universe. Κατασκευάζοντας τα παραπάνω 2 κομμάτια κώδικα έχουμε σχεδόν ολοκληρώσει το πρόγραμμα μας.

Αναλύοντας περαιτέρω την έννοια της δομής scene graph συμπεραίνουμε κάποιες χρήσιμες πληροφορίες. Όπως αναφέρθηκε παραπάνω η δομή scene graph αποτελείται από nodes, που διαχωρίζονται σε group και leaf nodes και από τα NodeComponets. Όμως τα nodes αναφέραμε ότι υποδεικνύουν συγκεκριμένες κλάσεις, στις οποίες περιέχονται όλες οι πληροφορίες για το αντικείμενο το οποίο αντιπροσωπεύουν. Συνεπώς όταν αναφερόμαστε σε κάποιο node είτε group node είτε leaf node ουσιαστικά αναφερόμαστε στις αντίστοιχες κλάσεις. Περιγράφουμε λοιπόν σε αυτό το σημείο την έννοια της κάθε κλάσης.

Node Class (κλάση node) [1]

Ή node κλάση αποτελεί μια περιληπτική υπερκλάση των υποκλάσεων group και leaf. Υπερκλάση αποκαλείται μία κλάση που περιέχει σημαντικές μεθόδους που κληρονομούν όλες οι υποκλάσεις. Μπορούμε να το φανταστούμε σαν ένα γονίδιο που περιέχει συγκεκριμένες πληροφορίες και που προέρχεται από το γενετικό υλικό του γονέως και το

κληρονομούν τα παιδιά. Οι υποκλάσεις της node κλάσης (group και leaf) είναι αυτές που συνθέτουν την δομή scene graph.

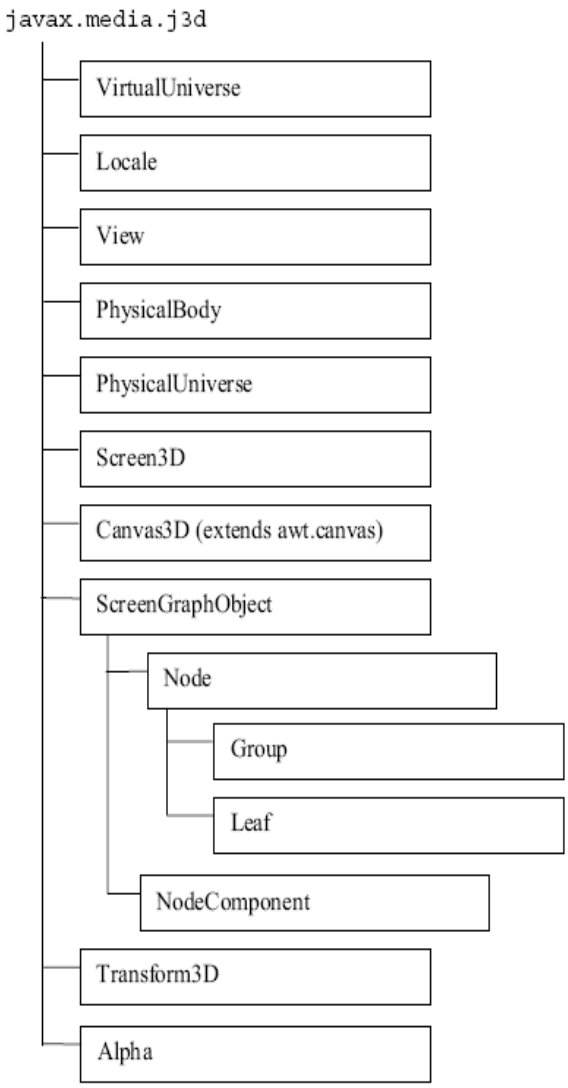
Group Class (κλάση group) [1]

Η κλάση group αποτελεί με την σειρά της υπερκλάση για τον προσδιορισμό της θέσης και του προσανατολισμού των αντικειμένων στο virtual universe. Υποκλάσεις της αποτελούν οι BranchGroup και TransformGroup κλάσεις.

Leaf Class (κλάση leaf) [1]

Η κλάση leaf αποτελεί υπερκλάση για τον προσδιορισμό του σχήματος, της συμπεριφοράς, του φωτισμού και των ηχητικών εφέ. Υποκλάσεις της αποτελούν οι Shape3d, Light, Sound, Behavior κλάσεις. Κάθε αντικείμενο που δημιουργείται σε αυτές τις κλάσεις δεν επιτρέπεται να έχει παιδιά (βλ. σχέση γονέως-παιδιού) αλλά μπορεί να παραπέμπουν σε NodeComponets.

NodeComponent Class (κλάση NodeComponent) [1] Η κλάση NodeComponent αποτελεί υπερκλάση για τον προσδιορισμό της γεωμετρίας, της εμφάνισης, της υφής και των ιδιοτήτων του υλικού ενός Shape3d node της κλάσης leaf. Οι κλάσεις NodeComponent δεν αποτελούν μέρος της δομής scene graph αλλά κάποιο σημείο του προγράμματος μέσα στην scene graph μπορεί να παραπέμπει σε αυτές. Μια κλάση NodeComponent μπορούν να την χρησιμοποιήσουν παραπάνω από ένα Shape3d nodes. Στο σχήμα 1.8 παρατηρούμε μια περιληπτική σύνοψη της ιεραρχίας των κλάσεων της Java3D API.



Σχήμα 1.8. *Ιεραρχία κλάσεων της Java 3D API* **[1]**

1.8. ΔΗΜΙΟΥΡΓΙΑ ΕΝΟΣ ΠΡΟΓΡΑΜΜΑΤΟΣ JAVA 3D.

Ακολουθώντας μια σειρά από συγκεκριμένα βήματα μπορούμε να κατασκευάσουμε ένα πρόγραμμα Java 3D:

- Δημιουργία ενός αντικειμένου **Canvas3D [1]**. Το αντικείμενο αυτό αποτελεί την επίπεδη επιφάνεια στην οποία θα προβάλλονται τα αντικείμενα που έχουν τοποθετηθεί στο virtual universe ώστε να γίνονται αντιληπτά από το ανθρώπινο μάτι. Αυτή η επιφάνεια ονομάζεται και **image plate [1].**
- Δημιουργία ενός VirtualUniverse αντικειμένου. Είναι ουσιαστικά το αντικείμενο που αντιπροσωπεύει τον εικονικό χώρο που τοποθετούνται όλα τα αντικείμενα που έχουμε κατασκευάσει.
- Δημιουργία ενός Locale αντικειμένου και σύνδεση του με το VirtualUniverse αντικείμενο.

- Κατασκευή της δομής View branch graph, δομή που αποτελεί κομμάτι της συνολικής δομής scene graph. Για την κατασκευή της απαιτούνται 5 βήματα**[1]**

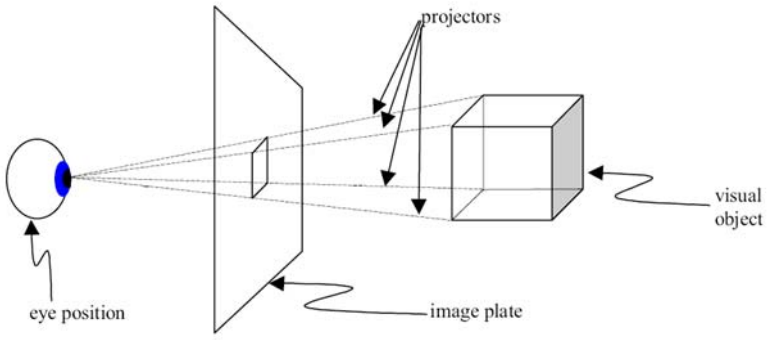
  - Δημιουργία ενός View αντικειμένου.
  - Δημιουργία ενός View Platform αντικειμένου.
  - Δημιουργία ενός Physical Body αντικειμένου.

- Δημιουργία ενός Physical Environment αντικειμένου.
- Προσθήκη των αντικειμένων View Platform, Physical Body, Physical Environment, Canvas3D στο αντικείμενο View.

- Κατασκευή της δομής Content branch graph, δομή που αποτελεί κομμάτι της συνολικής δομής scene graph.
- Μεταγλώττιση (compile) της δομής branch graph (content & view).
- Προσθήκη του branch graph στο αντικείμενο Locale.

Τα παραπάνω βήματα αποτελούν μία γενική καθοδήγηση στη δημιουργία μιας εφαρμογής Java 3D. Δεν λαμβάνονται υπόψη αρκετές λεπτομέρειες καθώς αποτελούν ένα πεδίο που δεν απασχολεί το θέμα αυτής της διπλωματικής εργασίας. Στην εικόνα 1.6 παρουσιάζεται η θεμελιώδης σχέση μεταξύ του image plate του virtual universe και της θέσης του ανθρώπινου ματιού.

Είναι προκαθορισμένη στην βιβλιοθήκη Java 3D API η θέση που καταλαμβάνει η επιφάνεια image plate καθώς και η θέση και κατεύθυνση του ορθοκανονικού συστήματος αξόνων (x,y,z) που χρησιμοποιεί. Συγκεκριμένα η image plate τοποθετείται στο κέντρο του εικονικού χώρου. Η διεύθυνση του z άξονα είναι αυτή που ενώνει το μάτι της εικόνας 1.6 με το αντικείμενο που προβάλλεται στην image plate με θετική κατεύθυνση προς το αντικείμενο. Η διεύθυνση του x άξονα είναι μια οριζόντια γραμμή πάνω στην image plate με θετική κατεύθυνση προς τα δεξιά. Η διεύθυνση του y άξονα είναι μια κάθετη γραμμή πάνω στην image plate με θετική κατεύθυνση προς τα επάνω. Το σημείο (0,0,0) είναι το κέντρο της image plate.



Εικόνα 1.14. Θεμελιώδης σχέση μεταξύ επιφάνειας image plate, του τρισδιάστατου εικονικού χώρου virtual universe και του ανθρώπινου ματιού [**1**]

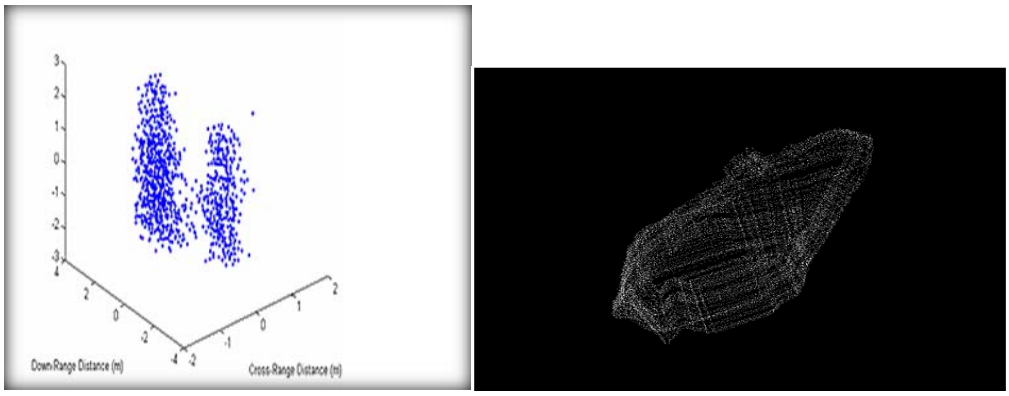
Χρησιμοποιώντας όλες τις παραπάνω γνώσεις κατασκευάστηκε ο τρισδιάστατος Viewer που παρουσιάζεται σε αυτή την διπλωματική εργασία. Τα βήματα που ακολουθηθήκαν για την κατασκευή του είναι το κομμάτι που θα απασχολήσει το 2^{ο} κεφάλαιο της διπλωματικής εργασίας.

2

ΑΝΑΛΥΣΗ ΤΗΣ ΜΕΘΟΔΟΛΟΓΙΑΣ ΚΑΤΑΣΚΕΥΗΣ ΤΟΥ ΤΡΙΔΙΑΣΤΑΤΟΥ T4T VIEWER.

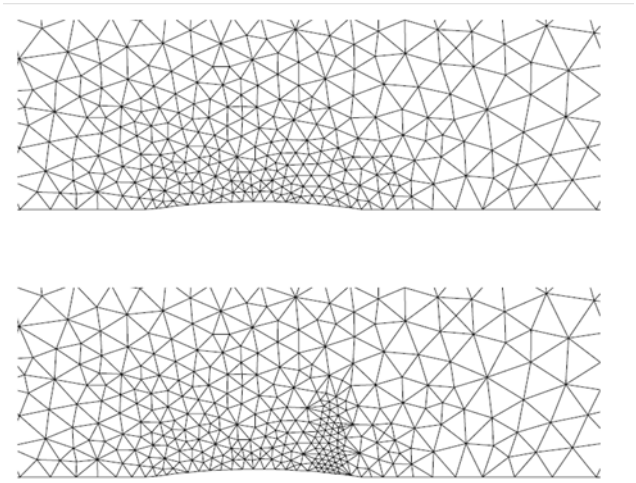
2.1. ΕΙΣΑΓΩΓΗ

Ο λόγος κατασκευής του Viewer ήταν η ανάγκη που δημιουργήθηκε να μπορούν να απεικονίζονται σε τριδιάστατη μορφή πτερυγώσεις στροβιλομηχανών που έχουν σχεδιαστεί παραμετρικά στο πρόγραμμα T4T, ένα πρόγραμμα που αναπτύχθηκε κατά τη διάρκεια προηγούμενης μεταπτυχιακής εργασίας. Η γενική ιδέα για την κατασκευή του T4T Viewer 3D ήταν να αναπτυχθεί ένα λογισμικό σε java, το οποίο να είναι ικανό να διαχειρίζεται το **νέφος των σημείων** που ορίζουν κάθε μια ξεχωριστή επιφάνεια τις αντίστοιχης πτερύγωσης αλλά και τον τρόπο που ενώνονται αυτά τα σημεία κατασκευάζοντας το αντίστοιχο **πλέγμα** της επιφάνειας της πτερύγωσης. Το πλέγμα που επιλέχθηκε αποτελείται από τριγωνικά στοιχεία. Στην εικόνα 2.1 παρουσιάζονται 2 νέφη σημείων. Στην αριστερή πλευρά ένα νέφος σημείων μέσα σε σύστημα αξόνων και στην δεξιά ένα νέφος σημείων μέσα στον εικονικό χώρο.



Εικόνα 2.1. Απεικόνιση νέφους σημείων σε σύστημα αξόνων(αριστερά) και στο χώρο(δεξιά) [3]

Στην εικόνα 2.2 παρουσιάζονται 2 τριγωνικά πλέγματα.



Εικόνα 2.2. Απεικόνιση τριγωνικών πλεγμάτων [3]

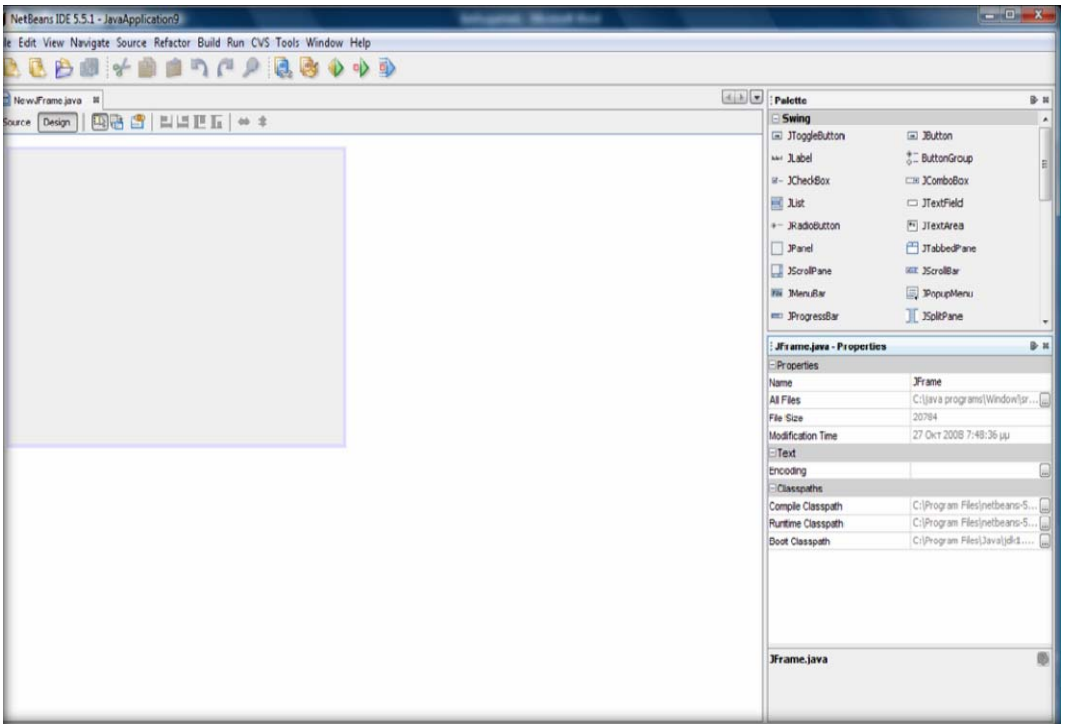
Επιπλέον απαιτήσεις για την κατασκευή του τρισδιάστατου Viewer ήταν να υπάρχει δυνατότητα κάποιων λειτουργικών δυνατοτήτων. Τέτοιες είναι η ικανότητα περιστροφής του αντικειμένου που απεικονίζεται, με συγκεκριμένη κίνηση του ποντικιού του Η\Υ, η μετακίνηση του αντικειμένου με παρόμοιο τρόπο, η μεγέθυνση του αντικειμένου και η σμίκρυνσή του, καθώς και μία σειρά από άλλες πρακτικές δυνατότητες, οι οποίες θα αναλυθούν στη συνέχεια. Επίσης είναι σημαντικό να γίνεται αντιληπτός και κατανοητός από το χρήστη ο προσανατολισμός κάθε αντικειμένου, που παρουσιάζεται στον viewer, οπότε είναι αναγκαία και η εισαγωγή ενός ορθοκανονικού συστήματος συντεταγμένων (x,y,z). Η μεθοδολογία που ακολουθήθηκε για την κατασκευή του λογισμικού ακολουθεί στην επόμενη ενότητα. Προγραμματιστικές λεπτομέρειες και θέματα που αφορούν τον κώδικα που αναπτύχθηκε θα παρουσιαστούν στο κεφάλαιο 3.

2.2. ΜΕΘΟΔΟΛΟΓΙΑ ΚΑΤΑΣΚΕΥΗΣ ΤΟΥ T4T VIEWER 3D

2.2.1 JFrame

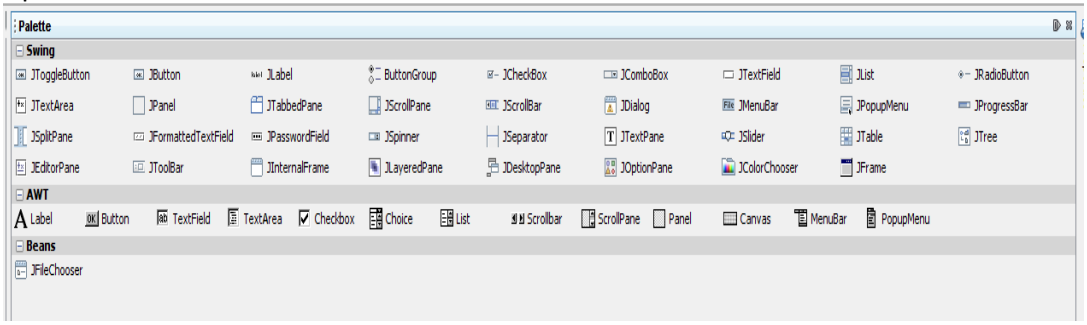
Όπως αναφέρθηκε προηγουμένως το λογισμικό που ανπτύχθηκε αποτελεί ουσιαστικά ένα παράθυρο στο οποίο θα απεικονίζονται τριδιάστατα αντικείμενα. Το πρώτο βήμα για την εργασία ήταν η δημιουργία αυτού του παραθύρου, πάνω στο οποίο θα προστεθούν και όλα τα επιπλέον εργαλεία που θα ικανοποιούν όλες τις απαιτήσεις για την λειτουργία του viewer. Το πρόγραμμα NetBeans και οι προκαθορισμένες κλάσεις του μας παρέχουν ένα έτοιμο πλαίσιο παραθύρου, πάνω στο οποίο μπορούμε να αναπτύξουμε τις δικές μας εφαρμογές, το οποίο ονομάζεται JFrame. Στο περιβάλλον εργασίας κατά τη δημιουργία και χρήση ενός JFrame μας δίνεται η δυνατότητα να επέμβουμε στο προγραμματιστικό κομμάτι της δημιουργίας του παραθύρου(**Source**) ώστε να εισάγουμε, μέσω κατάλληλου κώδικα, δικές μας προσθήκες ή αλλαγές (π.χ. την αλλαγή του μεγέθους, ή την ικανότητα να συνεργάζεται με άλλα παράθυρα κ.α.). Επιπλέον μας δίνεται η δυνατότητα να εφαρμόσουμε αλλαγές κατευθείαν πάνω στο πλαίσιο του παραθύρου (**Design**), όπως την εισαγωγή εξαρτημάτων (π.χ. κουμπιών), που στην συνέχεια θα τα προγραμματίσουμε να

έχουν συγκεκριμένη λειτουργία, ή την αλλαγή των ιδιοτήτων του. Στην εικόνα 2.3 παρουσιάζεται το περιβάλλον εργασίας κατά την χρήση ενός JFrame. Στο μέσον της εικόνας φαίνεται το κενό πλαίσιο ενός παραθύρου στην μορφή **Design,** που αποτελεί και τη βάση για την κατασκευή όλων των υπολοίπων εφαρμογών.



Εικόνα 2.3. Περιβάλλον εργασίας Design του JFrame

Στο περιβάλλον εργασίας Design του JFrame της εικόνας 2.3 στα δεξιά της εικόνας παρουσιάζεται μια καρτέλα 'εργαλειοθήκη' που ονομάζεται **palette.** Σε αυτή υπάρχουν όλα τα εξαρτήματα που μας παρέχονται από το πρόγραμμα NetBeans για τη διευκόλυνση μας στην κατασκευή εφαρμογών. Τα εξαρτήματα αυτά ονομάζονται γενικότερα **components** και χωρίζονται σε 2 κατηγορίες πάνω στην palette ανάλογα με το πακέτο της java που χρησιμοποιούν.



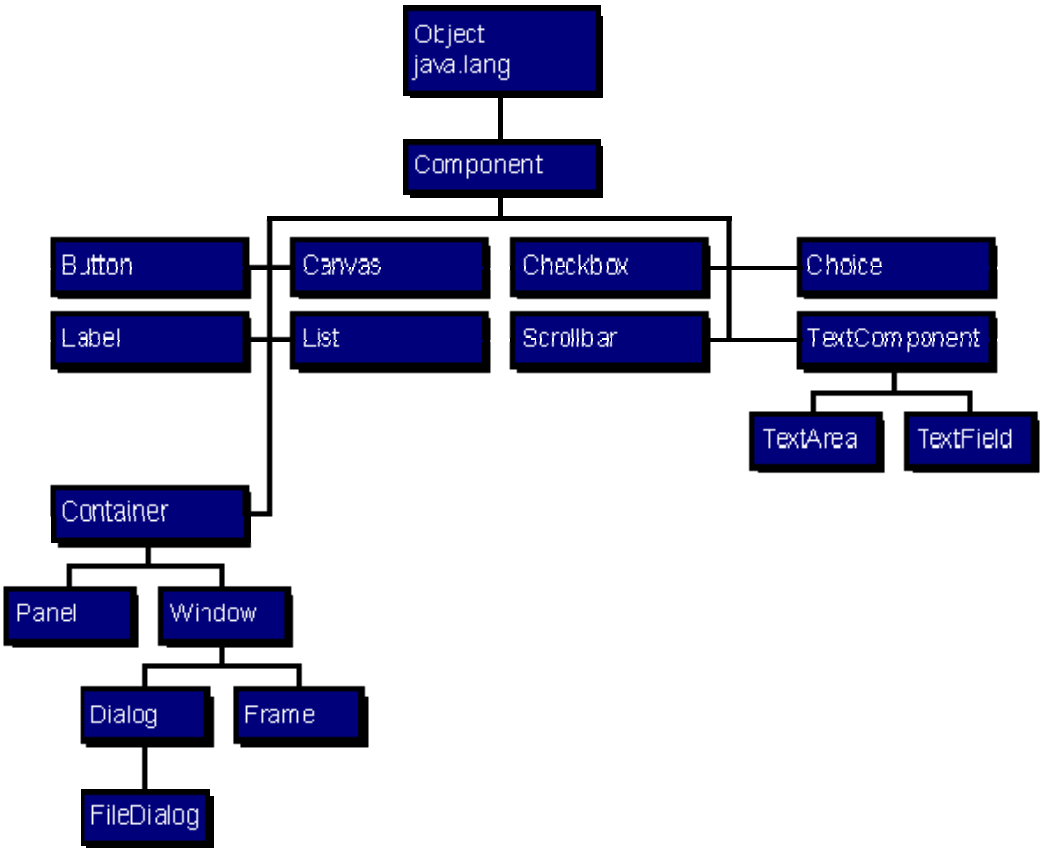
Εικόνα 2.4. Pallete του JFrame

- Η πρώτη κατηγορία είναι αυτή που χρησιμοποιεί το πακέτο java.awt. Όπως έχουμε αναφέρει στο κεφάλαιο 1 το πακέτο awt είναι ένα πακέτο που μας παρέχει όλα τα εργαλεία για την κατασκευή των παραθύρων όπου θα απεικονίζονται τα 3D γραφικά. Το πακέτο java.awt περιέχει:

  – ***Components***

- ***Containers***
- ***LayoutManagers***

Στο σχήμα 2.1 παρατηρούμε την ιεραρχία των κλάσεων του πακέτου java.awt, που μας επιτρέπει να κατανοήσουμε τη σχέση μεταξύ των αρχικών κλάσεων και των υποκλάσεων τους, καθώς και τον τρόπο που κληρονομούνται από τις υποκλάσεις οι ιδιότητες των αρχικών τους κλάσεων. Η ιεραρχία ξεκινάει από το πάνω μέρος του σχήματος και προχωράει προς τα κάτω.



Σχήμα 2.1. *Ιεραρχία κλάσεων της Java.AWT* [**1**]

Όλα τα AWT **Components [1]** εμπεριέχουν την κλάση java.awt, ή όπως συνηθίζεται να καλούμε στον προγραμματισμό με java, κάνουν **extend** την κλάση java.awt. Αυτό πρακτικά σημαίνει ότι κληρονομούν τις ιδιότητες που τους παρέχει αυτή η αρχική αυτή κλάση. Οι βασικές ιδιότητες των components είναι:

- ***Color (χρώμα)***
- ***Location (τοποθεσία)***
- ***Size (μέγεθος)***
- ***Font (φόντο)***
- ***Visibility (ορατότητα αντικειμένου)***

Εκτός από τις ιδιότητες κληρονομούν και αρκετές μεθόδους για τον χειρισμό των ιδιοτήτων αυτών, ώστε να υπάρχει η δυνατότητα να επέμβουμε και να μεταβάλουμε όποιες επιθυμούμε. Κάποια βασικά AWT components είναι τα εξής:

- ***Button***
- ***Checkbox***

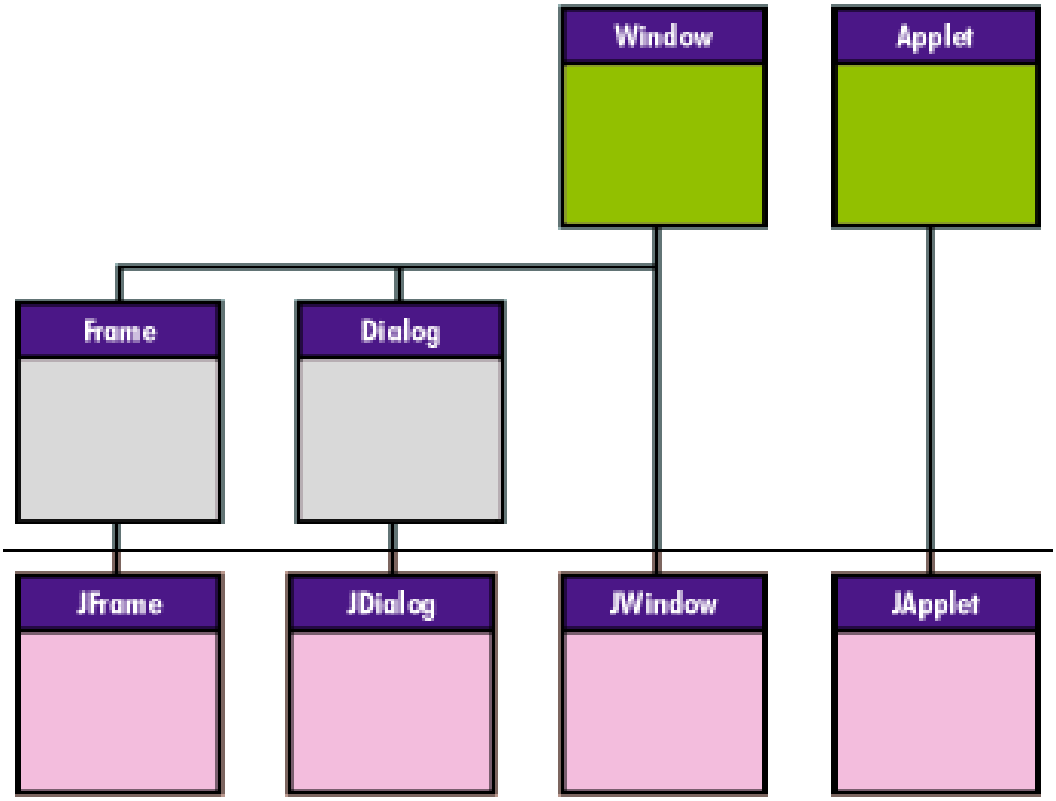
- ***Label***
- ***List***
- ***Container***
- ***MenuBar***
- ***MenuItem***
- ***Scrollbar***
- ***TextArea***
- ***TextField***
- ***Canvas***

Τοποθετώντας ένα εξάρτημα στο παράθυρο, όπως για παράδειγμα ένα Button, ενημερώνεται αυτόματα ο κώδικας κατασκευής του παραθύρου, δηλώνοντας το εξάρτημα που προσθέσαμε και δίνοντας του τις αρχικές από την κλάση java.awt ιδιότητες του. Στη συνέχεια μπορούμε να επέμβουμε και να αλλάξουμε η να προσθέσουμε όποια ιδιότητα επιθυμούμε, χρησιμοποιώντας την κατάλληλη κάθε φορά μέθοδο.

Τα εξαρτήματα **Containers [1]** αποτελούν ουσιαστικά μια υποκατηγορία των components, καθώς έχουν την ιδιότητα να μπορούν να εμπεριέχουν άλλα components. Τα βασικότερα είναι τα ακόλουθα:

- ***Window***
- ***Dialog***
- ***Frame***
- ***Applet***

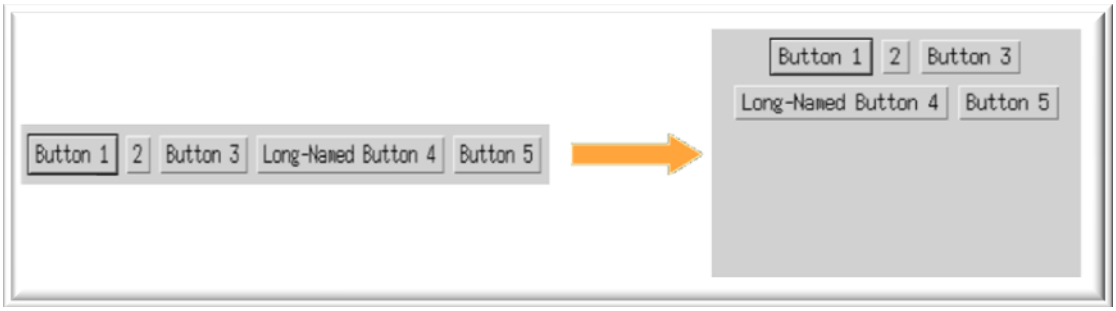
Το JFrame, με το οποίο ξεκινήσαμε την κατασκευή του παραθύρου, αποτελεί ουσιαστικά ένα container, που είναι υποκλάση του AWT container Frame, δηλαδή κληρονομεί όλες τις ιδιότητες και τις μεθόδους του αντικειμένου Frame. Πάνω στο JFrame θα τοποθετηθούν components, όπως Buttons CheckBoxes κ.α., με τα οποία θα συνεχίσουμε την κατασκευή του παραθύρου. Στο σχήμα 2.2 παρατηρούμε την ιεραρχία των κλάσεων, που ανήκουν στο πακέτο java.awt.containers, δίνοντάς μας μια σχηματική απεικόνιση της σχέσης κλάσεων-υποκλάσεων και τον τρόπο που κληρονομούνται οι ιδιότητες μιας υποκλάσης από την αρχική της κλάση. Η ιεραρχία ξεκινάει από το πάνω μέρος του σχήματος και συνεχίζει προς τα κάτω.



Σχήμα 2.2. *Ιεραρχία κλάσεων της Java.AWT.Containers* **[1]**

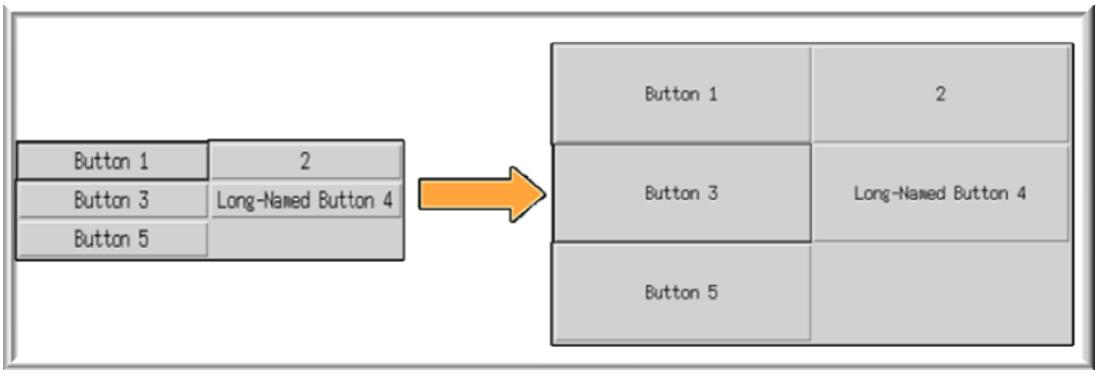
Η τελευταία κατηγορία του πακέτου java.awt είναι μια κατηγορία αντικειμένων που καλούμε **Layout Managers [5]** . Οι Layout Managers καθορίζουν τη θέση-διάταξη των components πάνω στο παράθυρο, το μέγεθος τους αλλά και την συμπεριφορά που θα έχει κάθε component, σε περίπτωση που ο χρήστης κάνει μία λειτουργία που ονομάζουμε resize και αφορά στην αυξομείωση των διαστάσεων του παραθύρου. Οι Layout Managers είναι οι εξής:

- **FlowLayout Manager: [5]** Τα components τοποθετούνται ανά γραμμή από αριστερά προς τα δεξιά. Στην εικόνα 2.5 παρουσιάζεται ένα παράδειγμα στο οποίο χρησιμοποιείτε Flowlayout Manager.



