ΓΕΩΠΟΝΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΓΕΝΙΚΟ ΤΜΗΜΑ

ΤΟΜΕΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ, ΜΑΘΗΜΑΤΙΚΩΝ ΚΑΙ ΣΤΑΤΙΣΤΙΚΗΣ

ΕΡΓΑΣΤΗΡΙΟ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΓΕΩΠΛΗΡΟΦΟΡΙΚΗ

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΑΤΡΙΒΗ

«Ανθρωποκεντρική πλοήγηση σε αγροτικούς και περιβαλλοντικούς χώρους»

Ξενοφών Χ. Γιωργουδέλλης

Ηλεκτρολόγος Μηχανικός & Μηχανικός Η/Υ

Επιβλέπων: Καθ. Θ. Τσιλιγκιρίδης

ΑΘΗΝΑ 2010

ΓΕΩΠΟΝΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΓΕΝΙΚΟ ΤΜΗΜΑ

ΤΟΜΕΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ, ΜΑΘΗΜΑΤΙΚΩΝ ΚΑΙ ΣΤΑΤΙΣΤΙΚΗΣ

ΕΡΓΑΣΤΗΡΙΟ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΓΕΩΠΛΗΡΟΦΟΡΙΚΗ

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΑΤΡΙΒΗ

«Ανθρωποκεντρική πλοήγηση σε αγροτικούς και περιβαλλοντικούς χώρους»

Ξενοφών Χ. Γιωργουδέλλης

Ηλεκτρολόγος Μηχανικός & Μηχανικός Η/Υ

Επιβλέπων: Καθ. Θ. Τσιλιγκιρίδης

ΑΘΗΝΑ 2010

Τίτλος:

«Ανθρωποκεντρική πλοήγηση σε αγροτικούς και περιβαλλοντικούς χώρους»

Ξενοφών Χ. Γιωργουδέλλης
Ηλεκτρολόγος Μηχανικός & Μηχανικός Η/Υ

Επιβλέπων: Καθ. Θ. Τσιλιγκιρίδης

Μέλη εξεταστικής επιτροπής:

- **Τσιλιγκιρίδης Θ., Καθηγητής Γ.Π.Α. (Επιβλέπων)**
- **Γιαλούρης Κ., Αναπληρωτής καθηγητής Γ.Π.Α.**
- **Κωστοπούλου Κ., Επίκουρη καθηγήτρια Γ.Π.Α.**

ΠΕΡΙΛΗΨΗ

Στην σημερινή εποχή η πλοήγηση έχει εξελιχτεί από μια υπηρεσία απευθυνόμενη σε λίγους σε μια υπηρεσία ευρέως διαδομένη και χρησιμοποιούμενη από σχεδόν όλες τις ομάδες της κοινωνίας. Με την εξέλιξη των υπηρεσιών πλοήγησης παρουσιαστήκαν και καινούριες ανάγκες καθώς και η απαίτηση από τους χρήστες των λογισμικών πλοήγησης η υπηρεσία της πλοήγησης να προσαρμοστεί στις απαιτήσεις του ίδιου του χρήστη και του επαγγέλματος του. Η απλή διαδικασία που πρόσφεραν οι πρώτες συσκευές, (δήλωση σημείου εκκίνησης, δήλωση σημείου προορισμού και παρουσίαση διαδρομής χωρίς ιδιαίτερη δυνατότητα καθορισμού παραμέτρων), δεν ήταν πλέον αρκετή για να ικανοποιήσει τον χρήστη. Ο χρήστης ήθελε η διαδικασία εύρεσης διαδρομής να είναι πιο ευέλικτη.

Για να ανταποκριθούν στις απαιτήσεις των χρηστών τα προγράμματα πλοήγησης πρόσφεραν παραπάνω δυνατότητες παραμετροποίησης, όπως δυνατότητα αποφυγής διαφόρων κόμβων (διόδια, πορθμεία κτλ), ιδιοποιήσεις για διάφορα σημεία ενδιαφέροντος (βενζινάδικα, εστιατόρια κτλ), καθώς και δυνατότητες παραμετροποιήσεις της διαδρομής με βάση την ταχύτητα και την διανυόμενη απόσταση. Είναι εύκολο να διαπιστωθεί ότι η τάση σχετικά με τα λογισμικά πλοήγησης γίνεται όλο και πιο ανθρωποκεντρική και η τεχνολογία κινείται προς την κατεύθυνση της συνεργασίας του λογισμικού με τον ανθρώπινο παράγοντα για την εύρεση της βέλτιστης διαδρομής.

Στην εργασία αυτή παρουσιάζεται ο σχεδιασμός, η ανάπτυξη και η υλοποίηση ενός Ηλεκτρονικού Συστήματος πλοήγησης με θέμα την ανθρωποκεντρική πλοήγηση σε αγροτικούς και περιβαλλοντικούς χώρους. Βασικός σκοπός της εργασίας είναι η κατασκευή ενός λογισμικού το όποιο λαμβάνοντας υπόψη του τις ιδιαιτερότητες των αγροτικών χώρων και τις απαιτήσεις του χρήστη να προσφέρει μεγάλη ευελιξία στην επιλογή της διαδρομής με βάση προτιμήσεις που συμπληρώνει.

Λέξεις κλειδιά: Ηλεκτρονικό σύστημα πλοήγησης, GPS, διαδρομή, navigation, αγροτικός χώρος, περιβαλλοντικούς χώρος

SUMMARY

In today era, navigation has evolved from a service available to a few people only, to a widely used service, which almost all of society group's use. With the navigation service evolution new requirements and needs were brought to the fore. It was soon apparent that there existed a need for the navigation service to become customized to the needs of the individual and his profession. The simple function offered by the first devices (declaration of starting point, declaration of ending point without the ability to change or take into account other parameters) was not enough to satisfy the modern user, the user needed the process of route finding to be more flexible.

In order to fulfill the demands the users the navigation software programs added new functions and gave the user the ability to affect parameters such as the ability to circumvent certain obstacles (tolls, docks etc), notifications about certain points of interest (gas stations, restaurants etc) and the ability to affect parameters concerning the route would be the shortest or the speediest. It can be easily deduced that the trend in navigation software is moving towards the collaboration between man and device in order to find the best suited route.

In this paper the architectural design, development and implementation of an Electronic navigation system is presented. The Electronic navigation system is centered in agricultural and environmental spaces. The main purpose of this paper is the development of a software package that will take into account both the demands of the user and the special conditions that apply to agricultural and environmental spaces and offer to the user great flexibility in choosing the route that he will follow in accordance to his needs.

Key words: Electronic navigation system, GPS, navigation, route, agricultural space, environmental space

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

Εικόνες

Πίνακες

ΕΥΧΑΡΙΣΤΙΕΣ

Ευχαριστώ θερμά τον καθηγητή Θ. Τσιλιγκιρίδη για την εμπιστοσύνη του να μου αναθέσει την παρούσα πτυχιακή μελέτη, καθώς και για την επίβλεψη και την καθοδήγηση του στην υλοποίησή της.

Θα ήθελα να εκφράσω τις ευχαριστίες μου στον αναπληρωτή καθηγητή κ. Κ. Γιαλούρη για την βοήθεια στο σχεδιασμό του λογισμικού πλοήγησης της μεταπτυχιακής μου εργασίας και για τον πολύτιμο χρόνο που μου αφιέρωσε. Οι συμβουλές και υποδείξεις του ήταν εύστοχες και πολύτιμες.

Ευχαριστώ την επίκουρη καθηγήτρια κ. Κ. Κωστοπούλου, τον επίκουρο καθηγητή κ. Γ. Παπαδόπουλο και τον λέκτορα κ. Γ. Λαγογιάννη για τις εύστοχες παρατηρήσεις τους.

Ευχαριστώ όλους τους καθηγητές μου και διδάσκοντες καθ' όλη τη διάρκεια της μεταπτυχιακής μου φοίτησης για τις πολύτιμες γνώσεις που μου μετέδωσαν και με βοήθησαν να γνωρίσω καλύτερα το αντικείμενο του κλάδου της Γεωπληροφορικής.

Τέλος, θα ήθελα να ευχαριστήσω τα προσφιλή μου πρόσωπα για την υποστήριξη τους κατά την εκπόνηση της εργασίας μου.

1. ΕΙΣΑΓΩΓΗ

1.1. Γενικά

Στη σημερινή εποχή οι υπηρεσίες πλοήγησης μπορούν να επιτρέψουν στους χρήστες να έχουν ακριβείς χωρικές πληροφορίες. Η χρήση συσκευών με δυνατότητα προσδιορισμού θέσης και διαδρομών που μπορεί να ακολουθήσει κάποιος για να φτάσει σε ένα προορισμό είναι ιδιαίτερα εκτεταμένη και ενσωματώνεται σε όλο και περισσότερες εφαρμογές. Σε πολλές περιπτώσεις όπως συμβαίνει στη κατηγορία των αυτοκινήτων η χρήση τους θεωρείται μάλλον αναμενόμενη παρά σαν επιλογή πολυτελείας. Παρόλα αυτά τα συστήματα δεν ανταποκρίνονται στις μεμονωμένες προτιμήσεις των χρηστών και δεν λαμβάνουν υπόψη τα ιδιαίτερα χαρακτηριστικά των διαδρομών, του περιβάλλοντος αλλά και του χρήστη. Συνήθως παρέχουν στο χρήστη τις πληροφορίες για τη συντομότερη ή τη γρηγορότερη διαδρομή που οδηγεί σε κάποιο επιθυμητό προορισμό. Ωστόσο μια διαδρομή που θέλει να ακολουθήσει ένας χρήστης χαρακτηρίζεται από διάφορες παραμέτρους στις οποίες ο χρήστης είναι ευαίσθητος και ως εκ τούτου λαμβάνει σοβαρά υπόψη του. Παραδείγματα τέτοιων παραμέτρων αποτελούν απόσταση που πρόκειται να διανυθεί, οι κλίσεις της διαδρομής, η πολυπλοκότητα της πορείας, το είδος του χρησιμοποιημένου οχήματος, οι επικρατούσες καιρικές συνθήκες, η ασφάλεια, κ.ά. (Winter & Corona, 2001).

1.2. Ορισμός προβλήματος

Με δεδομένο ότι πλέον τα συστήματα πλοήγησης γίνονται όλο και πιο δημοφιλή και προσιτά στον μέσο χρήστη, η απλή λειτουργία που παρέχουν μέχρι τώρα έχει αρχίσει και θεωρείται ελλιπής από πολλούς χρήστες. Στην πλειοψηφία τους τα συστήματα πλοήγησης οδηγούν κάποιο χρήστη από ένα σημείο Α σε ένα άλλο σημείο Β λαμβάνοντας υπόψη τους ένα πολύ μικρό αριθμό παραμέτρων τις οποίες μπορεί αυτός να τροποποιήσει. Αυτό έχει σαν αποτέλεσμα τα συστήματα πλοήγησης να χρησιμοποιούνται από το χρήστη ως λύση ανάγκης, μέχρι αυτός να αποκτήσει ιδία κρίση για την διαδρομή και να εφαρμόσει της δικές του προτιμήσεις σε ότι αφορά την επιλογή της διαδρομής. Με βάση τις παραπάνω παρατηρήσεις διαφαίνεται η ανάγκη υλοποίησης ενός συστήματος, το οποίο θα λαμβάνει υπόψη του παραμέτρους που μέχρι τώρα δεν υπάρχουν στα κλασσικά συστήματα πλοήγησης. Το σύστημα αυτό θα πρέπει να έχει σαν παραμέτρους εισόδου μεταβλητές που εκτός από το σημείο

εκκίνησης και σημείο προορισμού θα υποδηλώνουν τις προτιμήσεις του χρήστη για το είδος τις διαδρομής που θέλει να ακολουθήσει, για παράδειγμα θα υπάρχουν μεταβλητές σχετιζόμενες με την ασφάλεια της διαδρομής, τη πολυπλοκότητα ακόμα και το πόσο «ωραία» είναι η διαδρομή. Ο μηχανισμός εύρεσης της διαδρομής θα λαμβάνει υπόψη του τις μεταβλητές αυτές και θα τους αποδίδει ένα συγκεκριμένο βάρος σύμφωνα με παραμετροποίηση που θα έχει κάνει ο εκάστοτε χρήστης και στο τέλος θα καταλήγει στη πρόταση μιας διαδρομής που θα πλήρη όσο το δυνατόν περισσότερο τις ανάγκες του χρήστη.

1.3. Σκοπός μεταπτυχιακής μελέτης.

Σκοπός της μεταπτυχιακής μελέτης είναι η δημιουργία ενός συστήματος το οποίο θα επιτρέπει στον χρήστη να επιλέγει τη διαδρομή που θέλει να ακολουθήσει με βάση της παρακάτω παραμέτρους:

1. Είδος οχήματος
   - Άνθρωπος (με τα πόδια)
   - Μηχανή (μοτοσικλέτα)
   - Αγροτικό (φορτηγάκι)
   - Ιδιωτικής χρήσης επιβατικό όχημα
   - Ιδιωτικής χρήσης 4x4
   - Φορτηγό
   - Αγροτικό μηχάνημα (τρακτέρ)
2. Πλάτος δρόμου
3. Ποιότητα δρόμου
4. Κατάσταση οδοστρώματος
5. Εμφάνιση κλίσεων

Το προτεινόμενο σύστημα θα λαμβάνει υπόψη του τις παραπάνω παραμέτρους, με βαρύτητα την οποία θα επιλέγει ο χρήστης, και θα προτείνει τη διαδρομή που ταιριάζει περισσότερο στο προφίλ του.

1.4. Δομή μελέτης

Στο κεφάλαιο 2 παρουσιάζονται πληροφορίες που αφορούν στα Γεωγραφικά Πληροφοριακά Συστήματα (ΓΠΣ), τα είδη τους, καθώς και τα συστατικά στοιχεία τους. Στο κεφάλαιο 3 γίνεται μια ανασκόπηση της βιβλιογραφίας που αφορά στην πλοήγηση και του τρόπου που αντιμετωπίζονται τα αλγοριθμικά τα προβλήματα αυτού του τύπου. Στο κεφάλαιο 4

εξετάζονται οι παράμετροι μιας διαδρομής και οι επιπτώσεις που έχει καθένας από αυτούς στην ποιότητα της διαδρομής. Στο κεφάλαιο 5 περιγράφεται η αρχιτεκτονική του προτεινόμενου συστήματος, με την έννοια της σχεδίασης, ανάπτυξης και υλοποίησης του λογισμικού πλοήγησης καθώς και η διαδικασία υλοποίησης του ιστοτόπου που δημιουργήθηκε για την πρόσβαση χρηστών στην εφαρμογή και σε χάρτες περιοχών που έχουν διαμορφωθεί κατάλληλα ώστε να είναι συμβατοί με την εφαρμογή. Τέλος στο κεφάλαιο 6 δίνονται τα συμπεράσματα που προκύπτουν από την ανάλυση των αποτελεσμάτων και την χρήση του συστήματος.

2. Γεωγραφικά Πληροφοριακά Συστήματα

2.1. Γενικά

Τα Γεωγραφικά Συστήματα Πληροφοριών (ΓΣΠ) είναι πακέτα λογισμικού τα οποία έχουν την δυνατότητα συλλογής, αποθήκευσης, ανάκτησης, ανάλυσης και απεικόνισης των χωρικών δεδομένων (Wang, 2006) (Pamuk, 2006). Τα τελευταία χρόνια παρατηρείται θεαματική αύξηση του αριθμού, του μεγέθους, της πολυπλοκότητα και του εύρους των εφαρμογών GIS. Σήμερα, πολλές ιδιωτικές/κρατικές υπηρεσίες προσφέρουν ένα πλήθος υπηρεσιών που ενσωματώνουν τα ΓΣΠ στον σχεδιασμό στην ανάπτυξη και στην υλοποίηση των υπηρεσιών που προσφέρουν. Απτά παραδείγματα αποτελούν η Δασική υπηρεσία, το Κτηματολόγιο, κ.ά. έχουν άμεση σχέση με τα Γεωγραφικά Συστήματα Πληροφοριών (ΓΣΠ) και η αποτελεσματικότητα τους είναι άμεσα συνυφασμένη αυτά.

Ειδικότερα, τα τελευταία χρόνια τα ΓΣΠ έχουν γνωρίσει μεγάλη εξάπλωση και στον ιδιωτικό τομέα, καθώς οι δυνατότητες σύνδεσης της πληροφορίας που έχουν οι επιχειρήσεις τόσο για τους πελάτες τους όσο και για τα πάγια στοιχεία τους μπορεί πλέον να συνδυαστεί και με τη χωρική πληροφορία με αποτέλεσμα το ενιαίο σύστημα που θα προκύψει να βρει επιτυχημένη εφαρμογή τόσο σαν εργαλείο διαχείρισης και διοίκησης, όσο και σαν εργαλείο έρευνας αγοράς και προώθησης προϊόντων. Με τους απλούστερους όρους το ΓΣΠ είναι η συγχώνευση της Χαρτογραφίας και των Βάσεων δεδομένων.

Οι πιο συνηθισμένοι ορισμοί που δίνουμε στα ΓΣΠ είναι για να τα περιγράψουμε σαν ένα σύνολο εργαλείων, είτε σαν μία βάση δεδομένων, είτε τέλος σαν ένα σύστημα που αναπτύσσεται για τις ανάγκες ενός οργανισμού.:
Στην πρώτη περίπτωση ο Burrough (Burrough & McDonnell, 1996) ορίζει τα ΓΣΠ σαν ένα σύνολο εργαλείων για τη συλλογή, την αποθήκευση, την ανάκτηση, το μετασχηματισμό και την παρουσίαση χωρικών δεδομένων που προέρχονται από τον πραγματικό κόσμο και για ένα ορισμένο σύνολο σκοπών. Από μία άλλη οπτική πλευρά ο Aronoff (Aronoff, 1989) αντιμετωπίζει τα ΓΣΠ σαν μία βάση χωρικών δεδομένων, η οποία περιέχει όλες εκείνες τις διαδικασίες που απαιτούνται για την αποθήκευση και διαχείριση γεωαναφερμένων δεδομένων. Τέλος, μία τρίτη πτυχή των ΓΣΠ είναι η λειτουργία τους σε ένα περιβάλλον ενός οργανισμού όπου μιλάμε πλέον για ένα σύστημα λήψης αποφάσεων που περιλαμβάνει την ενσωμάτωση δεδομένων με σαφή τοποθεσία στο χώρο για την επίλυση ενός προβλήματος (Wiener, Schnee, & Mallot, 2004).

Το Γεωγραφικό Σύστημα Πληροφοριών (ΓΣΠ) είναι μια οργανωμένη συλλογή εξοπλισμού, λογισμικού, γεωγραφικών δεδομένων και προσωπικού, σχεδιασμένη έτσι ώστε να συγκεντρώνει, αποθηκεύει, ενημερώνει, επεξεργάζεται, αναλύει και παρουσιάζει όλους τους τύπους των γεωγραφικών δεδομένων. Συνδέει τις γεωγραφικές τοποθεσίες με πληροφορία (οντότητες χαρτών), παράγοντας θεματικούς χάρτες (με περιγραφικά δεδομένα), και έτσι την οπτικοποιεί και βοηθά στην ανάλυση της.

Το ΓΣΠ είναι ένα ολοκληρωμένο σύστημα χρήσιμο για τη διαδικασία λήψης αποφάσεων, με αποτέλεσμα να χρησιμοποιείται από πολλούς θεσμικούς φορείς τα τελευταία χρόνια. Η οπτικοποίηση της πληροφορίας που διαθέτει επιτυγχάνεται με την παραγωγή από αυτό διαγραμμάτων, χαρτών, πινάκων και αναφορών.

2.2. Είδη Γεωγραφικών συστημάτων πληροφοριών

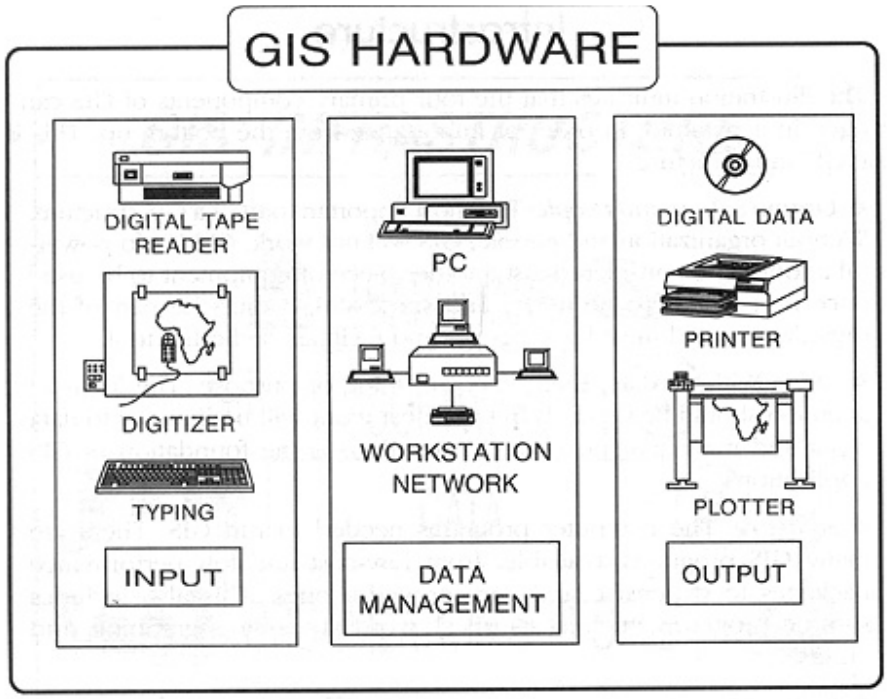
Υπάρχουν δύο είδη Γεωγραφικών Συστημάτων Πληροφοριών (ΓΣΠ), τα ψηφιδωτά που διαχειρίζονται δεδομένα όπως δορυφορικές εικόνες, αεροφωτογραφίες και άλλα είδη πλαισίων, στα οποία η πληροφορία συνδέεται με τα εικονοστοιχεία (pixels) των εικόνων και τα

διανυσματικά (Moreno, 2005) (Moldes, 1995) που διαχειρίζονται δεδομένα τα οποία οργανώνονται σε ψηφιακά υπόβαθρα γραμμών, πολυγώνων και σημείων.

2.3. Στοιχεία ενός Γεωγραφικού Συστήματος Πληροφοριών

2.3.1. Υποδομή Hardware

Η υποδομή Hardware αποτελείται συνήθως από τα αντικείμενα που εμφανίζονται στην Εικόνα 1



Εικόνα 1. Κύρια μέρη της υποδομής Hardware των GIS

Ένας ψηφιοποιητής (digitizer) ή ένα scanner μπορεί να χρησιμοποιηθεί για την εισαγωγή γεωγραφικών δεδομένων με την μετατροπή χαρτών καθώς και εγγράφων σε ψηφιακή μορφή. Ο υπολογιστής περιέχει σκληρό δίσκο για την αποθήκευση των δεδομένων αλλά επίσης μπορούν να χρησιμοποιηθούν το δίκτυο, τα DVD και οι εξωτερικοί σκληροί δίσκοι και άλλες συσκευές για επιπλέον χώρο. Για την αναπαράσταση των δεδομένων χρησιμοποιείται κάποιος σχεδιαστής, εκτυπωτής ή οποιαδήποτε άλλη συσκευή παρουσίασης.

2.3.2. Υποδομή λογισμικού

Η υποδομή λογισμικού των Γεωγραφικών Συστημάτων Πληροφοριών βασίζεται σε πέντε βασικά στοιχεία:

- Εισαγωγή δεδομένων και επαλήθευση
- Αποθήκευση και διαχείριση δεδομένων
- Παρουσίαση και εξαγωγή δεδομένων
- Μετασχηματισμός δεδομένων
- Αλληλεπίδραση με το χρήστη.

Η εισαγωγή δεδομένων και η επαλήθευση τους περιλαμβάνει τη συλλογή χωρικών δεδομένων από υπάρχοντες χάρτες, παρατηρήσεις στο πεδίο καθώς και με τη βοήθεια αισθητήρων (τηλεπισκόπηση, αεροφωτογραφίες και όργανα καταγραφής) αλλά και τη μετατροπή τους σε ψηφιακή μορφή.

