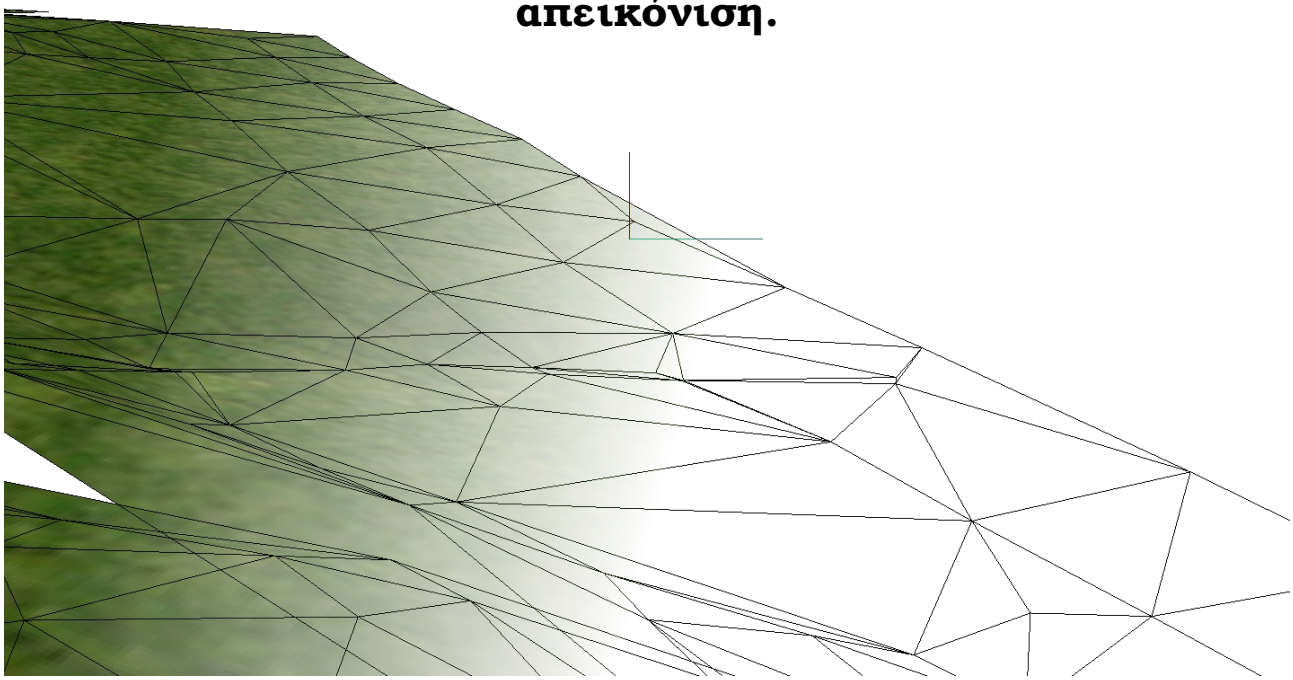
Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Αγρονόμων και Τοπογράφων Μηχανικών
Τομέας Τοπογραφίας

Διαχείριση τοπογραφικών μετρήσεων πεδίου κατά την εκπόνηση γεωδαιτικών ασκήσεων, με τρισδιάστατη απεικόνιση.



Διπλωματική Εργασία

Βιτάλης Στυλιανός

Επιβλέπων: Βεσκούκης Βασίλειος
Επίκουρος Καθηγητής Ε.Μ.Π.

- Αθήνα, Μάρτιος 2010 -

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Αγρονόμων και Τοπογράφων Μηχανικών

Διαχείριση τοπογραφικών μετρήσεων πεδίου κατά την εκπόνηση γεωδαιτικών ασκήσεων, με τρισδιάστατη απεικόνιση.

Διπλωματική Εργασία του Βιτάλη Στυλιανού

Αθήνα, Μάρτιος 2010

.....

Πρόλογος - Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή, κ. Βεσκούκη, για την ανάθεση τη διπλωματικής, τη σημαντική βοήθεια που μου προσέφερε κατά την εκπόνηση αυτή της Διπλωματικής Εργασίας, αλλά και για τη γενική συνεισφορά του στη δόμηση και ανάπτυξη με τις προτάσεις και υποδείξεις του, κατά το διάστημα αυτό της συνεργασίας μας.

Θα ήθελα, επίσης, να ευχαριστήσω τον κ. Πανταζή για τη συμμετοχή του στην αρχική διαμόρφωση του θέματος της διπλωματικής και την προσφορά των δοκιμαστικών μετρήσεων που αποδείχθηκαν απαραίτητα καθ' όλη αυτή τη διάρκεια ανάπτυξης της εφαρμογής της εργασίας.

Τέλος, για την ψυχολογική υποστήριξη αλλά και για τη συνολική συμπαράστασή που μου έχουν προσφέρει όλα αυτά τα χρόνια, θα ήθελα να ευχαριστήσω τους γονείς μου, τον αδερφό μου, τους φίλους μου και όλους αυτούς που είναι δίπλα μου.

Πίνακας Περιεχομένων

Κατάλογος Σχημάτων

Περίληψη

Σκοπός της **Διπλωματικής Εργασίας** αυτής είναι η ανάπτυξη ενός λογισμικού **διαχείρισης τοπογραφικών μετρήσεων αποτύπωσης** με χρήση **τρισδιάστατης** απεικόνισης. Παράλληλα, επιδίωξη είναι η διαμόρφωση ενός ευέλικτου *προγραμματιστικού περιβάλλοντος* αποτελούμενου από τις κατάλληλες *δομές*, *κλάσεις* και *βιβλιοθήκες* που θα καθιστούν την ενσωμάτωση των χαρακτηριστικών που υλοποιήθηκαν, εύκολη για κάθε ενδιαφερόμενο στο πεδίο αυτό.

Για την προσέγγιση ενός πιο συγκεκριμένου ρεαλιστικού προβλήματος επιχειρήθηκε η εφαρμογή να προσαρμοστεί στα πρότυπα του μαθήματος *Μεγάλες Γεωδαιτικές Ασκήσεις ΙΙ*, με στόχο να είναι χρήσιμο, για τους επιβλέποντες καθηγητές, στην παρακολούθηση της εξέλιξης των εργασιών και στην επισκόπηση της προόδου των σπουδαστών που συμμετέχουν.

Η ανάπτυξη του λογισμικού έγινε σε ***περιβάλλον .NET***, χρησιμοποιώντας τις ενσωματωμένες δυνατότητες σύνδεσης με τις βάσεις δεδομένων τύπου ***MS Access***. Παράλληλα, για την απεικόνιση των μετρήσεων χρησιμοποιήθηκε το σύνολο βιβλιοθηκών ***DirectX***. Η εκτέλεση των υπολοίπων ενεργειών βασίστηκε σε κώδικα που αναπτύχθηκε ειδικά για τη συγκεκριμένη *Διπλωματική Εργασία*, ή από κώδικα που προέκυψε από ανοικτού κώδικα βιβλιοθήκες (για τον τριγωνισμό **Delaunay** και την εξαγωγή **DXF**) με σεβασμό στην άδεια χρήσης τους.

Τελικά προέκυψε μία εφαρμογή που χρησιμεύει στην **καταχώρηση** δεδομένων μετρήσεων, στην **απεικόνισή** τους σε τρισδιάστατο περιβάλλον, στην **παραγωγή** τοπογραφικών προϊόντων (*τριγωνισμός*, *ισοϋψείς*) και την **εξαγωγή** τους σε *μορφή DXF*, που είναι συμβατή με την πλειοψηφία λογισμικών σχεδίασης.

Δόθηκαν, εν τέλει, κάποιες κατευθύνσεις για τις *δυνατότητες εξέλιξής* του πεδίου **διαχείρισης**, **αυτοματοποίησης** και **απεικόνισης** τοπογραφικών μετρήσεων χρησιμοποιώντας τις *δυνατότητες* των *σύγχρονων υπολογιστικών συστημάτων* και των *νέων τεχνολογιών*.

Abstract

The main object of this **Diploma Thesis** is the development of a software capable to **manage topographic measurement data** with **3D graphics** representation. Alongside, it is aimed to produce an flexible *programming environment* containing the essential *structures*, *classes* and *libraries* which will make those features' integration easier for everyone interested in this field of development.

To approach a more specific real situation of such type, we approached the subject based on the pattern of the course *Wide Geodetic Practice II*, with a target of making the application useful on monitoring the advance of exercises and reviewing the progress of students by the supervisors.

The software developed through the ***.NET framework*** environment, making use of the enhanced interaction with ***MS Access*** type Databases. Meanwhile, ***DirectX SDK*** was integrated for measurement review purposes. The rest of the application is based on source code written specifically for this Thesis, or based on some open source libraries modifications (for ***Delaunay*** *triangulation* and ***DXF*** *export*) with respect to the corresponding license agreement.

As result, we produced an application capable of **storing** measurement data, producing 3D surface **review**, **generating** topographic products (such as *triangulation* and *contours*) and **exporting** those data types on *DXF format*, widely supported by most drawing applications.

Finally, we've given some directions on *evolution possibilities* of measurement data **management**, **automatization** and **representation**, making use of the *modern computer systems'* and new *technologies' capabilities*.

1. Εισαγωγή

1.1 Η τεχνολογία στην επιστήμη της Τοπογραφίας

Η **Σχολή Αγρονόμων και Τοπογράφων Μηχανικών** αποτελεί, εδώ και χρόνια, μία από τις πιο εξελιγμένες, σε θέματα τεχνολογίας και υπολογιστών, τόσο στο χώρο του **Εθνικού Μετσοβίου Πολυτεχνείου**, όσο και μεταξύ όλων των εκπαιδευτικών ιδρυμάτων της **Ελλάδας**. Το αντικείμενο ενασχόλησης της σχολής, άλλωστε, καθιστά αναγκαία την σε καθημερινή βάση ενασχόληση των φοιτητών, καθηγητών και ερευνητών με τους υπολογιστές και τα αυτοματοποιημένα συστήματα, καθώς είναι πλέον μέρος της εργασίας του **Αγρονόμου Τοπογράφου Μηχανικού** η χρήση ηλεκτρονικών συσκευών αλλά και εφαρμογών που απαιτούνται για την ολοκλήρωση κάθε στοιχειώδους **Γεωδαιτικής** (και όχι μόνο) **εργασίας**. Ταυτόχρονα με τη χρήση των τεχνολογιών αυτών, όμως, η **Τοπογραφική επιστήμη** κατέχει κυρίαρχο ρόλο και στην εξέλιξή τους και κυρίως σε θέματα που αφορούν τεχνολογίες επίγειων μετρήσεων με χρήση ψηφιακών σημάτων (EDM και, μετέπειτα, Total Stations) ή με χρήση τεχνολογιών GPS και προχωρημένων τεχνικών πλοήγησης.

Όλα αυτά, αναδεικνύονται περισσότερο κατά την εκτέλεση μετρήσεων μεγάλης κλίμακας, εκεί όπου η διαχείριση του μεγάλου όγκου δεδομένων εξελίσσεται σε ένα ακόμα μεγάλο πρόβλημα που ο **Γεωδαίτης** καλείται να λύσει. Σε τέτοιες περιπτώσεις, κυρίαρχες ανάγκες είναι η *οργάνωση των δεδομένων* με τον καλύτερο δυνατό τρόπο, ώστε να είναι ευκολότερη η αναπαραγωγή και η παραπομπή σε αυτά, αλλά και η *συνεχής επαλήθευση* και *επισκόπηση της εξέλιξης των εργασιών* με τον πλέον γρήγορο και πρακτικό τρόπο. Αυτό, μπορεί να δώσει τη δυνατότητα στους υπευθύνους να επέμβουν όσο το δυνατόν πιο άμεσα και γρήγορα στα σημεία εκείνα που προκύπτουν *σφάλματα μετρήσεων* (*συστηματικά*, *τυχαία* ή *χονδροειδή*) και να πάρουν τις κατάλληλες αποφάσεις για την έγκαιρη αντιμετώπιση προβλημάτων που ενδεχομένως να προκύψουν.

1.2 Οι Μεγάλες Γεωδαιτικές Ασκήσεις

Τέτοιου είδους μεγάλης έκτασης τοπογραφικές εργασίες προκύπτουν κατά την εκπόνηση των **Μεγάλων Γεωδαιτικών Ασκήσεων**, δηλαδή των *θερινών πρακτικών εργασιών* των φοιτητών Τοπογράφων, που λαμβάνουν χώρα μετά το $4^{ο}$ και το $6^{ο}$ *εξάμηνο* της φοίτησής τους στη *Σχολή Αγρονόμων και Τοπογράφων Μηχανικών*. Ιδίως στην περίπτωση του μαθήματος **Μεγάλες Γεωδαιτικές Ασκήσεις ΙΙ**, οι φοιτητές έρχονται αντιμέτωποι με θέματα όπως *η οργάνωση των ατόμων σε ομάδες*, *η μετρήσεις ανεξάρτητα ανά τμήματα* και *η τελική διαχείριση* όλου αυτού του (αρχικά ανεξάρτητου) όγκου δεδομένων για την παραγωγή του *συνολικού χαρτογραφικού αποτελέσματος*. Έτσι, είναι προφανές ότι αυτή η πρακτική

άσκηση είναι η πλέον αντιπροσωπευτική των αντίστοιχων επαγγελματικών έργων αποτύπωσης υψηλής κλίμακας.

1.3 Σκοπός της Διπλωματικής Εργασίας (Στόχοι)

Σκοπός της *Διπλωματικής Εργασίας* αυτής είναι η μελέτη του προβλήματος της διαχείρισης μετρήσεων εργασιών αποτύπωσης υψηλής κλίμακας και η ανάπτυξη των μηχανισμών εκείνων που θα οριοθετήσουν ένα σαφές πλαίσιο από δομές, πρωτόκολλα και διαδικασίες για τη διευθέτηση αυτού του μεγάλου όγκου εργασιών. Με αυτό το πνεύμα, το ζητούμενο είναι *η ανάπτυξη ενός προγραμματιστικού περιβάλλοντος*, παράλληλα με *μία εφαρμογή* που θα διευκολύνουν το έργο της διαχείρισης των δεδομένων που προκύπτουν από τέτοιες εργασίες.

Έτσι, το τελικό αποτέλεσμα είναι **μία εφαρμογή** που:

- ✔ **Αποθηκεύει** και καθιστά δυνατή τη στοιχειώδη **κεντρική διαχείριση** όλων των **δεδομένων** που προκύπτουν από τις **μετρήσεις** στο πεδίο, μέσω ενός κεντρικού και χρηστικού περιβάλλοντος χρήστη

- ✔ **Απεικονίζει** σε **τρισδιάστατο (3D) χώρο** τα δεδομένα αυτά, “αναπαράγοντας” το **ανάγλυφο** της περιοχής με γραφικά πραγματικού χρόνου (real-time graphics)

- ✔ Εκτελεί κάποια βασική **επεξεργασία** επί των δεδομένων και **παράγει πληροφορία** που καθιστά καλύτερη την **επισκόπηση** της κατάστασης, την εξασφάλιση της ποιότητας των μετρήσεων και την ανάδειξη προβλημάτων και ανωμαλιών που πιθανόν να έχουν προκύψει στο πεδίο

- ✔ **Εξάγει** τα δεδομένα που διαχειρίζεται ή παράγει σε μία πρότυπη μορφή ώστε να είναι διαθέσιμα για την εκτέλεση **περαιτέρω επεξεργασίας** από άλλα προγράμματα

Είναι σαφές, λοιπόν, ότι η σκοπιμότητα αυτής της εφαρμογής είναι να δώσει στα χέρια του υπεύθυνου των μετρήσεων ένα εργαλείο που θα του επιτρέπει κάθε στιγμή να έχει σε ελάχιστο χρονικό διάστημα μια όσο το δυνατόν πιο γρήγορη και αντιπροσωπευτική εικόνα για την εξέλιξη των μετρήσεων που εκτελούν οι ομάδες-συνεργεία και να μπορεί να επιβλέπει-συντονίζει καλύτερα την πορεία των εργασιών.

Η διαμόρφωση της εφαρμογής έγινε στα πρότυπα των *Μεγάλων Γεωδαιτικών Ασκήσεων ΙΙ*, με σκοπό να είναι χρήσιμο για τους υπεύθυνους καθηγητές διευκολύνοντας το έργο της παρακολούθησης των ασκήσεων. Όμως ο απώτερος σκοπός δεν είναι να λυθούν απλά κάποιες πρακτικές δυσκολίες των συγκεκριμένων ασκήσεων, αλλά να χαραχθεί μια πορεία

και να οριστούν κάποιες αρχικές δομές και τεχνολογίες που στο μέλλον μπορούν να εξελιχθούν σε μία πλήρως αυτοματοποιημένη διαδικασία, κάτι που θα απαλλάξει τον Τοπογράφο από τη διαχείριση του όγκου των δεδομένων και θα του δώσει τη δυνατότητα να επικεντρωθεί στο ουσιώδες γεωδαιτικό του έργο.

2. Τοπογραφική Προσέγγιση

Σκοπός του κεφαλαίου αυτού είναι να δοθούν οι απαραίτητες διευκρινίσεις για τις λειτουργίες, τον τρόπο εκτέλεσης των τοπογραφικών εργασιών από την εφαρμογή και, γενικώς, τις έννοιες και τις παραδοχές που έγιναν κατά την ανάπτυξή της.

2.1 Προσέγγιση των εργασιών

Η εφαρμογή έχει ως σκοπό την παράλληλη διαχείριση ενός πλήρους κύκλου μετρήσεων για μία συνολική αποτύπωση μιας περιοχής. Αυτό σημαίνει ότι σε κάθε πλήρη άσκηση καταγράφονται λεπτομερώς **οι σπουδαστές** που μετρούν, **οι ομάδες** στις οποίες ανήκουν και η κάθε **μέτρηση** που έκανε μία ομάδα μια συγκεκριμένη ημερομηνία και ώρα στο πεδίο. Συνεπώς, ο στόχος είναι τελικά να έχουν καταγραφεί όλες οι μετρημένες **συντεταγμένες σημείων αναγλύφου** και να έχουν κατηγοριοποιηθεί πλήρως *ανά μέτρηση*, *ομάδα* και *σπουδαστή*.

Τα δεδομένα των μετρήσεων

Το βασικό αποτέλεσμα των επίγειων τοπογραφικών μετρήσεων είναι οι συντεταγμένες των σημείων που μετρήθηκαν (είτε με ταχυμετρία, είτε με επίλυση όδευσης για τις στάσεις οργάνων) και οι οποίες αποτελούνται από τις δύο οριζοντιογραφικές συντεταγμένες (***X***, ***Y***) και ένα υψόμετρο (*H*) που για την εφαρμογή λογίζεται ως ο 3^{ος} άξονας ***Z***. Με αυτή τη μορφή *αποθηκεύει*, *διαχειρίζεται*, *απεικονίζει* και *επεξεργάζεται* η εφαρμογή τα αποτελέσματα των μετρήσεων.

Τα σημεία που καταχωρούνται κάθε φορά πρέπει να είναι σαφώς ορισμένα σε κάποιο σύστημα συντεταγμένων που να καλύπτει τα παραπάνω. Ένα τέτοιο σύστημα μπορεί να είναι το ***ΕΓΣΑ 87*** ή το παλαιότερο ***HATT***, κάποιο άλλο **γεωδαιτικό σύστημα αναφοράς** ή κάποιο αυθαίρετο **σύστημα συντεταγμένων** χωρίς εξάρτηση σε υλοποιημένο *σύστημα αναφοράς*. Είναι προφανές, ασφαλώς, ότι σε κάθε περίπτωση πρέπει *όλες* οι μετρήσεις που αφορούν το ίδιο έργο αποτύπωσης να εκφράζονται στο ίδιο σύστημα αναφοράς και όχι σε διαφορετικά για κάθε μέτρηση.

Ένα σημαντικό ζήτημα είναι να έχει εξασφαλισθεί η ανεξαρτησία της **οριζοντιογραφίας** (*X*, *Y*) από την **υψομετρία** (*Z*) προκειμένου να είναι σωστή η διαχείρισή των σημείων από το πρόγραμμα. Αυτό γίνεται αντιληπτό, επί παραδείγματι, κατά την εκτέλεση της διαδικασίας του *τριγωνισμού*, καθότι η εφαρμογή, ως οφείλει, εκτελεί τη διαδικασία μόνο κατά την *οριζοντιογραφία* και αγνοεί την ύπαρξη των υψομέτρων κατά την παραγωγή των τριγώνων μεταξύ των σημείων. Αν, λοιπόν, δεν ικανοποιείται η απαιτούμενη ανεξαρτησία, τα

αποτελέσματα θα είναι λανθασμένα.

Μία χαρακτηριστική περίπτωση μη-συνιστώμενου συστήματος συντεταγμένων είναι το *WGS 84*, το γεωκεντρικό σύστημα που χρησιμοποιείται κατά κόρον στη μέτρηση συντεταγμένων με *GPS*, προτού αποδοθούν οι μετρήσεις σε κάποιο γεωδαιτικό σύστημα με τη χρήση ενός *Datum* και μιας *προβολής*. Το παραπλανητικό είναι ότι παρότι το *WGS 84* περιγράφει τα σημεία σε ένα *καρτεσιανό χώρο* με 3 συντεταγμένες (*X,Y,Z*), όπως και η εφαρμογή, δεν ορίζει την προαπαιτούμενη ανεξαρτησία μεταξύ τους, δηλαδή δεν υπάρχει κάποια από τις τρεις που περιγράφει σαφώς την υψομετρία. Αυτό, όπως γίνεται αντιληπτό, σημαίνει ότι αν περαστούν τέτοιες συντεταγμένες στο πρόγραμμα θα είναι, μεν, δυνατό να προβληθούν τα σημεία και να γίνει μια στοιχειώδης τρισδιάστατη απεικόνιση, στην πράξη, όμως, *δε θα υπάρχει* πραγματική υπόσταση για το τελικό αποτέλεσμα απεικόνισης.

Είδη αποδεκτών μετρήσεων

Το πρόγραμμα διαχειρίζεται αποτελέσματα ***επίγειων τοπογραφικών μετρήσεων***, κυρίως με την "παραδοσιακή" χρήση ενός *Γεωδαιτικού Σταθμού* (*Total Station*) που αποτυπώνει εξαρτημένος από ένα υλοποιημένο *γεωδαιτικό σύστημα αναφοράς* ή υλοποιεί ένα *ανεξάρτητο σύστημα*. Τέτοιου είδους συντεταγμένες, όμως, μπορούν να προκύψουν όχι μόνο από τέτοιες μεθόδους, αλλά και από ***μετρήσεις με γεωδαιτικό GPS*** ή με ***έμμεσες παρατηρήσεις μέσω φωτογραμμετρικών μεθόδων*** και άλλες γνωστές διεργασίες αποτύπωσης. Σε κάθε περίπτωση, εφόσον τα σημεία που έχουν παραχθεί μετατραπούν σε ένα γεωδαιτικό σύστημα αναφοράς, μπορούν να καταχωρηθούν από το πρόγραμμα με την ίδια επιτυχία. Σε περιπτώσεις, ωστόσο, που γίνεται συνδυασμένη διαχείριση σημείων από διαφορετικά είδη μετρήσεων, είναι πολύ σημαντικό να έχει εξασφαλιστεί ότι τα μεγέθη που περιγράφονται έχουν τη δυνατότητα να συμπεριληφθούν στο ίδιο έργο αποτύπωσης, είτε από την άποψη της *χρήσης ενιαίων μεγεθών* (πχ. ίδια είδη υψομέτρου παντού), είτε από τη σκοπιά της *ακρίβειας* και της *ποιότητας* του τελικού αποτελέσματος.

Για παράδειγμα, η από κοινού χρήση *επίγειων μετρήσεων ταχυμετρίας υψηλής ακρίβειας* με *μετρήσεις GPS μέσης ακρίβειας* είναι πιθανόν να προκαλέσουν προβληματικά συμπεράσματα κατά την επισκόπηση των εργασιών και είναι αδόκιμο να συμπεριλαμβάνονται από κοινού στο συνολικό έργο. Σε τέτοιες περιπτώσεις, ασφαλώς, σημαντικό είναι να έχει οριστεί και ποιες είναι οι **απαιτήσεις ακρίβειας** και τα χαρακτηριστικά του τελικού αποτελέσματος (πχ. **ισοδιάσταση**) καθότι στην περίπτωση χρήση μίας λιγότερο λεπτομερούς αποτύπωσης αναγλύφου είναι δυνατόν να γίνει ανοχή διαφορετικής ακρίβειας μετρήσεων, εφόσον οι *χαμηλότερης ακρίβειας μετρήσεις ικανοποιούν* τις *ελάχιστες απαιτήσεις* της αποτύπωσης.

Υψόμετρα

Δεδομένου ότι σκοπός της εφαρμογής είναι η αποτύπωση του αναγλύφου της μετρημένης περιοχής, πρέπει να δοθεί ιδιαίτερη σημασία στα υψόμετρα των σημείων που καταχωρούνται. Πρώτο και κύριον είναι να έχει εξασφαλισθεί ότι τα σημεία αυτά έχουν υψόμετρο που αντιπροσωπεύει την επιφάνεια της γης στο σημείο εκείνο, είναι δηλαδή **υψομετρικά**. Επομένως, αν ένα σημείο είναι ταχυμετρικό τεχνικού έργου (σπίτι, τοίχο, μάνδρα) και, σε αυτό το σημείο, δεν έχει καθοριστεί στο όργανο το σωστό *Ύψος Στόχου*, τότε το υψόμετρο που θα καταγραφεί είναι λανθασμένο και μη αντιπροσωπευτικό της επιφάνειας. Αυτό, θα παρουσιάσει ανωμαλίες στην απεικόνιση του αναγλύφου της περιοχής και θα οδηγήσει σε *λανθασμένα υψομετρικά στοιχεία* για την περιοχή. Γι αυτό, πρέπει να γίνει ενδελεχής έλεγχος σε κάθε σημείο που καταχωρείται, ώστε να έχει καταγεγραμμένο *πραγματικό υψόμετρο* ή, σε αντίθετη περίπτωση, πρέπει να προσδιορίζεται στο πρόγραμμα ότι δεν πρέπει να ληφθεί υπόψη για την υψομετρία. Η καλύτερη αντιμετώπιση του προβλήματος είναι, κατά τη διαδικασία μέτρησης στο πεδίο, να καταχωρείται σωστά το Ύψος *Στόχου* ακόμα και για *σημεία τεχνικών έργων*, που θεωρητικά δε μας ενδιαφέρει το υψόμετρο, ώστε να μην υπάρχει ιδιαίτερη αλλοίωση της απεικόνισης.

Ένα άλλο σημαντικό ζήτημα που πρέπει να έχει ξεκαθαριστεί είναι το **είδος του υψομέτρου των σημείων**. Στην περίπτωση των επίγειων μετρήσεων, γνωρίζουμε ότι πάντα τα υψόμετρα που μετρώνται είναι **ορθομετρικά** και, ως εκ τούτου, περιγράφουν την απόσταση του σημείου από τη *Μέση Στάθμη της Θάλασσας* και καθιστούν πραγματικά και αξιόπιστα τα αποτελέσματα υψομετρίας που εισάγονται στο πρόγραμμα. Στην περίπτωση, όμως, των μετρήσεων με *GPS* είναι πολύ πιθανόν τα υψόμετρα να είναι **γεωμετρικά** και, κατά συνέπεια, να είναι αδόκιμη η χρήση τους σε συνδυασμό με επίγειες μεθόδους ταχυμετρίας.

Σε τέτοιες περιπτώσεις και εφόσον η ακρίβεια της τελικής αποτύπωσης το επιτρέπει, τα *γεωμετρικά υψόμετρα* θα πρέπει να αναχθούν σε *ορθομετρικά* με χρήση ενός *μοντέλου γεωειδούς* ώστε να καταχωρηθούν στη βάση μετρήσεων και να είναι ορθά διαχειρίσιμα από την εφαρμογή. Σε περίπτωση δε γίνει κάτι τέτοιο, τότε θα πρέπει να μη ληφθούν υπόψη κατά την υψομετρία και απεικόνιση του αναγλύφου (να μη θεωρηθούν, δηλαδή, *υψομετρικά*). Αυτό είναι σημαντικό να γίνει ακόμα και στην περίπτωση που δε γίνεται συνδυασμός επίγειων μετρήσεων και μετρήσεων *GPS*, καθότι ακόμα και αν όλα τα υψόμετρα είναι *γεωμετρικά*, ενδεχομένως να μην υπάρχει πρόβλημα στη συσχέτιση των μετρήσεων μεταξύ τους, όμως τα προϊόντα επεξεργασίας της υψομετρία, όπως οι ισοϋψείς, *δε θα έχουν πραγματική υπόσταση* και θα είναι, πρακτικά, άχρηστα για τον τοπογράφο.

2.2 Μέθοδοι επεξεργασίας

Η εφαρμογή εκτελεί κάποιες βασικές μεθόδους επεξεργασίας που βοηθούν στην *επισκόπηση* και *αξιολόγηση* των απεικονιζόμενων μετρήσεων. Συγκεκριμένα, το πρόγραμμα εκτελεί τη διαδικασία του τριγωνισμού για κάθε μέτρηση ξεχωριστά, παράγοντας τα απαραίτητα τρίγωνα που απεικονίζουν το ανάγλυφο της περιοχής που αποτελείται από τα σημεία της επιφάνειας. Με χρήση αυτού του **τριγωνισμού** και **γραμμικής παρεμβολής**, παράγονται επίσης οι **ισοϋψείς** για κάθε μέτρηση, δίνοντάς μας μία καλύτερη εικόνα της *μορφολογίας* της περιοχής, της *συσχέτισης των μετρήσεων* μεταξύ τους και της *ταύτισης των υψομέτρων* σε γειτονικές ή επικαλυπτόμενες περιοχές μετρήσεων.

Το πρόβλημα του τριγωνισμού

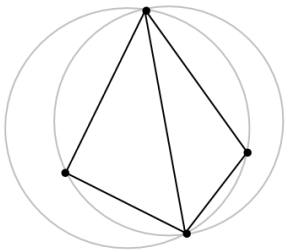
Ένα από τα βασικά προβλήματα που πρέπει να αντιμετωπιστεί από το πρόγραμμα, κυρίως αναφορικά με την απεικόνιση, είναι η δημιουργία της αποτυπωμένης επιφάνειας με βάση τα σημεία που έχουμε μετρήσει. Αυτό είναι τόσο ένα τοπογραφικό πρόβλημα, όσο και ένα ζήτημα παραγωγής των ψηφιακών **3D γραφικών**, καθότι είναι απαραίτητο για την απεικόνιση ενός τρισδιάστατου αντικειμένου να ορίζονται τρίγωνα μεταξύ των σημείων που το αποτελούν. Αυτό ακριβώς, δημιούργησε την ανάγκη για τη χρήση του **τριγωνισμού** στην εφαρμογή.