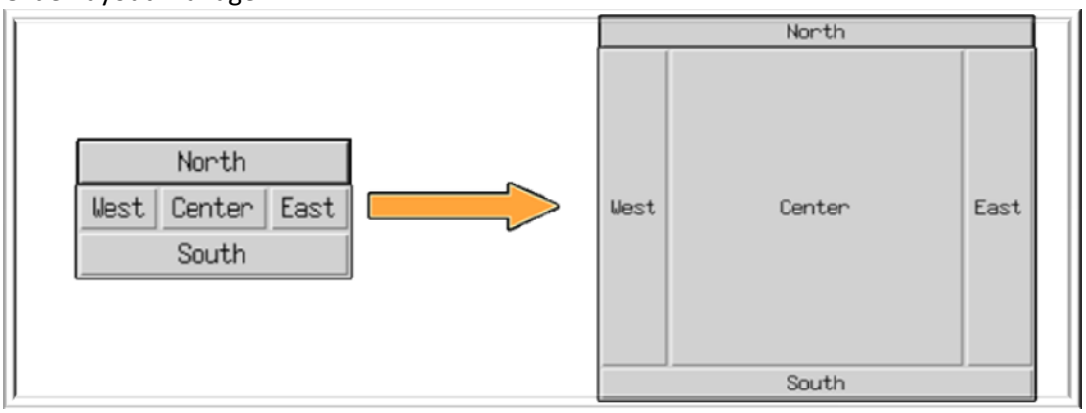
Εικόνα 2.5. FlowLayout Manager [5]

- **GridLayout Manager: [5]** Τα components τοποθετούνται σε διάταξη πίνακα μεγέθους (m,n) δηλαδή m γραμμών **x** n στηλών, μέγεθος που καθορίζεται από εμάς. Στην εικόνα 2.6 παρουσιάζεται ένα παράδειγμα GridLayout Manager.



Εικόνα 2.6. GridLayout Manager [5]

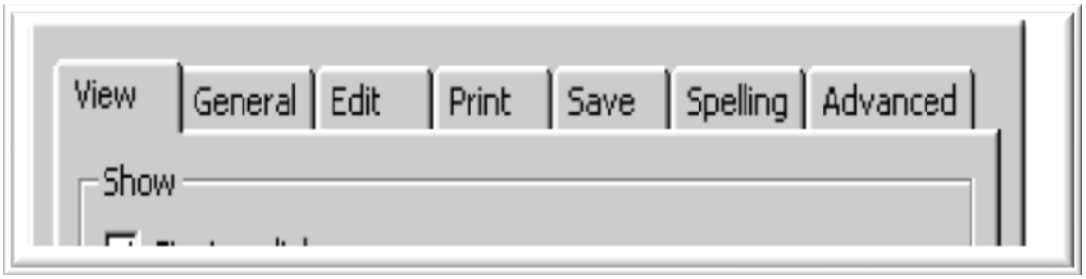
- **BorderLayout Manager: [5]** Η επιφάνεια του παραθύρου χωρίζεται σε πέντε περιοχές, τις Center, North, South, East και West. Τα components τοποθετούνται με βάση την περιοχή που θα καθορίσουμε να ανήκουν. Στην εικόνα 2.7 παρουσιάζεται ένα παράδειγμα BOrderLayout Manager.



Εικόνα 2.7. BorderLayout Manager [5]

- **CardLayout Manager: [5]**
  - Τοποθετεί τα components το ένα πάνω στο άλλο σε μορφή καρτελών.
  - Μόνο η πάνω καρτέλα είναι ορατή.
  - Κάθε καρτέλα προσδιορίζεται από αλφαριθμητική μεταβλητή (String).
  - Οι καρτέλες είναι συνήθως Panels (σ.σ. Panel είναι ένα AWT component)

Στην εικόνα 2.8 παρουσιάζεται ένα παράδειγμα CardLayout Manager.



Εικόνα 2.8. CardLayout Manager [5]

Η δεύτερη κατηγορία των εξαρτημάτων χρησιμοποίει μία εξέλιξη του AWT , ένα πακέτο που ονομάζεται AWT V2 ή Swing. Οι βασικές διαφορές από το πακέτο AWT είναι:

– Το πακέτο είναι 100% γραμμένο σε γλώσσα προγραμματισμού java.
– Απαιτεί JDK 1.1.2 ή υψηλότερο
– Πλουσιότερο σύνολο από Components (Αντικαθιστά components του AWT. Νέα, πιο σύνθετα components).

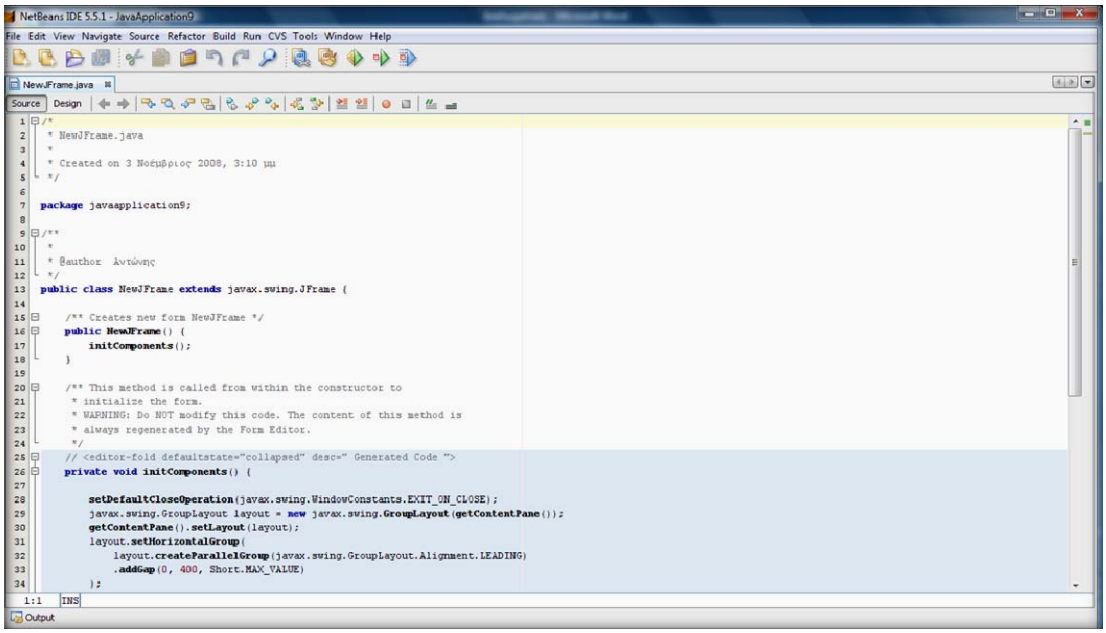
- Οι κλάσεις του Swing **[15]** χρησιμοποιούνται για τη δημιουργία GUIs (Graphical User Interfaces). Οι περισσότερες κλάσεις έχουν ίδια ονόματα με αυτές του AWT με την προσθήκη του προθέματος «J» - π.χ. JButton αντί του Button.
- 3 σημαντικές βελτιώσεις σε σχέση με το AWT

– δεν βασίζεται σε native components της πλατφόρμας.
– υποστηρίζει “Pluggable Look-and-Feel” ή PLAF.
– Βασίζεται στο Model-View-Controller (MVC).

Κάποια από τα βασικά Swing Components παρουσιάζονται στη συνέχεια:

– ***JComponent (όλα τα γνωστά από το AWT πακέτο components όπως Buttons, CheckBox κ.α.)***
– ***JComboBox***
– ***JLabel***
– ***JList***
– ***JMenuBar***
– ***JPanel***
– ***JPopupMenu***
– ***JScrollBar***
– ***JScrollPane***
– ***JFrame***

Το JFrame αποτελεί ένα Swing Component, ένα παράθυρο δηλαδή με όλα τα «συνήθη» χαρακτηριστικά: border, title, buttons για κλείσιμο, ελαχιστοποίηση κ.λπ. Στην εικόνα 2.9 παρουσιάζεται ενδεικτικά ένα κομμάτι του κώδικα κατασκευής του JFrame, ο οποίος παρέχεται έτοιμος από το NetBeans. Δίνεται η δυνατότητα στον χρήστη να επέμβει σε όποιο κομμάτι του κώδικα επιθυμεί για να πετύχει της αλλαγές που θέλει.



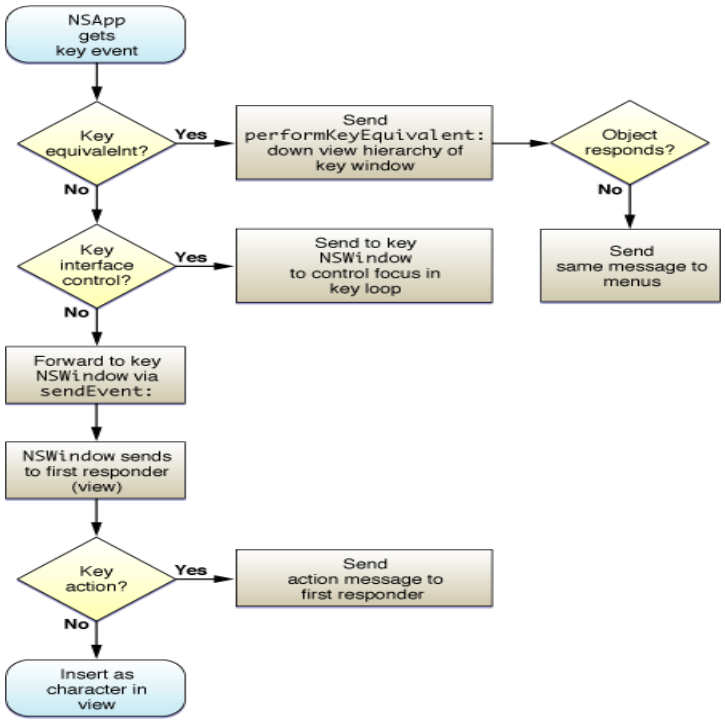
Εικόνα 2.9. Πηγαίος κώδικας δημιουργίας JFrame

Μια ακόμα σημαντική καρτέλα πάνω στο περιβάλλον εργασίας του JFrame είναι η καρτέλα ιδιοτήτων των components, που ονομάζεται **properties**. Χρησιμοποιώντας αυτή την καρτέλα, μπορούμε να διαχειριστούμε όλες τις ιδιότητες των εξαρτημάτων που έχουμε τοποθετήσει στο JFrame. Η γκάμα των ιδιοτήτων διαφέρει από εξάρτημα σε εξάρτημα αλλά αναφέρουμε μερικές για να γίνει κατανοητό τι είδους ιδιότητες μπορούν να ρυθμιστούν. Έτσι αναφορικά κάποιες από τις ιδιότητες είναι:

- ***Φόντο εξαρτήματος(RGB COLOR)***
- ***Χρωματισμός της επιφάνειας του***
- ***Θέση στο JFrame σε x,y συντεταγμένες***
- ***Μέγεθος***
- ***Ομάδα(group)***
- ***Γονέας συμφωνα με την σχέση κληρονομικότητας***
- ***Όνομα***
-

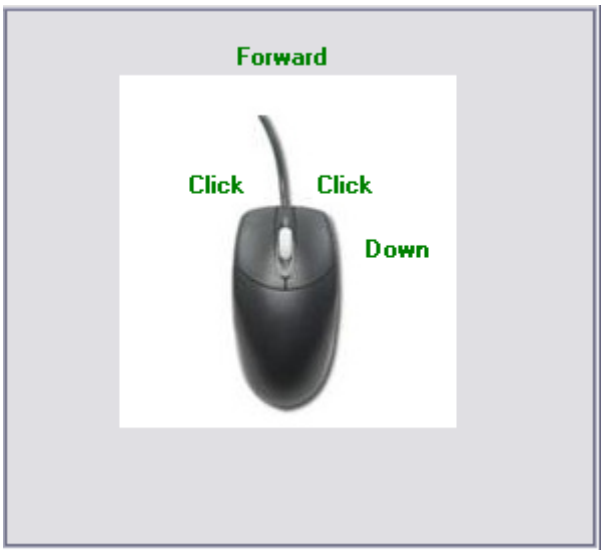
Οι ιδιότητες, που είναι δυνατό να τοποθετηθούν σε ένα εξάρτημα, είναι άπειρες και εξαρτώνται από τις ανάγκες μας και την φαντασία μας, ενώ στην καρτέλα properties έχουν κατηγοριοποιηθεί οι σημαντικότερες από αυτές. Το σημαντικότερο κομμάτι της ρύθμισης των ιδιοτήτων είναι ότι δεν χρειάζεται να επέμβουμε στον κώδικα αλλά αυτό γίνεται αυτόματα από το NetBeans. Σε περίπτωση που θέλουμε να προσθέσουμε μια ιδιότητα, που δεν μας παρέχεται στις επιλογές της καρτέλας properties, είμαστε αναγκασμένοι να την προγραμματίσουμε εμείς. Μια ακόμα σημαντική κατηγορία της καρτέλας properties είναι η ικανότητα προσθήκης σε ένα εξάρτημα κάποιων γεγονότων, τα οποία ονομάζονται **Events.** Τα γεγονότα αυτά είναι ουσιαστικά κάτι σαν 'πληροφοριοδότες', που πληροφορούν το εξάρτημα ότι κάποιο εξωτερικό γεγονός έχει συμβεί από τον χρήστη και το εξάρτημα το 'ακούει' και εκτελεί ότι είναι προγραμματισμένο να κάνει όταν συμβεί το event. Τέτοια εξωτερικά γεγονότα είναι το πάτημα ενός κουμπιού του ποντικιού του Η/Υ, η πληκτρολόγηση κάποιου χαρακτήρα στο πληκτρολόγιο του Η/Υ, το άνοιγμα ή το κλείσιμο κάποιου παραθύρου κ.α. Αναφέρουμε κάποια χαρακτηριστικά events:

- ✓ actionPerform: Εκτελείται μια λειτουργία που εμείς προγραμματίζουμε.
- ✓ KeyEvents (keyPressed-keyReleased-keyTyped): Εκτελείται μια λειτουργία όταν συμβεί κάποιο παρατεταμένο πάτημα στο πληκτρολόγιο ή απελευθέρωση κάποιου πατημένου κουμπιού του πληκτρολογίου ή απλώς με το πάτημα ενός κουμπιού του πληκτρολογίου. Στην εικόνα 2.10 παρουσιάζεται το 'μονοπάτι' της λειτουργίας ενός keyEvent.

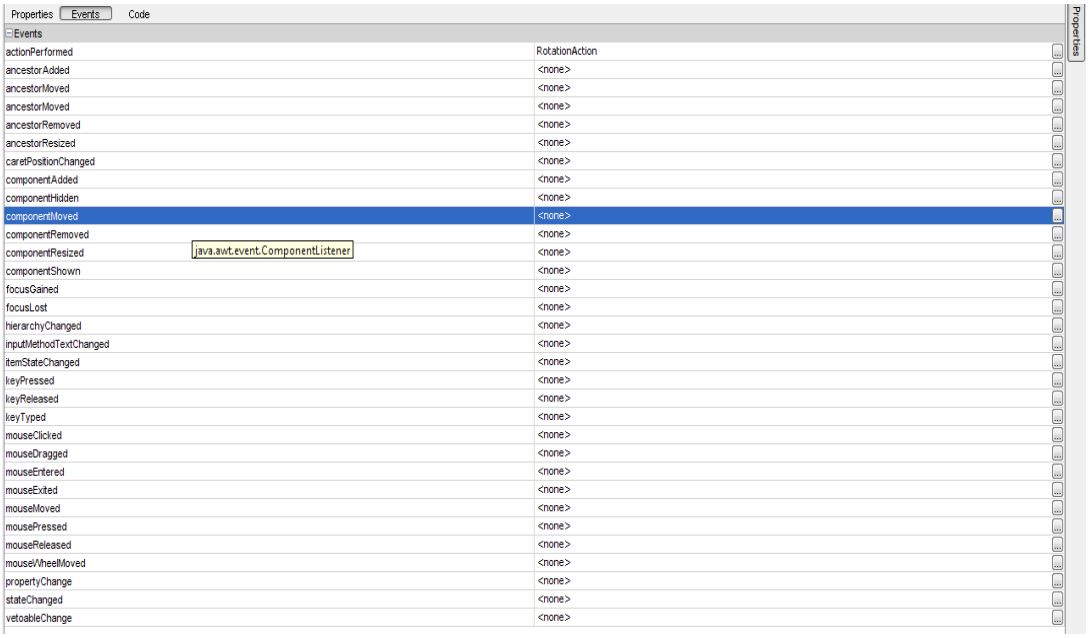


Εικόνα 2.10. Key Events path [5]

- ✓ MouseEvents (mousePressed-mouseReleased-mouseClicked-mouseDragged-mouseMoved-mouseEntered-mouseExited-mouseWheelMoved): Εκτελείται μια λειτουργία που θέλουμε να συμβεί όταν λάβει χώρα κάποια λειτουργία του ποντικιού, όπως πάτημα κάποιου κουμπιού ή του ρότορα, μετακίνηση του ποντικιού κ.α.



Εικόνα 2.11. Mouse Events [5]



Εικόνα 2.12. JFrame Events

Τελευταία κατηγορία της καρτέλας properties είναι μία κατηγορία που ονομάζεται **Code** και αφορά προγραμματιστικές ιδιότητες κάθε εξαρτήματος, όπως αν θα είναι private, protected ή public ως μεταβλητή κώδικα κ.α. Συνήθως δεν ασχολούμαστε με αυτή την κατηγορία καθώς είναι προτιμότερο να μεταβάλουμε τις προγραμματιστικές ιδιότητες των εξαρτημάτων μέσα στον ίδιο τον κώδικα.

Στο κενό πλαίσιο του παραθύρου JFrame (βλ. εικόνα 2.3) τοποθετούμε μία βάση απεικόνισης, που καλείται **JPanel**. Το JPanel αποτελεί ένα Swing component, το οποίο χρησιμοποιείται ως ο χώρος πάνω στον οποίο θα απεικονίζονται τα πάντα που σχεδιάστηκαν στον viewer3d. Μπορούμε να φανταστούμε το JPanel ως ένα κενό δωμάτιο, που αποτελεί ένα studio μιας εκπομπής, ενώ μέσα στον χώρο αυτό τοποθετούνται σκηνικά (scene-objects), τα οποία απεικονίζονται στον τηλεθεατή μέσω της τηλεόρασης.

2.2.2 Απαιτήσεις Viewer-Μεθοδολογία κατασκευής

Σύμφωνα με τις απαιτήσεις της εν λόγω εργασίας ο viewer θα πρέπει να έχει τις ακόλουθες δυνατότητες:

- ***Δυνατότητα περιστροφής αντικειμένων γύρω από άξονα x.***
- ***Δυνατότητα περιστροφής αντικειμένων γύρω από άξονα y.***
- ***Δυνατότητα περιστροφής αντικειμένων γύρω από άξονα z.***
- ***Δυνατότητα επαναφοράς αντικειμένου σε αρχική θέση.***
- ***Δυνατότητα «wireframe» απεικόνισης αντικειμένων.***
- ***Δυνατότητα «points» απεικόνισης αντικειμένων.***
- ***Δυνατότητα «shading» απεικόνισης αντικειμένων.***
- ***Δυνατότητα αλλαγής υλικού απεικόνισης αντικειμένου.***
- ***Δυνατότητα μεγέθυνσης και σμίκρυνσης αντικειμένων (zoom in & zoom out).***
- ***Δυνατότητα μετακίνησης αντικειμένων (translate).***
- ***Δυνατότητα περιστροφής αντικειμένων (rotate).***
- ***Δυνατότητα επιλογής ξεχωριστής εμφάνισης διαφορετικών επιφανειών.***
- ***Δυνατότητα επιλογής αριθμού σημείων ελέγχου, στην ακτινική και περιφερειακή διεύθυνση.***

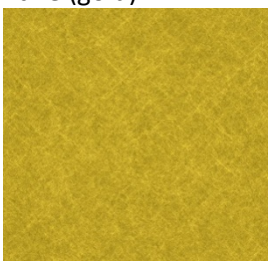
Με βάση την ανάγκη για ευκολία στο χειρισμό του viewer από τον χρήστη και για την πλήρη κάλυψη των απαιτήσεων κατά την κατασκευή του χρησιμοποιήθηκαν ειδικά εξαρτήματα, όπως κουμπιά (JButtons-JRadioButtons), εξαρτήματα JCheckBoxes ή JComboBoxes. Αυτά προστέθηκαν πάνω στο παράθυρο JFrame και στη συνέχεια προγραμματίστηκαν κατάλληλα ώστε να εκτελούν την επιθυμητή εργασία, λαμβάνοντας πάντα υπόψη τον είδη προκαθορισμένο σκοπό χρησιμοποίησης και λειτουργίας τους.

Για τις απαιτήσεις της περιστροφής γύρω από τους άξονες x,y,z χρησιμοποιήσαμε εξαρτήματα που ονομάζονται **JButtons.** Τα **JButtons** είναι εξαρτήματα που υπάρχουν στην εργαλειοθήκη του JFrame και αποτελούν components του πακέτου Swing. Επεμβαίνοντας στις ιδιότητες τους, δώσαμε τα επιθυμητά χαρακτηριστικά αλλά και ορίσαμε την λειτουργικότητα τους. Κάθε JButton ονομάστηκε με τέτοιο τρόπο ώστε να γίνεται εύκολα και άμεσα αντιληπτή από τον χρήστη η λειτουργία του. Έτσι για παράδειγμα το JButton που περιστρέφει κατά μία γωνιά α το αντικείμενο γύρω από τον άξονα **x** ονομάστηκε Xrotation. Με αυτή τη λογική ονομάστηκαν και τα υπόλοιπα JButtons ως YRotation και ZRotation. Κάθε φορά που πατιέται ένα από τα 3 αυτά JButtons το αντικείμενο που απεικονίζεται περιστρέφεται προς μία προκαθορισμένη γωνία γύρω από τον άξονα που αντιπροσωπεύει το κάθε κουμπί.

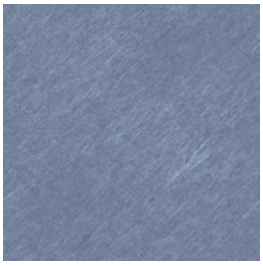
Πριν τοποθετηθούν τα JButtons στο παράθυρο JFrame και για σκοπούς καλύτερης απεικόνισης, εισάγαμε ένα εξάρτημα που καλούμε ‘μπάρα’ εργαλείων. Τέτοιες μπάρες εργαλείων χρησιμοποιούνται ευρέως σε εφαρμογές κατασκευής παραθύρων, καθώς διευκολύνουν τον χειρισμό παραπάνω του ενός components αλλά και προσφέρουν μια αρκετά καλύτερη απεικόνιση του παραθύρου, διαχωρίζοντας καλύτερα τον τριδιάστατο χώρο απεικόνισης από τα χρήσιμα εργαλεία της εφαρμογής. Αυτές οι μπάρες εργαλείων ονομάζονται **JToolBars**. Προστέθηκε μία JToolBar, όπου τοποθετήθηκαν όλα τα εξαρτήματα της εφαρμογής μας.

Για να καλύψουμε τις απαιτήσεις, που αφορούν στην αλλαγή εμφάνισης του αντικειμένου από όγκο σε πλέγμα ή νέφους σημείων και αντίστροφα αλλά και τις απαιτήσεις για ικανότητα αλλαγής υλικού, χρησιμοποιήσαμε κάποια εξαρτήματα που ονομάζονται **JComboBox**. Τα εξαρτήματα αυτά αποτελούν ένα είδος λίστας επιλογών. Η προκαθορισμένη λίστα είναι κενή και εμείς τη συμπληρώνουμε, ανάλογα με τις απαιτήσεις μας, με αντικείμενα που καλούνται Items. Κάθε Item είναι ένας δείκτης που δείχνει τι θα συμβεί όταν επιλεχθεί. Στο πρώτο JComboBox, που αφορά την ικανότητα που θέλουμε να προσδώσουμε στον viewer να μπορεί να ελέγχεται το υλικό της εκάστοτε επιφάνειας, προστέθηκαν τα εξής Items:

- Primitive (gold)



- Iron



- Aluminium

- Bronze

Το Item primitive είναι το προκαθορισμένο (Default) υλικό όλων των επιφανειών. Έχει υφή χρυσού και επιλέχτηκε λόγω της καλύτερης απεικόνισης που παρέχει αλλά και επειδή συνηθίζεται να χρησιμοποιείται στα σχεδιαστικά προγράμματα. Το Item Iron δίνει υφή σίδηρου στο αντικείμενο που απεικονίζεται, το Aluminum υφή αλουμινίου και το Bronze δίνει υφή χαλκού.

Στο δεύτερο JComboBox, που αφορά στην απεικόνιση του αντικειμένου σε σχέση με τον τρόπο εμφάνισης του, προστέθηκαν τα εξής Items:

- Shading
- Wireframe
- points

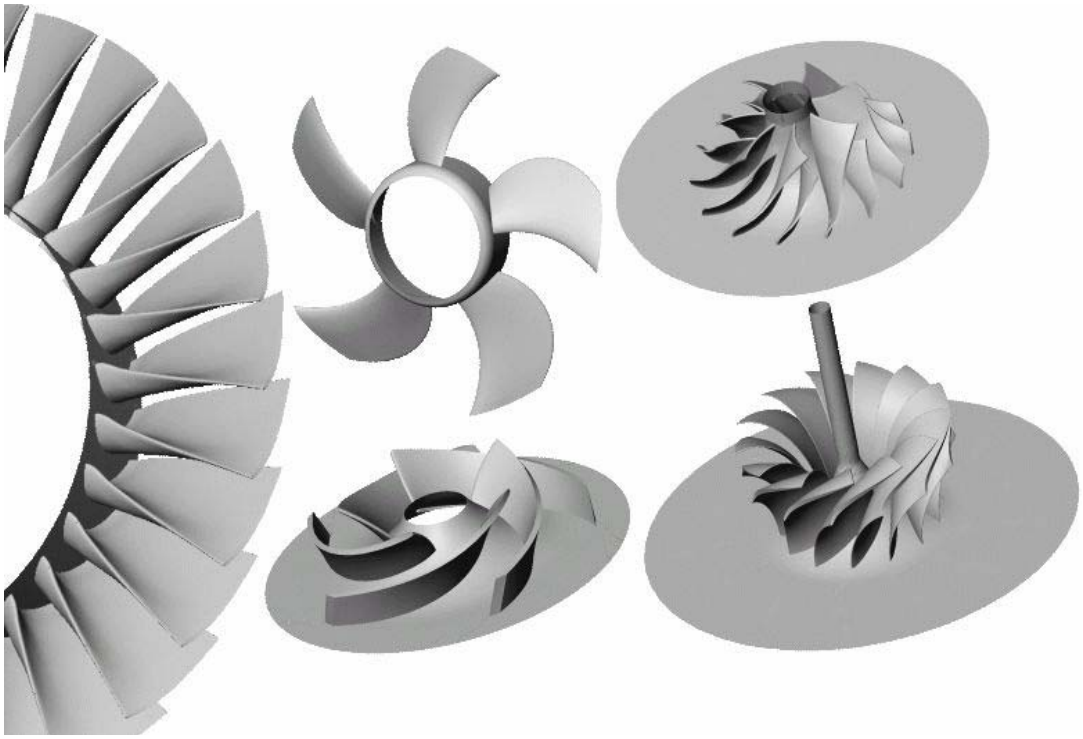
Όσον αφορά στην απεικόνιση των αντικειμένων παρατηρούμε ότι ο viewer θα παρέχει την δυνατότητα για την απεικόνιση σε τρεις μορφές, shading, wireframe και points.

Με τον όρο shading (σκίαση) εννοούμε την απεικόνιση του αντικειμένου σε μορφή όγκου. Κατά την απεικόνιση του όγκου παρουσιάζονται μόνον οι εξωτερικές επιφάνειες του συνόλου του όγκου. Υπάρχει δυνατότητα της απεικόνισης και των εσωτερικών επιφανειών του όγκου γεγονός, που κρίθηκε άσκοπο και σε αρκετές περιπτώσεις λανθασμένο.

Με τον όρο wireframe εννοούμε την απεικόνιση των αντικειμένων σε μορφή τριγωνικού πλέγματος, δηλαδή γραμμών που ενώνουν τα σημεία, τα οποία αποτελούν το νέφος σημείων από το οποίο ορίζεται το αντικείμενο.

Με τον όρο points εννοούμε την απεικόνιση των αντικειμένων σε μορφή νέφους σημείων από τα οποία και ορίζονται οι εξωτερικές τους επιφάνειες.

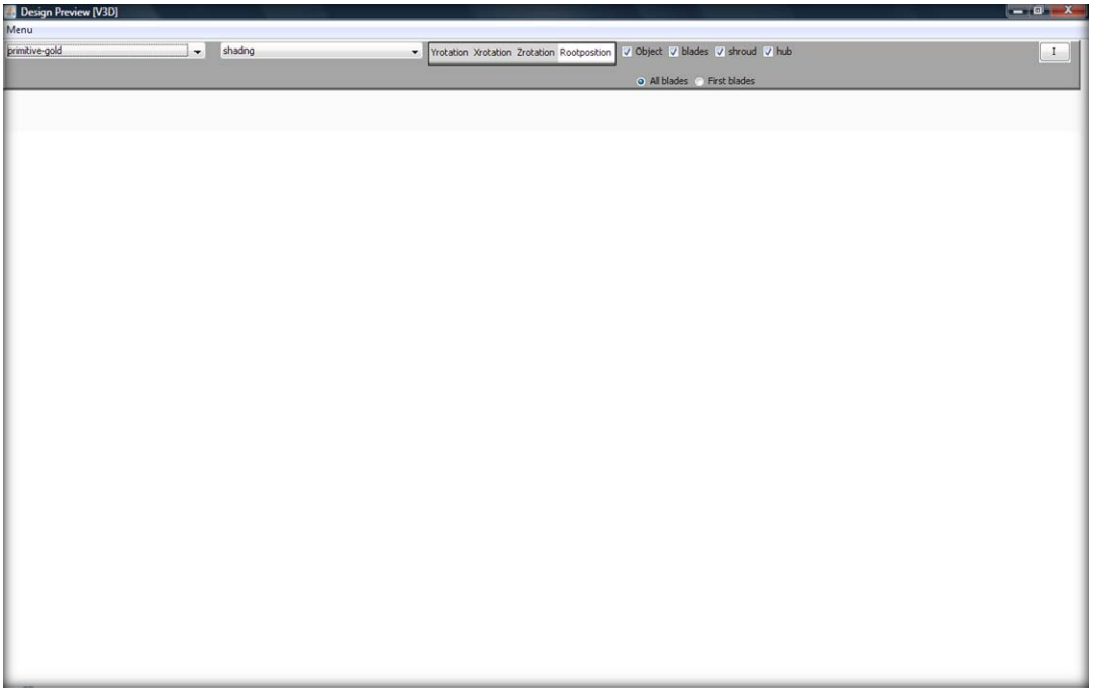
Μια ακόμη απαίτηση κατά την κατασκευή του viewer ήταν η ικανότητα επιλογής ξεχωριστής εμφάνισης των διαφορετικών επιφανειών του αντικειμένου. Αυτή η λειτουργία είναι πολύ σημαντική, ειδικά στο πρόγραμμα T4T που σχεδιάζει πτερυγώσεις στροβιλομηχανών, που η προβολή με λεπτομέρεια κάθε ξεχωριστής επιφάνειας είναι ιδιαίτερα σημαντική. Η γεωμετρία μίας τυπικής πτερωτής αποτελείται από τριών ειδών επιφάνειες: τα πτερύγια (blades), η πλήμνη (hub) και το κέλυφος (shroud). Η γεωμετρία όμως των πτερυγίων είναι αυτή που καθορίζει σε μεγάλο βαθμό τη συνολική γεωμετρία της πτερωτής. Για το λόγο αυτό η μεθοδολογία σχεδίασης που ακολουθείται από το T4T έχει ως πυρήνα τη σχεδίαση των πτερυγίων. Όσον αφορά στις επιφάνειες της πλήμνης και του κελύφους, η δημιουργία τους είναι αρκετά απλή, χωρίς αυτό όμως να υποβαθμίζει τη σημαντικότητά τους, όπως θα φανεί και στην συνέχεια. Για να μπορούμε να ελέγχουμε την απεικόνιση των επιφανειών χρησιμοποιήσαμε το εξάρτημα **JCheckBox** πακέτου Swing. Το εξάρτημα αυτό έχει την ιδιότητα να επιλέγεται και να από-επιλέγεται και να εκτελεί την αντίστοιχη λειτουργία που έχει προγραμματιστεί να κάνει αντίστοιχα. Εκμεταλλευόμενοι την ιδιότητα αυτή χρησιμοποιήσαμε 4 JCheckBoxes ένα για τισ επιφάνειες των πτερυγίων, ένα για την επιφάνεια του κελύφους, ένα για την επιφάνεια της πλήμνης και ένα για όλες τις επιφάνειες μαζί. Τα ονόματα κάθε ενός JCheckBox είναι αντίστοιχα της επιφάνειας που αντιπροσωπεύει το κάθε ένα. Έτσι το JCheckBox που αντιπροσωπεύει τις επιφάνειες των πτερυγίων εμφανίζεται πάνω στον viewer ως Blades. Αντίστοιχα τα υπόλοιπα ονομάστηκαν Hub, Shroud, και Object για την πλήμνη, το κέλυφος και όλο το αντικείμενο αντίστοιχα. Όταν επιλέγεται ένα από τα παραπάνω JCheckBoxes εμφανίζεται η επιφάνεια που αντιπροσωπεύει, ενώ στην αντίθετη περίπτωση εξαφανίζεται η επιφάνεια που αντιπροσωπεύει. Στην εικόνα 2.12 εμφανίζονται χαρακτηριστικές επιφάνειες πτερυγίων, πλήμνης και κελύφους.