Η αποθήκευση και διαχείριση δεδομένων επιτυγχάνεται με τη δημιουργία μίας βάσης χωρικών δεδομένων στην οποία η τοποθεσία, η τοπολογία και τα χαρακτηριστικά των γεωγραφικών στοιχείων όπως είναι οι γραμμές, τα πολύγωνα και τα σημεία που αναπαριστούν φαινόμενα του πραγματικού κόσμου συνδέονται με ένα οργανωμένο και δομημένο τρόπο.

Η αναπαράσταση των δεδομένων και της πληροφορίας έχει να κάνει με το πως παρουσιάζονται τα αποτελέσματα της γεωγραφικής ανάλυσης στους τελικούς αποδέκτες. Τα αποτελέσματα μπορούν να παρουσιάζονται με τη μορφή χαρτών, πινάκων ή και διαγραμμάτων και αναλόγως την περίπτωση σε αναλογική ή ψηφιακή μορφή.

Ο μετασχηματισμός των δεδομένων περιλαμβάνει κυρίως όλες εκείνες τις διαδικασίες που χρησιμοποιούνται για την διόρθωση λαθών στα δεδομένα ή για την ενημέρωση τους ώστε να εναρμονίζονται με τα υπόλοιπα γεωγραφικά δεδομένα καθώς και όλες εκείνες τις μεθόδους και λειτουργίες που εφαρμόζουμε για να απαντήσουμε ερωτήσεις μέσω των GIS. Οι περισσότερο γνωστοί και δημοφιλείς μετασχηματισμοί περιέχουν τις λειτουργίες αλλαγής κλίμακας και προβολής ή συστήματος συντεταγμένων, ανάκτηση δεδομένων με χρήση λογικών εκφράσεων (π.χ Boolean algebra) καθώς και υπολογισμούς επιφανειών και περιμέτρων.

Παρόλα ταύτα οι τρόποι μετασχηματισμού γεωγραφικών δεδομένων είναι απεριόριστοι καθώς ο χρήστης μπορεί να δημιουργήσει τους δικούς του ώστε να ικανοποιήσει τις ανάγκες της γεωγραφικής ανάλυσης.

Οι μηχανικοί λογισμικού των Γεωγραφικών Συστημάτων Πληροφοριών έχουν αναπτύξει στα περισσότερα πακέτα λογισμικού Γεωγραφικών Συστημάτων Πληροφοριών, επιφάνειες αλληλεπίδρασης (interfaces) με τους χρήστες ώστε να μπορούν να επιδράσουν στο σύστημα. Αυτό συνήθως επιτυγχάνεται με τη χρήση των μενού όμως για τους απαιτητικούς χρήστες των ΓΠΣ υπάρχει και η επιλογή του διερμηνευτή εντολών (CLI: Command Line Interpreter), όπου ο χρήστης αλληλεπιδρά με το λογισμικό γράφοντας τις εντολές που θέλει να εκτελεστούν ελέγχοντας με αυτόν τον τρόπο το σύστημα απόλυτα. Φυσικά δεν είναι δυνατόν να εκτελεστούν όλες οι λειτουργίες που επιθυμούν οι χρήστες μέσω αυτού του τρόπου αλληλεπίδρασης, για αυτό το λόγο τα περισσότερα συστήματα GIS παρέχουν τη δυνατότητα δημιουργίας προγραμμάτων, ώστε να προσαρμόζεται το λογισμικό στις ανάγκες των χρηστών μας με χρήση γλωσσών προγραμματισμού όπως είναι η Vbscript, Javascript, Python, C# κτλ.

2.3.3. Μέθοδοι εφαρμογής - Ειδικευμένο προσωπικό

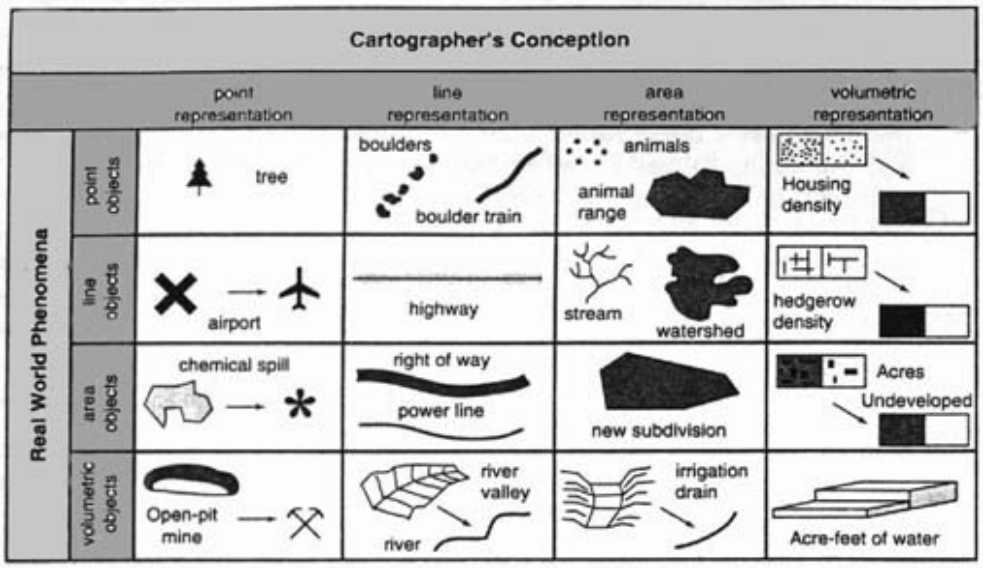
Οι παραπάνω υποδομές αν και σημαντικές παραμένουν περιορισμένης αξίας αν δεν χρησιμοποιούνται από ειδικευμένο προσωπικό στα Γεωγραφικά Συστήματα Πληροφοριών, ή αν δεν υπάρχει οριοθετημένο αντικείμενο εφαρμογής και μεθοδολογία για τη χρήση τους. Η συγκεκριμένη πτυχή στο χώρο των GIS έχει παραμεληθεί αρκετά τα τελευταία χρόνια παρόλα ταύτα αποτελεί αναγκαίο κομμάτι για τη σωστή και αποδοτική λειτουργία ενός ολοκληρωμένου Γεωγραφικού Συστήματος Πληροφοριών. Οι μέθοδοι αποτελούν εκείνα τα διαχειριστικά πρότυπα και τους κανόνες που απαιτούνται για την εφαρμογή της τεχνολογίας των GIS και την επίλυση του προβλήματος. Κάποια από αυτά είναι de facto και χρησιμοποιούνται για το μεγαλύτερο εύρος των εφαρμογών ενώ άλλα πρέπει να προσαρμοστούν ανάλογα το πρόβλημα, την περιοχή, τον οργανισμό και το είδος των δεδομένων. Το ειδικευμένο προσωπικό μπορεί να παίξει το ρόλο του διαχειριστή του συστήματος καθώς και του ειδικού ανάπτυξης σχεδίων για υλοποίηση. Οι άνθρωποι που απαιτούνται για την λειτουργία της τεχνολογίας των Γεωγραφικών Συστημάτων Πληροφοριών μπορούν να είναι τεχνικοί υπολογιστών, μάνατζερ, τεχνικοί GIS, ειδικοί εφαρμογών, αναλυτές αγοράς κτλ.

2.3.4. Δεδομένα

Τα γεωγραφικά ή χωρικά δεδομένα που διαχειρίζεται ΓΣΠ είναι οι οντότητες (χωρική πληροφορία) και οι πίνακες γνωρισμάτων τους (ποιοτικές και ποσοτικές ιδιότητες των χωρικών οντοτήτων). Χωρικές γεωγραφικές οντότητες είναι τα καθορισμένα γεωμετρικά σχήματα που έχουν συγκεκριμένη γεωγραφική θέση στο φυσικό κόσμο και απεικονίζονται στους χάρτες σύμφωνα με αυτή. Οι χωρικές γεωγραφικές οντότητες κατηγοριοποιούνται ως ακολούθως (Kao, 1968):

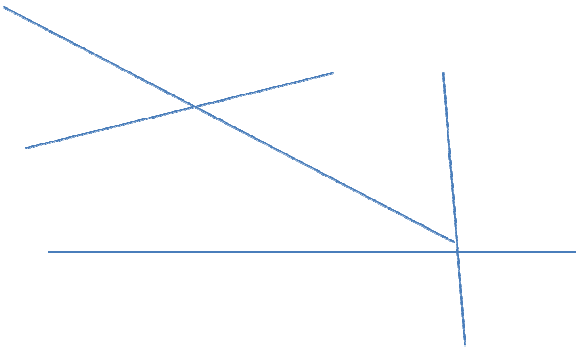
1. **Στοιχεία γραμμών**: Τιμές που παρατηρούνται για τη σύνδεση δυο σημείων στο χώρο.
2. **Στοιχεία Σημείων**: Τιμές που παρατηρούνται σε ορισμένα σημεία του χώρου.
3. **Στοιχεία Επιφανειών**: Τιμές για κάποια γεωγραφική μονάδα σαν σύνολο .

Οι οντότητες διαχωρίζονται σε χαρτογραφικά υπόβαθρα ανάλογα με το είδος τους. Βασική αρχή σε ένα Γεωγραφικό σύστημα πληροφοριών (ΓΣΠ) είναι το γεγονός ότι δεν μπορούν να συνυπάρχουν δύο ή περισσότερα είδη οντοτήτων στο ίδιο χαρτογραφικό υπόβαθρο. Ως *χαρτογραφικό υπόβαθρο* (layer) ορίζεται ο χάρτης (συνήθως σε ψηφιακή μορφή) που περιλαμβάνει αποτυπωμένες τις γεωγραφικές οντότητες του ιδίου είδους της περιοχής που παρουσιάζει.



Εικόνα 2. Παραδείγματα απεικόνισης γεωγραφικών οντοτήτων

Ως χαρακτηριστικά-ιδιότητες των οντοτήτων αναφέρονται οι περιγραφικές πληροφορίες που σχετίζονται με τις οντότητες των χαρτογραφικών υποβάθρων. Τα χαρακτηριστικά-ιδιότητες των χωρικών οντοτήτων αποθηκεύονται σε πίνακες στους οποίους κάθε γραμμή του πίνακα αντιπροσωπεύει μια χωρική οντότητα και κάθε στήλη αντιπροσωπεύει ένα χαρακτηριστικό-ιδιοτητα της χωρικής οντότητας (Εικόνα 3) Κάθε πίνακας σε ένα ΓΣΠ με περιγραφική πληροφορία συνδέεται με ένα θεματικό χαρτογραφικό υπόβαθρο.



Α/Α	Τύπος Δρόμου	Υλικό	Μήκος
1			
2	5	Άσφαλτός	500

Εικόνα 3. Παραδείγματα απεικόνισης γεωγραφικών οντοτήτων

Τοπολογία εννοούμε το σύνολο των γεωμετρικών κανόνων που πρέπει να ακολουθεί η γεωγραφική πληροφορία ανάλογα με την φύση της. Έτσι για παράδειγμα, εάν η πληροφορία είναι τα οικοδομικά τετράγωνα τότε τα πολύγωνα που τα αναπαριστούν θα πρέπει να ακολουθούν μεταξύ άλλων τους κανόνες. Δεν επιτρέπεται η αλληλοεπικάλυψη, δεν επιτρέπεται η ταύτιση των ορίων. Σε άλλες περιπτώσεις και για την ίδια γεωγραφική περιοχή ο κανόνας μπορεί να ισχύει αντίθετα. Για τους χάρτες, η τοπολογία προσδιορίζει σχέσεις ανάμεσα σε οντότητες, αναγνωρίζει γειτονικά πολύγωνα και μπορεί να προσδιορίσει μια οντότητα, όπως μια περιοχή, σαν σύνολο άλλων.

2.4. Αρχιτεκτονικές εφαρμογών

Η ανάλυση μέσω των Γεωγραφικών Συστημάτων Πληροφοριών (ΓΣΠ) γίνεται με τέσσερις τρόπους. Δηλαδή υπάρχουν τέσσερα είδη αρχιτεκτονικών για τις υλοποιήσιμες εφαρμογές Γεωγραφικά Συστήματα Πληροφοριών (ΓΣΠ) (Εικόνα 4).

1. Προγράμματα ΓΣΠ

   Όλα τα δεδομένα βρίσκονται σε ένα Η/Υ, στον οποίο είναι εγκατεστημένο το λογισμικό. Τοπικά γίνεται η χαρτογράφηση αλλά και οι αναλύσεις. Οι δυνατότητες είναι περιορισμένες όσον αφορά στην αντιμετώπιση πολύπλοκων προβλημάτων που απαιτούν πολλά είδη δεδομένων και αυξημένη υπολογιστική ισχύ.

2. Υπηρεσιακά ΓΣΠ

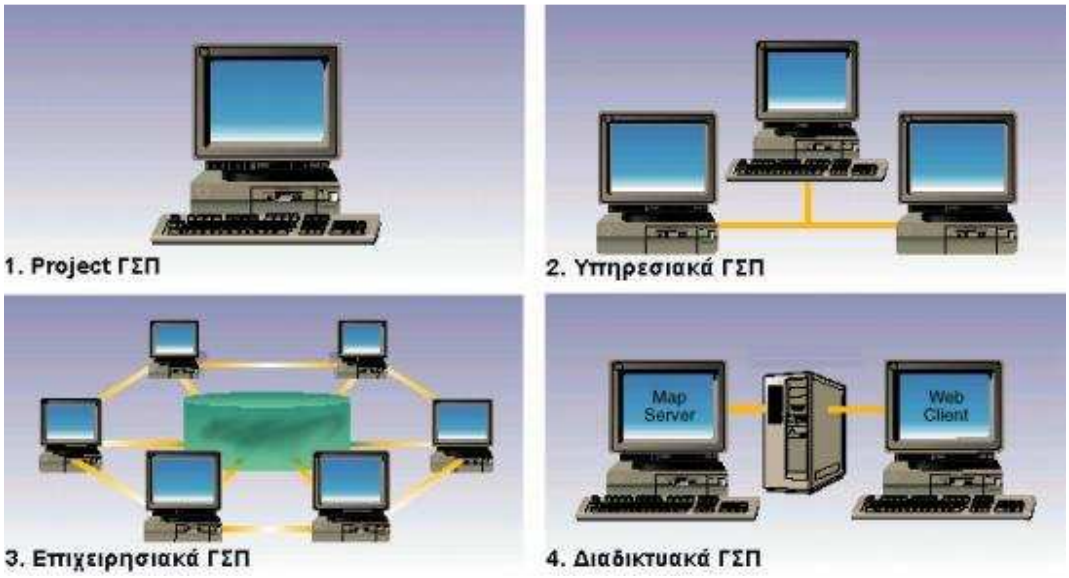
   Όλα τα δεδομένα είναι διαθέσιμα σε πολλούς χρήστες με πρόσβαση από διαφορετικούς Η/Υ στην κλίμακα του τμήματος της επιχείρησης. Υπάρχει ολοκλήρωση και ενοποίηση των δεδομένων από τα διαφορετικά τμήματα της επιχείρησης. Τα δεδομένα διατηρούνται και διαχειρίζονται κεντρικά. Τέλος, πολλά είδη εφαρμογών χρησιμοποιούνται για τις επιμέρους ανάγκες των τμημάτων.

3. Επιχειρησιακά ΓΠΣ

   Όλα τα σύνολα δεδομένων διαχειρίζονται ως η βασική πηγή για παραγωγή πληροφορίας στην κλίμακα της επιχείρησης. Αποθηκεύονται σε μια κεντρική βάση δεδομένων και είναι προσπελάσιμα μέσω ενός κεντρικού συστήματος διαχείρισης. Οι εφαρμογές ΓΣΠ παίζουν το ρόλο του πελάτη και λαμβάνουν τα δεδομένα από την κεντρική βάση δεδομένων (που βρίσκεται στον εξυπηρέτη) για τις αναλύσεις. Το τελικό αποτέλεσμα είναι η δημιουργία ενός ΓΣΠ αρχιτεκτονικής πελάτη/εξυπηρέτη.

4. Διαδικτυακά ΓΣΠ

   Τα δεδομένα βρίσκονται σε κεντρικό εξυπηρέτη όπου υπάρχει κατάλληλο λογισμικό που επιτρέπει την πρόσβαση μέσω του Παγκόσμιου Ιστού.



Εικόνα 4. Αρχιτεκτονικές ΓΣΠ

3. Πλοήγηση

Οι ερευνητές May, Ross και Bayer (May, Ross, & Bayer, 2003) αποδέχονται το γεγονός ότι η πλοήγηση σε μη οικείο περιβάλλον αποτελεί μια καθημερινή ανθρώπινη δραστηριότητα που απαιτεί έντονη γνωσιακή αντιληπτική δράση (Παρατήρηση - Ταύτιση - Μίμηση - Αφομοίωση). Για να αντιμετωπιστούν οι όποιες δυσκολίες πρέπει να γίνει αντιληπτό το όλο πλαίσιο πλοήγησης και να διαπιστωθούν οι ανάγκες των οδηγών για παρεχόμενη πληροφορία. Σκοπός της έρευνας των προαναφερομένων ερευνητών είναι να αναγνωρίσουν τι είδους πληροφορία χρησιμοποιούν οι οδηγοί κατά την πλοήγηση, εντός και εκτός αστικών οδικών δικτύων, να αντιληφθούν τον τρόπο χρήσης – αξιοποίησης αυτής της πληροφορίας και να προσδιορίσουν τις προδιαγραφές ενός δυνητικού άριστου βοηθήματος πλοήγησης.

3.1. Βιβλιογραφική ανάλυση πλοήγησης

Είναι κοινά αποδεκτό ότι η πλοήγηση σε ξένο περιβάλλον είναι δύσκολη. Η παρουσία εξωτερικών παραγόντων μπορεί να περιπλέξει την κατάσταση ακόμη περισσότερο και να δημιουργήσει περαιτέρω προβλήματα (για παράδειγμα, η ανεπάρκεια χώρου για να διενεργεί

τους απαραίτητους ελέγχους ο οδηγός του οχήματος επιτείνει το πρόβλημα πλοήγησης). Από την στιγμή που ο πλοηγούμενος δεν είναι να θέση να ακολουθήσει την βέλτιστη διαδρομή αισθάνεται άγχος και εκνευρισμό, ενώ η ενδεχόμενη χρονική καθυστέρηση μπορεί να προκαλέσει λάθη πλοήγησης με αποτέλεσμα η οδήγηση να καθίσταται επισφαλής. Οι κύριες ανάγκες πλοήγησης σταχυολογούνται παρακάτω:

1. Η σχεδίαση μιας διαδρομής.
2. Η αναγνώριση δυνητικών σημείων επιλογής που επηρεάζουν την πλοήγηση.
3. Η επεξεργασία και πρόκριση δράσεων – κινήσεων μέσω μη επιβεβαιωμένων, ή ασαφών επιλογών.
4. Η επιβεβαίωση των σωστών κινήσεων πλοήγησης.
5. Η διατήρηση εμπιστοσύνης ως προς την ορθότητα της πορείας.
6. Η ανάπτυξη της αντίληψης προσανατολισμού, όπου αυτή απαιτείται.
7. Η αναγνώριση – επιβεβαίωση της άφιξης στο τέλος της διαδρομής.

Αναφορικά με την ανάλυση του φαινόμενου της πλοήγησης, έχουν αναπτυχθεί πολλά εναλλακτικά πρότυπα:

1. Ο Burns (Burns, 1997) περιγράφει το πρότυπο του το οποίο βασίζεται στον γνωσιακό χάρτη (οι γνωσιακοί χάρτες, τους οποίους χρησιμοποιούν οι άνθρωποι για να προβλέψουν, να προσαρμοστούν και να ελέγξουν τον κόσμο γύρω τους, βασίζονται σε γνωσιακές κατασκευές ή σκέψεις με τις οποίες ερμηνεύουν την πραγματικότητα) του οδηγού, στη μνήμη του (βραχυπρόθεσμη και μακροπρόθεσμη) καθώς και στην αντίληψη του για το περιβάλλον (την πληροφορία που μπορεί να εξάγει από τον περίγυρό του)
2. Ο Zhai (Zhai, 1991) αναφέρεται σε ένα πρότυπο συμπεριφοράς το οποίο λαμβάνει υπ' όψη την διαδραστικότητα μεταξύ του οδηγού, του συστήματος πλοήγησης, του οχήματος και του περιβάλλοντος (η προσέγγιση του Zhai είναι περιπλοκότερη και άρρηκτα συνδεδεμένη με τις σημερινές τεχνολογικές εξελίξεις).
3. Ο Mark (Mark, 1989) εισάγει στο πρότυπο του έννοιες όπως η γεωγραφική βάση δεδομένων, η σχετική θέση οχήματος και σημείου άφιξης, ο σχεδιασμός του διαδρομής, παραγωγή εντολών και έλεγχο οχήματος

4. Οι Burnett και Porter (Burnett & Porter, 2002) εστιάζουν στην παροδική φύση της πλοήγησης και αναγνωρίζουν τις βασικές προδιαγραφές πληροφορίας, για κάθε μια φάση πλοήγησης.

3.2. Εύρεσης συντομότερης διαδρομής

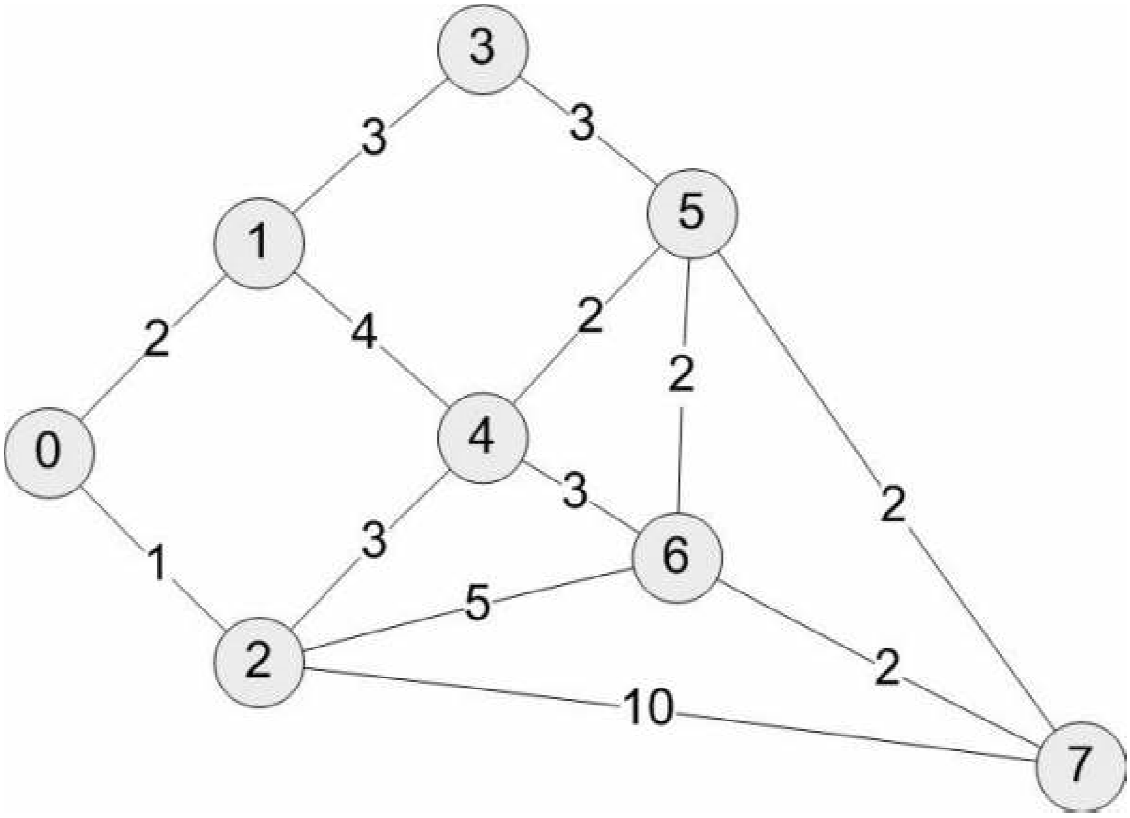
Το πρόβλημα εύρεσης της συντομότερης διαδρομής μαζί με τα προβλήματα βέλτιστης κάλυψης και μέγιστης ροής είναι ένα από τα πιο θεμελιώδη προβλήματα βελτιστοποίησης του δικτύου, όπου δίκτυο ορίζεται ένα σύνολο κόμβων (σημείων) τα οποία συνδέονται μεταξύ τους με συγκεκριμένα βάρη. Αυτό το πρόβλημα εμφανίζεται συχνά στην πράξη και αποτελεί τμήμα σχεδόν όλων των πακέτων λογισμικού δρομολόγησης. Πολλοί αλγόριθμοι βελτιστοποίησης εύρεσης της συντομότερης διαδρομής σε ένα δίκτυο έχουν μελετηθεί εδώ και αρκετά χρόνια (Ahuja, Mehlhorn, Orlin, & Tarjan, 1988) (Bellman, 1958) (Dial, 1969) (Dijkstra E. , 1959) (Ford Jr & Fulkerson, 1962) (Fredman & Tarjan, 1987) (Gabow & Tarjan, 1989) (Gallo & Pallottino, 1988) (Goldberg, 1993). Ωστόσο, η πρόοδος στις μεθόδους και στη θεωρία αλγορίθμων για την εύρεση βέλτιστων διαδρομών σε δίκτυα είναι μια διαδικασία σε συνεχή εξέλιξη μονοπάτια εξακολουθούν να γίνονται (Blivice, 1974) (Hill, 1984).

