# ΘΑΛΗΣ – Πανεπιστήμιο Πειραιά
# Μεθοδολογικές προσεγγίσεις για τη μελέτη της ευστάθειας σε προβλήματα λήψης αποφάσεων με πολλαπλά κριτήρια

## Δ2 – Ανάπτυξη μέτρων αξιολόγησης ευστάθειας σε αναλυτικές-συνθετικές διαδικασίες

## Π2 – Τεχνική έκθεση (ανάπτυξη μέτρων αξιολόγησης ευστάθειας σε αναλυτικές-συνθετικές διαδικασίες)

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

# Στοιχεία παραδοτέου

**Δράση:** Δ2 – Ανάπτυξη μέτρων αξιολόγησης ευστάθειας σε αναλυτικές-συνθετικές διαδικασίες

**Τίτλος παραδοτέου:** Τεχνική έκθεση (ανάπτυξη μέτρων αξιολόγησης ευστάθειας σε αναλυτικές-συνθετικές διαδικασίες)

**Τύπος παραδοτέου:** S - PU

**Έκδοση:** 02

**Ημερομηνία:** 1 Μαρτίου 2013

**Υπεύθυνος σύνταξης:** Καθηγητής Ιωάννης Σίσκος

**Ομάδας σύνταξης:** Καθηγητής Διονύσης Γιαννακόπουλος
Καθηγητής Αθανάσιος Σπυριδάκος
Αναπληρωτής Καθηγητής Ευάγγελος Γρηγορούδης
Δρ. Νικόλαος Τσότσολας
Δρ. Ιώαννης Πολίτης
Νικόλαος Χριστοδουλάκης, MSc.
Prof. Christian Hurson
Γεωργία Μουριάδου, MSc.

# Περιεχόμενα

# 1. Εισαγωγή

Στόχος της Αναλυτικής – Συνθετικής Προσέγγισης είναι να εκτιμηθεί το μοντέλο προτιμήσεων του αποφασίζοντος μέσα από μια διαδικασία στην οποία αποκαλύπτονται και δομούνται οι αναλυτικές προτιμήσεις του αποφασίζοντος πάνω στα κριτήρια αξιολόγησης. Σύμφωνα με την αναλυτική συνθετική προσέγγιση, ο αποφασίζων εκφράζει τις σφαιρικές του προτιμήσεις σε ένα σύνολο εναλλακτικών ενεργειών (σύνολο αναφοράς) αξιολογημένων σε ένα συνεπές σύνολο κριτηρίων και με βάση αυτές προσεγγίζεται το μοντέλο της αθροιστικής χρησιμότητας (αξίας).

Οι μεθοδολογικές προσεγγίσεις της Αναλυτικής-Συνθετικής προσέγγισης καταλήγουν στην κατασκευή του παρακάτω Αθροιστικού Μοντέλου Χρησιμότητας (Keeney and Raiffa, 1976, Keeney, 1992) με βάση το οποίο δημιουργείται η κατάταξη των εναλλακτικών αποφάσεων:

$$U(\mathbf{g}) = \sum_{i=1}^{n} p_i u_i(g_i)$$

$$u(g_{i*}) = 0, \quad u(g_i^*) = 1 \text{ όπου } i=1,2,\ldots,n$$

$$\sum_{i=1}^{n} p_i = 1$$

$$p_i \geq 0, \text{ όπου } i=1,2,\ldots,n$$

όπου $\mathbf{g} = (g_1, g_2, \ldots, g_n)$ είναι η βαθμολόγηση των εναλλακτικών αποφάσεων στα $n$ κριτήρια $g_{i*}$ και $g_i^*$ είναι η λιγότερο και περισσότερο προτιμητέα τιμή του κριτηρίου $i$ αντίστοιχα και $u_i(g_i)$, $p_i$ είναι η συνάρτηση αξιών και η βαρύτητα του κριτηρίου $i$.

Η εκτίμηση των βαρών πραγματοποιείται με την επίλυση του παρακάτω γραμμικού προγράμματος (Jacquet-Lagreze and Siskos, 1982, Siskos, 1980) για την περίπτωση της μεθόδου UTA II.

$$[\min] F = \sum_{i=1}^{k} \sigma^+(a_i) + \sigma^-(a_i)$$

με περιορισμούς:

$$\begin{cases} \sum_{i=1}^{n} p_i u_i[g_i(a_m) + \sigma^+(a_m) - \sigma^-(a_m) - \sum_{i=1}^{n} p_i u_i[g_i(a_{m+1}) + \sigma^+(a_{m+1}) - \sigma^-(a_{m+1}) \geq \delta, \text{ αν } a_m \, \mathrm{P} \, a_{m+1} \\ \text{ή} \\ \sum_{i=1}^{n} p_i u_i[g_i(a_m) + \sigma^+(a_m) - \sigma^-(a_m) - \sum_{i=1}^{n} p_i u_i[g_i(a_{m+1}) + \sigma^+(a_{m+1}) - \sigma^-(a_{m+1}) = 0, \text{ αν } a_m \mathrm{I} a_{m+1} \\ \text{για } m = 1, 2, ..., k-1 \\ \sum_{i=1}^{n} p_i = 1 \\ p_i \geq 0, \quad \text{για } i = 1, 2, ..., n \\ \sigma^+(a_i) \geq 0, \quad \sigma^-(a_i) \geq 0, \quad \text{για } j = 1, 2, ..., k \end{cases}$$

όπου $\delta$ είναι ένας μικρός θετικός αριθμός, $g_i(a_m)$ είναι η τιμή της εναλλακτικής απόφασης στο $i$ κριτήριο, $u_i[g_i(a_m)]$ είναι η αντίστοιχη τιμή της συνάρτησης αξιών και $\sigma^+(a_j)$, $\sigma^-(a_j)$ είναι τα σφάλματα υπερ-υπο-εκτίμησης που αντιστοιχούν στην εναλλακτική απόφαση $a_j$.

Η επίλυση του πολυκριτηριακού γραμμικού προβλήματος μπορεί να δώσει:

1. Μία μόνο λύση, κυρίαρχα σε προβλήματα με υψηλή δόμηση, πράγμα το οποίο δεν συναντάται συχνά.
2. Άπειρες Λύσεις, που είναι και το σύνηθες σε προβλήματα με χαμηλή δόμηση.
3. Καμία λύση.

Στην περίπτωση των άπειρων λύσεων, οι μεθοδολογικές προσεγγίσεις της Αναλυτικής Συνθετικής Προσέγγισης, στο πλαίσιο της διαχείρισης της παρουσίασης και της περαιτέρω επεξεργασίας του προβλήματος, ακολουθούν την παρακάτω προσέγγιση διαχείρισης των αποτελεσμάτων:

Α) Υπολογίζονται τα μοντέλα προτιμήσεων που μεγιστοποιούν (ή και ελαχιστοποιούν) καθένα από τα επιμέρους κριτήρια. Ακολούθως υπολογίζεται η μέση λύση (Κεντροβαρής) που προέρχεται από τον συγκερασμό λύσεων της μεγιστοποίησης (ή και ελαχιστοποίησης) των κριτηρίων.

Εκτός από τους αλγορίθμους αναζήτησης λύσεων με επίλυση ΓΠ μεγιστοποίησης ή/και ελαχιστοποίησης έχουν προταθεί και αλγόριθμοι αναλυτικής αναζήτησης του συνόλου των λύσεων που ισοδυναμούν με υπολογισμό των κορυφών του υπερπολυέδρου των πολλαπλών βέλτιστων λύσεων. Ένας τέτοιος αλγόριθμος είναι αυτός των Manas-Nedoma που εγγυάται την εύρεση όλων κορυφών υλοποιώντας ένα Hamiltonian Path (Siskos, 1984).

Αξίζει να μελετηθεί πάντως αν τα αποτελέσματα που προκύπτουν από την επίλυση μόνο των προς μεγιστοποίηση ΓΠ (προσέγγιση που υλοποιούν κάποια ΣΥΑ) είναι ίδια με τα αποτελέσματα που προκύπτουν από την μεγιστοποίηση και ελαχιστοποίηση των κριτηρίων ή ακόμα και από την μεγιστοποίηση – ελαχιστοποίηση των μερικών αξιών ανά σημείο της κλίμακας, ή τέλος με τα αποτελέσματα που προκύπτουν από την επίλυση αναλυτικών αλγορίθμων όπως των Manas-Nedoma.

Β) Η ανάλυση των αποτελεσμάτων εστιάζεται στην παρουσίαση της μέσης (κεντροβαρούς) λύσης και του εύρους (min, max) των βαρών των κριτηρίων.

Γ) Στις αναδράσεις παρέχεται η δυνατότητα να επιλέξει ο αποφασίζων ή ο αναλυτής αποφάσεων το ή τα κριτήρια το οποία επιθυμεί να μεγιστοποιήσει και εκτιμάται το νέο

σταθμισμένο μοντέλο προτιμήσεων στο οποίο τα επιλεγμένα κριτήρια μετέχουν με τη λύση που τα μεγιστοποιεί.

Στη μελέτη αυτή εστιάζουμε στην ανάλυση του συνόλου των λύσεων του Πολυκριτηριακού Γραμμικού Προβλήματος στην περίπτωση που έχουν προσδιορισθεί άπειρες λύσεις και την αξιοποίηση των αποτελεσμάτων της ανάλυσης του ν-διάστατου χώρου, προκειμένου να διευκολυνθεί η διαδικασία προσέγγισης ενός μοντέλου προτιμήσεων, όσο το δυνατόν περισσότερο συνεπούς προς τις πραγματικές προτιμήσεις του αποφασίζοντος μέσα από διαδραστικές διαδικασίες και διαλόγους με τον αποφασίζοντα .

# 2. Ευστάθεια του Μοντέλου προτιμήσεων

## 2.1 Γενικά

Η ευστάθεια του μοντέλου προτιμήσεων επηρεάζεται τόσο από τις εκφρασμένες προτιμήσεις του αποφασίζοντος όσο και από τις επιλογές που έχουν γίνει στο πλαίσιο των διαδικασιών κατασκευής του μοντέλου προτιμήσεων (μοντελοποίηση κριτηρίων, βαθμολόγηση των εναλλακτικών στα κριτήρια, επιλογή συνόλου αναφοράς). Σε κάθε περίπτωση επίλυσης του Πολυκριτήριου Γραμμικού προβλήματος απαιτείται να εκτιμηθεί η ευστάθεια του ν-διάστατου υποχώρου των λύσεων. Ένας σημαντικός στόχος είναι να προσδιορισθούν δείκτες που να μπορούν να εκφράσουν το βαθμό της ευστάθειας του ν-διάστατου υποχώρου. Αυτοί μπορεί να είναι:

1. Οι μέγιστες και ελάχιστες τιμές της βαρύτητας των κριτηρίων που προκύπτουν από την ανάλυση μεταβελτιστοποίησης. Οι τιμές αυτές μπορούν να δείξουν το εύρος των τιμών που μπορεί να πάρει η βαρύτητα του κριτηρίου. Στην περίπτωση που το εύρος είναι μεγάλο (για πολλά από τα κριτήρια) τότε έχουμε και χαμηλό βαθμό ευστάθειας στο εκτιμημένο μοντέλο προτιμήσεων. Οι μέγιστες και ελάχιστες τιμές των κριτηρίων μπορούν να δώσουν το ν-διάστατο ορθογώνιο υπερ-πολύεδρο το οποίο περιγράφει το υπερ-πολύεδρο των λύσεων του γραμμικού προβλήματος.

   $\mu_i = (\mathrm{Max}(p_i) - \mathrm{Min}(p_i))$, i=1,3,…n., $p_i$ η βαρύτητα του κριτηρίου $i$

   Η διερεύνηση του δείκτη μπορεί να δώσε μια πρώτη εικόνα της δομής του συνόλου των λύσεων, παρέχοντας τα όρια στα οποία μπορούν να κινηθούν οι τιμές των βαρών των κριτηρίων από την επίλυση του γραμμικού προβλήματος.

2. Οι κανονικοποιημένες Ευκλείδειες Αποστάσεις μεταξύ των κορυφών του υπερ-πολύεδρου (ο υπολογισμός των κορυφών μπορεί να γίνει με την επίλυση αλγορίθμου Manas-Nedoma). Αν οι αποστάσεις είναι μικρές και δεν υπάρχουν σημαντικές διαφορές μεταξύ τους , τότε μπορούμε να ισχυριστούμε ότι υπάρχει υψηλός βαθμός ευστάθειας του εκτιμημένου ν-διάστατου χώρου του μοντέλου προτιμήσεων. Σημαντικές διαφοροποιήσεις στις αποστάσεις σημαίνει ότι το υπερ-πολύεδρο εμφανίζει ένα ακανόνιστο σχήμα και συνεπώς η Κεντροβαρική λύση δεν αντιστοιχεί σε ένα αντιπροσωπευτικό μοντέλο προτιμήσεων και απαιτείται περαιτέρω διερεύνηση.

   $DK_iK_j$ $i,j=1,2,\ldots,\rho$ και $i \neq j$ (Κανονικοποιημένες Ευκλείδειες Αποστάσεις των Κορυφών)

   Στους παραπάνω δείκτες είναι χρήσιμη και η περαιτέρω επεξεργασία τους με τον υπολογισμό της μέγιστης, ελάχιστης, μέσης τιμής και της τυπικής απόκλισης. Ένας δείκτης που θα μπορούσε να υπολογισθεί είναι ο μέσος δείκτης ευστάθειας (Average Stability Index - ASI) για κάθε κριτήριο $i$ ο οποίος θα μπορούσε να οριστεί ως η μέση τιμή της κανονικοποιημένης τυπικής απόκλισης των τιμών του παραμέτρων $k$ του κριτηρίου $i$ του προβλήματος που θα προκύπτουν κατά τη μεταβελτιστοποίηση:

$$ASI(i) = 1 - \frac{1}{n_{par}} \sum_{k=1}^{n_{par}} \frac{S_k}{Norm}$$

όπου $S_k$ η τυπική απόκλιση των εκτιμώμενων τιμών του παραμέτρων $k$ του κριτηρίου $i$, $n_{par}$ ο αριθμός των παραμέτρων και *Norm* ένας συντελεστής κανονικοποίησης, τέτοιος ώστε να επιτρέπει στο δείκτη ASI να λάβει τιμές στο διάστημα [0,1]. Αναλυτικότερα ο τύπος γράφεται ως εξής:

$$ASI(i) = 1 - \frac{1}{n_{par}} \frac{\sum_{k=1}^{n_{par}} \sqrt{\left( n_{sol} \left( \sum_{j=1}^{n_{sol}} \left( par_k^j \right)^2 \right) - \left( \sum_{j=1}^{n_{sol}} par_k^j \right)^2 \right)}}{\frac{n_{sol}}{n_{par}} \sqrt{\left( n_{par} - 1 \right)}}$$

Όπου, $par_k^j$ η τιμή της παραμέτρου $k$ που προκύπτει από την επίλυση του $j$ γραμμικού προγράμματος μεταβελτιστοποίησης και τέλος $n_{sol}$ ο συνολικός αριθμός των προς επίλυση γραμμικών προγραμμάτων στα πλαίσια της ανάλυσης μεταβελτιστοποίησης.

Ειδικότερα στην περίπτωση που οι παράμετροι είναι οι μερικές χρησιμότητες του κριτηρίου $i$ για κάθε σημείο της κλίμακας του κριτηρίου $i$, η οποία κλίμακα έχει χωριστεί σε ($\alpha_i$-1) ίσα διαστήματα, έχουμε $n_{par}$=$a_i$-1. Για παράδειγμα, $u_k^j$ είναι η μερική χρησιμότητα του σημείου $k$ της κλίμακας του κριτηρίου, με $k$=1,…,$a_i$-1, που προκύπτει από την επίλυση του $j$ γραμμικού προγράμματος μεταβελτιστοποίησης. Σε αυτή την περίπτωση η παραπάνω σχέση γράφεται ως εξής:

$$ASI(i) = 1 - \frac{1}{\alpha_i - 1} \frac{\sum_{k=1}^{n_{par}} \sqrt{\left( n_{sol} \left( \sum_{j=1}^{n_{sol}} \left( u_k^j \right)^2 \right) - \left( \sum_{j=1}^{n_{sol}} u_k^j \right)^2 \right)}}{\frac{n_{sol}}{\alpha_i - 1} \sqrt{\left( \alpha_i - 2 \right)}}$$

3. Ο Όγκος του υπερ-πολύεδρου που προκύπτει από την επίλυση του αλγόριθμου Manas Nedoma. Ο Όγκος του Υπερ-πολύεδρου (0 V 1) αποτελεί έναν δείκτη της Ευστάθειας του εκτιμημένου μοντέλου προτιμήσεων, ο οποίος συνδυαζόμενος με τις αποστάσεις τω κορυφών του υπερ-πολυέδρου από το βαρύκεντρο (μέση τιμή, τυπική απόκλιση, μέγιστη και ελάχιστη τιμή) δίνουν μια συνολική εικόνα της κανονικότητας ή όχι του υπερ-πολυέδρου.