Εικόνα 2.13. Επιφάνειες πτερυγίων, πλήμνης, κελύφους [6]

Κάθε στροβιλομηχανή μπορεί να είναι μονοβάθμια ή πολυβάθμια. Αυτό σημαίνει ότι μπορεί να αποτελείται από μία μόνο πτερύγωση με συγκεκριμένο αριθμό πτερυγίων ή από πολλές συνεχόμενες πτερυγώσεις με διαφορετικό συνήθως αριθμό πτερυγίων η κάθε μία. Για αυτό το λόγο κρίθηκε σκόπιμο να παρέχεται η δυνατότητα επιλογής απεικόνισης ολοκληρωμένων πτερυγώσεων στον viewer3d ή η απεικόνιση μόνο ενός πτερυγίου κάθε πτερύγωσης. Η σχεδίαση ενός μόνο πτερυγίου μοιάζει να μην έχει φυσική υπόσταση. Κατά την παραμετρική σχεδίαση όμως του πολυβάθμιων πτερυγώσεων στροβιλομηχανών αυτό που συμβαίνει είναι να σχεδιάζεται ένα πτερύγιο κάθε πτερύγωσης και με κάποια τεχνική αντιγραφής της γεωμετρίας του, περιστρέφονται αντίγραφα του γύρω από τον άξονα με βήμα δεδομένης γωνίας, σχηματίζοντας τη συνολική πτερύγωση. Ο λόγος που θέλουμε να υπάρχει αυτή η δυνατότητα σχετίζεται με θέματα ταχύτητας και ανάλυσης του viewer3d. Σε πολύπλοκες γεωμετρίες με μεγάλο αριθμό πτερυγίων το νέφος σημείων είναι τεράστιο με αποτέλεσμα να μειώνεται η ταχύτητα απεικόνισης (rendering). Αυτό μας αναγκάζει να μειώσουμε με κάποια τεχνική το νέφος σημείων, με αποτέλεσμα να μειώνεται η ποιότητα απεικόνισης της επιφάνειας των πτερυγίων. Έχοντας τη δυνατότητα να εμφανίζουμε μόνο ένα πτερύγιο κάθε πτερύγωσης μπορούμε να αυξήσουμε την ανάλυση κάθε πτερυγίου και να μειώσουμε το συνολικό μέγεθος των σημείων, πετυχαίνοντας υψηλή ποιότητα απεικόνισης της αντίστοιχης επιφάνειας, με μείωση του χρόνου απόκρισης. Ασχολούμαστε μόνο με το νέφος σημείων των πτερυγώσεων και όχι με αυτά του κελύφους και της πλήμνης καθώς αυτές είναι συνήθως πιο απλές γεωμετρίες, οι οποίες μπορούν να περιγραφούν εύκολα και από λίγα σημεία.

Για να εφαρμόσουμε στην πράξη αυτή την λειτουργία χρησιμοποιήσαμε ένα εξάρτημα του πακέτου Swing που λέγεται **JRadioButton**. Αυτή η κατηγορία κουμπιών δίνει τη δυνατότητα να ομαδοποιούνται και η επιλογή του ενός να ακυρώνει την επιλογή των άλλων, με αποτέλεσμα να ενεργοποιείται αποκλειστικά η λειτουργία του επιλεγμένου JRadioButton. Το ένα JRadioButton παρουσιάζεται στον viewer3d με το όνομα AllBlades και όταν το επιλέγουμε απεικονίζεται το σύνολο των πτερυγώσεων. Το δεύτερο JRadioButton παρουσιάζεται στον viewer3d με το όνομα FirstBlades και όταν το επιλέγουμε απεικονίζεται μόνο το πρώτο πτερύγιο κάθε πτερύγωσης. Όλα τα εξαρτήματα έχουν τοποθετηθεί πάνω σε ένα JPanel (γκρι χρώμα) Στην εικόνα 2.13 παρουσιάζεται το ολοκληρωμένο παράθυρο JFrame με όλα τα εξαρτήματα που του έχουν προστεθεί.



Εικόνα 2.14. Παράθυρο JFrame με JPanel,JToolBar και τα JButtons-JRadioButtons-JComboBox-JCheckBox

Σε πλήρη αντιστοίχηση η λειτουργία που εκτελεί κάθε JButton είναι:

- *Yrotation JButton*: Περιστρέφει το αντικείμενο κατά 90 μοίρες γύρω από τον άξονα **y**.
- *Xrotation JButton*: Περιστρέφει το αντικείμενο κατά 90 μοίρες γύρω από τον άξονα **x**.
- *Zrotation JButton*: Περιστρέφει το αντικείμενο κατά 90 μοίρες γύρω από τον άξονα **z**.
- *Rootposition JButton*: Επαναφέρει το αντικείμενο στην αρχική του θέση.
- *Primitive item*: Αλλάζει το υλικό του αντικειμένου σε χρυσό(default).
- *Iron item*: Αλλάζει το υλικό του αντικειμένου σε σίδηρο.
- *Aluminium item*: Αλλάζει το υλικό του αντικειμένου σε αλουμίνιο.
- *Bronze item*: Αλλάζει το υλικό του αντικειμένου σε χαλκό.
- *Wireframe item*: Αλλάζει την απεικόνιση αντικειμένου σε wireframe.
- *Points item*: Αλλάζει την απεικόνιση αντικειμένου σε σημεία (points).
- *Shading item*: Αλλάζει την απεικόνιση αντικειμένου σε όγκο (shading).
- *Object JCheckBox:* Απεικονίζονται όλες οι επιφάνειες του αντικειμένου.
- *Blades JCheckBox:* Απεικονίζονται οι επιφάνειες των πτερυγίων του αντικειμένου.
- *Hub JCheckBox:* Απεικονίζεται η επιφάνεια της πλήμνης του αντικειμένου.
- *Shroud JCheckBox:* Απεικονίζεται η επιφάνεια του κελύφους του αντικειμένου.
- *AllBlades JRadioButton:* Απεικόνιση του συνόλου των πτερυγίων.
- *FirstBlades JRadioButton:* Απεικόνιση ενός μόνον πτερυγίου από κάθε πτερύγωση.

Για να επιτευχθεί η επιθυμητή λειτουργία κάθε JButton ήταν αναγκαίο να επέμβουμε στον κώδικα κατασκευής κάθε εξαρτήματος, ο οποίος παρέχεται έτοιμος, και να κατασκευάσουμε μία δομή προγράμματος με χρήση κατάλληλων εντολών η οποία να εκτελεί τη λειτουργία που επιθυμούμε. Το προγραμματιστικό κομμάτι αυτής της εργασίας θα αναλυθεί στο επόμενο κεφάλαιο.

2.2.3 ΔΗΜΙΟΥΡΓΙΑ ΑΛΛΗΛΕΠΙΔΡΑΣΗΣ T4T VIEWER 3D-MOUSE.

Αναλύοντας την κατασκευή του παραθύρου JFrame καλύψαμε 14 από τις 17 απαιτήσεις στην λειτουργία του viewer. Οι 3 τελευταίες αφορούσαν στην ικανότητα μεγέθυνσης και σμίκρυνσης (zoom in & zoom out) του αντικειμένου, τη δυνατότητα περιστροφής (rotation) και τη δυνατότητα μεταφοράς (translation) μέσα στον viewer. Για να επιτευχθούν αυτές οι απαιτήσεις κατασκευάστηκε μια δομή κώδικα, η οποία επιτρέπει την αλληλεπίδραση του ποντικιού του H\Y με τον viewer. Αυτή η αλληλεπίδραση παρέχεται έτοιμη από την βιβλιοθήκη Java3d API, καθώς μας παρέχει έτοιμες κλάσεις που ενσωματώνουμε στο λογισμικό μας, δημιουργώντας αντικείμενα που δείχνουν σε αυτές τις κλάσεις (NodeCompnets). Συγκεκριμένα χρησιμοποιούμε τις κλάσεις MouseZoom (για την μεγέθυνση και σμίκρυνση του αντικειμένου), MouseRotate (για την περιστροφή του αντικειμένου) και MouseTranslate (για την μεταφορά του αντικειμένου). Η αλληλεπίδραση αυτή αφορά στην 'ειδοποίηση' του viewer και των αντικειμένων που εμφανίζονται σε αυτόν, όταν ενεργοποιηθεί κάποια λειτουργία του ποντικιού. Οι πιθανές λειτουργίες του, ή αλλιώς τα πιθανά 'γεγονότα', από τα οποία ειδοποιείτε ο viewer είναι:

- Παρατεταμένο πάτημα του αριστερού κουμπιού του ποντικιού και μετακίνηση του.
- Παρατεταμένο πάτημα του δεξιού κουμπιού του ποντικιού και μετακίνηση του.
- Παρατεταμένο πάτημα του ρότορα του ποντικιού και μετακίνηση του.

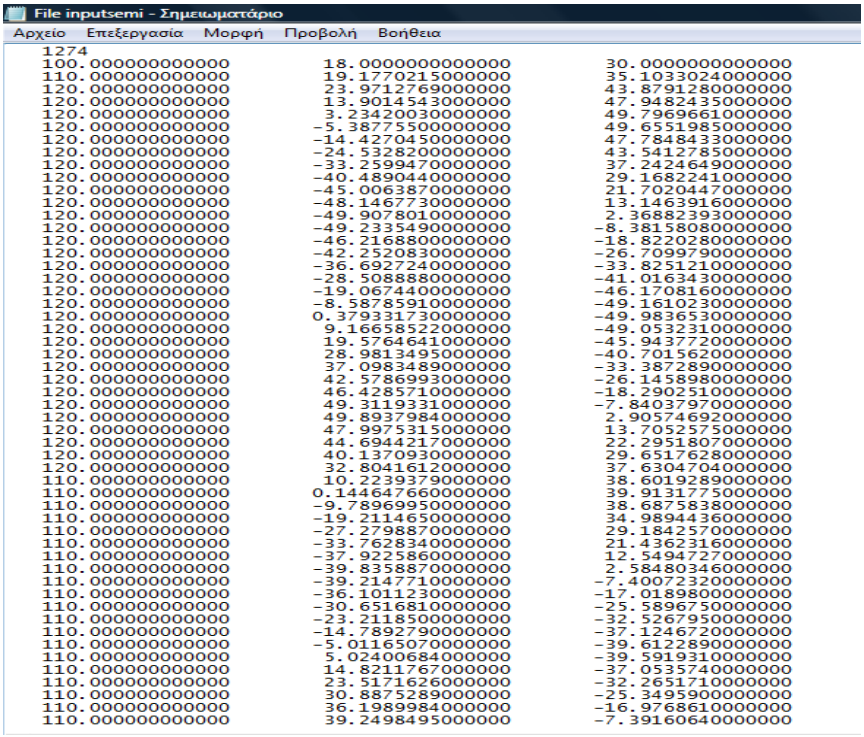
Όταν συμβαίνει το πρώτο γεγονός ειδοποιείται η κλάση MouseRotate και προκαλείται περιστροφή του αντικειμένου (rotation).

Όταν συμβαίνει το δεύτερο γεγονός ειδοποιείται η κλάση MouseTranslate και προκαλείται μετακίνηση του αντικειμένου (translate), ανάλογη μάλιστα της μετακίνησης του ποντικιού στην επιφάνεια του τραπεζιού.

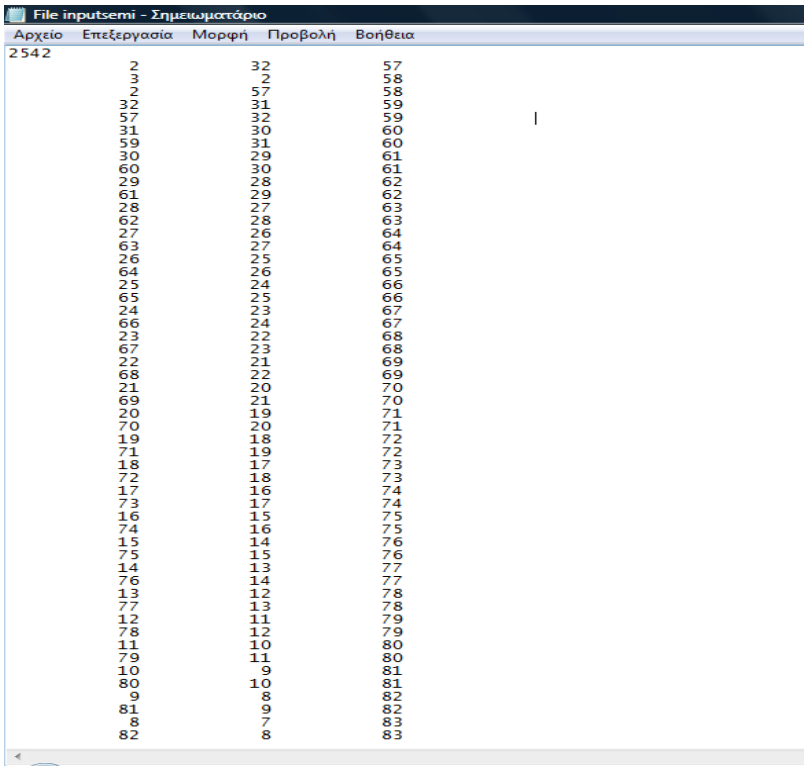
Όταν συμβαίνει το τρίτο γεγονός ειδοποιείται η κλάση MouseZoom και προκαλείται μεγέθυνση ή σμίκρυνση του αντικειμένου ανάλογα με την μετακίνηση του ποντικιού. Εάν η μετακίνηση είναι με κατεύθυνση προς τον χρήστη προκαλείται μεγέθυνση (zoom in) ενώ στην αντίθετη περίπτωση προκαλείτε σμίκρυνση (zoom out).

2.3. ΔΗΜΙΟΥΡΓΙΑ ΑΡΧΕΙΟΥ ΕΙΣΟΔΟΥ ΔΕΔΟΜΕΝΩΝ

Σε περίπτωση που ο χρήστης επιθυμεί να περάσει κάποια γεωμετρία στον Viewer3d που να μην έχει σχεδιαστεί παραμετρικά στο T4T, αλλά πιθανώς σε κάποιο άλλο πρόγραμμα σχεδιασμού, δίνεται η δυνατότητα από το πρόγραμμα να χρησιμοποιήσει ένα αρχείο μορφής txt, το οποίο μπορεί ο viewer3d να διαβάσει και να εμφανίσει την εν λόγω γεωμετρία. Η κατασκευή κατάλληλης δομής κώδικα δίνει στο viewer την δυνατότητα να διαβάζει από πόσα σημεία αποτελείται το νέφος των σημείων, στη συνέχεια να διαβάζει τις συντεταγμένες στο χώρο κάθε σημείου ξεχωριστά, στη συνέχεια από πόσα τρίγωνα θα αποτελείται το τριγωνικό πλέγμα και τέλος τον τρόπο που ενώνονται τα σημεία ώστε να κατασκευαστεί το πλέγμα. Στην εικόνα 2.14 παρουσιάζεται ένα τέτοιο αρχείο κειμένου. Στην εικόνα 2.15 παρουσιάζεται η συνέχεια του αρχείου στο όπου μετά το πέρας όλων των συντεταγμένων των σημείων δίνεται ο αριθμός των τριγώνων που αποτελούν το πλέγμα και στη συνέχεια ο τρόπος που ενώνονται τα σημεία ώστε να σχηματίσουν τα τρίγωνα από τα οποία αποτελείται το πλέγμα.



Εικόνα 2.15. Αρχείο κειμένου - αριθμός σημείων, συντεταγμένες σημείων



Εικόνα 2.16. Αρχείο κειμένου - αριθμός τριγώνων, ενώσεις σημείων

2.4. ΣΧΕΔΙΑΣΜΟΣ ΓΕΩΜΕΤΡΙΑΣ ΑΝΤΙΚΕΙΜΕΝΟΥ

Ο σχεδιασμός της γεωμετρίας του αντικειμένου αποτελεί μια σχετικά πολύπλοκη διαδικασία, η οποία περιγράφεται στα παρακάτω βήματα:

1. Δημιουργούμε ένα αντικείμενο τύπου γεωμετρίας, το οποίο ονομάζεται GeometryArray. Αναλόγως με το τι πλέγμα χρησιμοποιούμε υπάρχουν και οι κατάλληλες υποκατηγορίες του αντικειμένου, δηλαδή αν χρησιμοποιούμε (όπως στην εργασία μας) τριγωνικό τότε το κατάλληλο αντικείμενο που δημιουργούμε είναι το TriangleArray. Περισσότερες λεπτομέρειες περιέχονται στο κεφάλαιο 3.
2. Αποθηκεύουμε σε αυτό το αντικείμενο όλες τις πληροφορίες που έχουμε δώσει παραμετρικά ή έχουμε διαβάσει από το αρχείο κειμένου, που αφορούν το πλήθος των σημείων, τις συντεταγμένες τους, το πλήθος των τριγώνων του πλέγματος και τις ενώσεις τους.
3. Προσδίδουμε στο αντικείμενο τις ιδιότητες (βλ 2.4) που επιθυμούμε, όπως η ικανότητα χρωματισμού, η ικανότητα προσδιορισμού των κάθετων διανυσμάτων, η ικανότητα προσθήκης κειμένου πάνω στο αντικείμενο κ.α.
4. Προσδιορίζουμε τα κάθετα διανύσματα. Αυτό το βήμα είναι ιδιαίτερα σημαντικό για τον αποτελεσματικό σχεδιασμό της γεωμετρίας και θα αναλυθεί περεταίρω στη συνέχεια.
5. Προσθέτουμε στο αντικείμενο όλες τις επιθυμητές ιδιότητες. Τέτοιες ιδιότητες αφορούν τον τύπο χρωματισμού, το υλικό, ιδιότητες απεικόνισης κ.α.
6. Δημιουργούμε ένα αντικείμενο Shape 3d.
7. Αποθηκεύουμε το αντικείμενο τύπου γεωμετρίας με όλες τις ιδιότητες του στο αντικείμενο τύπου Shape 3d.
8. Απεικονίζουμε το Shape 3d στον viewer.

Τα παραπάνω βήματα αποτελούν μια πρώτη και πιο γενική προσέγγιση της σχεδίασης της γεωμετρίας του αντικειμένου. Κρίνεται άσκοπο να εισέλθουμε σε αυτό το εδάφιο σε προγραμματιστικές λεπτομέρειες, καθώς τα θέματα που αφορούν στον κώδικα που αναπτύχθηκε θα αποτελέσουν αντικείμενο παρουσίασης του επόμενου κεφαλαίου. Πρέπει να σημειωθεί ότι η δημιουργία αντικειμένων είναι μια διαδικασία αρκετά εύκολη, καθώς μας παρέχονται έτοιμες κλάσεις από την βιβλιοθήκη Java3d API. Έτσι για παράδειγμα δημιουργώντας ένα Shape3d αντικείμενο ουσιαστικά δημιουργούμε ένα αντικείμενο κλάσης Shape3D της βιβλιοθήκης που έχει όλες τις ιδιότητες που κληρονομεί από την κλάση αυτή. Εμείς έπειτα μπορούμε να προσδώσουμε επιπλέον ιδιότητες.

Ενώ όλα τα παραπάνω βήματα αποτελούν σημαντικούς κρίκους της αλυσίδας των βημάτων για την σχεδίαση της γεωμετρίας ενός αντικειμένου, κρίνεται σκόπιμο να γίνει μια ιδιαίτερη αναφορά στον προσδιορισμό των κάθετων διανυσμάτων. Αυτό συμβαίνει διότι ο σωστός προσδιορισμός των κάθετων διανυσμάτων είναι και η βάση για την σωστή σχεδίαση των επιφανειών του αντικειμένου. Μια σημαντική λειτουργία που πετυχαίνουμε με τον κατάλληλο προσδιορισμό των κάθετων διανυσμάτων είναι η σωστή προοπτική του αντικειμένου στο χώρο και η σωστή ανάκλαση από το αντικείμενο των φωτισμών που χρησιμοποιούνται από τον viewer. Για ένα υποθετικό τρίγωνο που ορίζεται από τα διανύσματα θέσης των κορυφών του $\vec{p}_0, \vec{p}_1, \vec{p}_2$ οι μαθηματικές εξισώσεις για τον προσδιορισμό του κάθετου στο τρίγωνο διάνυσμα είναι:

$$\bar{t}_1 = \bar{p}_1 - \bar{p}_0 \quad \bar{t}_2 = \bar{p}_2 - \bar{p}_0$$
$$\bar{n} = \bar{t}_1 \times \bar{t}_2$$
$$\bar{n} \cdot \bar{t}_1 = \bar{n} \cdot \bar{t}_2 = 0$$

2.5. ΚΑΘΟΡΙΣΜΟΣ ΙΔΙΟΤΗΤΩΝ ΕΜΦΑΝΙΣΗΣ ΑΝΤΙΚΕΙΜΕΝΟΥ.

Η Java 3D API μας παρέχει ένα πακέτο εντολών για τον πλήρη προσδιορισμό των ιδιοτήτων εμφάνισης ενός αντικειμένου. Οι ιδιότητες αυτές ονομάζονται Attributes και χωρίζονται στις εξής κατηγορίες:

- Point Attributes: Πρόκειται για όλες τις ιδιότητες που αφορούν στα σημεία του πλέγματος στο χώρο. Τέτοιες ιδιότητες είναι το μέγεθος (σε pixels) τους και ο καθορισμός της καμπυλότητας τους. Για να γίνουν εμφανείς αυτές οι ιδιότητες πρέπει να επιλεχθεί από τον χρήστη η Points μορφή του αντικειμένου.
- Polugon Attributes: Πρόκειται για τις ιδιότητες του αντικειμένου με την μορφή όγκου. Τέτοιες ιδιότητες είναι ο καθορισμός της απόκρυψης ή όχι επιφάνειας ανάλογα με το βάθος που βρίσκεται, η αναστροφή κατεύθυνσης των κάθετων διανυσμάτων κ.α. Για να γίνουν εμφανείς οι ιδιότητες αυτές πρέπει να επιλεχθεί από το χρήστη η Shade μορφή του αντικειμένου.
- Line Attributes: Πρόκειται για τις ιδιότητες που αφορούν στις γραμμές που ενώνουν τα σημεία στο πλέγμα. Οι ιδιότητες αυτές αφορούν στον καθορισμό του τύπου γραμμής, δηλαδή αν θα είναι συνεχής ή διακεκομμένη με παύλες ή διακεκομμένη με τελείες, το πάχος της γραμμής (σε pixels), καθώς και στον καθορισμό της καμπυλότητάς τους. Για να γίνουν εμφανείς οι ιδιότητες αυτές πρέπει να επιλεχθεί από το χρήστη η Wireframe μορφή του αντικειμένου.
- Coloring Attributes : Πρόκειται για τις ιδιότητες που αφορούν στο χρωματισμό του αντικειμένου σε όποια μορφή και αν βρίσκεται. Εκτός από το χρώμα καθορίζεται και ο τύπος του χρωματισμού, όπως ομοιόμορφος χρωματισμός ή όχι κ.α.
- Transparency Attributes: Πρόκειται για τις ιδιότητες που αφορούν στο επίπεδο διαφάνειας του αντικειμένου καθώς και στον καθορισμού του τύπου διαφάνειας.
- Rendering Attributes: Πρόκειται για τις ιδιότητες που αφορούν στον τρόπο και στη διάταξη που θα απεικονίζονται τα αντικείμενα, καθώς και σε λειτουργίες που αφορούν στο βάθος της οπτικής του viewer. Σε περιπτώσεις δηλαδή που υπάρχουν περισσότερα από ένα αντικείμενα στον εικονικό χώρο μπορούμε να επιλέξουμε αν θα είναι εμφανή και τα δύο ή όχι, αλλά και να δώσουμε μια σωστή αίσθηση του βάθους στο οποίο βρίσκονται με σημείο αναφοράς τον χρήστη καθώς και πληθώρα άλλων επιλογών. Εδώ πρέπει να επισημάνουμε ότι με τον όρο rendering εννοούμε τη διαδικασία απεικόνισης από την στιγμή που θα θέσουμε σε εφαρμογή το πρόγραμμα μέχρι να το τερματίσουμε. Έτσι πολλές φορές θα χρησιμοποιούμε την φράση 'κατά την διάρκεια του rendering' εννοώντας την διάρκεια απεικόνισης των αντικειμένων.

Στο σχήμα 2.3 παρουσιάζονται σε πίνακα οι προκαθορισμένες από την βιβλιοθήκη Java 3D API τιμές των ιδιοτήτων για κάθε μια περίπτωση. Οι ιδιότητες είναι σε μορφή εντολών java.

Attributes Class	**Parameter**	**Default Value**
ColoringAttributes	color	white (1, 1, 1)
	shade model	SHADE_GOURAUD
LineAttributes	line width	1.0

	line pattern	PATTERN_SOLID
	line antialiasing enable	false
PointAttributes	point size	1.0
	point antialiasing enable	false
PolygonAttributes	cull face	CULL_BACK
	backFaceNormalFlip	false
	polygon mode	POLYGON_FILL
	polygonOffset	0.0
	polygonOffsetFactor	0.0
RenderingAttributes	depthBufferEnable	true
	depthBufferWriteEnable	true
	alphaTestFunction	ALWAYS
	alphaTestValue	0.0
	visible	true
	ignoreVertexColors	false
	rasterOpEnable	false
	rasterOp	ROP_COPY
TextureAttributes	textureMode	REPLACE
	textureBlendColor	black (0, 0, 0, 0)
	transform	identity
	textureColorTable	null
	perspectiveCorrectionMode	NICEST
TransparencyAttributes	transparencyMode	NONE
	transparencyValue	0.0
	srcBlendFunction	SRC_BLEND_ALPHA
	dstBlendFunction	BLEND_ONE_MINUS_ALPHA

Σχήμα 2.3. *Προκαθορισμένες τιμές ιδιοτήτων στην Java 3D API* **[1]**

Όπως αναφέρθηκε παραπάνω, αυτές οι τιμές στις ιδιότητες είναι προκαθορισμένες από την βιβλιοθήκη της java που χρησιμοποιούμε. Κατά την κατασκευή του Viewer 3D προέκυψαν διάφορες απαιτήσεις, που μας οδήγησαν στη μεταβολή αρκετών από τις παραπάνω τιμές των ιδιοτήτων. Αναφέρουμε σε αυτό το σημείο όλες τις τιμές όλων των ιδιοτήτων του αντικειμένου και τον λόγο που επιλέξαμε τις συγκεκριμένες τιμές.

- Coloring Attributes (ιδιότητες χρωματισμού): Στις ιδιότητες χρωματισμού έγινε αλλαγή στο shade model του κομματιών που απεικονίζονται, ιδιότητα που αφορά στη φωτοσκίαση των αντικειμένων. Η τιμή που δώσαμε είναι SHADE_FLAT. Αυτή η αλλαγή έγινε για να δώσουμε στις επιφάνειες των αντικειμένων μια μορφή λεία και ομαλή.

- Line Attributes (ιδιότητες γραμμών αντικειμένου): Στις ιδιότητες των γραμμών του αντικειμένου, έχουμε μεταβάλλει την τιμή σε 0.5 στην ιδιότητα 'line width'. Αυτή η αλλαγή αφορά το πάχος της γραμμής με μετρικό σύστημα το pixel. Η προκαθορισμένη τιμή είναι 1 pixel πάχος και κρίθηκε αναγκαία η αλλαγή σε ½ τουpixel καθώς στον viewer θα απεικονίζονται πολύπλοκες γεωμετρίες όπου η ευκρίνεια μεταξύ των γραμμών θα πρέπει

να είναι αρκετά υψηλή. Μειώνοντας το πάχος μειώνουμε το χώρο που καταλαμβάνει κάθε γραμμή αυξάνοντας την απόσταση μεταξύ των γραμμών στο σύνολο της επιφάνειας αυξάνοντας έτσι και την ευκρίνεια της. Άλλη μία ιδιότητα που μεταβάλλαμε από την προκαθορισμένη τιμή της είναι η ιδιότητα 'line antialiasing enable'. Η ιδιότητα αυτή αφορά στην καμπυλότητα των pixels που αποτελούν τις γραμμές της μορφής wireframe. Για λόγους καλύτερης απεικόνισης μεταβάλλαμε την τιμή από false σε true, δίνοντας έτσι μεγαλύτερη καμπυλότητα στα pixels.

- Point Attributes (ιδιότητες σημείων αντικειμένου): Στις ιδιότητες των σημείων του αντικειμένου, έχουμε μεταβάλλει την τιμή της ιδιότητας 'point size' από 1 σε 2. Αυτή η ιδιότητα αφορά στο μέγεθος των σημείων που αποτελούν το νέφος σημείων που ορίζει το κάθε αντικείμενο. Το μετρικό σύστημα του μεγέθους είναι το pixel και με την αλλαγή που έγινε κάθε σημείο αποτελείται από 2 pixels για μεγαλύτερη ευκρίνεια του νέφους σημείων. Επίσης μεταβλήθηκε η τιμή της ιδιότητας 'point antialiasing enable' από false σε true. Ομοίως με τις ιδιότητες γραμμών αυτή η ιδιότητα αφορά στην καμπυλότητα των pixels την οποία μεγαλώσαμε.

- Polygon Attributes (ιδιότητες πολυγώνου): Όπως έχει αναφερθεί παραπάνω οι ιδιότητες πολυγώνου αφορούν τις ιδιότητες των αντικειμένων όταν βρίσκονται σε shading απεικόνιση. Από τις προκαθορισμένες τιμές των επιμέρους ιδιοτήτων της ομάδας ιδιοτήτων Polygon Attributes μεταβλήθηκαν 2. Η πρώτη είναι η ιδιότητα 'cull face' της οποίας η τιμή μεταβλήθηκε από CULL_BACK σε CULL_NONE. Αυτή η ιδιότητα αφορά στην απόκρυψη η εμφάνιση επιφανειών των αντικειμένων ανάλογα με το βάθος που βρίσκονται πάνω στον viewer. Υπάρχει δυνατότητα να εμφανίζονται μόνον οι επιφάνειες που βρίσκονται σε πιο κοντινή απόσταση αποκρύπτοντας τις υπόλοιπες, ή να εμφανίζονται οι επιφάνειες που βρίσκονται σε πιο μακρινή απόσταση αποκρύπτοντας τις υπόλοιπες, ή τέλος να μην αποκρύπτεται καμία επιφάνεια. Η τελευταία επιλογή είναι αυτή που κρίθηκε ως πιο σωστή και αυτή επιλέξαμε. Να σημειωθεί ότι αυτή η ιδιότητα επηρεάζει όλες τις μορφές απεικόνισης (shading, wireframe, points). Η δεύτερη ιδιότητα που μεταβάλλαμε είναι η 'backFaceNormalFlip', στην οποία μεταβάλλαμε την τιμή από false σε true. Αυτή η ιδιότητα αφορά στην κατεύθυνση των κάθετων διανυσμάτων των επιφανειών ενός αντικειμένου που βρίσκονται σε μεγαλύτερο βάθος. Συγκεκριμένα η προεπιλεγμένη τιμή δίνει στα κάθετα διανύσματα φορά προς το εσωτερικό του όγκου έτσι ώστε να είναι παράλληλα με τα κάθετα διανύσματα των επιφανειών του αντικειμένου που βρίσκονται σε μικρότερο βάθος. Αυτή η φορά δεν μας ικανοποιεί διότι επηρεάζει μη ρεαλιστικά την αντανάκλαση του φωτός από τα αντικείμενα και για αυτό το λόγο τα αναστρέφουμε.

- Rendering Attributes (ιδιότητες απεικόνισης αντικειμένων): Οι ιδιότητες αυτές αφορούν κυρίως λειτουργίες που συνδέονται με το βάθος της απεικόνισης και με λειτουργίες που επιτρέπουν να επιλέγουμε αν ένα αντικείμενο θα είναι ορατό ή όχι κατά τη διάρκεια του rendering.

Σε αυτό το σημείο κρίνεται σκόπιμο να αναφέρουμε μία σημαντική λειτουργία που συνδέεται με το καθορισμό των ιδιοτήτων ενός αντικειμένου. Αυτή η λειτουργία σχετίζεται με την ικανότητα των ιδιοτήτων του αντικειμένου να διαφοροποιούνται ενώ έχει είδη γίνει μεταγλώττιση του προγράμματος. Δηλαδή όταν είναι αναγκαίο να επέμβουμε στις ιδιότητες ενός αντικειμένου ενώ έχει γίνει η μεταγλώττιση του προγράμματος και το αντικείμενο είναι 'ζωντανό' (**live**), πρέπει να εισάγουμε στο πρόγραμμα μας μια σειρά εντολών που έχουν την γενική ονομασία **Capabilities** αντικειμένου. Παραδείγματος χάριν για να αλλάξουμε την μορφή του αντικειμένου από shade σε wireframe (ορολογίες που εξηγήθηκαν παραπάνω) πρέπει να ορίσουμε να επιτρέπεται η αλλαγή της μορφής

εμφάνισης του αντικειμένου δηλαδή να εισάγουμε στο πρόγραμμα τα σχετικά με τις ιδιότητες πολυγώνου capabilities του αντικειμένου. Τα capabilities είναι ένα από τα πιο χρήσιμα εργαλεία στον τομέα των γραφικών, καθώς οι αλληλεπιδράσεις των δεδομένων εισόδου και της μεταβολής της τρισδιάστατης εικόνας είναι άμεσες και απαιτούν ρεαλισμό και αρκετά μεγάλη ταχύτητα. Για λόγους σιγουριάς είναι προτιμότερο να εισάγουμε σε κάθε αντικείμενο, όποιας μορφής και αν είναι, όλα τα δυνατά παρεχόμενα capabilities ώστε να μην αντιμετωπίσουμε πρόβλημα κατά την διαδικασία της απεικόνισης.

2.6. ΚΑΘΟΡΙΣΜΟΣ ΠΕΡΙΒΑΛΛΟΝΤΟΣ T4T VIEWER 3D-LIGHTS.

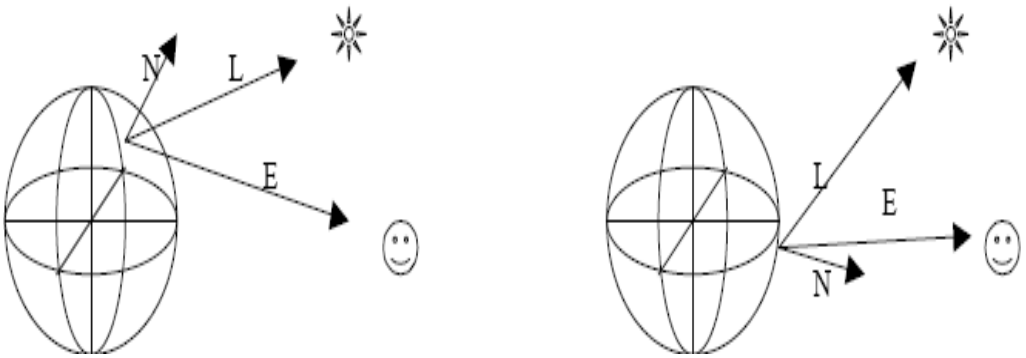
Οι βασικές απαιτήσεις για την λειτουργία ενός τρισδιάστατου viewer3d αξιολογήθηκαν, πριν ξεκινήσει η εφαρμογή της κατασκευής του καταλλήλου περιβάλλοντος και ταξινομήθηκαν ως εξής:

- Απαίτηση για ένα περιβάλλον υψηλής ευκρίνειας των αντικειμένων που παρουσιάζονται στον viewer.
- Απαίτηση κατάλληλου φωτισμού και σκίασης των αντικειμένων.
- Απαίτηση για ένα περιβάλλον εργασίας που δεν θα 'κουράζει' τα μάτια του χρήστη.
- Απαίτηση για ένα περιβάλλον εργασίας που θα κυμαίνεται σε παρόμοια πλαίσια με αυτά είδη γνωστών και ευρέως χρησιμοποιουμένων τρισδιάστατων viewers.