Οι περισσότεροι αλγόριθμοι για την εύρεση της συντομότερης διαδρομής βασίζονται σε ελαφρές τροποποιήσεις (Cantone & Faro, 2004) αλγόριθμου του Dijkstra (Dijkstra E. , 1959) ο όποιος έχει χρήση σε πολλές εφαρμογές (Kim, Lee, Oh, & Choi, 2009) (Devlin, McDonnell, & Ward, 2008)

3.3. Ψευδογράφημα

Ψευδογράφημα (Εικόνα 4) είναι ένα σύνολο κόμβων (κορυφών) και ένα σύνολο ακμών, τέτοιο ώστε κάθε ακμή συνδέει ένα κόμβο με έναν άλλο κόμβο.
Στα ψευδογραφήματα μπορεί να συμβαίνουν δύο καταστάσεις που δεν συμβαίνουν στα γραφήματα:

1. Μπορεί να έχουν πολλαπλές συνδέσεις.
2. Είναι δυνατόν να υπάρχουν βρόγχοι.



Εικόνα 4. Ψευδογράφημα

3.4. Ο αλγόριθμος Dijkstra

Ο αλγόριθμος του E.W. Dijkstra είναι ένας αλγόριθμος ο οποίος χρησιμοποιείται για τον υπολογισμό τη ελάχιστης διαδρομής ανάμεσα σε μια κορυφή και σε όλες τις άλλες σε συνεκτικά ψευδογραφήματα (Dijkstra E. W., 1959). Ένα ψευδογράφημα χαρακτηρίζεται συνεκτικό αν κάθε ζεύγος κορυφών του μπορεί να συνδεθεί με μία διαδρομή.

3.4.1. Λειτουργία του αλγορίθμου

1. Αρχικά, θα πρέπει να ορίσουμε ότι για κάθε ζεύγος κορυφών **v** και **u** του ψευδογραφήματος, w(**v**,**u**) είναι το ελάχιστο βάρος μιας ακμής που ενώνει τις κορυφές **v** και **u**.
2. Επιπλέον πρέπει να δημιουργήσουμε τρεις λίστες οι οποίες θα αποθηκεύουν τις αποστάσεις από την αρχική κορυφή, τις προηγούμενες κορυφές, τις ελεγμένες κορυφές αντίστοιχα.

3. Θα πρέπει επίσης να κρατούμε και την τρέχουσα κορυφή για την οποία γίνεται ο έλεγχος αποστάσεων.
4. Οι τιμές στην λίστα που αποθηκεύει τις αποστάσεις θα πρέπει αρχικά να τεθούν ίσες με άπειρο, εκτός από την αρχική κορυφή που θα πρέπει να έχει τιμή μηδέν.
5. Η λίστα με τις ελεγμένες κορυφές θα πρέπει να είναι κενή, ενώ η λίστα με τις προηγούμενες κορυφές θα πρέπει να έχει τιμές NULL δηλαδή μη ορισμένες.
6. Τέλος, θα πρέπει να ορίσουμε ότι αρχικά η τρέχουσα κορυφή θα είναι η κορυφή από την οποία έχουμε υποδείξει ότι θα ξεκινήσει ο αλγόριθμος. Στη συνέχεια ακολουθεί το κυρίως κομμάτι του αλγορίθμου του Dijkstra.

3.4.2. Κύριο τμήμα του αλγορίθμου

1. Προσθέτουμε την τρέχουσα κορυφή στη λίστα με τις ελεγμένες κορυφές
2. Ενημερώνουμε τις λίστες των αποστάσεων και των προηγούμενων κορυφών, βάσει των γειτονικών κορυφών της τρέχουσας κορυφής
3. Υπολογίζουμε το τρέχον ελάχιστο μονοπάτι από την αρχική κορυφή προς την αντίστοιχη γειτονική κορυφή της τρέχουσας κορυφής που να μην περιέχεται στην λίστα με τις ελεγμένες κορυφές
4. Αντικαθιστούμε την τρέχουσα κορυφή με την κορυφή στην οποία αντιστοιχεί το ελάχιστο μονοπάτι
5. Επανάληψη του βρόχου έως που όλες οι κορυφές να περιέχονται στην λίστα των ελεγμένων κορυφών

3.4.3. Ψευδοκώδικας

Στον παρακάτω αλγόριθμο, η εντολή u := extract_min(Q) ψάχνει για την κορυφή u στο σύνολο κορυφών Q, η οποία έχει τη μικρότερη τιμή dist[u]. Η κορυφή αυτή αφαιρείται από το σύνολο Q και επιστρέφεται στον χρήστη. Η length(u, v) υπολογίζει το μήκος μεταξύ των δύο γειτονικών κόμβων u και v. Η μεταβλητή alt στη γραμμή 10 είναι το μήκος της διαδρομής από τον αρχικό κόμβο στο γειτονικό κόμβο v αν περνούσε μέσω του u. Αν η διαδρομή αυτή είναι μικρότερη από την μικρότερη διαδρομή που έχει καταγραφεί μέχρι στιγμής για τον v, τότε η τρέχουσα

αυτή διαδρομή αντικαθίσταται από τη διαδρομή alt. Ο πίνακας previous αποτελείται από ένα δείκτη για τον κόμβο επόμενου βήματος πάνω στο γράφημα προκειμένου να ληφθεί η συντομότερη διαδρομή για τον αρχικό κόμβο (Wikipedia).

```
1 function Dijkstra(Graph, source):
2     for each vertex v in Graph:          // Αρχικοποίηση
3         dist[v] := infinity              // Άπειρη αρχική απόσταση από τη πηγή μέχρι το σημείο το v
4         previous[v] := undefined
5         dist[source] := 0                // Απόσταση από το αρχικό σημείο μέχρι το αρχικό σημείο
6         Q := copy(Graph)                 // Όλα τα σημεία δεν έχουν βελτιστοποιηθεί - οπότε και
                                           ανήκουν στο σύνολο Q
7         while Q is not empty:
8             u := extract_min(Q)          // Αφαίρεσε και δώσε σαν αποτέλεσμα το βέλτιστο
                                                                vertex από τα σημεία στο σύνολο Q
9             for each neighbor v of u:    // Έλεγξε εάν το v έχει εξεταστεί
10                alt = dist[u] + length(u, v)
11                if alt < dist[v]         // Έλεγξε την απόσταση
12                dist[v] := alt
13                previous[v] := u
14    return previous[]
```

4. Παράμετροι πλοήγησης

4.1. Παράγοντες επιλογής διαδρομής

Στην επικοινωνία ανθρώπου υπολογιστή σε ένα σύστημα πλοήγησης οι ιδιότητες του περιβάλλοντος έχουν αρχίσει να αποκτούν μεγάλο ενδιαφέρον (Cheverst, Davies, Mitchell, Friday, & Efstratiou, 1998) (Li, 2006) (Oulasvirta, Kurvinen, & Kankainen, 2003) (Paay, 2003)
Για να προσδιοριστεί η ποιότητα μιας συγκεκριμένης διαδρομής, διάφορες ιδιότητες διαδρομών πρέπει να ληφθούν υπόψη.

1. **Ευκολία**: Η ευκολία μιας διαδρομής σχετίζεται από πολλούς διαφορετικούς παράγοντες. Η ευκολία μιας διαδρομής να έχει σχέση με την απόσταση, την ανηφόρα, τη συχνότητα χρήσης, την ευρύτητα της πορείας ή περιβαλλοντικές συνθήκες όπως ο καιρός.
2. **Ασφάλεια**: Στην ασφάλεια μιας διαδρομής περιλαμβάνονται οι πτυχές που προκαλούν αισθήματα ανασφάλειας στους ανθρώπους όπως για παράδειγμα εάν διαδρομή είναι σε δρόμο κοντά σε γκρεμό.
3. **Απλότητα**: Λαμβάνοντας υπόψη την επιλογή μεταξύ μιας πιο σύντομης χρονικά αλλά πιο σύνθετης διαδρομής και μίας μεγαλύτερης χρονικής διάρκειας αλλά απλούστερης διαδρομής, οι περισσότεροι χρήστες θα επιλέξουν την πιο απλή διαδρομή. Η απλότητα μιας διαδρομής εξαρτάται από τον αριθμό των σημείων απόφασης που χρειάζεται να πάρει ο οδηγός κατά μήκος της διαδρομής.

Φυσικές παράμετροι	**Ψυχολογικές παράμετροι**	**Νοητικοί παράμετροι**
Απόσταση	Ασφάλεια	Πολυπλοκότητα
Υψομετρικές διαφορές		
Ιδιότητες δρόμου (πλάτος, επίστρωση κτλ)		
Κλιματολογικές συνθήκες		

Πίνακας 1. Παράμετροι ποιότητας διαδρομής

4.2. Επιρροή των φυσικών παραμέτρων στην ποιότητα διαδρομών

Επιρροή των φυσικών παραμέτρων στην ποιότητα διαδρομών (Πίνακας 1)

1. **Απόσταση:** Το μήκος μιας δεδομένης διαδρομής είναι ένας κρίσιμος παράγοντας κατά την υπολογισμό της ποιότητας διαδρομών. Η απόσταση που καλύπτεται είναι ένας από τους κρίσιμους παράγοντες που καθορίζουν εάν ένας προορισμός επιτυγχάνεται με το περπάτημα ή εάν άλλοι τρόποι μεταφοράς επιλέγονται. Σε κανονικές περιστάσεις, ένας χρήστης θα επιλέξει την μικρότερη σε απόσταση διαδρομή ανάμεσα από ένα σύνολο

δυνατών διαδρομών για να φθάσει σε έναν επιθυμητό προορισμό. Η προθυμία να διανυθεί μια ορισμένη απόσταση ποικίλλει χωριστά. (Blivice, 1974) (Sorrows & Hirtle, 1999)

2. **Υψομετρικές διαφορές:** Αν και οι άνθρωποι προσπαθούν να αποφεύγουν τις μακρύτερες διαδρομές, σε μερικές περιπτώσεις οι μακρύτερες διαδρομές προτιμώνται εάν η συντομότερη διαδρομή είναι απότομη. Οι διαφορές στο επίπεδο κλίσεις είναι επίσης κρίσιμοι παράγοντες των ιδιοτήτων διαδρομών (Wiener, Schnee, & Mallot, 2004).
3. **Ιδιότητες δρόμου:** Μια από τις βασικές απαιτήσεις των χρηστών για την ποιότητα μίας διαδρομής είναι η διαδρομή να παρουσιάζει αρκετό χώρο για να ελιχθούν χωρίς να υπάρχει κίνδυνος μεγάλη πιθανότητα διασταύρωσης της πορείας τους με την πορεία άλλου χρήστη. Ο χώρος που διαθέτει ο χρήστης για να είναι σημαντικός παράγοντας της ποιότητας της διαδρομής. (Shriver, 1997)

4.3. Επιρροή των ψυχολογικών παραμέτρων στην ποιότητα διαδρομών

Ασφάλεια: Η ασφάλεια είναι ένας πολύ κρίσιμος παράγοντας της ποιότητας διαδρομών. Τα συναισθήματα της ταλαιπωρίας και της αβεβαιότητας μπορούν να οδηγήσουν τους χρήστες να αποφύγουν συγκεκριμένες διαδρομές (Millonig, 2005). Μια διαδρομή που παρουσιάζει κακή ποιότητα από πλευρά ασφάλειας θεωρείται από τους χρήστες λιγότερο επιθυμητή από μια πιο ασφαλή διαδρομή. Διαδρομές που παρουσιάζουν μικρή ορατότητα και δυνατότητα έλεγχου για την πορεία που ακολουθούν άλλοι χρήστες, που δεν έχουν γίνει τα κατάλληλα έργα όπως η τοποθέτηση μπαρών ανάμεσα σε δυο αντίθετα ρεύματα πορείας, που υπάρχει λάθος κλίση ή ανεπαρκής κλίση του οδοστρώματος στις στροφές προκαλούν στους χρήστες αισθήματα ανασφάλειας που λειτουργούν αρνητικά για την επιλογή μιας διαδρομής που τα παρουσιάζει.

4.4. Επιρροή των νοητικών παραμέτρων στην ποιότητα διαδρομών

Πολυπλοκότητα: Οι χρήστες κατά την επιλογή και κατά τη διάρκεια της πλοήγησης προσπαθούν να ελαχιστοποιήσουν το μέγεθος της διανοητικής εργασίας που θα χρειαστεί να αφιερώσουν. Αυτό έχει σαν αποτέλεσμα, οι διαδρομές που προσφέρουν ένα μικρότερο αριθμό σημείων απόφασης να προτιμώνται από άλλες διαδρομές ακόμα κι αν μια εναλλακτική

διαδρομή είναι κοντύτερη. Όσο λιγότερες πληροφορίες χρειάζεται να αναφερθούν, τόσο μικρότερος είναι ο κίνδυνος να παρεκκλίνει κάποιος από τη πορεία του. Συνεπώς η απλούστερη πορεία προσφέρει μια υψηλή ποιότητα, δεδομένου ότι και οι διανοητικές προσπάθειες και ο φόβος είναι μειωμένοι. Η απλότητα μιας διαδρομής καθορίζεται από τον αριθμό σημείων απόφασης που απαιτούνται και την πολυπλοκότητα των διατομών που συναντάει ο χρήστης (Grum, 2005).

4.5. Παράμετροι συστήματος πλοήγησης

Προκειμένου να είναι χρήσιμο ένα σύστημα πλοήγησης πρέπει να πλήρη τις ακόλουθες προϋποθέσεις:

1. Ο χρήστης θα πρέπει να υποστηρίζεται στην πλοήγηση παρά να οδηγείται σε μια συγκεκριμένη θέση.
2. Ο χρήστης θα πρέπει να μην εστιάζει στην αλληλεπίδραση με τη συσκευή αλλά στον αρχικό στόχο .
3. Ο χειρισμός πρέπει να είναι απλός και κατανοητός: Για να διευκολύνει το χειρισμό το σύστημα πρέπει να διαλέγει αυτόματα τις κατάλληλες παραμέτρους αφήνοντας όμως στο χρήστη την ευθύνη για το σύστημα πλοήγησης.
4. Απλή και κατανοητή απεικόνιση: Η χαρτογράφηση μεταξύ της απεικόνισης και της πραγματικότητας πρέπει να είναι εύκολη και γρήγορη καθώς και ακριβής.

5. ΥΛΙΚΑ ΚΑΙ ΜΕΘΟΔΟΙ

5.1. Απαιτήσεις χρηστών του συστήματος πλοήγησης

Οι υποστηριζόμενοι από το σύστημα χρήστες είναι (Εικόνα 5)

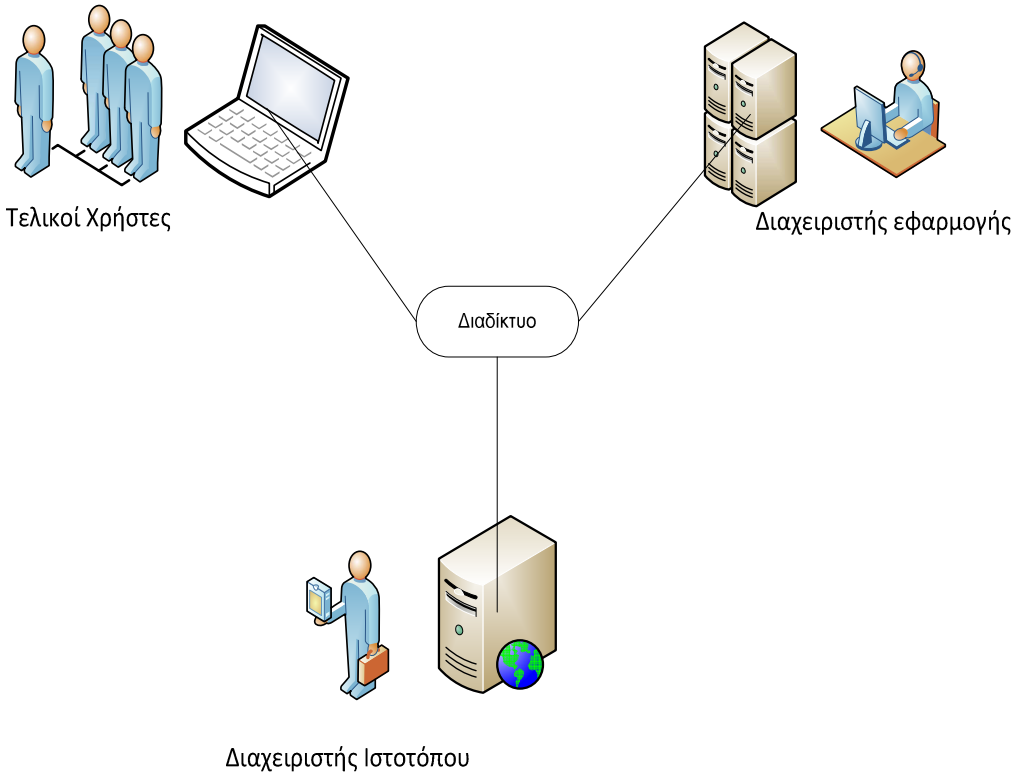
1. Ο διαχειριστής της ιστοσελίδας
2. Ο διαχειριστής της εφαρμογής
3. Ο τελικός χρήστης.

Ειδικότερα:

1. **Διαχειριστής της ιστοσελίδας**: Ο διαχειριστής της ιστοσελίδας είναι υπεύθυνος για την ομαλή λειτουργία του ιστοτόπου, φροντίζει για την προσθήκη και την ανανέωση νέων

χαρτών και είναι υπεύθυνος για την εγγραφή των τελικών χρηστών στον ιστοτόπο, ώστε αυτοί να αποκτήσουν πρόσβαση στους χάρτες και στο λογισμικό πλοήγησης.

2. **Διαχειριστής της εφαρμογής**: Ο διαχειριστής της εφαρμογής είναι υπεύθυνος για το σχεδιασμό νέων πακέτων χαρτών, για τοποθεσίες στις οποίες δεν υπήρχε δυνατότητα πλοήγησης, καθώς και για την ενημέρωση των υπαρχόντων χαρτών με μελλοντικές πληροφορίες.
3. **Τελικός χρήστης**: Ο τελικός χρήστης είναι ο χρήστης της εφαρμογής πλοήγησης, ο όποιος έχει την μεγαλύτερη επαφή με την εφαρμογή, και ενδιαφέρεται για την εύρεση της βέλτιστης διαδρομής με βάση παραμέτρους που καθορίζει αυτός.



Εικόνα 5. Χρήστες του συστήματος πλοήγησης

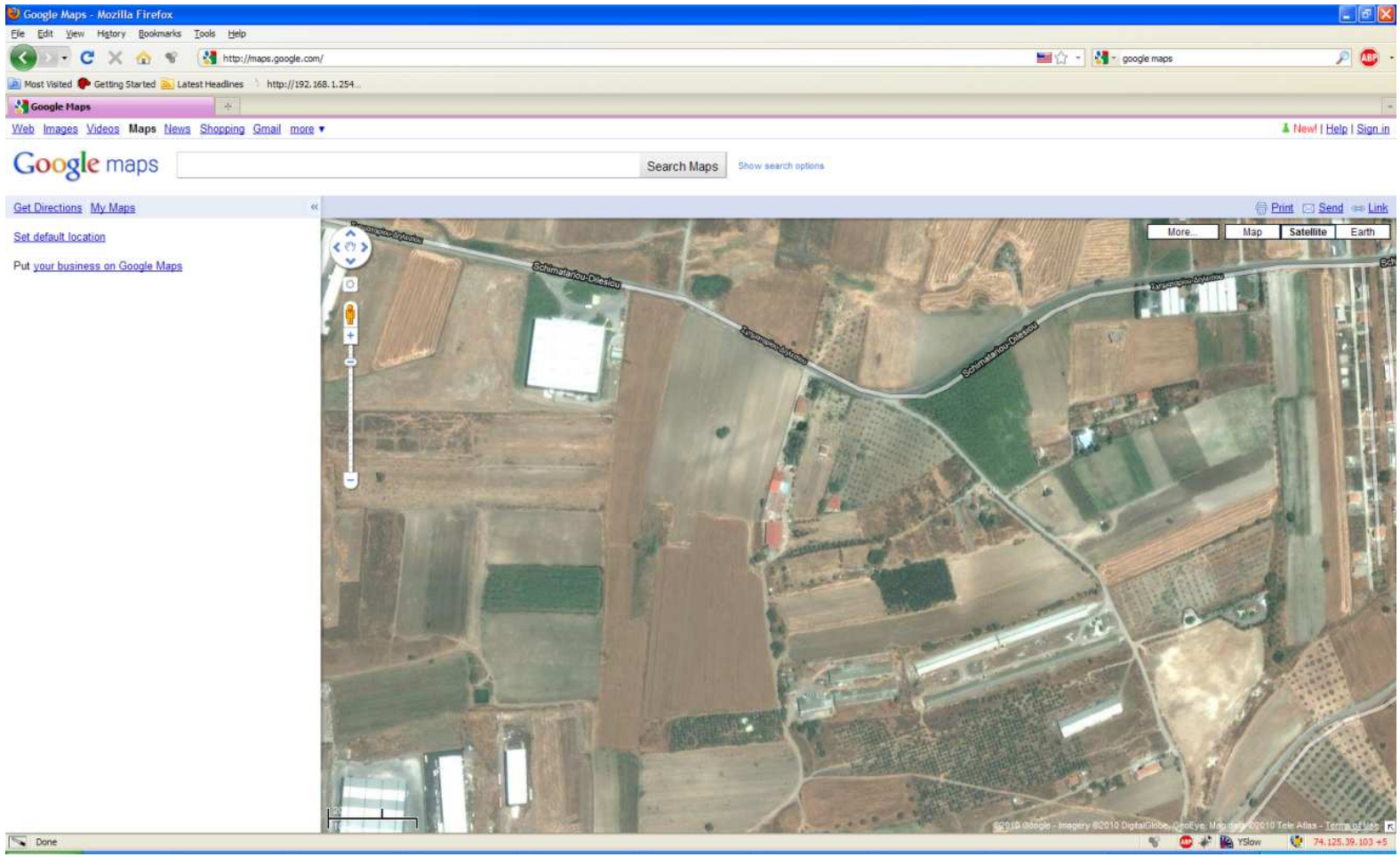
5.2. Αρχιτεκτονική

5.2.1. Διαδικασία υλοποίησης του shapefile

Η βάση λειτουργίας κάθε λογισμικού πλοήγησης είναι το αρχείο που περιέχει τη χωρική πληροφορία. Στην περίπτωση του λογισμικού που αναπτύχθηκε η μορφή αυτή είναι το shapefile της ESRI. Τα βήματα που ακολουθήθηκαν για την κατασκευή ώστε να ταιριάζει στους σκοπούς της παρούσας έρευνας αναφέρονται παρακάτω.

Καταρχήν επιλέχθηκε η αγροτική περιοχή που μας ενδιέφερε (Εικόνα 6) και αποθηκεύθηκε η δορυφορική εικόνα της περιοχής από το Google Earth. Στη συνέχεια η μορφή της εικόνας μετατράπηκε σε μορφή tiff και φορτώθηκε στο ArcMap (Εικόνα 7). Παράλληλα δημιουργήθηκε από το λογισμικό ArcCatalog το shapefile για επεξεργασία γραμμών με σύστημα αναφοράς

WSG 84. Και αυτό το αρχείο αποθηκεύτηκε στο project της εφαρμογής μαζί με την εικόνα της περιοχής που μας ενδιέφερε.