Για έναν πλήρη έλεγχο της Ευστάθειας του Εκτιμημένου Μοντέλου Προτιμήσεων διαθέσιμα είναι τα παρακάτω στοιχεία:

a) Δεδομένα Μοντελοποίησης του Προβλήματος Απόφασης:

Εναλλακτικές Ενέργειες A={$a_i$, $i$=1,2,…,$n$}

Κριτήρια: G = {$g_j$, $j$=1,2,…,$k$}

Βαθμολόγηση των Εναλλακτικών στα Κριτήρια: $g_{ij}$, $i$=1,2,..,$n$, $j$=1,2,…,$k$

Σύνολο Αναφοράς A'={$a'_i$, $i$=1,2,…,$n$} υποσύνολο του A

Κατάταξη Εναλλακτικών του Συνόλου Αναφοράς: $\mathrm{R}(a'_i),\ a'_i \in A'$

b) Δεδομένα Μοντέλου Προτιμήσεων

Συναρτήσεις Αξιών Κριτηρίων $u_j=u_j(g_j)$, $j=1,2,\ldots k.$, $u_j(g_{j*})=0$ και $u_j(g_j^*)=1$

Βαρύτητες Κριτηρίων: $p_j$, $j=1,2,\ldots,k$

Μέγιστες - Ελάχιστες τιμές Βαρύτητας Κριτηρίων $Pmax_j$, $Pmin_j$, $j=1,2,\ldots,k$

Σφάλματα υπερ-υπο-εκτίμησης $\sigma_i^+$, $\sigma_i^-$, όπου $i=1,2,..,n$

c) Δεδομένα που προκύπτουν από την αξιοποίηση του μοντέλου προτιμήσεων

Μερικές Αξίες Εναλλακτικών Ενεργειών στα Κριτήρια $u_{ij} = u_i(g_j)$

Ολικές Αξίες Εναλλακτικών Ενεργειών: $U(\mathbf{g}) = \sum_{i=1}^{n} p_i u_i(g_i)$

## 2.2 Αίτια της Αστάθειας του Μοντέλου προτιμήσεων

Ένα σημαντικό ερώτημα που τίθεται είναι τα αίτια που προκαλούν την αστάθεια στο εκτιμημένο μοντέλο προτιμήσεων.

- Αστάθεια μπορεί να προκαλείται από την αβεβαιότητα που μπορεί να εμφανίζεται σε παραμέτρους του προβλήματος απόφασης όπως είναι η βαθμολόγηση των εναλλακτικών στα κριτήρια, η φύση των κριτηρίων, η επιλογή του συνόλου αναφοράς, κ.ά.
- Η αστάθεια στο εκτιμημένο μοντέλο προτιμήσεων μπορεί να εκφράζει αυτό το οποίο πραγματικά έχει στο μυαλό του ο αποφασίζων. Από τις εκφρασμένες σφαιρικές προτιμήσεις του αποφασίζοντος προκύπτουν διαστήματα στις βαρύτητες των κριτηρίων που μπορεί να εκφράζουν αυτό που στη δεδομένη χρονική στιγμή αντανακλά στις προτιμήσεις του.
- Ένας άλλος παράγοντας που μπορεί να δημιουργεί χαμηλό βαθμό ευστάθειας είναι οι δυσκολίες που μπορεί να υπάρχουν στη διαδικασία άντλησης των προτιμήσεων του αποφασίζοντος. Για πολλούς λόγους οι αντιδράσεις του αποφασίζοντος στις διαδραστικές διαδικασίες εξόρυξης των προτιμήσεων του μπορεί να μην ανταποκρίνονται στις πραγματικές συνθήκες του προβλήματος.
- Τελευταία, αλλά εξίσου σημαντική είναι η περίπτωση της μη επαρκούς μοντελοποίησης του προβλήματος, δηλαδή ης ύπαρξης αδυναμιών στην μοντελοποίηση της συνεπούς οικογένειας των κριτηρίων, στην αξιολόγηση των εναλλακτικών στα κριτήρια στην επιλογή του συνόλου αναφοράς κ.ά.

Κυρίαρχο στοιχείο του ελέγχου της ευστάθειας είναι κι ο προσδιορισμός της αιτίας ή των αιτιών που παράγουν την αστάθεια όπως και ο βαθμός που την επηρεάζουν. Στην αναζήτηση αυτή και όσον αφορά το τελευταίο μπορούν να εξετασθούν τα παρακάτω:

1. Πόσο ορθολογικές είναι οι σφαιρικές προτιμήσεις του αποφασίζοντος (προδιάταξη) στο σύνολο αναφοράς;

   Αν η προδιάταξη δεν έχει προέλθει από μια ορθολογική έκφραση των προτιμήσεων τότε υπάρχει το ενδεχόμενο σε εντοπισμένα κριτήρια να εμφανίζεται μεγάλη

απόκλιση ανάμεσα στις μέγιστες και ελάχιστες τιμές της βαρύτητας των κριτηρίων αυτών. Συνεπώς, η ύπαρξη μεγάλου εύρους μεταξύ της μέγιστης και ελάχιστης τιμής του κριτηρίου αποτελεί σημείο περαιτέρω διερεύνησης της δομής της προδιάταξης του αποφασίζοντος και της αμφισβήτησης της μέσω ενός διαλόγου που θα πρέπει να αναπτυχθεί με τον αποφασίζοντα.

2. Πόσο καλά γνωρίζει ο Αποφασίζων τις Εναλλακτικές Ενέργειες του Συνόλου Αναφοράς;

   Αν ο αποφασίζων δεν γνωρίζει καλά τις εναλλακτικές ενέργειες που έχουν επιλεγεί στο σύνολο αναφοράς τότε είναι πιθανό να εμφανίζονται ασυνέπειες στην κατάταξη που έχει δώσει. Αυτό είναι λογικό να δώσει λύσεις στο γραμμικό πρόβλημα με χαμηλή ευστάθεια. Διερεύνηση του βαθμού γνώσης των εναλλακτικών του συνόλου αναφοράς και της προδιάταξης μπορεί να δώσει χρήσιμες πληροφορίες και γνώση για την αναθεώρηση του συνόλου αναφοράς.

3. Οι Εναλλακτικές Ενέργειες είναι αντιπροσωπευτικές του χώρου των Αποφάσεων;

   Όπως έχει ήδη αναφερθεί το σύνολο αναφοράς θα πρέπει να καλύπτει δυο συνθήκες. Η πρώτη αφορά αυτό το οποίο αναφέρθηκε στα προηγούμενα: Οι εναλλακτικές θα πρέπει να είναι οικίες στον αποφασίζοντα, έτσι ώστε να μπορεί να εκφράσει προτίμηση. Η δεύτερη συνθήκη αφορά το βαθμό κάλυψης του χώρου των αποφάσεων. Οι εναλλακτικές ενέργειες του Συνόλου Αναφοράς θα πρέπει να είναι αντιπροσωπευτικές του χώρου των Αποφάσεων. Επιλογή Εναλλακτικών Ενεργειών από ορισμένες μόνο περιοχές του χώρου των αποφάσεων, μπορεί να οδηγήσει σε λύσεις που δεν έχουν ενσωματώσει όλο το φάσμα των τιμών σε ένα κριτήριο και συνεπώς να μην είναι αντιπροσωπευτικές.

4. Είναι συνεπής ή οικογένεια κριτηρίων;

   Η έλλειψη ενός η περισσότερων κριτηρίων ή η ανεπαρκής κάλυψη των συνθηκών της συνέπειας της οικογένειας των κριτηρίων μπορεί να οδηγήσει σε χαμηλή ευστάθεια και σε μεγάλο εύρος μεταξύ των μέγιστων και ελάχιστων τιμών της βαρύτητας των κριτηρίων. Ξεκινώντας από τα κριτήρια με την μεγαλύτερη απόκλιση και εστιάζοντας στη δομή και τον τρόπο με τον οποίο έχουν κατασκευαστεί τα κριτήρια. Σημεία που επίσης θα πρέπει να εξεταστούν είναι αν υπάρχουν κριτήρια που είναι εξαρτημένα μεταξύ τους (επικάλυψη) ή αν υπάρχουν σημεία θεώρησης του προβλήματος που δεν έχουν εκφραστεί επαρκώς στα κριτήρια.

5. Η βαθμολόγηση των εναλλακτικών στα κριτήρια ανταποκρίνεται στην πραγματική φύση τους;

   Στο πλαίσιο της εφαρμογής των μεθόδων σε πολλά προβλήματα η βαθμολόγηση των εναλλακτικών στα κριτήρια γίνεται με βάση κάποιες παραδοχές ή μετρήσεις στις οποίες δεν μπορούμε να έχουμε την επιθυμητή ακρίβεια. Με την ανάλυση των δεδομένων και ανάλογα με τις εναλλακτικές που μετέχουν στο σύνολο αναφοράς θα πρέπει να επανεξετασθεί η βαθμολόγηση των εναλλακτικών στα κριτήρια.

Σε κάθε περίπτωση, ακόμη και όταν έχει προσεγγισθεί ένα ευσταθές μοντέλο προτιμήσεων, ο έλεγχος θα πρέπει να εστιασθεί στο βαθμό που αυτό ανταποκρίνεται στις πραγματικές προτιμήσεις του αποφασίζοντος. Είναι αξιοσημείωτο ότι ακόμη και όταν το εκτιμημένο μοντέλο προτιμήσεων παρουσιάζει ευστάθεια και δεν έχει ασυμφωνίες με τις αρχικές προτιμήσεις του αποφασίζοντος, δεν μπορούμε να πούμε με βεβαιότητα ότι αντιπροσωπεύει

απόλυτα τον αποφασίζοντα. Θα πρέπει να γίνει ένας διάλογος προκειμένου να αντληθούν περισσότερες πληροφορίες σχετικά με:

- την ιεράρχηση της βαρύτητας των κριτηρίων ξεκινώντας από την κεντροβαρή λύση, σε περίπτωση που δεν την αποδέχεται. Στην περίπτωση αυτή οδηγούμαστε σε μια μετα-μετα-βέλτιστη ανάλυση με την επίλυση γραμμικού προγράμματος. Η άντληση των πληροφοριών που αφορούν τις προτιμήσεις του αποφασίζοντας σχετικά με την ιεράρχηση των κριτηρίων μπορούν να γίνουν έμμεσα με την αξιοποίηση πραγματικών ή εικονικών λύσεων. Η συγκεκριμένη ανάδραση θα πρέπει να εξεταστεί περεταίρω ως προς τη σύνδεσή της με τη βασική φιλοσοφία της αναλυτικής-συνθετικής προσέγγισης.
- το ρόλο και την τιμή του δ στη λογική ότι αυτό αποτελεί κατώφλι σημαντικής διαφοροποίησης δυο διαδοχικών εναλλακτικών μη ισοδύναμων λύσεων στην κατάταξη
- σε περίπτωση προβληματικής β, αναζήτηση λύσης συμβατής με τις προτιμήσεις του αποφασίζοντα που μεγιστοποιεί τις διαφορές χρησιμότητας μεταξύ των εναλλακτικών προκειμένου να προκύψουν αντίστοιχες κλάσεις ισοδυναμίας
- στην άντληση περισσότερων πληροφοριών όσον αφορά στο εύρος των τιμών των βαρυτήτων των κριτηρίων με έμμεσο τρόπο χρησιμοποιώντας πραγματικές ή εικονικές εναλλακτικές αποφάσεις (διεύρυνση του συνόλου αναφοράς) και ζητώντας να εκφράσει τις προτιμήσεις του ο αποφασίζων.

# 3. Ενέργειες για την αύξηση της Ευστάθειας του μοντέλου προτιμήσεων

Η αύξηση της Ευστάθειας του Μοντέλου Προτιμήσεων μπορεί να γίνει με δυο εναλλακτικές προσεγγίσεις.

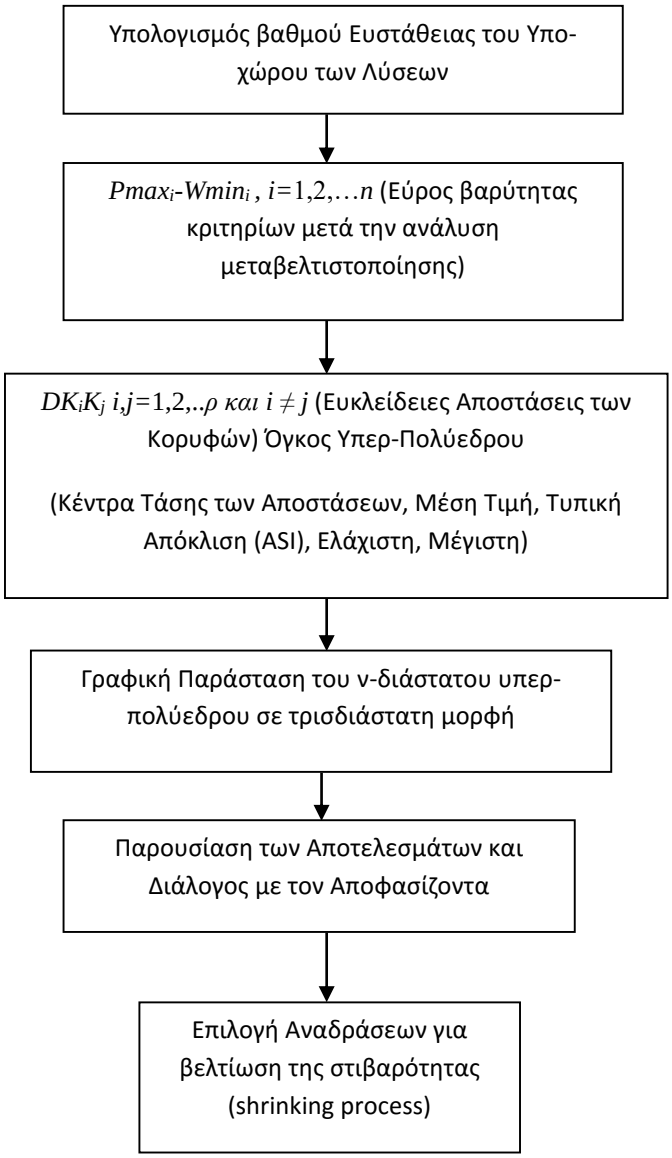
Η πρώτη αφορά την περαιτέρω εκλέπτυνση του υπερ-πολυέδρου μέσα από ένα διάλογο με τον αποφασίζοντα (π.χ. να δώσει περισσότερες πληροφορίες σχετικά με τις προτιμήσεις του στα κριτήρια), έτσι ώστε να οδηγηθούμε σε ένα περισσότερο κανονικοποιημένο υπερ-πολύεδρο μικρότερο του αρχικού. Για να γίνει αυτό εφικτό θα πρέπει να έχουμε μια ανάλυση των αποτελεσμάτων στη γλώσσα του αποφασίζοντος, έτσι ώστε αυτός να μπορεί να εκφράσει από μια γνωστή θέση τις περισσότερο εστιασμένες προτιμήσεις. Ουσιαστικά, αφορά μια νέα ανάδραση (shrinking process) μέσα από την οποία στόχος είναι να προσδιορισθεί ένα περισσότερο στιβαρό μοντέλο. Η τεκμηρίωση της βελτίωσης της στιβαρότητας του μοντέλου θα γίνει μέσω του ελέγχου βελτίωσης των δεικτών ευστάθειας που παρουσιάστηκαν νωρίτερα.

Παραδείγματα:

i. Να ζητηθεί από τον αποφασίζοντα να δώσει περισσότερο εστιασμένες πληροφορίες σχετικά με τις προτεραιότητές του στα κριτήρια και στη συνέχεια να κατασκευασθεί ένα γραμμικό μοντέλο που θα οδηγήσει σε περιορισμό του υπερ-πολυέδρου.
ii. Να ζητηθεί από τον αποφασίζοντα να δώσει για συγκεκριμένα κριτήρια το εύρος των τιμών των βαρών των κριτηρίων με βάση τις ελάχιστες και μέγιστες εκτιμημένες τιμές. Αυτό μπορεί να γίνει άμεσα ή έμμεσα με ερωτήσεις σύγκρισης.

Η δεύτερη προσέγγιση αφορά αναδράσεις που μπορούν να γίνουν σε όλα τα βήματα της διαδικασία προσέγγισης του μοντέλου προτιμήσεων. Προϋπόθεση για να προχωρήσουμε αποτελεί η διεξοδική παρουσίαση και ανάλυση του εκτιμημένου υπερ-πολυέδρου των λύσεων του γραμμικού προβλήματος. Από την ανάλυση μπορεί να προκύψει μια ή και περισσότερες αναδράσεις που αφορούν τα στάδια της κατασκευής του μοντέλου προτιμήσεων, δηλαδή: α) τη μοντελοποίηση των κριτηρίων, β) την βαθμολόγηση των εναλλακτικών στα κριτήρια γ) την επιλογή του Συνόλου Αναφοράς, δ) την προδιάταξη των εναλλακτικών (σφαιρικές προτιμήσεις του Αποφασίζοντος).

Στο Διάγραμμα που ακολουθεί δίνονται συνοπτικά τα βήματα της διαδικασίας εκτίμησης της στιβαρότητας του μοντέλου προτιμήσεων και των αναδράσεων για την αύξηση της στιβαρότητας του μοντέλου προτιμήσεων.