Για να πετύχουμε καλή ποιότητα απεικόνισης των αντικειμένων κατά την παρουσίαση τους πάνω στον viewer ήταν αναγκαίο να χρησιμοποιήσουμε όσον το δυνατόν περισσότερους και κατάλληλους φωτισμούς, που θα φωτίζουν το αντικείμενο. Οι φωτισμοί που χρησιμοποιήσαμε και η λειτουργία τους, καθώς και η λογική του φωτισμού των αντικειμένων στην java3d αναλύεται στη συνέχεια.

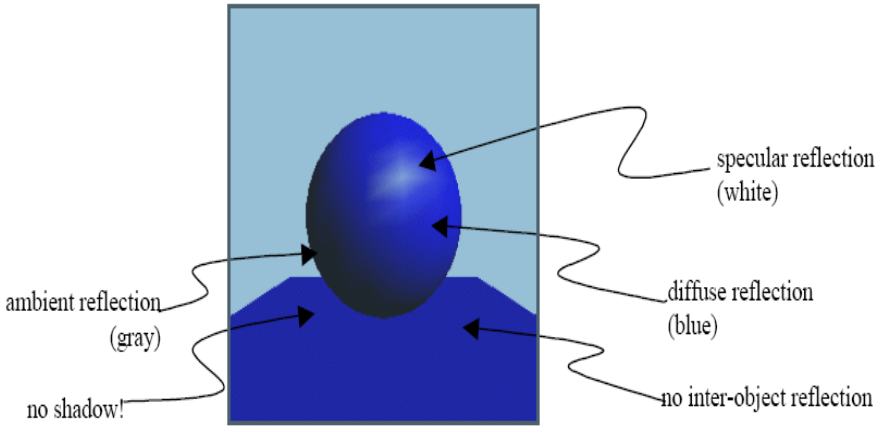
Στον πραγματικό κόσμο, τα χρώματα που αντιλαμβανόμαστε είναι ένας συνδυασμός των φυσικών ιδιοτήτων του αντικειμένου, των χαρακτηριστικών των πηγών φωτός, των σχετικών θέσεων αντικειμένων με τις πηγές φωτός, και των γωνιών από τις οποίες το αντικείμενο εμφανίζεται. Η Java3d API χρησιμοποιεί ένα μοντέλο φωτισμού για να προσεγγίσει το φυσικό φωτισμό του πραγματικού κόσμου. Η σκίαση των αντικειμένων βασίζεται στον συνδυασμό των ιδιοτήτων υλικού των αντικειμένων και των φωτισμών που υπάρχουν στον εικονικό χώρο. Η σκίαση προκύπτει εφαρμόζοντας ένα μοντέλο φωτισμού σε ένα οπτικό αντικείμενο, παρουσία πηγών φωτός. Η σκίαση των οπτικών αντικειμένων στην Java3d εξαρτάται από πολλούς παράγοντες. Αυτό το τμήμα παρέχει μια συνοπτική επισκόπηση του μοντέλου φωτισμού της Java3d, του μοντέλου χρωματισμού, και των μοντέλων σκίασης. Δεδομένου ότι ένα μεγάλο μέρος του μοντέλου φωτισμού και της σκίασης στη java3d είναι βασισμένο σε OpenGL, περισσότερες πληροφορίες μπορούν επίσης να βρεθούν στις αναφορές για OpenGL.

ΜΟΝΤΕΛΟ ΦΩΤΙΣΜΟΥ: **[1]** Στον πραγματικό κόσμο, τα χρώματα που αντιλαμβανόμαστε είναι ένας συνδυασμός των σωματικών ιδιοτήτων του αντικειμένου, των χαρακτηριστικών των πηγών φωτός, το σχετικών θέσεων αντικειμένων "στις πηγές φωτός, και της γωνίας από τις οποίες το αντικείμενο εμφανίζεται. Η Java3d χρησιμοποιεί ένα μοντέλο φωτισμού για να προσεγγίσει τη φυσική του πραγματικού κόσμου. Η πρότυπη εξίσωση φωτισμού εξαρτάται από τρία διανύσματα: το κάθετο στην επιφάνεια (N), η κατεύθυνση του φωτισμού (L) και η κατεύθυνση της οπτικής του viewer (E) εκτός από τις υλικές ιδιότητες του αντικειμένου και των χαρακτηριστικών του φωτισμού. Η εικόνα 2.16 εμφανίζει τα τρία διανύσματα για δύο πλευρές μιας σφαιρικής επιφάνειας. Τα διανύσματα για κάθε πλευρά μπορούν να έχουν διαφορετικές κατευθύνσεις, ανάλογα με τις ιδιότητες της σκηνής του εικονικού χώρου. Όταν τα διανύσματα (E) ποικίλλουν και υπολογίζονται κατά τη διάρκεια του rendering. Επομένως κάθε πλευρά της σφαίρας δίνει ενδεχομένως διαφορετική σκίαση.



Εικόνα 2.17.Χαρακτηριστικά διανύσματα εξίσωσης φωτισμού [1]

Το μοντέλο φωτισμού της java3d χρησιμοποίει τρία από τα είδη των πραγματικών αντανακλάσεων φωτισμών: ambient (περιβάλλον), diffuse (διάχυση), και specular (κατοπτρισμός). Η αντανάκλαση φωτισμού λόγω περιβάλλοντος προκύπτει από το φυσικό χρώμα του περιβάλλοντος χώρου και αφορά χαμηλά επίπεδα φωτισμού σε μια σκηνή. Η αντανάκλαση λόγω διάχυσης είναι η κανονική αντανάκλαση ενός οπτικού αντικείμενου σε μία πηγή φωτός. Η αντανάκλαση λόγω κατοπτρισμού αφορά στις υψηλές αντανακλάσεις ενός οπτικού αντικείμενου σε μία πηγή φωτός. Η αντανάκλαση προφανώς αυξάνει όσο πιο κάθετο είναι το διάνυσμα (L) στην επιφάνεια.



Εικόνα 2.18. Αντανακλάσεις φωτισμού στην Java3d (ambient,diffuse,specular) [1]

ΜΟΝΤΕΛΟ ΧΡΩΜΑΤΙΣΜΟΥ: **[1]** Το μοντέλο χρωματισμού δεν βασίζεται στη φυσική διάσταση των χρωμάτων. Η Java3d απεικονίζει το χρώμα των φώτων και των υλικών ως συνδυασμό κόκκινου, πράσινου, και μπλε. Το λευκό χρώμα, είτε ως φως είτε ως υλικό χρώμα, είναι ο συνδυασμός και των τριών συστατικών στη μέγιστη ένταση. Κάθε φως παράγει ένα χρώμα, που προσδιορίζεται από ένα, όπως συνηθίζουμε να αναφέρουμε, RGB(red-green-blue) συνδυασμό. Το μοντέλο φωτισμού βασίζεται στον RGB χρωματισμό του φωτός σε συνδυασμό με αυτόν του αντικειμένου. Παραδείγματος χάριν, μια κόκκινη σφαίρα παρουσία ενός μπλε φωτός δεν θα είναι ορατή δεδομένου ότι το μπλε φως δεν θα αντανακλαστεί από ένα κόκκινο αντικείμενο. Στην πραγματικότητα, το χρώμα είναι ένας συνδυασμός πολλών μηκών κύματος του φωτός, όχι μόνο τρία. Το RGB μοντέλο χρώματος αντιπροσωπεύει πολλά χρώματα, αλλά όχι όλα.

ΜΟΝΤΕΛΟ ΣΚΙΑΣΗΣ: **[1]** Η σκίαση ενός αντικειμένου είναι αποτέλεσμα της σκίασης κάθε πλευράς του αντικειμένου και προκύπτει από το άθροισμα των επιμέρους σκιάσεων που προκαλεί κάθε μία πηγή φωτός.

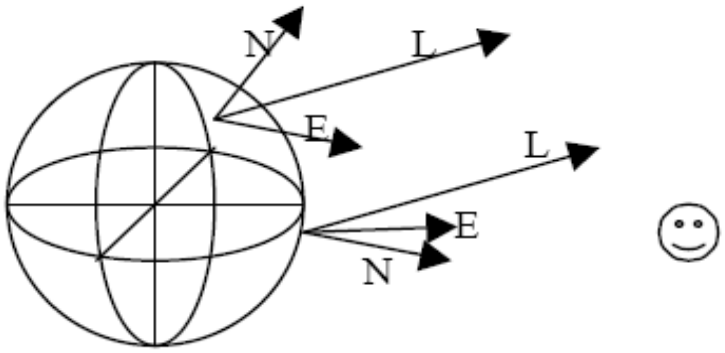
Τα είδη φωτισμών που μπορούμε να χρησιμοποιήσουμε για τον φωτισμό του εικονικού χώρου είναι τα εξής:

- **AMBIENT**

Ο φωτισμός ambient προκύπτει από την αντανάκλαση του αντικειμένου λόγω ύπαρξης άλλων αντικειμένων στο χώρο ή λόγω του περιβάλλοντος(χωρίς φωτεινή πηγή) ή συνδυασμό των δυο. Πρόκειται για χαμηλού επιπέδου αντανάκλαση και άρα ο φωτισμός είναι χαμηλής έντασης, ίδιος προς όλες τις κατευθύνσεις και σε όλες τις θέσεις του χώρου. Φανταστείτε το κάτω μέρος ενός θρανίου που βρίσκεται κάτω από μία λάμπα. Η κάτω επιφάνεια αντανακλά φως ενώ δεν φωτίζεται. Αυτό συμβαίνει λόγω της αλληλεπίδρασης με τα υπόλοιπα αντικείμενα.

- **DIRECTIONAL LIGHT**

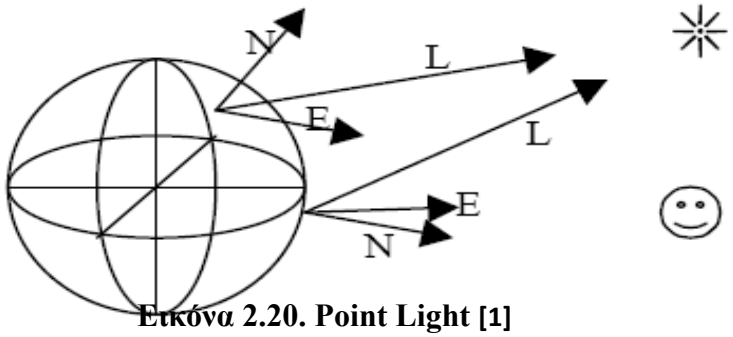
Ο φωτισμός directional προσεγγίζει πολύ μακρινές φωτεινές πηγές όπως ο ήλιος, ο φακός κ.α. Ο φωτισμός αυτός έχει συγκεκριμένη κατεύθυνση και μοναδική, αλλά όχι συγκεκριμένη τοποθεσία καθώς η ένταση του φωτισμού δεν επηρεάζεται από την απόσταση μεταξύ της φωτεινής πηγής και του αντικειμένου. Στην εικόνα 2.18 παρουσιάζεται η συμπεριφορά των διανυσμάτων φωτισμού για 2 διαφορετικές πλευρές του αντικειμένου. Τα διανύσματα (L) είναι πάντα παράλληλα.



Εικόνα 2.19. Directional Light [1]

- **POINT LIGHT**

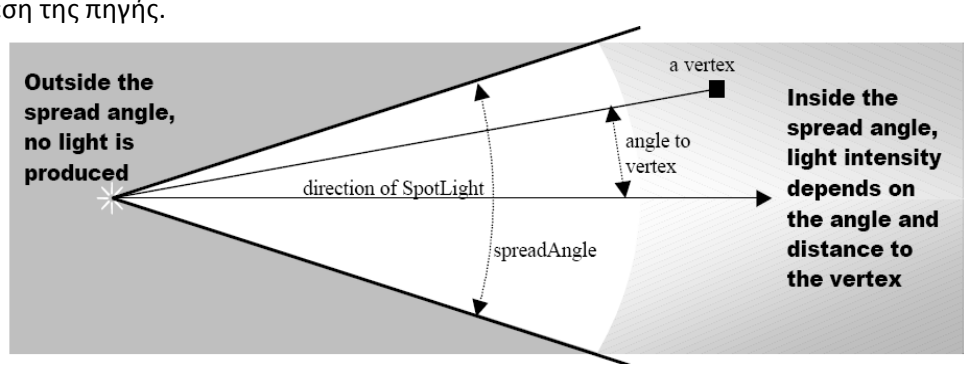
Ο φωτισμός Point είναι το ακριβώς αντίθετο του φωτισμού directional. Διαχέεται προς όλες τις κατευθύνσεις και η ένταση του επηρεάζεται από την απόσταση μεταξύ αντικειμένου και φωτεινής πηγής. Προσεγγίζει το φως που παράγουν πηγές όπως το κερί, οι λάμπες κ.α. Στην εικόνα 2.19 παρουσιάζεται η συμπεριφορά των διανυσμάτων φωτισμού για δυο διαφορετικές πλευρές του αντικειμένου. Τα διανύσματα (L) δεν είναι παράλληλα.



Εικόνα 2.20. Point Light [1]

SPOT LIGHT

Αποτελεί μια παραλλαγή του point light καθώς έχει τα ίδια χαρακτηριστικά, με τη διαφορά ότι ο χρήστης μπορεί να ρυθμίσει την ένταση του φωτός καθώς και το φάσμα μεγέθους της επιρροής του σχετικά με τη θέση του. Στην εικόνα 2.20 παρουσιάζονται οι παράμετροι που μπορούμε να ρυθμίσουμε, που είναι η γωνία (άνοιγμα της δέσμης φωτός) και η θέση της πηγής.



Εικόνα 2.20. Spot Light [1]

Στο viewer3d χρησιμοποιήσαμε 7 φωτισμούς με τους 4 να είναι τύπου directional light και οι 2 τύπου point light, καθώς και ένα φωτισμό ambient. Η λογική της επιλογής των φωτισμών ήταν ‘όσο πιο φωτεινό τόσο πιο καλό’. Οι φωτισμοί τοποθετήθηκαν με τρόπο τέτοιο ώστε να περιβάλουν το αντικείμενο, ενώ το χρώμα που επιλέξαμε για όλους είναι το λευκό για να μην δημιουργούνται χρωματικές αλλοιώσεις.

Μία άλλη παράμετρος που επηρεάζει το περιβάλλον του viewer είναι ο χρωματισμός του φόντου. Η επιφάνεια αυτή είναι μία πρόσθετη πάνω στο JPanel επιφάνεια που είναι υποχρεωτικό να εισάγουμε για να κατασκευάσουμε εικονικούς χώρους. Στην java η επιφάνεια αυτή καλείται καμβάς(canvas). Το χρώμα του καμβά επιλέχτηκε με κριτήρια την ευκρίνεια των αντικειμένων σε όλους τους πιθανούς χρωματισμούς που παρέχονται από τον viewer αλλά και την ανάγκη για ένα ήπιο χρώμα για να μην κουράζει τα μάτια του χρήστη. Επίσης μία παράμετρος που μας ενδιέφερε κατά την επιλογή ήταν να μην διαφέρει από τα είδη ευρέως χρησιμοποιούμενα προγράμματα, ώστε να είναι οικείο το περιβάλλον στο χρήστη.

3

ΑΝΑΛΥΣΗ ΚΩΔΙΚΑ ΤΟΥ VIEWER3D

3.1 ΕΙΣΑΓΩΓΗ

Στο κεφάλαιο αυτό δίδονται προγραμματιστικού περιεχομένου πληροφορίες για το λογισμικό που αναπτύχθηκε. Σημειώνεται ότι οι πληροφορίες στο διαδίκτυο είναι συχνά συγκεχυμένες, ιδιαίτερα στο κομμάτι των παραδειγμάτων, ενώ περιορίζονται συνήθως σε απλές εφαρμογές. Το Java3d API αποτελεί ένα πρωτοποριακό, ευέλικτο και γρήγορο API, ιδανικό για την δημιουργία γραφικών. Ο viewer3d περιλαμβάνει περίπου 5000 γραμμές κώδικα, το οποίο είναι ιδιαίτερα μικρό μέγεθος, αποτέλεσμα που προέκυψε λόγω της ευελιξίας της java3d. Στη συνέχεια θα δοθεί αναλυτικά η διαδικασία που ακολουθήθηκε για την ανάπτυξη του λογισμικού.

Μια νέα εφαρμογή ξεκινάει με την δημιουργία ενός πακέτου διάφορων στοιχείων που ονομάζεται project. Στο project εμπεριέχονται όλες οι κλάσεις που θα δημιουργήσουμε, καθώς και όλες οι βιβλιοθήκες που χρησιμοποιούνται και αφορούν έτοιμες κλάσεις. Κάθε νέα βιβλιοθήκη που θέλουμε να χρησιμοποιήσουμε πρέπει να την εισάγουμε στο project (load). Για την δημιουργία κάθε νέας κλάσης επιλέγουμε το project στο οποίο θέλουμε να την αποθηκεύσουμε και κάνοντας αριστερό κλίκ εμφανίζονται οι πιθανές επιλογές που έχουμε. Εμείς επιλέγουμε την εντολή **New** και από τις επιλογές που εμφανίζονται επιλέγουμε **Java Class.** Ακολουθώντας αυτή την διαδικασία κατασκευάζουμε μία νέα κλάση, που ανήκει στο project από το οποίο δημιουργήθηκε και η οποία περιέχει το όνομα της κλάσης και μία συνάρτηση που έχει το όνομα της κλάσης χωρίς κανένα όρισμα. Αυτή η συνάρτηση καλείτε **constructor,** την σημασία της οποίας θα αναλύσουμε στη συνέχεια.

Σε πολλά σημεία της εργασίας έγινε αναφορά στον όρο constructor. Οι constructors αποτελούν συναρτήσεις που έχουν το ίδιο όνομα με την κλάση στην οποία ανήκουν. Στις συναρτήσεις αυτές κατασκευάζεται το αντικείμενο της κλάσης. Όπως παρατηρήσαμε σε όλες τις κλάσεις του προγράμματος τα αντικείμενα των κλάσεων δημιουργήθηκαν μέσα στους constructor, είτε άμεσα είτε με την κλήση κάποιων κατάλληλων μεθόδων. Κάθε κλάση είναι υποχρεωτικό να περιέχει έναν constructor ακόμα και αν αυτός είναι κενός. Κάθε κλάση μπορεί να έχει και παραπάνω από έναν constructor ποτέ όμως με τα ίδια ορίσματα. Επίσης στους constructors καθορίζεται το μέγεθος της μνήμης που καταλαμβάνει η κλάση.

Στην εφαρμογή δημιουργήθηκαν 8 κλάσεις, τα ονόματα των οποίων παρουσιάζονται στη συνέχεια:

- Viewer3d.java
- HubClass.java
- ShroudClass.java
- BladesClass.java
- AllBladesClass.java
- Axis.java
- LoaderClass.java
- V3D.java

Με τον ίδιο τρόπο που περιγράψαμε για τις κλάσεις εργαστήκαμε και για την δημιουργία του παραθύρου JFrame, που αναλύσαμε στο κεφάλαιο 2, με την διαφορά ότι αντί για την επιλογή Java Class επιλέγουμε την JFrame Form. Το JFrame το ονομάσαμε V3D. Ουσιαστικά πρόκειται για προκαθορισμένη κλάση από το NetBeans σε μορφή αντικειμένου JFrame. Κάθε κλάση μπορεί να περιέχει μια μέθοδο που καλείτε main() η οποία κάνει την κλάση ανεξάρτητη κατά την εκτέλεση του προγράμματος ή να καλείται από μία άλλη κλάση που περιέχει main() μέθοδο, καθιστώντας την εξαρτημένη. Σε κάθε εφαρμογή πρέπει να υπάρχει μια τουλάχιστον μέθοδος main() για να είναι δυνατή η εκτέλεση του προγράμματος. Στην εφαρμογή μας η main() υπάρχει στον κώδικα του JFrame, από το οποίο καλούνται άμεσα ή έμμεσα όλες οι άλλες κλάσεις. Κάθε κλάση είναι υπεύθυνη για την δημιουργία ενός αντικειμένου, το κάθε ένα από τα οποία αποτελεί κομμάτι του συνόλου του viewer3d. Θα αναλύσουμε παρακάτω κάθε μία κλάση και τα αντικείμενα που δημιουργούν.

3.2. Viewer3d.java

Viewer3d.java: Η κλάση αυτή είναι η κεντρική κλάση του προγράμματος. Είναι η κλάση η οποία καλεί όλες της υπόλοιπες χρησιμοποιώντας τα αντικείμενα που δημιουργούν και δίνοντας όλες τις επιθυμητές ιδιότητες σε κάθε ένα από αυτά, δημιουργεί όλα τα απαραίτητα «σκηνικά». Καλείτε με την σειρά της από το παράθυρο V3D, που περιέχει την main(). Συγκεκριμένα στην κλάση αυτή εκτελούνται οι ακόλουθες εργασίες:

- Δημιουργία του εικονικού χώρου(virtual universe).
- Δημιουργία του αρχικού αντικειμένου τύπου Branchgroup, που θα εμπεριέχει τα τριδιάστατα προς απεικόνιση αντικείμενα.
- Δημιουργία των τριδιάστατων αντικειμένων, χρησιμοποιώντας τις ιδιότητες που κατασκευάζονται στις υπόλοιπες κλάσεις.
- Δημιουργία αντικειμένων ιδιοτήτων.
- Ρύθμιση των τιμών των ιδιοτήτων για κάθε ένα από τα αντικείμενα.
- Δημιουργία αντικειμένου εμφάνισης.
- Αποθήκευση των ιδιοτήτων στο αντικείμενο εμφάνισης.
- Αποθήκευση της εμφάνισης στα τρισδιάστατα αντικείμενα.
- Δημιουργία αντικειμένων υφής.
- Ρύθμιση ιδιοτήτων των αντικειμένων υφής .
- Αποθήκευση των τιμών της υφής στα τριδιάστατα αντικείμενα.
- Δημιουργία αντικειμένων φωτισμού.
- Ρύθμιση ιδιοτήτων αντικειμένων φωτισμού.
- Αποθήκευση φωτισμών στο αρχικό Branchgroup αντικείμενο.
- Δημιουργία αλληλεπίδρασης zoom, rotate, translate του viewer3d και του ποντικιού του H/Y.

- Αποθήκευση της αλληλεπίδρασης στο αρχικό Branchgroup αντικείμενο.
- Δημιουργία αντικειμένου φόντου.
- Ρύθμιση ιδιοτήτων του αντικειμένου φόντου.
- Αποθήκευση του αντικειμένου φόντου στο αρχικό Branchgroup αντικείμενο.
- Αποθήκευση όλων των τρισδιάστατων αντικειμένων στο αρχικό Branchgroup αντικείμενο.
- Δημιουργία όλων των μεθόδων ελέγχου του viewer3d.

Ότι δημιουργείται σε αυτή την κλάση απεικονίζεται πάνω στο JPanel του παραθύρου JFrame V3D, της βάσης δηλαδή του viewer3d. Για το λόγο αυτό η κλάση επιλέγεται να αναγνωρίζεται από τον compiler ως ένα JPanel. Αυτή η ιδιότητα για μία κλάση μπορεί να επιτευχθεί όταν στην αρχή της κλάσης προσθέσουμε την εντολή **extends** την οποία αναλύσαμε στο κεφάλαιο 1**.** Με τον τρόπο αυτό η κλάση κληρονομεί όλες τις μεθόδους της προκαθορισμένης κλάσης από την Java3D API JPanel και δημιουργεί πλέον ένα τέτοιο αντικείμενο:

public class Viewer3d extends JPanel {

Με αυτό τον τρόπο διευκολύνεται η αλληλεπίδραση της κλάσης με το JFrame V3D.

Η δημιουργία του αντικειμένου της κλάσης 'εικονικός χώρος και σκηνικά' γίνεται στον **constructor** της κλάσης, που αναπτύσσεται η δομή κώδικα που αναφέραμε ως scene graph. Εδώ πρέπει να σημειώσουμε ότι η κλάση χρησιμοποιεί 2 constructors, όπου ο ένας χρησιμοποιείται για τη δημιουργία αντικειμένων κλάσης που έχουν σχεδιαστεί παραμετρικά στο T4T και ο άλλος για αντικείμενα που τα χαρακτηριστικά τους προέρχονται εξωτερικά από αρχείο εισόδου txt. Ο πρώτος constructor έχει ορίσματα τις γεωμετρίες που προέρχονται από την παραμετρική σχεδίαση στο T4T και τον αριθμό των πτερυγίων κάθε πτερύγωσης. Ο λόγος που έχει αυτά τα ορίσματα θα αναλυθεί σε επόμενη ενότητα. Ο δεύτερος constructor δεν έχει ορίσματα είναι όπως συνηθίζεται να αποκαλείτε **κενός**. Ο τρόπος δημιουργίας του εικονικού χώρου είναι συγκεκριμένος και κατά κάποιο τρόπο τυποποιημένος. Συγκεκριμένα δημιουργείται αρχικά ένα αντικείμενο GraphicsConfiguration μέσω της εντολής:

GraphicsConfiguration config =SimpleUniverse.getPreferredConfiguration();

Έπειτα δημιουργούμε τον canvas που αναφέραμε στο κεφάλαιο 2. Ο canvas για να δημιουργηθεί χρησιμοποιεί ένα αντικείμενο GraphicsConfiguration, όπου και χρησιμοποιούμε αυτό που δημιουργήσαμε παραπάνω.

canvas3D = new Canvas3D(config);

Για να ρυθμίσουμε τις ιδιότητες του canvas χρησιμοποιούμε τις εντολές **set** και **get,** δηλαδή για παράδειγμα για να ρυθμίσουμε το χρώμα του background του canvas χρησιμοποιούμε την εντολή canvas3D.setBackground(Color). Έπειτα δημιουργούμε ένα αντικείμενο SimpleUniverse. Στο αντικείμενο αυτό αποθηκεύουμε τον canvas που έχουμε είδη δημιουργήσει.

SimpleUniverse simpleU = new SimpleUniverse(canvas3D);

Ρυθμίζουμε τις ιδιότητες του αντικειμένου SimpleUniverse με τις εντολές get, set. Στη συνέχεια δημιουργούμε το αρχικό αντικείμενο Branchgroup, το οποίο περιέχει όλα τα τρισδιάστατα προς απεικόνιση αντικείμενα.

BranchGroup scene = createSceneGraph();

Για να το δημιουργήσουμε καλούμε στον constructor την μέθοδο **createSceneGraph()**. Η δομή του κώδικα που αναπτύσσεται εντός της μεθόδου είναι η δομή που αναφέραμε στο κεφάλαιο 1 ως branch graph. Η μέθοδος αυτή ονομάστηκε έτσι καθώς έχει προγραμματιστεί για να κατασκευάζει όλα τα 'σκηνικά' του viewer3d, δηλαδή τα τριδιάστατα αντικείμενα με όλες τις ιδιότητες που επιθυμούμε, να τα αποθηκεύει σε ένα αντικείμενο τύπου BranchGroup και στη συνέχεια να επιστρέφει αυτό το αντικείμενο ώστε να αποθηκευτεί στον εικονικό χώρο. Στην μέθοδο αυτή προγραμματίσαμε τις ακόλουθες εργασίες:

- Δημιουργία αντικειμένου BranchGroup, του αρχικού δηλαδή αντικειμένου του εικονικού χώρου και στο οποίο αποθηκεύονται όλα τα υπόλοιπα αντικείμενα. Εδώ το ονομάσαμε objRoot.

```
objRoot = new BranchGroup();
```

- Αποθήκευση αντικειμένων που δημιουργούνται στις υπόλοιπες κλάσεις. Παράδειγμα το shroudshape είναι αντικείμενο τύπου κλάσης(εδώ ShroudClass) και στο οποίο αποθηκεύονται όλα τα στοιχεία (μέθοδοι, γεωμετρίες, μεταβλητές) που δημιουργούνται στον constructor κλάσης ShroudClass. Η μεταβλητή shroud είναι μεταβλητή τύπου γεωμετρίας και χρησιμοποιείτε για να αποθηκεύσουμε την γεωμετρία που κατασκευάζεται στη κλάση ShroudClass. Η σχέση που ακολουθείται μεταξύ αντικειμένου γεωμετρίας και αντικειμένου κλάσης είναι η **reference.**

```
shroudshape=new ShroudClass(geometries);
shroud=shroudshape.triangles;
```

- Εισαγωγή σε κάθε αντικείμενο γεωμετρίας που δημιουργούμε των κατάλληλων capabilities. Παράδειγμα εισαγωγή capability για την ικανότητα αποθήκευσης texture.

```
shroud.setCapability(shroud.ALLOW_TEXCOORD_READ);
```

- Δημιουργία αντικειμένων ιδιοτήτων και ρύθμιση επιθυμητών τιμών για κάθε ιδιότητα. Παράδειγμα δημιουργίας αντικειμένου ιδιότητας σημείων και ρύθμιση του μεγέθους των σημείων σε pixel.

```
PointAttributes point=new PointAttributes();
point.setPointSize(1);
```

- Δημιουργία αντικειμένου εμφάνισης και αποθήκευσης σε αυτό όλων των κατάλληλα ρυθμισμένων αντικειμένων ιδιοτήτων. Παράδειγμα η δημιουργία αντικειμένου εμφάνισης app και αποθήκευση σε αυτό του αντικειμένου τύπου PointAtrributes παραπάνω.

```
Appearance app = new Appearance();
app.setPointAttributes(point);
```

- Εισαγωγή capabilities στο αντικείμενο εμφάνισης. Παράδειγμα εισαγωγή capability που επιτρέπει σε ένα live αντικείμενο να αλλάξουν οι ρυθμίσεις στο αντικείμενο ιδιοτήτων χρωματισμού.

```
app.setCapability(app.ALLOW_COLORING_ATTRIBUTES_READ);
```

➢ Δημιουργία τριδιάστατων αντικειμένων, αποθήκευση του αντικειμένου εμφάνισης στα τρισδιάστατα αντικείμενα καθώς και ρύθμιση των capabilities για κάθε ένα από τα τριδιάστατα αντικείμενα. Παράδειγμα η δημιουργία τριδιάστατου αντικειμένου triangle3Dblade. Για τη δημιουργία του τριδιάστατου αντικειμένου χρησιμοποιείται το κατάλληλο αντικείμενο γεωμετρίας. Αποθηκεύουμε έπειτα το αντικείμενο εμφάνισης app και ρυθμίζουμε έπειτα το capability που επιτρέπει την αλλαγή ρυθμίσεων του αντικειμένου εμφάνισης όταν το τριδιάστατο αντικείμενο είναι σε κατάσταση live.

```
triangle3Dblade=new Shape3D(blade);
triangle3Dblade.setAppearance(app);

triangle3Dblade.setCapability(triangle3Dblade.ALLOW_APPEARANCE_READ);
```

➢ Δημιουργία αντικειμένου Texture2D, αποθήκευση της επιθυμητής εικόνας στο αντικείμενο και ρύθμιση όλων των επιθυμητών παραμέτρων. Για την αποθήκευση του επιθυμητού texture δημιουργούμε ένα αντικείμενο τύπου TextureLoader, που είναι το αντικείμενο στο οποίο αποθηκεύεται η διεύθυνση όπου το πρόγραμμα θα βρει την εικόνα, καθώς και το όνομα της εικόνας ώστε το πρόγραμμα να την αναγνωρίσει. Έπειτα δημιουργούμε ένα αντικείμενο τύπου Image στο οποίο αποθηκεύεται η εικόνα που με την διεύθυνση και ονομασία που έχουν αποθηκευτεί στο αντικείμενο TextureLoader. Το Texture2D αποθηκεύει τελικά την εικόνα και χρησιμοποιείται ως υφή (texture). Η ρύθμιση παραμέτρων αφορά στον τρόπο επικάλυψης των αντικειμένων από το texture κ.α.

```
String dir = System.getProperty("user.dir");
String filename = "\\Images\\goldmetal.jpg";
TextureLoader txtloader=new TextureLoader(dir+filename,this);
ImageComponent2D image = txtloader.getImage();
Texture2D texture1=new Texture2D(Texture.BASE_LEVEL, Texture.RGBA,
image.getWidth(), image.getHeight());
```

➢ Δημιουργία αντικειμένων φωτισμού, αντικειμένου Background και αντικειμένων Text3D και ρύθμιση των παραμέτρων τους. Παράδειγμα η δημιουργία ενός αντικειμένου φωτισμού DirectionalLight και ρύθμιση χρώματος, δημιουργία φόντου (background) και ρύθμιση χρώματος και κειμένου Text3D “X” που το χρησιμοποιούμε σε κατάλληλη θέση ώστε να ονομαστεί ο x άξονας του (x,y,z) ορθοκανονικού συστήματος που χρησιμοποιούμε.

```
DirectionalLight lightD4 = new DirectionalLight();
lightD4.setColor(white);
Background back=new Background(0.8f,0.9f,0.9f);
Text3D X = new Text3D(font3D,"X" ,new Point3f(20.0f,0.0f,0.0f));
```

➢ Δημιουργία αντικειμένου TransformGroup και ρύθμιση των capabilities του. Όλα τα αντικείμενα χρειάζονται ένα ‘μεταφορικό μέσο’ για να τοποθετηθούν στον εικονικό χώρο. Το ρόλο αυτό έχει το αντικείμενο τύπου TransformGroup. Κάθε αντικείμενο που τοποθετείται μέσα στο group αποτελεί ένα παιδί του, σύμφωνα με την σχέση γονέα-παιδιού. Το group είναι το αντικείμενο που κατασκευάσαμε και ένα από τα capabilities που ρυθμίσαμε είναι να επιτρέπεται η ρύθμιση των παιδιών του όταν είναι σε κατάσταση live.

```
group=new TransformGroup();
group.setCapability(group.ALLOW_CHILDREN_READ);
```

➢ Δημιουργία αλληλεπίδρασης mouse-viewer3d. Παράδειγμα η δημιουργία αντικειμένου τύπου Mousezoom, που επιτρέπει τη μεγέθυνση ή σμίκρυνση των αντικειμένων κατά την απεικόνισή τους. Το group που εμφανίζεται είναι το παραπάνω αντικείμενο τύπου TransformGroup και είναι το αντικείμενο που μεταφέρει το αντικείμενο Mousezoom στον εικονικό χώρο. Η σχέση group και Mousezoom αντικειμένου είναι σχέση γονέα(group)-παιδιού(zoom).