Εικόνα 6. Περιοχή ενδιαφέροντος στο google earth

Στην επόμενη φάση έγινε η γεωαναφορά της εικόνας της περιοχής που μας ενδιέφερε

Εικόνα 7. Project Arc Map

Μετά το τέλος της φάσης της γεωαναφοράς προστέθηκαν στο αρχείο ιδιοτήτων του shapefile (dbf) τα παρακάτω πεδία στις ήδη υπάρχουσες ιδιότητες (Εικόνα 8).

1. Πλάτος
2. Είδος δρόμου (άσφαλτος, χαλίκι, χωματόδρομος)
3. Κατάσταση οδοστρώματος
4. Εμφάνιση κλίσεων
5. Σημείο Εκκίνησης
6. Σημείο Τερματισμού
7. Μήκος

Attributes of Dilesi2

Distance	Width	Surface	SurfaceCon	Slope	StreetBel
11,36	0	0	0	0	0
11,6	0	0	0	0	0
11,6	0	0	0	0	0
11,6	0	0	0	0	0
11,6	0	0	0	0	0
11,6	0	0	0	0	0
11,6	0	0	0	0	0
11,6	0	0	0	0	0
11,6	0	0	0	0	0
11,6	0	0	0	0	0
11,6	0	0	0	0	0
11,6	0	0	0	0	0
11,6	0	0	0	0	0
11,6	0	0	0	0	0
11,6	0	0	0	0	0
11,6	0	0	0	0	0
11,6	0	0	0	0	0
11,6	0	0	0	0	0
11,6	0	0	0	0	0
11,6	0	0	0	0	0
11,6	0	0	0	0	0
11,6	0	0	0	0	0

Εικόνα 8. Ιδιότητες shapefile

Για την διευκόλυνση της περαιτέρω επεξεργασίας της πληροφορίας προστέθηκε και ένα πεδίο το οποίο υποδηλώνει ότι η συγκεκριμένη γραμμή ανήκει σε ένα ευρύτερο σύνολο γραμμών που συνδέονται που μοιράζονται τα ίδια χαρακτηριστικά (Πλάτος, Είδος δρόμου, Κατάσταση οδοστρώματος, εμφάνιση κλίσεων).

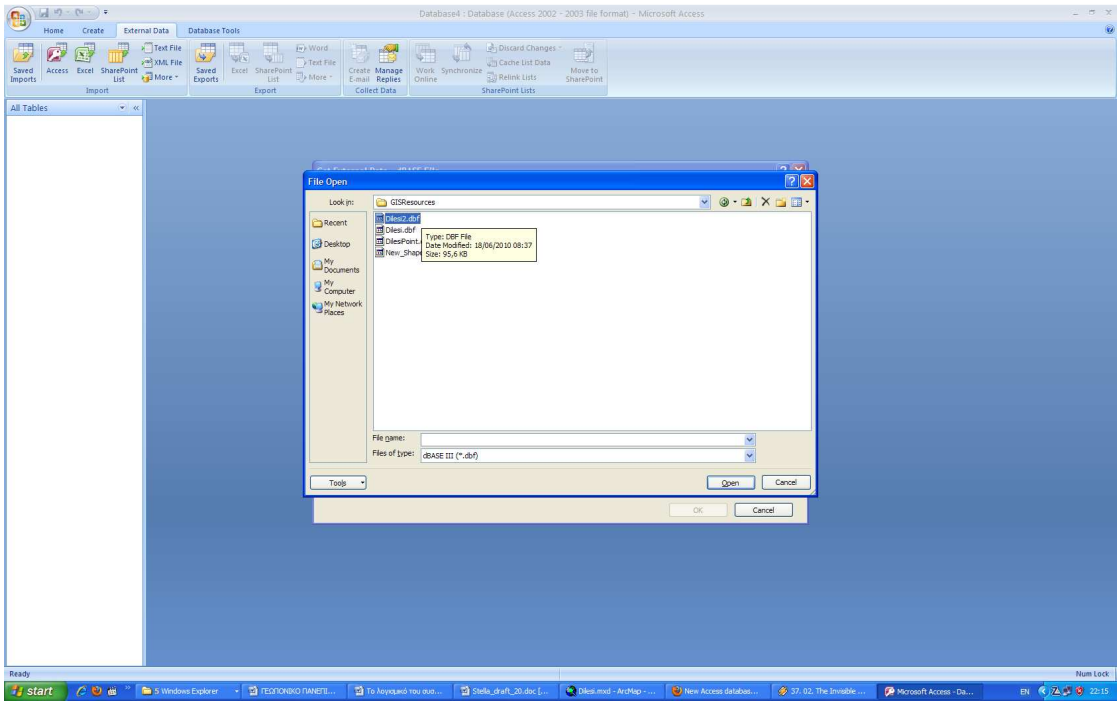
Στην συνέχεια η διαδρομές που εμφανίζονταν στην περιοχή ενδιαφέροντος ψηφιοποιήθηκαν και για κάθε ευρύτερο σύνολο συνδεόμενων γραμμών βαθμολογήθηκε το κάθε χαρακτηριστικό τους όπως φαίνεται στον παρακάτω πίνακα (Πίνακας 2).

Βαθμολογία	Πλάτος	Είδος δρόμου	Κατάσταση οδοστρώματος	Εμφάνιση κλίσεων
1	Όσο μία μηχανή	Χωματόδρομος	Πολύ κακή	Μεγάλη κλίση
2	Όσο ένα αυτοκίνητο	Χαλικόδρομος	Κακή	Μέτρια κλίση
3	Όσο δύο αυτοκίνητα ή 1 μεγάλο όχημα	Άσφαλτος	Μέτρια	Καθόλου κλίση
4	Όσο ένα μεγάλο όχημα και 1 αυτοκίνητο		Καλή	
5	Όσο δυο μεγάλα οχήματα		Πολύ καλή	

Πίνακας 2. Πίνακας Βαθμολογίας

5.2.2. Διαδικασία υλοποίησης Σχεσιακής Βάσης

Για την πιο γρήγορη πρόσβαση στα δεδομένα άλλα και την δυνατότητα χρήσης της γλώσσας SQL οι ιδιότητες των γραμμών μεταφέρθηκαν σε πινάκα στην Access (Εικόνα 9).



Εικόνα 9. Εισαγωγή αρχείου dbf

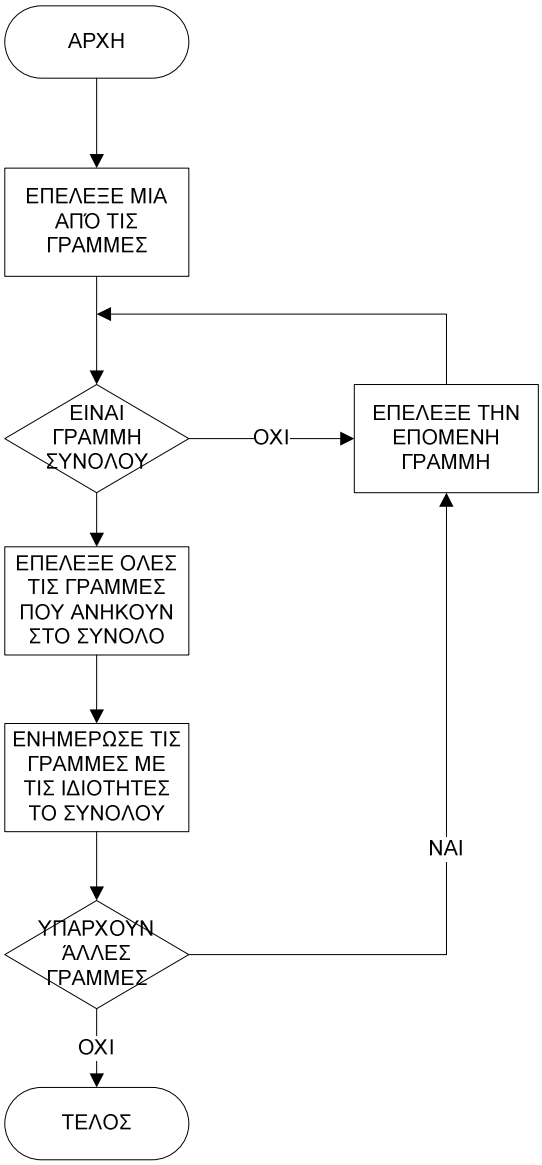
Στην συνέχεια, επειδή η πληροφορία που έχουμε αφορά τις γραμμές μόνο προς την μία κατεύθυνση (έχουμε την πληροφορία για τη γραμμή από το σημείο Α μέχρι το σημείο Β αλλά χρειαζόμαστε και την ίδια πληροφορία από το σημείο Β μέχρι το Α) χρησιμοποιούμε την

γλώσσα SQL ώστε να ενώσουμε των πίνακα ιδιοτήτων με τον εαυτό του έχοντας όμως αλλάξει τη δεύτερη φόρα τη στήλη που συμβολίζει το σημείο A με τη στήλη με το σημείο B (Εικόνα 10).

```
SELECT Roads.Id, Roads.Distance, Roads.Width, Roads.Surface, Roads.SurfaceCon, Roads.Slope,
Roads.StreetBel, Roads.XFrom, Roads.XTo
FROM Roads;
UNION
SELECT Roads.Id, Roads.Distance, Roads.Width, Roads.Surface, Roads.SurfaceCon, Roads.Slope,
Roads.StreetBel, Roads.XTo,Roads.XFrom
FROM Roads;
```

Εικόνα 10. Union Query

Με αυτό τον τρόπο σχηματίζουμε το πίνακα που περιέχει και τις δύο κατευθύνσεις.
Αφού τελειώσουμε την παραπάνω διεργασία πρέπει να ενημερώσουμε τα επιμέρους κομμάτια των γραμμών με τις ιδιότητες των γραμμών στο σύνολο στις οποίες ανήκουν. Η διαδικασία αυτή γίνεται χρησιμοποιώντας τη γλώσσα VBA (Εικόνα 12) και υλοποιώντας των αλγόριθμο (Εικόνα 11). που φαίνεται παρακάτω



Εικόνα 11. Αλγόριθμος ενημέρωσης ιδιοτήτων

```
Dim db As Database
Dim rs As DAO.Recordset
Dim SQL As String
Dim UpdateSQL As String
'Open connection to current Access database
Set db = CurrentDb()

'Create SQL statement to retrieve value from RoadsDerived table
SQL = "SELECT Width, Surface, SurfaceCon, Slope,id " & _
   " FROM RoadsDerived WHERE (((RoadsDerived.Id)=[streetBel]))"

Set rs = db.OpenRecordset(SQL)

rs.MoveFirst

Do While Not rs.EOF
    UpdateSQL = " Update RoadsDerived SET " & _
      " Width = " & rs!Width & "," & _
      " Surface = " & rs!Surface & "," & _
      " SurfaceCon = " & rs!SurfaceCon & "," & _
      " Slope = " & rs!Slope & " " & _
      " where [streetBel]=" & rs!id & " and RoadsDerived.Id<>[streetBel]"

    DoCmd.RunSQL UpdateSQL
   rs.MoveNext
Loop

 DoCmd.SetWarnings True
```

Εικόνα 12. Ενημέρωσης ιδιοτήτων VBA

Αφού δημιουργήσουμε πλέον τον ολοκληρωμένο και συμπληρωμένο πίνακα με τα στοιχειά των γραμμών, πρέπει να ενσωματώσουμε στη δομή της βάσης τους πίνακες αναφοράς (Πίνακας 3, 4, 5, 6).

StreetSlope	
ID	**Desc**
0	
1	Μεγάλη κλίση
2	Μέτρια κλίση
3	Μικρή κλίση

Πίνακας 3. Εμφάνιση κλίσεων

StreetSurface	
ID	**Desc**
0	
1	Χωματόδρομος
2	Δρόμος με χαλίκι
3	Άσφαλτος

Πίνακας 4. Είδος δρόμου

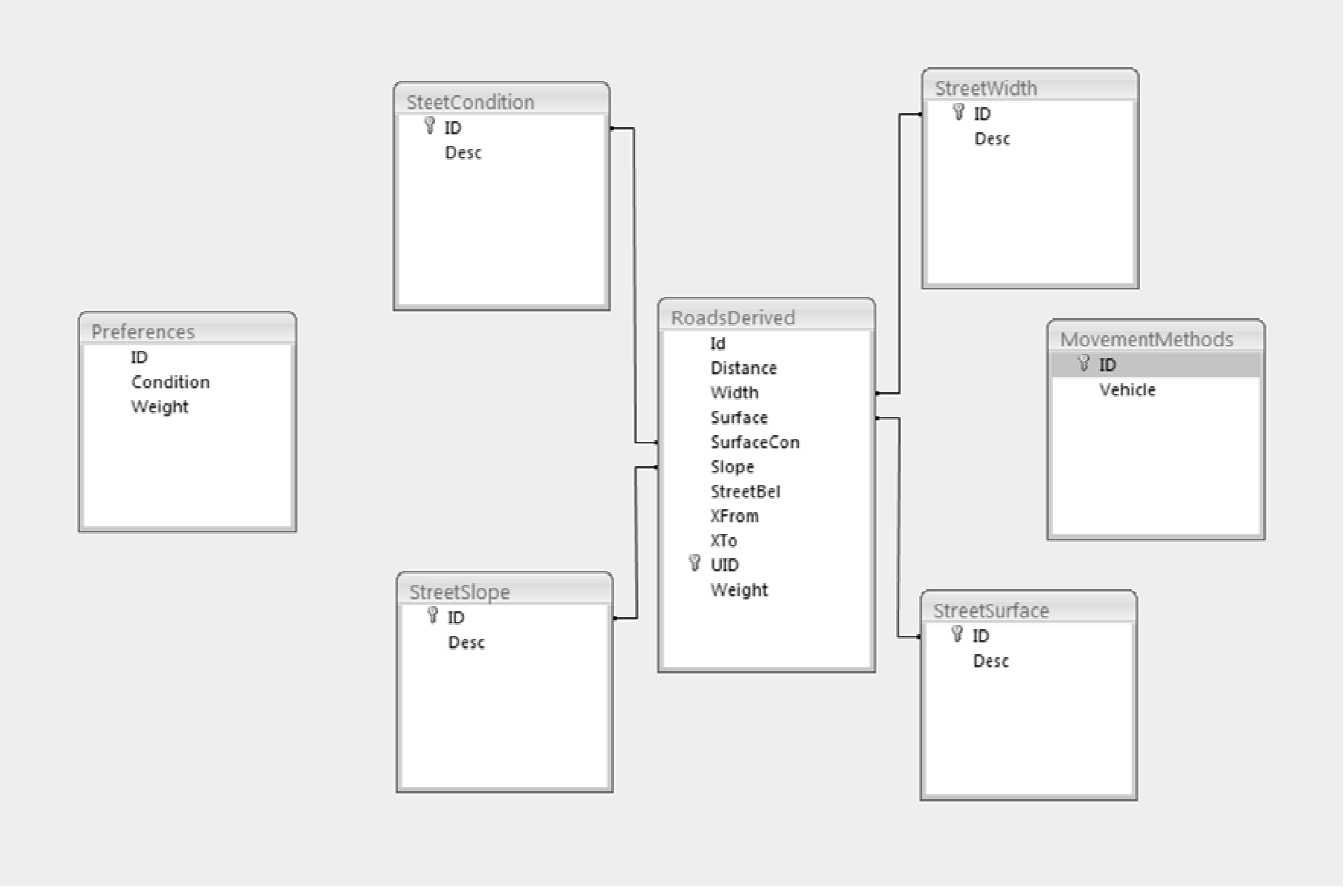
StreetWidth	
ID	**Desc**
0	
1	Όσο μία μηχανή
2	Όσο ένα αυτοκίνητο
3	Όσο δύο αυτοκίνητα ή ένα μεγάλο όχημα
4	Όσο ένα μεγάλο όχημα και ένα αυτοκίνητο
5	Όσο δύο μεγάλα οχήματα

Πίνακας 5. Πλάτος δρόμου

MovementMethods	
ID	**Vehicle**
1	Με τα πόδια
2	Με μοτοσυκλέτα
3	Με επιβατικό αμάξι
4	Με αγροτικό ή 4x4
5	Φορτηγό
6	Αγροτικό Μηχάνημα

Πίνακας 6. Τρόποι μετακίνησης

Μετά και από την δημιουργία των πινάκων αναφοράς έχουμε την τελική μορφή της βάσης που θα χρησιμοποιηθεί στο λογισμικό (Εικόνα 13).



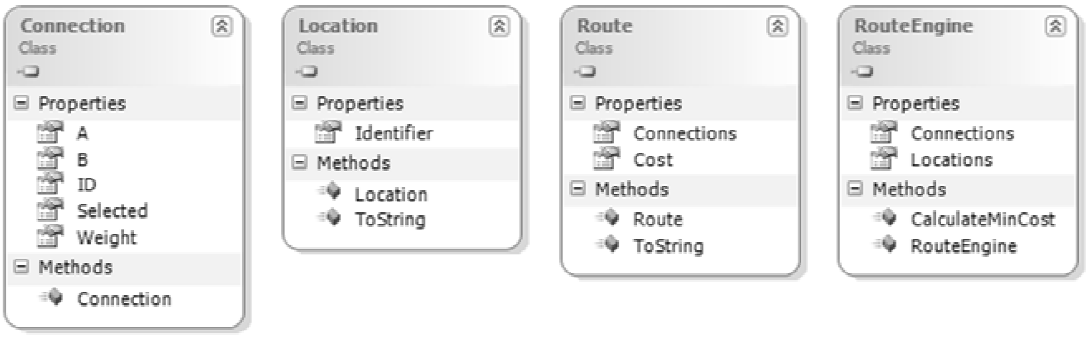
Εικόνα 13. Σχεσιακό διάγραμμα βάσης

5.2.3. Αλγόριθμος εύρεσης διαδρομής

Ο αλγόριθμος εύρεσης συντομότερης διαδρομής στηρίζεται στον αλγόριθμο του Dijkstra. Με την προτεινόμενη τροποποίηση, στον αλγόριθμο Dijkstra το βάρος της διαδρομής ανάμεσα σε δυο σημεία είναι το ίδιο είτε η διαδρομή ξεκινά από το σημείο Α μέχρι το σημείο Β, είτε ξεκινά από το σημείο Β μέχρι το σημείο Α. Στον αλγόριθμο που υλοποιείται στην δική μας περίπτωση πρέπει να πάρουμε υπόψη μας ότι μπορεί να υπάρχουν περιπτώσεις στις οποίες διαδρομή είναι προσβάσιμη μόνο προς τη μια κατεύθυνση (μονόδρομος). Για αυτό το λόγο σε προηγούμενη φάση κατασκευής χρησιμοποιήσαμε την γλώσσα SQL ώστε να ενώσουμε των πίνακα ιδιοτήτων με τον εαυτό του έχοντας όμως αλλάξει έχοντας όμως αλλάξει τη δεύτερη φόρα τη στήλη που συμβολίζει το σημείο Α με τη στήλη με το σημείο Β .

Ο αλγόριθμος εύρεσης συντομότερης διαδρομής έχει υλοποιηθεί με τη βοήθεια των παρακάτω τεσσάρων κλάσεων (Εικόνα 14).

1. **Connection** (Εικόνα 15): Η κλάση αυτή περιέχει πληροφορίες για την σύνδεση ανάμεσα σε δυο σημεία. Η κλάση έχει κατασκευαστεί με τέτοιο τρόπο ώστε να λαμβάνει υπόψη της τη κατεύθυνση της σύνδεσης. Αυτό έχει σαν αποτέλεσμα ότι η σύνδεση του σημείου Α με το σημείο Β να μην συνεπάγετε σύνδεση του σημείου Β με το σημείο Α.
2. **Location** (Εικόνα 16): Η κλάση αυτή αποθηκεύει ένα σημείο που ξεκινά η καταλήγει μία σύνδεση.
3. **RouteEngine** (Εικόνα 18): Η κλάση αυτή υπολογίζει τις διαδρομές από ένα σημείο προς όλα τα άλλα.
4. **Route** (Εικόνα 17): Η κλάση περιέχει πληροφορίες για τη διαδρομή ανάμεσα σε 2 σημεία.



Εικόνα 14. Κλάσεις υπολογισμού συντομότερης διαδρομής.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace RouteEngine
{
    public class Connection
    {
        Location _a, _b;
        int _weight;
        bool selected=false;
        string _id;

        public bool Selected
        {
            get
            {
                return selected;
            }
            set
            {
                selected = value;
            }
        }

        public Connection(Location a, Location b, int weight,string ID)
        {
            _a = a;
            _b = b;
            _weight = weight;
            _id = ID;
        }
        public Location B
        {
            get
            {
                return _b;
            }
```

```
            set
            {
                _b = value;
            }
        }

        public Location A
        {
            get
            {
                return _a;
            }
            set
            {
                _a = value;
            }
        }

        public int Weight
        {
            get
            {
                return _weight;
            }
            set
            {
                _weight = value;
            }
        }

        public string ID
        {
            get
            {
                return _id;
            }
            set
            {
                _id = value;
            }
        }
    }
}
  }
```

Εικόνα 15. Κλάση Connection

```
namespace RouteEngine
{
    public class Location
    {

        string _identifier;
```

```
        public Location()
        {

        }
        public string Identifier
        {
            get
            {
                return _identifier;
            }
            set
            {
                _identifier = value;
            }
        }
        public override string ToString()
        {
            return _identifier;
        }
    }
}
```

Εικόνα 16. Κλάση Location

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace RouteEngine
{
    public class Route
    {
        int _cost;
        List<Connection> _connections;
        string _identifier;

        public Route(string _identifier)
        {
            _cost = int.MaxValue;
            _connections = new List<Connection>();
            this._identifier = _identifier;
        }


        public List<Connection> Connections
        {
            get
            {
                return _connections;
            }
            set
            {
                _connections = value;
            }
        }
        public int Cost
        {
            get
            {
                return _cost;
            }
            set
            {
                _cost = value;
            }
        }

        public override string ToString()
        {
            return "Id:" + _identifier + " Cost:" + Cost;
        }
    }
}
```

Εικόνα 17. Κλάση Route

```
 using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace RouteEngine
{
    public class RouteEngine
    {
        List<Connection> _connections;
        List<Location> _locations;

        public List<Location> Locations
        {
            get
            {
                return _locations;
            }
            set
            {
                _locations = value;
            }
        }
        public List<Connection> Connections
        {
            get
            {
                return _connections;
            }
            set
            {
                _connections = value;
            }
        }

        public RouteEngine()
        {
            _connections = new List<Connection>();
            _locations = new List<Location>();
        }


        public Dictionary<Location, Route> CalculateMinCost(Location
_startLocation)
        {
            Dictionary<Location, Route> _shortestPaths = new
Dictionary<Location, Route>();
            List<Location> _handledLocations = new List<Location>();

            foreach (Location location in _locations)
            {
                _shortestPaths.Add(location, new
Route(location.Identifier));
            }

            _shortestPaths[_startLocation].Cost = 0;
```

```
            while (_handledLocations.Count != _locations.Count)
            {
                List<Location> _shortestLocations =
(List<Location>)(from s in _shortestPaths

orderby s.Value.Cost

select s.Key).ToList();


                Location _locationToProcess = null;

                foreach (Location _location in _shortestLocations)
                {
                    if (!_handledLocations.Contains(_location))
                    {
                        if (_shortestPaths[_location].Cost ==
int.MaxValue)
                            return _shortestPaths;
                        _locationToProcess = _location;
                        break;
                    }
                }

                var _selectedConnections = from c in _connections
                                           where c.A ==
_locationToProcess
                                           select c;

                foreach (Connection conn in _selectedConnections)
                {
                    if (_shortestPaths[conn.B].Cost > conn.Weight +
_shortestPaths[conn.A].Cost)
                    {
                        _shortestPaths[conn.B].Connections =
_shortestPaths[conn.A].Connections.ToList();
                        _shortestPaths[conn.B].Connections.Add(conn);
                        _shortestPaths[conn.B].Cost = conn.Weight +
_shortestPaths[conn.A].Cost;
                    }
                }
                _handledLocations.Add(_locationToProcess);
            }


            return _shortestPaths;
        }
    }
}
```

Εικόνα 18. Κλάση RouteEngine

Όπως φαίνεται από τις παραπάνω κλάσεις ο αλγόριθμός εύρεσης συντομότερης διαδρομής δέχεται σαν είσοδο μόνο μια παράμετρο, οπότε πρέπει για να μπορέσουμε να καταλήξουμε σε μια διαδρομή να δημιουργήσουμε μια συνάρτηση που να δέχεται τις προτιμήσεις του χρήστη και να παράγει το αθροιστικό βάρος για όλη τη διαδρομή. Το τελικό βάρος παράγεται με βάση τα παρακάτω στοιχεία καθώς και την αλληλεπίδραση των οχημάτων με τις ιδιότητες των δρόμων (Πίνακας 7)