Ο *τριγωνισμός* είναι η *γεωμετρική πράξη* που αποδίδει για ένα συγκεκριμένο πλήθος σημείων, ένα αντίστοιχο πλήθος τριγώνων που καλύπτουν όλη την επιφάνεια των σημείων αυτών, χωρίς να υπάρχουν τρίγωνα που να επικαλύπτονται και χωρίς να υπάρχουν σημεία που δεν ανήκουν σε τουλάχιστον ένα τρίγωνο. Η γεωμετρική πράξη αυτή είναι πολύ σημαντική και χρήσιμη, όταν εφαρμόζεται σε σημεία αποτύπωσης, διότι ο τριγωνισμός είναι ένα εργαλείο που χρησιμοποιείται για την παραγωγή περαιτέρω τοπογραφικής πληροφορίας, όπως οι ισοϋψείς.

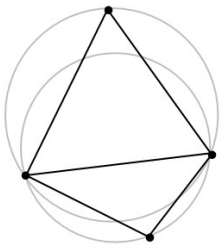
Τριγωνισμός κατά Delaunay

Για την παραγωγή των τριγώνων που αποτελούν το τρισδιάστατο ανάγλυφο χρησιμοποιήθηκε ο *τριγωνισμός κατά **Delaunay***. Ο *τριγωνισμός κατά Delaunay* ορίζεται ως εξής:

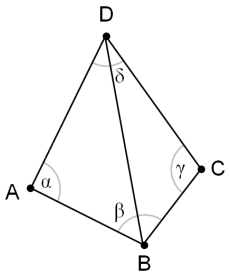
*"Έστω σύνολο **Σ** από **ν** σημεία στο επίπεδο. Τρία σημεία **κ**, **λ**, **μ** που ανήκουν στο **Σ** αποτελούν κορυφές ενός τριγώνου αν ο περιγεγραμμένος κύκλος που ορίζουν δεν περικλείει άλλα σημεία του **Σ**"*



Σχήμα 2.1: Περίπτωση γειτονικών τριγώνων που δεν ικανοποιούν το κριτήριο του Delaunay



Σχήμα 2.2: Η αλλαγή της διχοτόμου πλευράς ικανοποιεί το κριτήριο Delaunay



Σχήμα 2.3: Η εναλλακτική διατύπωση του Delaunay με το άθροισμα γωνιών

Ο παραπάνω ορισμός ορίζει μία σαφή λογική επίλυσης του προβλήματος του τριγωνισμού για τα σημεία αποτύπωσης. Μια επαρκής σχηματική εξήγηση μπορούμε να δούμε και στο Σχήμα 2.1, όπου βλέπουμε ένα παράδειγμα τριγώνων που δεν ικανοποιούν την απαίτηση του *Delaunay*. Συγκεκριμένα, παρατηρούμε ότι και οι δύο περιγεγραμμένοι κύκλοι των 2 τριγώνων, περιέχουν το $4^{ο}$ σημείο που δεν ανήκει στο τρίγωνο. Σε αυτή την περίπτωση, η *εναλλαγή της διχοτόμου πλευράς* (Σχήμα 2.2) εξασφαλίζει ότι, πλέον, το *κριτήριο του Delaunay* ικανοποιείται για τα δύο γειτονικά τρίγωνα.

Έτσι ο συνολικός τριγωνισμός μπορεί να υλοποιηθεί με έναν απλό αλγόριθμο που παράγει αρχικά στοιχειώδη τρίγωνα μεταξύ γειτονικών σημείων και, εν συνεχεία, ελέγχει την ικανοποίηση του παραπάνω κριτηρίου για κάθε ζεύγος γειτονικών τριγώνων. Πρακτικά, για τον έλεγχο δε χρειάζεται η σχεδίαση των περιγεγραμμένων κύκλων, αλλά αρκεί για κάθε ζεύγος τριγώνων να υπολογίσουμε το άθροισμα των δύο μη-τεμνόμενων γωνιών. Στο Σχήμα 2.3 βλέπουμε αυτή την εναλλακτική διατύπωση του *κριτηρίου Delaunay* με το άθροισμα των γωνιών *α* και *γ*. Αν το άθροισμα αυτό είναι **μεγαλύτερο** από **180°** τότε πρέπει να γίνει εναλλαγή της διχοτόμου πλευράς για να παραχθούν τα σωστά τρίγωνα.

Το μεγάλο πλεονέκτημα του *τριγωνισμού κατά Delaunay*, είναι ότι **μεγιστοποιεί** την **ελάχιστη γωνία** των τριγώνων. Οδηγεί, δηλαδή, σε ένα αποτέλεσμα που είναι όσο το δυνατόν καλύτερο οπτικά, για την τρισδιάστατη απεικόνιση του αναγλύφου της περιοχής, ενώ παράλληλα εξασφαλίζει ένα αξιόπιστο υπόβαθρο για την περαιτέρω επεξεργασία της πληροφορίας της αποτύπωσης.

Απόδοση ισοϋψών

*"**Ισοϋψής καμπύλη** είναι ο γεωμετρικός τόπος των σημείων που έχουν το ίδιο υψόμετρο και προκύπτει από την τομή της ισοδυναμικής επιφάνειας, παράλληλης με τη **Μ.Σ.Θ.**, με το έδαφος."*

Η επιτομή της τοπογραφικής αποτύπωσης, ίσως, είναι η απόδοση των ισοϋψών καμπυλών για την περιοχή που μετρήθηκε. Έτσι, θεωρήθηκε απαραίτητη η ενσωμάτωση της δυνατότητας αυτόματης σχεδίασης των ισοϋψών καμπυλών από την εφαρμογή για το

ανάγλυφο της περιοχής που έχει αποτυπωθεί.

Το πρόβλημα του υπολογισμού των *ισοϋψών καμπυλών* είναι αρκετά σύνθετο και αφορά πεδίο τόσο της *τοπογραφικής επιστήμης*, όσο και της πρακτικής εφαρμογής του *τοπογραφικού σχεδίου*. Προσεγγίζοντας, όμως, το πρόβλημα από προγραμματιστική σκοπιά, είναι σαφές ότι έπρεπε να βρεθεί ένα σαφής τρόπος ορισμού των σημείων που ορίζουν τις ισοϋψείς, ομοίως με τη σαφήνεια με την οποία ο *τριγωνισμός Delaunay* καθορίζει τη διαδικασία σχεδιασμού των τριγώνων.

Για την επίλυση του προβλήματος, προσεγγίσαμε το πρόβλημα από τη *σχεδιαστική σκοπιά* του, καθορίζοντας συγκεκριμένους κανόνες που εξασφαλίζουν την αρτιότητα του αποτελέσματος. Έτσι, αποφασίσαμε ότι θα ακολουθηθεί η συνήθης τακτική της *γραμμικής παρεμβολής*[1] για τη δημιουργία των σημείων "ακέραιων υψομέτρων" (ανάλογα την ***ισοδιάσταση***) τα οποία όταν ενωθούν μεταξύ τους (συνεχόμενα) θα αποδώσουν τις *υψομετρικές καμπύλες*. Ο ορισμός αυτός, όμως, δεν είναι αρκετά συγκεκριμένος για τις απαιτήσεις της εφαρμογής.

Έτσι, το πρόβλημα προσεγγίστηκε ως εξής: σε κάθε τρίγωνο που έχει προκύψει από την τριγωνισμό θα υπολογίζουμε ξεχωριστά τα σημεία των ισοϋψών με *γραμμική παρεμβολή* και θα τα ενώνουμε μεταξύ τους. Το τελικό αποτέλεσμα είναι ότι θα προκύψουν οι τελικές συνεχείς ισοϋψείς καμπύλες.

Πιο συγκεκριμένα, τα βήματα υπολογισμού της διαδικασίας είναι:

- Για **κάθε τρίγωνο** υπολογίζουμε το **ελάχιστο** και το **μέγιστο σημείο υψομέτρου**.

- Για **κάθε ισοϋψή** που περιέχεται μεταξύ των σημείων αυτών, υπάρχει κομμάτι της ισοϋψούς που τέμνει το τρίγωνο αυτό. Η μία πλευρά που *τέμνει*, μάλιστα, θα είναι πάντα αυτή μεταξύ του **χαμηλότερου** και του **υψηλότερου σημείου**, οπότε το ένα σημείο της ευθείας προκύπτει από *γραμμική παρεμβολή* μεταξύ των δύο.

- Απομένει το 2ο σημείο της ισοϋψούς καμπύλης (ευθείας, στην πράξη) που τέμνει το τρίγωνο αυτό. Το σημείο αυτό θα ανήκει σε μία από τις δύο εναπομείνασες πλευρές του τριγώνου: είτε αυτή του **χαμηλού σημείου** με το **μεσαίο**, είτε αυτή του **μεσαίου σημείου** με το **υψηλότερο**. Αν το **μεσαίο σημείο** έχει **μεγαλύτερο υψόμετρο** από την *ισοϋψή*, τότε η *ισοϋψής* τέμνει την πρώτη από τις 2 αυτές πλευρές και άρα το 2ο σημείο της υπολογίζεται από τη *γραμμική παρεμβολή* του **χαμηλού** και του **μεσαίου σημείου**. Σε αντίθετη περίπτωση, υπολογίζεται από τη *γραμμική παρεμβολή* του **μεσαίου** και του **υψηλού σημείου** του τριγώνου.

1 *Γραμμική παρεμβολή: είναι η γεωμετρική μέθοδος προσδιορισμού σημείου μίας ευθείας.*

Με αυτό τον τρόπο, σε κάθε τρίγωνο και για κάθε ισοϋψή προκύπτουν 2 σημεία που ενώνονται και συνολικά, μαζί με τα γειτονικά τρίγωνα, παράγουν τις τελικές ισοϋψείς καμπύλες για το ανάγλυφο αυτό.

3. Προγραμματιστική Προσέγγιση

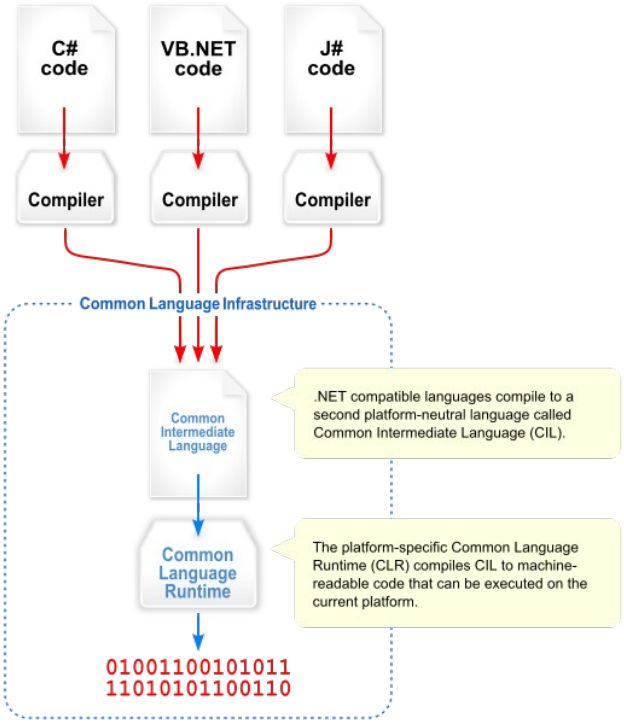
Σε αυτό το κεφάλαιο περιγράφονται οι τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξη του προγράμματος, οι δομές δεδομένων και η σχηματική τους μορφή, προκειμένου να γίνει κατανοητό το πως δομήθηκε η εφαρμογή.

3.1 Τεχνολογίες που χρησιμοποιήθηκαν

Η πλατφόρμα .NET

Το **Microsoft .NET Framework** είναι ένα *Πλαίσιο Λογισμικού* (*Software Framework*) για το λειτουργικό σύστημα *Microsoft Windows*®, που περιλαμβάνει ένα μεγάλο αριθμό ***βιβλιοθηκών*** με προγραμματιστικές λύσεις, για τα συνηθισμένα προβλήματα που αντιμετωπίζει ένας προγραμματιστής, και μία ***εικονική μηχανή*** (*virtual machine*) που διαχειρίζεται την εκτέλεση των εφαρμογών που έχουν γραφτεί ειδικά για το συγκεκριμένο πλαίσιο.

Η *.NET* έχει ως σκοπό την απλοποίηση της ανάπτυξης εφαρμογών «κρύβοντας» τις τεχνικές λεπτομέρειες υλοποίησης πολλών λειτουργιών, όπως διαχείριση μνήμης, επικοινωνία μέσω δικτύου, είσοδο/έξοδο από συσκευές και αφήνοντας το προγραμματιστή ελεύθερο να επικεντρωθεί στην «λογική» του προγράμματος.



Σχήμα 3.1: Σχηματική απεικόνιση της δομής του ενιαίου περιβάλλοντος γλωσσών

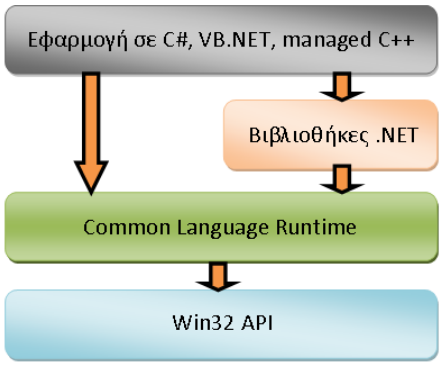
Χαρακτηρίζεται ως *managed πλατφόρμα* με την έννοια ότι δημιουργεί ένα ελεγχόμενο και ασφαλές περιβάλλον μέσα στο οποίο μπορεί να τρέξει μια εφαρμογή. Η ασφάλεια έγκειται, για παράδειγμα, στον έλεγχο της δέσμευσης και προσπέλασης της μνήμης (δεν υπάρχουν *pointers*, δεν μπορείς να προσπελάσεις μια θέση μνήμης εκτός πίνακα), στο τύπο των μεταβλητών και δεδομένων (δεν μπορείς να θέσεις μια *float* τιμή σε μια *ακέραια* μεταβλητή) ή στην αυτόματη υλοποίηση δικλείδων ασφαλείας.

Το *.NET* υποστηρίζει πληθώρα *γλωσσών προγραμματισμού* οι οποίες είναι ειδικά

σχεδιασμένες για αυτό, όπως **C#**, **Visual Basic .NET**, **J++** και **managed C++**. Στην πραγματικότητα, το *.NET* καταλαβαίνει μόνο μια γλώσσα προγραμματισμού την **Microsoft Intermediate Language** (**MSIL**). Συνεπώς, οποιαδήποτε γλώσσα προγραμματισμού μπορεί να μεταγλωττιστεί σε *MSIL* μπορεί να τρέξει στην πλατφόρμα *.NET*. Ο χρήστης μπορεί ακόμα να γράψει απευθείας ένα πρόγραμμα σε *MSIL* στο *Notepad*, να το *μεταγλωττίσει* (*compile*) και να το εκτελέσει στη *.NET*!

Τρόπος λειτουργίας της .NET

Η *πλατφόρμα .NET* βασίζεται σε ένα πλαίσιο που καλύπτει τα *χαμηλού επιπέδου στοιχεία* του *λειτουργικού συστήματος* με πιο προσιτές εντολές για το χρήστη. Καρδιά του συστήματος αποτελεί το λεγόμενο **Common Language Runtime** (**CLR**). Η οντότητα αυτή είναι το *managed περιβάλλον* μέσα στο οποίο τρέχουν οι *εφαρμογές .NET*. Κατά μια έννοια κρύβει το δύσχρηστο *Win32 API* που χρησιμοποιείται συχνά για προγραμματισμό εφαρμογών *Windows* και παρουσιάζει στον προγραμματιστή ένα απλούστερο και περισσότερο εύχρηστο περιβάλλον εντολών (βλ. Σχήμα 3.2).



Σχήμα 3.2: Ο τρόπος με τον οποίο η .NET αποκρύπτει τις δύσχρηστες χαμηλού επιπέδου βιβλιοθήκες με πιο απλές και εύχρηστες

Επίσης το *.NET* παρέχει και μια πληθώρα *βιβλιοθηκών* με έτοιμες λειτουργίες που μπορεί να χρησιμοποιήσει ο χρήστης για την ανάπτυξη των εφαρμογών.

Το *.NET*, όπως και η *Java* χαρακτηρίζονται από την λεγόμενη "**επί τόπου**" (**Just-in-time**) **μεταγλώττιση**. Ο κώδικας, στη γλώσσα προγραμματισμού που χρησιμοποιεί ο χρήστης, μεταγλωττίζεται αρχικά σε *MSIL* η οποία αποθηκεύεται σε ένα εκτελέσιμο *.exe* αρχείο ή σε μια βιβλιοθήκη *.dll*. Όταν ο χρήστης τρέξει το πρόγραμμα που ανέπτυξε, το *CLR* διαβάζει το *MSIL κώδικα* του αρχείο και *"επί τόπου"* το μεταγλωττίζει σε *εγγενή κώδικα (native code) Windows* έτοιμο προς εκτέλεση, και στην συνέχεια τον εκτελεί. Αυτό το επιπλέον βήμα πριν την εκτέλεση του κώδικα διαφοροποιεί μια *managed εφαρμογή σε .NET* από μια *unmanaged (κλασική) σε C++*, για παράδειγμα. Το πρόγραμμα σε *C++* είναι ήδη μεταγλωττισμένο σε *εγγενή Windows κώδικα* και τρέχει απευθείας.

Ένα ακόμα χαρακτηριστικό του *CLR* είναι η *αυτοματοποιημένη διαχείριση μνήμης*. Σε κλασσικές γλώσσες προγραμματισμού (*unmanaged*) όπως η *C++*, όταν ο χρήστης δεσμεύσει μια ποσότητα μνήμης για να αποθηκεύσει ένα αντικείμενο πρέπει να είναι πολύ προσεκτικός στο να την αποδεσμεύσει, να την επιστρέψει στο σύστημα δηλαδή, όταν δεν την χρειάζεται. Αν το αγνοήσει αυτό συστηματικά, τότε θα δημιουργηθεί η επονομαζόμενη ***διαρροή μνήμης*** (***memory leak***), δηλαδή η διαθέσιμη μνήμη του συστήματος θα ελαττώνεται διαρκώς και σε κάποιο σημείο τα *Windows* θα στερέψουν από ελεύθερη μνήμη.

Αντιθέτως το *CLR* προσφέρει ένα *μηχανισμό συλλογής "σκουπιδιών"* (*Garbage Collection*). Ο χρήστης μπορεί να ζητήσει όση μνήμη χρειάζεται από το σύστημα και να μην ασχοληθεί με την απελευθέρωση της. Ο *Garbage Collector* υλοποιεί μηχανισμούς που του επιτρέπουν να «καταλάβει» πότε μια δεσμευμένη ποσότητα μνήμης δεν χρησιμοποιείται πλέον και αυτόματα την απελευθερώνει για μετέπειτα χρήση.

Πλεονεκτήματα της πλατφόρμας .NET

Το *.NET* έχει πολλά πλεονεκτήματα για την ανάπτυξη εφαρμογών:

- ✔ Είναι εγγενώς **αντικειμενοστραφής** πλατφόρμα.
- ✔ Είναι **ανεξάρτητο** από γλώσσα προγραμματισμού. Σε μια εφαρμογή ένας προγραμματιστής μπορεί να γράφει κώδικα σε *C#*, άλλος σε *VB .NET* και άλλος σε *managed C++* και τα τμήματα που αναπτύσσει ο καθένας να συνεργάζονται μεταξύ τους χωρίς προβλήματα.
- ✔ Η χρήση ***assemblies*** κάνει πολύ εύκολη την επαναχρησιμοποίηση κώδικα.
- ✔ Παρέχει πολύ εύκολη εγκατάσταση. Αρκεί να αντιγράψουμε το κατάλογο της εφαρμογής σε ένα άλλο υπολογιστή και αυτή θα τρέξει άμεσα. **Δεν υπάρχει installation**, δεν πειράζει το *registry*.
- ✔ Παρέχει πληθώρα **έτοιμων λειτουργιών** που κάνουν την ανάπτυξη κώδικα πολύ εύκολη.
- ✔ **Αυτοματοποιημένη διαχείριση μνήμης**, ο προγραμματιστής δεν χρειάζεται να ασχοληθεί με αποδέσμευση μνήμης.

Μειονεκτήματα .NET

Το .NET έχει 2 μειονεκτήματα που αφορούν ειδικά την ανάπτυξη εφαρμογών με γραφικά και πολύπλοκες δομές:

- ✗ **Αυτοματοποιημένη διαχείριση μνήμης**, ο προγραμματιστής (θεωρητικά) <u>δεν</u> έχει τη δυνατότητα διαχείρισης της μνήμης με το βέλτιστο τρόπο.
- ✗ Το *CLR* εισάγει μια (μικρή ίσως) **καθυστέρηση** στην εκτέλεση της εφαρμογής.

Το πρώτο, αν και είναι πολύ χρήσιμο χαρακτηριστικό για γενικό προγραμματισμό, σε μία εφαρμογή που διαχειρίζεται κρίσιμα στοιχεία όπως τρισδιάστατα γραφικά μπορεί να δημιουργήσει προβλήματα.

Το δεύτερο αφορά την "επί τόπου" μεταγλώττιση που υποστηρίζει το *CLR*. Από τη μία εισάγει μια καθυστέρηση στην εκκίνηση της εφαρμογής, μιας και πρέπει να μεταγλωττιστεί ο κώδικας, από την άλλη ο *μεταγλωττιστής* ο ίδιος δεν είναι βέλτιστος με την έννοια ότι δεν παράγει το καλύτερο δυνατό *εγγενή* (*native*) *κώδικα* για τα *Windows*. Αυτό, αφαιρεί τις ουσιώδεις δυνατότητες βελτιστοποίησης που διαθέτουν οι χαμηλού επιπέδου γλώσσες προγραμματισμού.

Πρακτική χρήση της .NET

Η *πλατφόρμα .NET* είναι ένα περιβάλλον λειτουργίας που πραγματοποιεί *"επί τόπου" μεταγλώττιση* κατά την εκτέλεση μιας εφαρμογής που τη χρησιμοποιεί. Έτσι, για να μπορέσει κάποιος να "τρέξει" ένα *πρόγραμμα .NET* θα πρέπει να έχει εγκαταστήσει το αντίστοιχο *περιβάλλον τελικού χρήστη (End-User Runtime Environment)*. Στην περίπτωσή μας, χρησιμοποιήθηκε το **.NET Framework 2.0** και άρα σε κάθε υπολογιστή που σκοπεύει κάποιος να εκτελέσει την εφαρμογή θα πρέπει να έχει εγκαταστήσει το *.NET Framework 2.0 Redistributable Package*[2] που περιέχει όλα τα απαραίτητα στοιχεία (περιβάλλον και βιβλιοθήκες) που χρειάζονται.

Το DirectX API

Το **DirectX** είναι ένα σύνολο *Διασυνδέσεων Προγραμματισμού Εφαρμογών* (*Application Programming Interface - API*) που έχει ως στόχο τη διαχείριση εργασιών πολυμέσων από προγραμματιστική σκοπιά. Αυτό σημαίνει ότι το *DirectX* προσφέρει ένα πλήρες προγραμματιστικό περιβάλλον που επιτρέπει την αναπαραγωγή/εκτέλεση/επεξεργασία πολυμέσων όπως εικόνα, ήχος και 3D γραφικά μέσω ενός πλήρους σετ βιβλιοθηκών που επιτρέπουν την εκτέλεση των παραπάνω λειτουργιών πιο εύκολα από ότι χρησιμοποιώντας διαφορετικές και εξειδικευμένες τεχνολογίες για κάθε αντικείμενο ξεχωριστά.

Η χρησιμότητα του *DirectX* είναι ότι επικαλύπτει τις τεχνικές διαφορές, μεταξύ των διαφορετικών υπολογιστικών συστημάτων, σε ένα ενιαίο κέλυφος που εγγυάται στον προγραμματιστή ότι αφενός μεν θα παράγει κώδικα που θα λειτουργεί σε κάθε σύστημα και, αφετέρου, δε θα χρειαστεί να διεισδύσει σε χαμηλού επιπέδου προγραμματιστικά προβλήματα που να αφορούν συγκεκριμένο *υλικό υπολογιστών* (***hardware***) ή ιδιαιτερότητες των εκάστοτε *οδηγών υλικού* (***drivers***).

2 http://tinyurl.com/758p8

Συστατικά του DirectX

Το *DirectX* δεν είναι κάτι παραπάνω από ένα πλήθος βιβλιοθηκών για τη διαχείριση πολυμέσων. Από αυτές, μόνο 2 χρησιμοποιούνται από την εφαρμογή αυτή:

1. **DirectX Graphics:** Είναι η καρδιά των γραφικών και αποτελείται από τις υποβιβλιοθήκες (*Direct3D*, *DirectDraw*, *DXGI*, *Direct2D*, *DirectWrite*). Χρησιμοποιείται κατά κόρον από την εφαρμογή για τη σχεδίαση της 3Δ απεικόνισης της επιφάνειας των μετρήσεων

2. **DirectInput:** Βιβλιοθήκη που αναλαμβάνει τη διαχείριση των συσκευών εισόδου (πληκτρολόγιο, ποντίκι, χειριστήρια κλπ.). Χρησιμοποιείται στην εφαρμογή για τον έλεγχο των εντολών μέσω του ποντικιού (*Pan*, *Zoom*, *Rotate*).

Ενδεικτικά, αναφέρεται ότι υπάρχουν πολλές ακόμα βιβλιοθήκες της *DirectX* που επιτρέπουν την εκτέλεση υπολογισμών στην κάρτα γραφικών (DirectCompute), την επικοινωνία μέσω δικτύου/διαδικτύου (*DirectPlay*), την αναπαραγωγή/διαχείριση ήχων (*DirectSound*) και άλλα που δεν αφορούν το πεδίο δράσης της εφαρμογής.

Βασική λειτουργία των γραφικών

Σε αυτό το σημείο είναι χρήσιμο να γίνει μια παρουσίαση του τρόπου λειτουργίας των 3D γραφικών σε ένα σύστημα υπολογιστών.

Ένα τρισδιάστατο αντικείμενο απαιτεί δύο βασικά συστατικά για να μπορέσει να απεικονιστεί: τα σημεία (**Vertices**) και τους δείκτες (**Indices**). Αυτά τα δομικά στοιχεία "συντίθενται" μεταξύ τους για να παράγουν το τελικό αποτέλεσμα της επιφάνειας.

Ένα **vertex** περιγράφει τη θέση του σε ένα καρτεσιανό σύστημα με τις συντεταγμένες του (*X,Y,Z*). Εκτός αυτού, όμως, τα *vertices* έχουν και άλλες ιδιότητες όπως χρώμα, συντεταγμένες υφής (*U,V*) για την περίπτωση που επισυνάπτεται σε αυτά κάποια εικόνα (*texture*) και ένα μοναδιαίο διάνυσμα (*normal*) που ορίζει την αντανάκλαση του φωτός πάνω στο σημείο.

Τα *vertices*, αφού δημιουργηθούν και καταχωρηθούν σε ένα πίνακα, πρέπει να δηλωθούν σε ένα **vertex buffer** που θα αναλάβει τη "διεκπεραίωσή" τους από την *κάρτα γραφικών* (*GPU*) σε συνδυασμό με τα *indices*.

Ένα **index** δεν είναι παρά ένα αριθμός, που συμβολίζει τη θέση του σημείου που θα σχεδιαστεί κάθε φορά, από τον πίνακα των *vertices* που δηλώθηκε στο *vertex buffer*. Τρία *indices* στη σειρά ορίζουν ένα τρίγωνο που σχεδιάζεται και, έτσι, από ένα πλήθος συνεχόμενων σχεδιασμένων τριγωνικών επιφανειών εμφανίζεται η συνολική μορφή του τρισδιάστατου αντικειμένου.

Όπως και τα *vertices*, τα *indices* καταχωρούνται σε ένα πίνακα και, εν συνεχεία, δηλώνονται σε ένα **index buffer**. Το τελικό αποτέλεσμα απεικόνισης προκύπτει όταν μέσω του *DirectX* καλούνται οι σχετικές συναρτήσεις που δηλώνουν το *vertex buffer* και το *index buffer* και το *DirectX* αναλαμβάνει να μεταφέρει τα δεδομένα στην κάρτα γραφικών και να φροντίσει για την προβολή τους, ανά πάσα στιγμή.

Ασφαλώς, για τη συνολική απεικόνιση απαιτούνται πολλά περισσότερα από την απλή δήλωση των παραπάνω. Σημαντική είναι η διαχείριση της προβολής, των διαφόρων παραμέτρων απεικόνισης, του φωτισμού, της διαχείρισης των υφών (*textures*) και άλλα θέματα που δεν αφορούν τους σκοπούς αυτής της εργασίας. Η επιστήμη της προβολής των 3D γραφικών, ωστόσο, παρουσιάζει πολλές ομοιότητες με την τοπογραφία και ζητήματα διαχείρισης συντεταγμένων, προβολής στο επίπεδο κλπ. είναι κοινές μεταξύ των δύο πεδίων.

Πρακτική χρήση του DirectX

Όπως στην περίπτωση της *.NET*, έτσι και για κάθε εφαρμογή που κάνει χρήση της τεχνολογίας *DirectX* θα πρέπει να προϋπάρχουν οι αντίστοιχες βιβλιοθήκες εγκατεστημένες στον υπολογιστή που θα εκτελεστεί η εφαρμογή. Για τη συγκεκριμένη εφαρμογή, το πακέτο που χρειάζεται να υπάρχει είναι το πακέτο *DirectX End-User Runtimes (August 2009)*[3] που εξασφαλίζει την ύπαρξη του κατάλληλου περιβάλλοντος για την εφαρμογή, σε ότι αφορά τη χρήση του *DirectX API*.

3.2 Βιβλιοθήκες

Εκτός από τις παραπάνω ανεπτυγμένες και εκτενείς τεχνολογίες, χρησιμοποιήθηκαν και αναπτύχθηκαν βιβλιοθήκες που εκτελούν επιμέρους ενέργειες για την εφαρμογή. Οι βιβλιοθήκες αυτές περιέχονται υπό τη μορφή αρχείων *.dll* και αρκεί να βρίσκονται στον ίδιο φάκελο με την εφαρμογή για να είναι προσβάσιμες από αυτήν.

Τριγωνισμός

Το ζήτημα του τριγωνισμού καλύφθηκε εκτενώς όσον αφορά τη θεωρητική πλευρά του στο κεφάλαιο *"Τοπογραφική Προσέγγιση"*. Η χρήση του *τριγωνισμού Delaunay* απαιτεί την υλοποίηση του αλγορίθμου λειτουργίας του από μία βιβλιοθήκη που θα δέχεται ως είσοδο τις συντεταγμένες των σημείων και θα εξάγει το σύνολο των τριγώνων που τα συνδέουν.