MouseZoom zoom=new MouseZoom();
zoom.setTransformGroup(group);

➢ Εισαγωγή όλων των αντικειμένων στο αντικείμενο TransformGroup. Παράδειγμα η εισαγωγή του τρισδιάστατου αντικειμένου triangle3Dhub που αφορά το την πλήμνη της πτερύγωσης. Ακολουθείται πάντα η σχέση γονέα-παιδιού.

group.addChild(triangle3Dhub);

➢ Εισαγωγή του αντικειμένου TransformGroup στο αντικείμενο BranchGroup ακολουθώντας την σχέση γονέα-παιδιού.
objRoot.addChild(group);

➢ Επιστροφή αντικειμένου BranchGroup.

return objRoot;

Το αντικείμενο που επιστρέφει η μέθοδος createSceneGraph() αποθηκεύεται στον εικονικό χώρο στο.

simpleU.addBranchGraph(scene);

Το τελευταίο μέρος της κλάσης Viewer3D είναι η δημιουργία των μεθόδων ελέγχου του viewer3d. Πρόκειται για τις μεθόδους που κατασκευάσαμε ώστε να μπορούμε να καλύψουμε τις απαιτήσεις της λειτουργίας του viewer3d, όπως η ικανότητα wireframe απεικόνισης. Οι μέθοδοι είναι όλες τύπου void, δηλαδή δεν επιστρέφουν τίποτα και ενεργοποιούνται όταν ο χρήστης χρησιμοποιήσει κάποιο από τα εξαρτήματα του viewer3d. Συγκεκριμένα οι μέθοδοι είναι:

- ✓ Yactionperform(): Μέθοδος που περιστρέφει τα αντικείμενα που απεικονίζονται γύρω από τον y άξονα και κατά 45 μοίρες κάθε φορά που καλείται. Η μέθοδος καλείται όταν ο χρήστης πατήσει το Yrotation JButton. Η διαδικασία της περιστροφής γίνεται μέσω πολλαπλασιασμού κατάλληλων πινάκων περιστροφής.

- ✓ Xactionperform(): Μέθοδος που περιστρέφει τα αντικείμενα που απεικονίζονται γύρω από τον x άξονα και κατά 45 μοίρες κάθε φορά που καλείται. Η μέθοδος καλείται όταν ο χρήστης πατήσει το Xrotation JButton. Η διαδικασία της περιστροφής γίνεται μέσω πολλαπλασιασμού κατάλληλων πινάκων περιστροφής.

- ✓ Zactionperform(): Μέθοδος που περιστρέφει τα αντικείμενα που απεικονίζονται γύρω από τον z άξονα και κατά 45 μοίρες κάθε φορά που καλείται. Η μέθοδος καλείται όταν ο

χρήστης πατήσει το Zrotation JButton. Η διαδικασία της περιστροφής γίνεται μέσω πολλαπλασιασμού κατάλληλων πινάκων περιστροφής.

- ✓ Rootactionperform(): Μέθοδος που επιστρέφει τα αντικείμενα στην αρχική θέση απεικόνισης τους. Η μέθοδος καλείται όταν ο χρήστης πατήσει το Rootposition JButton. Η διαδικασία της περιστροφής στην αρχική θέση γίνεται μέσω πολλαπλασιασμού κατάλληλων πινάκων περιστροφής.

- ✓ WireFrameperform(): Μέθοδος που αλλάζει την απεικόνιση των αντικειμένων από όποια μορφή βρίσκονται σε WireFrame μορφή. Η μέθοδος καλείται όταν ο χρήστης επιλέξει το item wireframe του ViewOption JComboBox εξαρτήματος του viewer3d. Η διαδικασία αυτή επιτυγχάνεται μέσω της αλλαγής των ρυθμίσεων των ιδιοτήτων πολυγώνου σε μορφή γραμμών.

```
PolygonAttributes poly= app.getPolygonAttributes();
poly.setPolygonMode(poly.POLYGON_LINE);
```

- ✓ Pointperform(): Μέθοδος που αλλάζει την απεικόνιση των αντικειμένων από όποια μορφή βρίσκονται σε Points μορφή. Η μέθοδος καλείται όταν ο χρήστης επιλέξει το item points του ViewOption JComboBox εξαρτήματος του viewer3d. Η διαδικασία αυτή επιτυγχάνεται μέσω της αλλαγής των ρυθμίσεων των ιδιοτήτων πολυγώνου σε μορφή σημείων.

```
PolygonAttributes poly=app.getPolygonAttributes();
poly.setPolygonMode(poly.POLYGON_POINT);
```

- ✓ Shadingperform(): Μέθοδος που αλλάζει την απεικόνιση των αντικειμένων από όποια μορφή βρίσκονται σε shading μορφή. Η μέθοδος καλείται όταν ο χρήστης επιλέξει το item shading του ViewOption JComboBox εξαρτήματος του viewer3d. Η διαδικασία αυτή επιτυγχάνεται μέσω της αλλαγής των ρυθμίσεων των ιδιοτήτων πολυγώνου σε μορφή όγκου.

```
PolygonAttributes poly=app.getPolygonAttributes();
poly.setPolygonMode(poly.POLYGON_FILL);
```

- ✓ ChangeMaterial1(): Μέθοδος που αλλάζει την υφή του αντικειμένου σε iron. Η μέθοδος καλείται όταν ο χρήστης επιλέξει το item iron του MaterialOption JComboBox εξαρτήματος του viewer3d. Η διαδικασία αυτή επιτυγχάνεται μέσω της αλλαγής της διεύθυνσης και της ονομασίας της εικόνας που αποθηκεύουμε στο αντικείμενο TextureLoader.

```
String dir = System.getProperty("user.dir");

String filename = "\\Images\\iron.jpg";

TextureLoader txtloader=new TextureLoader(dir+filename,this);
```

- ✓ ObjectVisible(): Μέθοδος που όταν καλείται προκαλεί την απεικόνιση στον viewer3d του συνόλου των γεωμετριών του αντικειμένου. Η μέθοδος καλείται όταν ο χρήστης ενεργοποιεί το JCheckBox object εξάρτημα του viewer3d, ασχέτως αν κάποιο από τα

υπόλοιπα JCheckBoxes είναι ενεργό, ή όταν είναι ενεργά τα blades, hub, shroud JCheckBoxes ασχέτως αν το object JCheckBox είναι ενεργό.

- ✓ NothingVisible(): Μέθοδος που προκαλεί την αφαίρεση του αντικειμένου από τον viewer3d. Οι γεωμετρίες αφαιρούνται με την εντολή **removeAllGeometries()**. Η μέθοδος αυτή καλείται όταν ο χρήστης απενεργοποιήσει όλα τα εξαρτήματα JCheckBoxes.

- ✓ AddBladesToObject(): Μέθοδος που προκαλεί την απεικόνιση της γεωμετρίας των πτερυγίων στις είδη υπάρχουσες γεωμετρίες. Αυτό θα συμβεί όταν ο χρήστης ενεργοποιήσει το JCheckBox blades εξάρτημα του viewer3d.

- ✓ AddHubToObject(): Μέθοδος προκαλεί την απεικόνιση της γεωμετρίας της πλήμνης στις είδη υπάρχουσες γεωμετρίες. Αυτό θα συμβεί όταν ο χρήστης ενεργοποιήσει το JCheckBox hub εξάρτημα του viewer3d.

- ✓ AddShroudToObject(): Μέθοδος που προκαλεί την απεικόνιση της γεωμετρίας του κελύφους στις είδη υπάρχυσες γεωμετρίες. Αυτό θα συμβεί όταν ο χρήστης ενεργοποιήσει το JCheckBox shroud εξάρτημα του viewer3d.

- ✓ RemoveBlades(): Μέθοδος που προκαλεί την αφαίρεση της γεωμετρίας των πτερυγίων από τις είδη υπάρχουσες γεωμετρίες. Αυτό θα συμβεί όταν ο χρήστης απενεργοποιήσει το JCheckBox blades εξάρτημα του viewer3d.

- ✓ RemoveHub(): Μέθοδος που προκαλεί την αφαίρεση της γεωμετρίας της πλήμνης από τις είδη υπάρχουσες γεωμετρίες. Αυτό θα συμβεί όταν ο χρήστης απενεργοποιήσει το JCheckBox hub εξάρτημα του viewer3d.

- ✓ RemoveShroud(): Μέθοδος που προκαλεί την αφαίρεση της γεωμετρίας του κελύφους από τις είδη υπάρχουσες γεωμετρίες. Αυτό θα συμβεί όταν ο χρήστης απενεργοποιήσει το JCheckBox shroud εξάρτημα του viewer3d.

3.3. Axis.java

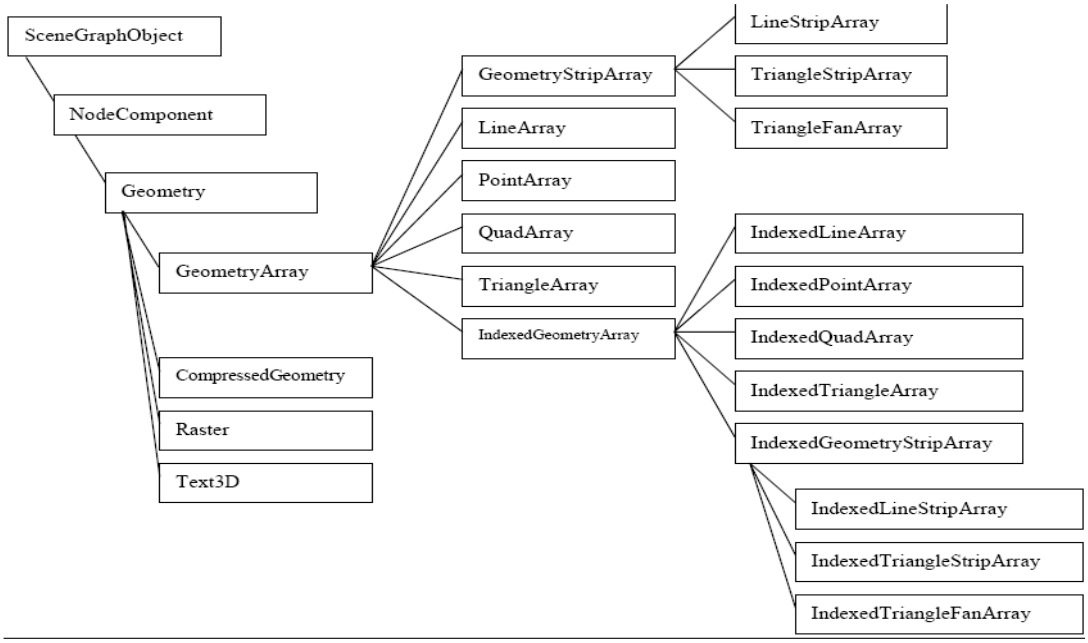
Η κλάση Axis δημιουργεί το αντικείμενο «άξονες». Το σύστημα αξόνων που χρησιμοποιείται είναι καρτεσιανό ορθοκανονικό σύστημα αξόνων (x,y,z). Οι άξονες που εμφανίζονται στον viewer3d είναι ένα τρισδιάστατο αντικείμενο και για τον λόγο αυτό ο compiler πρέπει να αντιλαμβάνεται την κλάση αυτή, και ότι δημιουργεί, σαν ένα τρισδιάστατο αντικείμενο. Ομοίως με την κλάση viewer3d χρησιμοποιούμε την εντολή extends, όπως παρουσιάζεται παρακάτω.

public class Axis extends Shape3D{

Το Shape3D είναι το τρισδιάστατο αντικείμενο στην Java3D , είναι δηλαδή ένας δείκτης σε προκαθορισμένη από την βιβλιοθήκη κλάση που δημιουργεί το αντικείμενο ‘τρισδιάστατο αντικείμενο’.

Στον constructor της κλάσης δημιουργείται το αντικείμενο της κλάσης που είναι η γεωμετρία των αξόνων μέσω της κλήσης μίας μεθόδου που καλείται createGeometry().Η μέθοδος αυτή είναι τύπου κλάσης GeometryArray. Αυτό στην γλώσσα της Java3d σημαίνει ότι είναι μια μέθοδος που κατασκευάζει και επιστρέφει αντικείμενα τύπου γεωμετρίας. Ο

τύπος κλάσης GeometryArray έχει υποκλάσεις, όπου ο διαχωρισμός γίνεται με βάση τον τρόπο ένωσης των σημείων που κατασκευάζουν την γεωμετρία, δηλαδή τον τύπο του πλέγματος της γεωμετρίας. Οι υποκατηγορίες παρουσιάζονται στο σχήμα 3.1.



Σχήμα 3.1 *GeometryArray SubClasses.* **[1]**

Για παράδειγμα η υποκλάση TriangleArray αναφέρεται σε γεωμετρίες με τριγωνικό πλέγμα. Οι IndexedGeometryArray κλάσεις και υποκλάσεις είναι αυτές που χρησιμοποιούμε στην εφαρμογή μας.

Η μέθοδος αυτή χρησιμοποιεί ένα αντικείμενο τύπου IndexedLineArray, στο οποίο αποθηκεύει τα κατάλληλα στοιχεία ώστε να δημιουργηθεί η γεωμετρία που επιθυμούμε. Ο constructor της κλασης IndexedLineArray δέχεται ως ορίσματα τον αριθμό των σημείων που θα χρησιμοποιήσουμε, τον τύπο με τον οποίο θα δίνουμε τα σημεία ώστε να τα διαβάζει η συνάρτηση και τον αριθμό των σημείων ένωσης. Η μορφή της συνάρτησης είναι η παρακάτω.

IndexedLineArray axisLines=new IndexedLineArray(simeia,typos,enwseis)

Ο τρόπος που αποθηκεύουμε τα δεδομένα είναι ο εξής. Ρυθμίζουμε τα σημεία δίνοντας τους ένα δείκτη από το 0 μέχρι τον συνολικό αριθμό των σημείων, καθώς και τις συντεταγμένες x,y,z στο χώρο. Ο τρόπος που ρυθμίζονται παρουσιάζεται παρακάτω.

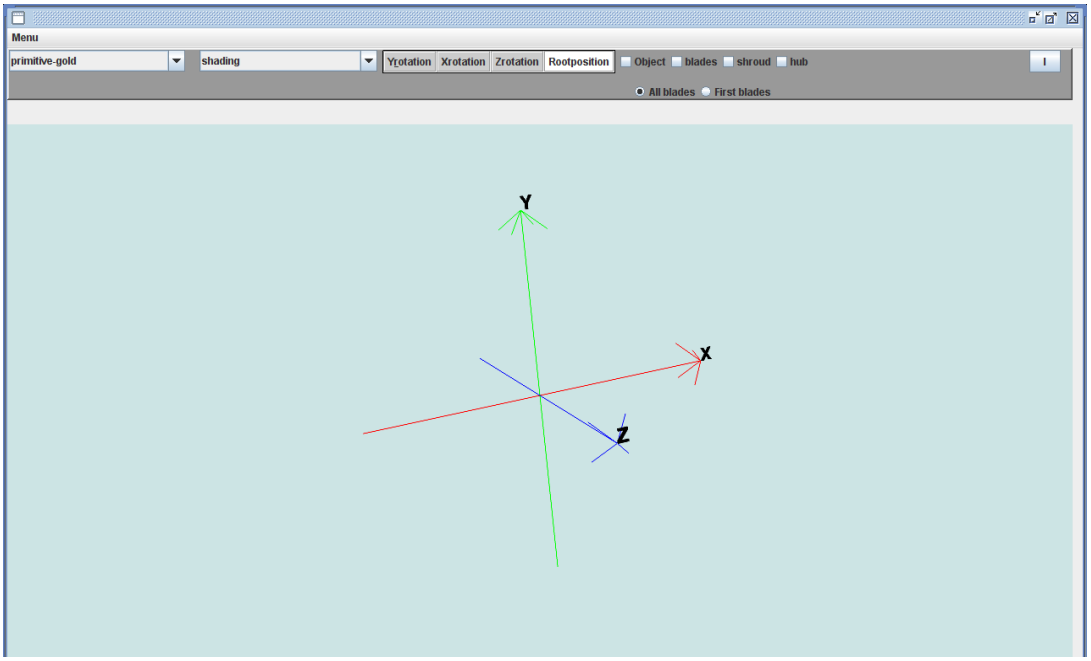
axisLines.setCoordinate(0, new Point3f(-1f, 0.0f, 0.0f))

Η εντολή που χρησιμοποιείτε για την ρύθμιση των συντεταγμένων είναι η setCoordinate. Στο παράδειγμα μας το 0 είναι ο δείκτης αυτού του σημείου που έχει συντεταγμένες x=-1 y=0 z=0. Το Point3f είναι το σημείο με συντεταγμένες x,y,z όπως ορίζεται στην Java3d είναι δηλαδή ένας δείκτης σε κλάση της βιβλιοθήκης που δημιουργεί το αντικείμενο 'σημείο στο χώρο'. Έπειτα ρυθμίζουμε με τη σειρά τον τρόπο που θα ενωθούν ανά 2 σχηματίζοντας ευθύγραμμα τμήματα. Η ρύθμιση αυτή γίνεται με την εντολή:

```
axisLines.setCoordinateIndex( 0, 0);
axisLines.setCoordinateIndex( 1, 1);
```

Σύμφωνα με το παράδειγμα η πρώτη ένωση αφορά τα σημεία ένωσης 0 και 1 που αντιστοιχούν στα σημεία με δείκτη 0 και 1. Αυτός ο τόπος ένωσης και κατασκευής της γεωμετρίας ονομάζεται **Index**. Κάθε δείκτης είναι μοναδικός και μπορεί να αντιστοιχεί μόνο σε ένα σημείο. Κάθε σημείο όμως μπορεί να αντιστοιχεί σε πολλά διαφορετικά σημεία ένωσης. Από αυτή την παρατήρηση προκύπτει ότι τα σημεία ένωσης είναι πάντα ίδια ή περισσότερα από τα σημεία δείκτες. Για να διακρίνουμε καλύτερα τους άξονες χρησιμοποιήσαμε χρωματική διαφοροποίηση. Αυτή η ρύθμιση επιτυγχάνεται χρησιμοποιώντας την εντολή setColor για το χρωματισμό κάθε σημείου και την εντολή setColorIndex για το χρωματισμό των ενώσεων. Η διαδικασία είναι όμοια της ρύθμισης των σημείων και των ενώσεων.

Εργαζόμενοι όπως περιγράψαμε παραπάνω δημιουργήσαμε το αντικείμενο «άξονες», που παρουσιάζεται στην εικόνα 3.1.



Εικόνα 3.1. Αντικείμενο «άξονες»

Οι προσθέσεις των ονομάτων τω αξόνων κατασκευάστηκαν στην κλάση Viewer3d που ρυθμίζονται όλες οι ιδιότητες των αντικειμένων. Οι άξονες είναι ένα τριδιάστατο αντικείμενο που εμφανίζεται στον viewer3d οπότε έχει όλες τις δυνατότητες όπως οποιοδήποτε άλλο αντικείμενο κατασκευάζουμε, όπως δυνατότητα περιστροφής, zoom in zoom out κ.α. Στην εικόνα 3.1 οι άξονες έχουν περιστραφεί και έχει γίνει zoom out για καλύτερη απεικόνιση. Η θέση (0,0,0) βρίσκεται στο κέντρο της οθόνης και η απόσταση από το κέντρο του JPanel προς κάποια γωνία του έχει τιμή ίση με 1. Δηλαδή το σημείο x=1,y=-1,z=0 είναι η κάτω δεξιά γωνία του JPanel.

3.4. LoaderClass.java

Η κλάση αυτή δημιουργήθηκε για να δώσει μία επιπλέον χρήσιμη ιδιότητα στον viewer3d. Ενώ ο λόγος κατασκευής του viewer ήταν η ανάγκη να απεικονίζονται σε τριδιάστατη μορφή αντικείμενα και συγκεκριμένα πτερυγώσεις που σχεδιάζονται παραμετρικά στο T4T, θεωρήθηκε χρήσιμο να μπορεί να λειτουργεί και εξωτερικά από το πρόγραμμα T4T, δηλαδή να μπορεί κάποιος χρήστης που έχει κατασκευάσει ένα κατάλληλο αρχείο σημείων και ενώσεων των σημείων ενός αντικειμένου να απεικονίσει το αντικείμενο σε τριδιάστατη μορφή στον viewer3d. Για να συμβεί αυτό πρέπει το αντίστοιχο αρχείο να έχει την μορφή που περιγράψαμε στην ενότητα 2.2.

Η κλάση χρησιμοποιεί έναν constructor χωρίς όρισμα, ο οποίος δημιουργεί το αντικείμενο της κλάσης καλώντας 2 μεθόδους, την **callCoordinateData()** και την **createShape3d(coords,connections).** Η πρώτη μέθοδος είναι τύπου void ενώ η δεύτερη είναι τύπου κλάσης IndexedTriangleArray, που είναι υποκλάση (βλ.σχ.3.1.) της κλάσης IndexedGeometryArray. Αυτό σημαίνει ότι η μέθοδος αυτή επιστρέφει γεωμετρίες που κατασκευάζονται χρησιμοποιώντας τριγωνικό πλέγμα με index τρόπο κατασκευής. Η callCoordinateData() δεν έχει καθόλου ορίσματα ενώ η createShape3d() έχει 2 ορίσματα, που θα τα αναλύσουμε παρακάτω.

Η callCoordinateData() είναι η μέθοδος που κατασκευάσαμε ώστε να είναι δυνατή η ανάγνωση του αρχείου txt από τον viewer. Μέσα στην μέθοδο δίνουμε την διεύθυνση όπου ο compiler θα βρει το επιθυμητό αρχείο. Μια τέτοια διεύθυνση(\\export_files) παρουσιάζεται παρακάτω, καθώς και ο τρόπος αποθήκευσης του αρχείου.

```
String dir = System.getProperty("user.dir")
String path=\\export_files
File input=new File(dir + path,"points_cloud.txt")
```

Εφόσον αποθηκευτεί το αρχείο γίνεται η κατάλληλη επεξεργασία για να αποθηκευτούν κατάλληλα τα δεδομένα που μας δίνει. Με εντολές που διαγράφουν τα κενά και τα σχόλια που αποτελούν άχρηστη πληροφορία και με τη αποθήκευση των σημείων και των ενώσεων σε κατάλληλους πίνακες τελειώνει η λειτουργία της μεθόδου αυτής. Οι πίνακες που δημιουργούνται είναι οι coords και connections. Ο πίνακας coords είναι ένας πίνακας με μία στήλη και γραμμές ίσες με τον αριθμό των σημείων του νέφους σημείων της γεωμετρίας. Ο τύπος του πίνακα είναι Point3d. Αυτό σημαίνει ότι στον πίνακα αποθηκεύονται στοιχεία που είναι αντικείμενα της κλάσης Point3d. Η κλάση αυτή δημιουργεί αντικείμενα 'σημεία στο χώρο' δηλαδή σημεία με συντεταγμένες x,y,z. Η διαφορά των αντικειμένων Point3f και Point3d αφορά στην ακρίβεια της τιμής της κάθε συντεταγμένης καθώς τα Point3f αντικείμενα χρησιμοποιούν αριθμούς απλής ακρίβειας και τα Point3d αντικείμενα διπλής ακρίβειας. Με αυτό τον τρόπο αποθηκεύουμε σε κάθε γραμμή του πίνακα coords ένα σημείο του νέφους σημείων σε μορφή συντεταγμένων (x,y,z). Ο τρόπος δημιουργίας του παρουσιάζεται παρακάτω

```
Point3d coords=new Point3d[simeia]
```

Ο πίνακας connections είναι ο πίνακας που χρησιμοποιούμε για να αποθηκεύσουμε τις ενώσεις των σημείων. Η διάσταση του πίνακα είναι γραμμές ίσες με τα τρίγωνα του τριγωνικού πλέγματος και 3 στήλες. Με αυτό τον τρόπο αποθηκεύεται σε κάθε γραμμή μια τριάδα αριθμών που αντιστοιχούν σε δείκτες(index) των σημείων του νέφους σημείων. Ο τρόπος που δημιουργούμε τον πίνακα παρουσιάζεται παρακάτω.

```
Int [][] connections=new int[trigwna][3]
```

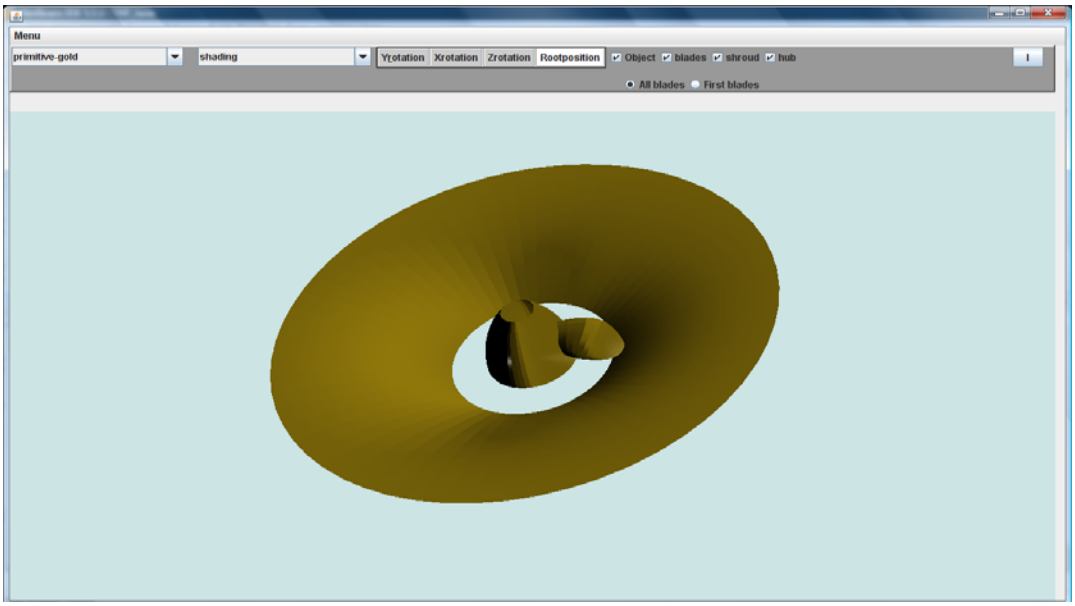
Όταν κατασκευαστούν οι πίνακες καλείται η μέθοδος createShape3d() που δέχεται σαν ορίσματα αυτούς τους 2 πίνακες. Η μέθοδος αυτή χρησιμοποιεί τα στοιχεία που περιέχουν οι πίνακες για να κατασκευάσει την γεωμετρία του αντικειμένου. Η μέθοδος χρησιμοποιεί ένα αντικείμενο τύπου κλάσης IndexedTriangleArray. Ο constructor της κλάσης IndexedTriangleArray δέχεται ως ορίσματα αριθμό σημείων νέφους σημείων, τύπο σημείων, τα κάθετα διανύσματα στα τρίγωνα του πλέγματος σε μορφή συντεταγμένων (Point3d), και αριθμό σημείων που ενώνονται. Ο τρόπος που δημιουργείται το αντικείμενο παρουσιάζεται παρακάτω.

IndexedTriangleArray triangles =new IndexedTriangleArray(simeia,coordinates,normals,3*trigwna)

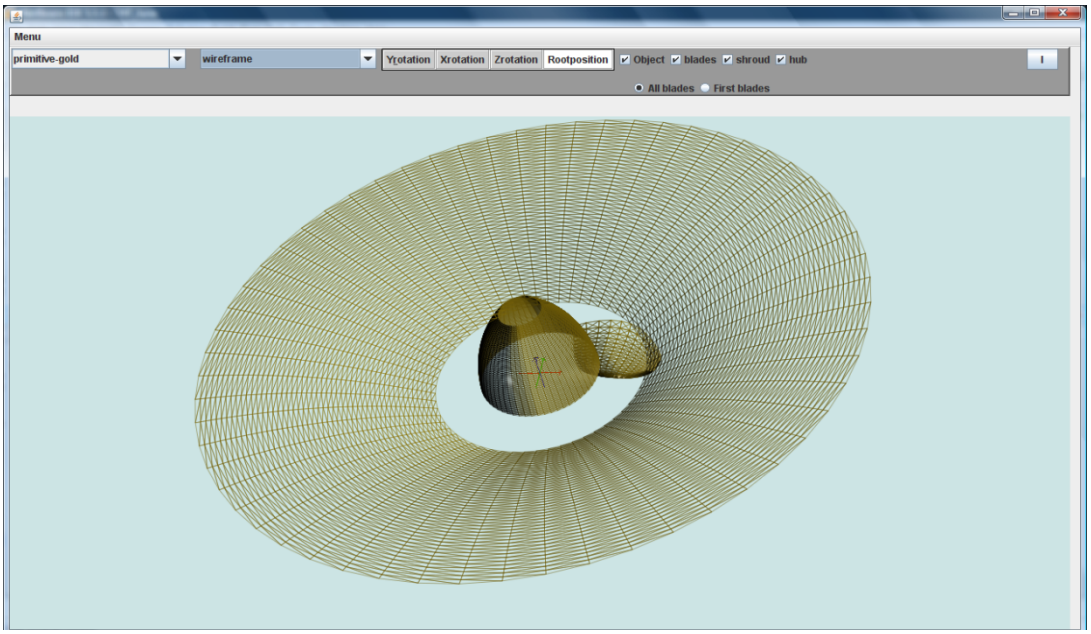
Το αντικείμενο που δημιουργήσαμε χρησιμοποιεί και αποθηκεύει τα δεδομένα των πινάκων. Έπειτα δημιουργούνται τα κάθετα διανύσματα για κάθε τρίγωνο του πλέγματος. Αποθηκεύονται στο αντικείμενο και τα κάθετα διανύσματα και η μέθοδος επιστρέφει τη γεωμετρία του αντικειμένου. Οι εντολές που χρησιμοποιούνται για την αποθήκευση των δεδομένων είναι της μορφής set(), όπως φαίνεται στο παράδειγμα που ακολουθεί.

```
triangles.setCoordinate(i,coords[i]);
triangles.setCoordinateIndex(i,connections[m][n]);
triangles.setNormal(connections[j][count],savenormals[j]);
triangles.setNormalIndex(i,connections[m][n]);
```

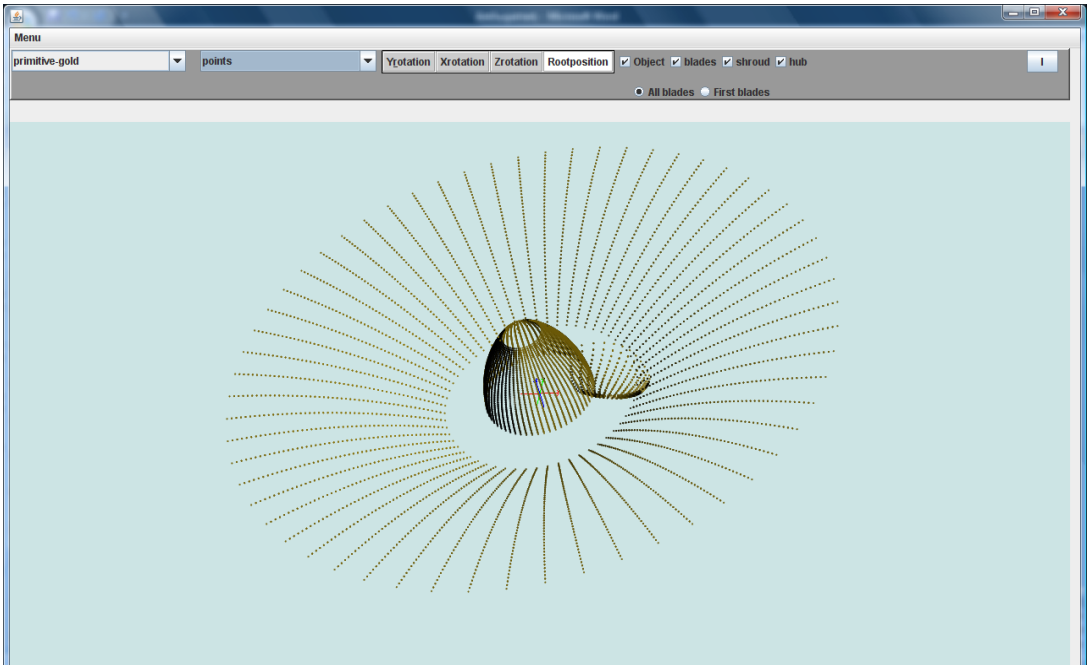
Τα εξωτερικά χαρακτηριστικά του αντικειμένου καθορίζονται στην κλάση Viewer3d. Στην εικόνα 3.2 παρουσιάζεται ένα αντικείμενο που κατασκευάστηκε από αρχείο txt σε μορφή shading, στην εικόνα 3.3 σε μορφή wireframe και στην εικόνα 3.4 μορφή points. Το αντικείμενο αποτελείται από ένα πτερύγιο, την πλήμνη και το κέλυφος.



Εικόνα 3.2. Shading απεικόνιση αντικειμένου από txt file.



Εικόνα 3.3. Wireframe απεικόνιση αντικειμένου από txt file.



Εικόνα 3.4.Points απεικόνιση αντικειμένου από txt file.

3.5. V3D.java(JFrame)

Πρόκειται για κλάση που δημιουργεί το NetBeans και είναι προκαθορισμένη, από το ίδιο, κατά την δημιουργία της να είναι extend στην JFrame κλάση του Java3D Api. Με αυτό τον τρόπο η κλάση παίρνει όλες τις ιδιότητες ενός JFrame αντικειμένου και είναι ένα έτοιμο πλαίσιο παραθύρου με όλα τα εργαλεία που χρειαζόμαστε για να κατασκευάσουμε εφαρμογές που χρησιμοποιούν windows (παράθυρα). Σε αυτό το σημείο κρίνεται σκόπιμο να αναλύσουμε τον τρόπο που λειτουργεί και χρησιμοποιείται το V3D. Η κλάση περιέχει την μοναδική main() μέθοδο που χρησιμοποιείται στην ανάπτυξη του κώδικα της εφαρμογής μας. Αυτό σημαίνει ότι οποιαδήποτε απεικόνιση συμβαίνει στον viewer3d είναι αποτέλεσμα της λειτουργίας της συγκεκριμένης κλάσης.