Τρόποι μετακίνησης:

1. Άνθρωπος (με τα πόδια)
2. Μηχανή (μοτοσικλέτα)
3. Αγροτικό (φορτηγάκι)
4. Ιδιωτικής χρήσης επιβατικό όχημα
5. Ιδιωτικής χρήσης 4x4
6. Φορτηγό
7. Αγροτικό μηχάνημα (τρακτέρ)

Ιδιότητες Δρόμων

1. Πλάτος
2. Είδος δρόμου (άσφαλτος, χαλίκι, χωματόδρομος)
3. Κατάσταση οδοστρώματος
4. Εμφάνιση κλίσεων

Διανυόμενη απόσταση

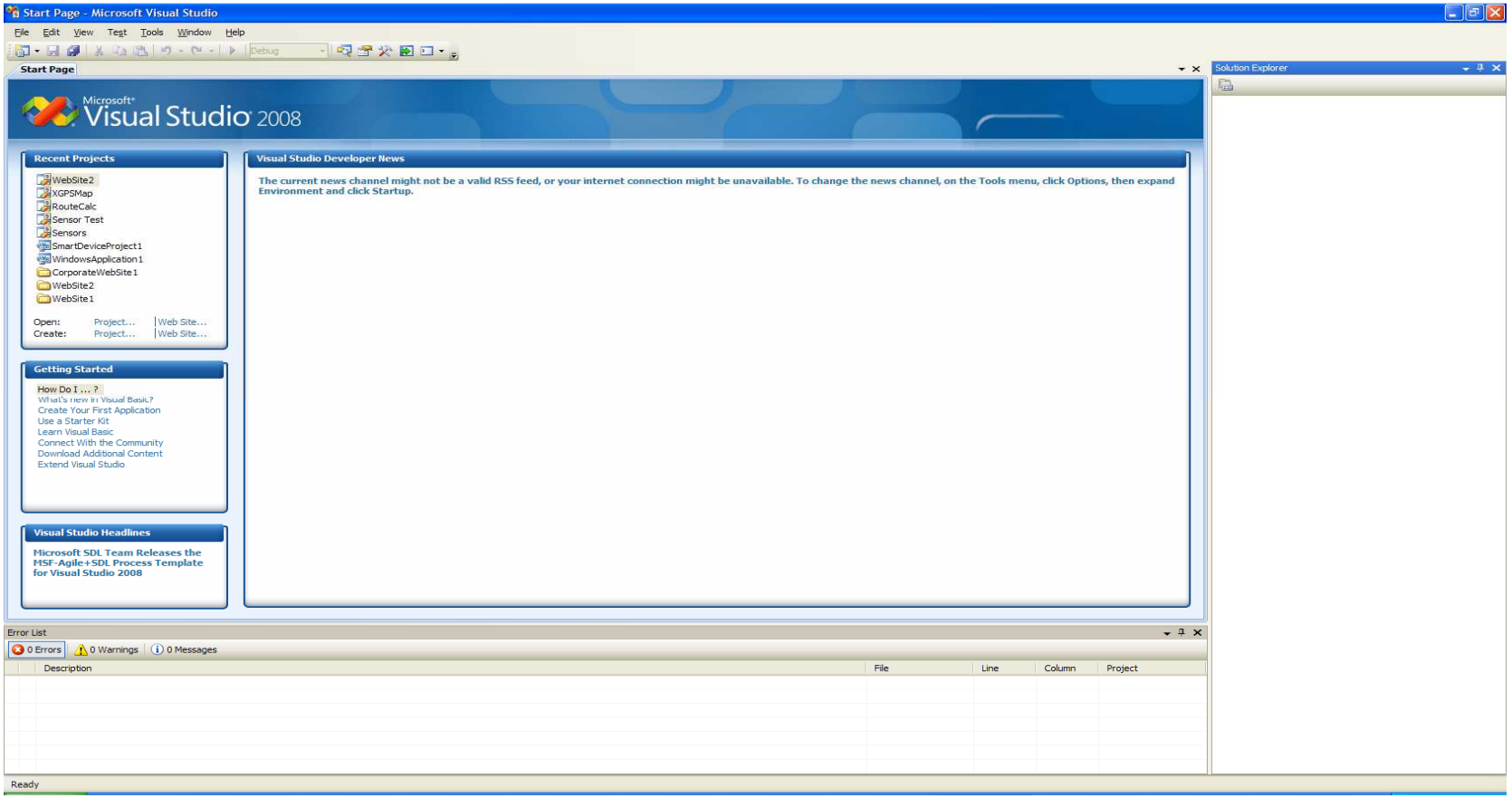
	Πλάτος	Είδος δρόμου	Κατάσταση οδοστρώματος	Εμφάνιση κλίσεων
Άνθρωπος (με τα πόδια)	>0	>0	>1	>2
Μηχανή (μοτοσικλέτα)	>0	>0	>1	>0
Αγροτικό (φορτηγάκι)	>1	>0	>0	>0

Ιδιωτικής χρήσης επιβατικό όχημα	>1	>1	>2	>0
Ιδιωτικής χρήσης 4x4	>1	>0	>0	>0
Φορτηγό	>2	>1	>2	>1
Αγροτικό μηχάνημα (τρακτέρ)	>2	>0	>0	>1

Πίνακας 7. Αλληλεπίδραση οχημάτων σε σχέση με ιδιότητες δρόμων

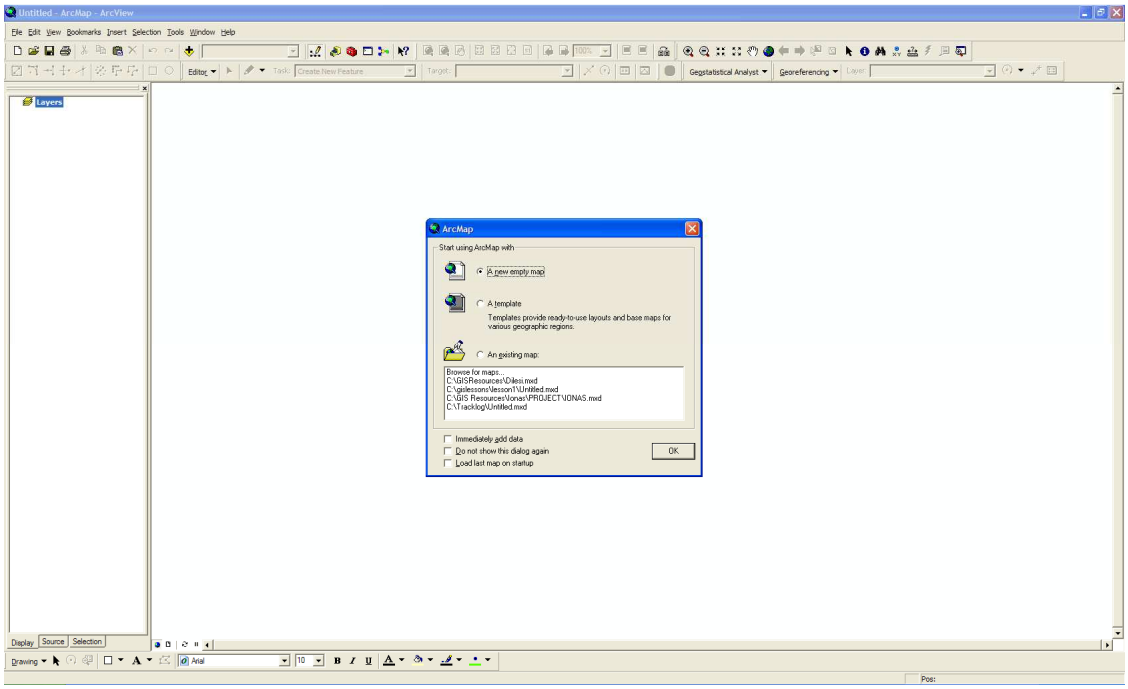
5.3. Σχεδιασμός Λογισμικού πλοήγησης

Το λογισμικό του συστήματος πλοήγησης κατασκευάστηκε με τη χρήση της γλώσσας προγραμματισμού C# (Microsoft, The C# Language, 2010). Η C# είναι μια γλώσσα προγραμματισμού της Microsoft για τη .NET πλατφόρμα. Συνδυάζει μερικά από τα καλύτερα χαρακτηριστικά γνωρίσματα των σύγχρονων γλωσσών προγραμματισμού όπως η Java, C++ και Visual Basic. Η C# είναι μια αντικειμενοστραφής γλώσσα που υποστηρίζει κληρονομικότητα. Το ολοκληρωμένο περιβάλλον ανάπτυξης που χρησιμοποιήθηκε για την ανάπτυξη της εφαρμογής είναι το Visual Studio 2008 (Microsoft, Visual Studio Professional 2008, 2010) (Εικόνα 19).



Εικόνα 19. Visual Studio 2008

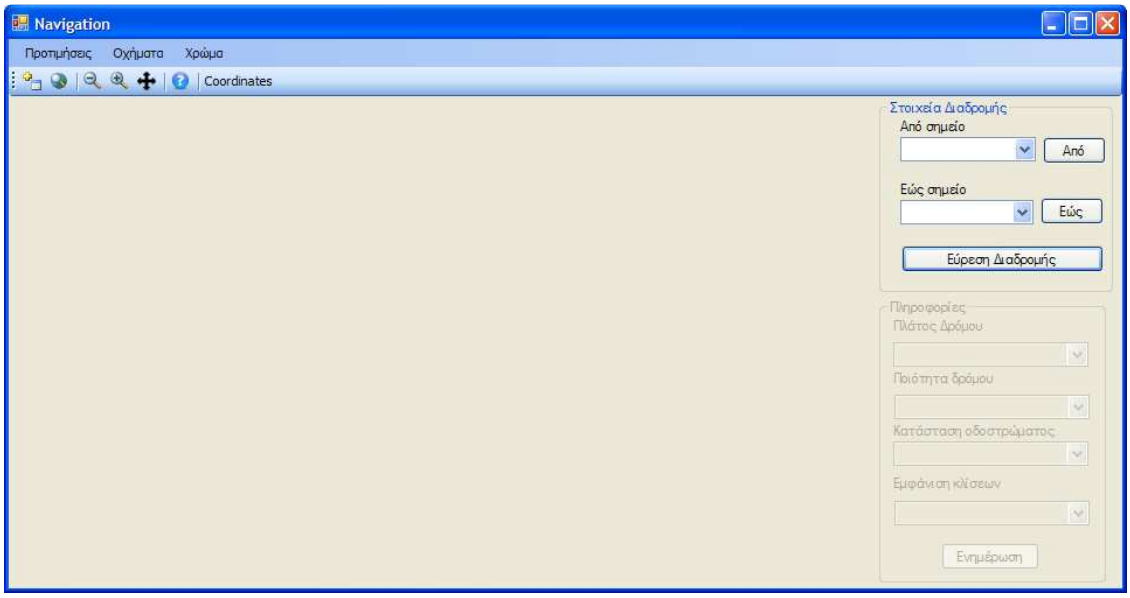
Το αρχείο χωρικών δεδομένων δημιουργήθηκε και επεξεργάστηκε από το λογισμικό ArcGis (ESRI, 2010). Το ArcGIS (Εικόνα 20) είναι μια ολοκληρωμένη συλλογή από προϊόντα λογισμικού που αφορούν τα ΓΣΠ . Παρέχει μια πλατφόρμα για διαδικασίες χωρικής ανάλυσης, διαχείρισης δεδομένων και απεικόνισης.



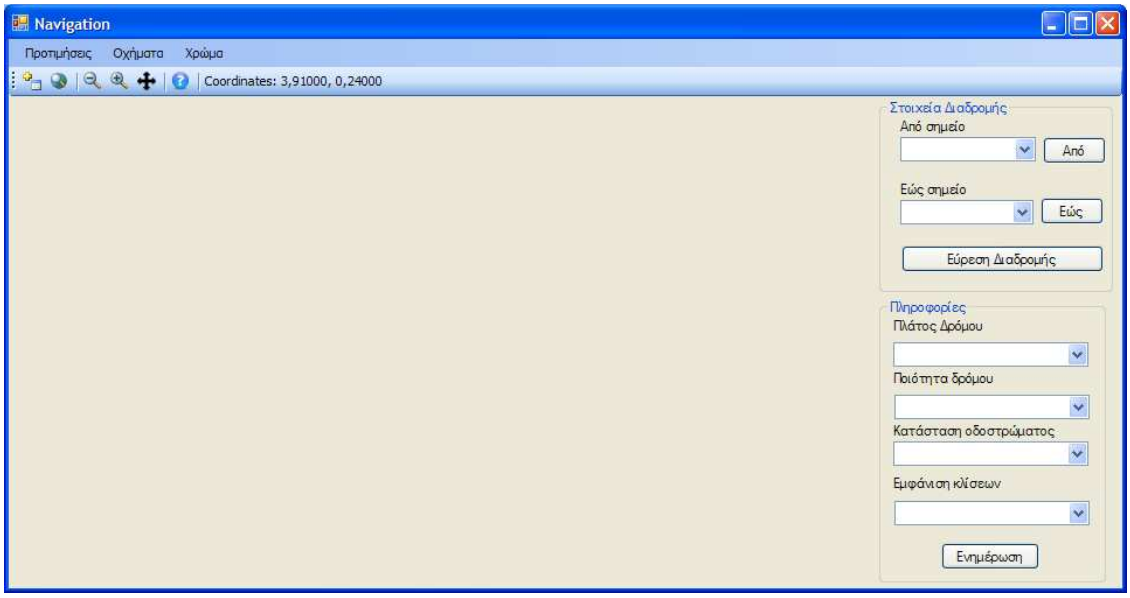
Εικόνα 20. ArcMap

Η σχεσιακή βάση που χρησιμοποιήθηκε για την αποθήκευση των δεδομένων και των παραμέτρων είναι της μορφής mdb και επεξεργάστηκε με την Access από το ολοκληρωμένο πακέτο εφαρμογών Office XP (Microsoft, Microsoft Office XP, 2010). Η Access επιλέχθηκε καθαρά για εκπαιδευτικούς σκοπούς καθώς είναι ευρέως εγκατεστημένο πακέτο λογισμικού ακόμα και σε απλούς προσωπικούς υπολογιστές, ώστε με αυτό το τρόπο ο κάθε χρήστης να μπορεί με σχετική ευκολία να δει τη δομή της βάσης. Αξίζει να σημειωθεί ότι άλλες σχεσιακές βάσεις δεδομένων θα πρόσφεραν πολύ μεγαλύτερη ταχύτητα επεξεργασίας δεδομένων στο σύστημα.

Το λογισμικό πλοήγησης υλοποιήθηκε σε δυο εκδόσεις. Μία έκδοση για τον απλό χρήστη (Εικόνα 21), στην οποία ο χρήστης δεν μπορεί αλλάξει τις ιδιότητες των δρόμων και μία έκδοση για το διαχειριστή (Εικόνα 22) του συστήματος στην όποια οι ιδιότητες των δρόμων μπορούν πάρουν διαφορετική τιμή εάν αυτό κριθεί απαραίτητο από το διαχειριστή.



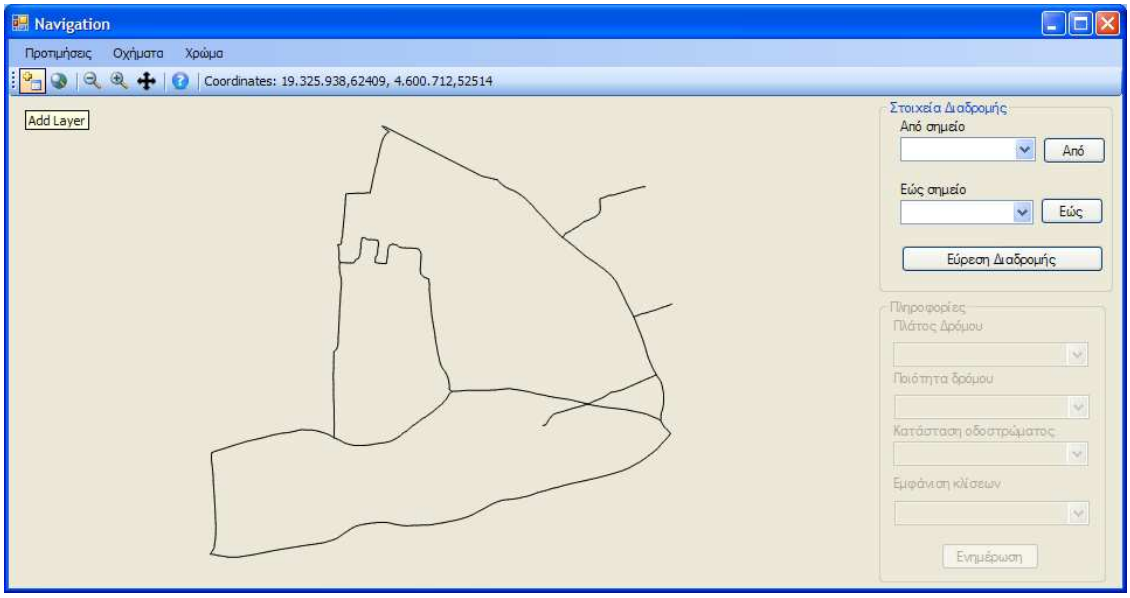
Εικόνα 21. Λογισμικό πλοήγησης απλού χρήστη



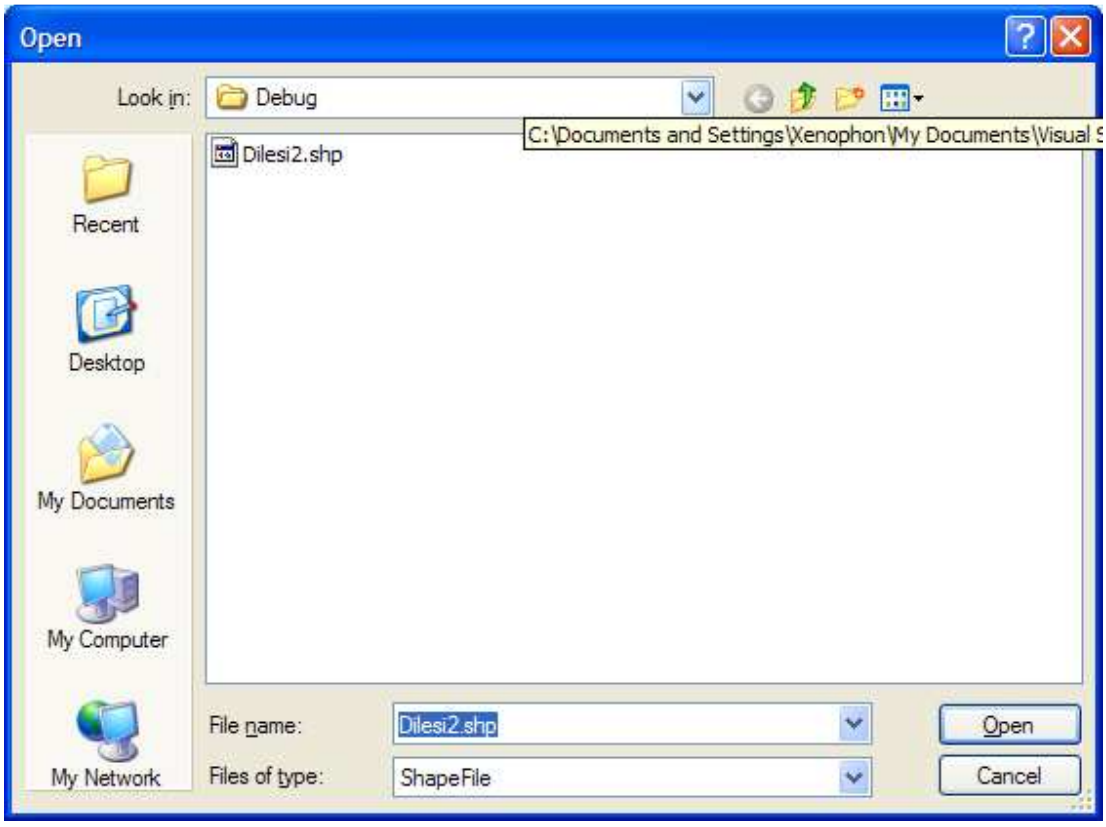
Εικόνα 22. Λογισμικό πλοήγησης διαχειριστή του συστήματος

5.3.1. Μενού διαχείρισης χάρτη

Για να μπορέσει ο χρήστης να χρησιμοποιήσει την εφαρμογή θα πρέπει να φορτώσει ένα κατάλληλα διαμορφωμένο χάρτη (Εικόνα 23,24). Εάν υπάρχει τέτοιος χάρτης διαθέσιμος μπορεί να φορτωθεί στην εφαρμογή πατώντας το κουμπί  .



Εικόνα 23. Λογισμικό πλοήγησης Φόρτωση Χάρτη

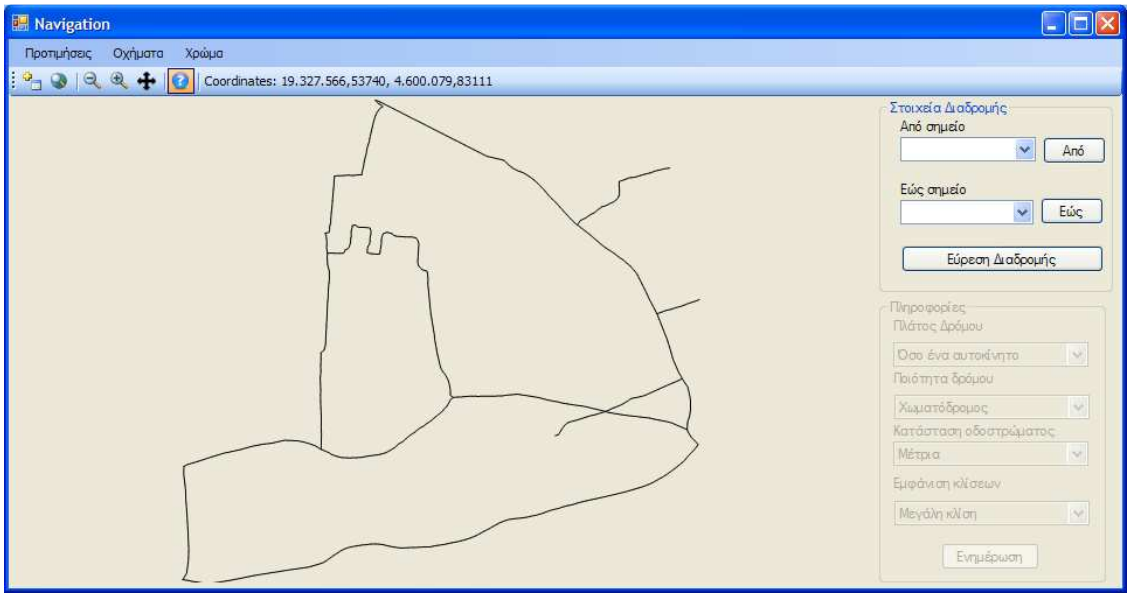


Εικόνα 24. Λογισμικό πλοήγησης Φόρτωση Χάρτη

Αφού φορτώσουμε το χάρτη μπορούμε να πλοηγηθούμε σε αυτόν με την χρήση των κουμπιών  . Τα κουμπιά αυτά μας επιτρέπουν να κάνουμε τις παρακάτω λειτουργίες πού αναφέρονται με τη σειρά που έχουν τα κουμπιά από δεξιά προς τα αριστερά είναι:

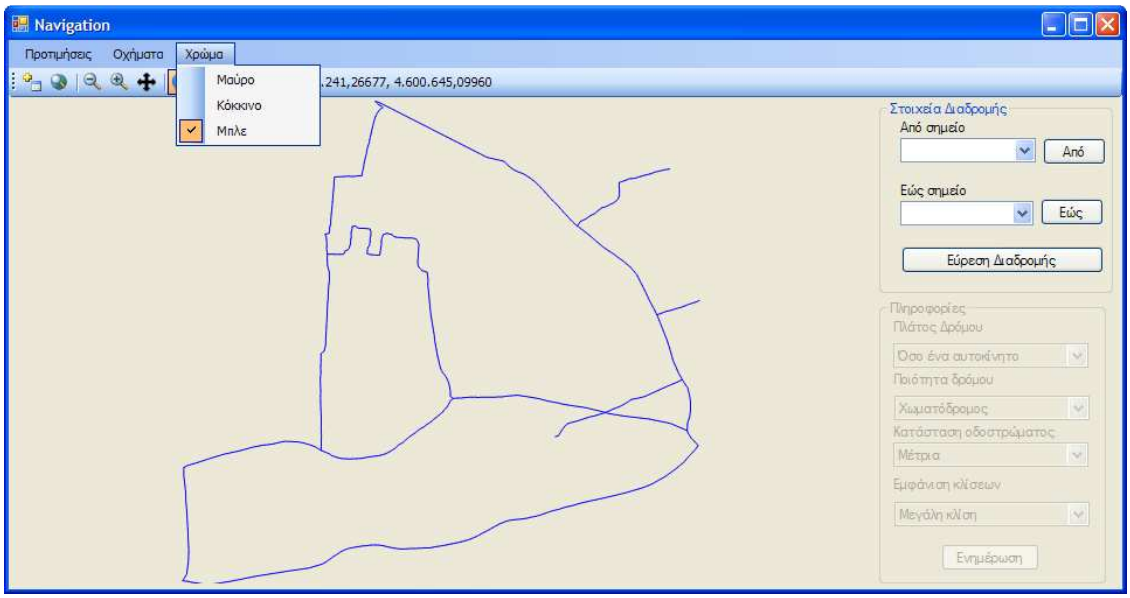
1. Εμφάνιση όλου του χάρτη
2. Μεγέθυνση
3. Σμίκρυνση
4. Μετακίνηση χάρτη
5. Ερώτημα.