Μία πολύ πρακτική και εύχρηστη υλοποίηση του τριγωνισμού Delaunay σε βιβλιοθήκη *.NET 2.0* έχει γίνει από τον *Morten Nielsen*. Συγκεκριμένα, αφορά τριγωνισμό στο δισδιάστατο χώρο, κάτι που ικανοποιεί απόλυτα τις ανάγκες της συγκεκριμένης εφαρμογής, αφού όπως έχει ήδη διευκρινισθεί, ο τριγωνισμός αφορά μόνο τις οριζοντιογραφικές

3 http://tinyurl.com/lbq2bg

συντεταγμένες (*X,Y*) και αγνοεί την ύπαρξη του υψομέτρου. Δεδομένου ότι η βιβλιοθήκη είναι ελεύθερη προς χρήση και επεξεργασία και ικανοποιεί τα παραπάνω, θεωρήθηκε μια επαρκής λύση στο πρόβλημα αυτό και έτσι δε χρειάστηκε να υλοποιηθεί ο αλγόριθμος του *Delaunay* από την αρχή.

Στη βιβλιοθήκη ορίζονται δύο είδη αντικειμένων που χρησιμεύουν στην εκτέλεση της πράξης, τα σημεία (**Point**) και τα τρίγωνα (**Triangle**). Ένα αντικείμενο *Point* αποτελείται από τις συντεταγμένες *X*, *Y*, ενώ ένα *Triangle* αποτελείται από *3 ακέραιους αριθμούς* που συμβολίζουν το κάθε σημείο που είναι γωνία του τριγώνου (ο αριθμός είναι η θέση του σημείου στον πίνακα-λίστα των σημείων), όμοια με τα *indices* των 3D γραφικών.

Η χρήση της βιβλιοθήκης είναι σχετικά απλή. Απαιτείται η δημιουργία μίας λίστας σημείων (**List<Point>**) όπου καταχωρούνται οι οριζοντιογραφικές συντεταγμένες των σημείων του τριγωνισμού. Εν συνεχεία, αυτή η λίστα εισάγεται ως όρισμα στη συνάρτηση **Triangulate()** η οποία επιστρέφει ως αποτέλεσμα τη λίστα των τριγώνων που ικανοποιούν το *κριτήριο του Delaunay*.

Από όλα τα παραπάνω, είναι σαφής η συσχέτιση μεταξύ του αντικειμένου *Point*, της βιβλιοθήκης *Triangulation*, και *Vertex*, της βιβλιοθήκης *Direct3D*, όπως επίσης και του αντικειμένου *Triangle* με τα *Indices* των γραφικών (κάθε *Triangle* περιέχει *3 Indices*). Έτσι, η βιβλιοθήκη *Triangulation* χρησιμοποιείται για το σκοπό της αυτόματης παραγωγής των *Indices* με βάση τη λίστα των *Vertices* που έχουν δηλωθεί για κάθε μέτρηση προκειμένου να γίνει η απεικόνιση των γραφικών.

DXF Library

Ένα κοινό πρόβλημα που αντιμετωπίζουν τέτοιου είδους εφαρμογές διαχείρισης δεδομένων σε τρισδιάστατο χώρο είναι η εξαγωγή των δεδομένων σε μία μορφή που να εξασφαλίζει την πρακτικότητα, την καθολικότητα (υποστήριξη από όλα τα *λογισμικά CAD*) και να ικανοποιεί τα παγκόσμια πρότυπα. Με γεγονός ότι το **DXF** είναι η *πρότυπη μορφή διακίνησης γραφικών δεδομένων*, όπως και τεχνικών σχεδίων (τοπογραφικών, αρχιτεκτονικών κλπ.) επιλέχθηκε αυτό το είδους αρχείου για να εξυπηρετεί τους σκοπούς μεταφοράς των δεδομένων της εφαρμογής σε άλλα λογισμικά.

Δεδομένης της εκτεταμένης χρήσης του *DXF* σε προγράμματα παντός χρήσης γραφικών ή *CAD* υπάρχουν αρκετές υλοποιήσεις της σε βιβλιοθήκες, αν και όχι τόσο πολύ σε *περιβάλλον .NET*. Θέσαμε ως απαραίτητο κριτήριο, για την επιλογή της κατάλληλης, να είναι εύχρηστη και, κυρίως, ανοιχτή σε επεμβάσεις και σε λογική για να μπορεί να εξελιχθεί παράλληλα με το προγραμματιστικό περιβάλλον που κατασκευάστηκε ώστε να μπορεί να ικανοποιήσει, εκτός από τις υπάρχουσες, και ενδεχόμενες αυξημένες μελλοντικές ανάγκες.

Επιλέχθηκε μία **ανοικτού κώδικα** (**open source**) υλοποίηση από τον *Cosmin Radoi*, η οποία τελεί υπό την ***άδεια χρήσης MIT*** [4] που επιτρέπει τη χρησιμοποίηση και μετ-επεξεργασία χωρίς περιορισμούς, εφόσον αναφέρονται επαρκώς οι απαραίτητες διευκρινίσεις για το δημιουργό.

Ένα από τα βασικά πλεονεκτήματα αυτής της βιβλιοθήκης είναι η εξαιρετική ευελιξία που παρέχει για την επέκταση των υπαρχόντων αντικειμένων ή και την προσθήκη καινούργιων. Έτσι, παρότι αρχικά δεν υποστηρίζονταν όλα όσα χρειάζονταν από την εφαρμογή (πχ. τα σημεία και οι γραμμές είναι μόνο 2 συντεταγμένες και όχι υψόμετρο), επεκτείναμε τη χρήση της ώστε να υποστηρίζεται πλήρως και η 3η διάσταση που είναι σημαντική για τις μετρήσεις. Ως οφείλεται σε τέτοιες περιπτώσεις, οι βελτιώσεις που εφαρμόσαμε στη βιβλιοθήκη κατατέθηκαν στο επίσημο *αποθετήριο* της (*git*) ώστε να είναι διαθέσιμες για την κοινότητα του ελεύθερου λογισμικού.

3.3 Δομή της εφαρμογής

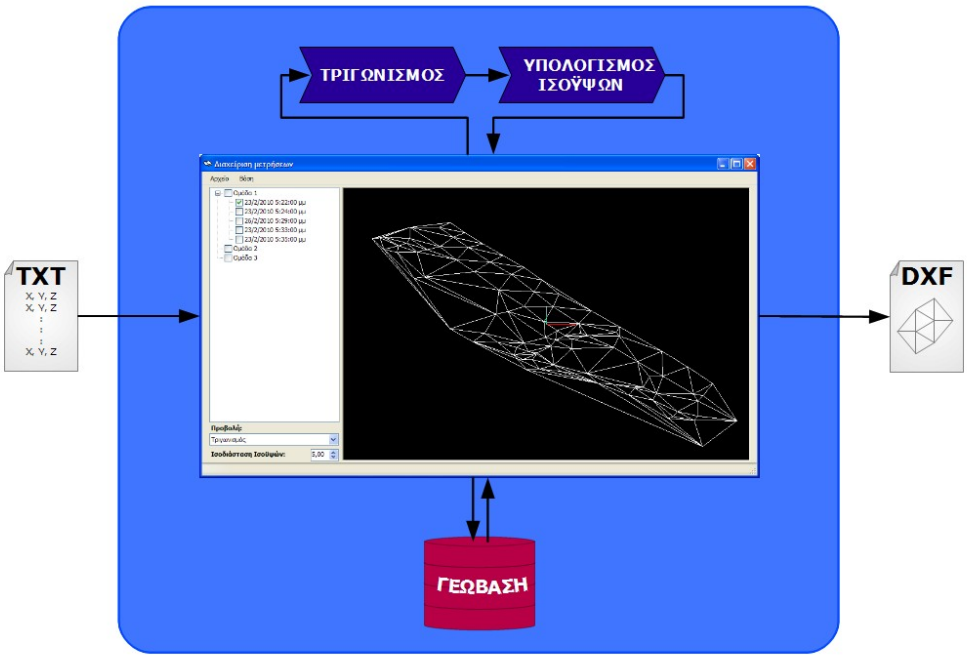
Ροή λειτουργίας (workflow)

Η εφαρμογή έχει ένα συγκεκριμένο ρόλο στη διαδικασία διαχείρισης των μετρήσεων ενός μεγάλου τοπογραφικού έργου αποτύπωσης. Σκοπός είναι να *"παραλάβει"* ως **δεδομένα εισόδου** (**input**) τα *σημεία* σε συντεταγμένες κάποιου γεωδαιτικού συστήματος, να τα *καταχωρήσει* σε μία **κεντρική γεωβάση** μαζί με τα ποιοτικά της χαρακτηριστικά (ομάδα μέτρησης, σπουδαστές κλπ.), να *εκτελέσει* τις **πράξεις επεξεργασίας** (τριγωνισμός, υπολογισμός ισοϋψούς) και να *εξάγει* το πρωτογενές υλικό οργανωμένο, παράλληλα με τα παραγόμενα δεδομένα ως **έξοδο** (**output**) σε μορφή ικανή να χρησιμοποιηθεί από άλλα προγράμματα (*CAD*, *GIS*) για πιο λεπτομερή επεξεργασία.

Αυτή η λογική απεικονίζεται σχεδιαστικά στο Σχήμα 3.3 με ένα γενικό τρόπο. Η χρωματισμένη, με γαλάζιο χρώμα, περιοχή συμβολίζει το *πεδίο δράσης* της εφαρμογής. Στην ουσία, οριοθετεί εκείνες τις ενέργειες που εκτελούνται από το περιβάλλον που δημιουργήθηκε για να εκτελούνται οι απαραίτητες ενέργειες διαχείρισης και επεξεργασίας.

Το πρόγραμμα βασίζεται σε μία **γεωβάση** στην οποία διατηρεί τα στοιχεία που περιγράφουν τις ασκήσεις: οι **σπουδαστές**, οι **ομάδες** και οι **μετρήσεις** (που αποτελούνται από **σημεία**). Σε αυτή τη *γεωβάση* εισάγονται ως δεδομένα, μέσω της εφαρμογής, *συντεταγμένες* από ένα απλό *αρχείο κειμένου* (*txt*) που περιέχει τα σημεία για μία μέτρηση. Μέσα από το πρόγραμμα, αναθέτουμε αυτή τη μέτρηση σε μία συγκεκριμένη *ομάδα σπουδαστών* και καθορίζουμε την ημερομηνία και ώρα εκτέλεσης της.

4 http://www.opensource.org/licenses/mit-license.php



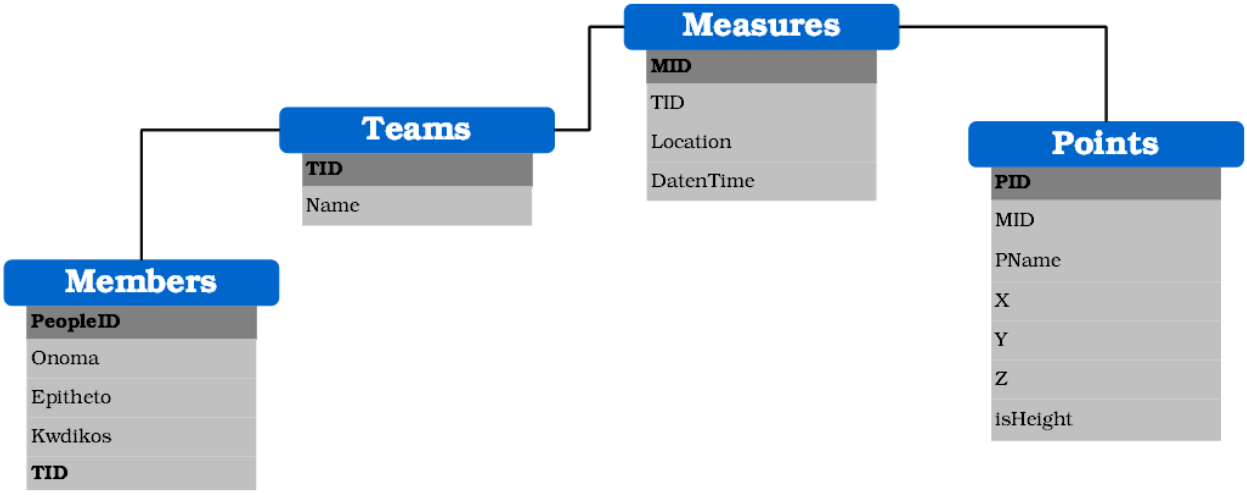
Σχήμα 3.3: Γενική ιδέα τρόπου λειτουργίας της εφαρμογής.

Έχοντας καταχωρημένα όλα τα δεδομένα της εφαρμογής στη *γεωβάση*, μας δίνεται η δυνατότητα να εκτελέσουμε τις βασικές λειτουργίες: **προβολή των μετρήσεων**, **επεξεργασία** τους και **επισκόπηση** της κατάστασης. Μέσα από αυτή τη διαδικασία, μπορούμε να διαχειριστούμε τα δεδομένα με καλύτερο τρόπο: να **επιλέξουμε** όσα θέλουμε να κρατήσουμε, να **διαγράψουμε** όσα θεωρούμε λανθασμένα, άχρηστα ή ανεπαρκή και, εν τέλει, να **εξάγουμε** το τελικό συνδυαστικό αποτέλεσμα.

Η μορφή εξόδου της πληροφορίας είναι ένα **αρχείο σχεδίου DXF**, που υποστηρίζεται από τα περισσότερα σχεδιαστικά προγράμματα. Αυτό το αρχείο σκοπεύει να είναι το βασικό υπόβαθρο για την τελική σχεδίαση του τοπογραφικού διαγράμματος, αφού είναι απαλλαγμένο από τις προβληματικές μετρήσεις, περιέχει επιπλέον στοιχεία που διευκολύνουν το έργο του σχεδιαστή (τριγωνισμός, ισοϋψείς) και απομένει η υλοποίηση της τελικής σχεδίασης τεχνικών έργων και αντικειμένων για την παραγωγή του τελικού χάρτη.

Δομή δεδομένων στη βάση

Τα δεδομένα που διαχειρίζεται η εφαρμογή είναι οργανωμένα στη βάση με τέτοιο τρόπο που κάθε πίνακας να ορίζει μία οντότητα. Οι πίνακες που περιέχονται στη γεωβάση είναι οι σπουδαστές (**Members**), οι ομάδες (**Teams**), οι μετρήσεις (**Measures**) και τα σημεία (**Points**). Όλα αυτά συνδέονται μεταξύ τους με τον τρόπο που δείχνει το διάγραμμα συσχέτισης των πινάκων στο Σχήμα 3.4.



Σχήμα 3.4: Διάγραμμα συσχέτισης των πινάκων της γεωβάσης

Ο πίνακας **Members** περιέχει τους σπουδαστές που συμμετέχουν στις μετρήσεις. Το κλειδί του πίνακα είναι ο κωδικός **PeopleID**, που παράγεται αυτόματα από τη βάση προκειμένου να αντιπροσωπεύει κατά μοναδικό τρόπο το φοιτητή. Τα πεδία **Onoma**, **Epitheto** και **Kwdikos** περιέχουν τα αντίστοιχα περιγραφικά δεδομένα για το φοιτητή, δηλαδή το ονοματεπώνυμο και τον κωδικό (*Αριθμός Μητρώου*). Τέλος, είναι το πεδίο **TID** που υλοποιεί τη σύνδεση *"πολλά-προς-ένα"* μεταξύ των σπουδαστών και των ομάδων (πολλοί σπουδαστές μπορούν να ανήκουν σε μία ομάδα).

Ο πίνακας των ομάδων (**Teams**) περιέχει απλώς το κλειδί **TID**, που επίσης παράγεται αυτόματα για να ορίζει τη μοναδικότητα της κάθε ομάδας, και το όνομα της ομάδας (πεδίο **Name**).

Ο πίνακας **Measures** έχει κλειδί το μοναδικό αριθμό **MID** για κάθε μέτρηση. Ακολουθεί το πεδίο **TID** που, ομοίως με το αντίστοιχο του *Members*, υλοποιεί τη σχέση *"πολλά-προς-ένα"* μεταξύ των μετρήσεων και των ομάδων (πολλές μετρήσεις μπορούν να "ανήκουν" σε μία ομάδα). Τα πεδία **Location** και **DatenTime** περιγράφουν στοιχεία της μέτρησης όπως η περιοχή μέτρησης και η ημερομηνία και ώρα που εκτελέστηκαν.

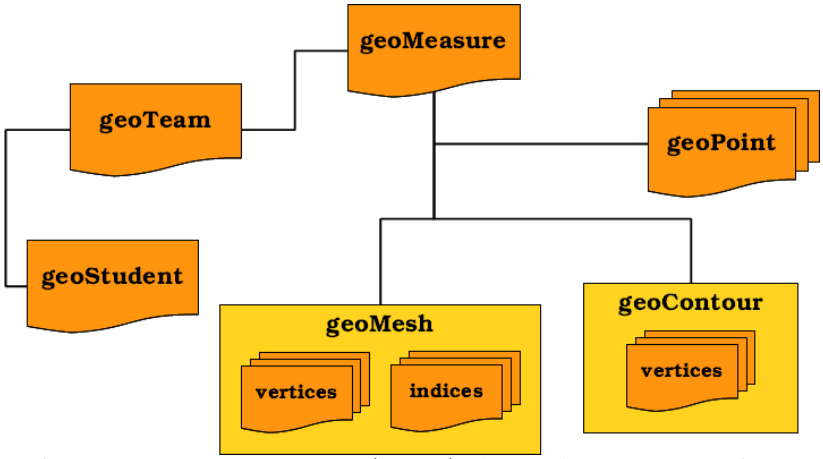
Τέλος, ο "βασικός" πίνακας **Points** περιέχει τα σημεία που μετρήθηκαν στις ασκήσεις. Το πεδίο **PID** αποτελεί κλειδί μοναδικότητας για τα σημεία και παράγεται αυτόματα από τη βάση για να τα χαρακτηρίζει. Το πεδίο **MID** ορίζει σε ποια μέτρηση "χτυπήθηκε" το σημείο και υλοποιεί τη σχέση *"πολλά-προς-ένα"* μεταξύ σημείων και μετρήσεων (πολλά σημεία ανήκουν σε μία μέτρηση). Ο κωδικός του σημείου, όπως αυτός είναι δηλωμένος από το όργανο και το *κροκί*, καταχωρείται στο πεδίο **Pname**. Ασφαλώς, στον πίνακα υπάρχουν και τα 3 πεδία που περιέχουν τις συντεταγμένες των μετρήσεων (**X**, **Y**, **Z**), τα πλέον χρήσιμα δεδομένα της *γεωβάσης*. Τέλος, υπάρχει το πεδίο **isHeight** που προσδιορίζει, με μία

δήλωση αλήθειας, αν το σημείο είναι *υψομετρικό*, αν δηλαδή θα ληφθεί υπόψη κατά την *απεικόνιση* και την *παραγωγή ισοϋψών*.

Όπως προκύπτει από τις παραπάνω σχέσεις, κάθε *Ομάδα* περιέχει *Σπουδαστές* και *Μετρήσεις* και, κάθε *Μέτρηση* με τη σειρά της, περιέχει *Σημεία*. Αυτό μας δίνει τη δυνατότητα να "φιλτράρουμε" τα σημεία κατά μετρήσεις, ομάδες ή ακόμα και σπουδαστές.

Δομή δεδομένων στη μνήμη

Η παραπάνω υλοποίηση των δεδομένων αφορά τη βάση και τον τρόπο που αυτά αποθηκεύονται μόνιμα σε ένα κεντρικό αρχείο. Εκτός αυτού, όμως, το πρόγραμμα χρειάζεται να φορτώσει τις πληροφορίες αυτές στη μνήμη και να τις χειριστεί γρήγορα και ευέλικτα, όταν απαιτείται να γίνει προβολή και επεξεργασία επί των μετρήσεων. Έτσι, προέκυψε η ανάγκη να δημιουργηθούν οι αντίστοιχες προγραμματιστικές **δομές** (**structs**) που θα διατηρούν τα αποτελέσματα των μετρήσεων όσο εκτελείτε η εφαρμογή, προκειμένου να είναι διαχειρίσιμα με μία παρόμοια λογική με αυτήν της βάσης.



Σχήμα 3.5: Οι προγραμματιστικές δομές (structs) που δημιουργήθηκαν για τους σκοπούς της εφαρμογής

Στο Σχήμα 3.5 περιγράφεται η διάταξη των δομών που δημιουργήθηκαν για το πρόγραμμα. Στην ουσία, είναι το ουσιαστικότερο κομμάτι του περιβάλλοντος που δημιουργήθηκε για να αναπτυχθεί η εφαρμογή. Η αντιστοιχία μεταξύ των πινάκων της γεωβάσης και των structs του κώδικα είναι προφανής. Οι σπουδαστές (*Members* → **geoStudent**), οι ομάδες (*Teams* → **geoTeam**), οι μετρήσεις (*Measures* → **geoMeasure**) και τα σημεία (*Points* → **geoPoint**) υπάρχουν ως οντότητες και στις δύο περιπτώσεις, απλώς η υλοποίηση των προγραμματιστικών δομών απαιτεί μια διαφορετική προσέγγιση στο ζήτημα. Επιπλέον με τη χρήση των παραπάνω δομών, συμπεριλαμβάνεται και ένα εγγενές αντικείμενο της βιβλιοθήκης *Direct3D* που αντιπροσωπεύει τη γραφική οντότητα ενός αντικειμένου και ονομάζεται **Mesh**. Έτσι, κάθε μέτρηση περιέχει μία υλοποίηση του αντικείμενου *Mesh* με ονομασία **geoMesh** που διατηρεί τα δεδομένα των 3D γραφικών

(**vertices**, **indices**). Τέλος, δημιουργείται ένα τελευταίο αντικείμενο που ονομάζεται **geoContour**, το οποίο επί της ουσίας περιέχει απλώς **2 vertices** που ορίζουν το τμήμα της *ισοϋψούς* σε κάθε τρίγωνο.

Για την φόρτωση των δεδομένων χρησιμοποιούνται οι δομές αυτές που "ξεκινούν" από τις μετρήσεις. Δηλαδή, στο κεντρικό περιβάλλον της εφαρμογής φορτώνεται ένα πίνακας *geoMeasure[]* που καταχωρεί τις μετρήσεις που έχουν φορτωθεί στη μνήμη κάθε στιγμή. Κάθε *geoMeasure*, με τη σειρά του, περιέχει ένα πίνακα *geoPoint[]* που περιέχει τις συντεταγμένες. Όταν μία μέτρηση φορτώνεται από τη βάση για να απεικονιστεί στο 3D περιβάλλον, προστίθεται ένα νέο στοιχείο *geoMeasure* στον πίνακα των μετρήσεων το οποίο, κατόπιν, αρχικοποιείται ως προς τα δεδομένα και τα γραφικά.

Η αρχικοποίηση του καινούργιου *geoMeasure* γίνεται με την φόρτωση των *geoPoint* από τη βάση και, εν συνεχεία, γίνονται οι κατάλληλες διεργασίες για τον τριγωνισμό. Με βάση αυτά τα στοιχεία, καθορίζονται τα γραφικά (*geoMesh*) με την αντιστοιχία *σημεία* → *vertices* και *τριγώνα* → *indices*. Τέλος, γίνεται ο υπολογισμός του πίνακα *geoContour[]*, που περιέχει τις γραμμές που υλοποιούν τις ισοϋψείς με χρήση του ειδικού αλγορίθμου ισοϋψών (βλέπε *Αλγόριθμος υπολογισμού ισοϋψών*).

3.4 Ειδικοί υπολογισμοί

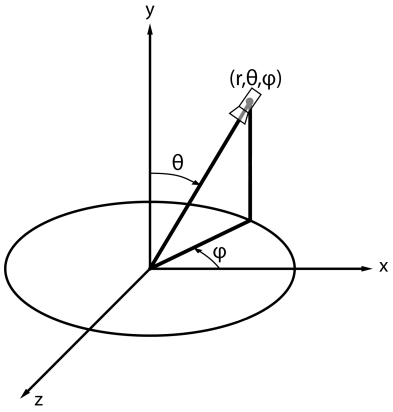
Κίνηση της κάμερας γραφικών

Ένα ζήτημα που πρέπει να ορίζεται πάντα κατά την απεικόνιση τρισδιάστατων γραφικών στην οθόνη, είναι το σημείο από το οποίο "βλέπει" κανείς το *3D αντικείμενο*. Αυτό γίνεται γιατί η δομή του αντικειμένου στις 3 διαστάσεις πρέπει να προβληθεί σε ένα επίπεδο για να παραχθεί κάθε στιγμή το *"καρέ"* που απεικονίζεται στην οθόνη (από τη στιγμή που η οθόνη είναι ένα αντικείμενο 2 διαστάσεων). Επομένως, για κάθε "καρέ" πρέπει να είναι δηλωμένα όχι μόνο τα σημεία και τα αντικείμενα που θα απεικονιστούν, αλλά και το σημείο στο οποίο τοποθετείται ο "παρατηρητής".

Το ρόλο αυτό τον αναλαμβάνει η **κάμερα** (**camera**) των γραφικών. Η *κάμερα* υλοποιεί μία συγκεκριμένη προβολή του τρισδιάστατου χώρου από ένα συγκεκριμένο σημείο στο οποίο "στήνεται" προς ένα σημείο στο οποίο "κοιτάζει". Έτσι, κάθε στιγμή πρέπει να ορίσουμε *δύο σημεία*: τη **θέση** της (**camera position**) και το **στόχο** της (**camera target**). Αυτά τα δύο σημεία δεν αρκεί να υπολογιστούν μία φορά, αφού θέλοντας να δώσουμε την ελευθερία στο χρήστη να περιηγηθεί στον 3Δ κόσμο των μετρήσεων, υλοποιήσαμε ένα σύστημα κίνησης σε αυτό με βάση τα δεδομένα εισόδου από το ποντίκι. Έτσι, ο ενδιαφερόμενος μπορεί να **κινηθεί** στο χώρο (**pan**), να **περιστραφεί** γύρω από το σημείο-στόχος (**rotate**) και να **πλησιάσει/απομακρυνθεί** από το σημείο αυτό (**zoom**).

Κάθε μία από αυτές τις 3 πράξεις, αλλάζει τις συντεταγμένες των 2 σημείων ελέγχου της κάμερας (**position**, **target**). Επομένως, προέκυψε η ανάγκη να γίνουν οι κατάλληλοι γεωμετρικοί υπολογισμοί που με βάση την κατάσταση της προβολής κάθε στιγμή, θα υπολογίζουν τις συντεταγμένες για τα 2 αυτά σημεία.

Η υλοποίηση των υπολογισμών αυτών γίνεται με τη χρήση ενός κλασικού **συστήματος πολικών συντεταγμένων**, διατηρώντας στη μνήμη: τις συντεταγμένες του σημείου-στόχου ($\mathbf{x_t}$, $\mathbf{y_t}$, $\mathbf{z_t}$), την οριζόντια γωνία $\boldsymbol{\phi}$ και την κατακόρυφη $\boldsymbol{\vartheta}$ (Σχήμα 3.6). Αυτά τα **5 μεγέθη** επαρκούν για να υπολογιστούν, τελικά, οι καρτεσιανές συντεταγμένες της θέσης της κάμερας.



Σχήμα 3.6: Το πολικό σύστημα συντεταγμένων που υλοποιείται για τη θέση της κάμερας

Ο λόγος που χρησιμοποιήθηκαν πολικές συντεταγμένες για την περιγραφή της θέσης της κάμερας, σε σχέση με το στόχο, είναι ότι αυτό διευκόλυνε τον τρόπο με τον οποίο οι κινήσεις του ποντικιού "μεταφράζονται" στις καρτεσιανές συντεταγμένες των δύο σημείων ελέγχου της κάμερας. Έτσι, όταν είναι πατημένο το πλήκτρο που ενεργοποιεί την περιστροφή (*αριστερό-κλικ*) για κάθε οριζόντια κίνηση του ποντικιού, προστίθεται αντίστοιχη ποσότητα στη *γωνία φ*, ενώ για κάθε κατακόρυφη κίνηση, προστίθεται αντίστοιχη ποσότητα στη *γωνία ϑ*.

Όταν είναι πατημένο το πλήκτρο που ενεργοποιεί τη μετακίνηση -*pan*- (*δεξί-κλικ*) τότε οι οριζόντια κίνηση και η κάθετη κίνηση μεταβάλουν τις καρτεσιανές συντεταγμένες $\boldsymbol{x_t}$ και $\boldsymbol{y_t}$ του σημείου-στόχου. Το ιδιαίτερο αναφορικά με την κίνηση αυτή, είναι ότι δεν υπάρχει απόλυτη αντιστοίχηση μεταξύ των αξόνων *X*, *Y* της κίνησης του ποντικού επί της οθόνης με τις συντεταγμένες x_t, y_t. Αυτό γίνεται γιατί, προκειμένου να είναι "φυσική" η κίνηση, λαμβάνεται υπόψη η *γωνία φ* για να αποφασιστεί πόσο επιδρά κάθε κίνηση του ποντικού σε κάθε διάσταση του σημείου-στόχου.

Τελικά, ο κώδικας που μεταφράζει τις κινήσεις του ποντικιού στις αλλαγές στα σημεία της κάμερας είναι:

```
1   private void ReadInput()
2   {
3       MouseState mstate = dmouse.CurrentMouseState;
4          if (mstate.GetMouseButtons()[0] != 0)
5          {
6               hangle -= (float)mstate.X / 50f;
7               vangle += (float)mstate.Y / 50f;
8          }
9          else if (mstate.GetMouseButtons()[1] != 0)
10         {
11                      ctarget.X -= ((float)mstate.Y / 5f) * (float)Math.Sin(hangle) +
        ((float)mstate.X / 5f) * (float)Math.Sin(hangle + Math.PI/2);
12              ctarget.Y -= ((float)mstate.Y / 5f) * (float)Math.Cos(hangle) + ((float)mstate.X /
        5f) * (float)Math.Cos(hangle + Math.PI/2);
```

```
13          }
14
15          czoom -= (float)mstate.Z / 80f;
16          if (czoom < 10)
17              czoom = 10;
18  }
```

Ο κώδικας που υπολογίζει τις καρτεσιανές συντεταγμένες της θέσης της κάμερας, από τις αντίστοιχες πολικές, είναι:

```
19  cposition.X = (float)czoom * (float)Math.Sin(hangle) * (float)Math.Cos(vangle) + ctarget.X;
20  cposition.Y = (float)czoom * (float)Math.Cos(hangle) * (float)Math.Cos(vangle) + ctarget.Y;
21  cposition.Z = (float)czoom * (float)Math.Sin(vangle) + ctarget.Z;
```

Αλγόριθμος υπολογισμού ισοϋψών

Ο αλγόριθμος υπολογισμού ισοϋψών αναφέρθηκε περιγραφικά στο κεφάλαιο *"Τοπογραφική Προσέγγιση"*. Εδώ θα εξετάσουμε την υλοποίησή του από προγραμματιστική άποψη και τον τρόπο με τον οποίο αντιμετωπίστηκαν οι δυσκολίες της υλοποίησής του.