**Σχήμα 3.1: Διαδικασία εκτίμησης της στιβαρότητας του μοντέλου προτιμήσεων**

# 4. Το Σύστημα RAVI (Robustness Analysis with Visual and Interactive approaches)

## 4.1 Λειτουργικότητα

Το Λογισμικό που αναπτύσσεται παρέχει τη δυνατότητα της τρισδιάστατης παρουσίασης του υπερ-πολύεδρου των λύσεων του Γραμμικού Προβλήματος με πολλούς εναλλακτικούς τρόπους. Η λειτουργία του λογισμικού θα περιλαμβάνει: α) την παρουσίαση του Υπερ-Πολύεδρου των λύσεων και β) την διαδραστική παρουσίαση του Μοντέλου προτιμήσεων με βάση στοχευμένες επιλογές του χειριστή.

Είσοδος για το λογισμικό θα είναι: α) οι κορυφές του Υπερ-πολύεδρου όπως θα έχουν προκύψει από την επίλυση των Γραμμικών Προβλημάτων κατά το στάδιο της μεταβελτιστοποίησης και β) τα κριτήρια, οι εναλλακτικές και η βαθμολόγησή τους στα κριτήρια (στην περίπτωση της UTA II και οι συναρτήσεις μερικών αξιών),

## 4.2 Παρουσίαση του Μοντέλου

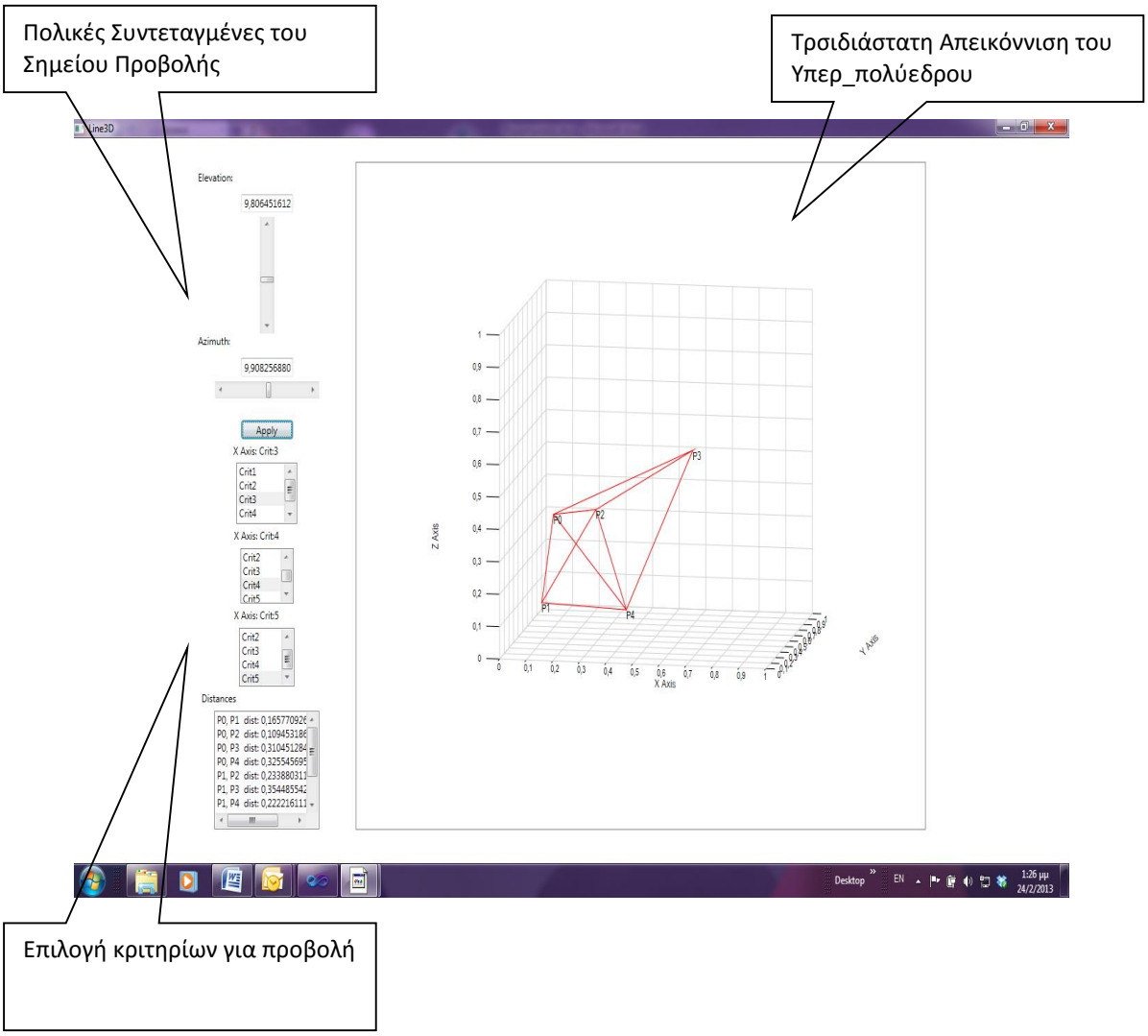
Ο Αναλυτής Αποφάσεων θα μπορεί να επιλέγει κάθε φορά τρεις μεταβλητές που θα αντιστοιχίζονται σε τρεις άξονες συντεταγμένων και να παρουσιάζει σε τρισδιάστατη μορφή το μέρος του υπερ-πολυέδρου που αφορά τις τρεις μεταβλητές του (κριτήρια). Χρησιμοποιούνται πολικές συντεταγμένες για την παρουσίαση του στερεού (ανύψωση και αζιμούθιο) και με αυτόν τον τρόπο ο χειριστής μπορεί να έχει θέα του στερεού από όλα τα σημεία θέασης.

Στις κορυφές του υπερ-πολυέδρου θα εμφανίζονται και το διάνυσμα των συντεταγμένων του $(x_1,x_2,x_3,x_4,...,x_n)$. Σε μορφή πίνακα θα μπορούν να εμφανίζονται και οι κανονικοποιημένες ευκλείδειες αποστάσεις μεταξύ των σημείων του υπερ-πολυέδρου όπως και οι απόλυτες αποστάσεις τους ως προς τους άξονες.

## 4.3 Διαδραστική Παρουσίαση του Μοντέλου Προτιμήσεων

Βασική λειτουργία του Λογισμικού είναι η δυνατότητα της διασύνδεσης των σημείων του Συνόλου των Λύσεων με το μοντέλο προτιμήσεων. Σε μια φόρμα που θα περιλαμβάνει σε τρισδιάστατη μορφή το υπερ-πολύεδρο, ο χειριστής θα έχει τη δυνατότητα να επιλέγει σημείο του υπερ-πολύεδρου και αυτόματα να παρουσιάζεται το μοντέλο προτιμήσεων στο δεύτερο μισό της φόρμας. Από το μοντέλο προτιμήσεων θα εμφανίζεται: α) Τα βάρη των κριτηρίων, β) οι συναρτήσεις μερικών αξιών, γ) οι εναλλακτικές ενέργειες με την κατάταξή τους, την προ-διάταξη και τις ολικές αξίες.

Η λειτουργία αυτή δημιουργεί τις προϋποθέσεις μιας ακόμη μορφής ανάδρασης μεταξύ αποφασίζοντος και μοντέλου.




**Σχήμα 4.1: Βασική εικόνα RAVI**

# 5. Μέθοδοι και Τεχνικές παρουσίασης των 3D γραφημάτων

## 5.1 Γενικά

Ένα από τα βασικά προβλήματα της τεχνολογίας των Ηλεκτρονικών Υπολογιστών σήμερα είναι η οπτικοποίηση των ν-διάστατων χώρων στις δυο ή τις τρεις διαστάσεις. Η τεχνολογία μπορεί να υποστηρίξει την αναπαράσταση ενός σχήματος ή όγκου σε δυο διαστάσεις και με τεχνικές να δώσει την αίσθηση του τρισδιάστατου που στην πραγματικότητα αποτελεί μια ψευδαίσθηση.

Η τεχνολογία στον τομέα των Γραφικών έχει προχωρήσει με αποτέλεσμα τα τεχνικά προβλήματα (στο γραφικό περιβάλλον GUI) που αφορούσαν στα γραφικά να έχουν λυθεί, καθώς αφενός οι αποθηκευτικές μνήμες των Η/Υ έχουν αυξηθεί σε ταχύτητα και χωρητικότητα, και αφετέρου οι επεξεργαστές έχουν ταχύτητες που επιτρέπουν την επεξεργασία μεγάλου όγκου δεδομένων (όπως αυτές των γραφικών). Ταυτόχρονα σε γραφικό περιβάλλον, μεθοδολογίες και τεχνικές που βασίζονται στην Προβολική και Διανυσματική Γεωμετρία έχουν δώσει ενδιαφέρουσες λύσεις .

Η ανάλυση πολυδιάστατων δεδομένων αποτελεί ένα από τα βασικά θέματα με τα οποία ασχολείται ένας μεγάλος αριθμός ερευνητών. Η οπτικοποίηση της δομής των πολυδιάστατων δεδομένων αποτελούσε πάντα ένα πρόβλημα στοχεύοντας στην απλούστευση και καλύτερη κατανόηση της μορφής και της ιδιαιτερότητας των περιπτώσεων που μελετούνται, αντιμετωπίζοντας το πρόβλημα της πολυπλοκότητας που εγγενώς υπάρχει από την πολυδιάστατη φύση των δεδομένων. Στον Γραμμικό Προγραμματισμό η ανάγκη αυτή είναι ακόμη μεγαλύτερη, καθώς σε πολλές περιπτώσεις απαιτείται η μελέτη του υπερ-πολύεδρου των εφικτών λύσεων ή των λύσεων σε προβλήματα με πολλούς αγνώστους (περισσότερο από τρεις). Η πολυπλοκότητα του προβλήματος αυτού αυξάνει όσο μεγαλύτερος είναι ο αριθμός των διαστάσεων του προβλήματος. Στόχος της εργασίας αυτής είναι να διερευνήσει με ποια μεθοδολογικά και τεχνολογικά εργαλεία είναι δυνατόν να διευκολυνθεί η μελέτη των υπερ-πολυέδρων, μέσα από την ανάλυση των λύσεων που προκύπτουν από την επίλυση γραμμικών προβλημάτων με πολλούς αγνώστους.

Βασικός άξονας της μελέτης αυτής είναι να αναδειχθεί ο καλύτερα τεχνικά και μεθοδολογικά τρόπος για την οπτικοποίηση ν-διάστατων δεδομένων σε τρισδιάστατη μορφή στο δισδιάστατο χώρο.

Στην παρούσα Τεχνική Έκθεση παρουσιάζονται οι εναλλακτικές μεθοδολογικές προσεγγίσεις και τεχνικές για την γραφική αποτύπωση ν-διάστατων δεδομένων μέσω Η/Υ. Για την καλύτερη κατανόηση σε πρώτη φάση αναλύονται οι βασικές μεθοδολογίες για την δημιουργία γραφικών τριών διαστάσεων στον δισδιάστατο χώρο, που αποτελούν βασικά στοιχεία της ανάπτυξης γραφικών για μεγάλου αριθμού διαστάσεων δεδομένων. Στην συνέχεια η ανάλυση προχωρά στις εφαρμογές και στις μελέτες περίπτωσης που αφορούν το αντικείμενο αυτού του ερευνητικού έργου.

## 5.2 Τρισδιάστατες Γραφικές Απεικονίσεις

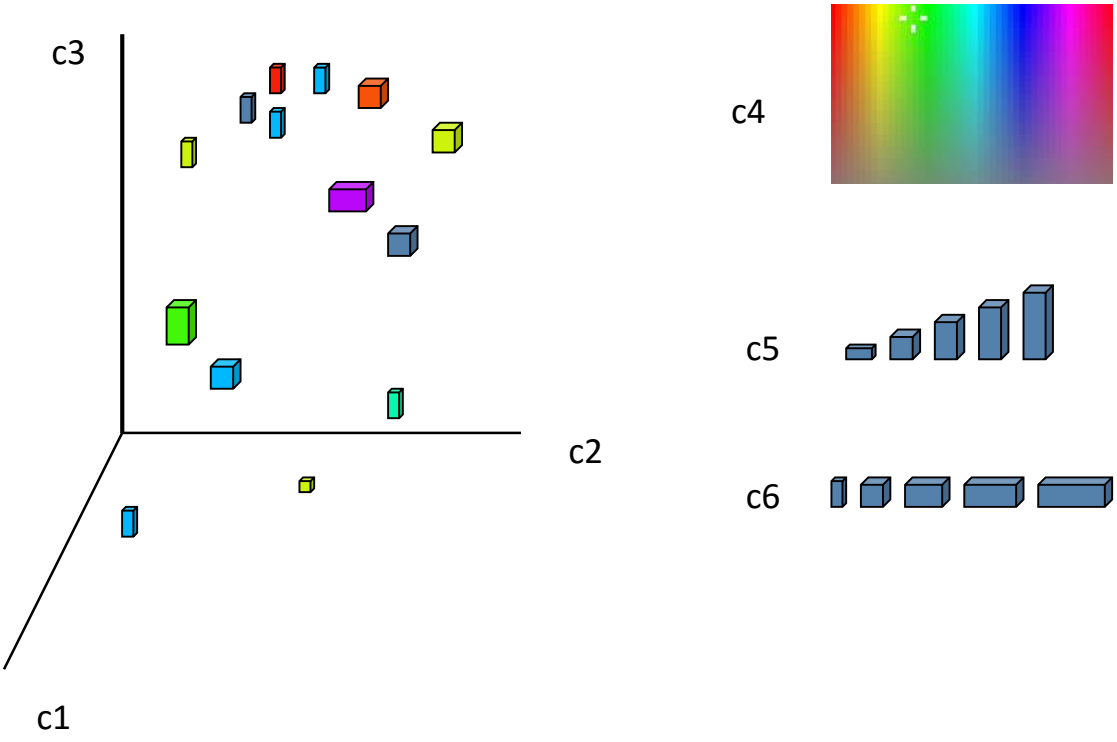
Η αποτύπωση Ν-διάστατων δεδομένων στις δυο διαστάσεις έτσι ώστε να μπορεί το ανθρώπινο μυαλό να τις επεξεργαστεί, συνιστά μια από τους τομείς έρευνας αυτών που ασχολούνται με τις γραφικές απεικονίσεις.

Για να απεικονίσουμε ν-διάστατα δεδομένα σε τρισδιάστατη μορφή υπάρχουν οι παρακάτω τέσσερις εναλλακτικοί τρόποι.

**Α) Χρησιμοποιώντας ανάγλυφα**

Καθορίζουμε δυο μεταβλητές χ, ψ ως χωρικές μεταβλητές και όλες τις άλλες τις αποτυπώνουμε χρησιμοποιώντας πρότυπα όπως είναι το χρώμα, το σχήμα, το μέγεθος κ.ά. για να τις προβάλουμε στον δισδιάστατο χώρο.

Στο γράφημα που ακολουθεί δίνεται παραδειγματικά η παρουσίαση ν-διάστατων δεδομένων στο επίπεδο με ανάγλυφα.

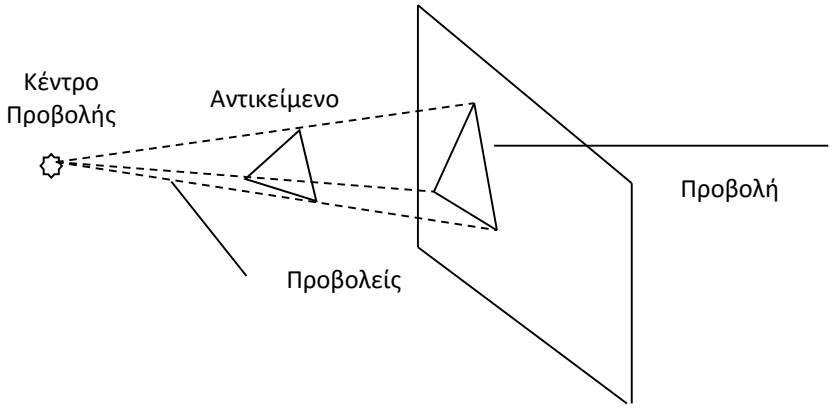



**Σχήμα 5.1: Παρουσίαση ν-διάστατων δεδομένων στο επίπεδο με ανάγλυφα**

**Β) Με την απλή προοπτική προβολή του αντικειμένου**

Θεωρούμε ότι υπάρχει ένα σημείο (το σημείο Θέασης) και απεικονίζουμε το αντικείμενο σαν να προβάλλεται στο επίπεδο, όπως ακριβώς γίνεται και η προβολή μια διαφάνειας. Η αίσθηση της τρισδιάστατης μορφής γίνεται με το ίδιο τρόπο που γίνεται και με τα απλά μηχανικά διαφανειοσκόπια

Στο σχήμα που ακολουθεί παρουσιάζεται γραφικά ο τρόπος με τον οποίο προβάλουμε ένα τρισδιάστατο αντικείμενο στο επίπεδο με προοπτική προβολή .