Η λειτουργία Ερώτημα (Εικόνα 25) μας επιτρέπει να αναζητήσουμε τις ιδιότητες του δρόμου στο χάρτη και φαίνεται παρακάτω.



Εικόνα 25. Λογισμικό πλοήγησης λειτουργία Query

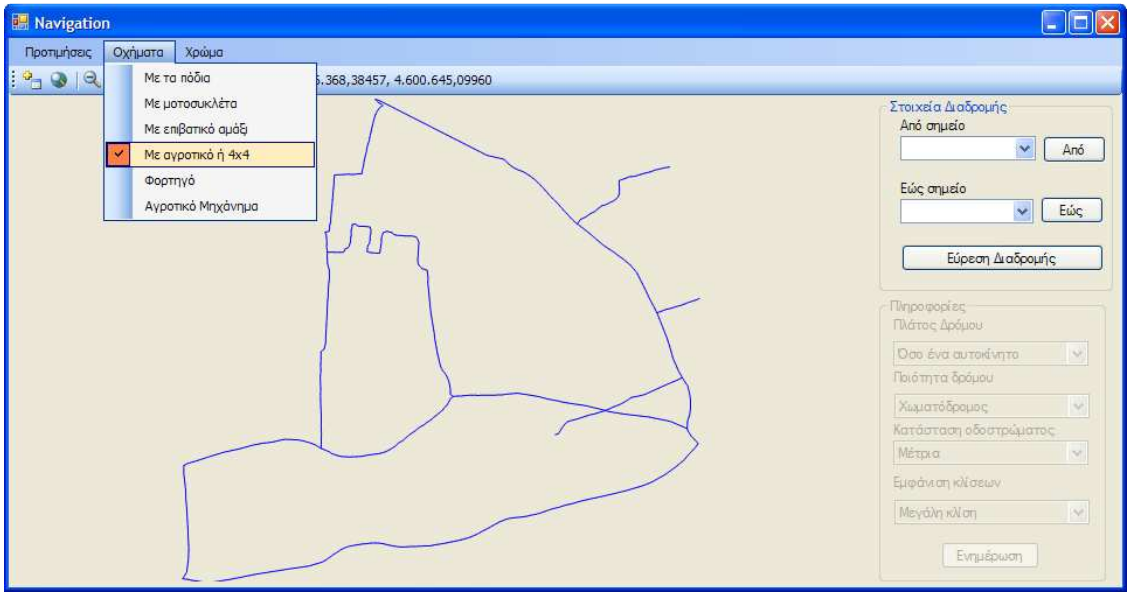
Επίσης μέσω του λογισμικού πλοήγησης μπορούμε να επιλέξουμε το χρώμα (Εικόνα 26) με το οποίο θα εμφανίζεται ο χάρτης εάν το αρχικό χρώμα δεν μας ικανοποίει.



Εικόνα 26. Λογισμικό πλοήγησης Επιλογή χρώματος

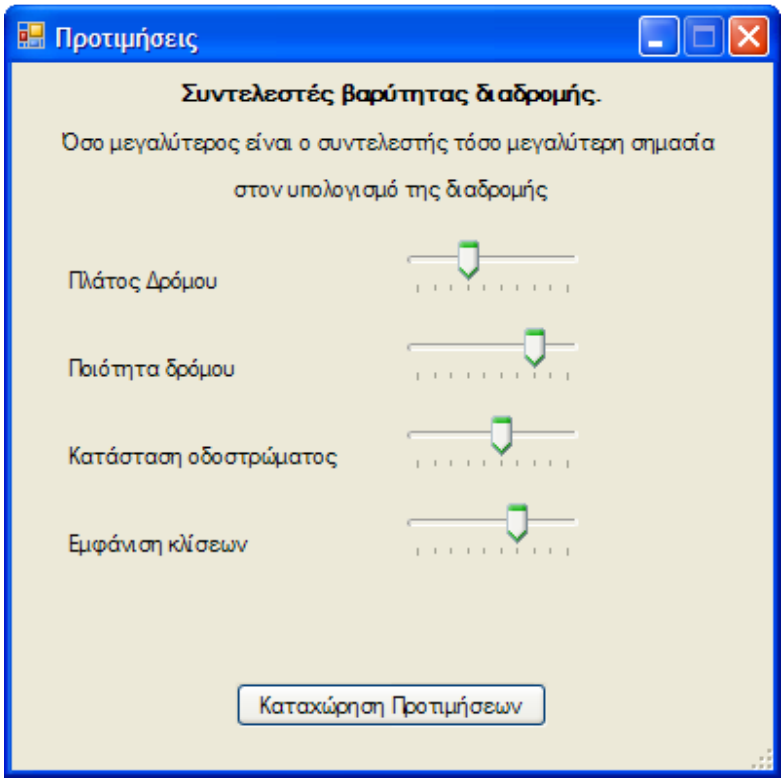
5.3.2. Μενού πλοήγησης

Αφού φορτωθεί ο χάρτης που ενδιαφέρει τον χρήση και ρυθμιστούν οι οπτικοί παράμετροι της εφαρμογής (επίπεδο Zoom, χρώμα κτλ), ο χρήστης πρέπει να καθορίσει τις προτιμήσεις του για την εύρεση της διαδρομής. Καταρχήν πρέπει να δηλώσει για ποιο τύπο οχήματος τον ενδιαφέρει να βρει την διαδρομή. Η λειτουργία αυτή υλοποιείται μέσω της επιλογής 'Οχήματα' (Εικόνα 27).



Εικόνα 27. Λογισμικό πλοήγησης Επιλογή οχήματος

Αφού οριστεί το όχημα για το όποιο ενδιαφέρει το χρήστη η διαδρομή, υπάρχει η δυνατότητα να ορίσει ο χρήστης τους συντελεστές βαρύτητας για τη διαδρομή που θα υπολογιστεί. Η δυνατότητα αυτή δίνεται μέσα από της επιλογής 'Προτιμήσεις' (Εικόνα 28).

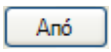


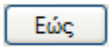
Εικόνα 28. Λογισμικό πλοήγησης Συντελεστές βαρύτητας

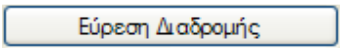
Όσο πιο δεξιά είναι τοποθετημένος ο δείκτης για κάθε κατηγορία βαρύτητας της διαδρομής τόσο μεγαλύτερη σημασία θα έχει η συγκεκριμένη κατηγορία για τον υπολογισμό ποιότητας της διαδρομής. Δηλαδή εάν ο συντελεστής βαρύτητας 'Πλάτος Δρόμου' μετακινηθεί τέρμα δεξιά τότε η διαδρομή που θα μας προτείνει το λογισμικό πλοήγησης θα δώσει μεγάλο βάρος στο να υπάρχουν στην διαδρομή δρόμοι με μεγάλο πλάτος.

5.3.3. Εύρεσης διαδρομής

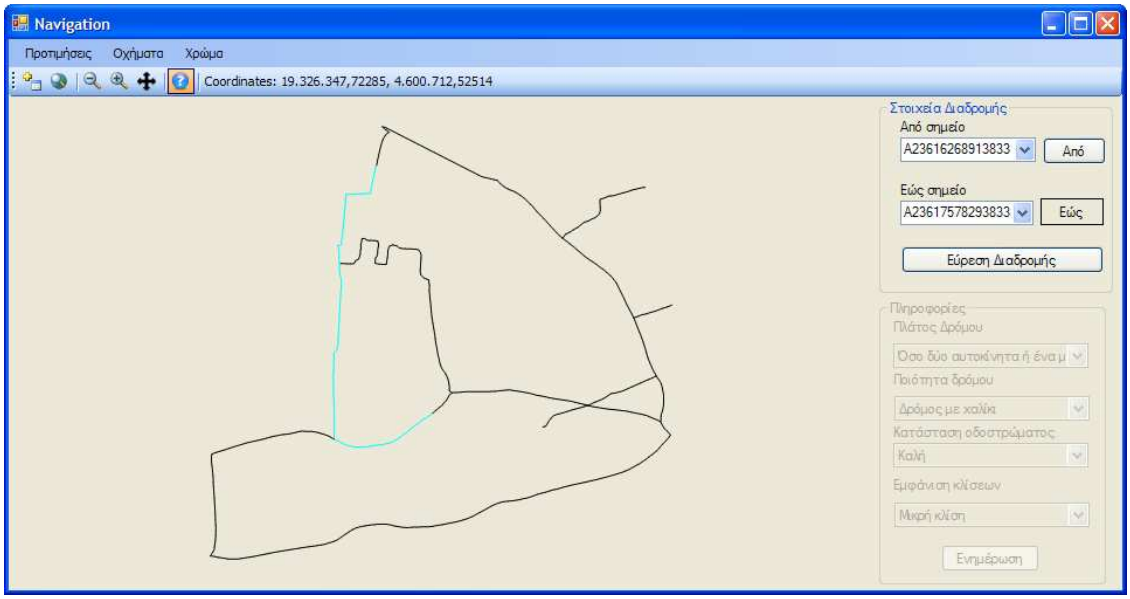
Αφού ο χρήστης έχει συμπληρώσει τις προτιμήσεις τόσο στο είδος οχήματος που τον ενδιαφέρει και όσο αφορά τους συντελεστές βαρύτητας της διαδρομής μπορεί να προχωρήσει στην εύρεση της βέλτιστης διαδρομής με βάση τις προτιμήσεις του. Η διαδικασία που πρέπει να ακολουθήσει είναι η εξής:

1. Επιλεγεί την αρχή τη διαδρομής του πατώντας το κουμπί 

2. Επιλεγεί το τέλος της διαδρομής πατώντας το κουμπί 

3. Πατάει το κουμπί 

Κάνοντας τις παραπάνω ενέργειες στην οθόνη της εφαρμογής εμφανίζεται η διαδρομή που πρέπει να ακολουθήσει ο χρήστης (Εικόνα 29).



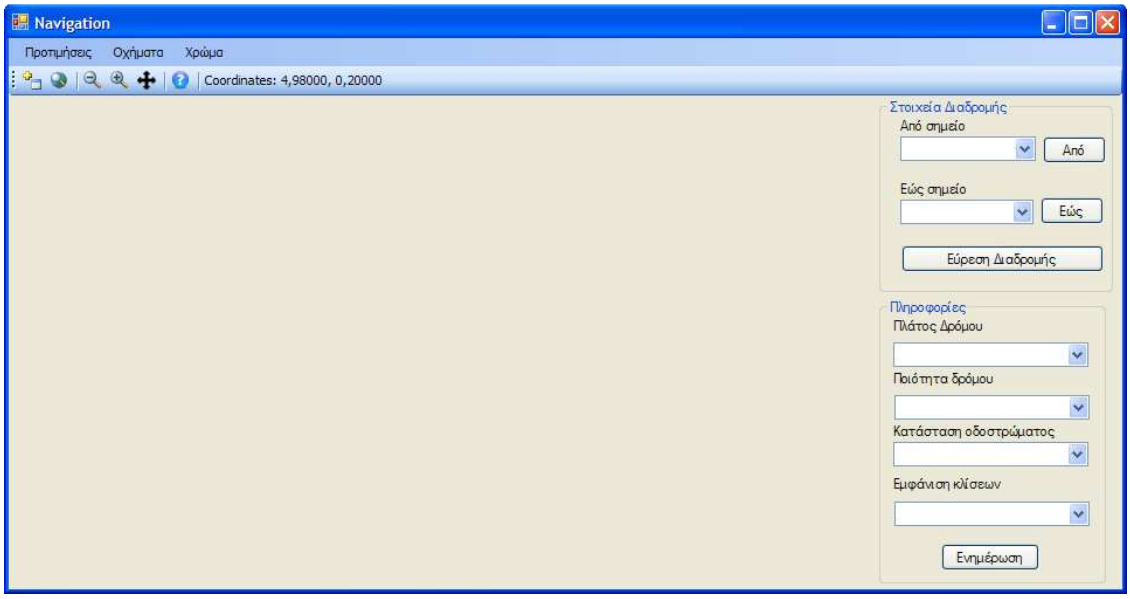
Εικόνα 29. Λογισμικό πλοήγησης Εύρεση διαδρομής

5.3.4. Ενημέρωση παραμέτρων δρόμων

Όπως σε όλα τα δυναμικά συστήματα έτσι και στους χάρτες πλοήγησης οι παράμετροι που καθορίζουν τους δρόμους δεν μένουν σταθεροί. Διάφοροι εξωγενείς παράγοντες μπορούν να προκαλέσουν αλλαγή στην κατάσταση που επικρατεί σε ένα δρόμο. Για παράδειγμα, μια εργασία ασφαλτόστρωσης θα βελτιώσει αισθητά την ποιότητα ενός δρόμου. Για το λόγο αυτό στο λογισμικό πλοήγησης υπάρχει η δυνατότητα της ενημέρωσης των παραμέτρων του δρόμου από το χρήστη, όπως νομίζει αυτός ότι έχουν διαμορφωθεί.

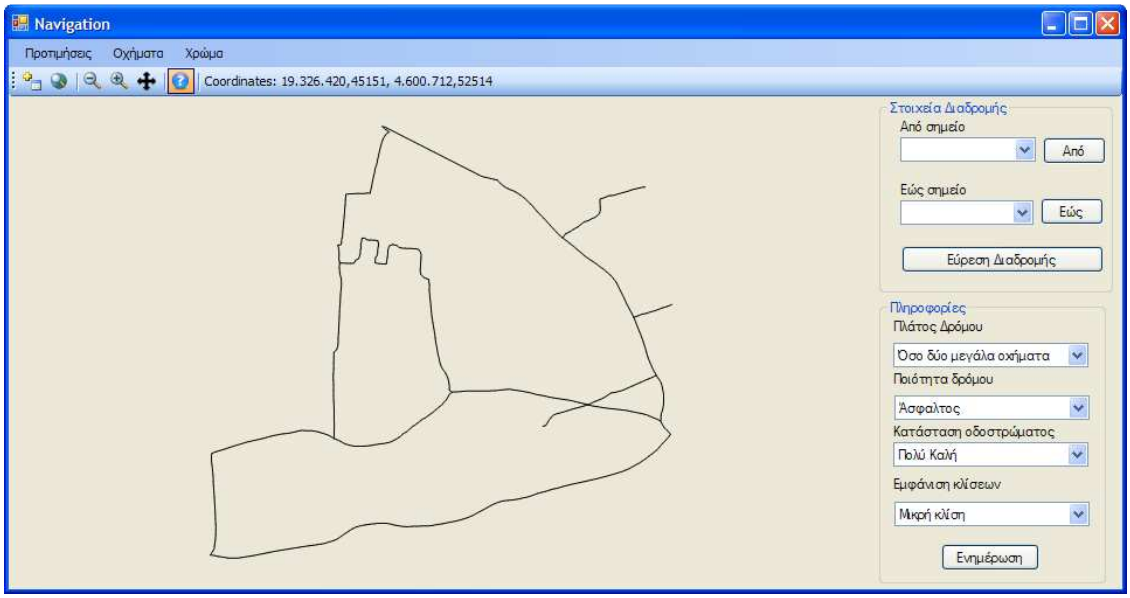
Για να μπορέσει ο χρήστης να αλλάξει τις τιμές στις παραμέτρους των δρόμων πρέπει να έχει εγκαταστήσει στον υπολογιστή του την έκδοση για το διαχειριστή του συστήματος

(Εικόνα 30). Η έκδοση αυτή διακρίνεται από την απλή έκδοση από το γεγονός ότι οι παράμετροι του δρόμου δεν είναι πλέον απενεργοποιημένες.



Εικόνα 30. Λογισμικό πλοήγησης Έκδοση διαχειριστή του συστήματος

Για να γίνει η αλλαγή στις ιδιότητες του δρόμου αρκεί ο χρήστης να επιλέξει το ευθύγραμμο τμήμα που τον ενδιαφέρει με τη χρήση του κουμπιού  και να ενημερώσει κατάλληλα τις τιμές (Εικόνα 31).



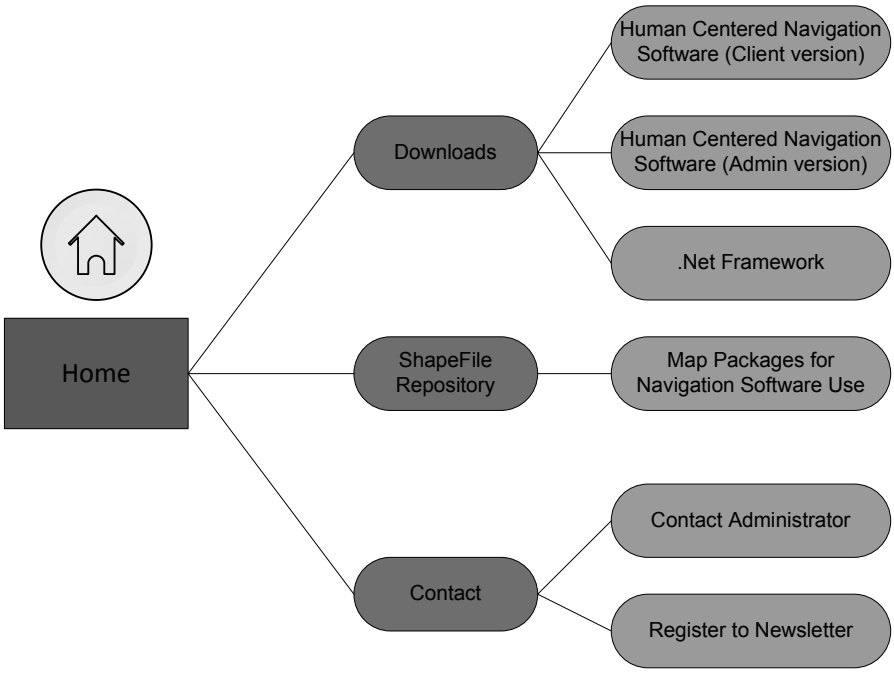
Εικόνα 31. Λογισμικό πλοήγησης Επιλεγμένο τμήμα με τις ιδιότητες του

5.4. Σχεδιασμός Ιστοτόπου

Ο ενδιαφερόμενος για την εφαρμογή πλοήγησης και για χάρτες που μπορούν να συνεργαστούν μαζί της μπορεί να τα κατεβάσει από το Διαδίκτυο. Για το λόγο αυτό δημιουργήθηκε κατάλληλος ιστοτόπος, ώστε ο τελικός χρήστης να έχει πρόσβαση στην τελευταία έκδοση του προγράμματος και στους πιο ενημερωμένους χάρτες. Για τη δημιουργία του ιστοτόπου χρησιμοποιήθηκε το Microsoft Visual Studio 2008. Οι σελίδες του ιστοτόπου αναπτύχθηκαν με την τεχνολογία ASP.NET (Microsoft, ASP.NET Wiki: ASP.NET, 2010), η βάση δεδομένων είναι σε μορφή SQL Server 2005 (Microsoft, SQL Server 2005 Technologies, 2010)και η πρόσβαση των σελίδων του ιστοτόπου γίνεται με τη βοήθεια του Microsoft IIS server (Microsoft, The Official Microsoft IIS Site, 2010). Η αποστολή και λήψη ερωτημάτων μεταξύ χρήστη και της βάσης δεδομένων γίνεται με τη χρήση γλώσσας SQL. Η γλώσσα προγραμματισμού που χρησιμοποιήθηκε είναι η Microsoft C#.

Ο ιστοτόπος η δομή του οποίου φαίνεται στην εικόνα 32 περιλαμβάνει το βασικό μενού πλοήγησης (κεντρική σελίδα) με τους παρακάτω συνδέσμους:

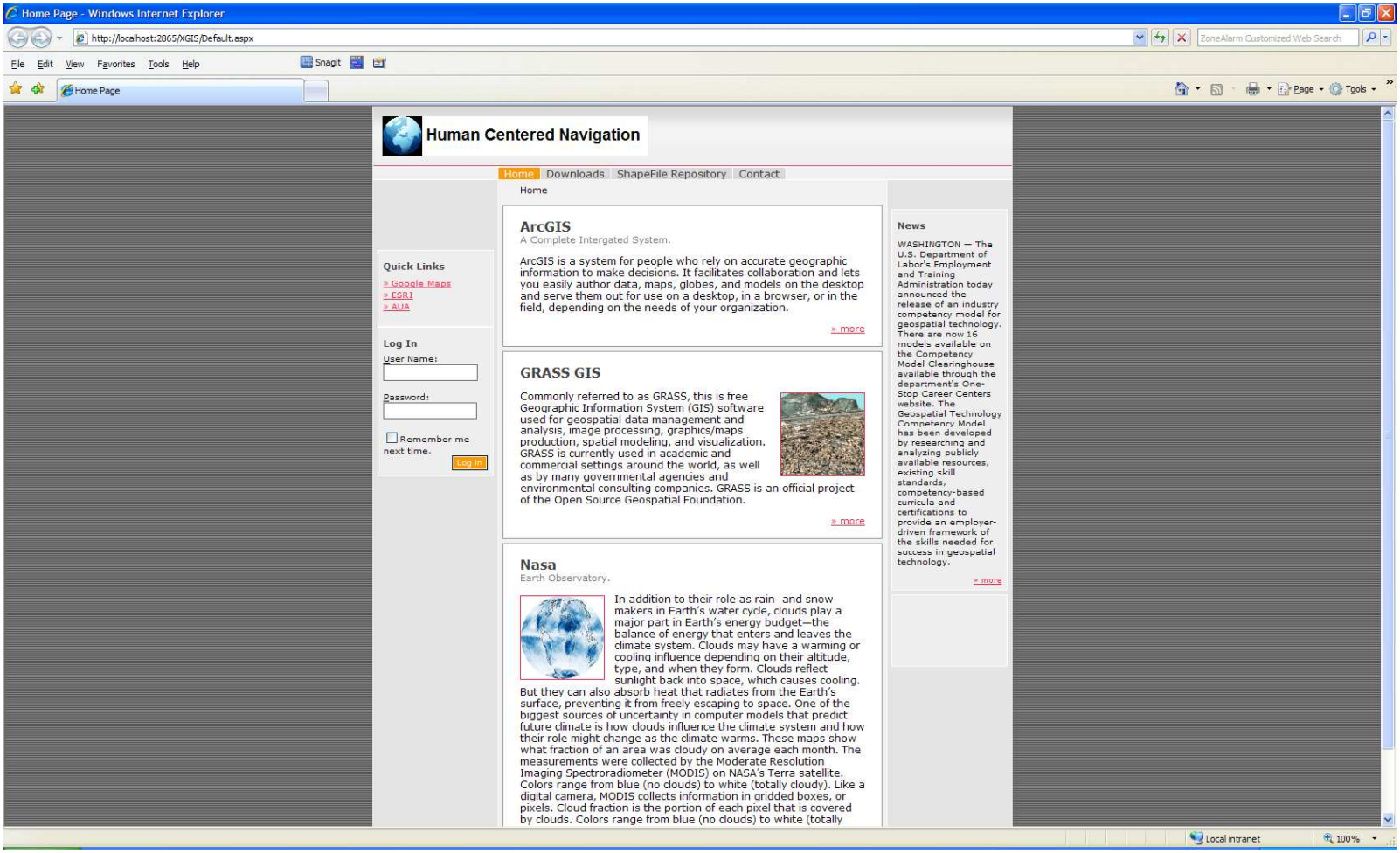
1. Αρχική σελίδα (Home).
2. Σελίδα για κατέβασμα του προγράμματος πλοήγησης και των πακέτων λογισμικών που είναι απαραίτητα για να τρέξει (Downloads).
3. Σελίδα για κατέβασμα πακέτων χαρτών (ShapeFile Repository).
4. Σελίδα για επικοινωνία του χρήστη με τον διαχειριστή του ιστοτόπου (Contact).