Για την εκτέλεση της πράξης της παραγωγής των ισοϋψών δομήθηκε μία συνάρτηση, με ονομασία *SetupContrours(geoMeasure meas, float isodistance)* που λαμβάνει ως ορίσματα τη μέτρηση για την οποία θα παράξει τις ισοϋψείς και την ισοδιάσταση. Θα πρέπει, ασφαλώς, η μέτρηση να έχει αρχικοποιηθεί, να έχουν καταχωρηθεί σημεία σε αυτήν και να έχει προηγηθεί η διαδικασία του τριγωνισμού, προκειμένου να είναι δυνατός ο υπολογισμός των ισοδυναμικών καμπυλών.

Η συνάρτηση αυτή επαναλαμβάνει την ίδια διαδικασία για κάθε τρίγωνο μεταξύ των σημείων. Η διαδικασία αυτή μπορεί να χωριστεί σε 2 μέρη:

1. **Υπολογισμός** σημείων **χαμηλότερου**, **υψηλότερου** και **μέσου** υψομέτρου.
2. **Υπολογισμός** των **σημείων** της **ισοϋψούς** με βάση αυτά και εφαρμόζοντας γραμμική παρεμβολή σε δύο πλευρές ανάλογα την περίπτωση.

Ο κώδικας της συνάρτησης αυτής παρατίθεται παρακάτω:

```
22  private void setupCountours(ref geoMeasure meas, float isodistance)
23          {
24              meas.contours = new geoContour[0];
25              geoPoint minp, midp, maxp, fpoint, spoint;
26              float height;
27
28              // Ξεκινάμε την επανάληψη για κάθε τρίγωνο
29              for (int i = 2; i < meas.indices.Length; i += 3)
30              {
31                  // Υπολογίζουμε το χαμηλότερο και το υψηλότερο σημείο του τριγώνου
32                  minp = meas.Points[meas.indices[i]];
33                  midp = meas.Points[meas.indices[i]];
34                  maxp = meas.Points[meas.indices[i]];
35                  for (int j = i - 2; j < i; j++)
36                  {
37                      if (meas.Points[meas.indices[j]].Z < minp.Z)
38                          minp = meas.Points[meas.indices[j]];
39                      else if (meas.Points[meas.indices[j]].Z > maxp.Z)
40                          maxp = meas.Points[meas.indices[j]];
41                  }
```

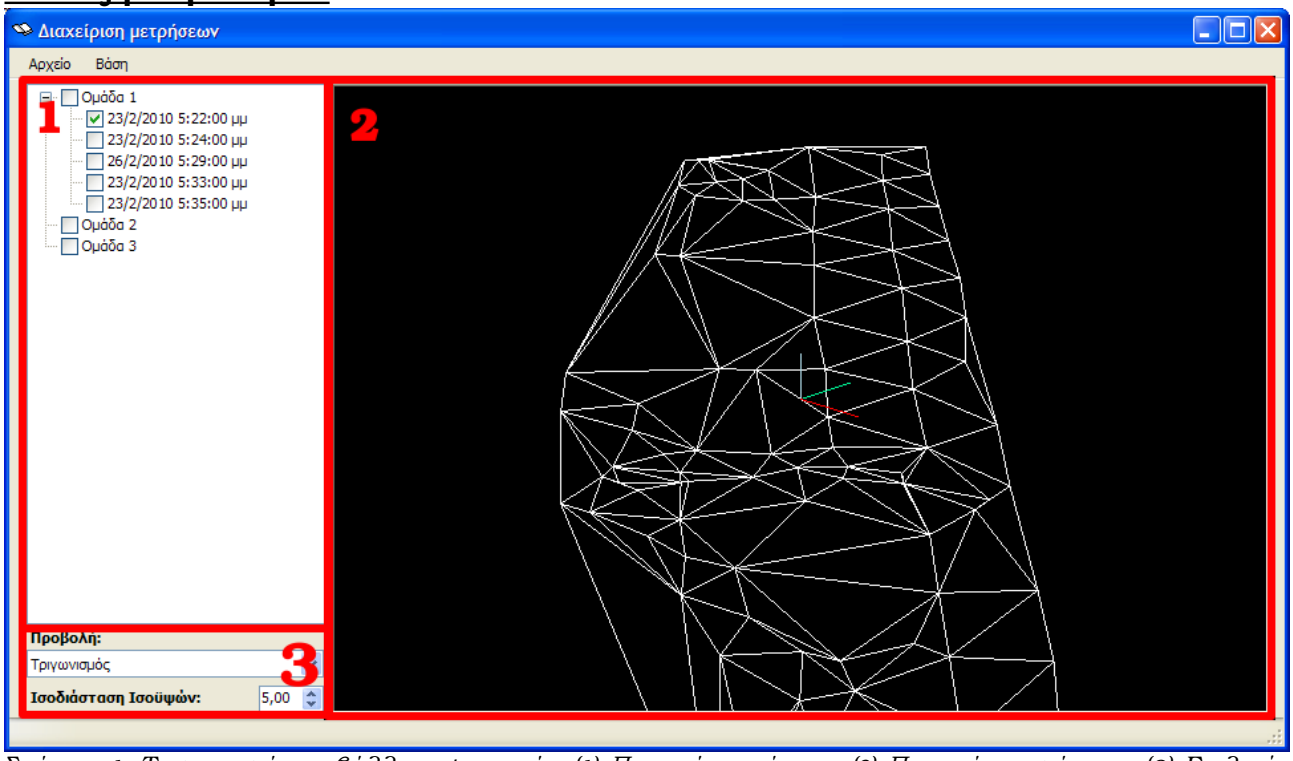
```
42
43                  for (int j = i - 2; j <= i; j++)
44                  {
45                      if (meas.Points[meas.indices[j]].X != minp.X &&
        meas.Points[meas.indices[j]].Y != minp.Y && meas.Points[meas.indices[j]].Z != minp.Z &&
        meas.Points[meas.indices[j]].X != maxp.X && meas.Points[meas.indices[j]].Y != maxp.Y &&
        meas.Points[meas.indices[j]].Z != maxp.Z)
46                          midp = meas.Points[meas.indices[j]];
47                  }
48
49                  minp.Y = -minp.Y;
50                  midp.Y = -midp.Y;
51                  maxp.Y = -maxp.Y;
52
53                  // Για κάθε μία ισοϋψή που περνάει μεταξύ των 2 σημείων αυτών, υπολογίζουμε τις
        πλευρές θα περάσει
54                  height = minp.Z - minp.Z % isodistance;
55                  while (height < maxp.Z)
56                  {
57                      if (height > minp.Z)
58                      {
59                          // Αφού υπάρχει ισοϋψής, το ένα σημείο περνάει ανάμεσα από το χαμηλότερο
        και το υψηλότερο σημείο. Κάνουμε γραμμική παρεμβολή.
60                          fpoint = linearInterpolation(minp, maxp, height);
61
62                          // Αν το ενδιάμεσο σημείο είναι πάνω από το ύψος της ισοϋψούς, τότε το
        2ο σημείο της ισοϋψούς περνάει ανάμεσα από το χαμηλότερο και το μεσαίο, αλλιώς περνάει από
        το μεσσαίο και το υψηλότερο.
63                          if(midp.Z > height)
64                          {
65                              spoint = linearInterpolation(minp, midp, height);
66                          }
67                          else
68                          {
69                              spoint = linearInterpolation(midp, maxp, height);
70                          }
71
72                          Array.Resize<geoContour>(ref meas.contours, meas.contours.Length + 1);
73                          meas.contours[meas.contours.GetUpperBound(0)] = new geoContour(fpoint,
        spoint, device);
74                      }
75
76                      height += isodistance;
77                  }
78
79                  meas.hascontour = true;
80              }
81          }
```

4. Παρουσίαση εφαρμογής

Σκοπός του κεφαλαίου είναι να παρουσιαστεί το περιβάλλον χρήσης και να δοθεί η περιγραφή ενός πλήρους κύκλου εργασιών.

4.1 Κεντρικό παράθυρο

Διάταξη παραθύρου



Σχήμα 4.1: Το κεντρικό περιβάλλον εφαρμογής: (1) Περιοχή μετρήσεων, (2) Περιοχή απεικόνισης, (3) Επιλογές προβολής

Στο κεντρικό παράθυρο (Σχήμα 4.1) εκτελούνται όλες οι βασικές ενέργειες του χρήστη. Εδώ γίνεται η διαχείριση των μετρήσεων, η εμφάνισή τους στο 3D περιβάλλον και η εξαγωγή τους. Η φόρμα του μπορεί να χωριστεί σε 3 βασικά κομμάτια, όπως είναι διαμορφωμένα και στο Σχήμα 4.1:

1. **Περιοχή μετρήσεων:** Εδώ εμφανίζεται, σε δενδροειδή μορφή, η βασική δομή των καταχωρημένων μετρήσεων της ενεργής βάσης.

2. **Περιοχή απεικόνισης:** Σε αυτή την περιοχή γίνεται η τρισδιάστατη απεικόνιση των ενεργών μετρήσεων.

3. **Επιλογές προβολής:** Εδώ μπορούμε να ορίσουμε τις βασικές παραμέτρους της

προβολής.

Στην περιοχή μετρήσεων παρουσιάζονται οι ομάδες και οι μετρήσεις της ανοιγμένης γεωβάσης σε δενδροειδή μορφή. Στο πρώτο επίπεδο βρίσκονται οι ομάδες που συμμετέχουν στις μετρήσεις και στο δεύτερο εμφανίζονται οι μετρήσεις που ανήκουν σε κάθε ομάδα με βάση την ημερομηνία. Κάθε μέτρηση βρίσκεται μαζί με το αντίστοιχο κουτί ελέγχου της (*check-box*) το οποίο ενεργοποιεί/απενεργοποιεί, αντιστοίχως, την εμφάνισή της στην 3Δ απεικόνιση.

Η περιοχή απεικόνισης είναι το παράθυρό προς την τρισδιάστατη προβολή των μετρήσεων. Για να ξεκινήσει η απεικόνιση πρέπει να επιλεγεί η πρώτη μέτρηση που θέλουμε να δούμε. Κατόπιν, μέσα από το παράθυρο αυτό μπορούμε να κινηθούμε στο χώρο με τις 3 βασικές ενέργειες:

- **Μετακίνηση (pan):** Η μετακίνηση υλοποιείται κρατώντας πατημένο το *δεξί-κλικ* του ποντικιού και κουνώντας το προς κάποια κατεύθυνση. Η κίνηση γίνεται σαν ο δείκτης να "πιάνει" την προβολή από το σημείο στο οποίο βρίσκεται και να τη μεταφέρει επί του οριζόντιου επιπέδου (αυτού που υλοποιείται από τις διαστάσεις Χ και Υ).

- **Περιστροφή (rotate):** Υλοποιείται κρατώντας πατημένο το *αριστερό-κλικ* του ποντικιού και κουνώντας το προς κάποια κατεύθυνση. Η οριζόντια κίνηση μας περιστρέφει γύρω από τον άξονα Z, ενώ η κατακόρυφη αλλάζει την κατακόρυφη γωνία από την οποία κοιτάζουμε τις μετρήσεις (η γωνία μεταξύ του άξονα προβολής και του οριζόντιου επιπέδου.

- **Μεγέθυνση/σμίκρυνση (zoom):** Η μεγέθυνση και σμίκρυνση μας μετακινεί πιο κοντά ή πιο μακριά από το σημείο-στόχο της κάμερας και υλοποιείται με το *"ροδάκι"* (*scroll-wheel*) του ποντικιού. Η κίνηση της ρόδας *προς τα πάνω* μας φέρνει πιο κοντά στις μετρήσεις (*μεγέθυνση*), ενώ η κίνηση *προς τα κάτω* μα απομακρύνει από αυτές (*σμίκρυνση*).

Για να είναι πιο αντιληπτή η έννοια του χώρου στον οποίο βλέπουμε τις μετρήσεις, εμφανίζονται στο *σημείο-στόχο* (*target point*) της κάμερας οι 3 άξονες του χώρου: ο **άξονας Χ** με **κόκκινο** χρώμα, ο **άξονας Υ** με **πράσινο** και ο **άξονας Ζ** (υψόμετρα) με **γαλάζιο**. Έτσι, είναι ευκολότερο να αντιληφθούμε το πως κινούμαστε στο χώρο (πχ., το *pan* που γίνεται κατά το οριζόντιο επίπεδο).

Τέλος, στο αριστερό-κάτω μέρος της φόρμας υπάρχουν οι επιλογές για την απεικόνιση. Από εδώ μπορούμε να επιλέξουμε το είδος της απεικόνισης και την ισοδιάσταση των

ισοϋψών. Τα είδη προβολής είναι:

- **Σημεία:** εμφανίζονται μόνο τα σημεία των μετρήσεων στο χώρο.
- **Τριγωνισμός:** εμφανίζεται η μορφολογία του εδάφους με συνεχόμενα τρίγωνα. Παρουσιάζεται, στην ουσία, το αποτέλεσμα του *τριγωνισμού Delaunay*.
- **Επιφάνειες:** παρουσιάζει το έδαφος ως χρωματιστές επιφάνειες με χρήση φωτισμού.
- **Φωτορεαλισμός:** γίνεται μία απόπειρα φωτορεαλιστικής απεικόνισης, δηλαδή το ανάγλυφο αναπαρίσταται με την πλήρη επιφάνειά του, κάποια *υφή* (*texture*) και ένα στοιχειώδη φωτισμό.

Κεντρικό μενού

Για τις βασικές ενέργειες της εφαρμογής, υπάρχει το κλασικό κεντρικό μενού των *Windows*, στο επάνω μέρος του παραθύρου. Αυτό περιέχει τα μενού *Αρχείο* και *Βάση* που περιγράφονται λεπτομερώς στα ακόλουθα δύο υποκεφάλαια.

Μενού "Αρχείο"

Το μενού ***Αρχείο*** διαθέτει επιλογές για το αρχείο γεωβάσης της εφαρμογής. Από εδώ μπορούμε να δημιουργήσουμε μια νέα βάση, να ανοίξουμε μία υπάρχουσα αποθηκευμένη και να αλλάξουμε τις ιδιότητες της ήδη ανοικτής βάσης και να γίνει εξαγωγή της απεικόνισης σε μορφή *DXF* για χρήση από άλλα προγράμματα. Η διαδικασία της δημιουργίας μίας βάσης καλύπτεται από το κεφάλαιο *"Δημιουργία αρχείου εργασίας"*, συνεπώς η αναφορά στην επιλογή ***Δημιουργία...*** είναι περιττή σε αυτό το σημείο.

Η επιλογή ***Άνοιγμα...*** θα εμφανίσει ένα παράθυρο διαλόγου που μας επιτρέπει να περιηγηθούμε στους δίσκους του συστήματος και να επιλέξουμε το αρχείο της βάσης που θέλουμε να ανοίξουμε. Σε αυτό το σημείο είναι χρήσιμο να σημειωθεί ότι η βάση έχει κατάληξη *mdb,* και άρα είναι πρακτικά ένα αρχείο του προγράμματος *MS Access*. Είναι, συνεπώς, απαραίτητο να γνωρίζουμε εκ των προτέρων ότι το αρχείο που θα ανοίξουμε έχει τη δομή μίας βάσης της εφαρμογής, καθώς στο παράθυρο αυτό ενδεχομένως να εμφανίζονται και αρχεία που ναι μεν είναι *mdb*, αλλά δεν έχουν την κατάλληλη δομή για να τα διαχειριστεί η εφαρμογή.

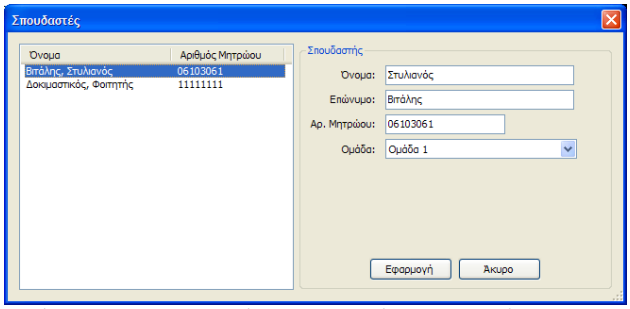
Μέσω της επιλογής ***Εξαγωγή...*** μας δίνεται η δυνατότητα να εξάγουμε τις υπάρχουσες μετρήσεις σε μορφή DXF, την πρότυπη μορφή αρχείου (standard file format) για σχέδια που υποστηρίζεται από τη συντριπτική πλειοψηφία προγραμμάτων CAD, GIS, 3D modeling κ.α. Κάνοντας κλικ εκεί, εμφανίζεται το κλασικό παράθυρο διαλόγου αποθήκευσης των

Windows, μέσω του οποίου μπορούμε να επιλέξουμε τον προορισμό στον οποίο θα αποθηκευθεί το αρχείο, όπως και το όνομά του. Πρέπει να σημειωθεί ότι κατά την εξαγωγή σε DXF, θα αποθηκευθούν μόνο οι εμφανιζόμενες μετρήσεις, προκειμένου να υπάρχει η δυνατότητα να παραχθεί ένα υπόβαθρο μόνο με τις επιθυμητές μετρήσεις. Κάθε μέτρηση θα εξαχθεί σε 3 ξεχωριστά επίπεδα (layers): ένα για τα σημεία, ένα για τον τριγωνισμό και ένα για τις ισοϋψείς. Κάθε επίπεδο έχει για όνομα την ημερομηνία των μετρήσεων και, αν είναι επίπεδο τριγωνισμού ή ισοϋψών, συνοδεύεται από την αντίστοιχη ένδειξη (triangulation ή contours).

Τελευταία είναι η επιλογή ***Έξοδος*** από την εφαρμογή. Η επιλογή θα τερματίσει την απεικόνιση, θα κλείσει όλες τις διεργασίες που εκτελούνται στο παρασκήνιο της εφαρμογής με ασφάλεια και, κατόπιν, θα τερματίσει την ίδια την εφαρμογή. Καλό είναι να προτιμάτε η επιλογή αυτή, αντί οποιουδήποτε άλλου τρόπου εξόδου (πχ., το κουμπί κλεισίματος παραθύρου -X- πάνω-δεξιά στο παράθυρο των Windows).

Μενού "Βάση"

Στο βασικό μενού υπάρχουν τα χαρακτηριστικά της βάσης. Οι δύο διαθέσιμες επιλογές, ***Σπουδαστές...*** και ***Ιδιότητες...*** μας οδηγούν στα παράθυρα διαλόγου όπου μπορούμε, αντίστοιχα, να αλλάξουμε τα δεδομένα για τους σπουδαστές και να διορθώσουμε τις ιδιότητες της βάσης που ορίσαμε αρχικά (κατά τη δημιουργία).

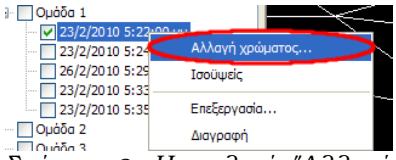


Σχήμα 4.2: Το παράθυρο διαλόγου διαχείρισης των σπουδαστών

Η επιλογή *Σπουδαστές...* προβάλει το παράθυρο που εμφανίζεται στο Σχήμα 4.2, μέσα από το οποίο μπορούν να προστίθενται, αφαιρούνται και επεξεργάζονται οι φοιτητές της βάσης και να ανατίθενται σε κάθε ομάδα. Από την αριστερή λίστα, επιλέγουμε το σπουδαστή που θέλουμε να διορθώσουμε, αλλάζουμε τα στοιχεία του και πατάμε το κουμπί *Εφαρμογή* για να εφαρμοστούν οι αλλαγές στον πίνακα και στη βάση (γίνεται αυτόματη αποθήκευση). Για να προστεθεί ένας νέος σπουδαστής, πατάμε δεξί-κλικ στη λίστα σπουδαστών και επιλέγουμε *Νέος σπουδαστής*. Στη λίστα θα μπει μία καινούργια καταχώρηση που μπορούμε, πλέον, να επεξεργαστούμε κανονικά σαν έναν υπάρχοντα σπουδαστή.

Μενού μετρήσεων

Εκτός από την *προβολή/απόκρυψη* μίας μέτρησης, που γίνεται μέσω των *check-boxes*, μπορούμε να εκτελέσουμε και κάποιες άλλες ενέργειες σε αυτές. Αυτό γίνεται μέσω του ***μενού-ελέγχου μετρήσεων*** που εμφανίζεται κάνοντας **δεξί-κλικ** σε μία μέτρηση στην *Περιοχή μετρήσεων (1)* της εφαρμογής.



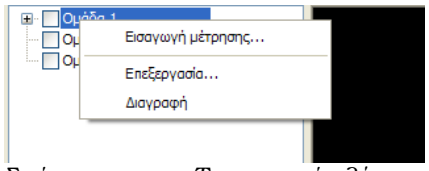
Σχήμα 4.3: Η επιλογή "Αλλαγή χρώματος..." στο μενού-ελέγχου μέτρησης

Μία πολύ χρήσιμη δυνατότητα που δίνεται μέσω του *μενού-ελέγχου μετρήσεων*, είναι ο χρωματισμός της κάθε μίας με διαφορετική απόχρωση. Αυτό γίνεται με την επιλογή ***Αλλαγή χρώματος...*** (βλ. Σχήμα 4.3). Κάνοντας κλικ σε αυτή την επιλογή, εμφανίζεται ένα παράθυρο διαλόγου με χρώματα από τα οποία μπορούμε να επιλέξουμε το επιθυμητό για να "βάψουμε" τα σημεία και τα τρίγωνα της μέτρησης αυτής. Πρέπει να επισημανθεί ότι ο χρωματισμός μετρήσεων αφορά μόνο τα είδη προβολής *Σημεία* και *Τριγωνισμός*, αφού όταν είναι ενεργοποιημένος ο *Φωτορεαλισμός* δεν υπάρχουν χρώματα, αλλά υφές.

Από το *μενού-ελέγχου μετρήσεων* μπορούμε να *ενεργοποιήσουμε/απενεργοποιήσουμε* και την εμφάνιση των ισοϋψών. Όταν εμφανιστεί το μενού (με δεξί-κλικ, όπως αναφέρθηκε και προηγουμένως) η επιλογή ***Ισοϋψείς*** θα είναι τσεκαρισμένη αν η συγκεκριμένη μέτρηση έχει ενεργή την εμφάνισή τους. Κάνοντας απλό κλικ στην επιλογή, αλλάζουμε την κατάσταση προβολή των ισοϋψών (από *ενεργοποιημένες* → *απενεργοποιημένες* ή από *απενεργοποιημένες* → *ενεργοποιημένες*). Όπως αναφέρθηκε ήδη, οι ισοϋψείς είναι πάντα με λευκό χρώμα.

Στο μενού-ελέγχου μετρήσεων μπορούμε να εκτελέσουμε και άλλες βασικές διεργασίες γι αυτές, όπως ***Επεξεργασία...*** και ***Διαγραφή***. Όπως είναι προφανές, η πρώτη επιλογή θα μας μεταφέρει στο παράθυρο διαλόγου διαχείρισης των μετρήσεων, όπου μπορούμε να αλλάξουμε την ημερομηνία, την ομάδα και τα σημεία της μέτρησης. Η δεύτερη επιλογή, μπορεί να αφαιρέσει τη μέτρηση και τα σημεία της από τη γεωβάση. Θα πρέπει να δοθεί προσοχή στο γεγονός ότι τόσο η επεξεργασίας όσο και η διαγραφή είναι ενέργειες μη-αναστρέψιμες.

Μενού ομάδων



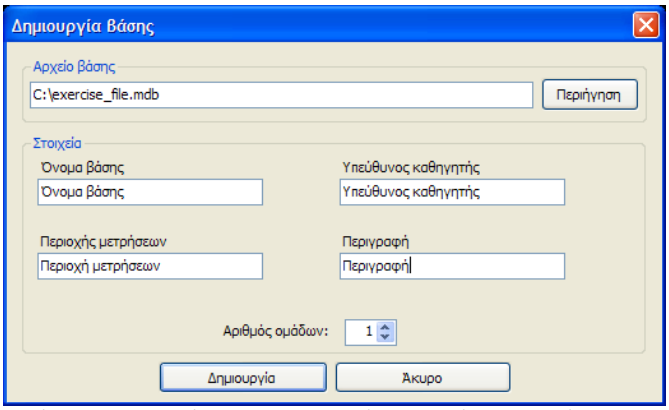
Σχήμα 4.4: Το μενού-ελέγχου ομάδας

Όπως οι μετρήσεις, έτσι και οι ομάδες έχουν το δικό τους μενού-ελέγχου για να γίνονται οι κατάλληλες ενέργειες σε αυτές. Οι επιλογές που δίνονται είναι: ***Εισαγωγή μέτρησης...***, ***Επεξεργασία...*** και ***Διαγραφή***. Η πρώτη επιλογή είναι πολύ βασική, καθώς μέσω αυτής μπορούμε και προσθέτουμε μετρήσεις στη βάση. Το συγκεκριμένο αντικείμενο καλύπτεται λεπτομερώς στο κεφάλαιο *Εισαγωγή μετρήσεων*.

Οι άλλες δύο επιλογές αφορούν στην επεξεργασία της ομάδας και στη διαγραφή της, αντίστοιχα. Η επεξεργασία δίνει απλώς τη δυνατότητα για τη μετονομασία της, στην ουσία. Η διαγραφή θα αφαιρέσει τόσο την ομάδα, όσο και τις μετρήσεις της από τη βάση. Είναι σημαντικό ότι, όπως και στις μετρήσεις, οι όποιες αλλαγές στη βάση δεν μπορούν να αναιρεθούν οπότε πρέπει να είναι πολύ προσεκτική η χρήση των εντολών αυτών.

4.2 Δημιουργία αρχείου εργασίας

Το πρώτο βήμα για τη χρήση του προγράμματος σε μία τοπογραφική εργασία αποτύπωσης είναι η δημιουργία του αντίστοιχου αρχείου εργασίας. Το αρχείο αυτό δεν είναι παρά μία βάση τύπου *MS Access* που καταχωρεί τα στοιχεία της με τον τρόπο που περιγράφηκε στο κεφάλαιο *"Δομή δεδομένων στη βάση"*.



Σχήμα 4.5: Παράθυρο δημιουργίας αρχείου εργασίας

Η δημιουργία αρχείου γίνεται από το μενού ***Αρχείο → Δημιουργία...*** . Θα εμφανιστεί, έτσι, ένα παράθυρο όπου θα οριστούν οι κατάλληλες ιδιότητες για το αρχείο αποτύπωσης.

Ασφαλώς, αρχικά θα πρέπει να επιλεγεί το όνομα του νέου αρχείου που θα δημιουργηθεί. Αυτό γίνεται είτε πληκτρολογώντας την πλήρη διαδρομή (path) μαζί με το όνομα του αρχείου, ασφαλώς, στο αντίστοιχο πλαίσιο κειμένου (text-box), είτε κάνοντας κλικ στο κουμπί *Περιήγηση* για να εμφανιστεί ένα παράθυρο διαλόγου που κάνει πιο εύκολη τη διαδικασία επιλογής του προορισμού.

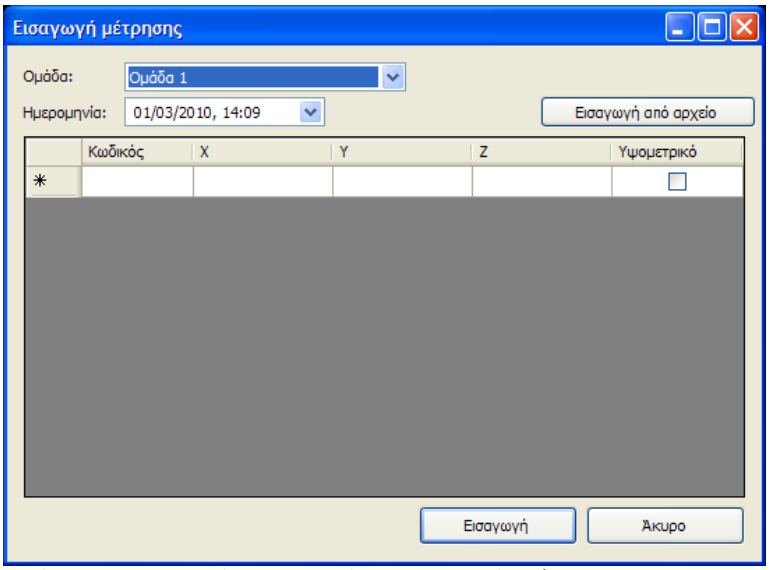
Εν συνεχεία, ζητούνται κάποια περιγραφικά στοιχεία για τη βάση που βοηθούν στην καλύτερη περιγραφή της: το **Όνομα βάσης**, ο **Υπεύθυνος καθηγητής**, η **Περιοχή μετρήσεων** και μία **Περιγραφή** της βάσης. Ο χρήστης έχει την ελευθερία να χρησιμοποιήσει τα πεδία αυτά όπως θεωρεί εκείνος καλύτερα, άλλωστε ο σκοπός είναι να περιγράφουν επαρκώς τις βασικές ιδιότητες της βάσης, καθώς το όνομα του αρχείου δεν είναι ικανό να περιέχει όλες εκείνες τις ποιοτικές πληροφορίες που πιθανόν να θέλουμε να "θυμάται" η βάση.

Εκτός των παραπάνω, μας ζητείται να καθορίσουμε έναν *Αριθμό ομάδων* που συμμετέχουν στις μετρήσεις. Η διάρθρωση των ομάδων μπορεί να επεξεργαστεί και αργότερα από το κεντρικό παράθυρο της εφαρμογής, όπως είδαμε και στο προηγούμενο κεφάλαιο, επομένως ο σκοπός της επιλογής είναι απλώς να μας απαλλάξει από ενδεχόμενο αρχικό φόρτο εργασίας και δεν είναι δεσμευτική η χρήση του.

Πατώντας το κουμπί *Δημιουργία*, το πρόγραμμα κατασκευάζει τη βάση, δημιουργώντας τη δομή πινάκων που χρειάζεται για να καταχωρηθούν τα δεδομένα. Κατόπιν, το κεντρικό παράθυρο εφαρμογής είναι έτοιμο για εργασία στην καινούργια βάση.

4.3 Εισαγωγή μετρήσεων

Η εισαγωγή μετρήσεων είναι η πεμπτουσία της εφαρμογής. Απαιτείται προσοχή κατά την καταχώρηση των δεδομένων, για να είμαστε σίγουροι ότι τα σημεία που θα προστεθούν πληρούν τις απαραίτητες προϋποθέσεις που ορίστηκαν στο με ακρίβεια στο κεφάλαιο *"Προσέγγιση των εργασιών"*.