Η κλάση έχει 2 constructor. Ο πρώτος constructor δεν δέχεται κανένα όρισμα και δημιουργεί το αντικείμενο της κλάσης το οποίο προέρχεται από εξωτερικό αρχείο. Αυτό συμβαίνει με την εξής διαδικασία. Δημιουργείτε στο κώδικα του V3D, ένα αντικείμενο της κεντρικής κλάσης Viewer3d. Το αντικείμενο(triangle) παίρνει τις ιδιότητες του αντικειμένου κλάσης Viewer3d που δημιουργείται στο constructor χωρίς ορίσματα της κλάσης Viewer3d:

Viewer3d triangle= new Viewer3d();

Μέσα στον constructor που δεν χρησιμοποιεί ορίσματα της κλάσης Viewer3d καλείται η μέθοδος createSceneGraph(), την οποία περιγράψαμε στην ενότητα 3.2. Μέσα στην μέθοδο αυτή δημιουργείται ένα αντικείμενο της κλάσης LoaderClass με ίδιο τρόπο όπως στο V3D και το αντικείμενο κλάσης Viewer3d:

LoaderClass load=new LoaderClass();

Το αντικείμενο load, που παρουσιάζεται στο παράδειγμα, έχει όλες τις ιδιότητες του αντικειμένου που δημιουργείται στον constructor της κλάσης LoaderClass. Στην προηγούμενη ενότητα αναλύσαμε ότι το αντικείμενο που δημιουργεί ο constructor της LoaderClass είναι μία γεωμετρία καθορισμένη από τα σημεία της και τις ενώσεις των σημείων της. Άρα αυτή η γεωμετρία αποθηκεύεται στην μεταβλητή load, της προσδίδονται έπειτα τα επιθυμητά χαρακτηριστικά κατά την ανάπτυξη της μεθόδου createSceneGraph(), έπειτα αποθηκεύεται στο αντικείμενο 'εικονικός χώρος και σκηνικά' που δημιουργείται στον constructor της κλάσης Viewer3d και τελικά τα προς εμφάνιση αντικείμενα αποθηκεύονται στην μεταβλητή κλάσης Viewer3d triangle που δημιουργήσαμε στον κενό constructor του V3D. Η main() καλεί τον κενό constructor του V3D και επιτυγχάνεται η απεικόνιση:

```
public static void main(String args[]) {

    java.awt.EventQueue.invokeLater(new Runnable() {
      public void run() {
        new V3D().setVisible(true);
      }
    });
  }
```

Η εντολή new V3D().setVisible(true); είναι η κλήση στον κενό constructor του V3D και η εντολή για απεικόνιση. Αυτή η διαδικασία, δηλαδή η αλληλεπίδραση των κλάσεων χρησιμοποιώντας η μία τα αντικείμενα που κατασκευάζει η άλλη, δημιουργεί μία σχέση μεταξύ των κλάσεων που ονομάζεται αναφορική ή όπως αποκαλείτε στην βιβλιογραφία **reference relationship.** Η σχέση αυτή έχει την ίδια σημασία και ονομασία με την σχέση που ορίστηκε στο κεφάλαιο 1 και που αφορούσε τα Nodes και τα NodeComponets. Αυτό ήταν αναμενόμενο καθώς και οι κλάσεις αποτελούν σύνθετα αντικείμενα, για τις οποίες ισχύουν οι θεμελιώδεις σχέσεις μεταξύ αντικειμένων, όπως η σχέση κληρονομικότητας και η αναφορική.

Ο δεύτερος constructor της κλάσης και πιο βασικός δέχεται 2 ορίσματα. Το πρώτο όρισμα αφορά στα στοιχεία της γεωμετρίας που σχεδιάστηκε παραμετρικά στο T4T. Το δεύτερο όρισμα αφορά τον αριθμό των πτερυγίων που έχει κάθε βαθμίδα πτερύγωσης της στροβιλομηχανής. Ο τρόπος που λειτουργεί το πρόγραμμα χρησιμοποιώντας τον constructor αυτό είναι ο εξής. Όταν τελειώσει το στάδιο της παραμετρικής σχεδίασης κάποιας γεωμετρίας τα δεδομένα της αποθηκεύονται σε μία μεταβλητή τύπου αντικειμένου **LinkedList.** Ο τύπος αντικειμένων LinkedList είναι ένας πολύ χρήσιμος τύπος μεταβλητών. Προέρχεται από την προκαθορισμένη κλάση List της βιβλιοθήκης που δημιουργεί τέτοια αντικείμενα. Έχει την ιδιότητα να αποθηκεύει οποιοδήποτε τύπο αντικειμένου (ακόμα και κενά αντικείμενα) σε θέσεις προκαθορισμένες σε μορφή λίστας. Με αυτό τον τρόπο μπορούμε να διαχειριστούμε κάθε αντικείμενο ξεχωριστά, καλώντας την θέση στην οποία βρίσκεται. Ο τρόπος που δημιουργούμε μια LinkedList μεταβλητή παρουσιάζεται στο παράδειγμα που ακολουθεί:

```
LinkedList[] final_geometries=new LinkedList();
```

Οι γεωμετρίες που αποθηκεύονται στην μεταβλητή τύπου LinkedList μετά το πέρας της σχεδίασης στο T4T είναι οι εξής:

- Γεωμετρία πτερυγίων.
- Γεωμετρία πλήμνης.
- Γεωμετρία κελύφους.

Οι γεωμετρίες είναι ουσιαστικά το νέφος σημείων που ορίζει την κάθε μία γεωμετρία ξεχωριστά. Κάθε μία γεωμετρία με την σειρά, ξεκινώντας από τα πτερύγια και καταλήγοντας στο κέλυφος, αποθηκεύεται σε μία θέση της λίστας της μεταβλητής τύπου LinkedList. Έτσι κάθε θέση της λίστας αποτελείται από μία σειρά σημείων Point3d δηλαδή σημείων με συντεταγμένες (x,y,z).

Επίσης με το πέρας της παραμετρικής σχεδίασης δημιουργείται ακόμα μία μεταβλητή τύπου πίνακα μιας στήλης και n γραμμών(nx1) με στοιχεία ακέραιους αριθμούς. Κάθε θέση του πίνακα περιέχει και έναν αριθμό, που αντιστοιχεί στον αριθμό πτερυγίων της πτερύγωσης της βαθμίδας, που αντιστοιχεί στον αριθμό γραμμής του ακέραιου αριθμού. Προφανώς ο αριθμός n των γραμμών του πίνακα είναι ίδιος με τον αριθμό βαθμίδων της πτερύγωσης. Ο τρόπος δημιουργίας της μεταβλητής παρουσιάζεται στο παρακάτω παράδειγμα:

```
Int [] blades=new int[n]
```

Το επόμενο βήμα μετά την δημιουργία των μεταβλητών και την αποθήκευση των στοιχείων τους είναι η εισαγωγή τους ως ορίσματα στον constructor της V3D κλάσης. Αυτό επιτυγχάνεται δημιουργώντας μία μεταβλητή κλάσης V3D στο σημείο κατασκευής των 2 αυτών μεταβλητών και μέσω της μεταβλητής κλάσης καλείται ο constructor, στον οποίο περνούν ως ορίσματα οι μεταβλητές αυτές:

```
V3D v3d=  new V3D(final_geometries,blades);
```

Όταν οι γεωμετρίες και ο αριθμός πτερυγίων έχουν αποθηκευτεί στον constructor καλείται μέσα σε αυτόν με την ίδια διαδικασία ο constructor της κλάσης Viewer3d, ο οποίος δέχεται τα ίδια ορίσματα, τα οποία και αποθηκεύουμε. Αυτό συμβαίνει διότι η ολοκλήρωση των αντικειμένων απαιτεί την πλεγματοποίηση του νέφους των σημείων με κατάλληλο τρόπο, ώστε να μπορεί να δημιουργηθεί η επιθυμητή επιφάνεια, αλλά και την προσθήκη όλων των επιθυμητών ιδιοτήτων του αντικειμένου. Η πλεγματοποίηση γίνεται στις ξεχωριστές κλάσεις για κάθε αντικείμενο που έχουμε δημιουργήσει και η κλήση των κλάσεων αυτών γίνεται από τον constructor της κεντρικής κλάσης Viewer3d. Η προσθήκη των επιθυμητών ιδιοτήτων γίνεται στην κλάση Viewer3d, όπως αναλυτικά περιγράψαμε στο κεφάλαιο 2. Έχοντας αποθηκεύσει από τον constructor της V3D κλάσης τις μεταβλητές 'γεωμετρίες(final_geometries)' και τον 'αριθμό πτερυγίων(blades)' στον constructor της Viewer3d κλάσης, μπορούμε να τις μεταφέρουμε στους constructor των κλάσεων που θα ολοκληρώσουν το στάδιο της πλεγματοποίησης. Έπειτα θα αποκτήσουν τις επιθυμητές ιδιότητες και θα αποθηκευτούν ολοκληρωμένα στον constructor της V3D κλάσης για να γίνει η απεικόνιση τους.

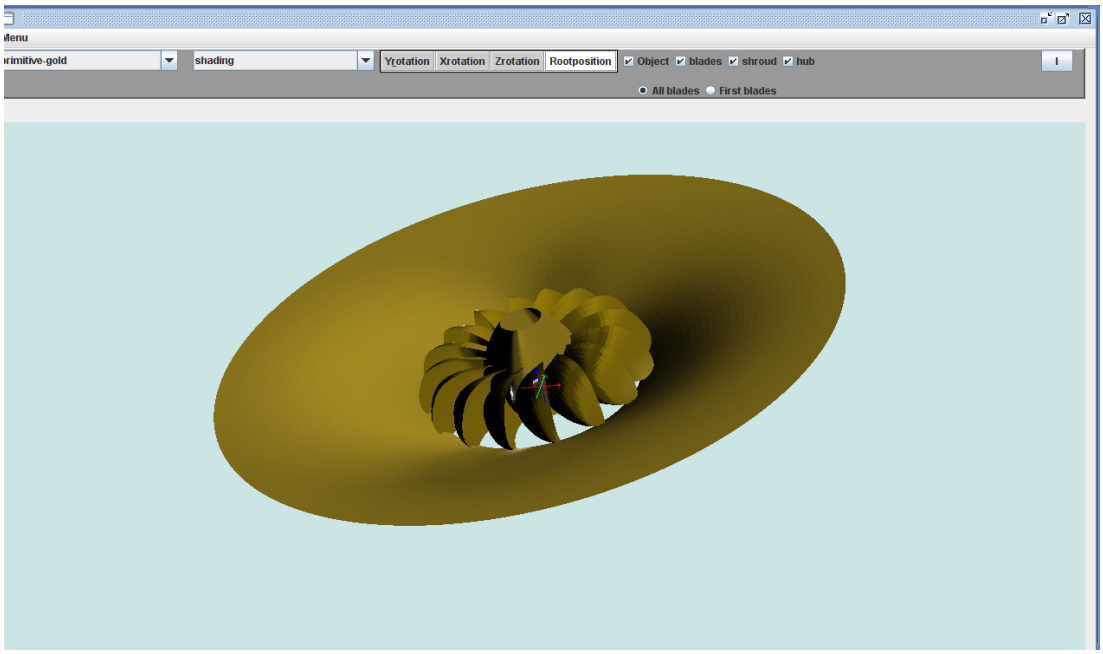
Η κλάση περιέχει επίσης και μία σειρά από μεθόδους ελέγχου των εξαρτημάτων(components), που έχουμε τοποθετήσει πάνω στον Viewer3D. Κατά την τοποθέτηση ενός εξαρτήματος πάνω στο JFrame παράθυρο στη μορφή Design(βλ. καφάλαιο 2) και κατά την προσθήκη ενός Event στο εξάρτημα αυτό(π.χ. ActionEvent), δημιουργείται αυτόματα η μέθοδος ελέγχου του Event που προσθέσαμε, ώστε να καθορίσουμε ότι θέλουμε να συμβεί κατά την εκτέλεση του συγκεκριμένου Event. Ένα τέτοιο παράδειγμα μεθόδου ακολουθεί στο παραδειγμα παρακάτω.

```
private void RotationAction(java.awt.event.ActionEvent evt) {}
```

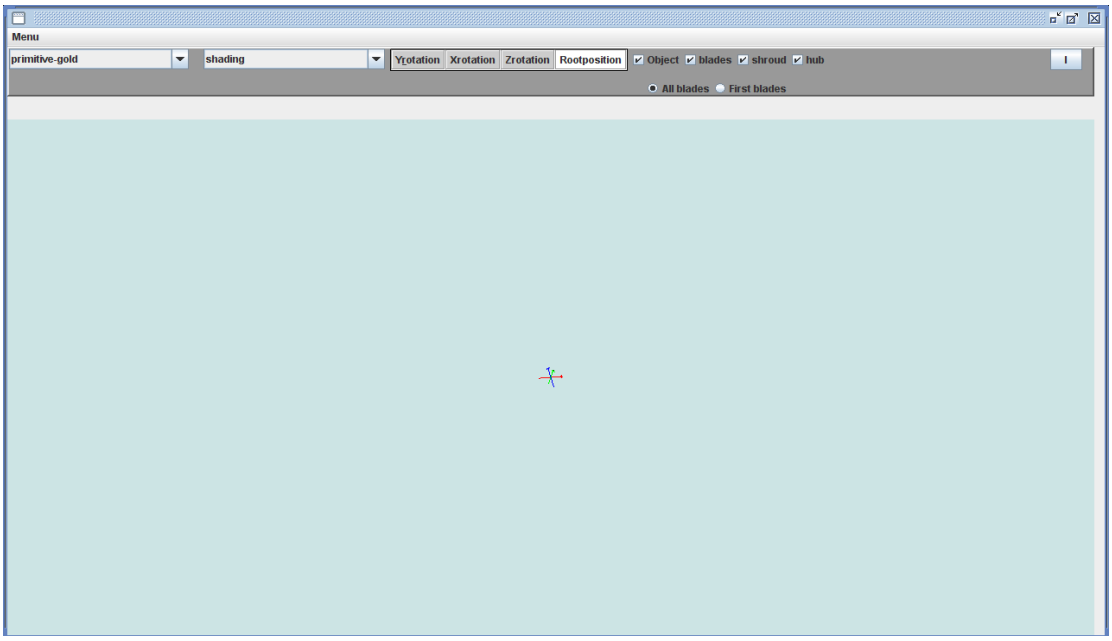
Κάθε μία από τις μεθόδους προγραμματίστηκε ανάλογα με το εξάρτημα και την λειτουργία που επιθυμούμε να έχει έτσι για παράδειγμα η μέθοδος ελέγχου του

εξαρτήματος Yrotation καλεί την μέθοδο Yactionperform() της κλάσης Viewer3d, που προκαλεί την περιστροφή του αντικειμένου κατά 45 μοίρες.

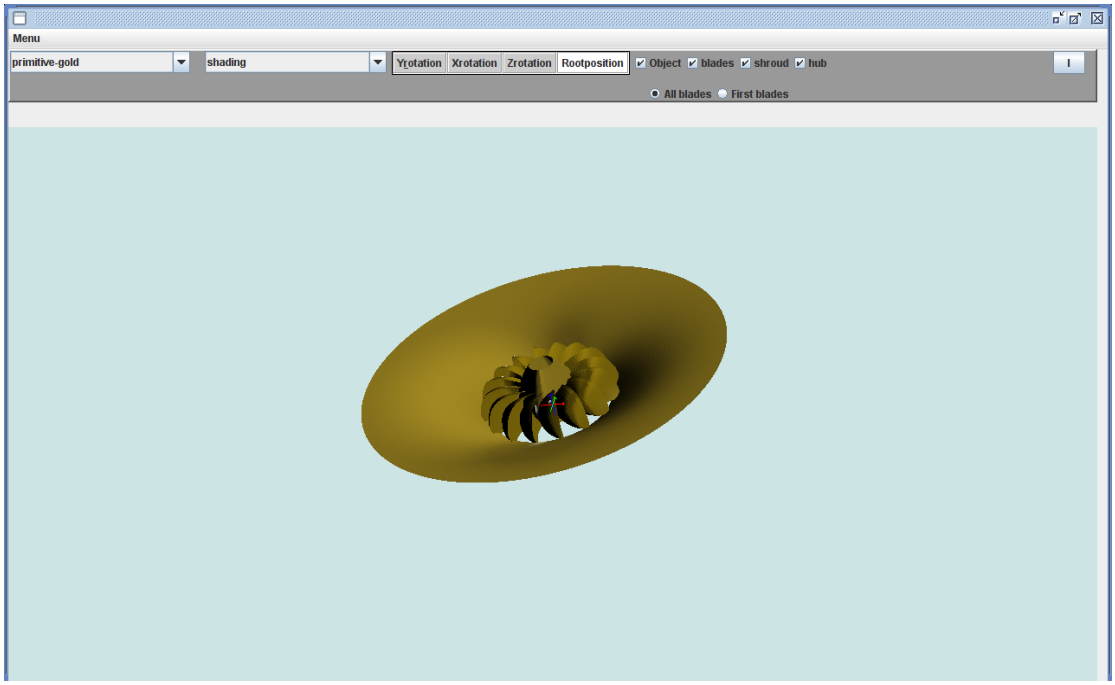
Εκτός από τις μεθόδους ελέγχου των Event των εξαρτημάτων υπάρχει και ένα πακέτο μεθόδων που καθορίζουν αλληλεπίδραση του ποντικιού με τον Viewer3D. Η αλληλεπίδραση αυτή καλείται MouseEvent. Συγκεκριμένα όταν συμβεί κάποια λειτουργία του ποντικιού, όπως το πάτημα ενός κουμπιού ή η μετακίνηση του, μπορούμε να προγραμματίσουμε να συμβαίνουν κάποιες επιθυμητές λειτουργίες. Στην εφαρμογή μας οι επιθυμητές λειτουργίες ήταν αυτές που αναπτύχθηκαν στο κεφάλαιο 2 (Zoom, Rotate, Translate) αλλά και αυτή που προγραμματίστηκε στην κλάση αυτή. Η λειτουργία αυτή είναι να εξαφανίζονται όλα τα αντικείμενα κατά το πάτημα ενός κουμπιού του mouse(MousePressed) και να εμφανίζονται ξανά με την απελευθέρωση του κουμπιού(MouseReleased) έχοντας όμως αποθηκεύσει οποιαδήποτε αλλαγή του αντικειμένου προκλήθηκε στις ιδιότητές του στο διάστημα μεταξύ της του πατήματος και της ελευθέρωσης του κουμπιού. Έστω το αντικείμενο της εικόνας 3.5 στο οποίο με το πάτημα του ρότορα του mouse και την μετακίνηση του σε κατεύθυνση αντίθετη από την οθόνη προς το χρήστη, προκαλείται η λειτουργία zoom out. Με το πάτημα του ρότορα εξαφανίζεται το αντικείμενο όπως παρουσιάζεται στην εικόνα 3.6, με την μετακίνηση του mouse προκαλείται αλλαγή της θέσης του εξαφανισμένου αντικειμένου όπου και αποθηκεύεται στις ιδιότητες του και με την απελευθέρωση του ρότορα εμφανίζεται το αντικείμενο στη νέα του θέση, εικόνα 3.7.



Εικόνα 3.5.Αντικείμενο σε αρχική θέση.



Εικόνα 3.6.MousePressed and Zoom out.



Εικόνα 3.7.MouseReleased.

Αυτή η ιδιότητα κρίθηκε αναγκαία για λόγους μεγαλύτερης ταχύτητας απεικόνισης. Τα αντικείμενα που απεικονίζονται στον viewer3d είναι αντικείμενα που οι επιφάνειες τους σχεδιάστηκαν με βάση ένα νέφος σημείων που μπορεί να αποτελείται από εκατομμύρια(!) σημεία. Για παράδειγμα ένα αντικείμενο μπορεί να αποτελείται από ένα νέφος σημείων που περιέχει περίπου 4.000.000 σημεία. Για να γίνει κάποια αλλαγή όπως η θέση του πρέπει να πολλαπλασιαστεί κάθε ένα σημείο με το διάνυσμα μετατόπισης που σημαίνει 4.000.000 πράξεις για κάθε παραμικρή αλλαγή θέσης. Αν επιθυμούμε να μεταφέρουμε το αντικείμενο από μια θέση α σε μία θέση β για κάθε ενδιάμεση κατάσταση ο υπολογιστής κάνει 4.000.000 πράξεις. Αυτό το γεγονός δημιουργεί προβλήματα ταχύτητας στον υπολογιστή. Επιλέξαμε λοιπόν να διαγράφεται η πολύπλοκη γεωμετρία κατά την έναρξη της μεταφοράς, να αποθηκεύεται μόνο το διάνυσμα μεταφοράς από την θέση α στη θέση β

στην μνήμη του υπολογιστή και έπειτα να σχεδιάζεται και πάλι η πολύπλοκη γεωμετρία λαμβάνοντας υπόψη το διάνυσμα μεταφοράς. Με αυτό τον τρόπο καταφέρνουμε να αυξήσουμε την ταχύτητα απόκρισης του λογισμικού.

Για να καταλαβαίνει η κλάση ότι κάποιο MouseEvent έλαβε χώρα πρέπει να προσθέσουμε στα αντικείμενα της κλάσης ένα αντικειμένου που καλείται MouseListener. Αυτό τα αντικείμενο έχει την ιδιότητα να 'ακούει' την ενεργοποίηση κάποιας λειτουργίας του ποντίκι και να ειδοποιεί την κλάση. Η πρόσθεση αυτού του αντικειμένου γίνεται χρησιμοποιώντας την σχέση κληρονομικότητας, δηλαδή η κλάση κληρονομεί τις ιδιότητες της κλάσης που δημιουργεί τα αντικείμενα MouseListeners. Για να το πετύχουμε αυτό χρησιμοποιούμε την εντολή **implements** στη δήλωση της κλάσης:

public class V3D extends javax.swing.JFrame implements MouseListener {}

3.6. AllBladesClass.java

Η κλάση AllBladesClass είναι η κλάση που κατασκευάζει το τριγωνικό πλέγμα του συνόλου της πτερύγωσης, πάνω στο οποίο θα εμφανιστεί η επιφάνεια των πτερυγίων. Η κλάση χρησιμοποιεί ένα constructor που δέχεται 2 ορίσματα. Το πρώτο όρισμα είναι η μεταβλητή τύπου LinkedList, που περιέχει τις γεωμετρίες σε μορφή νέφους σημείων του συνόλου του αντικειμένου. Το άλλο όρισμα είναι η μεταβλητή τύπου πίνακα που περιέχει τον αριθμό των πτερυγίων κάθε βαθμίδας πτερύγωσης. Οι μεταβλητές αυτές προέρχονται από την παραμετρική σχεδίαση του αντικειμένου μέσω της διαδρομής:

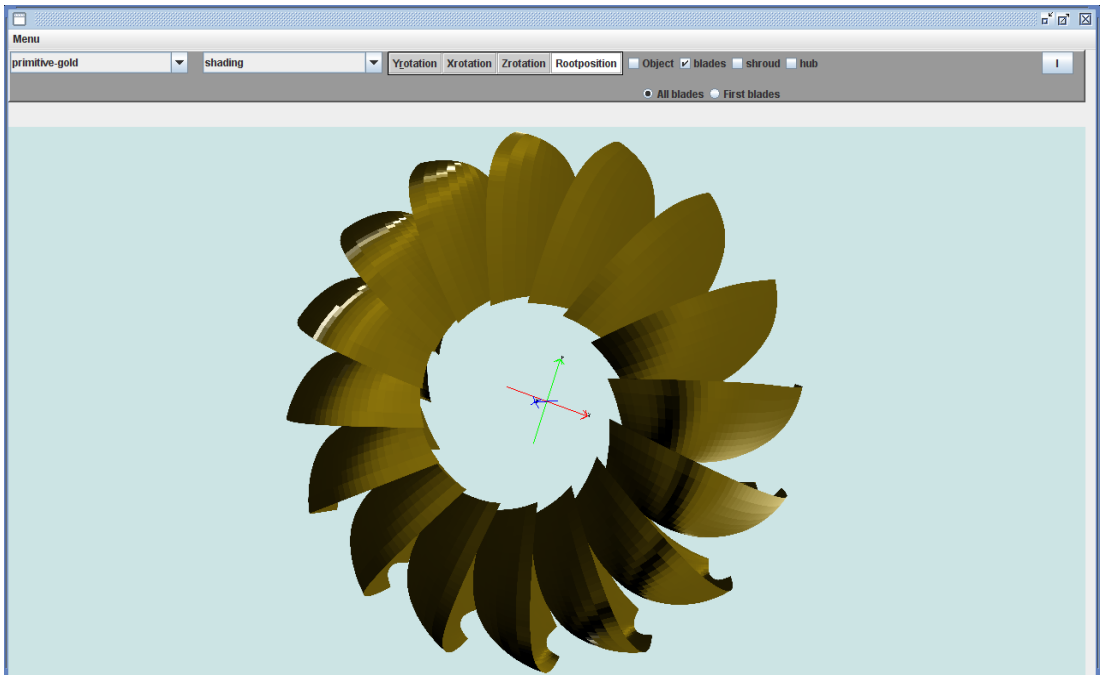
T4T→V3D(LinkedList,int[])→Viewer3d(LinkedList,int[])→AllBladesClass(LinkedList,int[])

Τα δεδομένα των μεταβλητών αυτών αποθηκεύονται σε τοπικές μεταβλητές μέσα στον constructor της κλάσης. Ακολουθώντας την ίδια διαδικασία με αυτήν της κλάσης LoaderClass δημιουργούμε ένα πίνακα (coords [Point3d]) για να αποθηκεύσουμε τα σημεία του συνόλου της πτερύγωσης. Τα σημεία είναι σε μορφή συντεταγμένων x,y,z. Επειδή η μεταβλητή τύπου LinkedList περιέχει τα σημεία του συνόλου του αντικειμένου (πτερύγια, κέλυφος, πλήμνη) απομονώνουμε τις θέσεις του LinkedList που περιέχουν τα σημεία της πτερύγωσης. Είναι προκαθορισμένη η σειρά με την οποία αποθηκεύονται τα σημεία των μερών του αντικειμένου στο LinkedList. Συγκεκριμένα πρώτα αποθηκεύονται τα σημεία της πτερύγωσης, έπειτα της πλήμνης και τελικά του κελύφους. Χρησιμοποιώντας με αυτό τον τρόπο όλες τις θέσεις του LinkedList πλην των 2 τελευταίων που αφορούν την πλήμνη και το κέλυφος αποθηκεύουμε στον πίνακα που δημιουργήσαμε, όλα τα σημεία που μας αφορούν.

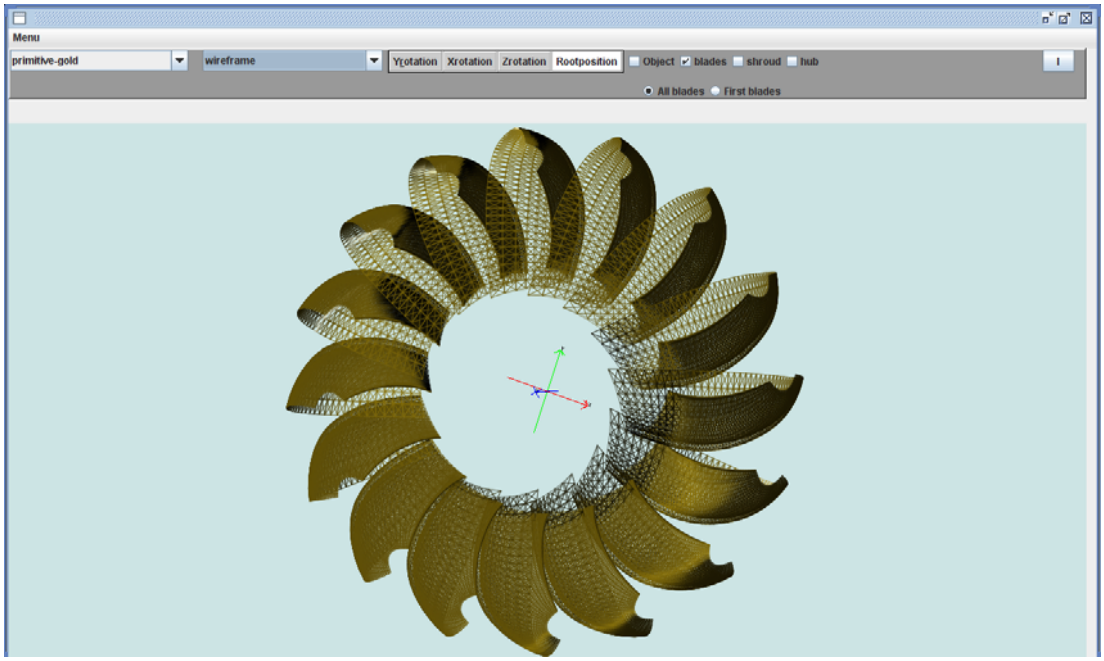
Στη μεταβλητή LinkedList στις θέσεις που αναφέρονται στις γεωμετρίες των πτερυγίων έχουν αποθηκευτεί μόνον οι γεωμετρίες του πρότυπου πτερυγίου κάθε βαθμίδας πτερύγωσης. Θα ήταν ανούσιο και χρονοβόρο να αποθηκεύουμε κάθε μία γεωμετρία κάθε πτερυγίου της ίδιας πτερύγωσης καθώς πρόκειται για την ίδια γεωμετρία που έχει περιστραφεί κατά μία γωνία γύρω από άξονα (μέθοδος pattern). Το μέγεθος της γωνίας εξαρτάται από το πλήθος των πτερυγίων της πτερύγωσης. Το πόσες φορές θα περιστραφεί το πτερύγιο το καθορίζει η μεταβλητή τύπου πίνακα που περιέχει τον αριθμό πτερυγίων κάθε βαθμίδας. Η γωνία περιστροφής για την βαθμίδα i θα είναι (360/αριθμό πτερυγίων της βαθμίδας i).

Προσδίδοντας σκίαση ώστε να δημιουργηθεί η επιφάνεια και όλες τις επιθυμητές ιδιότητες στην επιφάνεια του αντικειμένου στην κεντρική κλάση Viewer3d, απεικονίζεται με τον τρόπο που παρουσιάζεται στην εικόνα 3.8 η πτερύγωση ενός αντικειμένου σε μορφή

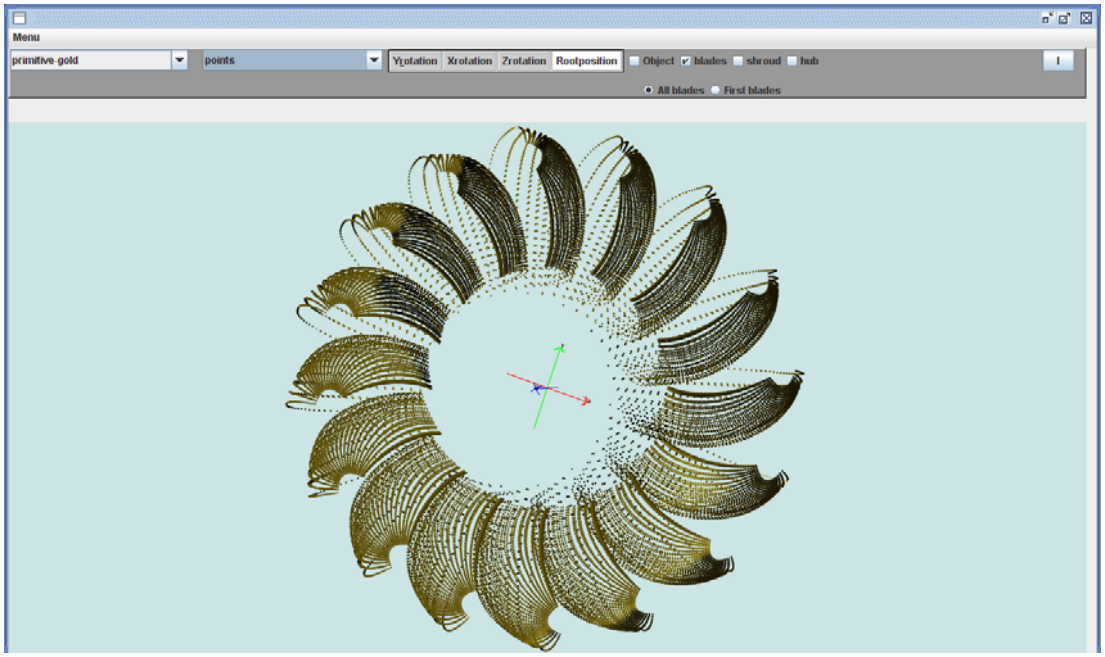
shading. Στην εικόνα 3.9. παρουσιάζεται η wireframe μορφή της πτερύγωσης και στην εικόνα 3.10 η μορφή points.



Εικόνα 3.8. Πτερύγωση σε μορφή shading.



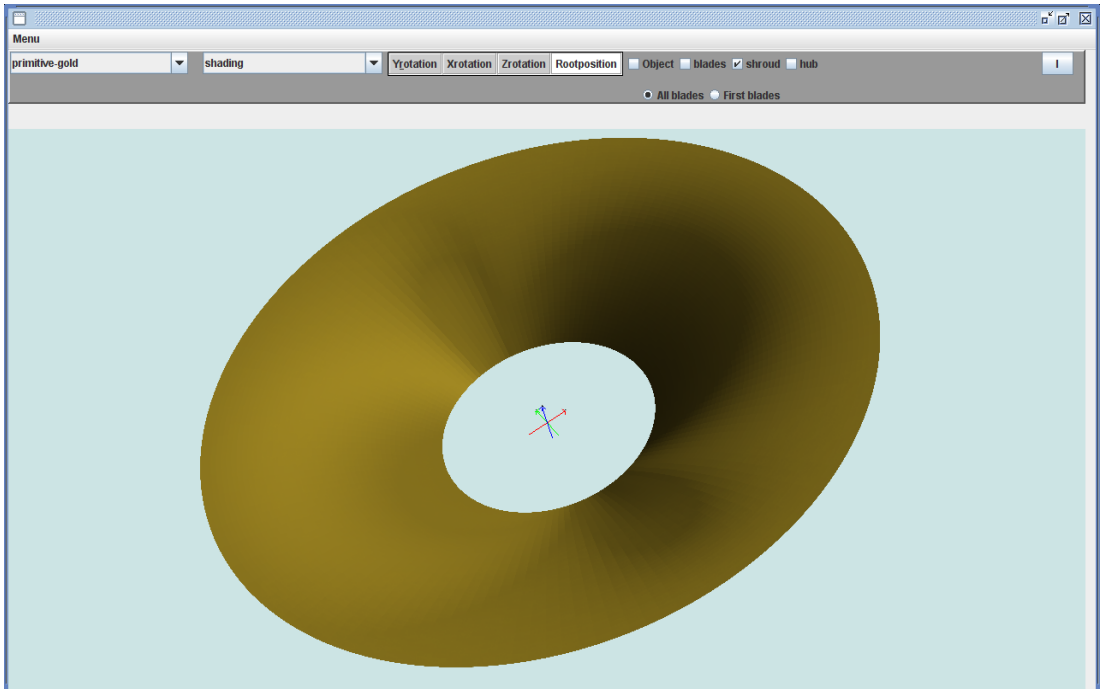
Εικόνα 3.9. Πτερύγωση σε μορφή wireframe.