Εικόνα 32. Δομή Ιστοτόπου

5.4.1. Αρχική Σελίδα (Home)

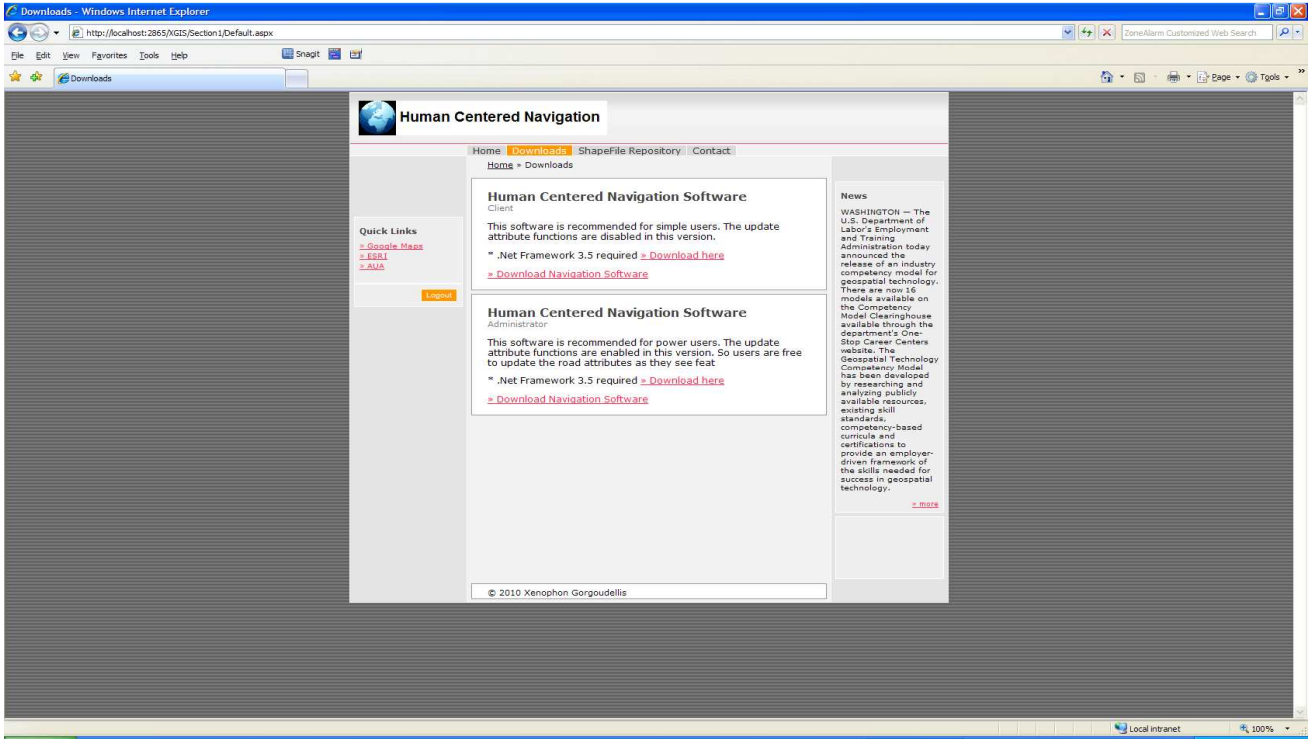
Στην αρχική ιστοσελίδα (Εικόνα 33) ο χρήστης μπορεί να δει γενικές πληροφορίες σχετικά με τα Γεωγραφικά Συστήματα Πληροφοριών, να διαβάσει τρέχοντα νέα που αφορούν το γενικότερο πεδίο της Γεωπληροφορικής καθώς και να συμπληρώσει τους κωδικούς του ώστε να αποκτήσει πρόσβαση στην υπόλοιπη δομή του ιστοτόπου. Ακόμα υπάρχει η δυνατότητα πρόσβασης σε άλλους σχετικούς ιστοτόπους.



Εικόνα 33. Αρχική ιστοσελίδα

5.4.2. Σελίδα Λογισμικού (Downloads)

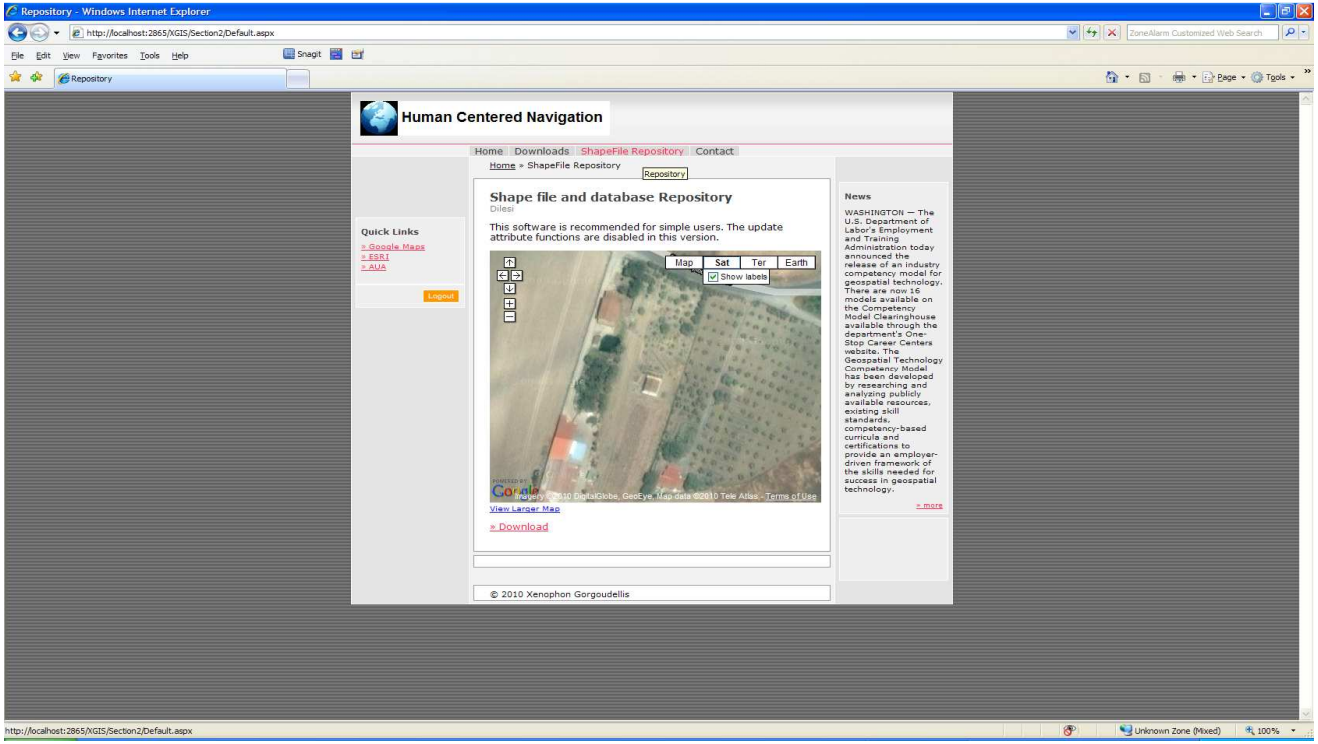
Στην ιστοσελίδα αυτή (Εικόνα 34) ο χρήστης μπορεί να κατεβάσει το λογισμικό πλοήγησης και στις δυο εκδόσεις του (την έκδοση για τον απλό χρήστη ή την έκδοση του διαχειριστή) καθώς και το σχετικό λογισμικό που απαιτείται για να τρέχει το λογισμικό πλοήγησης στον προσωπικό υπολογιστή του. Για να έχει πρόσβαση στην συγκεκριμένη σελίδα ο χρήστης πρέπει να έχει συμπληρώσει τους κωδικούς πρόσβασης που του έχουν δοθεί.



Εικόνα 34. Ιστοσελίδα downloads

5.4.3. Σελίδα για κατέβασμα πακέτων χαρτών (ShapeFile Repository)

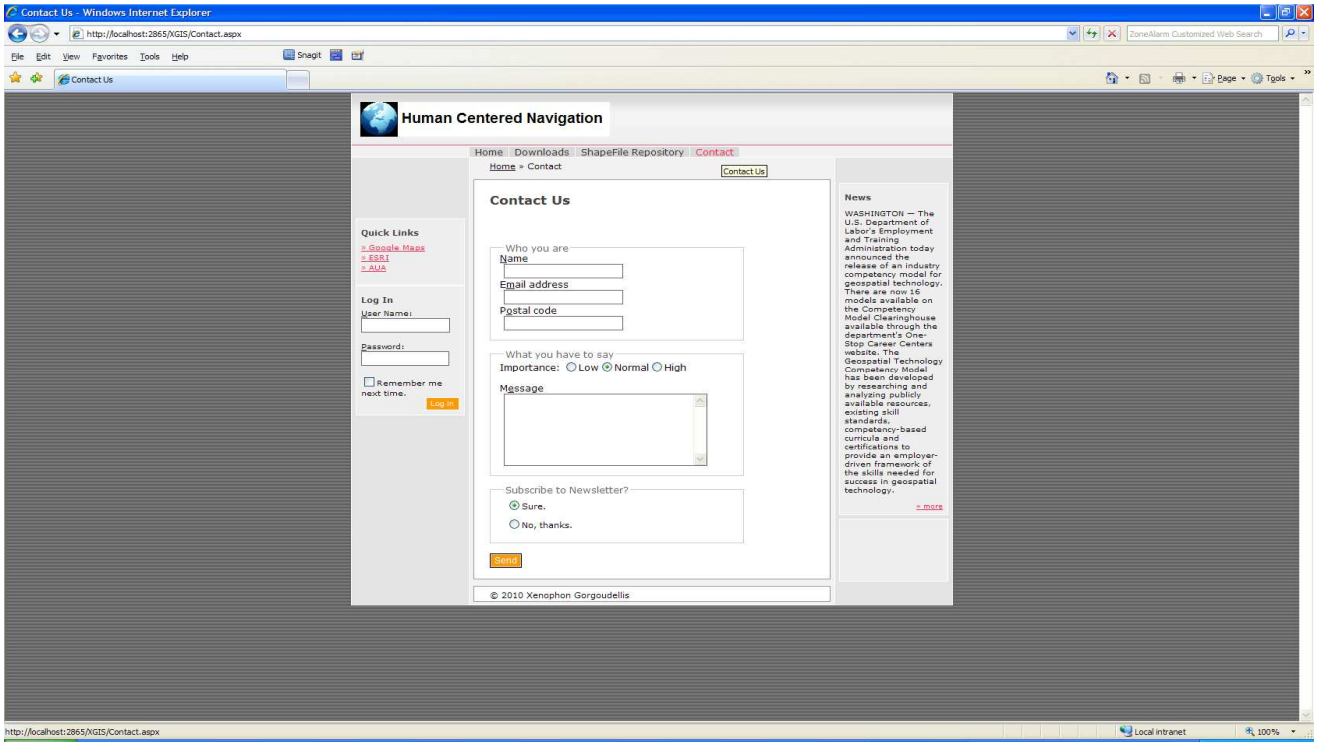
Στην ιστοσελίδα (Εικόνα 35) αυτή ο χρήστης μπορεί να κατεβάσει το σχετικό πακέτο χαρτών που τον ενδιαφέρει για να τρέξει στο λογισμικό πλοήγησης ή να κατεβάσει την πιο ανανεωμένη έκδοση ενός πακέτου χαρτών που ήδη υπάρχει εγκατεστημένο στο λογισμικό πλοήγησης. Για την διευκόλυνση του χρήστη στην επιλογή του κατάλληλου πακέτου χαρτών κάθε πακέτο στην ιστοσελίδα εμφανίσει την περιοχή που αντιπροσωπεύει μέσω Google maps. Για να έχει πρόσβαση στην συγκεκριμένη σελίδα ο χρήστης πρέπει να έχει συμπληρώσει τους κωδικούς πρόσβασης που του έχουν δοθεί.



Εικόνα 35. Ιστοσελίδα ShapeFile Repository

5.4.4. Σελίδα για επικοινωνία του χρήστη με τον διαχειριστή του ιστοτόπου (Contact).

Στην ιστοσελίδα (Εικόνα 36) αυτή ο χρήστης μπορεί επικοινωνήσει με το διαχειριστή του ιστοτόπου για τυχόν προβλήματα που έχει αλλά και για να κάνει αίτηση ώστε να του δοθούν κωδικοί πρόσβασης στον ιστοτόπο.



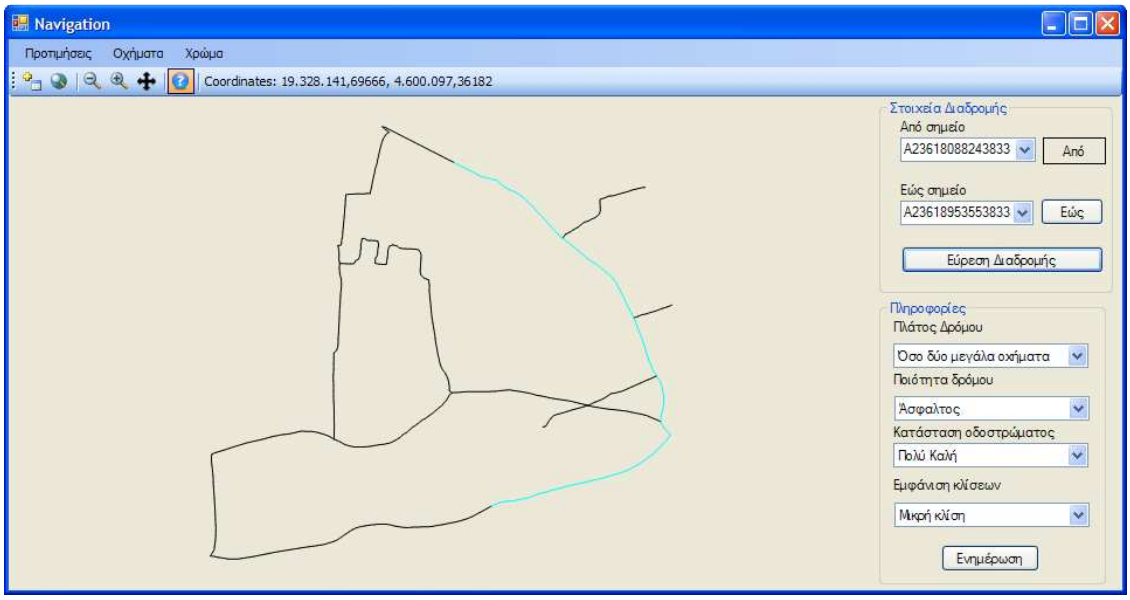
Εικόνα 36. Ιστοσελίδα Contact

6. ΑΠΟΤΕΛΕΣΜΑΤΑ ΕΦΑΡΜΟΓΗΣ

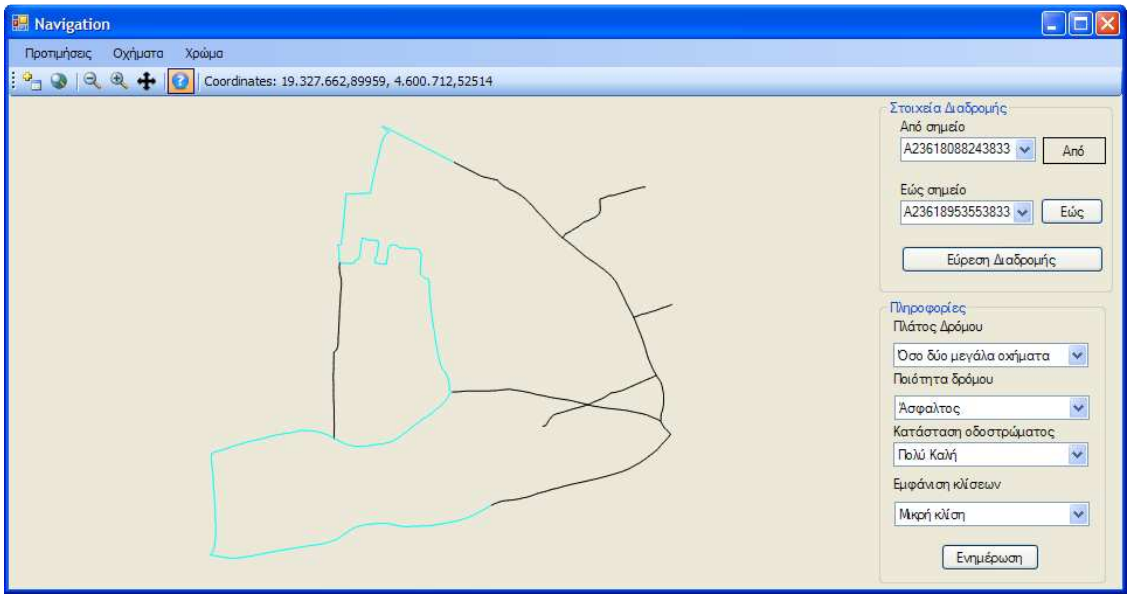
Η χρήση του λογισμικού πλοήγησης είναι ένα από τα πολλά εργαλεία που έχει ο χρήστης για λύσει το βασικό πρόβλημα της εύρεσης της βέλτιστης διαδρομής. Στη συγκεκριμένη περίπτωση στη διαδικασία εύρεσης της διαδρομής δόθηκε μεγάλο βάρος στην επίδραση παραμέτρων που παραδοσιακά δεν υπάρχουν στις εμπορικές εφαρμογές πλοήγησης. Οι παράμετροι που ενσωματώθηκαν στο λογισμικό πλοήγησης είναι:

1. Πλάτος δρόμου
2. Είδος δρόμου (άσφαλτος, χαλίκι, χωματόδρομος)
3. Κατάσταση οδοστρώματος
4. Εμφάνιση κλίσεων

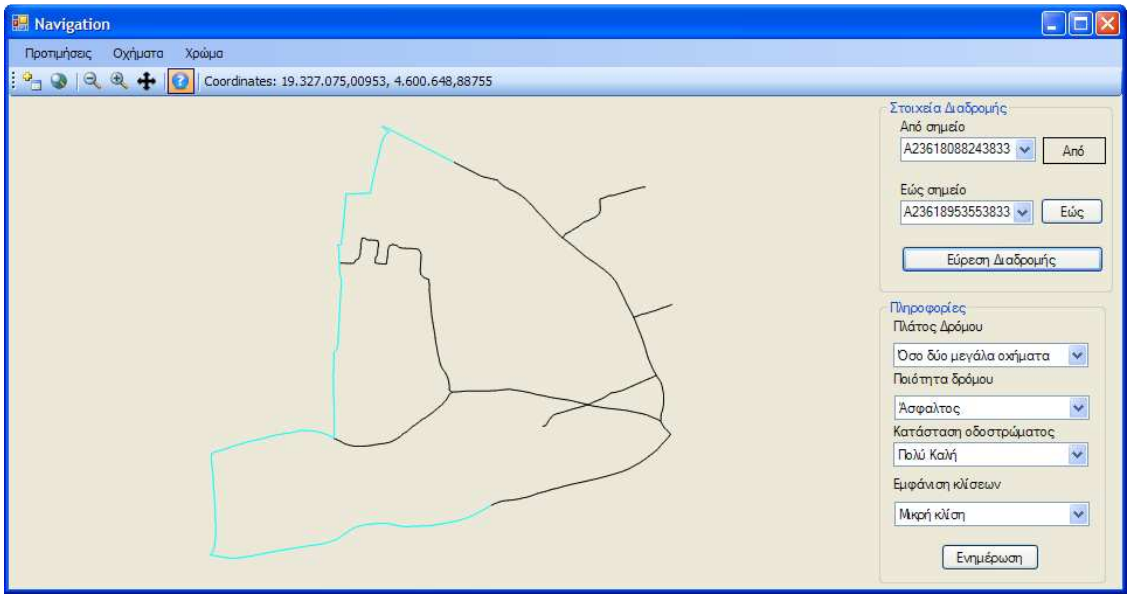
Πέρα από την συνηθισμένη παράμετρο της απόστασης. Από τη χρήση της εφαρμογής διαπιστώθηκε ότι οι διαδρομές που προτείνονται διαφέρουν αρκετά αναλόγως των προτιμήσεων του χρήστη (Εικόνα 37, 38, 39).



Εικόνα 37. Διαδρομή αγροτικό μηχάνημα

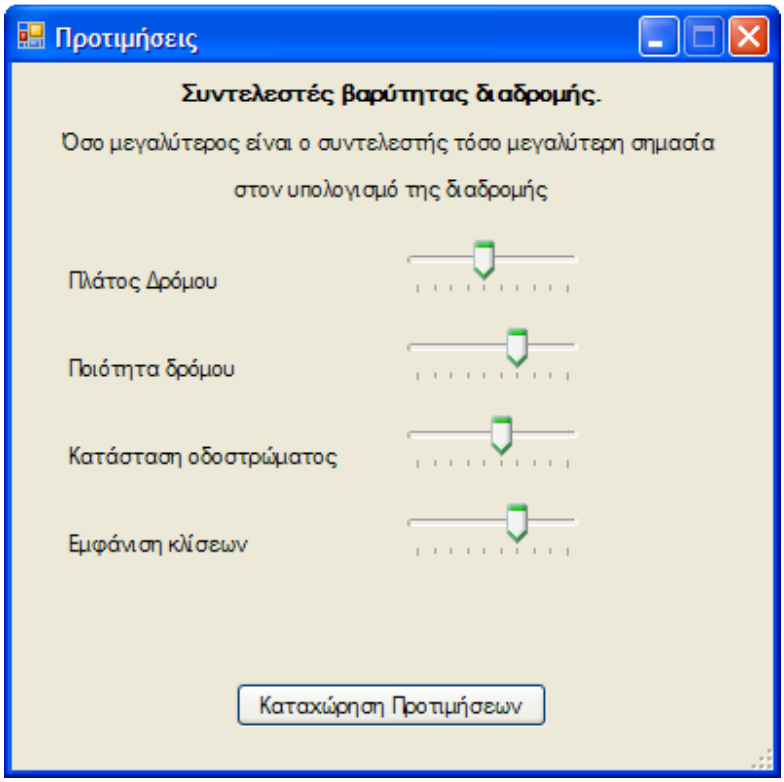


Εικόνα 38. Διαδρομή επιβατικό αυτοκίνητο



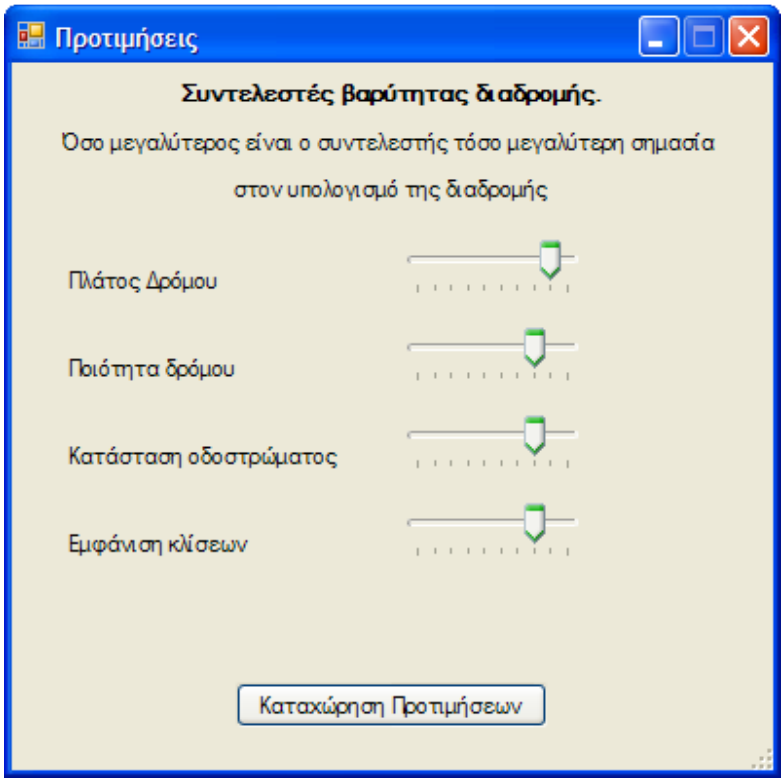
Εικόνα 39. Διαδρομή αγροτικό αυτοκίνητο

Και αποτελέσματα διαδρομής για ίδιο όχημα αλλά διαφορετικές παραμέτρους (Εικόνα 40, 41, 42, 43)

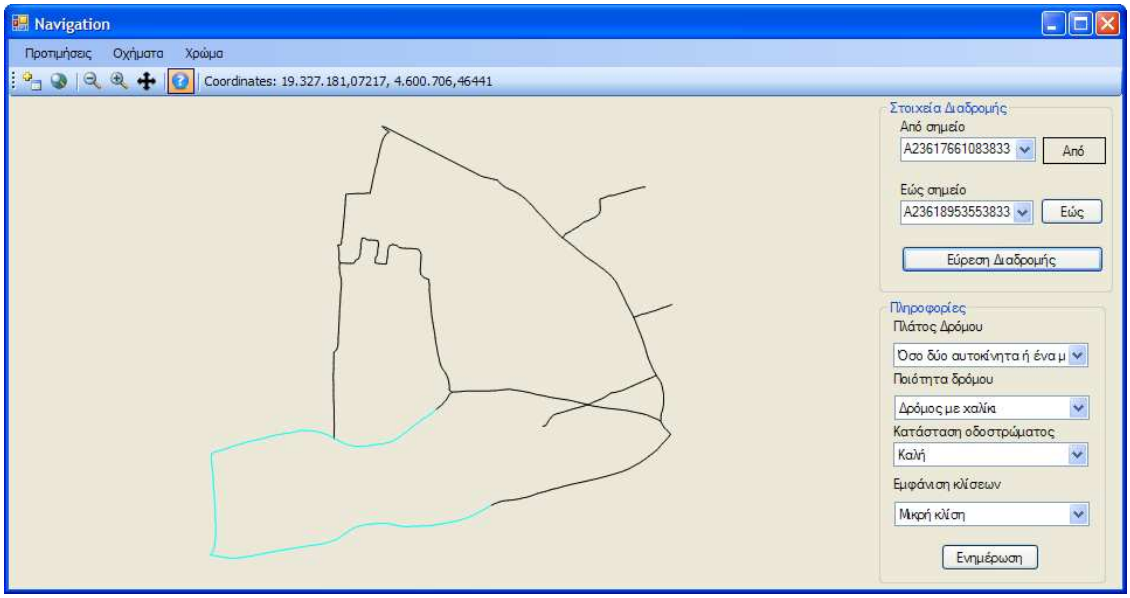


Εικόνα 40. Παράμετροι πλοήγησης

Εικόνα 41. Διαδρομή αγροτικό αυτοκίνητο



Εικόνα 42. Παράμετροι πλοήγησης



Εικόνα 43. Διαδρομή αγροτικό αυτοκίνητο

Είναι προφανές ότι η διαδρομές διαφέρουν αρκετά μεταξύ τους και ότι η συντομότερη διαδρομή δεν είναι πάντα η καλύτερη διαδρομή.