Σχήμα 4.6: Το παράθυρο διαλόγου εισαγωγής μέτρησης

Η εισαγωγή τον μετρήσεων γίνεται μέσω του μενού-ελέγχου των ομάδων στην *Περιοχή διαχείρισης (Περιοχή (1), Σχήμα 4.1)*. Κάνοντας κλικ στην επιλογή *Εισαγωγή μέτρησης...* εμφανίζεται ένα καινούργιο παράθυρο (Σχήμα 4.6) όπου πρέπει να καθορίσουμε τα στοιχεία της καινούργιας μέτρησης.

Αρχικά είναι τα περιγραφικά στοιχεία της μέτρησης: η **Ομάδα** και η **Ημερομηνία**. Από τη λίστα ομάδων επιλέγουμε ποια από τις ομάδες που συμμετέχουν στις εργασίες έκανε τη μέτρηση. Εν συνεχεία, ορίζουμε την ημερομηνία και ώρα στο σχετικό πλαίσιο (η ημερομηνία μπορεί να επιλεγεί από το σχετικό ημερολόγιο που εμφανίζεται στο πλαίσιο ή μέσω του πληκτρολογίου, αλλά η ώρα εισάγεται μόνο μέσω του πληκτρολογίου).

Αφού ρυθμιστούν τα παραπάνω, πρέπει να εισαχθούν οι συντεταγμένες των σημείων. Τα σημεία καταχωρούνται στη λίστα που βρίσκεται στο κέντρο του παραθύρου και καταλαμβάνει το μεγαλύτερό μέρος του. Τα στοιχεία των σημείων που καταχωρούνται σε 5 στήλες:

- **Κωδικός:** Ο κωδικός σημείου που του έχουμε δώσει κατά τη μέτρηση
- **X, Y, Z:** Οι τρεις στήλες των συντεταγμένων, όπου X και Y είναι οι οριζοντιογραφικές συντεταγμένες και Z το υψόμετρο

- **Υψομετρικό:** Η στήλη αυτή, που αποτελείται από κουτιά-ελέγχου (check-boxes) ορίζει αν το σημείο είναι υψομετρικό ή όχι. Ορίζει, δηλαδή, το αν θα ληφθεί υπόψη το συγκεκριμένο σημείο για την απεικόνιση του αναγλύφου.[5]

Στη λίστα σημείων, μπορούμε να καταχωρήσουμε "με το χέρι" τα σημεία σύμφωνα με τα παραπάνω. Επειδή, όμως, στις περισσότερες περιπτώσεις, οι συντεταγμένες θα έχουν προκύψει από κάποιο αρχείο οργάνου ή από ένα άλλο πρόγραμμα επεξεργασίας μετρήσεων (πχ. σε περίπτωση που έγινε μέτρηση σε πολικές συντεταγμένες και μετατράπηκαν σε οριζοντιογραφικές), υπάρχει η δυνατότητα να εισαχθούν σημεία στη λίστα από ένα αρχείο κειμένου (txt), στα πρότυπα των περισσοτέρων γεωδαιτικών σταθμών της αγοράς.

Το σκοπό αυτό εξυπηρετεί το κουμί ***Εισαγωγή από αρχείο***, που θα εμφανίσει ένα παράθυρο για να επιλέξουμε το αρχείο μετρήσεων. Κατόπιν, το πρόγραμμα θα προσπαθήσει να αναγνωρίσει τη δομή του αρχείου και να καταχωρήσει αυτόματα τα δεδομένα στη λίστα. Όλα τα σημεία θα θεωρηθούν, αρχικά, ως υψομετρικά και η επιλογή *Υψομετρικό* θα είναι ενεργοποιημένη. Επομένως, πρέπει να γίνει μία ανασκόπηση της λίστας και, με προσοχή, να απενεργοποιηθούν τα σημεία εκείνα των οποίων το υψόμετρο (*στήλη Z*) δεν είναι αντιπροσωπευτικό του αναγλύφου.

Η διαδικασία αυτή μπορεί να γίνει για πολλά αρχεία μαζί (πχ., αν τα σημεία μίας μέτρησης είναι περασμένα σε 2 αρχεία, μπορούμε να εισάγουμε αρχικά το ένα και μετά το άλλο), αλλά συνίσταται προσοχή στο να μη χρησιμοποιείται αυτό για συνδυασμό διαφορετικών μετρήσεων στην ίδια λίστα, καθώς αυτό αναιρεί τη χρησιμότητα της κατηγοριοποίησης σημείων ανά μέτρηση.

Αφού έχουν καταχωρηθεί τα σημεία στη λίστα μέσω αρχείου/ων, μπορούν να γίνουν αλλαγές και διορθώσεις σε αυτή μέσω του πληκτρολογίου χωρίς κανένα πρόβλημα. Η καταχώρηση των δεδομένων στη βάση γίνεται μέσω του κουμπιού *Εισαγωγή* που θα αποθηκεύσει απευθείας τις αλλαγές στη ενεργή γεωβάση και θα εμφανίσει τη νέα μέτρηση στη λίστα.

4.4 Παράδειγμα χρήσης της εφαρμογής (tutorial)

Για να γίνει απόλυτα κατανοητός ο τρόπος λειτουργίας και οι δυνατότητες του προγράμματος θα επιχειρήσουμε να παρουσιάσουμε, με τη βοήθεια ενός παραδείγματος, τη διαδικασία δημιουργίας και διαχείρισης ενός αρχείου μετρήσεων για μία εργασία τοπογραφικής αποτύπωσης. Για τους σκοπούς του παραδείγματος, η περιγραφή των μετρήσεων θα αφορά τις ασκήσεις του μαθήματος *Μεγάλες Γεωδαιτικές Ασκήσεις ΙΙ* που

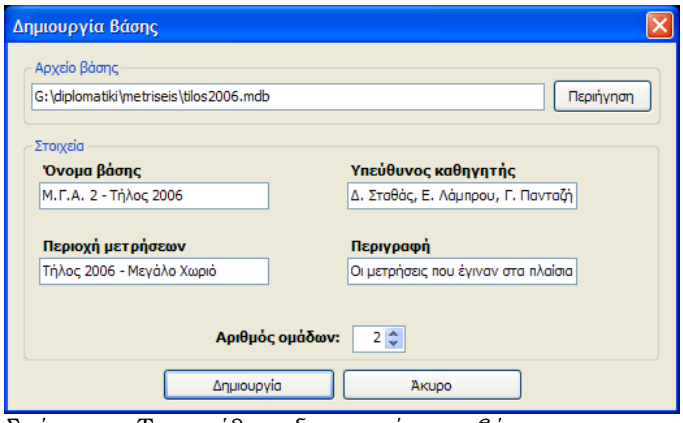
5 Σημειώνεται ότι αν ένα σημείο δε λογίζεται ως υψομετρικό, δε θα εμφανίζεται καθόλου στη τρισδιάστατη απεικόνιση, θα παραμένει όμως στη βάση και θα εξαχθεί κανονικά όταν παράγουμε ένα DXF

έγιναν στην *Τήλο*, το έτος 2006, όπου συμμετείχα ως φοιτητής. Ωστόσο, ως δεδομένα των σημείων, θα χρησιμοποιήσουμε δοκιμαστικές μετρήσεις που μας χορήγησε ο *κ. Γ. Πανταζής*.

Για να χρησιμοποιηθεί το πρόγραμμα, πρέπει να έχουν εκτελεστεί οι μετρήσεις ή μέρος αυτών και να γνωρίζουμε τις συντεταγμένες τους σε κάποιο γεωδαιτικό σύστημα αναφοράς (κατά προτίμηση *ΕΓΣΑ 87*).

Βήμα 1ο: Δημιουργία και διαμόρφωση αρχείου μετρήσεων

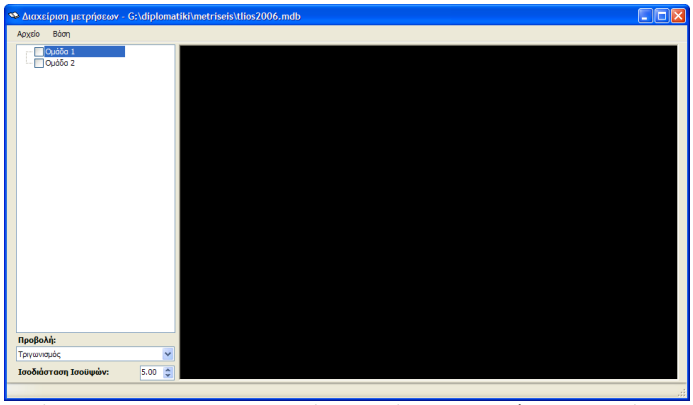
Η διαδικασία ξεκινά με τη δημιουργία μίας γεωβάση που θα διατηρεί τις μετρήσεις. Για το σκοπό αυτό, αφού εκκινήσει η εφαρμογή, επιλέγουμε *Αρχείο* → *Δημιουργία*... . Στο παράθυρο που εμφανίζεται, ορίζουμε τις αρχικές περιγραφικές ιδιότητες της βάσης βλέπε Σχήμα 4.7), επιλέγουμε το αρχείο της γεωβάσης όπου θα καταχωρούνται όλα τα στοιχεία και ορίζουμε έναν αρχικό αριθμό ομάδων για την εργασία.



Σχήμα 4.7: Το παράθυρο δημιουργίας της βάσης.

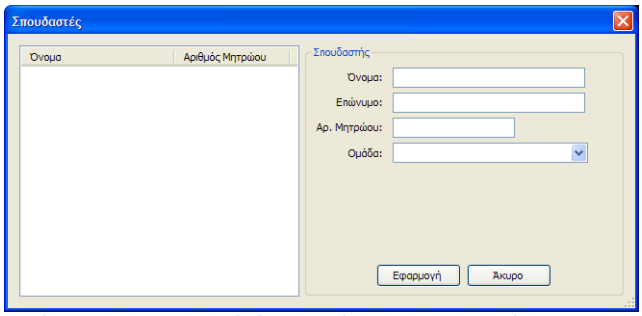
Το πρόγραμμα θα δημιουργήσει το αρχείο και θα το "ανοίξει" στο κεντρικό παράθυρο. Πλέον, όποια εργασία εκτελείται από την εφαρμογή θα καταχωρείται στη βάση άμεσα, όπως συνηθίζεται σε περιπτώσεις προγραμμάτων διαχείρισης βάσεων δεδομένων.

Στην περίπτωση του παραδείγματος, το κεντρικό παράθυρο της εφαρμογής έχει φορτώσει την καινούργια βάση και εμφανίζει τις 2 ομάδες που ζητήσαμε να δημιουργηθούν (Σχήμα 4.8). Σε αυτό το σημείο πρέπει να αρχίσουμε να καταχωρούμε τις υπόλοιπες αρχικές πληροφορίες που αφορούν την εργασία, όπως τους σπουδαστές που συμμετέχουν στις μετρήσεις.

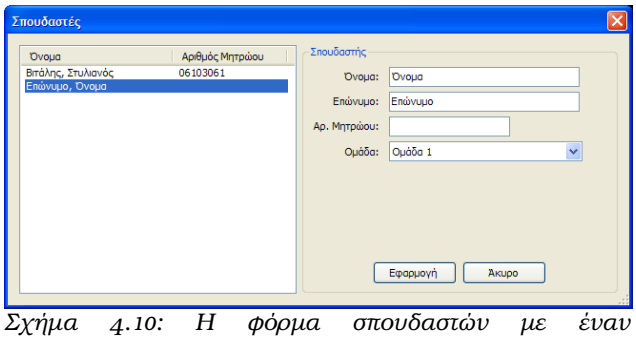


Σχήμα 4.8: Το κεντρικό παράθυρο αμέσως μετά τη δημιουργία της βάσης.

Επιλέγουμε το μενού *Βάση→Σπουδαστές…*, για να εμφανιστεί το αντίστοιχο παράθυρο που, προς το παρόν, είναι κενό (Σχήμα 4.9). Εδώ πρέπει να καταχωρηθεί ο κάθε σπουδαστής ξεχωριστά, με τα βασικά του στοιχεία. Για να προστεθεί ένα καινούργιος σπουδαστής βάση κάνουμε δεξί-κλικ στη λίστα σπουδαστών (αριστερά) και επιλέγουμε *Νέος Σπουδαστής*. Στη λίστα θα προστεθεί μία καταχώρηση με κάποια τυπικά αρχικά στοιχεία (*Όνομα*, *Επώνυμο*), τα οποία επεξεργαζόμαστε και, πατώντας *Εφαρμογή*, καταχωρούμε στη βάση. Για κάθε σπουδαστή πρέπει να συμπληρώσουμε το Ονοματεπώνυμο, τον Αρ. Μητρώου και, ασφαλώς, την ομάδα στην οποία ανήκει. Η ίδια διαδικασία ακολουθείται μέχρι να ολοκληρωθεί η λίστα των σπουδαστών των ασκήσεων.



Σχήμα 4.9: Η αρχική άδεια φόρμα σπουδαστών.



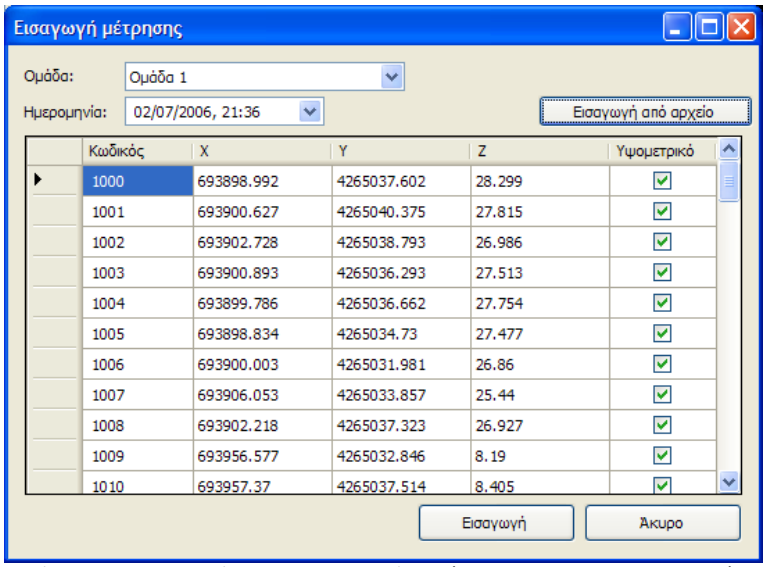
Σχήμα 4.10: Η φόρμα σπουδαστών με έναν καταχωρημένο σπουδαστή και μία καινούργια εγγραφή.

Κλείνοντας τη φόρμα Σπουδαστών επιστρέφουμε στο κεντρικό παράθυρο και η βάση είναι έτοιμη για τη κύρια εργασία των μετρήσεων.

Βήμα 2ο: Καταχώρηση των μετρήσεων

Έχοντας δημιουργήσει τη βάση με όλα τα απαραίτητα αρχικά δεδομένα, είμαστε έτοιμοι να εργαστούμε στο κυρίως κομμάτι της εφαρμογής, στην καταχώρηση και οργάνωση των μετρήσεων που έχουν γίνει. Αυτή η διαδικασία, ασφαλώς, δεν εκτελείται μόνο μία φορά, αλλά κάθε φορά που πραγματοποιείται μία νέα μέτρηση στο πεδίο και υπάρχουν νέα δεδομένα θα χρειαστεί να τα καταχωρήσουμε στη βάση για να την ενημερώσουμε, αναλόγως.

Για να εισάγουμε μία μέτρηση στη βάση, εμφανίζουμε το μενού-ελέγχου της ομάδας που την εκτέλεσε (με δεξί-κλικ στο όνομά της από τη λίστα) και επιλέγουμε *Εισαγωγή μέτρησης…* (βλέπε *Σχήμα 4.4* στο κεφάλαιο *"Μενού ομάδων"*). Θα εμφανιστεί το παράθυρο εισαγωγής μέτρησης, από όπου



Σχήμα 4.11: Παράθυρο εισαγωγής μέτρησης με καταχωρημένα σημεία

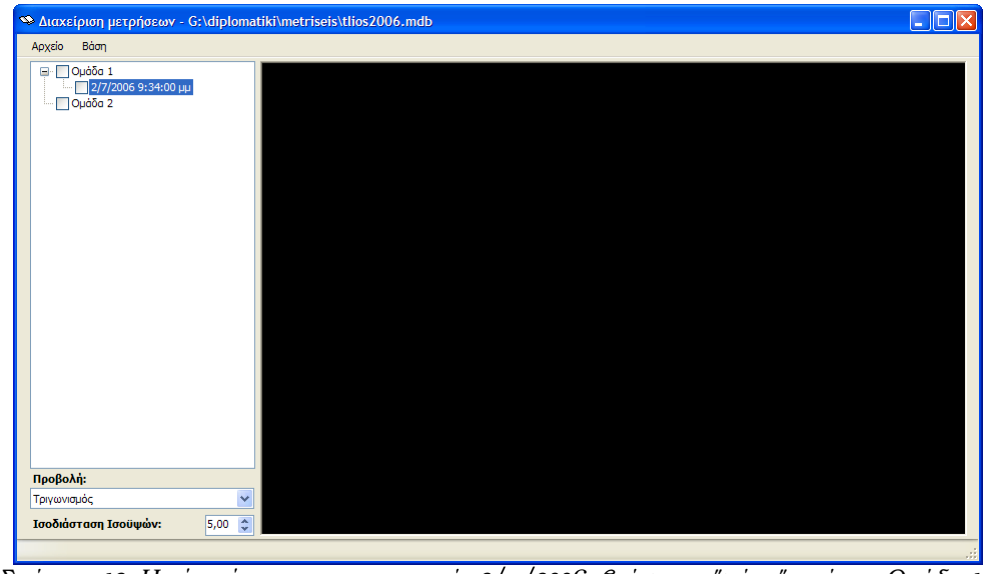
ορίζονται τα δεδομένα της καινούργιας μέτρησης (Σχήμα 4.11).

Το πρόγραμμα έχει επιλέξει αυτόματα την ομάδα για την οποία επιλέξαμε να εισαχθεί η μέτρηση, ωστόσο αν έχει γίνει λάθος, μπορεί να επιλεγεί άλλη ομάδα από τη σχετική λίστα αυτού του παραθύρου. Κατόπιν, επιλέγουμε μία ημερομηνία από το αντίστοιχο πλαίσιο, είτε πληκτρολογώντας την, είτε εμφανίζοντας το σχετικό "ημερολόγιο" που βοηθά στην επιλογή μίας άλλης ημέρας (πατώντας το αντίστοιχο κουμπί στα δεξιά του πλαισίου). Θα πρέπει να καταχωρήσουμε και την ώρα των μετρήσεων, αν αυτό είναι απαραίτητο για τις μετρήσεις (πχ., αν έχουν γίνει μετρήσεις πρωί και βράδυ την ίδια ημέρα).

Πλέον, πρέπει να καταχωρηθούν τα μετρημένα σημεία. Θα πρέπει, εδώ, να ορίσουμε τον *Κωδικό* σημείου για να τα αναγνωρίζουμε (τον κωδικό που δίνεται από το "κροκί" και το όργανο), τι συντεταγμένες *X*, *Y*, *Z* (όπου *Z*, το υψόμετρο) και να ορίσουμε αν θέλουμε το σημείο να ληφθεί υπόψη στην απεικόνιση και την υψομετρική "ανάλυση" του προγράμματος, αν είναι, δηλαδή, *υψομετρικό*. Υπάρχουν 2 επιλογές για την καταχώρηση σημείων. Είτε από αρχείο κειμένου, που συνήθως εξάγουν τα όργανα με καταγραφικό, είτε με το χέρι, αν δεν υπάρχει η αντίστοιχη δυνατότητα από το όργανο. Η διαδικασία είναι "ανοικτή" για τον τρόπο χρήσης της, δηλαδή μπορεί να εισαχθεί ένα (ή και περισσότερα) αρχείο σημείων και, κατόπιν, να επέμβουμε στη λίστα προσθέτοντας, αφαιρώντας ή και αλλάζοντας τα στοιχεία κάποιων σημείων (πχ., μπορεί να διορθώσουμε το υψόμετρο ενός σημείου για το οποίο δεν είχε οριστεί σωστά Ύψος Στόχου κατά τη μέτρηση).

Η εισαγωγή μετρήσεων από αρχείο γίνεται με το κουμπί *Εισαγωγή από αρχείο*. Θα εμφανιστεί ένα παράθυρο διαλόγου για να επιλέξουμε το αρχείο κατάληξης *txt* (ή άλλης) που περιέχει τις μετρήσεις. Αφού επιλεγεί το αρχείο, το πρόγραμμα προσπαθεί να αναγνωρίσει τις γραμμές του και να προσθέσει τα αντίστοιχα σημεία στη λίστα* του παραθύρου εισαγωγής μέτρησης. Εδώ, θα πρέπει με ιδιαίτερη προσοχή να γίνουν οι ενδεχόμενες διορθώσεις που θέλουμε και, κυρίως, να απενεργοποιηθεί η επιλογή *Υψομετρικό* για τα σημεία εκείνα που το υψόμετρο είναι λάθος (πχ., σημεία κτιρίων).

* *Η συνιστώμενη μορφή αρχείου σημείων είναι να περιέχει τα στοιχεία Κωδικός, X, Y, Z χωρισμένα με κενό και κόμματα, σύμφωνα και με το πρότυπο των δοκιμαστικών μετρήσεων του κ. Γ. Πανταζή.*

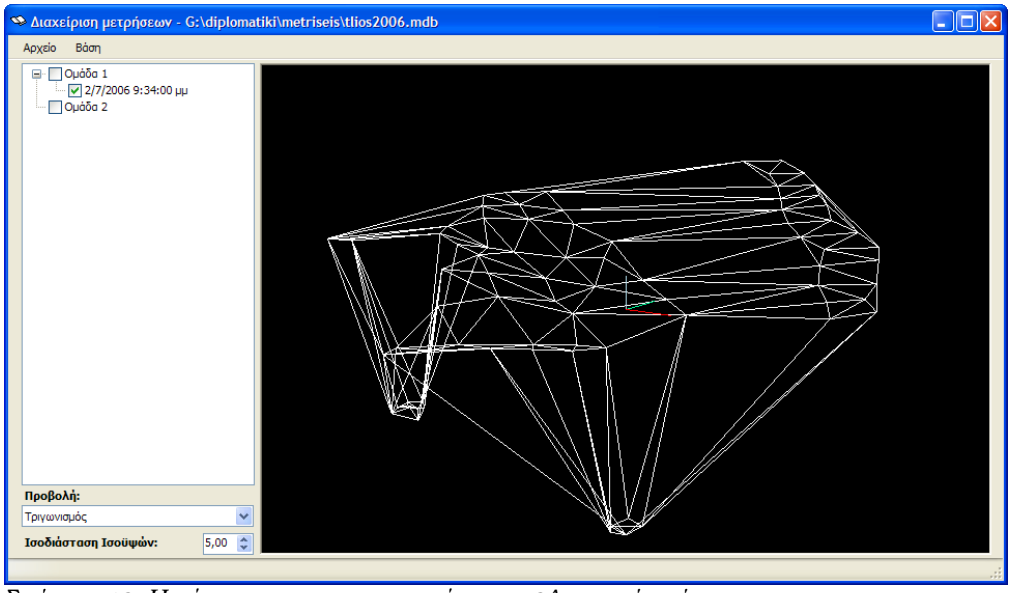


Σχήμα 4.12: Η νέα μέτρηση με ημερομηνία 2/7/2006, βρίσκεται "κάτω" από την Ομάδα 1

Αφού διαμορφωθούν πλήρως τα σημεία, πατώντας το κουμπί *Εισαγωγή*, τα νέα στοιχεία καταχωρούνται στη βάση και πλέον η μέτρηση εμφανίζεται “κάτω” από την ομάδα που την εκτέλεσε (Σχήμα 4.12). Η ίδια διαδικασία ακολουθείται για κάθε μέτρηση.

Βήμα 3ο: Εμφάνιση και επισκόπηση

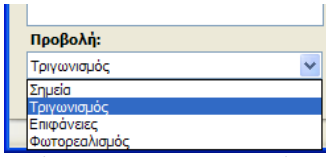
Έχοντας καταχωρήσει τις μετρήσεις, μπορούμε πλέον να δούμε την απεικόνισή τους στην περιοχή απεικόνισης, στο δεξί μέρος του παραθύρου. Για να εμφανιστεί μία μέτρηση στην περιοχή απεικόνιση αρκεί να επιλέξουμε το κουτί-ελέγχου (check-box) που βρίσκεται δίπλα στην ημερομηνία της μέτρησης (στη λίστα αριστερά).



Σχήμα 4.13: Η μέτρηση με ενεργοποιημένη την 3Δ απεικόνισή της

Αν “τσεκάρουμε” τη μέτρηση που εισάγαμε προηγουμένως, θα εμφανιστεί στο παράθυρο μία εικόνα σαν αυτήν στο Σχήμα 4.13. Η περιοχή απεικόνισης έχει παράξει ένα τρισδιάστατο

μοντέλο για το ανάγλυφο της περιοχής και το απεικονίζει υπό τη μορφή *τριγωνισμού*. Μπορούμε να "περιηγηθούμε" στον εικονικό χώρο κρατώντας πατημένο το αριστερό-κλικ του ποντικιού και κουνώντας το για να στραφούμε γύρω από το μοντέλο. Για να κινηθούμε στο χώρο, κρατάμε το δεξί-κλικ το ποντικιού και το κινούμε προς την κατεύθυνση που θέλουμε να "μεταφέρουμε" το μοντέλο (pan). Τέλος, μπορούμε να εστιάσουμε ή να απομακρυνθούμε από τις μετρήσεις με χρήση της ροδέλας του ποντικιού (προς τα πάνω, για να πλησιάσουμε το μοντέλο, προς τα κάτω για να απομακρυνθούμε).

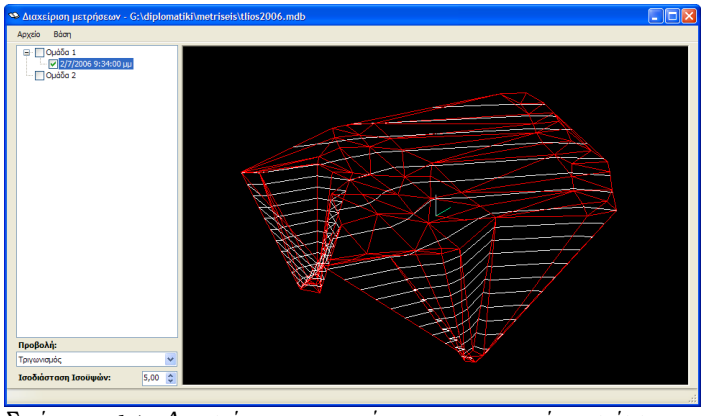


Σχήμα 4.14: Η λίστα επιλογών απεικόνισης

Εκτός από την αλλαγή της θέσης προβολής της απεικόνισης, μπορούμε να επιλέξουμε ανάμεσα σε διαφορετικά είδη προβολής. Αυτό γίνεται από το πλαίσιο-λίστας (drop-down list) στο αριστερό-κάτω μέρος της φόρμας (Σχήμα 4.14). Οι επιλογές είναι *Σημεία*, *Τριγωνισμός*, *Επιφάνειες* και *Φωτορεαλισμός*.

Η κάθε μία προσφέρει τα πλεονεκτήματά της κατά την επισκόπηση των εργασιών. Η προβολή *Σημείων* βοηθάει κατά την εμφάνιση των ισοϋψών σε συνδυασμένες μετρήσεις, καθώς παρουσιάζει μία πιο "καθαρή" απεικόνιση. Ο *Τριγωνισμός* ενδείκνυται περισσότερο για να γίνει αντιληπτό το ανάγλυφο της αποτυπωμένης περιοχής. Η επιλογή *Επιφάνειες* μπορεί να βοηθήσει στις συγκρίσεις μεταξύ επιφανειών, ιδίως ότι προσπαθούμε να δούμε λεπτομερώς κάποια σημεία μετρήσεων. Τέλος, η επιλογή *Φωτορεαλισμός* δίνει μία εμφάνιση κοντά στην πραγματικότητα, προσπαθώντας να δημιουργήσει την ψευδαίσθηση ενός πραγματικού αναγλύφου με στοιχειώδεις υφές και φωτισμό.

Δύο χρήσιμα χαρακτηριστικά της επισκόπησης του αναγλύφου είναι η χρήση χρωμάτων για την κάθε μέτρηση και η εμφάνιση των ισοϋψών. Αυτό γίνεται μέσω του μενού-ελέγχου μετρήσεων (με δεξί-κλικ στη σχετική μέτρηση) και κάνοντας κλικ στις αντίστοιχες επιλογές *Αλλαγή χρώματος*... και *Ισοϋψείς*. Στην 1η περίπτωσης, πατώντας τη σχετική επιλογή, διαλέγουμε από ένα πλήθος χρωμάτων μέσα από ένα παράθυρο με διαφορετικές αποχρώσεις. Στη 2η, κάνοντας κλικ στην επιλογή *Ισοϋψείς* θα δούμε να απεικονίζονται οι σχετικές υψομετρικές καμπύλες. Είναι χρήσιμο αυτά τα δύο χαρακτηριστικά να χρησιμοποιούνται παράλληλα, καθώς όταν οι γραμμές του τριγωνισμού και οι ισοϋψείς έχουν το ίδιο χρώμα, είναι δύσκολο να γίνει διάκριση μεταξύ των δύο. Στο Σχήμα 4.15 βλέπουμε μία απεικόνιση με χρώμα και ισοϋψείς.