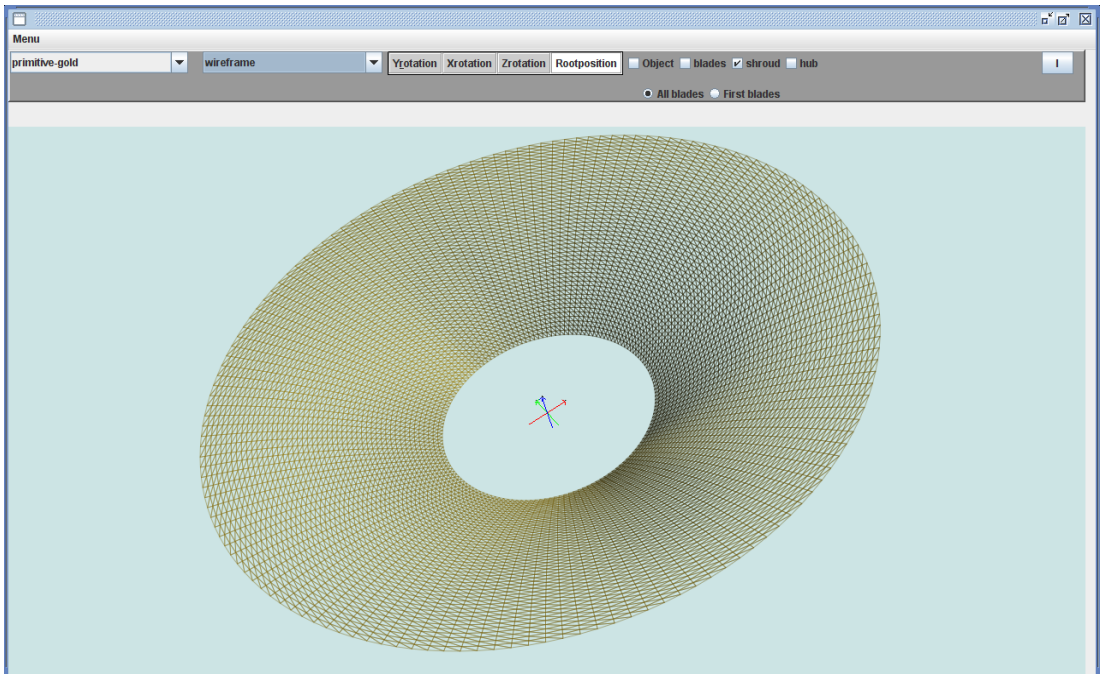
Εικόνα 3.10. Πτερύγωση σε μορφή points.

3.7. BladeClass.java-ShroudClass.java-HubClass.java.

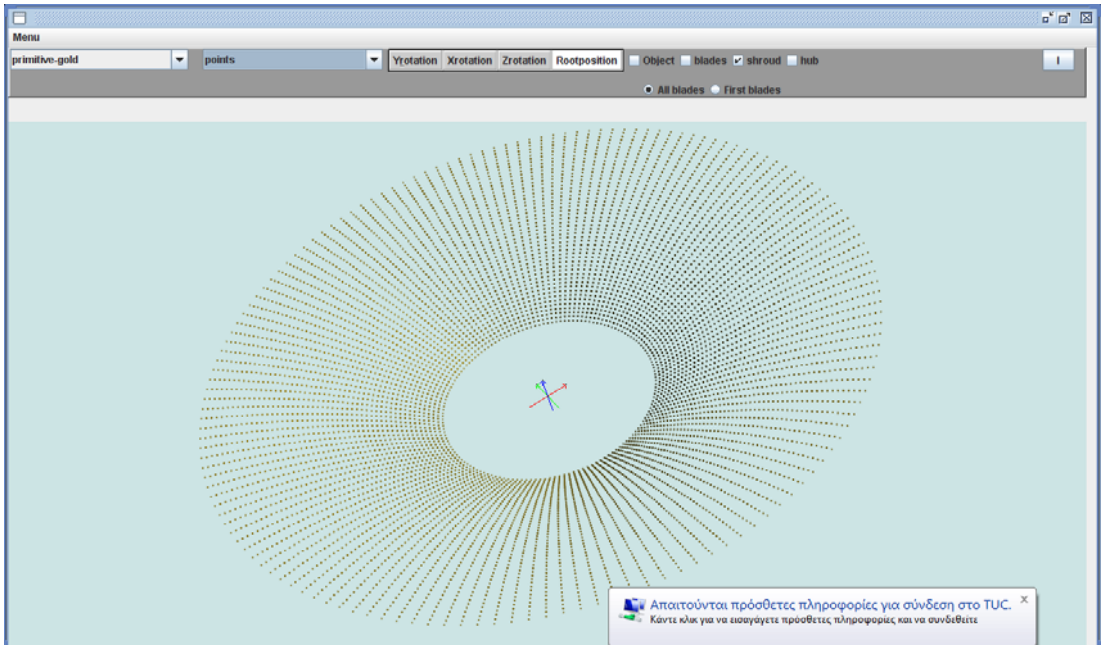
Το πακέτο αυτών των τριών κλάσεων λειτουργεί ακριβώς με τον τρόπο που λειτουργεί η κλάση AllBlades.java. Οι μόνη διαφοροποίηση είναι ότι οι constructor των κλάσεων αυτών δέχονται μόνο ένα όρισμα αυτό του αντικειμένου τύπου LinkedList που είναι αποθηκευμένο το νέφος σημείων του συνόλου του αντικειμένου. Αυτό συμβαίνει διότι οι μεν κλάσεις ShroudClass.java και HubClass.java κατασκευάζουν τα αντικείμενα πλέγμα κέλυφος και πλέγμα πλήμνης αντίστοιχα, με αποτέλεσμα η πληροφορία του αριθμού των πτερυγίων σε κάθε βαθμίδα πτερύγωσης να είναι άχρηστη, η δε κλάση BladeClass.java δημιουργεί μόνο τα πλέγματα των πρότυπων (πρώτων) πτερύγιων κάθε βαθμίδας πτερύγωσης οπότε επίσης δεν χρειάζεται η πληροφορία του αριθμού των πτερυγίων για κάθε βαθμίδα πτερύγωσης. Επίσης μια διαφορά έγκειται στις θέσεις του LinkedList που κάθε κλάση χρησιμοποίει. Συγκεκριμένα η κλάση ShroudClass.java χρησιμοποιεί την τελευταία θέση του αντικειμένου LinkedList, όπου και περιέχεται το νέφος σημείων του κελύφους. Η κλάση HubClass.java χρησιμοποιεί την προτελευταία θέση του LinkedList, όπου και περιέχεται το νέφος σημείων της πλήμνης και τέλος η κλάση BladeClass.java χρησιμοποιεί όλες τις θέσεις του LinkedList πλην των 2 τελευταίων. Τα αντικείμενα που δημιουργεί κάθε κλάση παρουσιάζονται στις εικόνες που ακολουθούν. Η ShroudClass όπως αναφέρθηκε δημιουργεί το πλέγμα του κελύφους και στις εικόνες 3.11, 3.12, 3.13 παρουσιάζεται το κέλυφος ενός αντικειμένου στις μορφές shading, wireframe και points αντίστοιχα.



Εικόνα 3.11. Κέλυφος σε μορφή shading.

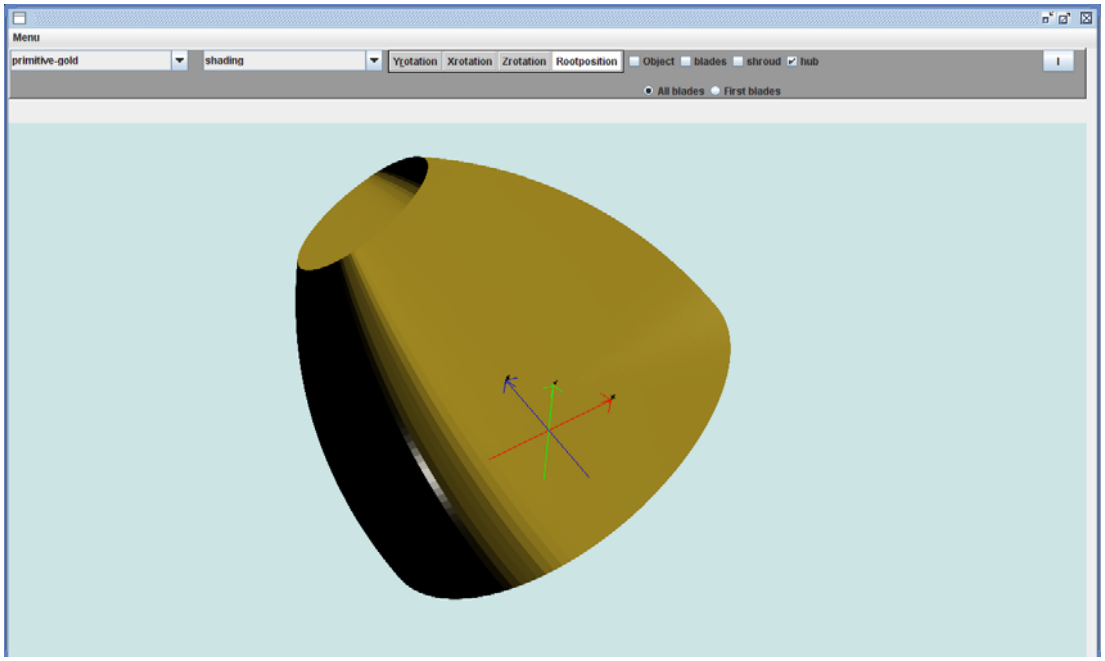


Εικόνα 3.12. Κέλυφος σε μορφή wireframe.

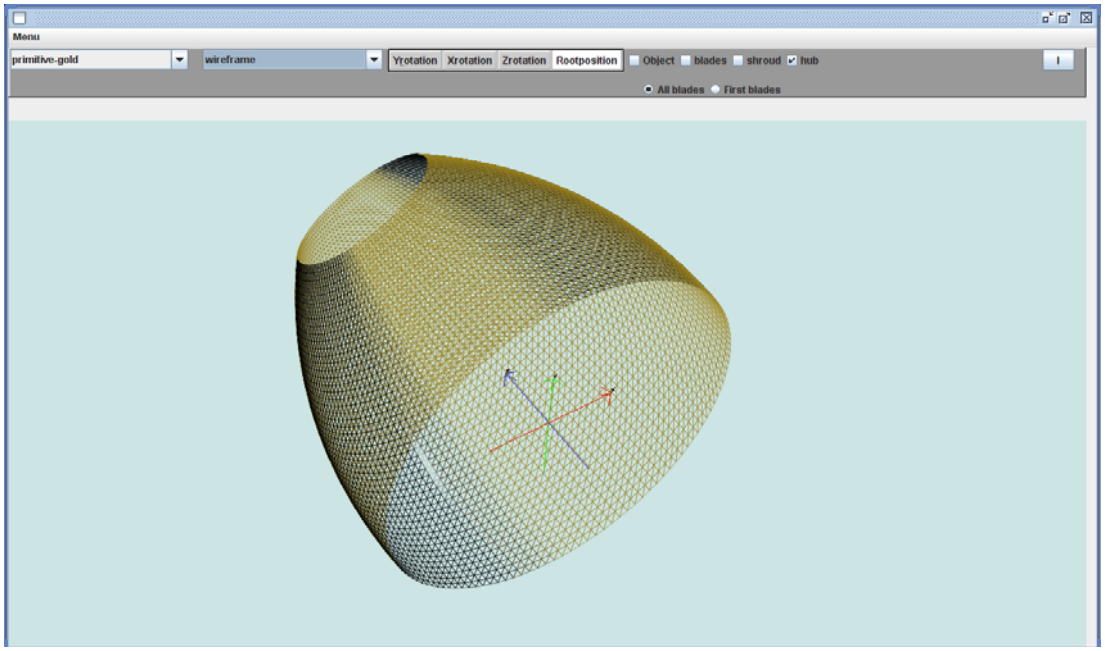


Εικόνα 3.13. Κέλυφος σε μορφή points.

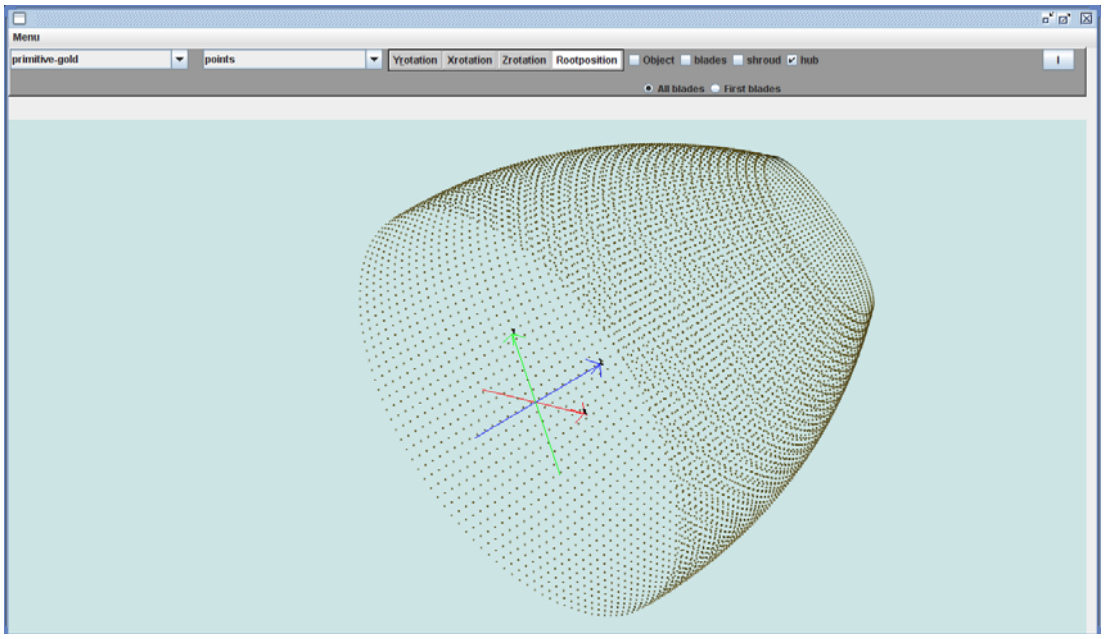
Η HubClass όπως αναφέρθηκε δημιουργεί το πλέγμα της πλήμνης και στις εικόνες 3.14, 3.15, 3.16 παρουσιάζεται η πλήμνη ενός αντικειμένου στις μορφές shading, wireframe και points αντίστοιχα.



Εικόνα 3.14. Πλήμνη αντικειμένου σε μορφή shading.

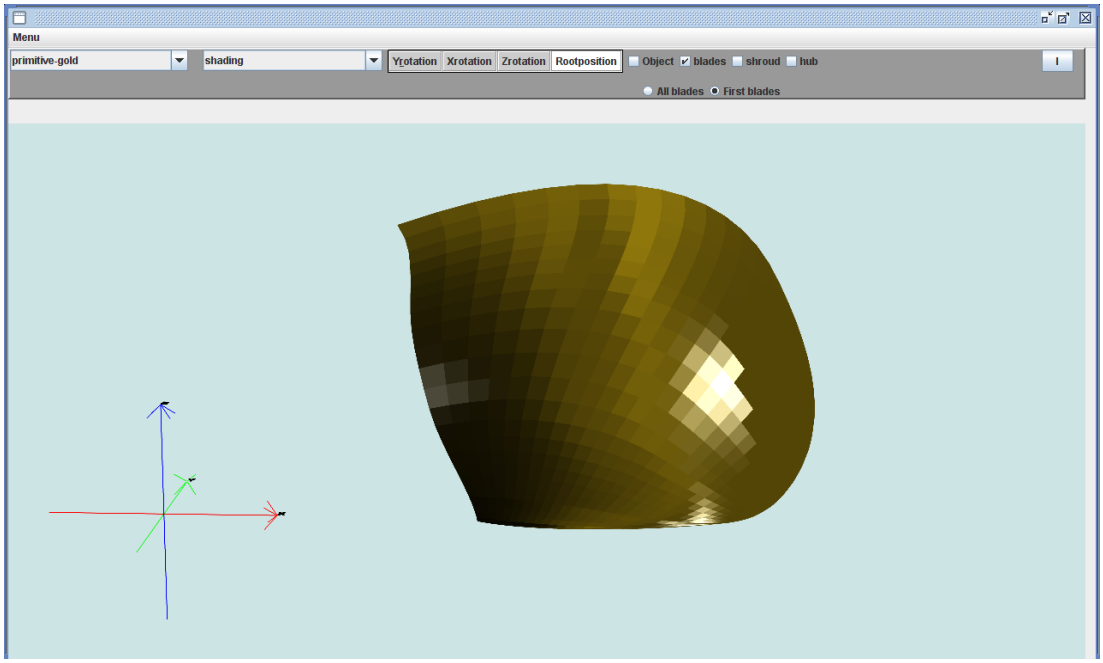


Εικόνα 3.13. Πλήμνη αντικειμένου σε μορφή wireframe.

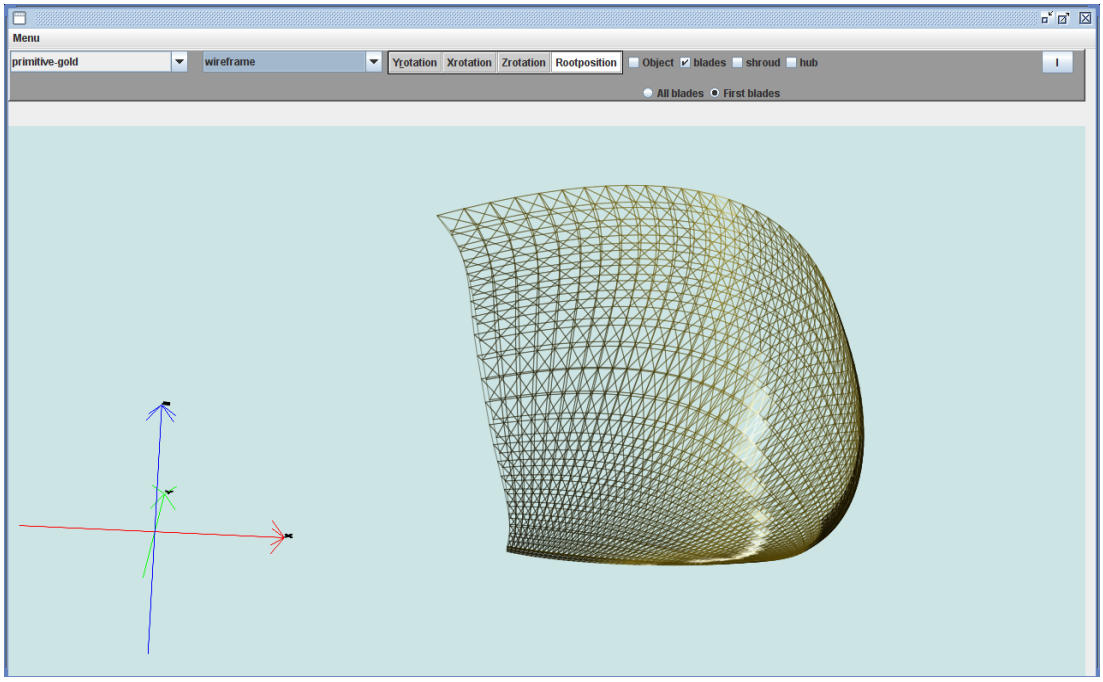


Εικόνα 3.13. Πλήμνη αντικειμένου σε μορφή points.

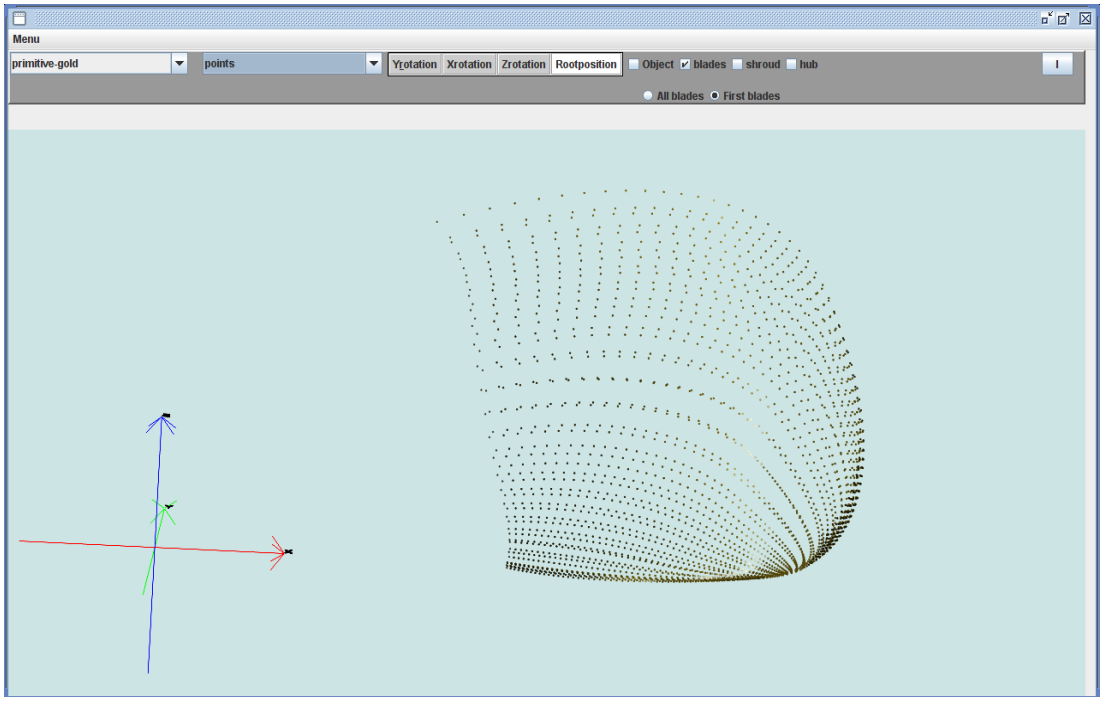
Η BladeClass όπως αναφέρθηκε δημιουργεί το πλέγμα των πρότυπων (πρώτων) πτερυγίων κάθε βαθμίδας πτερύγωσης. Στις εικόνες 3.17, 3.18, 3.19 παρουσιάζεται ένα πτερύγιο ενός αντικειμένου με μία βαθμίδα πτερύγωσης. Εικόνες πολυβάθμιων πτερυγώσεων θα παρουσιαστούν στο κεφάλαιο 4.



Εικόνα 3.17. Πρότυπο πτερύγιο με 1 βαθμίδα πτερύγωσης σε μορφή shading.



Εικόνα 3.18. Πρότυπο πτερύγιο με 1 βαθμίδα πτερύγωσης σε μορφή wireframe.



Εικόνα 3.19. Πρότυπο πτερύγιο με 1 βαθμίδα πτερύγωσης σε μορφή points.

4

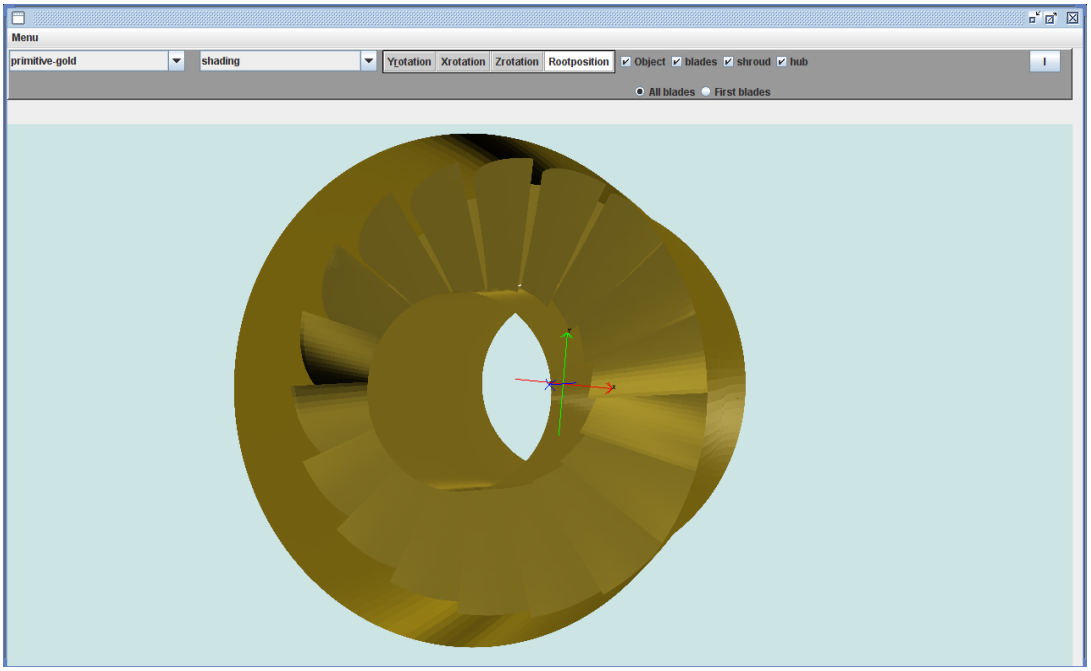
Εφαρμογές

4.1. Εισαγωγή

Το T4T πλέον διαθέτει ένα αυτόνομο τρισδιάστατο viewer3d για την αποτελεσματική απεικόνιση των αντικειμένων που σχεδιάζει. Μπορεί να υστερεί αρκετών εμπορικών πακέτων σε αρκετά σημεία αλλά ικανοποιεί σε αρκετά μεγάλο βαθμό με την λειτουργικότητα και την ταχύτητα σχεδίασης και απεικόνισης των αντικειμένων. Στο κεφάλαιο αυτό παρουσιάζονται παραδείγματα πτερυγώσεων στροβιλομηχανών, που παρήχθησαν με το T4T και η απεικόνιση τους γίνεται με την χρήση του Viewer3D.

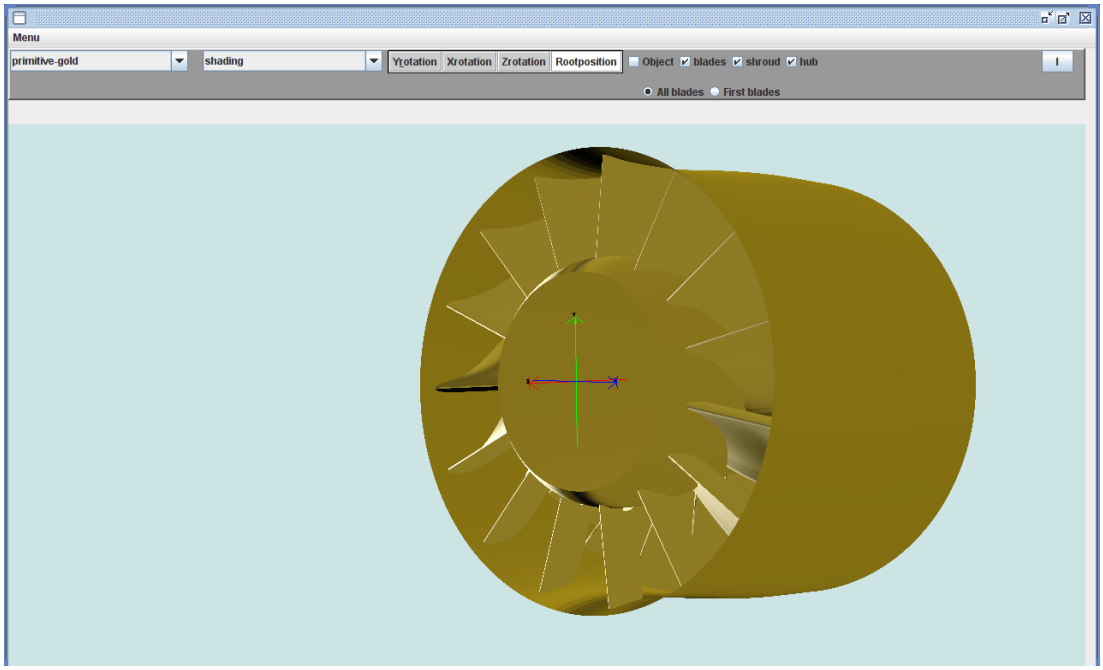
4.2. Στρόβιλος

Στην Εικόνα 4.1 απεικονίζεται ένας πολυβάθμιος στρόβιλος από την μπροστινή όψη δηλαδή από την πλευρά του IGV (Inlet Guide Vanes), του οποίου η γεωμετρία ορίστηκε με τη χρήση του T4T. Ο στρόβιλος απαρτίζεται συνολικά από τριάντα πέντε NURBS επιφάνειες. Δύο επιφάνειες εκ περιστροφής (πλήμνη και κέλυφος) και τριάντα τρία πτερύγια. Τα πτερύγια είναι τοποθετημένα σε δύο διαδοχικές πτερυγώσεις (στάτορας – ρότορας) και προέκυψαν με τη διαδικασία της αντιγραφής και μεταφοράς των πρότυπων πτερυγίων (διαδικασία pattern).



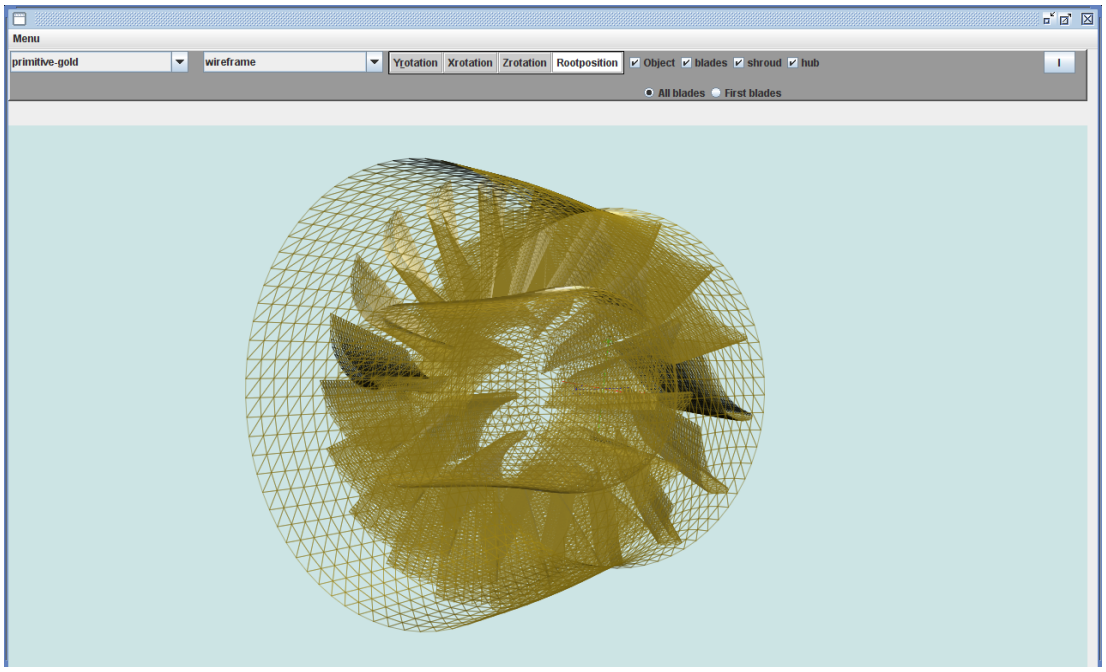
Εικόνα 4.1. Πολυβάθμιος στρόβιλος σε μορφή shading (μπροστινή όψη).

Στην εικόνα 4.2 παρουσιάζεται η πίσω όψη του στροβίλου, η πλευρά δηλαδή που εμφανίζεται ο ρότορας.



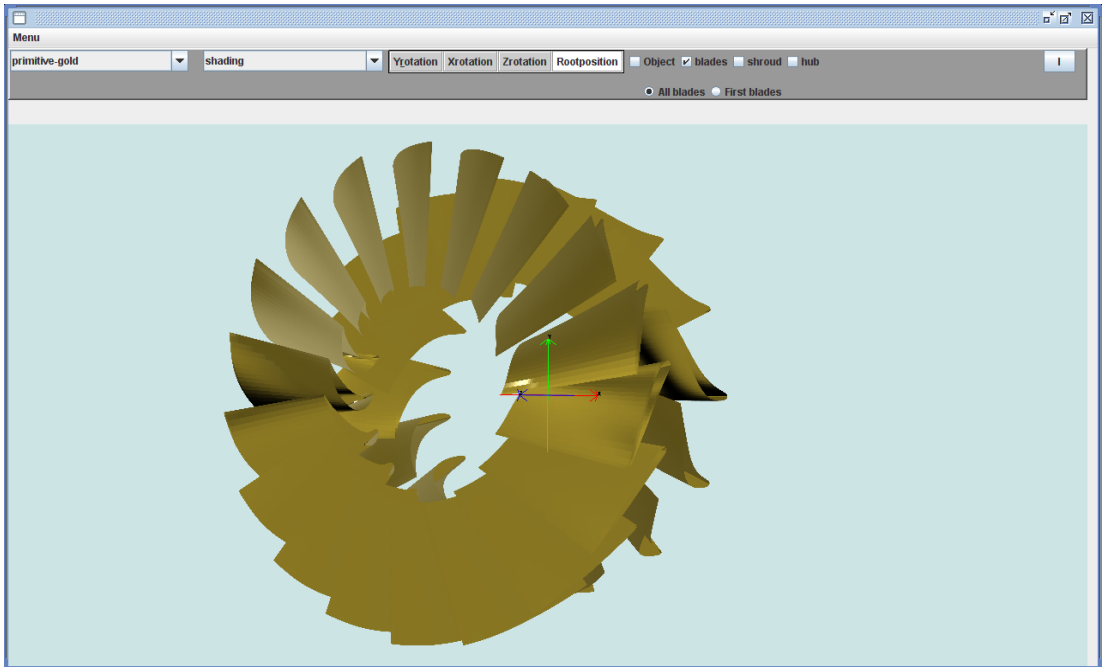
Εικόνα 4.2. Πολυβάθμιος στρόβιλος σε μορφή shading (πίσω όψη).

Στην εικόνα 4.3 παρουσιάζεται η wireframe μορφή του στρόβιλου.



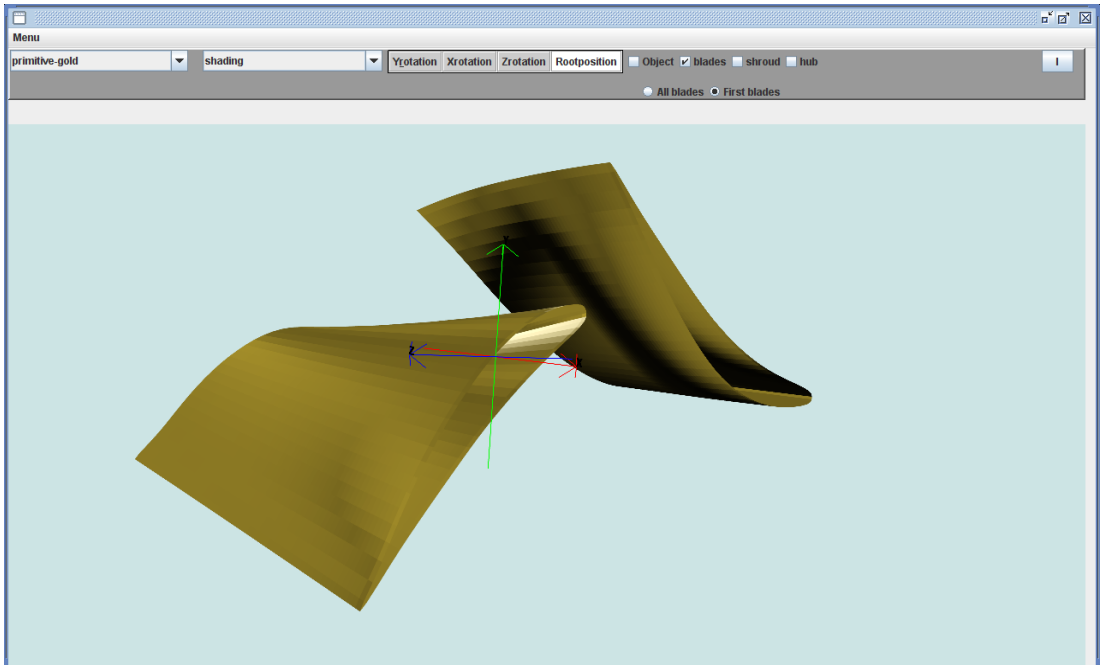
Εικόνα 4.3. Πολυβάθμιος στρόβιλος σε μορφή wireframe.