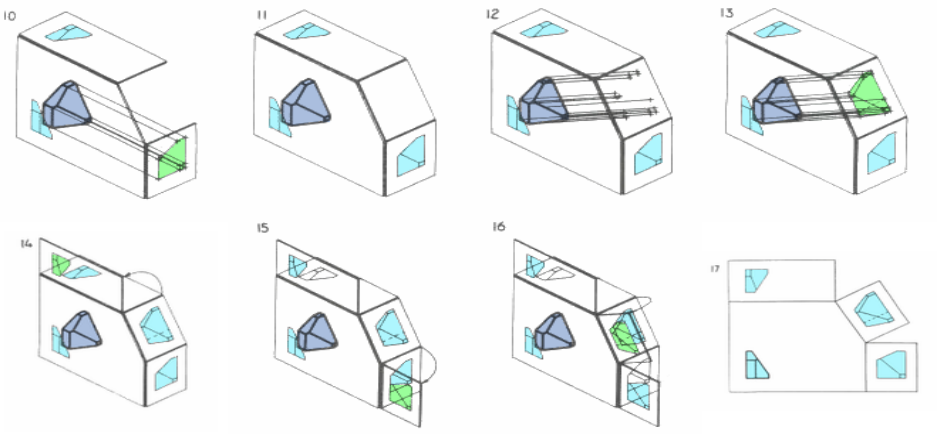



**Σχήμα 5.2: Προβολή τρισδιάστατου αντικειμένου στο επίπεδο**

Ένα από τα χαρακτηριστικά της Προοπτικής Προβολής είναι ότι το στερεό δεν εμφανίζεται στο μέγεθός του. Επίσης δύο αντικείμενα ίδιου μεγέθους έχουν διαφορετική προβολή ανάλογα με τη θέση τους και το σημείο προβολής.

**Γ) Αξιοποιώντας την ορθογραφική προβολή με την οποία προβάλουμε δυο ή τρεις χωρικές μεταβλητές αγνοώντας τις υπόλοιπες**

Η ορθογραφική προβολή προσομοιάζει την προβολή ενός αντικειμένου σε επίπεδο, όπως αυτό γίνεται σε ένα διαφανιοσκόπιο ή στον κινηματογράφο. Οι ακτίνες της προβολής είναι παράλληλες προς το τρισδιάστατο αντικείμενο που προβάλλεται. Στην ορθογραφική προβολή περιλαμβάνονται δυο τεχνικές. Η πρώτη multiview orthographic projection δημιουργούνται πολλαπλές γραφικές προβολές του τρισδιάστατου αντικειμένου παράλληλες προς τους άξονες του τρισδιάστατου χώρου. Οι προβολές τοποθετούνται σε σχετικές θέσεις σύμφωνα με δυο σχήματα: Προβολή πρώτης γωνίας και προβολή τρίτης γωνίας. Φαίνεται σαν να προβάλλεται σε ένα κύβο ή παραλληλεπίπεδο έξι επιφανειών. Τρεις προβολές είναι ικανές να δώσουν εποπτεία του αντικειμένου (πρόσθια προβολή, προβολή κορυφής, πλευρική προβολή).



**Σχήμα 5.3: Ορθογωνική προβολή τρισδιάστατου αντικειμένου**

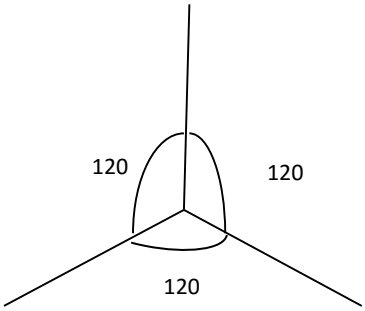
Παράδειγμα: Ένα στερεό προβάλλεται στις επιφάνειες παράλληλα και υπό γωνία ως προς τους άξονες συντεταγμένων. Στο τελικό διάγραμμα έχουμε τέσσερις προβολές του στερεού (παράλληλες προς τους άξονες x,y,z και υπό γωνία).

**Δ) Αξιοποιώντας την Αξονομετρική Προβολή**

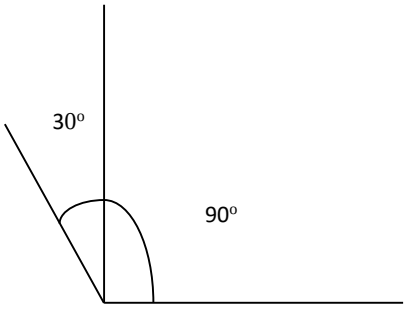
Η Αξονομετρική Προβολή είναι παράλληλη προβολή κατά την οποία το αντικείμενο στρέφεται ως προς έναν ή περισσότερους άξονες του επιπέδου της προβολής. Η εικόνα του αντικειμένου προβάλλεται έτσι ώστε να δίνει η αίσθηση της θέασης υπό γωνία και να είναι ορατές περισσότερες από μια πλευρές. Είναι χαρακτηριστικό ότι διατηρούνται οι μετρητικές σχέσεις των τριών διαστάσεων του Στερεού, η κατεύθυνση των πλευρών των στερεών (παράλληλες ευθείες αποτυπώνονται ως παράλληλες). Η Αξονομετρική Προβολή είναι αποτελεσματική στις περιπτώσεις που θέλουμε να προβάλουμε με διάφορους τρόπους ένα στερεό (από διαφορετικές γωνίες και διευθύνσεις). Στην Αξονομετρική Προβολή χρησιμοποιούμε το τρισδιάστατο Σύστημα Συντεταγμένων OXYZ στο οποίο έχουμε τη δυνατότητα να το μετασχηματίσουμε με τρόπο που να αναδεικνύει την προοπτική του στερεού που προβάλλεται.

Υπάρχουν τρεις τύποι της αξονομετρικής προβολής:

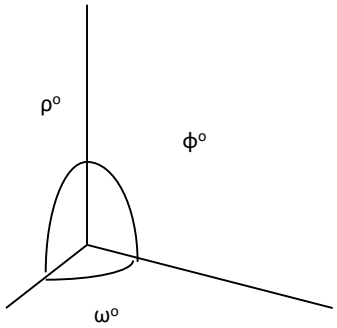
i. Ισομετρική, χρησιμοποιείται στο μηχανολογικό σχέδιο και οι τρεις άξονες εμφανίζονται ισοδύναμα συμπυκνωμένοι. Η κλίμακα συμπύκνωσης είναι προκαθορισμένη. Στο σχήμα που ακολουθεί οι άξονες σχηματίζουν γωνία 120 μοιρών μεταξύ τους. Δε επέρχονται μεταβολές στις αναλογίες των διαστάσεων των στερεών στους τρείς άξονες και τα μοναδιαία διανύσματα έχουν ίσο μήκος.



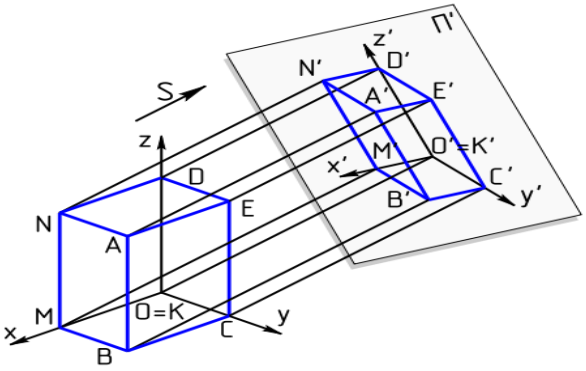

ii. Διμετρική είναι ο εκείνη η προβολή όπου οι δυο από τους τρεις άξονες είναι όμοια συμπυκνωμένοι. Οι δύο άξονες είναι κάθετοι μεταξύ τους και οι τρίτος σχηματίζει γωνία (30ο, 45ο, 60ο) με έναν από τους άξονες. Τα μοναδιαία διανύσματα των δυο πρώτων αξόνων έχουν ίσο μήκος (1) ενώ του τρίτου εξαρτάται από τη σχετική γωνία.



iii. Τριμετρική αξονική προβολή είναι εκείνη κατά την οποία οι τρείς άξονες είναι ανόμοια συμπυκνωμένοι. Οι κλίμακες συμπύκνωσης και οι γωνίες μεταξύ των αξόνων καθορίζονται από την γωνία θέασης του αντικειμένου.




Στην εικόνα που ακολουθεί δίνεται παράδειγμα τριμετρικής αξονικής προβολής στερεού.



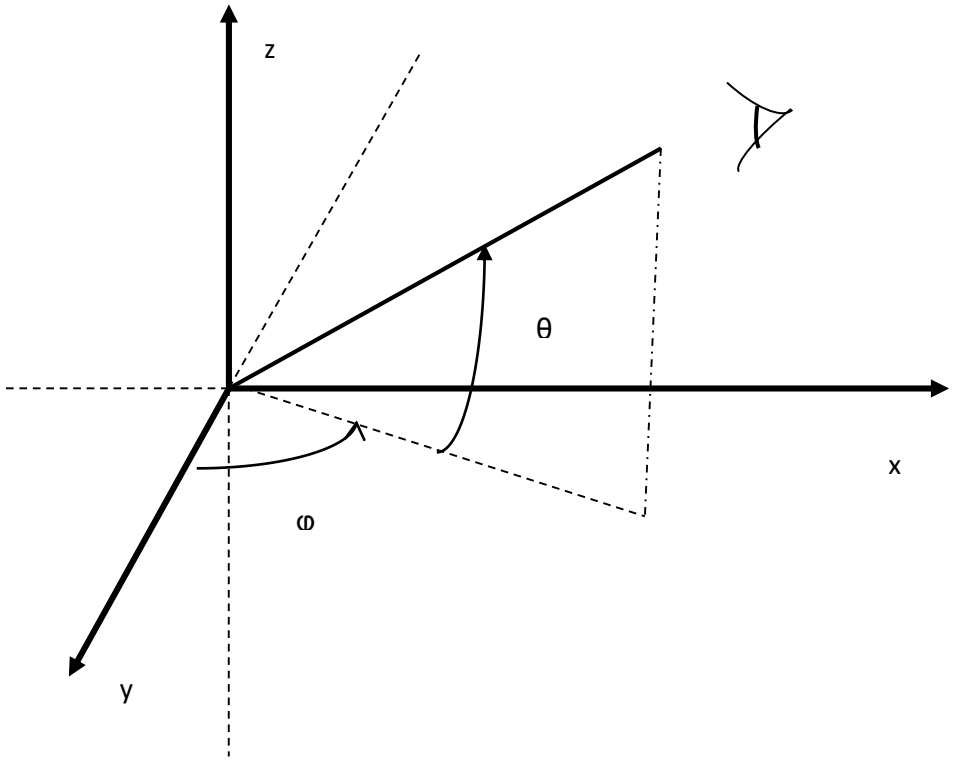

**Σχήμα 5.4: Τριμετρική αξονική προβολή στερεού**

## 5.3 Προβολές με την αξιοποίηση Πολικών Συντεταγμένων

Ένας ευέλικτος τρόπος εμφάνισης των τρισδιάστατων γραφικών είναι με τη χρήση πολικών συντεταγμένων. Πριν προχωρήσουμε στην διεξοδική ανάλυση του τρόπου με τον οποίο μπορούμε να εμφανίσουμε τρισδιάστατα γραφικά θα πρέπει να δώσουμε τους παρακάτω ορισμούς.

Το **αζιμούθιο** είναι η θετική γωνία στο επίπεδο x-y (στροφή όμοια με τους δείκτες του ωρολογίου) που σχηματίζεται από τη προβολή του σημείου στον τρισδιάστατο χώρο και ενός εκ των αξόνων Χ-Υ έτσι ώστε να διατηρείται η φορά όμοια με αυτή των δεικτών του ωρολογίου.

**Ανύψωση** είναι η γωνία που σχηματίζεται (θετική όταν είναι πάνω από το επίπεδο Χ-Υ και αρνητική κάτω) από το την ευθεία που συνδέει το σημείο μα την αρχή των αξόνων και την προβολή της στο επίπεδο Χ-Υ.




**Σχήμα 5.5: Τρισδιάστατα γραφικά με τη χρήση πολικών συντεταγμένων**

Η υλοποίηση Γραφικών γίνεται ευκολότερα διότι ο πίνακας μετασχηματισμού αποκτά την παρακάτω μορφή (με περιστροφή του συστήματος συντεταγμένων κατά φ στον άξονα z και κατά π/2-θ στο άξονα χ)

$$R_z(-\varphi)R_x(\theta-\pi/2)=\begin{pmatrix}\cos\varphi & -\sin\varphi & 0 & 0\\ \sin\varphi & \cos\varphi & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & \sin\theta & -\cos\theta & 0\\ 0 & \cos\theta & \sin\theta & 0\\ 0 & 0 & 0 & 1\end{pmatrix}$$

$$=\begin{pmatrix}\cos\varphi & -\sin\varphi\sin\theta & \sin\varphi\cos\theta & 0\\ \sin\varphi & \cos\varphi\sin\theta & -\cos\varphi\cos\theta & 0\\ 0 & \cos\theta & \sin\theta & 0\\ 0 & 0 & 0 & 1\end{pmatrix}$$

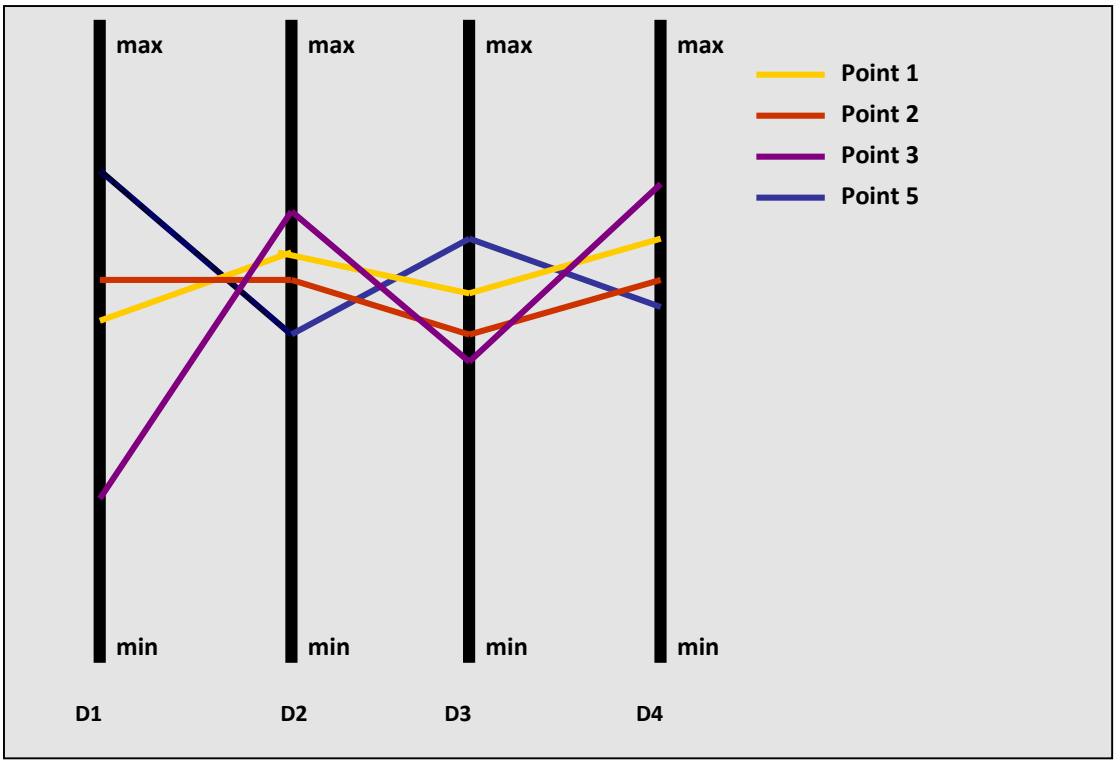
Για τη δημιουργία ενός Γραφικού ακολουθούμε την παρακάτω διαδικασία:

- Ορίζουμε το τρισδιάστατο Σύστημα Συντεταγμένων στο Επίπεδο το οποίο έχει αρχή το κέντρο του καμβά στον οποίο εμφανίζεται το γραφικό.
- Ορίζουμε τις δύο γωνίες αζιμούθιο (από -180 έως 180) και ανύψωση (από -90 έως 90).
- Υπολογίζουμε τον Πίνακα Μετασχηματισμού ανάλογα με τη πολικές συντεταγμένες σύμφωνα με τον παραπάνω τύπο

- Εισάγουμε τις συντεταγμένες των κορυφών του στερεού
- Εισάγουμε τις κορυφές που σχηματίζουν τα πολύγωνα του Στερεού
- Για κάθε Πολύγωνο:
    - Υπολογίσουμε τις συντεταγμένες των σημείων με βάση τον πίνακα μετασχηματισμού (πολλαπλασιασμός)
    - Σχηματίζουμε το πολύγωνο
    - Βρίσκουμε τη θέση του πολυγώνου (αν είναι ορατό σε σχέση με τη θέση του παρατηρητή). Αν είναι ορατό αποδίδουμε ένα χρώμα στο πολύγωνο
    - Εμφανίζουμε το πολύγωνο

## 5.4 Σύστημα Παράλληλων Συντεταγμένων

Χρησιμοποιούμε σύστημα παράλληλων συντεταγμένων. Κάθε παράλληλος άξονας εκφράζει μια μεταβλητή. Μια περίπτωση (πολυδιάστατο δεδομένο) αποτυπώνεται με μια τεθλασμένη γραμμή που ενώνει τα σημεία των παράλληλων αξόνων.