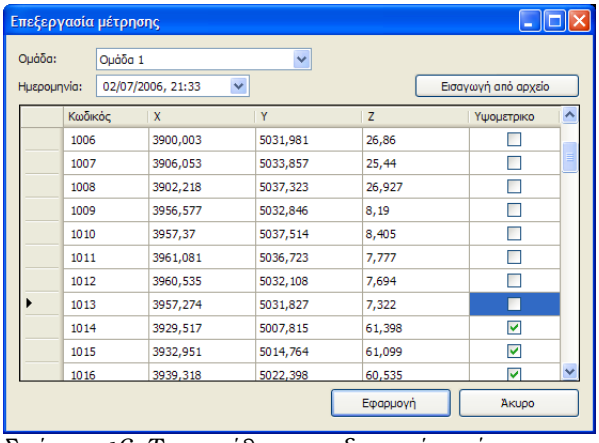


Σχήμα 4.15: Απεικόνιση της μέτρησης με χρήση κόκκινου χρώματος και εμφάνιση των ισοϋψών

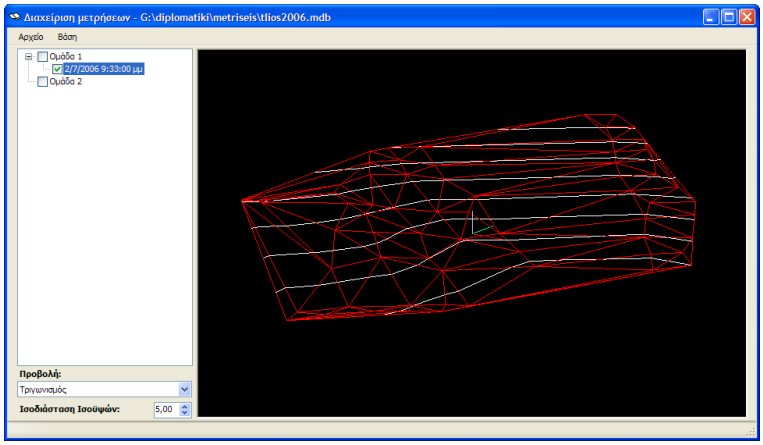
Βήμα 4ο: Επεξεργασία μετρήσεων

Η μέτρηση που εισάγαμε στο 2ο βήμα της διαδικασίας εμφανίζει στα Σχήματα 4.13 και 4.15 κάποιες προβληματικές περιοχές. Αυτό έγινε διότι δεν ορίσαμε σωστά κάποια μη-υψομετρικά σημεία, με αποτέλεσμα να γίνεται λανθασμένος τριγωνισμός και απόδοση ισοϋψών. Προκειμένου να διορθωθεί αυτό, πρέπει να ορίσουμε ποια σημεία δεν είναι υψομετρικά.

Μέσω του μενού-ελέγχου μετρήσεων (δεξί-κλικ στη μέτρηση) επιλέγουμε *Επεξεργασία…* για να εμφανιστεί το παράθυρο *Επεξεργασία μέτρησης* που είναι παρόμοιο με το αντίστοιχο για την εισαγωγή μίας καινούργιας. Οι λειτουργίες είναι παρεμφερείς και συνεπώς δε χρειάζεται να περιγραφούν από την αρχή. Έτσι, απλώς ανατρέχουμε στη λίστα σημείων για να διορθώσουμε εκείνα των οποίων το υψόμετρο δεν πρέπει να λαμβάνεται υπόψη (Σχήμα 4.16).



Σχήμα 4.16: Το παράθυρο επεξεργασίας μέτρησης με διορθωμένα τα μη-υψομετρικά σημεία.



Σχήμα 4.17: Η διορθωμένη επιφάνεια με σωστό ανάγλυφο και ισοϋψείς

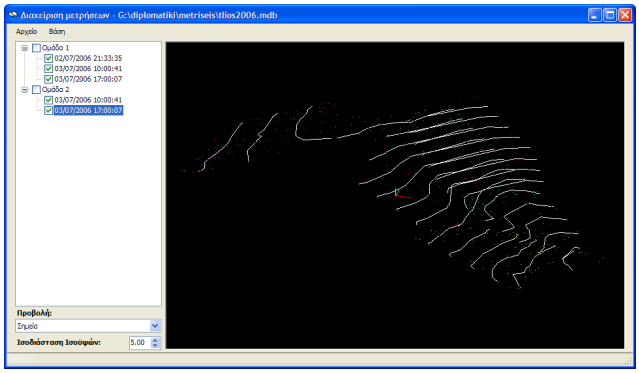
Πατώντας το κουμπί *Εφαρμογή*, οι αλλαγές αποθηκεύονται στη βάση και φορτώνονται τα διορθωμένα δεδομένα για να εμφανιστεί η σωστή απεικόνιση της μέτρησης. Όπως βλέπουμε και στο Σχήμα 4.17, η επιφάνεια έχει “απαλλαγεί” από τα λανθασμένα σημεία και παρουσιάζει ένα σωστό ανάγλυφο που ομοιάζει περισσότερο με το πραγματικό για την αντίστοιχη περιοχή.

Βήμα 5ο: Απεικόνιση και διαχείριση πολλαπλών μετρήσεων

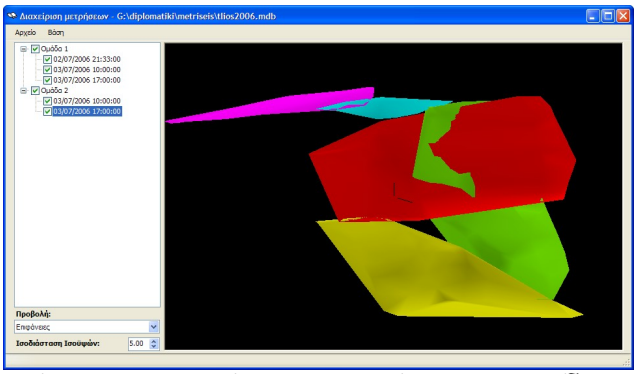
Η απεικόνιση μία μέτρηση, όπως είδαμε στο *3ο βήμα* είναι αρκετά χρήσιμη για να έχουμε μία πρώτη εικόνα για το αποτέλεσμα της αποτύπωσης στο πεδίο. Το ζητούμενο, όμως, είναι να μπορέσουμε να συνδυάσουμε τις μετρήσεις στην ίδια προβολή και να εντοπίσουμε ενδεχόμενες μη-αποτυπωμένες περιοχές, τη συσχέτιση των γειτονικών και επικαλυπτώμενων μετρήσεων και, γενικώς, την εξέλιξη της συνολικής αποτύπωσης ως

αποτέλεσμα του συνδυασμού των επιμέρους παρατηρήσεων.

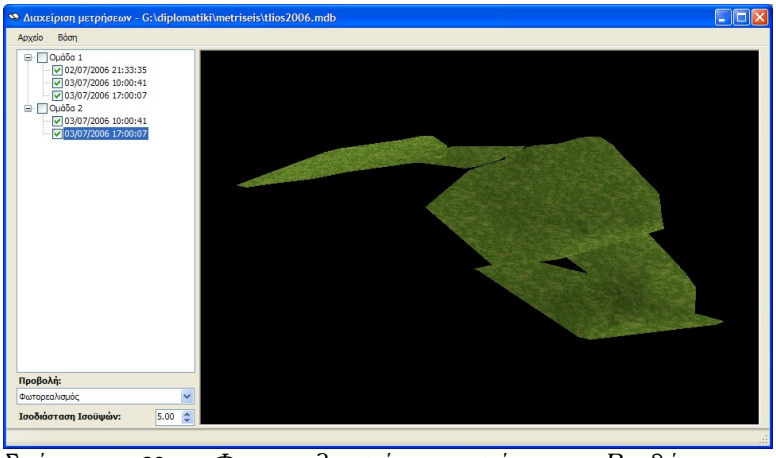
Έτσι, το σημαντικό είναι να χρησιμοποιήσουμε τις δυνατότητες απεικόνισης πολλαπλών μετρήσεων, με χρώματα, για να εντοπίσουμε προβληματικά σημεία ή για να αποφασίσουμε την επόμενη μέτρηση που θα εκτελεστεί. Αφού εισάγουμε πολλές μετρήσεις ταυτόχρονα, με τον τρόπο που παρουσιάστηκε στο 2^{ο} βήμα, και τις παραμετροποιήσουμε κατάλληλα για να είναι σωστή η απεικόνιση, έχουμε ένα ικανό υπόβαθρο για να κάνουμε παρατηρήσεις επί των αποτυπωμένων περιοχών.



Σχήμα 4.18: Χρήση απεικόνισης σημείων με ισοϋψείς. Μας δίνει τη δυνατότητα να εντοπίσουμε ασυμφωνίες στις ισοϋψεις επικαλυπτόμενων ή γειτονικών περιοχών



Σχήμα 4.19: Απεικόνιση με επιφάνειες. Εμφανίζονται καλύτερες οι τεμνόμενες επικαλυπτόμενες ή γειτονικές μετρήσεις, αναδεικνύοντας τα προβλήματα "ασυμφωνίας" υψομέτρων. Παρουσιάζονται καλύτερα και οι "κενές" περιοχές που δεν αποτυπώθηκαν.



Σχήμα 4.20: Φωτορεαλιστική απεικόνιση. Βοηθά στην "αναπαραγωγή" της πραγματικής εικόνας του αναγλύφου και βοηθά στην διαμόρφωση άποψης για την περιοχή (χρήσιμο για να αποφαίσουμε την επόμενη μέτρηση).

Στα σχήματα 4.18, 4.19 και 4.20 παρουσιάζονται συνδυασμένες μετρήσεις απεικονισμένες με διαφορετικούς τύπους και χρώματα, που μας βοηθούν να παράγουμε χρήσιμα συμπεράσματα για την περιοχή, να αξιολογήσουμε την ποιότητα της αποτύπωσης και να πάρουμε αποφάσεις για την εξέλιξη των εργασιών.

Βήμα 6ο: Εξαγωγή υποβάθρου

Όλα τα παραπάνω είναι χρήσιμα για να έχει ο υπεύθυνος την εικόνα της περιοχής και να μπορεί να αξιολογήσει το αποτέλεσμα. Όμως, σημαντικό είναι το εργαλείο να μπορεί να χρησιμοποιηθεί ως μέρος της παραγωγικής διαδικασίας του τελικού Τοπογραφικού Διαγράμματος, ώστε η διαχείριση των μετρήσεων και τα προϊόντα επεξεργασίας (τριγωνισμός, ισοϋψείς) να είναι εκμεταλλεύσιμα στην τελική σχεδίαση.

Για το σκοπό αυτό, το πρόγραμμα παρέχει τη δυνατότητα εξαγωγής του παρουσιαζόμενου αποτελέσματος σε μορφή *DXF*, που μπορεί να χρησιμοποιηθεί από όλα, σχεδόν, τα προγράμματα *CAD* ή *GIS*. Με αυτό τον τρόπο, ο χρήστης μπορεί να αποφασίσει ποιες μετρήσεις να συμπεριλάβει και ποιες να αποκλείσει στο τελικό σχέδιο, καθώς και να αξιοποιήσει τον τριγωνισμό και τις ισοϋψείς που υπολογίστηκαν αυτόματα από το πρόγραμμα. Έτσι, θα έχει ένα πλήρες υπόβαθρο που διευκολύνει την τελική σχεδίαση του διαγράμματος.

Για να γίνει αυτό, πρέπει να επιλέξουμε τις μετρήσεις που θέλουμε να εξάγουμε. Εμφανίζουμε, λοιπόν, στην απεικόνιση μόνο τις μετρήσεις εκείνες που θέλουμε να συμπεριληφθούν στο τελικό υπόβαθρο. Εν συνεχεία, επιλέγουμε από το κεντρικό μενού *Αρχείο→Εξαγωγή…* . Το πρόγραμμα ζητά ένα όνομα για το *DXF* αρχείου και τον κατάλογο προορισμού. Αφού οριστούν αυτά, θα παραχθεί το τελικό *DXF*, περιέχοντας τις ενεργές μετρήσεις, κάθε μία σε δικό της *επίπεδο* (*layer*) και με αντίστοιχα *επίπεδα* (*layers*) τριγωνισμού και ισοϋψών για κάθε μέτρηση.

5. Συμπεράσματα

Η ανάπτυξη της εφαρμογής αυτής ήταν μια παραγωγική εργασία με σκοπό να ικανοποιήσει κάποιες από τις σύγχρονες απαιτήσεις της διαχείρισης εργασιών με μεγάλο όγκο δεδομένων. Εκτός, όμως, από μία διαδικασία με αμιγώς δημιουργικό χαρακτήρα, η εκπόνηση αυτής της Διπλωματικής Εργασίας συνέβαλε στη "ζύμωση" της αρχικής ιδέας σε ένα πρακτικό συμπέρασμα αναφορικά με την προσέγγιση των μετρήσεων από τη σκοπιά της ηλεκτρονικής διαχείρισης.

Τα ζητήματα που ανέκυψαν ήταν πολλά: ποια και πόσα από τα στοιχεία των μετρήσεων θα πρέπει να καταχωρηθούν, σε ποιο βάθος γίνεται η καταγραφή των εργασιών και πόσα πράγματα μπορούν να προκύψουν από τη χρήση των ηλεκτρονικών υπολογιστών ως εργαλείο κεντρική διαχείρισης και ελέγχου εκτέλεσης τέτοιων μετρήσεων.

Το συμπέρασμα που προέκυψε είναι ότι οι δυνατότητες στην οργάνωση της πληροφορίας είναι απεριόριστες και ο μόνος παράγοντας που μπορεί να καθορίσει το βάθος του αντικειμένου αυτού είναι η δημιουργικότητα όσων ασχοληθούν με τη διαχείριση μετρήσεων μέσα από ένα τέτοιο περιβάλλον. Αυτό δεν προκύπτει ως μία αυθαίρετη σκέψη, αλλά είναι κάτι που αναδεικνύεται από τη διείσδυση της τεχνολογίας στη διαδικασία εργασιών αποτύπωσης που, ενδεχομένως χωρίς να γίνεται αντιληπτό από τον Τοπογράφο Μηχανικό, αποτελούν έτσι κι αλλιώς μια αντίστοιχη διαδικασία διαχείρισης των μετρήσεων. Επί παραδείγματι, η διατήρηση των ξεχωριστών αρχείων μετρήσεων που έχουν προκύψει από μετρήσεις σε μία απλή διαρθρωτική δομή φακέλων στο σκληρό δίσκο του υπολογιστή, όπως συνηθίζεται, είναι μια πράξη ηλεκτρονικής κατηγοριοποίησης και οργάνωσης της πληροφορίας που προέκυψε στο πεδίο.

Έτσι, υπάρχουν οι βάσεις για την ανάπτυξη ενός ενιαίου πλαισίου που θα μετουσιώνει τη διαδικασία αυτή σε ένα πλήρες περιβάλλον διαχείρισης και ελέγχου των μετρήσεων που, μελλοντικά, είναι ικανό να καταλήξει σε μία αυτοματοποίηση των καθημερινών ενεργειών ρουτίνας του Τοπογράφου. Αυτό, ασφαλώς, δεν έχει ως σκοπό να μειώσει τη χρησιμότητα του Τοπογράφου Μηχανικού, ούτε να ακυρώσει το ρόλο του. Αντιθέτως, αποσκοπεί στο να αναδείξει τον πραγματικό σκοπό, που είναι η λήψη αποφάσεων τόσο στο πεδίο (πως θα στηθεί η διαδικασία των μετρήσεων, πως θα υλοποιηθεί η βασική όδευση, ποια σημεία θα αποτυπωθούν κλπ.) όσο και στο γραφείο (ποιες μετρήσεις είναι ικανά ακριβείς για την αποτύπωση, ποια σημεία ενδεχομένως έχουν αποτυπωθεί λάθος), απαλλάσσοντας τον από τις διαδικασίες εκείνες που δεν ανήκουν στο επιστημονικό πεδίο του (μεταφορά πληροφορίας, αποθήκευση, οργάνωση κλπ.).

Ένα δεύτερο σημαντικό συμπέρασμα που προέκυψε κατά την ανάπτυξη αυτής της εργασίας είναι η ομοιότητα του αντικειμένου απεικόνισης τρισδιάστατων γραφικών (3D graphics) με την τοπογραφική επιστήμη. Δεδομένου ότι και στις δύο περιπτώσεις, η φιλοσοφία αφορά την διαχείριση σημείων σε ένα δεδομένο σύστημα συντεταγμένων (με τις όποιες διαφορές ενδεχομένως να υπάρχουν ανάμεσα στις δύο προσεγγίσεις), τη διαδικασία προβολής από τα στοιχεία αυτά στο επίπεδο (είτε στην οθόνη ενός υπολογιστή, είτε στο χάρτη μίας αποτύπωσης υπό τη μορφή διαγράμματος ή διατομής) είναι προφανές ότι και τα δύο αυτά πεδία μπορούν να συνυπάρξουν και να αναπτυχθούν περαιτέρω με έναν παράλληλο τρόπο. Και είναι, πράγματι, άξιο απορίας το γιατί οι τεχνολογία των τρισδιάστατων γραφικών και του επιστημονικού πεδίου του φωτορεαλισμού (virtualisation) δεν έχουν, ως τώρα, διεισδύσει σε ιδιαίτερο βαθμό στην καθημερινότητα του Τοπογράφου. Άλλωστε, με την κατάλληλη διαμόρφωση ενός περιβάλλοντος, το να προκύπτει μία πλήρης τρισδιάστατη απεικόνιση που θα αναπαριστά την περιοχή που αποτυπώθηκε (όχι μόνο όσον αφορά το ανάγλυφο, αλλά και τα αντικείμενα όπως κτίσματα, τοιχία, φράκτες κλπ.) είναι μία υπαρκτή δυνατότητα που δεν απαιτεί καμία “θυσία” από την πλευρά της προσωπικής εργασίας του μηχανικού. Αρκεί να υπάρχει ένα κατάλληλο δομημένο περιβάλλον που θα αναλάβει να αξιοποιεί την ήδη υπάρχουσα πληροφορία, για να υπάρξει αυτή η χρήσιμη προέκταση της απεικόνισης τοπογραφικών μετρήσεων.

6. Προτάσεις

6.1 Προτάσεις για εξέλιξη της εφαρμογής

Η εφαρμογή είχε σκοπό να καλύψει βασικές ανάγκες της διαχείρισης των εργασιών για το μάθημα *Μεγάλες Γεωδαιτικές Ασκήσεις ΙΙ*, αλλά και να αποτελέσει ένα υπόβαθρο για όποιον ενδιαφερόμενο επιθυμεί να εξελίξει αυτό το πεδίο. Ως εκ τούτου, ενθαρρύνεται η χρήση του υπάρχοντα κώδικα και του προγραμματιστικού πλαισίου που αναπτύχθηκε (δομές, βιβλιοθήκες, αλγόριθμοι) για άλλες εφαρμογές ή για βελτιστοποίηση της ήδη υπάρχουσας.

Ακολουθεί μία σειρά προτάσεων για εξέλιξη.

Απεικόνιση χαρακτηριστικών σημείων (κτίσματα, τοιχία κλπ.)

Ένα αντικείμενο που δεν καλύφθηκε από την εφαρμογή είναι η απεικόνιση των σημείων που δεν αναπαριστούν υψόμετρο. Είναι δυνατό, όμως, να μπορέσει να γίνει μία πληρέστερη απεικόνιση της αποτυπωμένης περιοχής που, εκτός από το ανάγλυφος της επιφάνειας, θα παρουσιάζει και τα αντικείμενα που αποτυπώθηκαν στο χώρο. Κάτι τέτοιο είναι δυνατό να επιτευχθεί με την περιγραφή του είδους σημείου (υψομετρικό, δρόμος, ρέμα, κτίσμα, τοιχίο, ξερολιθιά, φράκτης κλπ.) παράλληλα με την μία βιβλιοθήκη 3D αντικειμένων (3D objects) που θα υλοποιεί την απεικόνισής τους. Για παράδειγμα, όταν 1 σημείο έχει το χαρακτηρισμό “οδικό σήμα” θα εμφανίζεται εκεί το αντίστοιχο αντικείμενο της βιβλιοθήκη που αναπαριστά ένα σήμα οδικής κυκλοφορίας. Εδώ, βεβαίως, ανακύπτουν και ζητήματα όπως η διαχείριση, εκτός από σημειακών αντικειμένων (όπως το οδικό σήμα που παρουσιάστηκε στο παράδειγμα), των γραμμικών αντικειμένων (δρόμος, ποταμός, φράκτης κλπ.) ή των επιφανειακών (κτίσμα, λίμνη κλπ.). Αυτό μπορεί να αντιμετωπιστεί με χρήση των αντίστοιχων θεωριών διαχείρισης αντικειμένων στο χώρο που έχουν ήδη αναπτυχθεί από το επιστημονικό πεδίο των Γεωγραφικών Συστημάτων Πληροφοριών (G.I.S.).

Ηλεκτρονικό κροκί

Μία ενδιαφέρουσα μετεξέλιξη της διαδικασίας μέτρησης θα είναι η χρήση της τεχνολογίας για την απόδοση της ποιοτικής πληροφορίας στα αποτυπωμένα σημεία. Αυτό σημαίνει ότι με τις σύγχρονες τάσεις τις τεχνολογίας (touch-screens, tablet-PCs κλπ.) διατίθενται οι δυνατότητες να χρησιμοποιηθούν αυτά για την αυτοματοποίηση της ανάθεσης χαρακτηριστικών στα σημεία και την εξέλιξη του *κροκί*, που είναι ένα βασικό και απαραίτητο συστατικό της διαδικασίας μέτρησης στο πεδίο, στην ηλεκτρονική του μορφή.

Η ιδέα αυτή είναι σχετικά απλή και αρκετά προσιτή. Αντί για το σχεδιασμό του *κροκί* στο χαρτί και την μετέπειτα καταχώρηση του χαρακτηρισμού σημείων (και άλλων πληροφοριών) με το χέρι στον υπολογιστή που γίνεται η τελική αποτύπωση, είναι μπορεί να χρησιμοποιηθεί ένας υπολογιστή χειρός (είτε *PDA*, είτε *Tablet-PC)* με οθόνη αφής (*touch-screen*) και μία γραφίδα ώστε ο Τοπογράφος να δηλώνει εκεί, σε ένα κατάλληλα διαμορφωμένο σχεδιαστικό περιβάλλον, το σημείο που μετριέται με το όργανο αποδίδοντάς του, επί τόπου, το χαρακτηρισμό του σημείου. Έτσι, η πληροφορία θα καταχωρείται στο σύστημα αυτό και, κατόπιν, θα μεταφέρεται στον υπολογιστή παράλληλα με τις μετρήσεις από το όργανο αποτύπωσης (γεωδαιτικός σταθμός).

Η υλοποίηση ενός τέτοιου συστήματος δεν είναι ιδιαίτερα δύσκολη. Τα συστήματα αυτά περιέχουν πλήρη προγραμματιστικά περιβάλλοντα (*Software Development Kits*) που καθιστούν τη σχεδίαση ένα αρκετά προσιτό προγραμματιστικό στόχο, ενώ οι τεχνολογίες μεταφορές (USB, Bluetooth, GPRS, 3G HSDP κλπ.) είναι επίσης εύκολα προσβάσιμες από προγραμματιστικής άποψης.

Τα οφέλη από την ανάπτυξη ενός τέτοιου συστήματος στην παραγωγικότητα είναι προφανή και απεριόριστα. Εκτός την αυτοματοποίηση εργασιών όπως το ραπορτάρισμα σημείων και ο σχεδιασμός των αντικειμένων σε αυτά, με χρήση τεχνολογιών μεταφοράς δεδομένων σε πραγματικό χρόνο (*real-time transmission*) μπορεί το *κροκί* να "μετατρέπεται" σε επιτόπου σχεδιαστική αποτύπωση, ή τα δεδομένα να μεταφέροντα μέσω GPRS σε έναν κεντρικό εξυπηρετητή (*server*) που θα προβάλει το αποτέλεσμα σε πραγματικό χρόνο στον υπεύθυνο. Έτσι θα είναι δυνατή η επισκόπηση του αποτελέσματος σε πραγματικό χρόνο και θα είναι δυνατό να αντιμετωπιστούν προβλήματα όπως προβληματικές μετρήσεις, μη αποτυπωμένες μετρήσεις κ.α., τη στιγμή που το συνεργείο βρίσκεται ακόμα στην περιοχή.

6.2 Προτάσεις προς την επιστημονική κοινότητα

Όπως διατυπώθηκε ήδη από την ενότητα *"Συμπεράσματα"* οι δυνατότητες που δίνονται από τη χρήση των ηλεκτρονικών υπολογιστών για τη διαχείριση της πληροφορίας και των τρισδιάστατων γραφικών για την αξιοποίησή τους στην απεικόνιση, ανακύπτει μία ανάγκη να διαμορφωθεί ένα ενιαίο, προτυποποιημένο πλαίσιο που θα χρησιμεύσει ως οδηγός για την ανάπτυξη του πεδίου αυτού. Αυτό σημαίνει ότι μία συνεργασία μεταξύ της επιστήμης των Υπολογιστών και της Τοπογραφίας μπορεί να καταλήξει σε ένα δεδομένο κοινό κώδικα δεοντολογίας που θα καθορίζει:

1. τα **πρωτόκολλα δεδομένων** που μπορούν να αξιοποιηθούν στη διαδικασία μεταφορά πληροφορίας αποτύπωσης μεταξύ των διάφορων ηλεκτρονικών συσκευών (πχ., γεωδαιτικός σταθμός → υπολογιστής, γεωδαιτικό GPS → υπολογιστής, γεωδαιτικός σταθμός → γεωδαιτικό GPS)

2. τις **τεχνολογίες μεταφοράς** που μπορούν να ενσωματωθούν στα εργαλεία αποτύπωσης, ή χρησιμοποιούνται ήδη αλλά δεν αξιοποιούνται με τον αντίστοιχο τρόπο (πχ., bluetooth, GPRS)

3. το **περιβάλλον κανόνων** που θα καθορίζει μία συγκεκριμένη δομή και ένα δεδομένο τρόπο προσέγγισης των προγραμματισμών όταν επιχειρούν να ασχοληθούν με το αντικείμενο της διαχείρισης μετρήσεων, της αυτοματοποίησης των διαδικασιών που εκτελούνται σε αυτές και της απεικόνισης τους.

Όλα τα παραπάνω, ασφαλώς, χρήζουν εκτενούς μελέτης από την επιστημονική κοινότητα και, κυρίως, από τους ενδιαφερόμενους τομείς που μπορούν να συνεργαστούν για να εκμεταλλευτούν τις δυνατότητες που τους δίνονται να προσδιορίσουν ένα αντικείμενο που, μέχρι σήμερα, αποτελεί χώρο δράσης κυρίως εμπορικών προγραμμάτων, με ανεξάρτητες και ακαθόριστες διαφορετικές προσεγγίσεις που επικαλύπτονται και δεν εξελίσσουν εποικοδομητικά αυτό το αντικείμενο.

Βιβλιογραφία

- *Παγάνης Κ.Π.*, ***Σημειώσεις Τοπογραφικού Σχεδίου***, *Εκδόσεις Ε.Μ.Π., 2003*
- *Βέης Γ. - Μπιλλήρης Χ. - Παπαζήση Κ.*, ***Κεφάλαια Ανώτερης Γεωδαισίας***, *Εκδόσεις Ε.Μ.Π, 2005*
- *Λάμπρου Ε. - Πανταζής Γ. - Μπίθας Α. - Σιούλης Α. - Αγατζά Α. - Μπαλοδήμου*, ***Ειδικά Θέματα Γεωδιασίας***, *Εκδόσεις Ε.Μ.Π., 2000*
- *Κουτσόπουλος Κ.*, ***Γεωγραφικά Συστήματα Πληροφοριών και Ανάλυση Χώρου***, *Εκδόσεις Παπασωτηρίου, 2005*
- *Elmarsi R. - Navathe S.B.*, ***Θεμελιώδεις Αρχές Συστημάτων Βάσεων Δεδομένων***, *Δίαυλος, 2000*
- *Weibel, R. - M. Heller* ***"A framework for digital terrain modelling"***. *4th International Symposium on Spatial Data Handling, 1990*
- *Harrison L. T.*, ***Introduction to Game Engine Design Using DirectX and C#***, *Apress, 2003*
- *Deloura M. - Boer M. - Budge B. Christenses C. - Dalton P.*, ***Best of Game Programming Gems***, *Course Technology Cengage Learning, 2008*
- *Riemer Grootjans*, ***Direct X using C#: 3D Series***, *http://www.riemers.net, 2003-2008*
- *Nielsen Morten*, ***Delaunay Triangulation in .NET 2.0***, *9 Μαρτίου 2006* *http://www.sharpgis.net/post/2006/03/09/Delaunay-Triangulation-in-NET-20.aspx*
- *Radoi Cosmin*, ***DXF Library for C#: Sources***, *http://github.com/cos/dxflibrary, 2008-2009*
- *Αναγνώστου Κ.*, ***Εισαγωγή στη C# .NET***, *8 Ιανουαρίου 2009,* *http://videogameslab.wordpress.com/2009/01/08/intro-csharp-net/*