Στην εικόνα 4.4 παρουσιάζεται το σύνολο της πτερύγωσης του στροβίλου σε μορφή shading. Αποτελείται από 2 βαθμίδες πτερύγωσης με 15 πτερύγια η πρώτη και 18 η δεύτερη. Η πρώτη βαθμίδα αποτελεί τον στάτορα ενώ η δεύτερη τον ρότορα.



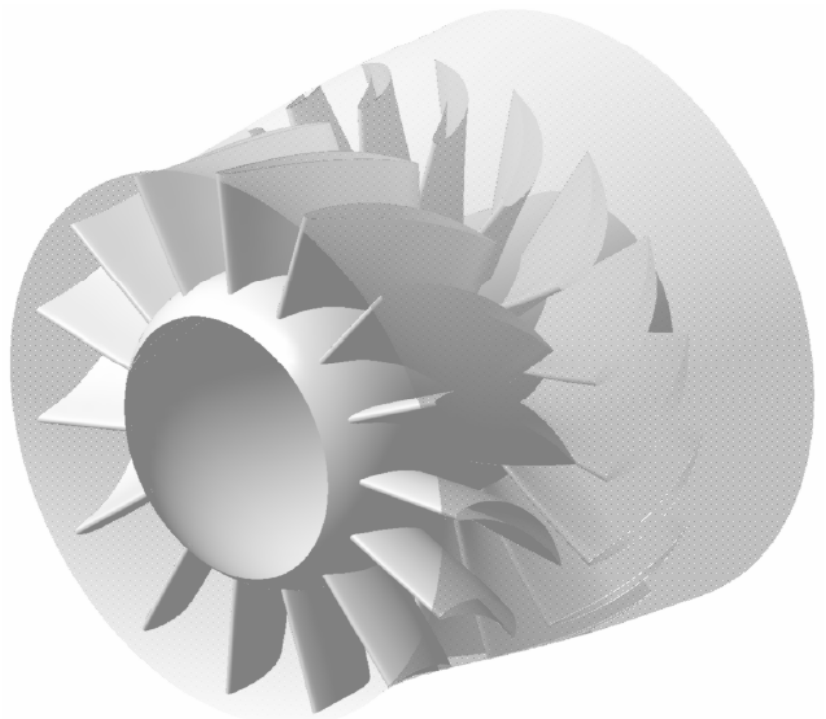
Εικόνα 4.4. Πτερύγωση στροβίλου σε μορφή shading.

Στην εικόνα 4.5 παρουσιάζονται τα πρότυπα πτερύγια της πτερύγωσης του στροβίλου σε μορφή shading.

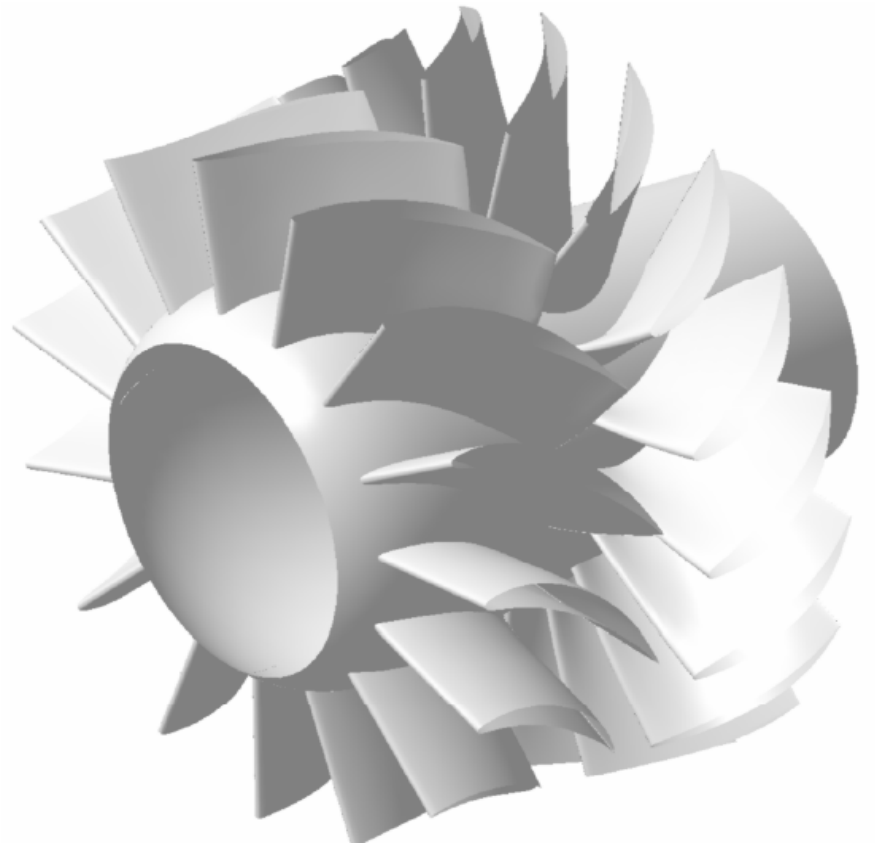


Εικόνα 4.5. Πρότυπα πτερύγια στροβίλου σε μορφή shading.

Για να αποκτήσουμε κάποια συγκριτική άποψη για την ποιότητα της απεικόνισης του Viewer3D παρουσιάζουμε την απεικόνιση του στροβίλου με χρήση ενός ευρέως χρησιμοποιούμενου σχεδιαστικού λογισμικού, του λογισμικού CATIA. Στην εικόνα 4.6 παρουσιάζεται το σύνολο του στροβίλου, ενώ στην εικόνα 4.7 η πλήμνη με την πτερύγωση.



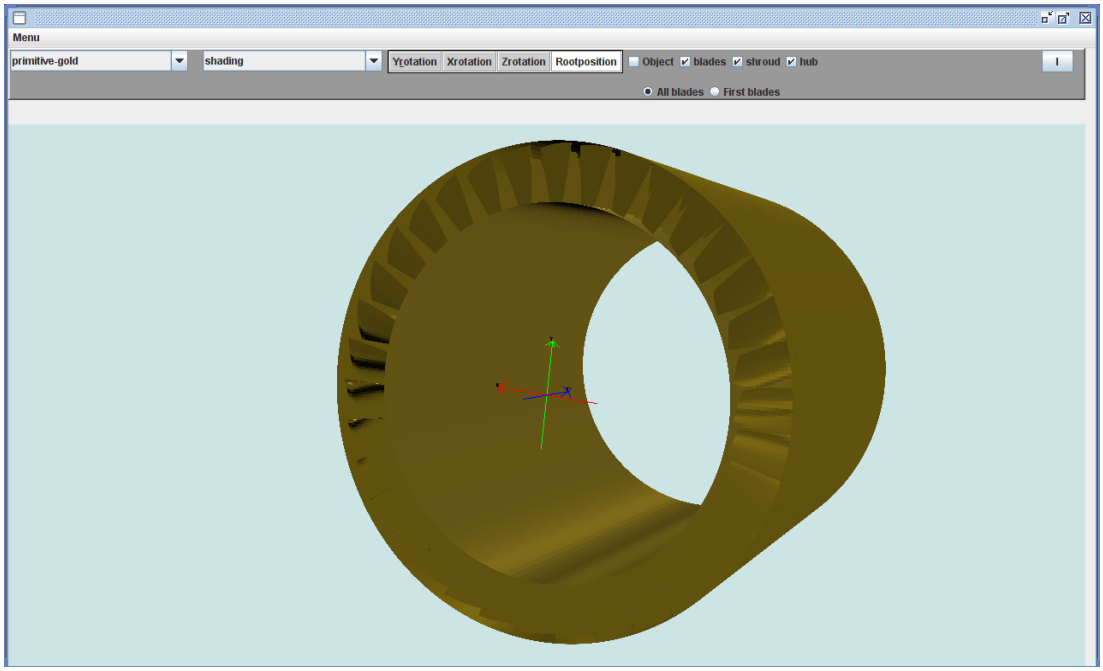
Εικόνα 4.6 Απεικόνιση στροβίλου με χρήση λογισμικού CATIA.(5)



Εικόνα 4.7 Απεικόνιση πλήμνης και πτερύγωσης του στροβίλου με χρήση λογισμικού CATIA.(5)

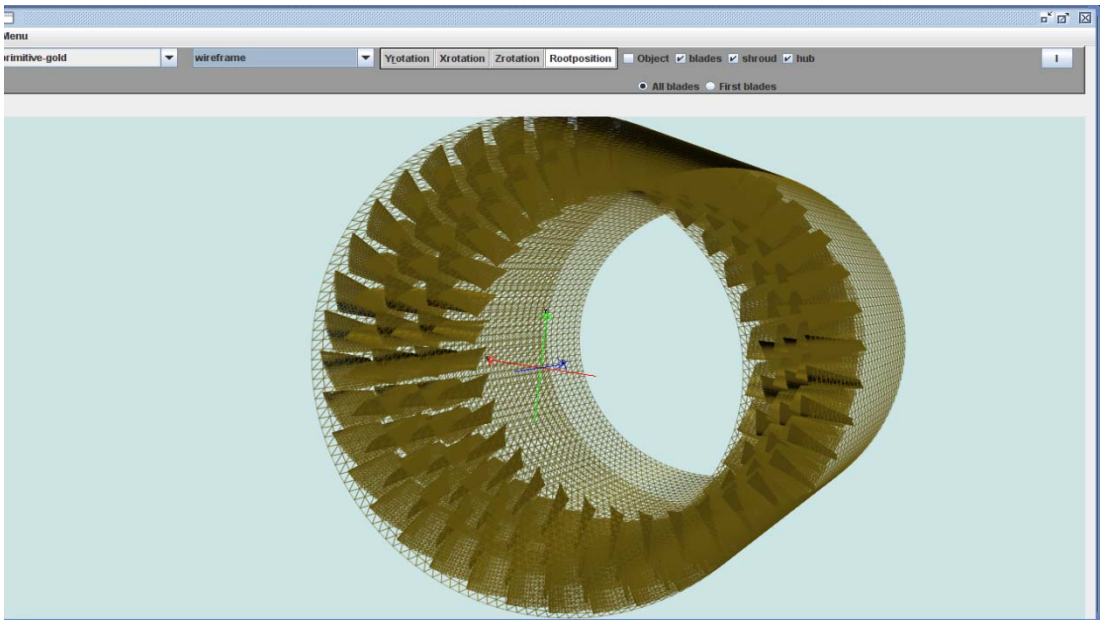
4.3. Συμπιεστής

Ο αξονικός συμπιεστής που θα παρουσιαστεί αποτελείται από 6 βαθμίδες πτερυγώσεων όπου η $1^{η}$,$3^{η}$ και $5^{η}$ έχουν ίδιο αριθμό και γεωμετρία πτερυγίων και η $2^{η}$, $4^{η}$ και $6^{η}$ ομοίως. Οι πτερυγώσεις έχουν δημιουργηθεί από συνολικά 213 Nurbs επιφάνειες και 2 επιπλέον επιφάνειες το κέλυφος και η πλήμνη. Στην εικόνα 4.8 παρουσιάζεται το σύνολο του συμπιεστή σε μορφή shading.



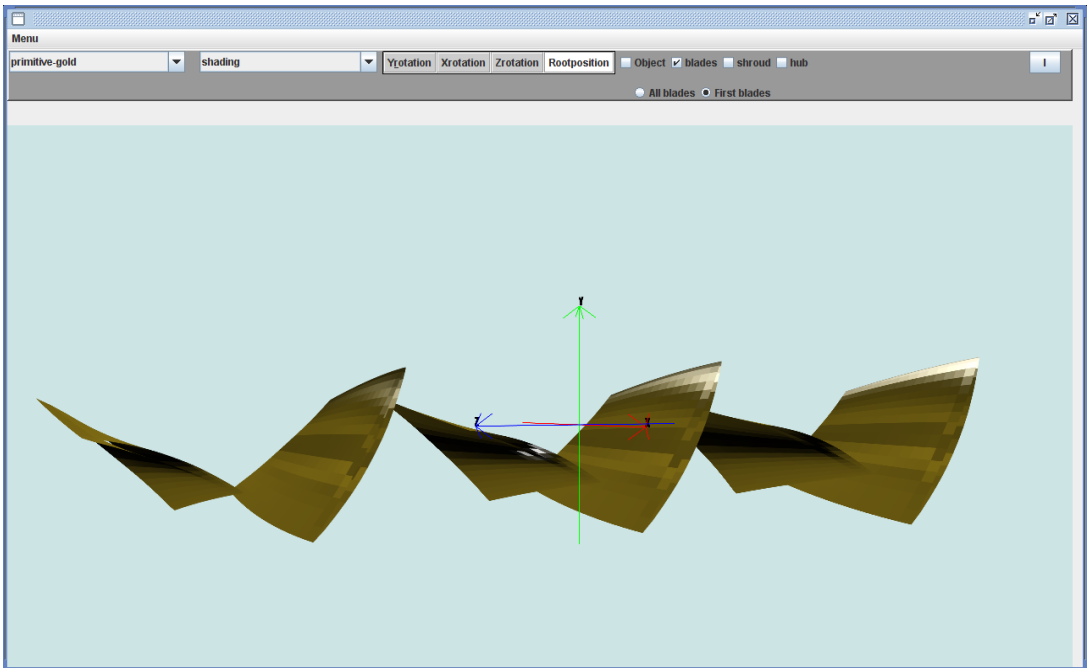
Εικόνα 4.8. Αξονικός συμπιεστής σε shading μορφή.

Στην εικόνα 4.9 παρουσιάζεται το σύνολο του συμπιεστή σε μορφή wireframe.



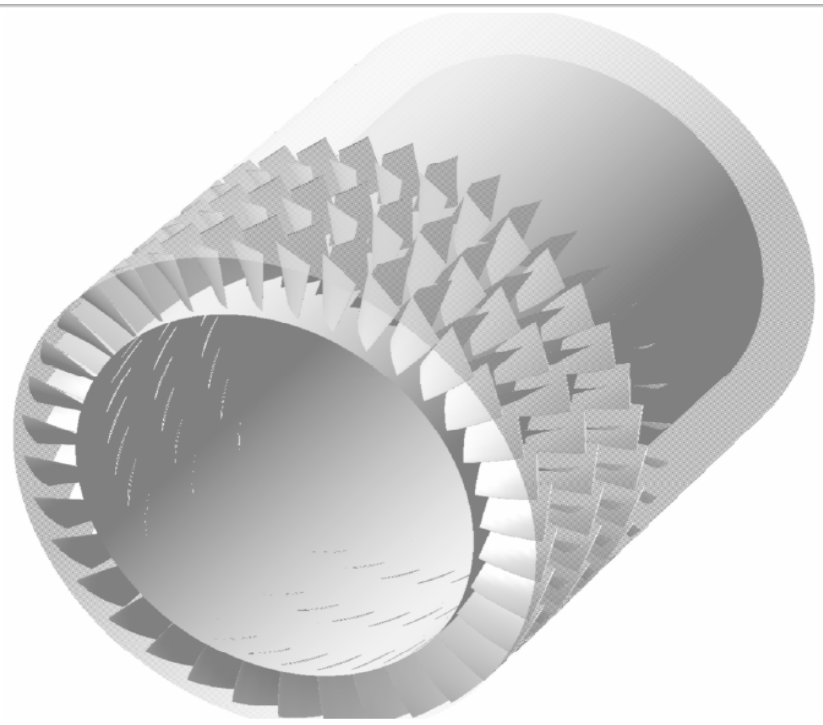
Εικόνα 4.9 Αξονικός συμπιεστής σε wireframe μορφή.

Στην εικόνα 4.10 παρουσιάζονται τα πρότυπα πτερύγια κάθε βαθμίδας πτερύγωσης. Το 1^o το 3^o και το 5^o είναι όμοια, καθώς και το 2^o , 4^o και 6^o . Η μορφή απεικόνισης είναι shading.

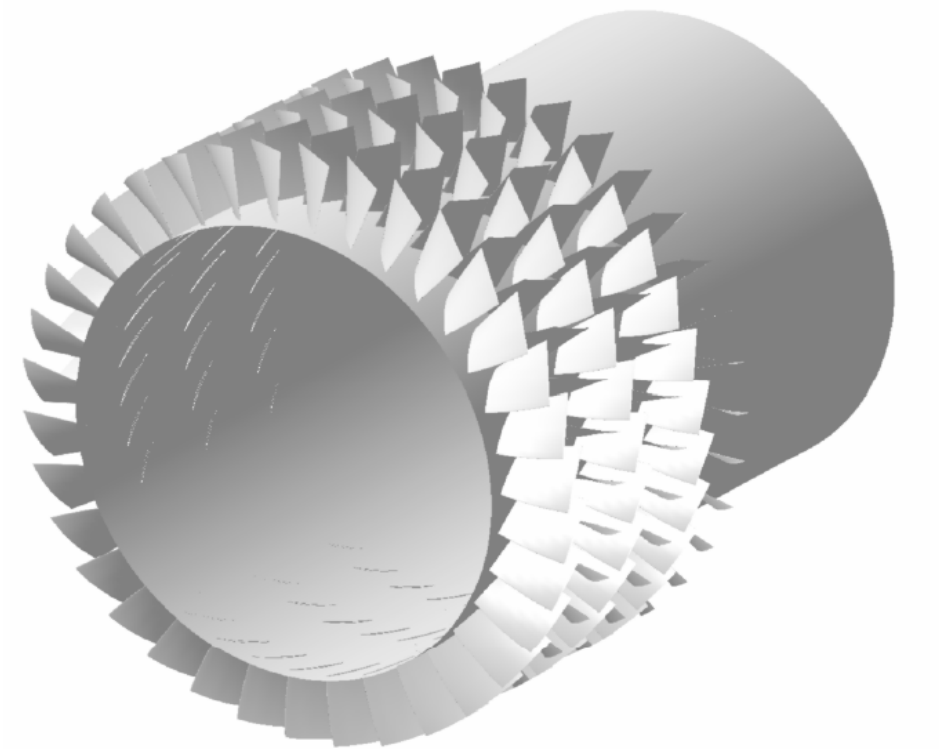


Εικόνα 4.10. Πρότυπα πτερύγια αξονικού συμπιεστή σε μορφή shading.

Παρουσιάζουμε τον ίδιο αξονικό συμπιεστή χρησιμοποιώντας το λογισμικό CATIA. Η σύγκριση οδηγεί στο συμπέρασμα σωστής και αρκετά ποιοτικής απεικόνισης του αντικειμένου στον Viewer3D. Στην εικόνα 4.11 παρουσιάζεται το σύνολο του αξονικού συμπιεστή ενώ στην εικόνα 4.12 παρουσιάζεται η πλήμνη και το σύνολο της πτερύγωσης του.



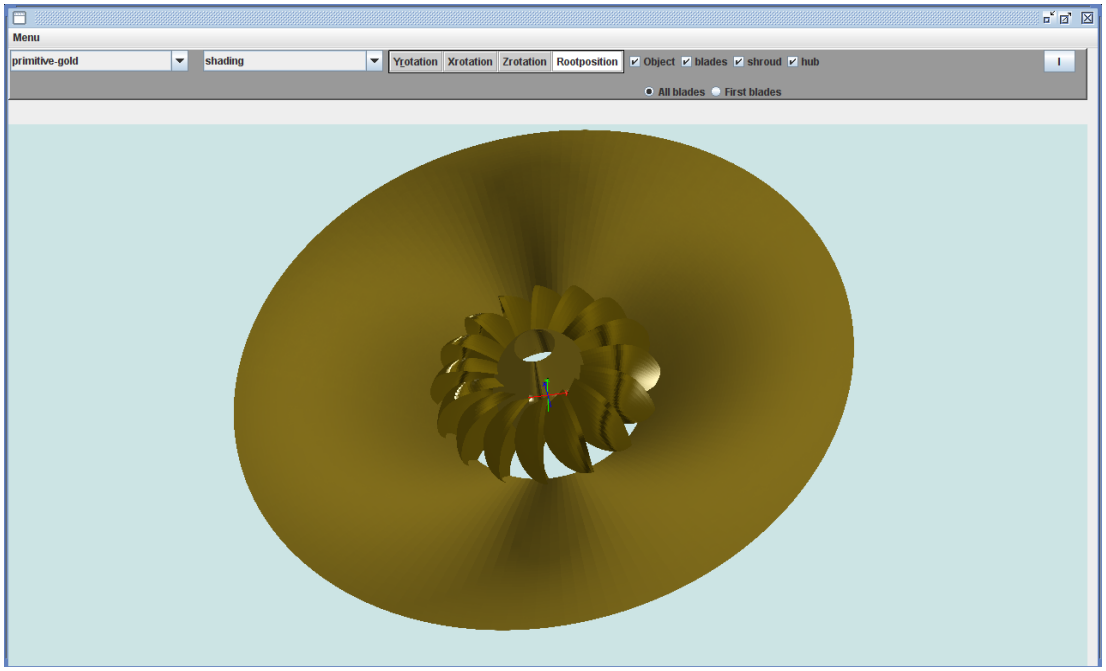
Εικόνα 4.11. Αξονικός συμπιεστής με χρήση του λογισμικού CATIA.(5)



Εικόνα 4.12. Πλήμνη και πτερύγωση αξονικού συμπιεστή με χρήση του λογισμικού CATIA.(5)

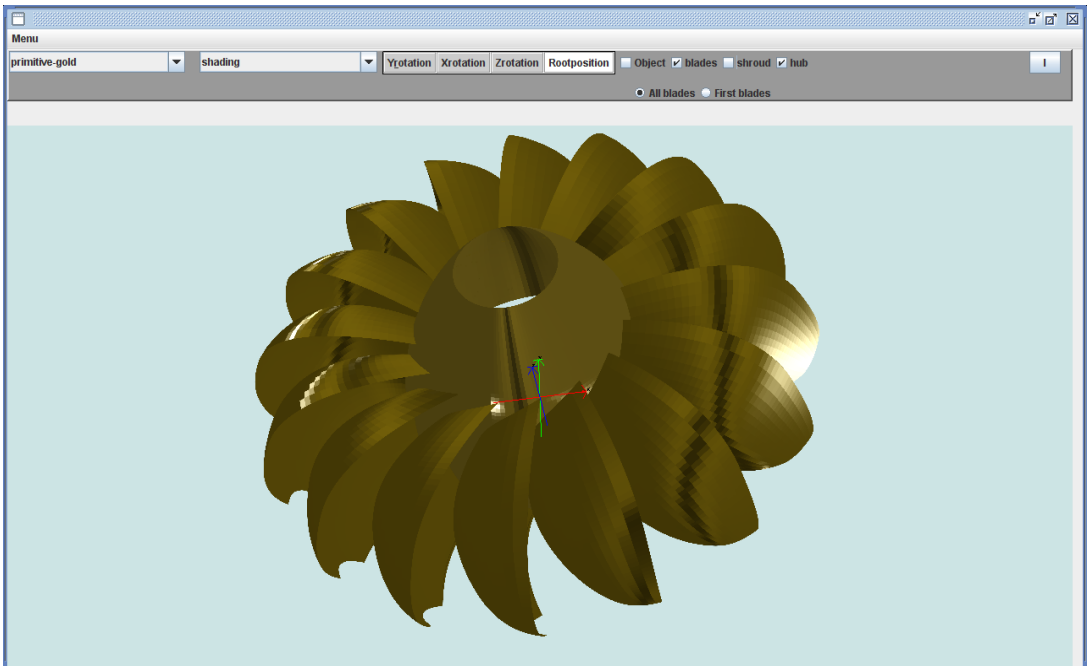
4.4. Υδροστρόβιλος τύπου Turgo

Ο υδροστρόβιλος που παρουσιάζεται αποτελείται από μία βαθμίδα 16 πτερυγίων, την πλήμνη και το κέλυφος. Χρησιμοποιήθηκε στις εικόνες αναφοράς του κεφαλαίου 3. Στην εικόνα 4.13 παρουσιάζεται το σύνολο του turgo. Η επιφάνεια του κελύφους είναι βοηθητική και δεν υπάρχει στην πραγματικότητα (δεν κατασκευάζεται).



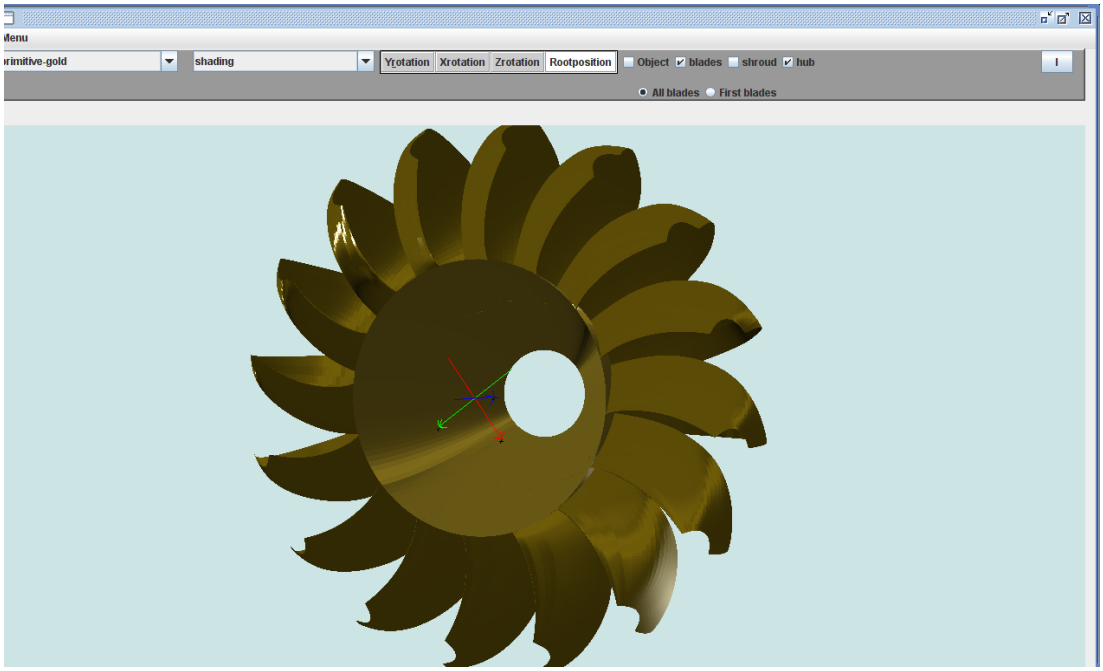
Εικόνα 4.13. Υδροστρόβιλος τύπου Turgo.

Στην εικόνα 4.14 παρουσιάζεται η πλήμνη και η πτερύγωση του υδροστρόβιλου.

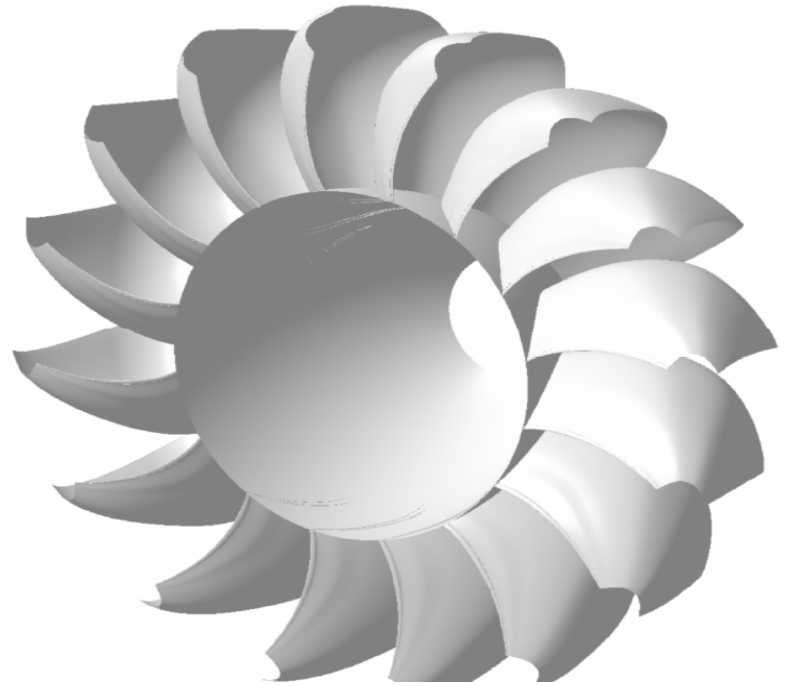


Εικόνα 4.14. Πλήμνη και πτερύγωση Υδροστρόβιλου τύπου Turgo.

Στις εικόνες 4.15 και 4.16 παρουσιάζεται η ίδια όψη της πλήμνης και της πτερύγωσης του υδροστρόβιλου, η πρώτη με χρήση του T4T και του Viewer3D και η δεύτερη με χρήση του λογισμικού CATIA.



Εικόνα 4.15. Πλήμνη και πτερύγωση Υδροστρόβιλου τύπου Turgo, Viewer3D.



Εικόνα 4.16. Πλήμνη και πτερύγωση Υδροστρόβιλου τύπου Turgo, λογισμικό CATIA.(5)

5

Μελλοντικές Επεκτάσεις

Στη παρούσα διπλωματική εργασία αναπτύχθηκε και παρουσιάστηκε ένα λογισμικό, το T4T Viewer3D, ικανό να απεικονίζει σε τρισδιάστατη μορφή πτερυγώσεις στροβιλομηχανών, αλλά και γεωμετρίες οποιουδήποτε σχήματος προερχόμενες από εξωτερικό αρχείο. Το λογισμικό αποτελεί μία συλλογή γεωμετρικών εργαλείων και εργαλείων απεικόνισης που συνδυάζονται κατάλληλα για την τριδιάστατη απεικόνιση των αντικειμένων που επιθυμούμε.

Οι μελλοντικές επεκτάσεις στο λογισμικό αυτό αφορούν τους εξής τομείς:

- **RENDERING.**Το rendering αφορά την ποιότητα της απεικόνισης σε συνδυασμό με την ταχύτητα της. Είναι λογικό όσο βελτιώνουμε την ποιότητα της απεικόνισης των αντικειμένων να μειώνεται η ταχύτητα, καθώς μεγαλύτερη ποιότητα επιφανειών αντιστοιχεί σε μεγαλύτερο νέφος σημείων ελέγχου της επιφάνειας, άρα περισσότερος υπολογιστικός φόρτος. Πρέπει να βρεθούν τα κατάλληλα εργαλεία και τεχνικές που με την βελτίωση της ποιότητας της απεικόνισης η ταχύτητα της να παραμένει ικανοποιητική.

- **ΔΥΝΑΜΙΚΟΣ ΕΛΕΓΧΟΣ ΣΗΜΕΙΩΝ ΕΛΕΓΧΟΥ.** Το λογισμικό που αναπτύξαμε δεν έχει την δυνατότητα άμεσης επέμβασης και τροποποίησης, κατά την απεικόνιση, των σημείων ελέγχου της γεωμετρίας (NURBS control points). Αυτή η προσθήκη θα προσδώσει νέα δυναμική στον Viewer3D καθώς η ικανότητα μετακίνησης των σημείων ελέγχου στην περιοχή απεικόνισης με την χρήση του ποντικιού θα διευκολύνει αρκετά τον χρήστη σε αλλαγές που επιθυμεί πάνω στα αντικείμενα, χωρίς να χρειαστεί να εκτελέσει το στάδιο της σχεδίασης από την αρχή.

- **ΔΗΜΙΟΥΡΓΊΑ ΥΠΟΛΟΓΙΣΤΙΚΟΥ ΠΛΕΓΜΑΤΟΣ.** Η επόμενη εξέλιξη που προτείνεται και βρίσκεται στα άμεσα σχέδια είναι η ενσωμάτωση εργαλείων κατασκευής και απεικόνισης υπολογιστικού πλέγματος για τη διακριτοποίηση του χωρίου της ροής. Το υπολογιστικό πλέγμα στη συνέχεια θα μπορεί να χρησιμοποιηθεί για την επίλυση της ροής με την χρήση κατάλληλου λογισμικού υπολογιστικής ρευστοδυναμικής.

Βιβλιογραφία

1. Bouvier, D.J., *Tutorial-Getting Started with Java3D API*, e-book by Sun Microsystems, 1999.
(http://java.sun.com/developer/onlineTraining/java3d/)

2. Dan Pilone, JAVA 3D, 2004.
(http://www2.sys-con.com/itsg/virtualcd/java/archives/0802/pilone/index.html)

3.DirectX Website:http://www.directX.org

4. Hortons, I, *Beginning Java 2 JDK*, 5 Edition, e-book by WROX, 2005.
(http://www.wrox.com/WileyCDA/WroxTitle/Ivor-Horton-s-Beginning-Java2-JDK-5-Edition.productCd-0764568744.html)

5. JAVAWORLD (http://www.javaworld.com)

6. Κοίνη Γ., Ανάπτυξη λογισμικού για την παραμετρική σχεδίαση πολυβάθμιων πτερυγώσεων συμπιεστών, Μεταπτυχιακή Διατριβή, Πολυτεχνείο Κρήτης, Τμήμα Μηχανικών Παραγωγής και Διοίκησης, 2007.

7. Oliver, J.H., *NURBS-Based Geometry for Integrated Structural Analysis,* NASA Lewis Research Center, Grant NAG3-1481, March 1997.

8. OpenGL Website: http://www.opengl.org

9. Piegl, L. & Tiller, W., *The NURBS Book*, Springer-Verlag, Berlin Heidelberg, 1995.

10. Sowizral, H., *Introduction to Programming with Java3D*, e-book by Sun Microsystems Inc., 2005 .
(http://www.sdsc.edu/~nadeau/Courses/Siggraph99/)

11. Sun Microsystems (http://www.sun.com)

12. Walsh, A. ,Gehringer, D., *Java 3D Jump Start*, e-book by Sun Microsystems, 2001.

13. Wikipedia (http://www.wikipedia.org).

14. Χαρίτος Δ., Μαρτάκος Δ., *Εικονική Πραγματικότητα*, Μεταπτυχιακό Πρόγραμμα Σπουδών, Β'Εξάμηνο, Τμήμα Πληροφορικής Πανεπιστημίου Αθηνών, 1999.

15. Χούσος Ν., *Οντοκεντρικός προγραμματισμός*, Προπτυχιακό Πρόγραμμα Σπουδών, Πολυτεχνειο Κρήτης, Τμήμα ΗΜΜΥ, 2001.