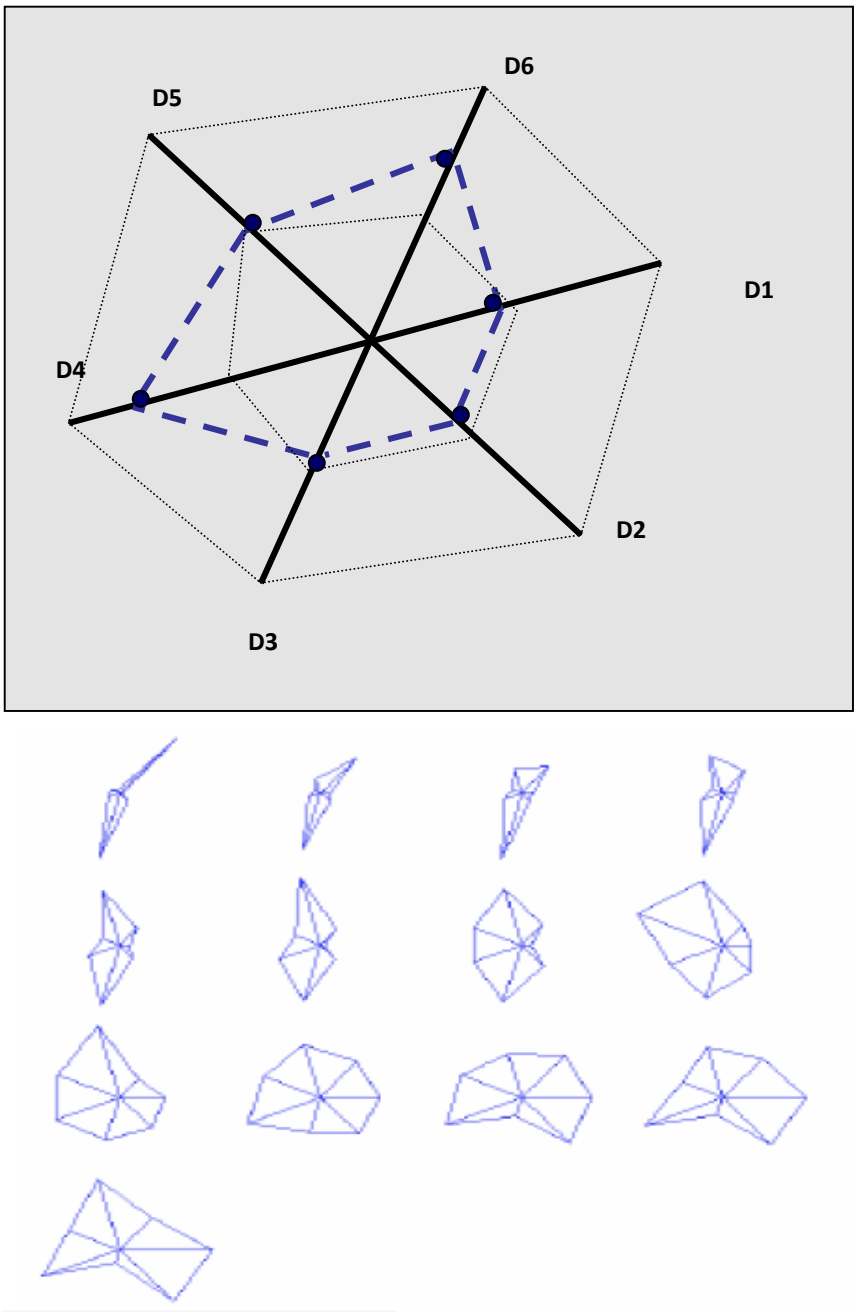



**Σχήμα 5.6: Αποτύπωση ν-διάστατων δεδομένων σε σύστημα παράλληλων συντεταγμένων**

Αποτελεί μια εύκολη λύση στην αποτύπωση ν-διάστατων δεδομένων. Υπάρχουν δυσκολίες στην περίπτωση μεγάλου όγκου δεδομένων καθώς δεν είναι εύκολο να διακριθούν οι περιπτώσεις.

## 5.5 Διαγράμματα Αστέρα

Στα διαγράμματα Αστέρα κατασκευάζουμε επίπεδο σύστημα συντεταγμένων με κοινή αρχή των αξόνων και οι γειτονικοί άξονες σχηματίζουν ίση γωνία μεταξύ τους. Η ευθεία που ενώνει σημεία των αξόνων που αντιστοιχούν στις τιμές ενός πολυδιάστατου δεδομένου εκφράζει και το αντίστοιχο σημείο του ν-δάστατου χώρους. Μπορούμε να αποτυπώσουμε πολλά σημεία κατασκευάζοντας ένα διάγραμμα αστέρα για κάθε σημείο.

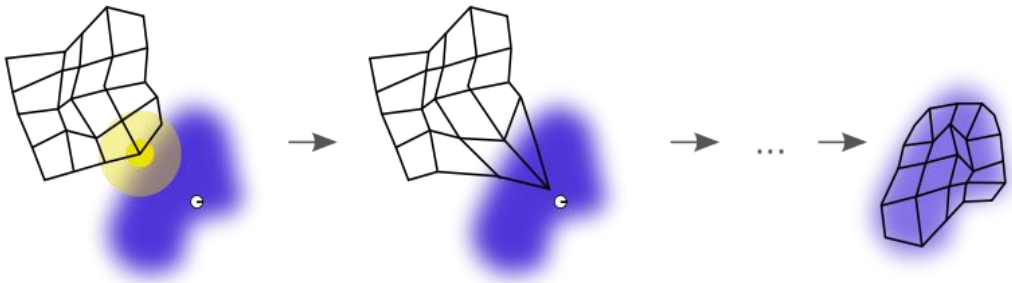



**Σχήμα 5.7: Παραδείγματα διαγραμμάτων αστέρα**

## 5.6 Αυτο-οργανούμενοι Χάρτες

Οι Αυτο-οργανούμενοι χάρτες βασίζονται στα τεχνητά νευρωνικά δίκτυα και μπορούν να παράγουν γραφήματα με 2 ή 3 διαστάσεων για δεδομένα μεγαλύτερης διάστασης.

Όπως σε όλα τα νευρωνικά δίκτυα στους αυτο-οργανούμενους χάρτες περιλαμβάνεται μια φάση εκπαίδευσης όπου χρησιμοποιώντας δείγματα δεδομένων και μια συνάρτηση γειτνίασης ανάμεσα στα σημεία του νευρωνικού δικτύου και των δεδομένων γίνεται η προσαρμογή.



**Σχήμα 5.7: Κατασκευή αυτό-οργανούμενων χαρτών**

# 6. Τρισδιάστατες Απεικονίσεις στο Σύστημα RAVI

## 6.1 Γενικά

Το Λογισμικό RAVI θα παρέχει τη δυνατότητα δημιουργίας γραφικών παρουσιάσεων σε πολλαπλές μορφές, οι οποίες αναπτύχθηκαν στο πλαίσιο της διερεύνησης του καλύτερου τρόπου για την οπτικοποίηση των αποτελεσμάτων της επίλυσης του πολυκριτηριακού γραμμικού προβλήματος.

Η βασική γραφική επιλογή βασίζεται στην Αξονομετρική Προβολή στην οποία χρησιμοποιούνται πολικές συντεταγμένες

## 6.2 Τεχνική Λύση

Η ανάπτυξη του Συστήματος πραγματοποιήθηκε σε περιβάλλον .NET σε γλώσσα προγραμματισμού C# και οι διεπαφές με το WINDOWS PRESENTATION FRAMEWORK (WPF).

Οι δομές που υποστηρίζονται για την Ανάπτυξη Εφαρμογών με γραφικά 3D στο περιβάλλον .NET είναι:

### 6.2.1 Σημείο - Διάνυσμα

*Vector3d (x,y,z) - Μονοδιάστατος Πίνακας με τρεις παραμέτρους*

Διαθέτουν τις παρακάτω ιδιότητες:

- Length – Το μέγεθος του διανύσματος.
- LengthSquared – Το τετράγωνο του μεγέθους του διανύσματος
- X – Η τιμή στο άξονα χ.
- Y – Η τιμή στον άξονα ψ
- Z – Η τιμή στον άξονα Z

Και τις παρακάτω μεθόδους:

- Add – Προσθέτει ένα διάνυσμα Vector3D σε ένα σημείο Point3D ή σε ένα άλλο διάνυσμα Vector3D.
- Subtract – Αφαιρεί ένα διάνυσμα Vector3D από ένα σημείο Point3D ή από ένα άλλο διάνυσμα Vector3D
- Multiply – Πολλαπλασιάζει ένα διάνυσμα Vector3D structure με ένα άλλο ή Matrix3D και επιστρέφει διάνυσμα Vector3D.
- Divide – Διαιρεί ένα διάνυσμα Vector3D structure με βάση μια κλίμακα και επιστρέφει σαν αποτέλεσμα ένα διάνυσμαVector3D.

- CrossProduct – Καρτεσιανό Γινόμενο δυο διανυσμάτων Vector3D.
- AngleBetween – Υπολογίζει τη απαιτούμενη γωνία περιστροφής του πρώτου διανύσματος ως προς το δεύτερο διάνυσμα
- Normalize – Κανονικοποίηση ενός Διανύσματος Vector3D.

*Point3d (X,Y,Z) – Μονοδιάστατος Πίνακας με τρεις παραμέτρους*

Το τρισδιάστατο σημείο Point3D έχει τις παρακάτω ιδιότητες

- X – Η συντεταγμένη χ.
- Y – Η συντεταγμένη ψ.
- Z – Η συντεταγμένη Z.

Και τις μεθόδους:

- Add – Προσθέτει ένα σημείο Point3D σε ένα διάνυσμα Vector3D κα επιστρέφει σημείο Point3D.
- Subtract – Αφαιρεί ένα σημείο Point3D ή ένα διάνυσμα Vector3D από ένα σημείο Point3D.
- Multiply – Μετασχηματίζει το σημείο Point3D structure από ένα προκαθορισμένο Matrix3D.
- Offset – Αλλαγές στις τιμές X, Y, και Z του σημείου Point3D structure με μια καθορισμένη τιμή

*Δομή Point4d*

Τρισδιάστατο σημείο στο ομογενές σύστημα συντεταγμένων και χρησιμοποιείται για τους μετασχηματισμούς.

Η δομή Point4D έχει τις παρακάτω τέσσερις δημόσιες ιδιότητες:

- X – Η συντεταγμένη χ.
- Y – Η συντεταγμένη ψ.
- Z – Η συντεταγμένη Z
- W – Το στοιχείο W του Point4D.

Και τις μεθόδους:

- Add – Πρόσθεση Point4D structure to another Point4D structure.
- Subtract – Αφαίρεση.
- Multiply – Μετασχηματισμός
- Offset – Αλλαγές στις τιμές X, Y, και Z του σημείου Point4D με μια καθορισμένη τιμή Point4D

*Η δομή Matrix3D*

Πίνακας 4 Χ 4 στο τρισδιάστατο ομογενοποιημένο σύστημα συντεταγμένων και έχει την παρακάτω μορφή:

$M_{11}$, $M_{12}$, $M_{13}$, $M_{14}$

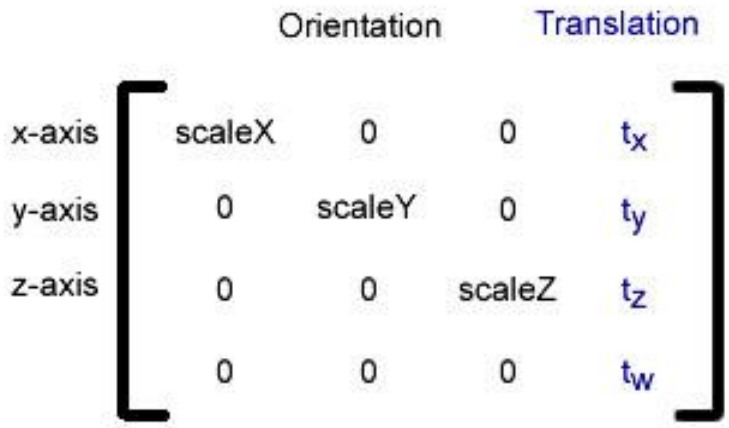
$M_{21}$, $M_{22}$, $M_{23}$, $M_{24}$

$M_{31}$, $M_{32}$, $M_{33}$, $M_{34}$

ΒήμαΧ, ΒήμαΥ, ΒήμαΖ, Μ44

Η δομή Matrix3D αποτελεί τον πίνακα μετασχηματισμού που καθορίζει τη θέση και τον προσανατολισμό ενός τρισδιάστατου αντικειμένου. Μέσω του πίνακα μπορούν να εκτελεστούν συναρτήσεις μετασχηματισμού συμπεριλαμβανομένης της μετάφρασης (επανατοποθέτηση των αξόνων x,y,z), περιστροφή και μεταβολή κλίμακας. Επίσης μέσω του πίνακα μπορεί να πραγματοποιηθεί και η προοπτική προβολή, η οποία απεικονίζει σημεία του τρισδιάστατου χώρους σε δυο διαστάσεις. Μέσω ενός πίνακα μπορούν να συνδυαστούν πολλαπλοί συνδυασμοί μετασχηματισμών.

Για κάθε αλλαγή της z ή για κάθε περιστροφή η μεταβολή των αξόνων δημιουργείται αυτόματα και ο αντίστοιχος πίνακας. Προσπέλαση στο πίνακα γίνεται με την ιδιότητα transform.matrix3d. Οι πρώτες τρεις γραμμές περιλαμβάνουν δεδομένα για τους τρεις άξονες x,y,z. Η τελευταία στήλη περιλαμβάνει δεδομένα για την μετάφραση. Ο προσανατολισμός και η κλίμακα περιλαμβάνονται στις τρεις πρώτες στήλες. Το διάνυσμα της κλίμακας είναι τα τρία πρώτα στοιχεία της διαγωνίου.




Η δομή περιλαμβάνει μεθόδους όπως appendTranslation(), appendRotation() και interpolateTo() για την επεξεργασία του πίνακα.

Οι μέθοδοι βοηθούν στην υλοποίηση της επεξεργασίας των πινάκων μετασχηματισμού:

- Determinant – Υπολογίζει τον προσδιοριστή του Matrix3D.
- HasInverse – Βρίσκει αν ο πίνακας έχει αντίστροφο.
- Identity – Μεταβάλλει τον Πίνακα Matrix3D σε identity Matrix3D.

- IsAffine – Προσδιορίζεται αν ο πίνακας Matrix3D είναι affine.
- IsIdentity – Προσδιορίζεται να ο πίνακας Matrix3D είναι identity.

### 6.2.2 Διαδικασία δημιουργίας 3d προβολών

Καθορίζουμε τη θέση της κάμερας P(x,y,z), το διάνυσμα N που δίνει την κατεύθυνση της κάμερας και το διάνυσμα U που καθορίζει την κατεύθυνση προς τα πάνω της κάμερας. Με αυτά τα στοιχεία κατασκευάζουμε τον πίνακα μετασχηματισμού:

Κατασκευάζουμε τρία διανύσματα του 3d χωρου: XScale, YScale, και ZScale.

- Κανονικοποιούμε τα διανύσματα N και U έτσι ώστε να είναι μοναδιαία διανύσματα.
- Θέτουμε την ZScale = N;.
- Υπολογίζουμε την YScale σύμωνα με τον τύπο

$$YScale = \frac{\bar{\bar{U}} - (\overline{UN})\bar{N}}{\sqrt{1 - (\overline{UN})^2}}$$

- Υπολογίζουμε την XScale σύμφωνα με τον τύπο

$$XScale = \frac{\bar{N} \times \bar{U}}{\sqrt{1 - (\overline{UN})^2}}$$

- Κατασκευάζουμε τον πίνακα M στον τρισδιάστατο ομογενοποιημένο σύστημα συντεταγμένων

| | | | |
|---|---|---|---|
| $XScale.X$ | $XScale.X$ | $XScale.X$ | |
| $XScale.y$ | $XScale.y$ | $XScale.y$ | |
| $XScale.z$ | $XScale.z$ | $XScale.z$ | |
| 0 | 0 | 0 | 0 |

Μεταφράζουμε τη θέση της κάμερας στο σύστημα συντεταγμένων και την ανάκλαση της σε σχέση με τον z. Ο πίνακας μετασχηματισμού δίνεται: V=T(-x, -y, -z) x M x S(1,1,-1), όπου T είναι ο πίνακας μετάφρασης και S είναι ο πίνακας της κλίμακας με sz = -1, που αντιστοιχεί στην ανάκλαση στον άξονα Z.