ΠΑΡΑΡΤΗΜΑ: Πηγαίος Κώδικας

Program.cs

```
1   using System;
2   using System.Collections.Generic;
3   using System.Windows.Forms;
4
5   namespace diplomatiki
6   {
7       static class Program
8       {
9           /// <summary>
10          /// The main entry point for the application.
11          /// </summary>
12          [STAThread]
13          static void Main()
14          {
15              Application.EnableVisualStyles();
16              Application.SetCompatibleTextRenderingDefault(false);
17              Application.Run(new mainForm());
18          }
19      }
20  }
```

mainForm.cs

```
1   using System;
2   using System.Collections.Generic;
3   using System.ComponentModel;
4   using System.Data;
5   using System.Data.OleDb;
6   using System.Drawing;
7   using System.Text;
8   using System.Windows.Forms;
9   using System.IO;
10  using Microsoft.DirectX;
11  using Microsoft.DirectX.Direct3D;
12  using Microsoft.DirectX.DirectInput;
13
14  namespace diplomatiki
15  {
16      public struct geoPoint
17      {
18          public string PID, Pname;
19          public float X, Y, Z;
20          public bool isHeight;
21
22          public geoPoint(float xx, float yy, float zz)
23          {
24              PID = "00";
25              Pname = "00";
26              X = xx;
27              Y = yy;
28              Z = zz;
29              isHeight = true;
30          }
31      }
32
33      public struct geoContour
34      {
35          public CustomVertex.PositionColored[] vertices;
36          public VertexBuffer vb;
37
38          public geoContour(geoPoint firstPoint, geoPoint secondPoint,
        Microsoft.DirectX.Direct3D.Device mydevice)
39          {
40              vertices = new CustomVertex.PositionColored[2];
```

```
41              vertices[0] = new CustomVertex.PositionColored(firstPoint.X, firstPoint.Y,
        firstPoint.Z, Color.White.ToArgb());
42              vertices[1] = new CustomVertex.PositionColored(secondPoint.X, secondPoint.Y,
        secondPoint.Z, Color.White.ToArgb());
43              vb = new VertexBuffer(typeof(CustomVertex.PositionColored), 2, mydevice,
        Usage.Dynamic | Usage.WriteOnly, CustomVertex.PositionColored.Format, Pool.Default);
44              vb.SetData(vertices, 0, LockFlags.None);
45          }
46      }
47
48      public struct geoMeasure
49      {
50          public geoPoint Trans;
51          public geoPoint[] Points;
52          public int measID, teamID;
53          public DateTime measDate;
54          public bool visible, hascontour, showcontour;
55          public Color lineColor;
56          public CustomVertex.PositionColoredTextured[] vertices;
57          public short[] indices;
58          public geoContour[] contours;
59          public Mesh geoMesh;
60      }
61
62      public struct geoStudent
63      {
64          public string FirstName, LastName, Kwdikos;
65          public int PeopleID, TID;
66      }
67
68      public partial class mainForm : Form
69      {
70          private Microsoft.DirectX.Direct3D.Device device;
71          private Microsoft.DirectX.DirectInput.Device dmouse;
72          private float hangle = 0f, vangle = 0.5f, czoom = 50f;
73          private Vector3 ctarget = new Vector3(0, 0, 0);
74          public OleDbConnection mdbconnection;
75          private geoMeasure[] measures;
76          private TreeNode sourcenode;
77          private CustomVertex.PositionColored[] cvert = new CustomVertex.PositionColored[6];
78          private VertexBuffer cvb;
79          private bool EngineUp = false, inView = false;
80          private int EngineState = 1; // 0: Σήμεία, 1: Τριγωνισμός, 2: Φωτορεαλισμός
81          private Texture texture;
82          private Material material;
83
84          public mainForm()
85          {
86              InitializeComponent();
87
88              cmbViewType.SelectedIndex = 1;
89
90              measures = new geoMeasure[0];
91          }
92
93          // ---- ΣΥΝΑΡΤΗΣΕΙΣ ΜΗΧΑΝΗΣ ΓΡΑΦΙΚΩΝ ----
94          // Εκκίνηση μηχανής γραφικών
95          public void InitializeDevice()
96          {
97              dmouse = new Microsoft.DirectX.DirectInput.Device(SystemGuid.Mouse);
98              dmouse.Acquire();
99
100             PresentParameters presentParams = new PresentParameters();
101             presentParams.Windowed = true;
102             presentParams.SwapEffect = SwapEffect.Discard;
103             presentParams.AutoDepthStencilFormat = DepthFormat.D16;
104             presentParams.EnableAutoDepthStencil = true;
105
106             device = new Microsoft.DirectX.Direct3D.Device(0,
        Microsoft.DirectX.Direct3D.DeviceType.Hardware, this.pictureBox1,
        CreateFlags.SoftwareVertexProcessing, presentParams);
107             device.RenderState.CullMode = Cull.None;
108             device.RenderState.FillMode = FillMode.WireFrame;
```

```
109
110             LoadTexturesAndMaterials();
111         }
112
113         // Φόρτωση αρχικών υλικών και υφών
114         private void LoadTexturesAndMaterials()
115         {
116             material = new Material();
117
118             material.Diffuse = Color.White;
119             material.Specular = Color.LightGray;
120             material.SpecularSharpness = 15.0F;
121
122             device.Material = material;
123
124             texture = TextureLoader.FromFile(device,
        Path.GetDirectoryName(Application.ExecutablePath) + "\\grass.jpg");
125         }
126
127         // Εκκίνηση μηχανής γραφικών
128         private void SetAxis()
129         {
130             cvert[0] = new CustomVertex.PositionColored(ctarget.X, ctarget.Y, ctarget.Z,
        Color.LightBlue.ToArgb());
131             cvert[1] = new CustomVertex.PositionColored(ctarget.X, ctarget.Y, ctarget.Z + 10,
        Color.LightBlue.ToArgb());
132             cvert[2] = new CustomVertex.PositionColored(ctarget.X, ctarget.Y, ctarget.Z,
        Color.Red.ToArgb());
133             cvert[3] = new CustomVertex.PositionColored(ctarget.X + 10, ctarget.Y, ctarget.Z,
        Color.Red.ToArgb());
134             cvert[4] = new CustomVertex.PositionColored(ctarget.X, ctarget.Y, ctarget.Z,
        Color.MediumSpringGreen.ToArgb());
135             cvert[5] = new CustomVertex.PositionColored(ctarget.X, ctarget.Y - 10, ctarget.Z,
        Color.MediumSpringGreen.ToArgb());
136             cvb = new VertexBuffer(typeof(CustomVertex.PositionColored), 6, device,
        Usage.Dynamic | Usage.WriteOnly, CustomVertex.PositionColored.Format, Pool.Default);
137             cvb.SetData(cvert, 0, LockFlags.None);
138         }
139
140         // Ανάγνωση δεδομένων εισόδου (κίνηση ποντικιού)
141         private void ReadInput()
142         {
143             MouseState mstate = dmouse.CurrentMouseState;
144
145             if (!inView)
146                 return;
147
148             if (mstate.GetMouseButtons()[0] != 0)
149             {
150                 hangle -= (float)mstate.X / 50f;
151                 vangle += (float)mstate.Y / 50f;
152             }
153             else if (mstate.GetMouseButtons()[1] != 0)
154             {
155                 ctarget.X -= ((float)mstate.Y / 5f) * (float)Math.Sin(hangle) + ((float)mstate.X
        / 5f) * (float)Math.Sin(hangle + Math.PI/2);
156                 ctarget.Y -= ((float)mstate.Y / 5f) * (float)Math.Cos(hangle) + ((float)mstate.X
        / 5f) * (float)Math.Cos(hangle + Math.PI/2);
157             }
158
159             czoom -= (float)mstate.Z / 80f;
160             if (czoom < 10)
161                 czoom = 10;
162         }
163
164         // Τοποθέτηση κάμερας
165         private void CameraPositioning(Vector3 CameraPosition, Vector3 CameraTarget)
166         {
167             device.Transform.Projection = Matrix.PerspectiveFovLH((float)Math.PI / 4f, 1, 1f,
        500f);
168             device.Transform.View = Matrix.LookAtLH(CameraPosition, CameraTarget, new Vector3(0,
        0, 1));
169
```

```
170             SetAxis();
171
172             device.RenderState.CullMode = Cull.None;
173
174             if (EngineState == 0)
175             {
176                 device.RenderState.FillMode = FillMode.Point;
177                 device.RenderState.Lighting = false;
178             }
179             else if (EngineState == 1)
180             {
181                 device.RenderState.FillMode = FillMode.WireFrame;
182                 device.RenderState.Lighting = false;
183             }
184             else
185             {
186                 device.RenderState.FillMode = FillMode.Solid;
187                 device.RenderState.Lighting = true;
188                 device.Lights[0].Type = LightType.Directional;
189                 device.Lights[0].Diffuse = Color.White;
190                 device.Lights[0].Direction = new Vector3(-0.5f, 0, 1f);
191                 device.Lights[0].Enabled = true;
192             }
193         }
194
195         // Τοποθέτηση κάμερας στο κέντρο μίας μέτρησης
196         private void targettomeasure(string MeasureId)
197         {
198             geoPoint measCenter = new geoPoint();
199
200             foreach (geoMeasure meas in measures)
201             {
202                 if(meas.measID == int.Parse(MeasureId))
203                 {
204                     foreach (geoPoint point in meas.Points)
205                     {
206                         measCenter.X += point.X / meas.Points.Length;
207                         measCenter.Y += point.Y / meas.Points.Length;
208                         measCenter.Z += point.Z / meas.Points.Length;
209                     }
210
211                     ctarget.X = measCenter.X;
212                     ctarget.Y = -measCenter.Y;
213                     ctarget.Z = measCenter.Z;
214                 }
215             }
216         }
217
218         // ΚΕΝΤΡΙΚΗ ΕΠΑΝΑΛΗΨΗ ΤΩΝ ΓΡΑΦΙΚΩΝ
219         private void Run()
220         {
221             EngineUp = true;
222
223             while (Created && EngineUp)
224             {
225                 //Διαβάζεται η κίνηση του ποντικιού
226                 ReadInput();
227
228                 //Υπολογίζεται η θέση της κάμερας, με βάση τις πολικές συντεταγμένες
229                 Vector3 cposition;
230                 cposition.X = (float)czoom * (float)Math.Sin(hangle) * (float)Math.Cos(vangle) +
    ctarget.X;
231                 cposition.Y = (float)czoom * (float)Math.Cos(hangle) * (float)Math.Cos(vangle) +
    ctarget.Y;
232                 cposition.Z = (float)czoom * (float)Math.Sin(vangle) + ctarget.Z;
233
234                 //Καλείται η συνάρτηση που εφαρμόζει τις συντεταγμένες κάμερας
235                 CameraPositioning(cposition, ctarget);
236
237                 //Γίνεται η εμφάνιση των γραφικών
238                 Render();
239
240                 Application.DoEvents();
```

```
241             }
242 
243             EngineUp = false;
244         }
245 
246         // ΣΥΝΑΡΤΗΣΗ ΕΜΦΑΝΙΣΗΣ ΓΡΑΦΙΚΩΝ (RENDERING)
247         private void Render()
248         {
249             device.Clear(ClearFlags.Target | ClearFlags.ZBuffer, Color.Black, 1.0f, 0);
250 
251             //---------Draw everything here
252             device.BeginScene();
253 
254             if (EngineState == 3)
255                 device.SetTexture(0, texture);
256             else
257                 device.SetTexture(0, null);
258 
259             foreach (geoMeasure meas in measures)
260             {
261                 if (meas.visible == true)
262                 {
263                     int numSubSets = meas.geoMesh.GetAttributeTable().Length;
264                     for (int i = 0; i < numSubSets; ++i)
265                         meas.geoMesh.DrawSubset(i);
266 
267                     if (meas.hascontour && meas.showcontour)
268                     {
269                         foreach (geoContour contour in meas.contours)
270                         {
271                             device.SetStreamSource(0, contour.vb, 0);
272                             device.DrawUserPrimitives(PrimitiveType.LineList, 1,
        contour.vertices);
273                         }
274                     }
275                 }
276             }
277 
278             device.SetStreamSource(0, cvb, 0);
279             device.DrawUserPrimitives(PrimitiveType.LineList, 3, cvert);
280 
281             device.EndScene();
282             //---------Stop drawing
283 
284             device.Present();
285         }
286 
287         // ---- ΣΥΝΑΡΤΗΣΕΙΣ ΔΙΑΧΕΙΡΙΣΗΣ ΜΕΤΡΗΣΕΩΝ ΣΤΗ ΜΝΗΜΗ ΚΑΙ ΣΤΗΝ ΑΠΕΙΚΟΝΙΣΗ ----
288 
289         // Φόρτωση μιας μέτρησης στη μνήμη
290         private void displayMeasure(string measureID, Color measColor)
291         {
292             for (int i = 0; i < measures.Length; i++)
293             {
294                 if (measures[i].measID == int.Parse(measureID))
295                 {
296                     measures[i].visible = true;
297                     return;
298                 }
299             }
300 
301             Array.Resize<geoMeasure>(ref measures, measures.Length + 1);
302             int j = measures.Length - 1;
303 
304             measures[j] = new geoMeasure();
305             measures[j].measID = int.Parse(measureID);
306             measures[j].Points = new geoPoint[0];
307 
308             OleDbCommand command1 = new OleDbCommand("SELECT PID, PName, X, Y, Z, isHeight FROM
        Points WHERE MID = " + measureID, mdbconnection);
309             OleDbDataReader reader1 = command1.ExecuteReader();
310 
311             while (reader1.Read())
```

```
312             {
313                 Array.Resize(ref measures[j].Points, measures[j].Points.Length + 1);
314                 measures[j].Points[measures[j].Points.Length - 1] = new geoPoint();
315                 measures[j].Points[measures[j].Points.Length - 1].PID =
        reader1.GetInt32(0).ToString();
316                 measures[j].Points[measures[j].Points.Length - 1].Pname =
        reader1.GetValue(1).ToString();
317                 measures[j].Points[measures[j].Points.Length - 1].X = (float)
        (reader1.GetDouble(2) % 10000);
318                 measures[j].Points[measures[j].Points.Length - 1].Y = (float)
        (reader1.GetDouble(3) % 10000);
319                 measures[j].Points[measures[j].Points.Length - 1].Z = (float)
        (reader1.GetDouble(4) % 10000);
320                 measures[j].Points[measures[j].Points.Length - 1].isHeight =
        reader1.GetBoolean(5);
321             }
322
323             reader1.Close();
324             command1 = new OleDbCommand("SELECT DatenTime, TID FROM Measures WHERE MID = " +
        measureID, mdbconnection);
325             reader1 = command1.ExecuteReader();
326             reader1.Read();
327             measures[j].measDate = DateTime.Parse(reader1.GetValue(0).ToString());
328             measures[j].teamID = reader1.GetInt32(1);
329
330             if (measures[j].Points.Length == 0)
331             {
332                 measures[j].visible = false;
333                 return;
334             }
335
336             SetupMeasure(ref measures[j], measColor);
337
338             measures[j].hascontour = false;
339             measures[j].showcontour = false;
340
341             setupCountours(ref measures[j], (float)numIsoDistance.Value);
342
343             measures[j].visible = true;
344
345             return;
346         }
347
348         // Αρχικοποίηση γραφικών μίας μέτρησης
349         private void SetupMeasure(ref geoMeasure meas, Color measColor)
350         {
351             meas.lineColor = measColor;
352
353             meas.vertices = new CustomVertex.PositionColoredTextured[0];
354             for (int i = 0; i < meas.Points.Length; i++)
355             {
356                 if (meas.Points[i].isHeight)
357                 {
358                     Array.Resize(ref meas.vertices, meas.vertices.Length + 1);
359                     meas.vertices[meas.vertices.Length - 1].Position = new
        Vector3(meas.Points[i].X, -meas.Points[i].Y, meas.Points[i].Z);
360                     meas.vertices[meas.vertices.Length - 1].Color = measColor.ToArgb();
361                     meas.vertices[meas.vertices.Length - 1].Tu = meas.Points[i].X / 50.0f;
362                     meas.vertices[meas.vertices.Length - 1].Tv = meas.Points[i].Y / 50.0f;
363                 }
364             }
365
366             List<Triangulator.Geometry.Point> points = new
        List<Triangulator.Geometry.Point>(meas.vertices.Length);
367
368             foreach (CustomVertex.PositionColoredTextured avertex in meas.vertices)
369             {
370                 points.Add(new Triangulator.Geometry.Point((double)avertex.X,
        (double)avertex.Y));
371             }
372
373             if (points.Count > 2)
374             {
```

```
375                 List<Triangulator.Geometry.Triangle> triangles =
        Triangulator.Delauney.Triangulate(points);
376
377                 meas.indices = new short[triangles.Count * 3];
378                 int i = 0;
379
380                 foreach (Triangulator.Geometry.Triangle tri in triangles)
381                 {
382                     meas.indices[i] = (short)tri.p1;
383                     meas.indices[i + 1] = (short)tri.p2;
384                     meas.indices[i + 2] = (short)tri.p3;
385                     i += 3;
386                 }
387             }
388
389             meas.geoMesh = new Mesh(meas.indices.Length / 3, meas.vertices.Length,
        MeshFlags.Managed, CustomVertex.PositionColoredTextured.Format, device);
390
391             meas.geoMesh.SetVertexBufferData(meas.vertices, LockFlags.None);
392             meas.geoMesh.SetIndexBufferData(meas.indices, LockFlags.None);
393
394             int[] adjac = new int[meas.geoMesh.NumberFaces * 3];
395             meas.geoMesh.GenerateAdjacency(0.5f, adjac);
396             meas.geoMesh.OptimizeInPlace(MeshFlags.OptimizeVertexCache, adjac);
397
398             return;
399         }
400
401         // Υπολογισμός και αρχικοποίηση ισοϋψών
402         private void setupCountours(ref geoMeasure meas, float isodistance)
403         {
404             meas.contours = new geoContour[0];
405             CustomVertex.PositionColoredTextured minp, midp, maxp;
406             geoPoint fpoint, spoint;
407             float height;
408
409             // Ξεκινάμε την επανάληψη για κάθε τρίγωνο
410             for (int i = 2; i < meas.indices.Length; i += 3)
411             {
412                 // Υπολογίζουμε το χαμηλότερο και το υψηλότερο σημείο του τριγώνου
413                 minp = meas.vertices[meas.indices[i]];
414                 midp = meas.vertices[meas.indices[i]];
415                 maxp = meas.vertices[meas.indices[i]];
416                 for (int j = i - 2; j < i; j++)
417                 {
418                     if (meas.vertices[meas.indices[j]].Z < minp.Z)
419                         minp = meas.vertices[meas.indices[j]];
420                     else if (meas.vertices[meas.indices[j]].Z > maxp.Z)
421                         maxp = meas.vertices[meas.indices[j]];
422                 }
423
424                 for (int j = i - 2; j <= i; j++)
425                 {
426                     if (meas.vertices[meas.indices[j]].X != minp.X &&
        meas.vertices[meas.indices[j]].Y != minp.Y && meas.vertices[meas.indices[j]].Z != minp.Z &&
        meas.vertices[meas.indices[j]].X != maxp.X && meas.vertices[meas.indices[j]].Y != maxp.Y &&
        meas.vertices[meas.indices[j]].Z != maxp.Z)
427                         midp = meas.vertices[meas.indices[j]];
428                 }
429
430                 // Για κάθε μία ισοϋψή που περνάει μεταξύ των 2 σημείων αυτών, υπολογίζουμε τις
        πλευρές θα περάσει
431                 height = minp.Z - minp.Z % isodistance;
432                 while (height < maxp.Z)
433                 {
434                     if (height > minp.Z)
435                     {
436                         // Αφού υπάρχει ισοϋψής, το ένα σημείο περνάει ανάμεσα από το χαμηλότερο
        και το υψηλότερο σημείο. Κάνουμε γραμμική παρεμβολή.
437                         fpoint = linearInterpolation(minp, maxp, height);
438
```

```
439                         // Αν το ενδιάμεσο σημείο είναι πάνω από το ύψος της ισοϋψούς, τότε το
        2ο σημείο της ισοϋψούς περνάει ανάμεσα από το χαμηλότερο και το μεσαίο, αλλιώς περνάει από
        το μεσσαίο και το υψηλότερο.
440                         if (midp.Z > height)
441                         {
442                             spoint = linearInterpolation(minp, midp, height);
443                         }
444                         else
445                         {
446                             spoint = linearInterpolation(midp, maxp, height);
447                         }
448
449                         Array.Resize<geoContour>(ref meas.contours, meas.contours.Length + 1);
450                         meas.contours[meas.contours.GetUpperBound(0)] = new geoContour(fpoint,
        spoint, device);
451                     }
452
453                     height += isodistance;
454                 }
455
456                 meas.hascontour = true;
457             }
458         }
459
460         // Επαναφόρτωση μίας μέτρησης
461         private void resetMeasure(string measureID, Color measColor)
462         {
463             for (int j = 0; j < measures.Length; j++)
464             {
465                 if (measures[j].measID == int.Parse(measureID))
466                 {
467                     measures[j].visible = false;
468
469                     measures[j] = new geoMeasure();
470                     measures[j].measID = int.Parse(measureID);
471                     measures[j].Points = new geoPoint[0];
472
473                     OleDbCommand command1 = new OleDbCommand("SELECT PID, PName, X, Y, Z,
        isHeight FROM Points WHERE MID = " + measureID, mdbconnection);
474                     OleDbDataReader reader1 = command1.ExecuteReader();
475
476                     while (reader1.Read())
477                     {
478                         Array.Resize(ref measures[j].Points, measures[j].Points.Length + 1);
479                         measures[j].Points[measures[j].Points.Length - 1] = new geoPoint();
480                         measures[j].Points[measures[j].Points.Length - 1].PID =
        reader1.GetInt32(0).ToString();
481                         measures[j].Points[measures[j].Points.Length - 1].Pname =
        reader1.GetValue(1).ToString();
482                         measures[j].Points[measures[j].Points.Length - 1].X = (float)
        (reader1.GetDouble(2) % 10000);
483                         measures[j].Points[measures[j].Points.Length - 1].Y = (float)
        (reader1.GetDouble(3) % 10000);
484                         measures[j].Points[measures[j].Points.Length - 1].Z = (float)
        (reader1.GetDouble(4) % 10000);
485                         measures[j].Points[measures[j].Points.Length - 1].isHeight =
        reader1.GetBoolean(5);
486                     }
487
488                     reader1.Close();
489                     command1 = new OleDbCommand("SELECT DatenTime, TID FROM Measures WHERE MID =
        " + measureID, mdbconnection);
490                     reader1 = command1.ExecuteReader();
491                     reader1.Read();
492                     measures[j].measDate = DateTime.Parse(reader1.GetValue(0).ToString());
493                     measures[j].teamID = reader1.GetInt32(1);
494
495                     SetupMeasure(ref measures[j], measColor);
496
497                     measures[j].hascontour = false;
498                     measures[j].showcontour = false;
499
500                     setupCountours(ref measures[j], (float)numIsoDistance.Value);
```

```
501
502                     measures[j].visible = true;
503
504                     return;
505                 }
506             }
507         }
508
509         // Εμφάνιση/απόκρυψη ισοϋψών
510         private void displayContour(string measureID)
511         {
512             for (int i = 0; i < measures.Length; i++)
513             {
514                 if (measures[i].measID == int.Parse(measureID))
515                 {
516                     measures[i].showcontour = !measures[i].showcontour;
517                     return;
518                 }
519             }
520
521             return;
522         }
523
524         // Απόκρυψη μιας μέτρησης
525         private void hideMeasure(string measureID)
526         {
527             for (int i = 0; i < measures.Length; i++)
528             {
529                 if (measures[i].measID == int.Parse(measureID))
530                 {
531                     measures[i].visible = false;
532                     return;
533                 }
534             }
535
536             return;
537         }
538
539         // Αλλαγή χρώματος μίας μέτρησης
540         private void changeMeasColor(string measureID, Color newColor)
541         {
542             for (int i = 0; i < measures.Length;i++)
543                 if (measures[i].measID == int.Parse(measureID))
544                 {
545                     measures[i].lineColor = newColor;
546
547                     for (int j = 0; j < measures[i].vertices.Length; j++)
548                     {
549                         measures[i].vertices[j].Color = newColor.ToArgb();
550                     }
551
552                     measures[i].geoMesh =
        measures[i].geoMesh.Clone(measures[i].geoMesh.Options.Value,
        CustomVertex.PositionColoredTextured.Format, device);
553                     measures[i].geoMesh.SetVertexBufferData(measures[i].vertices,
        LockFlags.None);
554                     measures[i].geoMesh.SetIndexBufferData(measures[i].indices, LockFlags.None);
555
556                     if (EngineState == 2)
557                     {
558                         measures[i].geoMesh =
        measures[i].geoMesh.Clone(measures[i].geoMesh.Options.Value,
        CustomVertex.PositionNormalColored.Format, device);
559                         measures[i].geoMesh.ComputeNormals();
560                     }
561                     else if (EngineState == 3)
562                     {
563                         measures[i].geoMesh =
        measures[i].geoMesh.Clone(measures[i].geoMesh.Options.Value,
        CustomVertex.PositionNormalTextured.Format, device);
564                         measures[i].geoMesh.ComputeNormals();
565                     }
566                 }
```

```
567         }
568
569         // ---- ΒΟΗΘΗΤΙΚΕΣ ΣΥΝΑΡΤΗΣΕΙΣ ----
570
571         // Εύρεση μέτρησης από το ID της
572         private geoMeasure getMeasurebyID(string measureID)
573         {
574             foreach (geoMeasure meas in measures)
575                 if (meas.measID == int.Parse(measureID))
576                     return meas;
577
578             return new geoMeasure();
579         }
580
581         // Υπολογισμός γραμμικής παρεμβολής
582         private geoPoint linearInterpolation(CustomVertex.PositionColoredTextured lowp,
        CustomVertex.PositionColoredTextured highp, float height)
583         {
584             geoPoint result = new geoPoint();
585
586             result.X = (height - lowp.Z) / (highp.Z - lowp.Z) * (highp.X - lowp.X) + lowp.X;
587             result.Y = (height - lowp.Z) / (highp.Z - lowp.Z) * (highp.Y - lowp.Y) + lowp.Y;
588             result.Z = height;
589
590             return result;
591         }
592
593         // ---- ΣΥΝΑΡΤΗΣΕΙΣ ΔΙΑΧΕΙΡΙΣΗΣ ΤΗΣ ΒΑΣΗΣ ----
594
595         // Άνοιγμα αρχείου γεωβάσης
596         private bool openDBConnection(string filename)
597         {
598             mdbconnection = new
        OleDbConnection(String.Format("Provider=Microsoft.Jet.OLEDB.4.0;Data Source={0};",
        filename));
599
600             try
601             {
602                 mdbconnection.Open();
603
604                 measures = new geoMeasure[0];
605
606                 reloadList();
607
608                 this.Text = "Διαχείριση μετρήσεων - " + filename;
609             }
610             catch
611             {
612                 MessageBox.Show("Παρουσιάστηκε σφάλμα κατά το άνοιγμα της βάσης δεδομένων.
        Παρακαλώ ελέξτε ότι η βάση είναι έγκυρη.");
613                 return false;
614             }
615
616             menuDBProperties.Enabled = true;
617             σπουδαστέςToolStripMenuItem.Enabled = true;
618             ιδιότητεςToolStripMenuItem.Enabled = true;
619             return true;
620         }
621
622         // Καταχώρηση νέας μέτρησης στη βάση
623         private void insertMeasuretoDatabase(geoMeasure insertMeasure)
624         {
625             string newID;
626             OleDbCommand command1 = new OleDbCommand(String.Format("INSERT INTO Measures (TID,
        [DatenTime]) VALUES ({0}, #{1}#)", insertMeasure.teamID,
        insertMeasure.measDate.ToString("MM/dd/yyyy HH:mm")), mdbconnection);
627             command1.ExecuteNonQuery();
628             newID = new OleDbCommand("SELECT @@IDENTITY",
        mdbconnection).ExecuteScalar().ToString();
629
630             for (int i = 0; i < insertMeasure.Points.Length; i++)
631             {
```

```
632                 command1.CommandText = String.Format("INSERT INTO Points (MID, Pname, X, Y, Z,
        isHeight) VALUES ({0}, '{1}', {2}, {3}, {4}, {5})", newID, insertMeasure.Points[i].Pname,
        insertMeasure.Points[i].X, insertMeasure.Points[i].Y, insertMeasure.Points[i].Z,
        insertMeasure.Points[i].isHeight);
633                 command1.ExecuteNonQuery();
634             }
635         }
636 
637         // Αποθήκευση αλλαγών μίας μέτρησης στη βάση
638         private void modifymeasuretoDB(geoMeasure insertMeasure)
639         {
640             OleDbCommand command1 = new OleDbCommand(String.Format("DELETE FROM Points WHERE MID
        = {0}", insertMeasure.measID), mdbconnection);
641             command1.ExecuteNonQuery();
642 
643             command1 = new OleDbCommand(String.Format("UPDATE Measures SET TID = {0},
        [DatenTime] = #{1}# WHERE MID = {2}", insertMeasure.teamID,
        insertMeasure.measDate.ToString("MM/dd/yyyy HH:mm"), insertMeasure.measID), mdbconnection);
644             command1.ExecuteNonQuery();
645 
646             for (int i = 0; i < insertMeasure.Points.Length; i++)
647             {
648                 if (insertMeasure.Points[i].PID != string.Empty)
649                 {
650                     command1.CommandText = String.Format("INSERT INTO Points (MID, Pname, X, Y,
        Z, isHeight) VALUES ({0}, '{1}', {2}, {3}, {4}, {5})", insertMeasure.measID,
        insertMeasure.Points[i].Pname, insertMeasure.Points[i].X, insertMeasure.Points[i].Y,
        insertMeasure.Points[i].Z, insertMeasure.Points[i].isHeight);
651                     command1.ExecuteNonQuery();
652                 }
653                 else
654                 {
655                     command1.CommandText = String.Format("INSERT INTO Points (PID, MID, Pname,
        X, Y, Z, isHeight) VALUES ({0}, {1}, '{2}', {3}, {4}, {5}, {6})",
        insertMeasure.Points[i].PID, insertMeasure.measID, insertMeasure.Points[i].Pname,
        insertMeasure.Points[i].X, insertMeasure.Points[i].Y, insertMeasure.Points[i].Z,
        insertMeasure.Points[i].isHeight);
656                     command1.ExecuteNonQuery();
657                 }
658             }
659 
660             resetMeasure(insertMeasure.measID.ToString(), insertMeasure.lineColor);
661         }
662 
663         // Φόρτωση λίστας ομάδων/μετρήσεων από τη βάση
664         private void reloadList()
665         {
666             TreeNode tempnode;
667             bool waschecked;
668             treeMainList.Nodes.Clear();
669 
670             OleDbCommand command1, command2;
671             OleDbDataReader reader1, reader2;
672 
673             command1 = new OleDbCommand("SELECT Name, TID FROM Teams", mdbconnection);
674             reader1 = command1.ExecuteReader();
675 
676             while (reader1.Read())
677             {
678                 tempnode = treeMainList.Nodes.Add(reader1.GetValue(1).ToString(),
        reader1.GetString(0));
679                 command2 = new OleDbCommand("SELECT DatenTime, MID FROM Measures WHERE TID=" +
        reader1.GetValue(1), mdbconnection);
680                 reader2 = command2.ExecuteReader();
681                 while (reader2.Read())
682                 {
683                     waschecked = false;
684                     foreach (geoMeasure meas in measures)
685                         if (meas.measID == (int)reader2.GetValue(1) && meas.visible)
686                             waschecked = true;
687                     tempnode.Nodes.Add(reader2.GetValue(1).ToString(),
        reader2.GetValue(0).ToString()).Checked = waschecked;
688                 }
```

```
689             }
690         }
691 
692         // ---- ΣΥΝΑΡΤΗΣΗ ΕΞΑΓΩΓΗΣ ΓΡΑΦΙΚΩΝ ΣΕ DXF ----
693         private void exporttodxf(string filename, bool exportTriangulation, bool exportContours)
694         {
695             DXFLibrary.Document doc = new DXFLibrary.Document();
696 
697             DXFLibrary.Tables tables = new DXFLibrary.Tables();
698             doc.SetTables(tables);
699 
700             DXFLibrary.Table layers = new DXFLibrary.Table("LAYER");
701             tables.addTable(layers);
702 
703             DXFLibrary.Point dxfpoint;
704             DXFLibrary.Line dxfline;
705 
706             foreach (geoMeasure meas in measures)
707             {
708                 if (meas.visible)
709                 {
710                     DXFLibrary.Layer layerPoints;
711                     layerPoints = new DXFLibrary.Layer(meas.measDate.ToString("s"), 0,
        "CONTINUOUS");
712                     layers.AddTableEntry(layerPoints);
713 
714                     foreach (geoPoint point in meas.Points)
715                     {
716                         dxfpoint = new DXFLibrary.Point((double)point.X, (double)point.Y,
        (double)point.Z, meas.measDate.ToString("s"));
717                         doc.add(dxfpoint);
718                     }
719 
720                     if (exportTriangulation)
721                     {
722                         layerPoints = new DXFLibrary.Layer(meas.measDate.ToString("s") + "-
        Triangulation", 30, "CONTINUOUS");
723                         layers.AddTableEntry(layerPoints);
724                         for (int i = 2; i < meas.indices.Length; i += 3)
725                         {
726                             dxfline = new DXFLibrary.Line(meas.measDate.ToString("s") + "-
        Triangulation", meas.vertices[meas.indices[i - 2]].X, meas.vertices[meas.indices[i - 2]].Y,
        meas.vertices[meas.indices[i - 2]].Z, meas.vertices[meas.indices[i - 1]].X,
        meas.vertices[meas.indices[i - 1]].Y, meas.vertices[meas.indices[i - 1]].Z);
727                             doc.add(dxfline);
728                             dxfline = new DXFLibrary.Line(meas.measDate.ToString("s") + "-
        Triangulation", meas.vertices[meas.indices[i - 1]].X, meas.vertices[meas.indices[i - 1]].Y,
        meas.vertices[meas.indices[i - 1]].Z, meas.vertices[meas.indices[i]].X,
        meas.vertices[meas.indices[i]].Y, meas.vertices[meas.indices[i]].Z);
729                             doc.add(dxfline);
730                             dxfline = new DXFLibrary.Line(meas.measDate.ToString("s") + "-
        Triangulation", meas.vertices[meas.indices[i - 2]].X, meas.vertices[meas.indices[i - 2]].Y,
        meas.vertices[meas.indices[i - 2]].Z, meas.vertices[meas.indices[i]].X,
        meas.vertices[meas.indices[i]].Y, meas.vertices[meas.indices[i]].Z);
731                             doc.add(dxfline);
732                         }
733                     }
734 
735                     if (exportContours)
736                     {
737                         layerPoints = new DXFLibrary.Layer(meas.measDate.ToString("s") + "-
        Contour", 50, "CONTINUOUS");
738                         layers.AddTableEntry(layerPoints);
739                         foreach (geoContour cont in meas.contours)
740                         {
741                             dxfline = new DXFLibrary.Line(meas.measDate.ToString("s") + "-
        Contour", cont.vertices[0].X, -cont.vertices[0].Y, cont.vertices[0].Z, cont.vertices[1].X,
        -cont.vertices[1].Y, cont.vertices[1].Z);
742                             doc.add(dxfline);
743                         }
744                     }
745                 }
746             }
```

```
747
748             FileStream f1 = new FileStream(filename, System.IO.FileMode.Create);
749             DXFLibrary.Writer.Write(doc, f1);
750             f1.Close();
751         }
752         // ---- ΣΥΝΑΡΤΗΣΕΙΣ ΔΙΑΧΕΙΡΙΣΗΣ ΑΝΤΙΚΕΙΜΕΝΩΝ ΤΗΣ ΦΟΡΜΑΣ WINDOWS ----
753         // ΣΥΝΑΡΤΗΣΕΙΣ ΒΑΣΙΚΟΥ ΜΕΝΟΥ
754         // Κλικ στο μενού Αρχείο->Δημιουργία...
755         private void menuCreateDB_Click(object sender, EventArgs e)
756         {
757             frmCreateDB newdb = new frmCreateDB();
758             if (newdb.ShowDialog() == DialogResult.OK)
759                 openDBConnection(newdb.txtFileName.Text);
760         }
761
762         // Κλικ στο μενού Αρχείο->Άνοιγμα
763         private void menuOpen_Click(object sender, EventArgs e)
764         {
765             if (openMDBDialog.ShowDialog() == DialogResult.OK)
766             {
767                 openDBConnection(openMDBDialog.FileName);
768             }
769         }
770
771         // Κλικ στο μενού Αρχείο->Εξαγωγή
772         private void exportDXF_Click(object sender, EventArgs e)
773         {
774             if (treeMainList.Nodes.Count == 0)
775             {
776                 MessageBox.Show("Δεν υπάρχουν στοιχεία στη βάση για να εξαχθούν.", "Κενή βάση");
777                 return;
778             }
779
780             if (dlgExport.ShowDialog() == DialogResult.Cancel)
781                 return;
782
783             exporttodxf(dlgExport.FileName, true, true);
784         }
785
786         // Κλικ στο μενού Αρχείο->Έξοδος
787         private void έξοδοςToolStripMenuItem_Click(object sender, EventArgs e)
788         {
789             Application.Exit();
790         }
791
792         // Κλικ στο μενού Βάση->Σπουδαστές
793         private void σπουδαστέςToolStripMenuItem_Click(object sender, EventArgs e)
794         {
795             frmSpoudastes SpoudastesForm = new frmSpoudastes(mdbconnection);
796             SpoudastesForm.ShowDialog();
797         }
798
799         // ΣΥΝΑΡΤΗΣΕΙΣ ΛΙΣΤΑΣ ΜΕΤΡΗΣΕΩΝ
800         // Τσεκάρισμα/ξετσεκάρισμα αντικειμένου στη λίστα μετρήσεων
801         private void treeMainList_AfterCheck(object sender, TreeViewEventArgs e)
802         {
803             if (e.Node.Level == 0)
804             {
805                 if (EngineUp == false)
806                 {
807                     EngineUp = true;
808                 }
809
810                 for (int i = 0; i < e.Node.Nodes.Count; i++)
811                 {
812                     e.Node.Nodes[i].Checked = e.Node.Checked;
813                     Application.DoEvents();
814                 }
815
816                 Run();
817             }
818             else if (e.Node.Level == 1)
819             {
```

```
820                 if (e.Node.Checked)
821                 {
822                     if (EngineUp == false)
823                         InitializeDevice();
824 
825                     displayMeasure(e.Node.Name, Color.White);
826                     if (measures.Length == 1)
827                         targettomeasure(e.Node.Name);
828 
829                     if (EngineUp == false)
830                         Run();
831                 }
832                 else
833                 {
834                     hideMeasure(e.Node.Name);
835                 }
836             }
837         }
838 
839         // Δεξί-κλικ στη λίστα μετρήσεων
840         private void treeView1_MouseClick(object sender, MouseEventArgs e)
841         {
842             if (e.Button == MouseButtons.Right)
843             {
844                 sourcenode = treeMainList.GetNodeAt(new Point(e.X, e.Y));
845                 treeMainList.SelectedNode = sourcenode;
846                 if (sourcenode.Level == 0)
847                 {
848                     ctxTeamMenu.Show(treeMainList, e.Location);
849                 }
850                 else if (sourcenode.Level == 1)
851                 {
852                     measShowHideContour.Checked =
       getMeasurebyID(treeMainList.SelectedNode.Name).showcontour;
853                     ctxMeasureMenu.Show(treeMainList, e.Location);
854                 }
855                 else
856                 {
857                     treeMainList.ContextMenuStrip = null;
858                 }
859             }
860         }
861 
862         // ΣΥΝΑΡΤΗΣΕΙΣ ΜΕΝΟΥ-ΕΛΕΓΧΟΥ ΟΜΑΔΑΣ/ΜΕΤΡΗΣΗΣ
863         // Κλικ στο μενού ομάδας "Εισαγωγή μέτρησης..."
864         private void menuInsertMeasure_Click(object sender, EventArgs e)
865         {
866             frmInsertMeasure insertmeas = new frmInsertMeasure();
867             insertmeas.returnMeasure = new geoMeasure();
868             insertmeas.returnMeasure.teamID = int.Parse(treeMainList.SelectedNode.Name);
869 
870             insertmeas.teamids = new string[treeMainList.Nodes.Count];
871             insertmeas.teamnames = new string[treeMainList.Nodes.Count];
872 
873             for (int i = 0; i < treeMainList.Nodes.Count; i++)
874             {
875                 insertmeas.teamids[i] = treeMainList.Nodes[i].Name;
876                 insertmeas.teamnames[i] = treeMainList.Nodes[i].Text;
877             }
878 
879             if (insertmeas.ShowDialog() != DialogResult.Cancel)
880             {
881                 insertMeasuretoDatabase(insertmeas.returnMeasure);
882                 reloadList();
883             }
884         }
885 
886         // Κλικ στο μενού μέτρησης "Αλλαγή χρώματος..."
887         private void αλλαγήΧρώματοςToolStripMenuItem_Click(object sender, EventArgs e)
888         {
889             if (colorDialog1.ShowDialog() == DialogResult.Cancel)
890                 return;
891 
```

```
892             changeMeasColor(treeMainList.SelectedNode.Name, colorDialog1.Color);
893         }
894 
895         // Κλικ στο μενού μέτρησης "Ισοϋψείς"
896         private void showContour_Click(object sender, EventArgs e)
897         {
898             displayContour(treeMainList.SelectedNode.Name);
899         }
900 
901         // Κλικ στο μενού μέτρησης "Επεξεργασία..."
902         private void επεξεργασίαToolStripMenuItem1_Click(object sender, EventArgs e)
903         {
904             frmEditMeasure editMeasure = new
    frmEditMeasure(getMeasurebyID(treeMainList.SelectedNode.Name));
905             editMeasure.teamids = new string[treeMainList.Nodes.Count];
906             editMeasure.teamnames = new string[treeMainList.Nodes.Count];
907 
908             for (int i = 0; i < treeMainList.Nodes.Count; i++)
909             {
910                 editMeasure.teamids[i] = treeMainList.Nodes[i].Name;
911                 editMeasure.teamnames[i] = treeMainList.Nodes[i].Text;
912             }
913 
914             if (editMeasure.ShowDialog() == DialogResult.OK)
915             {
916                 modifymeasuretoDB(editMeasure.returnMeasure);
917                 reloadList();
918             }
919         }
920 
921         // Κλικ στο μενού μέτρησης "Διαγραφή"
922         private void διαγραφήToolStripMenuItem_Click(object sender, EventArgs e)
923         {
924             if (MessageBox.Show("Είστε σίγουροι ότι θέλετε να διαγράψετε αυτές τις μετρήσεις;",
    "Διαγραφή μέτρησης", MessageBoxButtons.YesNo) == DialogResult.Yes)
925             {
926                 OleDbCommand command1 = new OleDbCommand("DELETE * FROM Points WHERE MID = " +
    treeMainList.SelectedNode.Name, mdbconnection);
927                 command1.ExecuteNonQuery();
928                 command1.CommandText = "DELETE * FROM Measures WHERE MID = " +
    treeMainList.SelectedNode.Name;
929                 command1.ExecuteNonQuery();
930 
931                 treeMainList.SelectedNode.Remove();
932             }
933         }
934 
935         // ΥΠΟΛΟΙΠΑ ΑΝΤΙΚΕΙΜΕΝΑ ΦΟΡΜΑΣ
936         // Αλλαγή τιμής combobox τύπου απεικόνισης
937         private void cmbViewType_SelectedIndexChanged(object sender, EventArgs e)
938         {
939             if (device == null)
940                 return;
941 
942             EngineState = cmbViewType.SelectedIndex;
943 
944             for (int i = 0; i < measures.Length; i++)
945                     changeMeasColor(measures[i].measID.ToString(), measures[i].lineColor);
946         }
947 
948         // Αλλαγή τιμής ισοδιάστασης ισοϋψών
949         private void numIsoDistance_ValueChanged(object sender, EventArgs e)
950         {
951             for (int i = 0; i < measures.Length; i++)
952                 setupCountours(ref measures[i], (float)numIsoDistance.Value);
953         }
954 
955         // Είσοδος δείκτη ποντικιού στην περιοχή απεικόνισης
956         private void pictureBox1_MouseEnter(object sender, EventArgs e)
957         {
958             inView = true;
959         }
960 
```

```
961         // Έξοδος δείκτη ποντικιού από την περιοχή απεικόνισης
962         private void pictureBox1_MouseLeave(object sender, EventArgs e)
963         {
964             inView = false;
965         }
966 
967         //Ενέργειες κατά το κλείσιμο του παραθύρου
968         private void mainForm_FormClosing(object sender, FormClosingEventArgs e)
969         {
970             EngineUp = false;
971             Application.DoEvents();
972             Application.Exit();
973         }
974     }
975 }
```