*Μετασχηματισμοί για την ορθογραφική Προβολή*

Η ορθογραφική προβολή στοχεύει στην προβολή του αντικειμένου μέσα σε έναν κύβο που έχει μήκος πλευράς 1. Καθώς δεν υπάρχει η παράμετρος της προοπτικής, οι μετασχηματισμοί είναι γραμμικοί.

*Ομογενοποιημένο Σύστημα Συντεταγμένων*

Σύστημα Συντεταγμένων που χρησιμοποιείται στην Προβολική Γεωμετρία. Παρέχει τη δυνατότητα στο σημείο, ακόμα και αν αυτό είναι στο άπειρο, να παρουσιάζεται με πεπερασμένο αριθμό συντεταγμένων.

Το επίπεδο προβολής μπορεί να θεωρηθεί σαν το Ευκλείδειο Επίπεδο με μερικά επιπλέον σημεία, τα σημεία του απείρου. Υπάρχει ένα σημείο στο άπειρο για κάθε διάσταση το οποίο ορίζεται ως το όριο σε σχέση με την διάσταση. Συνεπώς παράλληλες γραμμές σύμφωνα με την Ευκλείδεια Γεωμετρία τέμνονται στο άπειρο στην προβολική γεωμετρία. Ένα σημείο (X,Y) στο Ευκλείδειο επίπεδο ορίζεται με δυο λόγους (X/Z. Y/Z) και έτσι ένα σημείο αντιστοιχεί στην τριπλέτα (X,Y,Z) όπου Z<>0. Αυτή η τριπλέτα αποτελεί τις ομογενείς συντεταγμένες του σημείου (X,Y). Είναι αξιοσημείωτα τα ακόλουθα:

- Αν πολλαπλασιάσουμε και τις τρεις ομογενείς συντεταγμένες με τον ίδιο αριθμό, τότε το σημείο δεν αλλάζει. Ένα σημείο μπορεί να αναπαρασταθεί από πολλές ομογενείς συντεταγμένες.
- Η εξίσωση της ευθείας γίνεται: $l(x - a) + m(y - b) = 0$ όπου l,m <>0. Σε παραμετρική μορφή μπορεί να γραφεί $x = a + mt$, $y = b - lt$. Έστω Z=1/t τότε έχουμε για τις συντεταγμένες ενός σημείου της ευθείας $(a + m/Z, b - l/Z)=((aZ + m)/Z, (bZ - l)/Z)$. Αν το t τείνει στο άπειρο τότε το ζ τείνει στο 0 και το σημείο έχει ομογενείς συντεταγμένες $(m, -l, 0)$. Οι συντεταγμένες αυτές αποτελούν τις ομογενείς συντεταγμένες του σημείου στο άπειρο.

Εν κατακλείδι:

- Κάθε σημείο στο επίπεδο προβολής παρουσιάζεται σαν τριπλέτα $(X, Y, Z)$, και καλούνται ομογενείς συντεταγμένες ή προβολικές συντεταγμένες του σημείου **όπου** $X, Y$ και $Z$ δεν είναι όλα 0.
- Κάθε σημείο που εμφανίζεται με μια τριπλέτα ομογενών συντεταγμένων παραμένει αμετάβλητο αν πολλαπλασιάσουμε τις συντεταγμένες με τον ίδιο αριθμό.
- Όταν $Z<> 0$ το σημείο είναι το σημείο $(X/Z, Y/Z)$ στο Ευκλείδειο Επίπεδο.
- Όταν $Z = 0$ το σημείο βρίσκεται στο άπειρο.

*Οι Ομογενείς Συντεταγμένες στα Γραφικά Η/Υ*

Το Σύστημα Ομογενών Συντεταγμένων είναι θεμελιώδες στα Γραφικά Η/Υ διότι παρέχει τη δυνατότητα συνήθεις λειτουργίες όπως η μετάφραση, η περιστροφή, η μετατροπή της κλίμακας και οι προοπτικές προβολές να υλοποιηθούν

# Βιβλιογραφία

Greco, S., Mousseau, V., Slowinski, R., Ordinal regression revisited: multiple criteria ranking using a set of additive value functions, European Journal of Operational Research (2007), doi: 10.1016/j.ejor.2007.08.013

Grinstein, G. G. and M. O. Ward, "Introduction to Data Visualization", Information Visualization in Data Mining and Knowledge Discovery, Morgan Kaufmann Publishers, pp.21-45, 2001.

Hwang, C-l. and M-J. Lin (1988), Group Decision Making under Multiple Criteria in Lectures Notes in Economics and Mathematical Systems, Springer-Verlag,, Berlin, Heidelberg.

Jacquet-Lagreze, E. and M.F. Shakun (1984), Decision support system for semistructured buying decisions, European Journal of Operational Research, 16(1), 48-56.

Jacquet-Lagrèze, E. and Y. Siskos (1982), Assessing a set of additive utility functions for multicriteria decision making, European Journal of Operational Research, 10(2), 151-164.

Keeney, R.L. and H. Raiffa (1976), Decision with multiple objectives: Preferences and value tradeoffs, J. Wiley, NY.

P. E. Hoffman and G. G. Grinstein, "A Survey of Visualizations for High-Dimensional Data Mining", Information Visualization in Data Mining and Knowledge Discovery, Morgan Kaufmann Publishers, pp.47-82, 2001.

R. Spence, Information Visualization, Addison Wesley, ACM Press, 2000.

S.K. Card, J.D. Mackinlay, and B. Shneiderman, Information Visualization: Using Vision to Think, Morgan Kaufmann Publishers, 1999.

Siskos, Y. (1980), Comment modéliser les préférences au moyen de fonctions d'utilité additives, RAIRO Recherche Opérationnelle, 14, 53-82.

Siskos, Y. and D. Yannacopoulos, (1985), UTASTAR, an ordinal regression method for building additive value functions, Investigacao Operacional, 5(1), 39-53.

Siskos, Y., A. Spyridakos and D. Yannacopoulos, (1993) MINORA: A multicriteria decision aiding system for discrete alternatives, Journal of Information Science and Technology, in: Y. Siskos and C. Zopounidis (eds), Special Issue on Multicriteria Decision Support Systems, 2(2), 136-149.

Siskos, Y., A. Spyridakos and D. Yannacopoulos, (1999), Using artificial intelligent techniques into preference disaggregation analysis: The MIIDAS system, European Journal of Operational Research, 113, 281-299.

Siskos,Y., 1984. Le Traitement des Solutions Quasi Optimales en Programmation Linéaire Continue: Une Synthèse. RAIRO, Recherche Opérationnelle , 18 (4 ), 381-401

Spyridakos, A. (2010), Selecting reference set in disaggregation-aggregation multicriteria decision aid approach utilising clustering techniques with dissimilarity threshols, in A Respicio

et al. (Eds) Building the socio-technical gap in decision support systems, IOS Press, pp. 217-248.

Tufte, E.R., The Visual Display of Quantitative Information, Graphics Press, 1983.

Zrehen, S. (1993), "Analyzing Kohonen Maps with Geometry", Int. Conf. on Artificial Neural Networks Proceedings, Springer-Verlag, London 609 - 613.

# Πηγαίος Κώδικας Βιβλιοθήκης RAVI

**ΒΙΒΛΙΟΘΗΚΗ ΤΑΞΕΩΝ (CLASS)**

Η Τάξη ChartStyle χρησιμοποιείται για τη δημιουργία αντικειμένων τρισδιάστατων γραφικών παρουσιάσεων πολυέδρων.

```
using System;
using System.Collections.Generic;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Media;
using System.Windows.Media.Media3D;
using System.Windows.Shapes;

namespace Chart3DNoWPFEngine
{
   public class ChartStyle
   {
      private Canvas chartCanvas;
      private double xmin = -5;
      private double xmax = 5;
      private double ymin = -3;
      private double ymax = 3;
      private double zmin = -6;
      private double zmax = 6;
      private double xtick = 1;
      private double ytick = 1;
      private double ztick = 3;
      private FontFamily tickFont = new FontFamily("Arial Narrow");
      private double tickFontSize = (double)new FontSizeConverter().ConvertFrom("8pt");
      private Brush tickColor = Brushes.Black;
      private string title = "My 3D Chart";
      private FontFamily titleFont = new FontFamily("Arial Narrow");
      private double titleFontSize = (double)new FontSizeConverter().ConvertFrom("14pt");
      private Brush titleColor = Brushes.Black;
      private string xLabel = "X Axis";
      private string yLabel = "Y Axis";
      private string zLabel = "Z Axis";
      private FontFamily labelFont = new FontFamily("Arial Narrow");
      private double labelFontSize = (double)new FontSizeConverter().ConvertFrom("10pt");
      private Brush labelColor = Brushes.Black;
      private double elevation = 30;
      private double azimuth = -37.5;
      private bool isXGrid = true;
      private bool isYGrid = true;
      private bool isZGrid = true;
      private bool isColorBar = false;
      private Line gridline = new Line();
      private Brush gridlineColor = Brushes.LightGray;
      private double gridlineThickness = 1;
      private GridlinePatternEnum gridlinePattern = GridlinePatternEnum.Dash;
      private Line axisLine = new Line();
      private Brush axisColor = Brushes.Black;
      private AxisPatternEnum axisPattern = AxisPatternEnum.Solid;
```

```
private double axisThickness = 1;

public Canvas ChartCanvas
{
    get { return chartCanvas; }
    set { chartCanvas = value; }
}

public bool IsColorBar
{
    get { return isColorBar; }
    set { isColorBar = value; }
}

public Brush AxisColor
{
    get { return axisColor; }
    set { axisColor = value; }
}
public AxisPatternEnum AxisPattern
{
    get { return axisPattern; }
    set { axisPattern = value; }
}

public double AxisThickness
{
    get { return axisThickness; }
    set { axisThickness = value; }
}

public Brush GridlineColor
{
    get { return gridlineColor; }
    set { gridlineColor = value; }
}

public GridlinePatternEnum GridlinePattern
{
    get { return gridlinePattern; }
    set { gridlinePattern = value; }
}

public double GridlineThickness
{
    get { return gridlineThickness; }
    set { gridlineThickness = value; }
}

public FontFamily LabelFont
{
    get { return labelFont; }
    set { labelFont = value; }
}

public Brush LabelColor
{
    get { return labelColor; }
    set { labelColor = value; }
}

public double LabelFontSize
{
```

```
        get { return labelFontSize; }
        set { labelFontSize = value; }
    }

    public FontFamily TitleFont
    {
        get { return titleFont; }
        set { titleFont = value; }
         }

    public Brush TitleColor
    {
        get { return titleColor; }
        set { titleColor = value; }
    }

    public double TitleFontSize
    {
        get { return titleFontSize; }
        set { titleFontSize = value; }
    }

    public FontFamily TickFont
    {
        get { return tickFont; }
        set { tickFont = value; }
    }

    public Brush TickColor
    {
        get { return tickColor; }
        set { tickColor = value; }
    }

    public double TickFontSize
    {
        get { return tickFontSize; }
        set { tickFontSize = value; }
    }

    public bool IsXGrid
    {
        get { return isXGrid; }
        set { isXGrid = value; }
    }

    public bool IsYGrid
    {
        get { return isYGrid; }
        set { isYGrid = value; }
    }

    public bool IsZGrid
    {
        get { return isZGrid; }
        set { isZGrid = value; }
    }

    public string Title
    {
        get { return title; }
                set { title = value; }
    }
```

```
public string XLabel
{
   get { return xLabel; }
   set { xLabel = value; }
}
public string YLabel
{
   get { return yLabel; }
   set { yLabel = value; }
}
public string ZLabel
{
   get { return zLabel; }
   set { zLabel = value; }
}
public double Elevation
{
   get { return elevation; }
   set
   {
      elevation = value; }
}
public double Azimuth
{
   get { return azimuth; }
   set { azimuth = value; }
    }

public double Xmax
{
   get { return xmax; }
   set { xmax = value; }
}

public double Xmin
{
   get { return xmin; }
   set { xmin = value; }
}

public double Ymax
{
   get { return ymax; }
   set { ymax = value; }
}
public double Ymin
{
   get { return ymin; }
   set { ymin = value; }
}

public double Zmax
{
   get { return zmax; }
   set { zmax = value; }
}

public double Zmin
{
   get { return zmin; }
   set { zmin = value; }
}
```

```
public double XTick
{
    get { return xtick; }
    set { xtick = value; }
}

public double YTick
{
    get { return ytick; }
    set { ytick = value; }
}

public double ZTick
{
    get { return ztick; }
    set { ztick = value; }
}

public enum GridlinePatternEnum
{
    Solid = 1,
    Dash = 2,
    Dot = 3,
    DashDot = 4
}

public enum AxisPatternEnum
{
    Solid = 1,
    Dash = 2,
    Dot = 3,
    DashDot = 4
}

private Point3D[] CoordinatesOfChartBox()
{
    Point3D[] pta = new Point3D[8];
    pta[0] = new Point3D(Xmax, Ymin, Zmin);
    pta[1] = new Point3D(Xmin, Ymin, Zmin);
    pta[2] = new Point3D(Xmin, Ymax, Zmin);
    pta[3] = new Point3D(Xmin, Ymax, Zmax);
    pta[4] = new Point3D(Xmin, Ymin, Zmax);
    pta[5] = new Point3D(Xmax, Ymin, Zmax);
    pta[6] = new Point3D(Xmax, Ymax, Zmax);
    pta[7] = new Point3D(Xmax, Ymax, Zmin);
    Point3D[] pts = new Point3D[4];
    int[] npts = new int[4] { 0, 1, 2, 3 };
    if (elevation >= 0)
    {
        if (azimuth >= -180 && azimuth < -90)
            npts = new int[4] { 1, 2, 7, 6 };
        else if (azimuth >= -90 && azimuth < 0)
            npts = new int[4] { 0, 1, 2, 3 };
        else if (azimuth >= 0 && azimuth < 90)
            npts = new int[4] { 7, 0, 1, 4 };
        else if (azimuth >= 90 && azimuth <= 180)
            npts = new int[4] { 2, 7, 0, 5 };
    }
    else if (elevation < 0)
    {
        if (azimuth >= -180 && azimuth < -90)
            npts = new int[4] { 1, 0, 7, 6 };
        else if (azimuth >= -90 && azimuth < 0)
```

```
            npts = new int[4] { 0, 7, 2, 3 };
        else if (azimuth >= 0 && azimuth < 90)
            npts = new int[4] { 7, 2, 1, 4 };
        else if (azimuth >= 90 && azimuth <= 180)
            npts = new int[4] { 2, 1, 0, 5 };
    }
    for (int i = 0; i < 4; i++) pts[i] = pta[npts[i]];
    return pts;
}
private void AddAxes()
{
    Matrix3D m = Utility.AzimuthElevation(Elevation, Azimuth);
    Point3D[] pts = CoordinatesOfChartBox();
    for (int i = 0; i < pts.Length; i++)
        {
            pts[i] = Normalize3D(m, pts[i]);
        }
  //  DrawAxisLine(pts[0], pts[1]);
  //  DrawAxisLine(pts[1], pts[2]);
  //  DrawAxisLine(pts[2], pts[3]);
}
public Point3D Normalize3D(Matrix3D m, Point3D pt)
{
    Point3D result = new Point3D();

    // Normalize the point:
    double x1 = (pt.X - Xmin) / (Xmax - Xmin) - 0.5;
    double y1 = (pt.Y - Ymin) / (Ymax - Ymin) - 0.5;
    double z1 = (pt.Z - Zmin) / (Zmax - zmin) - 0.5;

    // Perform transformation on the point using matrix m:
    result.X = m.Transform(new Point3D(x1, y1, z1)).X;
    result.Y = m.Transform(new Point3D(x1, y1, z1)).Y;

    // Coordinate transformation from World to Device system:
    double xShift = 1.05;
    double xScale = 1;
    double yShift = 1.05;
    double yScale = 0.9;
    if (Title == "No Title")
    {
        yShift = 0.95;
        yScale = 1;
    }
    if (IsColorBar)
    {
        xShift = 0.95;
        xScale = 0.9;
    }
    result.X = (xShift + xScale * result.X) * ChartCanvas.Width / 2;
    result.Y = (yShift - yScale * result.Y) * ChartCanvas.Height / 2;
    return result;
}

private void AddTicks()
{
    Matrix3D m = Utility.AzimuthElevation(Elevation, Azimuth);
    Point3D[] pta = new Point3D[2];
    Point3D[] pts = CoordinatesOfChartBox();

    // Add x ticks:
    double offset = (Ymax - Ymin) / 30.0;
    double ticklength = offset;
```

```
        for (double x = Xmin; x <= Xmax; x = x + XTick)
        {
            if (Elevation >= 0)
            {
                if (Azimuth >= -90 && Azimuth < 90) ticklength = -offset;
            }
            else if (Elevation < 0)
            {
                if ((Azimuth >= -180 && Azimuth < -90) || Azimuth >= 90 && Azimuth <= 180) ticklength = -
(Ymax - Ymin) / 30;
            }
            pta[0] = new Point3D(x, pts[1].Y + ticklength, pts[1].Z);
            pta[1] = new Point3D(x, pts[1].Y, pts[1].Z);
            for (int i = 0; i < pta.Length; i++)
            {
                pta[i] = Normalize3D(m, pta[i]);
            }
            AddTickLine(pta[0], pta[1]);
        }

      // Add y ticks:
        offset = (Xmax - Xmin) / 30.0;
        ticklength = offset;
        for (double y = Ymin; y <= Ymax; y = y + YTick)
        {
            pts = CoordinatesOfChartBox();
            if (Elevation >= 0)
            {
                if (Azimuth >= -180 && Azimuth < 0) ticklength = -offset;
            }
            else if (Elevation < 0)
            {
                if (Azimuth >= 0 && Azimuth < 180) ticklength = -offset;
            }
            pta[0] = new Point3D(pts[1].X + ticklength, y, pts[1].Z);
            pta[1] = new Point3D(pts[1].X, y, pts[1].Z);
            for (int i = 0; i < pta.Length; i++)
            {
                pta[i] = Normalize3D(m, pta[i]);
            }
            AddTickLine(pta[0], pta[1]);
        }
            // Add z ticks:
        double xoffset = (Xmax - Xmin) / 45.0f;
        double yoffset = (Ymax - Ymin) / 20.0f;
        double xticklength = xoffset;
        double yticklength = yoffset;
        for (double z = Zmin; z <= Zmax; z = z + ZTick)
        {
            if (Elevation >= 0)
            {
                if (Azimuth >= -180 && Azimuth < -90)
                {
                    xticklength = 0;
                    yticklength = yoffset;
                }
                else if (Azimuth >= -90 && Azimuth < 0)
                {
                    xticklength = xoffset;
                    yticklength = 0;
                }
                else if (Azimuth >= 0 && Azimuth < 90)
                {
```

```
                xticklength = 0;
                yticklength = -yoffset;
            }
            else if (Azimuth >= 90 && Azimuth <= 180)
            {
                xticklength = -xoffset;
                yticklength = 0;
             }
        }
        else if (Elevation < 0)
        {
            if (Azimuth >= -180 && Azimuth < -90)
            {
                yticklength = 0;
                xticklength = xoffset;
            }
            else if (Azimuth >= -90 && Azimuth < 0)
            {
                yticklength = -yoffset;
                xticklength = 0;
            }
            else if (Azimuth >= 0 && Azimuth < 90)
            {
                yticklength = 0;
                xticklength = -xoffset;
            }
            else if (Azimuth >= 90 && Azimuth <= 180)
            {
                yticklength = yoffset;
                xticklength = 0;
            }
        }
        pta[0] = new Point3D(pts[2].X, pts[2].Y, z);
        pta[1] = new Point3D(pts[2].X + yticklength, pts[2].Y + xticklength, z);
        for (int i = 0; i < pta.Length; i++)
        {
            pta[i] = Normalize3D(m, pta[i]);
        }
        AddTickLine(pta[0], pta[1]);
    }
}

    private void AddTickLine(Point3D pt1, Point3D pt2)
    {
        Line line = new Line();
        line.Stroke = AxisColor;
        line.X1 = pt1.X;
        line.Y1 = pt1.Y;
        line.X2 = pt2.X;
        line.Y2 = pt2.Y;
        ChartCanvas.Children.Add(line);
    }

    private void AddGridlines()
    {
        Matrix3D m = Utility.AzimuthElevation(Elevation, Azimuth);
        Point3D[] pta = new Point3D[3];
        Point3D[] pts = CoordinatesOfChartBox();
        // Draw x gridlines:
        if (IsXGrid)
        {
            for (double x = Xmin; x <= Xmax; x = x + XTick)
            {
```

```
            pts = CoordinatesOfChartBox();
            pta[0] = new Point3D(x, pts[1].Y, pts[1].Z);
            if (Elevation >= 0)
            {
                if ((Azimuth >= -180 && Azimuth < -90) || (Azimuth >= 0 && Azimuth < 90))
                {
                    pta[1] = new Point3D(x, pts[0].Y, pts[1].Z);
                    pta[2] = new Point3D(x, pts[0].Y, pts[3].Z);
                }
                else
                {
                    pta[1] = new Point3D(x, pts[2].Y, pts[1].Z);
                    pta[2] = new Point3D(x, pts[2].Y, pts[3].Z);
                }
            }
            else if (Elevation < 0)
            {
                if ((Azimuth >= -180 && Azimuth < -90) || (Azimuth >= 0 && Azimuth < 90))
            pta[0] = new Point3D(pts[1].X, y, pts[1].Z);
            if (Elevation >= 0)
            {
                if ((Azimuth >= -180 && Azimuth < -90) || (Azimuth >= 0 && Azimuth < 90))
                {
                    pta[1] = new Point3D(pts[2].X, y, pts[1].Z);
                    pta[2] = new Point3D(pts[2].X, y, pts[3].Z);
                }
                else
                {
                    pta[1] = new Point3D(pts[0].X, y, pts[1].Z);
                    pta[2] = new Point3D(pts[0].X, y, pts[3].Z);
                }
            }
            if (elevation < 0)
            {
                if ((Azimuth >= -180 && Azimuth < -90) || (Azimuth >= 0 && Azimuth < 90))
                {
                    pta[1] = new Point3D(pts[0].X, y, pts[1].Z);
                    pta[2] = new Point3D(pts[0].X, y, pts[3].Z);
                }
                else
                {
                    pta[1] = new Point3D(pts[2].X, y, pts[1].Z);
                    pta[2] = new Point3D(pts[2].X, y, pts[3].Z);
                }
            }
            for (int i = 0; i < pta.Length; i++)
            {
                pta[i] = Normalize3D(m, pta[i]);
            }
            DrawGridline(pta[0], pta[1]);
            DrawGridline(pta[1], pta[2]);
        }
    }
    // Draw Z gridlines:
    if (IsZGrid)
    {
        for (double z = Zmin; z <= Zmax; z = z + ZTick)
        {
            pts = CoordinatesOfChartBox();
            pta[0] = new Point3D(pts[2].X, pts[2].Y, z);
            if (Elevation >= 0)
            {
                if ((Azimuth >= -180 && Azimuth < -90) || (Azimuth >= 0 && Azimuth < 90))
```

```
                {
                    pta[1] = new Point3D(pts[2].X, pts[0].Y, z);
                    pta[2] = new Point3D(pts[0].X, pts[0].Y, z);
                }
                else
                {
                    pta[1] = new Point3D(pts[0].X, pts[2].Y, z); pta[2] = new Point3D(pts[0].X, pts[1].Y, z);
                }
            }
            if (Elevation < 0)
            {
                if ((Azimuth >= -180 && Azimuth < -90) || (Azimuth >= 0 && Azimuth < 90))
                {
                    pta[1] = new Point3D(pts[0].X, pts[2].Y, z);
                    pta[2] = new Point3D(pts[0].X, pts[0].Y, z);
                }
                else
                {
                    pta[1] = new Point3D(pts[2].X, pts[0].Y, z);
                    pta[2] = new Point3D(pts[0].X, pts[0].Y, z);
                }
            }
            for (int i = 0; i < pta.Length; i++)
            {
                pta[i] = Normalize3D(m, pta[i]);
            }
            DrawGridline(pta[0], pta[1]);
            DrawGridline(pta[1], pta[2]);
        }
    }
}

private void DrawGridline(Point3D pt1, Point3D pt2)
{
    gridline = new Line();
    gridline.Stroke = GridlineColor;
    gridline.StrokeThickness = GridlineThickness;
    // AddGridlinePattern();
    gridline.X1 = pt1.X;
    gridline.Y1 = pt1.Y;
    gridline.X2 = pt2.X;
    gridline.Y2 = pt2.Y;
    ChartCanvas.Children.Add(gridline);
}
private void AddLabels()
{
    Matrix3D m = Utility.AzimuthElevation(Elevation, Azimuth);
    Point3D pt = new Point3D();
    Point3D[] pts = CoordinatesOfChartBox();
    TextBlock tb = new TextBlock();
    // Add x tick labels:
    double offset = (Ymax - Ymin) / 20;
    double labelSpace = offset;
    for (double x = Xmin; x <= Xmax; x = x + XTick)
    {
        if (Elevation >= 0)
        {
            if (Azimuth >= -90 && Azimuth < 90) labelSpace = -offset;
        }
        else if (Elevation < 0)
        {
            if ((Azimuth >= -180 && Azimuth < -90) || Azimuth >= 90 && Azimuth <= 180) labelSpace =
-offset;
```

```
        }
        pt = new Point3D(x, pts[1].Y + labelSpace, pts[1].Z);
        pt = Normalize3D(m, pt);
        tb = new TextBlock();
        tb.Text = x.ToString();
        tb.Foreground = TickColor;
        tb.FontFamily = TickFont;
        tb.FontSize = TickFontSize;
        tb.TextAlignment = TextAlignment.Center;
        ChartCanvas.Children.Add(tb);
        Canvas.SetLeft(tb, pt.X);
        Canvas.SetTop(tb, pt.Y);
    }
    // Add y tick labels:
    offset = (Xmax - Xmin) / 20;
    labelSpace = offset;
    for (double y = Ymin; y <= Ymax; y = y + YTick)
    {
        pts = CoordinatesOfChartBox();
        if (elevation >= 0)
        {
            if (azimuth >= -180 && azimuth < 0) labelSpace = -offset;
        }
        else if (elevation < 0)
        {
            if (azimuth >= 0 && azimuth < 180) labelSpace = -offset;
        }
        pt = new Point3D(pts[1].X + labelSpace, y, pts[1].Z);
        pt = Normalize3D(m, pt);
        tb = new TextBlock();
        tb.Text = y.ToString();
        tb.Foreground = TickColor;
        tb.FontFamily = TickFont;
        tb.FontSize = TickFontSize;
        tb.Measure(new Size(Double.PositiveInfinity, Double.PositiveInfinity));
        Size ytickSize = tb.DesiredSize;
        ChartCanvas.Children.Add(tb);
        Canvas.SetLeft(tb, pt.X - ytickSize.Width / 2);
        Canvas.SetTop(tb, pt.Y);
    }

    // Add z tick labels:
    double xoffset = (Xmax - Xmin) / 30.0;
    double yoffset = (Ymax - Ymin) / 15.0;
    double xlabelSpace = xoffset;
    double ylabelSpace = yoffset;
    tb = new TextBlock();
    tb.Text = "A";
    tb.Measure(new Size(Double.PositiveInfinity, Double.PositiveInfinity));
    Size size = tb.DesiredSize;
    for (double z = Zmin; z <= Zmax; z = z + ZTick)
    {
        pts = CoordinatesOfChartBox();
        if (Elevation >= 0)
        {
            if (Azimuth >= -180 && Azimuth < -90)
            {
                xlabelSpace = 0;
                ylabelSpace = yoffset;
            }
            else if (Azimuth >= -90 && Azimuth < 0)
            {
                xlabelSpace = xoffset;
```

```
                ylabelSpace = 0;
            }
            else if (Azimuth >= 0 && Azimuth < 90)
            {
                xlabelSpace = 0;
                ylabelSpace = -yoffset;
            }
            else if (Azimuth >= 90 && Azimuth <= 180)
            {
                xlabelSpace = -xoffset;
                ylabelSpace = 0;
            }
        }
        else if (Elevation < 0)
        {
            if (Azimuth >= -180 && Azimuth < -90)
            {
                ylabelSpace = 0;
                xlabelSpace = xoffset;
            }
            else if (Azimuth >= -90 && Azimuth < 0)
            {
                ylabelSpace = -yoffset;
                xlabelSpace = 0;
            }
            else if (Azimuth >= 0 && Azimuth < 90)
            {
                ylabelSpace = 0;
                xlabelSpace = -xoffset;
            }
            else if (Azimuth >= 90 && Azimuth <= 180)
            {
                ylabelSpace = yoffset;
                xlabelSpace = 0;
            }
        }
        pt = new Point3D(pts[2].X + ylabelSpace, pts[2].Y + xlabelSpace, z);
        pt = Normalize3D(m, pt);
        tb = new TextBlock();
        tb.Text = z.ToString();
        tb.Foreground = TickColor;
        tb.FontFamily = TickFont;
        tb.FontSize = TickFontSize;
        tb.Measure(new Size(Double.PositiveInfinity, Double.PositiveInfinity));
        Size ztickSize = tb.DesiredSize;
        ChartCanvas.Children.Add(tb);
        Canvas.SetLeft(tb, pt.X - ztickSize.Width - 1);
        Canvas.SetTop(tb, pt.Y - ztickSize.Height / 2);
    }

    // Add Title:
    tb = new TextBlock();
    tb.Text = Title;
    tb.Foreground = TitleColor;
    tb.FontSize = TitleFontSize;
    tb.FontFamily = TitleFont;
    tb.Measure(new Size(Double.PositiveInfinity, Double.PositiveInfinity));
    Size titleSize = tb.DesiredSize;
    if (tb.Text != "No Title")
    {
        ChartCanvas.Children.Add(tb);
        Canvas.SetLeft(tb, ChartCanvas.Width / 2 - titleSize.Width / 2);
        Canvas.SetTop(tb, ChartCanvas.Height / 30);
```

```
        }
        // Add x axis label:
        offset = (Ymax - Ymin) / 3;
        labelSpace = offset;
        double offset1 = (Xmax - Xmin) / 10;
        double xc = offset1;
        if (Elevation >= 0)
        {
            if (Azimuth >= -90 && Azimuth < 90) labelSpace = -offset;
            if (Azimuth >= 0 && Azimuth <= 180) xc = -offset1;
        }
        else if (Elevation < 0)
        {
            if ((Azimuth >= -180 && Azimuth < -90) || Azimuth >= 90 && Azimuth <= 180) labelSpace = -
offset;
            if (Azimuth >= -180 && Azimuth <= 0) xc = -offset1;
        }
        Point3D[] pta = new Point3D[2];
        pta[0] = new Point3D(Xmin, pts[1].Y + labelSpace, pts[1].Z);
        pta[1] = new Point3D((Xmin + Xmax) / 2 - xc, pts[1].Y + labelSpace, pts[1].Z);
        pta[0] = Normalize3D(m, pta[0]);
        pta[1] = Normalize3D(m, pta[1]);
        double theta = Math.Atan((pta[1].Y - pta[0].Y) / (pta[1].X - pta[0].X));
        theta = theta * 180 / Math.PI;
        tb = new TextBlock();
        tb.Text = XLabel;
        tb.Foreground = LabelColor;
        tb.FontFamily = LabelFont;
        tb.FontSize = LabelFontSize;
        tb.Measure(new Size(Double.PositiveInfinity, Double.PositiveInfinity));
        Size xLabelSize = tb.DesiredSize;
        TransformGroup tg = new TransformGroup();
        RotateTransform rt = new RotateTransform(theta, 0.5, 0.5);
        TranslateTransform tt = new TranslateTransform(pta[1].X +
        xLabelSize.Width / 2, pta[1].Y - xLabelSize.Height / 2);
        tg.Children.Add(rt);
        tg.Children.Add(tt);
        tb.RenderTransform = tg;
        ChartCanvas.Children.Add(tb);
        // Add y axis label:
        offset = (Xmax - Xmin) / 3;
        offset1 = (Ymax - Ymin) / 5;
        labelSpace = offset;
        double yc = YTick;
        if (Elevation >= 0)
        {
            if (Azimuth >= -180 && Azimuth < 0) labelSpace = -offset;
            if (Azimuth >= -90 && Azimuth <= 90) yc = -offset1;
        }
        else if (Elevation < 0)
        {
            yc = -offset1;
            if (Azimuth >= 0 && Azimuth < 180) labelSpace = -offset;
            if (Azimuth >= -90 && Azimuth <= 90)  yc = offset1;
        }
        pta[0] = new Point3D(pts[1].X + labelSpace, Ymin, pts[1].Z);
        pta[1] = new Point3D(pts[1].X + labelSpace, (Ymin + Ymax) / 2 + yc, pts[1].Z);
        pta[0] = Normalize3D(m, pta[0]);
        pta[1] = Normalize3D(m, pta[1]);
        theta = (double)Math.Atan((pta[1].Y - pta[0].Y) / (pta[1].X - pta[0].X));
        theta = theta * 180 / (double)Math.PI;
        tb = new TextBlock();
        tb.Text = YLabel;
```

```
tb.Foreground = LabelColor;
tb.FontFamily = LabelFont;
tb.FontSize = LabelFontSize;
tb.Measure(new Size(Double.PositiveInfinity, Double.PositiveInfinity));
Size yLabelSize = tb.DesiredSize;
tg = new TransformGroup();
tt = new TranslateTransform(pta[1].X - yLabelSize.Width / 2,
 pta[1].Y - yLabelSize.Height / 2);
rt = new RotateTransform(theta, 0.5, 0.5);
tg.Children.Add(rt);
tg.Children.Add(tt);
tb.RenderTransform = tg;
ChartCanvas.Children.Add(tb);

// Add z axis labels:
double zticklength = 10;
labelSpace = -1.3f * offset;
offset1 = (Zmax - Zmin) / 8;
double zc = -offset1;
for (double z = Zmin; z < Zmax; z = z + ZTick)
{
    tb = new TextBlock();
    tb.Text = z.ToString();
    tb.Measure(new Size(Double.PositiveInfinity, Double.PositiveInfinity));
    Size size1 = tb.DesiredSize;
    if (zticklength < size1.Width) zticklength = size1.Width;
}
double zlength = -zticklength;
if (Elevation >= 0)
{
    if (Azimuth >= -180 && Azimuth < -90)
    {
        zlength = -zticklength;
        labelSpace = -1.3f * offset;
        zc = -offset1;
    }
    else if (Azimuth >= -90 && Azimuth < 0)
    {
        zlength = zticklength;
        labelSpace = 2 * offset / 3;
        zc = offset1;
    }
    else if (Azimuth >= 0 && Azimuth < 90)
    {
        zlength = zticklength;
        labelSpace = 2 * offset / 3;
        zc = -offset1;
    }
    else if (Azimuth >= 90 && Azimuth <= 180)
    {
        zlength = -zticklength;
        labelSpace = -1.3f * offset;
        zc = offset1;
    }
}
else if (Elevation < 0)
{
    if (Azimuth >= -180 && Azimuth < -90)
    {
        zlength = -zticklength;
        labelSpace = -1.3f * offset;
        zc = offset1;
    }
```

```
            else if (Azimuth >= -90 && Azimuth < 0)
            {
                zlength = zticklength;
                labelSpace = 2 * offset / 3;
                zc = -offset1;
            }
            else if (Azimuth >= 0 && Azimuth < 90)
            {
                zlength = zticklength;
                labelSpace = 2 * offset / 3;
                zc = offset1;
            }
            else if (Azimuth >= 90 && Azimuth <= 180)
            {
                zlength = -zticklength;
                labelSpace = -1.3f * offset;
                zc = -offset1;
            }
        }
        pta[0] = new Point3D(pts[2].X - labelSpace, pts[2].Y, (Zmin + Zmax) / 2 + zc);
        pta[0] = Normalize3D(m, pta[0]);
        tb = new TextBlock();
        tb.Text = ZLabel;
        tb.Foreground = LabelColor;
        tb.FontFamily = LabelFont;
        tb.FontSize = LabelFontSize;
        tb.Measure(new Size(Double.PositiveInfinity, Double.PositiveInfinity));
        Size zLabelSize = tb.DesiredSize;
        tg = new TransformGroup();
        tt = new TranslateTransform(pta[0].X - zlength,
        pta[0].Y + zLabelSize.Width / 2);
        rt = new RotateTransform(270, 0.5, 0.5);
        tg.Children.Add(rt);
        tg.Children.Add(tt);
        tb.RenderTransform = tg;
        ChartCanvas.Children.Add(tb);
    }
        public void AddChartStyle()
        {
            AddTicks();
            AddGridlines();
            AddAxes();
            AddLabels();
        }

    }
}
```