frmCreateDB.cs

```
1   using System;
2   using System.Collections.Generic;
3   using System.ComponentModel;
4   using System.Data;
5   using System.Data.OleDb;
6   using System.Drawing;
7   using System.Text;
8   using System.Windows.Forms;
9 
10  namespace diplomatiki
11  {
12      public partial class frmCreateDB : Form
13      {
14          public frmCreateDB()
15          {
16              InitializeComponent();
17          }
18 
19          private void frmCreateDB_Load(object sender, EventArgs e)
20          {
21              txtFileName.Text = Application.ExecutablePath + "\\ΒάσηΜετρήσεων1.mdb";
22          }
23 
24          private void btnBrowse_Click(object sender, EventArgs e)
25          {
26              if (mdbFileDialog.ShowDialog() == DialogResult.OK)
27                  txtFileName.Text = mdbFileDialog.FileName;
28          }
29 
30          private void btnOK_Click(object sender, EventArgs e)
31          {
32              if (System.IO.File.Exists(txtFileName.Text))
33                  System.IO.File.Delete(txtFileName.Text);
34 
35              ADOX.CatalogClass newdb = new ADOX.CatalogClass();
36              newdb.Create(String.Format("Provider=Microsoft.Jet.OLEDB.4.0;Data Source={0};Jet
        OLEDB:Engine Type=5", txtFileName.Text));
37              newdb = null;
38 
39              OleDbConnection dbcon = new
        OleDbConnection(String.Format("Provider=Microsoft.Jet.OLEDB.4.0;Data Source={0};",
        txtFileName.Text));
40              dbcon.Open();
41 
42              //Δημιουργία πίνακα ιδιότητων της βάσης
43              OleDbCommand dbcmd = new OleDbCommand("CREATE TABLE Props (Name char(50), Val
        char(50))", dbcon);
44              dbcmd.ExecuteNonQuery();
45 
46              //Δημιουργία πίνακα ομάδων
47              dbcmd = new OleDbCommand("CREATE TABLE Teams (TID int NOT NULL PRIMARY KEY, Name
        char(30) NOT NULL)", dbcon);
48              dbcmd.ExecuteNonQuery();
49 
50              //Δημιουργία πίνακα σπουδαστών
```

```
51              dbcmd = new OleDbCommand("CREATE TABLE Members (PeopleID AUTOINCREMENT NOT NULL
        PRIMARY KEY, Onoma char(255), Epitheto char(255), Kwdikos char(255), TID int)", dbcon);
52              dbcmd.ExecuteNonQuery();
53
54              //Δημιουργία πίνακα μετρήσεων
55              dbcmd = new OleDbCommand("CREATE TABLE Measures (MID AUTOINCREMENT NOT NULL PRIMARY
        KEY, TID int, Location char(30), DatenTime DATETIME)", dbcon);
56              dbcmd.ExecuteNonQuery();
57
58              //Δημιουργία πίνακα σημείων
59              dbcmd = new OleDbCommand("CREATE TABLE Points (PID AUTOINCREMENT NOT NULL PRIMARY
        KEY, MID int, Pname char(10), X double, Y double, Z double, isHeight yesno)", dbcon);
60              dbcmd.ExecuteNonQuery();
61
62              for (int i = 1; i <= (int)numTeamsCount.Value; i++)
63              {
64                  dbcmd = new OleDbCommand(String.Format("INSERT INTO Teams VALUES ({0}, '{1}')",
        i, "Ομάδα " + i), dbcon);
65                  dbcmd.ExecuteNonQuery();
66              }
67
68              dbcon.Close();
69
70              this.DialogResult = DialogResult.OK;
71              this.Close();
72          }
73      }
74  }
```

frmSpoudastes.cs

```
1   using System;
2   using System.Collections.Generic;
3   using System.ComponentModel;
4   using System.Data.OleDb;
5   using System.Drawing;
6   using System.Text;
7   using System.Data;
8   using System.Windows.Forms;
9
10  namespace diplomatiki
11  {
12      public partial class frmSpoudastes : Form
13      {
14          OleDbConnection dbConnection;
15          geoStudent[] Spoudastes;
16
17          public frmSpoudastes(OleDbConnection argConnection)
18          {
19              InitializeComponent();
20              dbConnection = argConnection;
21          }
22
23          private void frmSpoudastes_Load(object sender, EventArgs e)
24          {
25              OleDbCommand comSpoudastes = new OleDbCommand("SELECT PeopleID, Onoma, Epitheto,
        Kwdikos, TID FROM Members", dbConnection);
26              OleDbDataReader readSpoudastes = comSpoudastes.ExecuteReader();
27
28              Spoudastes = new geoStudent[0];
29              int i = 0;
30
31              while (readSpoudastes.Read())
32              {
33                  Array.Resize<geoStudent>(ref Spoudastes, i + 1);
34                  Spoudastes[i].PeopleID = (int)readSpoudastes.GetValue(0);
35                  Spoudastes[i].FirstName = readSpoudastes.GetString(1);
36                  Spoudastes[i].LastName = readSpoudastes.GetString(2);
37                  Spoudastes[i].Kwdikos = readSpoudastes.GetString(3);
38                  Spoudastes[i].TID = (int)readSpoudastes.GetValue(4);
39                  i++;
40              }
41
```

```
42              OleDbCommand comOmades = new OleDbCommand("SELECT TID, [Name] FROM Teams",
        dbConnection);
43              OleDbDataReader readOmades = comOmades.ExecuteReader();
44
45              while (readOmades.Read())
46              {
47                  cmbOmada.Items.Add(readOmades.GetString(1));
48              }
49
50              reloadlist();
51          }
52
53          private void reloadlist()
54          {
55              int oldSel = -1;
56              if (listSpoudastes.SelectedItems.Count > 0)
57                  oldSel = listSpoudastes.SelectedIndices[0];
58              listSpoudastes.Items.Clear();
59
60              foreach (geoStudent Spoudastis in Spoudastes)
61                  listSpoudastes.Items.Add(Spoudastis.PeopleID.ToString(),
        Spoudastis.LastName.Replace("  ","") + ", " + Spoudastis.FirstName.Replace("  ", ""),
        -1).SubItems.Add(Spoudastis.Kwdikos);
62
63              if(oldSel != -1)
64                  listSpoudastes.Items[oldSel].Selected = true;
65          }
66
67          private void listSpoudastes_SelectedIndexChanged(object sender, EventArgs e)
68          {
69              if (listSpoudastes.SelectedItems.Count == 0)
70              {
71                  clearStoixeia();
72                  return;
73              }
74
75              fillStoixeia(listSpoudastes.SelectedIndices[0]);
76
77              return;
78          }
79
80          private void clearStoixeia()
81          {
82              txtOnoma.Text = String.Empty;
83              txtEpitheto.Text = string.Empty;
84              txtKwdikos.Text = string.Empty;
85              cmbOmada.SelectedIndex = -1;
86          }
87
88          private void fillStoixeia(int i)
89          {
90              txtOnoma.Text = Spoudastes[i].FirstName;
91              txtEpitheto.Text = Spoudastes[i].LastName;
92              txtKwdikos.Text = Spoudastes[i].Kwdikos;
93              cmbOmada.SelectedIndex = Spoudastes[i].TID;
94          }
95
96          private void savechanges(geoStudent spoudastis)
97          {
98              OleDbCommand comSpoudastes = new OleDbCommand("SELECT PeopleID FROM Members WHERE
        PeopleID = " + spoudastis.PeopleID, dbConnection);
99              if (comSpoudastes.ExecuteReader().HasRows)
100             {
101                 OleDbCommand comUpdateSpoudastes = new OleDbCommand(string.Format("UPDATE
        Members SET Onoma='{0}', Epitheto='{1}', Kwdikos='{2}', TID={3} WHERE PeopleID = {4}",
        spoudastis.FirstName, spoudastis.LastName, spoudastis.Kwdikos, spoudastis.TID,
        spoudastis.PeopleID), dbConnection);
102                 comUpdateSpoudastes.ExecuteNonQuery();
103             }
104             else
105             {
```

```
106                 OleDbCommand comInsertSpoudastes = new OleDbCommand(string.Format("INSERT INTO
        Members (Onoma, Epitheto, Kwdikos, TID) VALUES ('{0}', '{1}', '{2}', {3})",
        spoudastis.FirstName, spoudastis.LastName, spoudastis.Kwdikos, spoudastis.TID,
        spoudastis.PeopleID), dbConnection);
107                 comInsertSpoudastes.ExecuteNonQuery();
108             }
109         }
110
111         private void btnCancel_Click(object sender, EventArgs e)
112         {
113             if(listSpoudastes.SelectedItems.Count > 0)
114                 fillStoixeia(listSpoudastes.SelectedIndices[0]);
115         }
116
117         private void btnAccept_Click(object sender, EventArgs e)
118         {
119             if (listSpoudastes.SelectedItems.Count > 0)
120                 saveStoixeia(listSpoudastes.SelectedIndices[0]);
121
122             reloadlist();
123         }
124
125         private void saveStoixeia(int i)
126         {
127             Spoudastes[i].FirstName = txtOnoma.Text;
128             Spoudastes[i].LastName = txtEpitheto.Text;
129             Spoudastes[i].Kwdikos = txtKwdikos.Text;
130             Spoudastes[i].TID = cmbOmada.SelectedIndex;
131
132             savechanges(Spoudastes[i]);
133         }
134
135         private void νέοςΣπουδαστήςToolStripMenuItem_Click(object sender, EventArgs e)
136         {
137             int i = Spoudastes.Length;
138
139             Array.Resize<geoStudent>(ref Spoudastes, Spoudastes.Length + 1);
140
141             Spoudastes[i].FirstName = "Όνομα";
142             Spoudastes[i].LastName = "Επώνυμο";
143
144             reloadlist();
145
146             listSpoudastes.SelectedItems.Clear();
147             listSpoudastes.Items[listSpoudastes.Items.Count - 1].Selected = true;
148         }
149     }
150 }
```

frmInsertMeasure.cs

```
1   using System;
2   using System.Collections.Generic;
3   using System.ComponentModel;
4   using System.Data;
5   using System.Drawing;
6   using System.Text;
7   using System.Windows.Forms;
8   using System.IO;
9   using System.Data.OleDb;
10
11  namespace diplomatiki
12  {
13      public partial class frmInsertMeasure : Form
14      {
15          public geoMeasure returnMeasure;
16          public string[] teamids, teamnames;
17
18          public frmInsertMeasure()
19          {
20              InitializeComponent();
21          }
22
```

```
23          private void btnOK_Click(object sender, EventArgs e)
24          {
25              returnMeasure.Points = new geoPoint[dataShmeia.RowCount - 1];
26              returnMeasure.teamID = int.Parse(teamids[cmbTeam.SelectedIndex]);
27              returnMeasure.measDate = dateHmeromhnia.Value;
28
29              for (int i = 0; i < dataShmeia.RowCount - 1; i++)
30              {
31                  returnMeasure.Points[i].Pname = (string)dataShmeia.Rows[i].Cells["PCode"].Value;
32                  returnMeasure.Trans = new geoPoint(0, 0, 0);
33                  returnMeasure.Points[i].X = (float)((double)dataShmeia.Rows[i].Cells["X"].Value
        % 10000);
34                  returnMeasure.Trans.X = (float)(int)((double)dataShmeia.Rows[i].Cells["X"].Value
        / 10000);
35                  returnMeasure.Points[i].Y = (float)((double)dataShmeia.Rows[i].Cells["Y"].Value
        % 10000);
36                  returnMeasure.Trans.Y = (float)(int)((double)dataShmeia.Rows[i].Cells["Y"].Value
        / 10000);
37                  returnMeasure.Points[i].Z = (float)((double)dataShmeia.Rows[i].Cells["Z"].Value
        % 10000);
38                  returnMeasure.Trans.Z = (float)(int)((double)dataShmeia.Rows[i].Cells["Z"].Value
        / 10000);
39                  returnMeasure.Points[i].isHeight =
        (bool)dataShmeia.Rows[i].Cells["clmIsHeight"].Value;
40              }
41
42              this.DialogResult = DialogResult.OK;
43              this.Close();
44          }
45
46          private void frmInsertMeasure_Load(object sender, EventArgs e)
47          {
48              cmbTeam.Items.Clear();
49              for (int i = 0; i < teamids.Length; i++)
50                  cmbTeam.Items.Add(teamnames[i]);
51
52              cmbTeam.SelectedIndex = Array.IndexOf(teamids, returnMeasure.teamID.ToString());
53          }
54
55          private void btnImportFromTxt_Click(object sender, EventArgs e)
56          {
57              string line;
58              int i;
59              if (dlgImportTxt.ShowDialog() == DialogResult.Cancel)
60                  return;
61
62              FileStream fstream = new FileStream(dlgImportTxt.FileName, FileMode.Open,
        FileAccess.Read);
63              StreamReader reader = new StreamReader(fstream);
64
65              i = 0;
66              while (!reader.EndOfStream)
67              {
68                  line = reader.ReadLine();
69
70                  try
71                  {
72                      if (line.Split(' ').Length == 5)
73                          dataShmeia.Rows.Add(line.Split(':')[1], float.Parse(line.Split(':')[2]),
        float.Parse(line.Split(':')[3]), float.Parse(line.Split(':')[4]), true);
74                      else if (line.Split(new string[]{", "}, StringSplitOptions.None).Length ==
        4)
75                          dataShmeia.Rows.Add(line.Split(new string[] { ", " },
        StringSplitOptions.None)[0], double.Parse(line.Split(new string[] { ", " },
        StringSplitOptions.None)[1]), double.Parse(line.Split(new string[] { ", " },
        StringSplitOptions.None)[2]), double.Parse(line.Split(new string[] { ", " },
        StringSplitOptions.None)[3]), true);
76                      else
77                          dataShmeia.Rows.Add(i.ToString(), float.Parse(line.Split(' ')[0]),
        float.Parse(line.Split(' ')[1]), float.Parse(line.Split(' ')[2]), true);
78                      i++;
79                  }
80                  catch
```

```
81                  {
82                  }
83              }
84
85
86              reader.Close();
87              fstream.Close();
88          }
89      }
90  }
```

frmEditMeasure.cs

```
1   using System;
2   using System.Collections.Generic;
3   using System.ComponentModel;
4   using System.Data;
5   using System.Drawing;
6   using System.Text;
7   using System.Windows.Forms;
8   using System.IO;
9   using System.Data.OleDb;
10
11  namespace diplomatiki
12  {
13      public partial class frmEditMeasure : Form
14      {
15          public geoMeasure returnMeasure;
16          public string[] teamids, teamnames;
17
18          public frmEditMeasure(geoMeasure currentmeasure)
19          {
20              InitializeComponent();
21              returnMeasure = currentmeasure;
22          }
23
24          private void btnOK_Click(object sender, EventArgs e)
25          {
26              returnMeasure.Points = new geoPoint[dataShmeia.RowCount - 1];
27              returnMeasure.teamID = int.Parse(teamids[cmbTeam.SelectedIndex]);
28              returnMeasure.measDate = dateHmeromhnia.Value;
29
30              for (int i = 0; i < dataShmeia.RowCount - 1; i++)
31              {
32                  returnMeasure.Points[i].PID = (string)dataShmeia.Rows[i].Cells["PID"].Value;
33                  returnMeasure.Points[i].Pname = (string)dataShmeia.Rows[i].Cells["PCode"].Value;
34                  returnMeasure.Trans = new geoPoint(0, 0, 0);
35                  if ((float)dataShmeia.Rows[i].Cells["X"].Value > 10000)
36                  {
37                      returnMeasure.Points[i].X = (float)
        ((double)dataShmeia.Rows[i].Cells["X"].Value % 10000);
38                      returnMeasure.Trans.X = (float)(int)
        ((double)dataShmeia.Rows[i].Cells["X"].Value / 10000);
39                  }
40                  else
41                      returnMeasure.Points[i].X = (float)(dataShmeia.Rows[i].Cells["X"].Value);
42                  if ((float)dataShmeia.Rows[i].Cells["Y"].Value > 10000)
43                  {
44                      returnMeasure.Points[i].Y = (float)
        ((double)dataShmeia.Rows[i].Cells["Y"].Value % 10000);
45                      returnMeasure.Trans.Y = (float)(int)
        ((double)dataShmeia.Rows[i].Cells["Y"].Value / 10000);
46                  }
47                  else
48                      returnMeasure.Points[i].Y = (float)(dataShmeia.Rows[i].Cells["Y"].Value);
49                  if ((float)dataShmeia.Rows[i].Cells["Z"].Value > 10000)
50                  {
51                      returnMeasure.Points[i].Z = (float)
        ((double)dataShmeia.Rows[i].Cells["Z"].Value % 10000);
52                      returnMeasure.Trans.Z = (float)(int)
        ((double)dataShmeia.Rows[i].Cells["Z"].Value / 10000);
53                  }
54                  else
```

```
55                      returnMeasure.Points[i].Z = (float)(dataShmeia.Rows[i].Cells["Z"].Value);
56                  returnMeasure.Points[i].isHeight =
        (bool)dataShmeia.Rows[i].Cells["isHeight"].Value;
57              }
58
59              this.DialogResult = DialogResult.OK;
60              this.Close();
61          }
62
63          private void frmInsertMeasure_Load(object sender, EventArgs e)
64          {
65              cmbTeam.Items.Clear();
66              for (int i = 0; i < teamids.Length; i++)
67                  cmbTeam.Items.Add(teamnames[i]);
68
69              cmbTeam.SelectedIndex = Array.IndexOf(teamids, returnMeasure.teamID.ToString());
70
71              dateHmeromhnia.Value = returnMeasure.measDate;
72
73              foreach (geoPoint apoint in returnMeasure.Points)
74                  dataShmeia.Rows.Add(apoint.PID, apoint.Pname, apoint.X, apoint.Y, apoint.Z,
        apoint.isHeight);
75          }
76
77          private void btnImportFromTxt_Click(object sender, EventArgs e)
78          {
79              string line;
80              int i;
81              if (dlgImportTxt.ShowDialog() == DialogResult.Cancel)
82                  return;
83
84              FileStream fstream = new FileStream(dlgImportTxt.FileName, FileMode.Open,
        FileAccess.Read);
85              StreamReader reader = new StreamReader(fstream);
86
87              i = 0;
88              while (!reader.EndOfStream)
89              {
90                  line = reader.ReadLine();
91
92                  try
93                  {
94                      if (line.Split(' ').Length == 5)
95                          dataShmeia.Rows.Add(line.Split(':')[1], float.Parse(line.Split(':')[2]),
        float.Parse(line.Split(':')[3]), float.Parse(line.Split(':')[4]), true);
96                      else if (line.Split(new string[]{", "}, StringSplitOptions.None).Length ==
        4)
97                          dataShmeia.Rows.Add(line.Split(new string[] { ", " },
        StringSplitOptions.None)[0], double.Parse(line.Split(new string[] { ", " },
        StringSplitOptions.None)[1]), double.Parse(line.Split(new string[] { ", " },
        StringSplitOptions.None)[2]), double.Parse(line.Split(new string[] { ", " },
        StringSplitOptions.None)[3]), true);
98                      else
99                          dataShmeia.Rows.Add(i.ToString(), float.Parse(line.Split(' ')[0]),
        float.Parse(line.Split(' ')[1]), float.Parse(line.Split(' ')[2]), true);
100                     i++;
101                 }
102                 catch
103                 {
104                 }
105             }
106
107
108             reader.Close();
109             fstream.Close();
110         }
111     }
112 }
```