Η τάξη DataSeriesLine3D χρησιμοποιείται για την δημιουργία του γραφήματος καθώς περιλαμβάνει τις μεθόδου που ζωγραφίζεται το υπερπολύεδρο.

```
using System;
using System.Collections.Generic;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Media;
using System.Windows.Media.Media3D;
using System.Windows.Shapes;
namespace Chart3DNoWPFEngine
{
```

```
public class DataSeriesLine3D
{
    private Polyline lineSeries = new Polyline();
    private Brush lineColor;
    private double lineThickness = 1;
    private LinePatternEnum linePattern;
    private List<Point3D> point3DList = new List<Point3D>();
    private List<Point3D> point3Dlabel = new List<Point3D>();
    public List<Point3D> Point3DList
    {
        get { return point3DList; }
        set { point3DList = value; }
    }
    public List<Point3D> Point3DLabel
    {
        get { return point3Dlabel; }
        set { point3Dlabel = value; }
    }
    public Brush LineColor
    {
        get { return lineColor; }
        set { lineColor = value; }
    }

    public Polyline LineSeries
    {
        get { return lineSeries; }
        set { lineSeries = value; }
    }
    public double LineThickness
    {
        get { return lineThickness; }
        set { lineThickness = value; }
    }

    public LinePatternEnum LinePattern
    {
        get { return linePattern; }
        set { linePattern = value; }
    }

    public void AddLinePattern()
    {
        LineSeries.Stroke = LineColor;
        LineSeries.StrokeThickness = LineThickness;
        switch (LinePattern)
        {
            case LinePatternEnum.Dash:
                LineSeries.StrokeDashArray =
                    new DoubleCollection(new double[2] { 4, 3 });
                break;
            case LinePatternEnum.Dot:
                LineSeries.StrokeDashArray =
                    new DoubleCollection(new double[2] { 1, 2 });
                break;
            case LinePatternEnum.DashDot:
                LineSeries.StrokeDashArray =
                    new DoubleCollection(new double[4] { 4, 2, 1, 2 });
                break;
            case LinePatternEnum.None:
                LineSeries.Stroke = Brushes.Transparent;
                break;
        }
```

```
        }

        public enum LinePatternEnum
        {
            Solid = 1,
            Dash = 2,
            Dot = 3,
            DashDot = 4,
            None = 5
        }

        public void AddLine3D(ChartStyle cs)
        {
            Matrix3D m = Utility.AzimuthElevation(cs.Elevation, cs.Azimuth);
            Point3D[] pts = new Point3D[Point3DList.Count];
            for (int i = 0; i < Point3DList.Count; i++)
            {
                pts[i] = cs.Normalize3D(m, Point3DList[i]);
                LineSeries.Points.Add(new Point(pts[i].X, pts[i].Y));
            }
            AddLinePattern();
            cs.ChartCanvas.Children.Add(LineSeries);
        }
        public void AddPointsLabels(ChartStyle ls)
        {
            Matrix3D m = Utility.AzimuthElevation(ls.Elevation, ls.Azimuth);
            Point3D pts = new Point3D();
            for (int i = 0; i < Point3DLabel.Count; i++)
            {
                pts = ls.Normalize3D(m, Point3DLabel[i]);
                //LineSeries.Points.Add(new Point(pts[i].X, pts[i].Y));
                // na typvnei simeia
            TextBlock tb= new TextBlock();
            tb.Text = "P"+Convert.ToString(i);
           // tb.Foreground = TickColor;
            tb.FontFamily = new FontFamily("Arial Narrow");
         //   tb.FontSize = TickFontSize;
            tb.Measure(new Size(Double.PositiveInfinity, Double.PositiveInfinity));
            Size ztickSize = tb.DesiredSize;
            Canvas.SetLeft(tb, pts.X);
            Canvas.SetTop(tb, pts.Y);
            ls.ChartCanvas.Children.Add(tb);
            }
        }
    }
}
```

Το παράθυρο διαλόγου ανεπτυγμένο σε WPF.

```
<Window x:Class="Chart3DNoWPFEngine.Line3D"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Line3D" mc:Ignorable="d" xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006" Height="894" Width="1196"
Loaded="Window_Loaded">
    <Grid Width="1081" Height="783">
        <Grid.RowDefinitions>
            <RowDefinition Height="772*" />
            <RowDefinition Height="11*" />
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
```

```
            <ColumnDefinition Width="199"/>
            <ColumnDefinition Width="*"/>
        </Grid.ColumnDefinitions>
        <StackPanel Margin="0,10,0,0" Grid.RowSpan="2">
            <TextBlock Margin="0,0,0,5" Text="Elevation:" Height="22" Width="208" />
            <TextBox x:Name="tbElevation1" Text="0" Margin="0,0,0,5"
                     TextAlignment="Center" Width="75" TextChanged="tbElevation1_TextChanged" />
            <ScrollBar Height="135" Name="HElevetionBar" Width="21"
ValueChanged="scrollBar1_ValueChanged" Maximum="90" Minimum="-90" />
            <TextBlock Margin="0,0,0,5" Text="Azimuth:" Height="23" Width="209" />
            <TextBox x:Name="tbAzimuth1" Text="0" Margin="0,0,0,5"
                     TextAlignment="Center" Width="75" TextChanged="tbAzimuth1_TextChanged" />
            <ScrollBar Height="20" Name="scrollBar3" Width="151" IsEnabled="True"
Orientation="Horizontal"  Margin="0,0,0,5" ValueChanged="scrollBar3_ValueChanged" Minimum="-180"
Maximum="180" />
            <Button x:Name="Apply" Content="Apply" Click="Apply_Click"
                    Margin="0 20,0,0" Width="75"/>
            <Label Content="X Axis:" Height="27" Name="LblX" Width="107" />
            <ListBox Height="71" Name="LstX" Width="88" DataContext="{Binding}"
SelectionChanged="LstX_SelectionChanged">
                <ListBoxItem Content="Crit1" Selected="ListBoxItem_Selected_1" />
                <ListBoxItem Content="Crit2" />
                <ListBoxItem Content="Crit3" />
                <ListBoxItem Content="Crit4" />
                <ListBoxItem Content="Crit5" />
            </ListBox>
            <Label Content="Y Axis:" Height="28" Name="LblY" Width="105" />
            <ListBox DataContext="{Binding}" Height="64" Name="LstY" Width="77"
SelectionChanged="LstY_SelectionChanged">
                <ListBoxItem Content="Crit1"  />
                <ListBoxItem Content="Crit2"  />
                <ListBoxItem Content="Crit3" />
                <ListBoxItem Content="Crit4" />
                <ListBoxItem Content="Crit5" />
            </ListBox>
            <Label Content="Z Axis:" Height="28" Name="LblZ" Width="105" />
            <ListBox DataContext="{Binding}" Height="68" Name="LstZ" Width="80"
SelectionChanged="LstZ_SelectionChanged">
                <ListBoxItem Content="Crit1" />
                <ListBoxItem Content="Crit2"  />
                <ListBoxItem Content="Crit3" />
                <ListBoxItem Content="Crit4" />
                <ListBoxItem Content="Crit5" />
            </ListBox>
            <Label Content="Distances" Height="28" Name="label1" />
            <ListBox Height="137" Name="LstDist" Width="152" />
        </StackPanel>
        <Grid x:Name="chartGrid1" SizeChanged="chartGrid_SizeChanged" Margin="49,12,51,0"
Grid.Column = "1">

        </Grid>
        <Canvas x:Name="chartCanvas" Background="Transparent" ClipToBounds="True"
Grid.Column="1" Margin="18,0,12,17" />
        <Border BorderBrush="Gray" BorderThickness="1" Margin="29,0,28,0"
Grid.Column="1"></Border>
    </Grid>
</Window>
```

ΤΟ πρόγραμμα

```
using System;
using System.Collections.Generic;
```

```
using System.Windows;
using System.Windows.Controls;
using System.Windows.Media;
using System.Windows.Media.Media3D;

namespace Chart3DNoWPFEngine
{
   public partial class Line3D : Window
   {
      private ChartStyle cs;
      private DataSeriesLine3D ds;
      private int xax = 0;
      private int yax = 1;
      private int zax = 2;

      public Line3D()
      {
         InitializeComponent();
      }

      private void chartGrid_SizeChanged(object sender, SizeChangedEventArgs e)
      {
         chartCanvas.Width = chartGrid1.ActualWidth;
         chartCanvas.Height = chartGrid1.ActualHeight;
         AddChart();
      }

      private void AddChart()
      {
         chartCanvas.Children.Clear();
         cs = new ChartStyle();
         ds = new DataSeriesLine3D();
         cs.ChartCanvas = this.chartCanvas;
         cs.GridlinePattern = ChartStyle.GridlinePatternEnum.Solid;
         cs.Elevation = double.Parse(tbElevation1.Text);
         cs.Azimuth = double.Parse(tbAzimuth1.Text);
         cs.Xmin = 0;
         cs.Xmax = 1;
         cs.Ymin = 0;
         cs.Ymax = 1;
         cs.Zmin = 0;
         cs.Zmax = 1;
         cs.XTick = 0.1;
         cs.YTick = 0.1;
         cs.ZTick = 0.1;
         cs.Title = "No Title";
         cs.AddChartStyle();
         ds.LineColor = Brushes.Red;

         double[,] datapoints = { {0.2, 0.35, 0.15, 0.30, 0.40},
                                  {0.4, 0.5, 0.12, 0.22, 0.14},
                                  {0.17, 0.19, 0.32, 0.24, 0.43},
                                  {0.36, 0.54, 0.71, 0.09, 0.65},
                                  {0.78, 0.54, 0.42, 0.32, 0.11}};
         int dimdatx = 5;
         int dimdaty = 5;
         double[,] distances = new double[dimdatx, dimdaty];
         LstDist.Items.Clear();
         for (int ia = 0; ia < dimdatx; ia++)
         {
            for (int ja = ia + 1; ja < dimdatx; ja++)
            {
               for (int ka = 0; ka < dimdaty; ka++)
```

```
                {
                    distances[ia, ja] = distances[ia, ja] + Math.Pow((datapoints[ia, ka] - datapoints[ja, ka]), 2);
                }
                distances[ia, ja] = Math.Sqrt(distances[ia, ja] / dimdaty);
                LstDist.Items.Add("P" + Convert.ToString(ia) + ", P" + Convert.ToString(ja) + "  dist: " +
Convert.ToString(distances[ia, ja]));
            }
        }


        //AB
        double x = datapoints[0, xax];
        double y = datapoints[0, yax];
        double z = datapoints[0, zax];
        ds.Point3DList.Add(new Point3D(x, y, z));
        x = datapoints[1, xax];
        y = datapoints[1, yax];
        z = datapoints[1, zax];
        ds.Point3DList.Add(new Point3D(x, y, z));
        //BG
        x = datapoints[2, xax];
        y = datapoints[2, yax];
        z = datapoints[2, zax];
        ds.Point3DList.Add(new Point3D(x, y, z));
        //GA
        x = datapoints[0, xax];
        y = datapoints[0, yax];
        z = datapoints[0, zax];
        ds.Point3DList.Add(new Point3D(x, y, z));
        //AE
        x = datapoints[4, xax];
        y = datapoints[4, yax];
        z = datapoints[4, zax];
        ds.Point3DList.Add(new Point3D(x, y, z));
        //Eg
        x = datapoints[2, xax];
        y = datapoints[2, yax];
        z = datapoints[2, zax];
        ds.Point3DList.Add(new Point3D(x, y, z));
        //gd
        x = datapoints[3, xax];
        y = datapoints[3, yax];
        z = datapoints[3, zax];
        ds.Point3DList.Add(new Point3D(x, y, z));
        //DA
        x = datapoints[0, xax];
        y = datapoints[0, yax];
        z = datapoints[0, zax]; ;
        ds.Point3DList.Add(new Point3D(x, y, z));
        //ab
        x = datapoints[1, xax];
        y = datapoints[1, yax];
        z = datapoints[1, zax]; ;
        //BE
        ds.Point3DList.Add(new Point3D(x, y, z));
        x = datapoints[4, xax];
        y = datapoints[4, yax];
        z = datapoints[4, zax];
        ds.Point3DList.Add(new Point3D(x, y, z));
        //ED
        x = datapoints[3, xax];
        y = datapoints[3, yax];
        z = datapoints[3, zax];
```

```
            ds.Point3DList.Add(new Point3D(x, y, z));
            ds.AddLine3D(cs);

            for (int i = 0; i < dimdatx; i++)
            {
                x = datapoints[i, xax];
                y = datapoints[i, yax];
                z = datapoints[i, zax];
                ds.Point3DLabel.Add(new Point3D(x, y, z));
            }
            ds.AddPointsLabels(cs);

        }

        private void Apply_Click(object sender, RoutedEventArgs e)
        {
            AddChart();
        }


        private void scrollBar3_ValueChanged(object sender, RoutedPropertyChangedEventArgs<double>
e)
        {
            tbAzimuth1.Text = Convert.ToString(scrollBar3.Value);
            AddChart();
        }

        private void Window_Loaded(object sender, RoutedEventArgs e)
        {

        }

        private void tbElevation1_TextChanged(object sender, TextChangedEventArgs e)
        {
            //scrollBar2.Value = 30; double.Parse(tbElevation1.Text);
        }

        private void tbAzimuth1_TextChanged(object sender, TextChangedEventArgs e)
        {
          //  scrollBar3.Value = double.Parse(tbAzimuth1.Text);
        }

        private void scrollBar1_ValueChanged(object sender, RoutedPropertyChangedEventArgs<double>
e)
        {
            tbElevation1.Text = Convert.ToString(HElevetionBar.Value);
            AddChart();
        }


        private void LstX_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            xax = LstX.SelectedIndex;
            LblX.Content = "X Axis: Crit:" + Convert.ToString(xax+1);
        }

        private void LstY_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            yax = LstY.SelectedIndex;
            LblY.Content = "X Axis: Crit:" + Convert.ToString(yax+1);
        }

        private void LstZ_SelectionChanged(object sender, SelectionChangedEventArgs e)
```

```
        {
            zax = LstZ.SelectedIndex;
            LblZ.Content = "X Axis: Crit:" + Convert.ToString(zax+1);
        }

        private void ListBoxItem_Selected_1(object sender, RoutedEventArgs e)
        {

        }


    }
}
```