7. ΣΥΖΗΤΗΣΗ-ΣΥΜΠΕΡΑΣΜΑΤΑ

Στο σύστημα που παρουσιάστηκε στη παρούσα πτυχιακή απαιτήθηκε η εισαγωγή ενός μικρού αριθμού παραμέτρων στο λογισμικό πλοήγησης. Όπως είδαμε από τα αποτελέσματα της εφαρμογής ακόμα και αυτός ο μικρός αριθμός παραμέτρων ήταν ικανός να μας δώσει αρκετά διαφορετικά αποτελέσματα αναλόγως με τις τιμές εισόδου. Είναι προφανές ότι ένα ολοκληρωμένο σύστημα πλοήγησης που θα περίεχε ένα αρκετά μεγαλύτερο αριθμό παραμέτρων οι οποίες θα μπορούσαν να ενημερώνονται σε άμεση σύνδεση με το διαδίκτυο είτε από βάσεις δεδομένων που θα μας παρείχαν ακριβείς πληροφορίες για παράδειγμα για τη συγκεκριμένη μάρκα οχήματος και όχι μόνο για τη κατηγορία (για παράδειγμα το σύστημα αυτό θα μπορούσε να συνδεθεί και να βρει ακριβής πληροφορίες για ένα Ford Focus μοντέλο το 2005, όπως το βάρος, το μήκος, το βάρος οχήματος, τιμές κατανάλωσης καυσίμου) είτε από υπηρεσίες που θα μπορούσαν να μας δώσουν πληροφορίες για τις καιρικές συνθήκες που επικρατούν ή θα επικρατήσουν κατά τη διάρκεια της διαδρομής. Ακόμα το σύστημα θα μπορούσε να είναι συνδεμένο με συστήματα παρακολούθησης κυκλοφορίας και να λαμβάνει υπόψη του τη κίνηση στους δρόμους ή και ακόμα να λαμβάνει υπόψη τις προτεραιότητες στους δρόμους. Εάν το σύστημα υλοποιούταν σε αυτό το βαθμό τότε θα είχαμε πραγματικά τη βέλτιστη διαδρομή για το κάθε χρήστη προσωποποιημένη μόνο για αυτόν.

Ήδη αρχίζουν και εμφανίζονται εμπορικά πακέτα στην αγορά που προσπαθούν να ενσωματώσουν ανάλογες παραμέτρους όπως το MLS (Εικόνα 44) Destinator live TRAFFIC που ειδοποιεί τον οδηγό, σε πραγματικό χρόνο, για την κίνηση στους δρόμους της πόλης αλλά και τα έκτακτα συμβάντα και τον οδηγεί στον προορισμό του από εναλλακτικές διαδρομές (Technopress),

Εικόνα 44. MLS Destinator live TRAFFIC

ή το TomTom GO 1000 LIVE (Εικόνα 45) στο οποίο γίνεται πλοήγηση με υποβοήθηση λωρίδων για την διευκόλυνση της οδήγησης στους αυτοκινητόδρομους και σε δρόμους ταχείας κυκλοφορίας (Trustedreviews).

Εικόνα 45. MLS TomTom GO 1000 LIVE

ΒΙΒΛΙΟΓΡΑΦΙΑ

Ahuja, R. K., Mehlhorn, K., Orlin, J. B., & Tarjan, R. E. (1988). Faster algorithms for the shortest path problem. *Technical Report CS-TR-154-88* .

Aronoff, S. (1989). Geographic Information Systems: A Management Perspective. *WDL Publications* .

Bellman, R. E. (1958). On a pouting problem. *Quart. Appl. Math 16* , 87,90.

Blivice, S. (1974). Pedestrian Route Choice: a Study of Walking to Work in Munich. *Ph.D. Diss. University of Michigan* .

Burnett, G. E., & Porter, J. M. (2002). An empirical comparison of the use of distance versus landmark information within the Human-Machine Interface for vehicle navigation systems. *Shaker Publishing* , 49-64.

Burns, P. (1997). Navigation and the older driver. (L. University, Επιμ.) *Human Sciences* .

Burrough, P. A., & McDonnell, R. A. (1996). Principles of Geographical Information Systems. *Oxford University Press* .

Cantone, D., & Faro, S. (2004). Two-Levels-Greedy: A Generalization of Dijkstra's Shortest Path Algorithm. *Electronic Notes in Discrete Mathematics 17* , 81-86.

Cheverst, K., Davies, N., Mitchell, K., Friday, A., & Efstratiou, C. (1998). Developing a context-aware electronic tourist guide: Some issues and experience. *CHI* , 7-24.

Devlin, G. J., McDonnell, K., & Ward, S. (2008). Timber haulage routing in Ireland: an analysis using GIS and GPS. *Journal of Transport Geography 16* , 63-72.

Dial, R. B. (1969). Algorithm 360: shortest path forest with topological ordering. *Commun. ACM 12* , 632–633.

Dijkstra, E. (1959). A note on two problems in connection with graphs. *Numer. Math. 1* , 269–271.

Dijkstra, E. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik 1* , 269-271.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik 1* , 269–271.

ESRI. (2010). *ArcGIS: A Complete Integrated System*. Ανάκτηση 2010, από ESRI Web Site: http://www.esri.com/software/arcgis/

Ford Jr, L. R., & Fulkerson, D. R. (1962). Flows in Networks. *Princeton University Press* .

Fredman, M. L., & Tarjan, R. E. (1987). Fibonacci heaps and their uses in improved network optimization algorithm. *J. ACM 34* , 596–615.

Gabow, H. N., & Tarjan, R. .. (1989). Faster scaling algorithms for network problems. *SIAM J. Comput.* , 1013–1036.

Gallo, G., & Pallottino, S. (1988). Shortest paths algorithm. *Ann. Oper. Res. 13* , 3–79.

Goldberg, A. V. (1993). Scaling algorithms for the shortest paths problem. *Proceedings of the 4th ACM-SIAM Symposium on Discrete Algorithms*, (σσ. 222-231).

Grum, E. (2005). Danger of getting lost: Optimize a Path to Minimize Risk. *Proceedings,CORP 2005* .

Hill, M. R. (1984). Walking, Crossing Streets, and Choosing Pedestrian Routes. *NE: University of Lincoln* .

Kao, C. R. (1968). The Use of COmputers in the Processing and Analysis Geographic Information. *Spatial Analysis: A Reader in Statistical Geography* .

Kim, N., Lee, H. S., Oh, K. J., & Choi, J. Y. (2009). Context-aware mobile service for routing the fastest subway path. *Expert Systems with Applications 36* , 3319-3326.

Li, C. (2006). User preferences, information, transactions and location-based services: A study of urban pedestrian wayfinding. *Computers, Environment and Urban Systems* , 726-740.

Mark, D. M. (1989). A conceptual model for vehicle navigation systems. *Vehicle Navigation and Information Systems, IEEE* .

May, A., Ross, T., & Bayer, S. (2003). Drivers' Information Requirements when Navigating in an Urban Environment. *The Journal of Navigation* , 56, 89–100 .

Microsoft. (2010). *ASP.NET Wiki: ASP.NET*. Ανάκτηση 2010, από ASP.NET: http://wiki.asp.net/page.aspx/1332/aspnet/

Microsoft. (2010). *Microsoft Office XP*. Ανάκτηση 2010, από Microsoft Office XP: http://support.microsoft.com/ph/2533/en-gb

Microsoft. (2010). *SQL Server 2005 Technologies*. Ανάκτηση 2010, από Microsoft SQL Server 2005: http://www.microsoft.com/sqlserver/2005/en/us/product-information.aspx

Microsoft. (2010). *The C# Language*. Ανάκτηση 2010, από Visual C# Developer Center: http://msdn.microsoft.com/en-us/vcsharp/aa336809.aspx

Microsoft. (2010). *The Official Microsoft IIS Site*. Ανάκτηση 2010, από IIS: http://www.iis.net/

Microsoft. (2010). *Visual Studio Professional 2008*. Ανάκτηση 2010, από Microsoft Visual Studio: http://msdn.microsoft.com/en-us/vstudio/aa700830.aspx

Millonig, A. (2005). Menschliches Orientierungsverhalten – Eine Gegenuberstellung von Landmarkenbasierten und Zeichenbasierten Fusgangerleitsystemen. *Diploma Thesis, Dept. f. Raumentwicklung, Infrastruktur- und Umweltplanung* .

Moldes, F. (1995). Tecnología de los Sistemas de Información Geográfica. *Ra-Ma* .

Moreno, A. (2005). Sistemas y análisis de la información geográfica. Manual de autoaprendizaje con ArcGIS. *Ra-Ma* .

Oulasvirta, A., Kurvinen, E., & Kankainen, T. (2003). Understanding contexts by being there: Case studies in bodystorming. *Personal and Ubiquitous Computing* , 125-134.

Paay, J. (2003). Understanding and modelling physical environments for mobile location aware information services. *Mobile HCI* , 405-410.

Pamuk, A. (2006). Mapping Global Cities: GIS Methods in Urban Analysis. *Esri Pr.*

Shriver, K. (1997). Influence of Environmental Design on Pedestrian Travel Behavior in Four Austin Neighbourhoods. *Transportation Research Record 1578* , 64-75.

Sorrows, M., & Hirtle, S. (1999). The Nature of Landmarks for Real and Electronic Spaces. Στο D. M. C. Freksa (Επιμ.), *Spatial Information Theory, International Conference COSIT* (σσ. 37-50). Springer.

Technopress. (n.d.). *MLS Destinator Live Traffic*. Ανάκτηση από Technopress.gr: http://www.technopress.gr/2010/03/mls-destinator-live-traffic.html

Wang, F. (2006). Quantitative Methods and Applications in GIS. *CRC Pr I Llc.*

Wiener, J. M., Schnee, A., & Mallot, H. A. (2004). *Navigation Strategies in Regionalized Environments, Technical Report TR-121.* Max-Planck-Institut fur biologische Kybernetik, Universitat Tubingen.

Wikipedia. (n.d.). *Wikipedia*. Ανάκτηση από Dijkstra's algorithm - Wikipedia, the free encyclopedia: http://en.wikipedia.org/wiki/Dijkstra's_algorithm

Winter, S., & Corona, B. (2001). Datasets for Pedestrian Navigation Services. *Angewandte Geographische Informationsverarbeitung, Proceedings of the AGIT Symposium* , 84-89.

Zhai, S. (1991). An information structural model of vehicle navigation and its implications. *IEEE/SAE Conference on Vehicle Navigation and Information systems (VNIS '91).* Dearborn,Michigan.

Trustedreviews. (n.d.). *TomTom GO 1000 LIVE Review*. Ανάκτηση από Ψαρ Τεψη: http://www.trustedreviews.com/car-tech/review/2010/09/13/TomTom-GO-1000-LIVE/p1

8. ΠΑΡΑΡΤΗΜΑ - ΚΩΔΙΚΑΣ

Ο κώδικας της εφαρμογής που παρουσιάζεται εδώ αφορά τα κύρια μέρη της εφαρμογής. Το σύνολο του κώδικα βρίσκεται στο CD που συνοδεύει τη πτυχιακή.

8.1.1. Data Access Layer

```
using System;
using System.Data;
using System.Data.OleDb;
using XGPSMap.Xenophon;

namespace XGPS
{
    public class TPreferences
    {
        #region Class Member Declarations
        private Int32 _weight, _iD;
        private String _condition;
        #endregion


        #region Class Property Declarations
        public Int32 ID
        {
            get
            {
                return _iD;
            }
            set
            {
                Int32 iDTmp = value;
                if (iDTmp == 0)
                {
                    throw new ArgumentOutOfRangeException("ID", "ID
can't be 0");
                }
                _iD = value;
            }
        }


        public String Condition
        {
            get
            {
                return _condition;
            }
            set
            {
                _condition = value;
            }
        }
```

```
        public Int32 Weight
        {
            get
            {
                return _weight;
            }
            set
            {
                _weight = value;
            }
        }
        #endregion
        public TPreferences()
        {
            // Nothing for now.
        }



        public bool Update()
        {
            OleDbCommand cmdToExecute = new OleDbCommand();
            int _rowsAffected;
            cmdToExecute.CommandText = "UPDATE Preferences " +
                " SET " +
                " Weight = @iWeight " +
                " WHERE " +
                " ID = @iID ";


            cmdToExecute.CommandType = CommandType.Text;


            try
            {

                cmdToExecute.Parameters.Add(new
OleDbParameter("@iWeight", OleDbType.Integer, 4,
ParameterDirection.Input, true, 10, 0, "", DataRowVersion.Proposed,
_weight));
                cmdToExecute.Parameters.Add(new OleDbParameter("@iID",
OleDbType.Integer, 4, ParameterDirection.Input, false, 10, 0, "",
DataRowVersion.Proposed, _iD));
                // Execute query.
                _rowsAffected =
WinGateway.AccessExecuteNonQuery(cmdToExecute);

                if (_rowsAffected == 1)
                { return true; }
                else
                { return false; }

            }
            catch (Exception ex)
            {
```

```
                throw new Exception("Preferences::Update::Error
occured.", ex);
            }
            finally
            {

                cmdToExecute.Dispose();
            }
        }




        public DataTable SelectAll()
        {
            OleDbCommand cmdToExecute = new OleDbCommand();
            cmdToExecute.CommandText = "SELECT
[ID],[Condition],[Weight]" +
                " FROM [Preferences] " +
                " ORDER BY [ID] ASC";

            cmdToExecute.CommandType = CommandType.Text;
            DataTable toReturn;



            try
            {

                toReturn = WinGateway.AccessGetDataTable(cmdToExecute);
                return toReturn;
            }
            catch (Exception ex)
            {                throw new
Exception("Preferences::SelectAll::Error occured.", ex);
            }
            finally
            {
                cmdToExecute.Dispose();

            }
        }

        public bool UpdateWeight(int WidthWeight, int SurfaceWeight,
int SurfaceConWeight, int SlopeWeight,int DistanceWeight)
        {
            OleDbCommand cmdToExecute = new OleDbCommand();
            int _rowsAffected;

            WidthWeight =
Convert.ToInt32(System.Math.Pow(WidthWeight,3));
            SurfaceWeight =
Convert.ToInt32(System.Math.Pow(SurfaceWeight, 3));
            SurfaceConWeight =
Convert.ToInt32(System.Math.Pow(SurfaceConWeight, 3));
            SlopeWeight = Convert.ToInt32(System.Math.Pow(SlopeWeight,
3));
```

```
            DistanceWeight =
Convert.ToInt32(System.Math.Pow(DistanceWeight, 3));

            int den = WidthWeight + SurfaceWeight +
SurfaceConWeight+SlopeWeight;

            float relWidthWeight = (float)WidthWeight / den;
            float relSurfaceWeight = (float)SurfaceWeight / den;
            float relSurfaceConWeight = (float)SurfaceConWeight / den;
            float relSlopeWeight = (float)SlopeWeight / den;



                //string weightpart1 = "1/("+ WidthWeight.ToString() +
"*Width)+ 1/(" + SurfaceWeight.ToString() + "*Surface)+1/(" +
                //    SurfaceConWeight.ToString() + "*SurfaceCon)+1/("
+SlopeWeight.ToString() + "*Slope)+1/" +DistanceWeight.ToString();

            string weightpart1 = "1/((" + WidthWeight.ToString() +
"*Width*" + relWidthWeight.ToString().Replace(",",".") +") + (" +
                SurfaceWeight.ToString() + "*Surface*" +
relSurfaceWeight.ToString().Replace(",", ".") + ") + (" +
                SurfaceConWeight.ToString() + "*SurfaceCon*" +
relSurfaceConWeight.ToString().Replace(",", ".") + ") + (" +
                SlopeWeight.ToString() + "*Slope*" +
relSlopeWeight.ToString().Replace(",", ".") + "))";



             string fullweight = "2000000*Distance*(" + weightpart1 +
")";

                cmdToExecute.CommandText = "UPDATE RoadsDerived " +
                   " SET " +
                   " Weight =" + fullweight ;




            cmdToExecute.CommandType = CommandType.Text;


            try
            {

                // Execute query.
                _rowsAffected =
WinGateway.AccessExecuteNonQuery(cmdToExecute);

                if (_rowsAffected != 0)
                { return true; }
                else
                { return false; }

            }
            catch (Exception ex)
```

```
            {
                // some error occured. Bubble it to caller and
encapsulate Exception object
                throw new Exception("Preferences::Update::Error
occured.", ex);
            }
            finally
            {

                cmdToExecute.Dispose();
            }
        }


    }
}
```

```
using System;
using System.Data;
using System.Data.OleDb;
using XGPSMap.Xenophon;

namespace XGPS
{
    public class RoadsDerived
    {
        #region Class Member Declarations
        private Int32 _surfaceCon, _slope, _surface, _width;
        private string _ssurfaceCon, _sslope, _ssurface, _swidth;
        private Int32 _id, _distance, _streetBel;
        private Int32 _uID, _weight;
        private String _xFrom, _xTo;
        #endregion

        #region Class Property Declarations
        public Int32 Id
        {
            get
            {
                return _id;
            }
            set
            {
                _id = value;
            }
        }


        public Int32 Distance
        {
            get
            {
                return _distance;
            }
```

```
            set
            {
                _distance = value;
            }
        }


        public Int32 Width
        {
            get
            {
                return _width;
            }
            set
            {
                _width = value;
            }
        }


        public Int32 Surface
        {
            get
            {
                return _surface;
            }
            set
            {
                _surface = value;
            }
        }


        public Int32 SurfaceCon
        {
            get
            {
                return _surfaceCon;
            }
            set
            {
                _surfaceCon = value;
            }
        }


        public Int32 Slope
        {
            get
            {
                return _slope;
            }
            set
            {
                _slope = value;
            }
        }
```

```
        public string SWidth
        {
            get
            {
                return _swidth;
            }
            set
            {
                _swidth = value;
            }
        }


        public string SSurface
        {
            get
            {
                return _ssurface;
            }
            set
            {
                _ssurface = value;
            }
        }


        public string SSurfaceCon
        {
            get
            {
                return _ssurfaceCon;
            }
            set
            {
                _ssurfaceCon = value;
            }
        }


        public string SSlope
        {
            get
            {
                return _sslope;
            }
            set
            {
                _sslope = value;
            }
        }


        public Int32 StreetBel
        {
            get
            {
```

```
                return _streetBel;
            }
            set
            {
                _streetBel = value;
            }
        }


        public String XFrom
        {
            get
            {
                return _xFrom;
            }
            set
            {
                _xFrom = value;
            }
        }


        public String XTo
        {
            get
            {
                return _xTo;
            }
            set
            {
                _xTo = value;
            }
        }


        public Int32 UID
        {
            get
            {
                return _uID;
            }
            set
            {
                Int32 uIDTmp = (Int32)value;
                if (uIDTmp == 0)
                {
                    throw new ArgumentOutOfRangeException("UID", "UID
can't be 0");
                }
                _uID = value;
            }
        }


        public Int32 Weight
        {
            get
```

```
            {
                return _weight;
            }
            set
            {
                _weight = value;
            }
        }
        #endregion


        public RoadsDerived()
        {
            // Nothing for now.
        }

        public DataTable SelectOne()
        {
            OleDbCommand cmdToExecute = new OleDbCommand();
            DataTable toReturn;
            cmdToExecute.CommandText = "SELECT top 1
[Id],[Distance],[Width],
      [Surface],[SurfaceCon],[Slope],[StreetBel],[XFrom],[XTo],[UID],"
+
                " [Weight] " +
                " FROM [RoadsDerived] " +
                " WHERE [ID] = @iID ";


            cmdToExecute.CommandType = CommandType.Text;

            try
            {
                cmdToExecute.Parameters.Add(new OleDbParameter("@iID",
OleDbType.Integer, 4, ParameterDirection.Input, false, 10, 0, "",
DataRowVersion.Proposed, _id));
                toReturn = WinGateway.AccessGetDataTable(cmdToExecute);

                if (toReturn.Rows.Count == 1)
                {
                    _uID = Convert.ToInt32(toReturn.Rows[0]["uID"]);

                    _surfaceCon =
Convert.ToInt32(toReturn.Rows[0]["surfaceCon"]);
                    _slope =
Convert.ToInt32(toReturn.Rows[0]["slope"]);
                    _streetBel =
Convert.ToInt32(toReturn.Rows[0]["streetBel"]);
                    _surface =
Convert.ToInt32(toReturn.Rows[0]["surface"]);
                    _id = Convert.ToInt32(toReturn.Rows[0]["id"]);
                    _distance =
Convert.ToInt32(toReturn.Rows[0]["distance"]);
                    _width =
Convert.ToInt32(toReturn.Rows[0]["width"]);
                    _xFrom =
Convert.ToString(toReturn.Rows[0]["xFrom"]);
```

```
                    _xTo = Convert.ToString(toReturn.Rows[0]["xTo"]);

                    _swidth = GetWidthDesc(_surfaceCon);
                    _sslope = GetSlopeDesc(_slope);
                    _ssurface = GetSurfaceDesc(_surface);
                    _ssurfaceCon = GetSurfaceConditionDesc(_width);
                }

                return toReturn;
            }
            catch (Exception ex)
            {
                throw new Exception("RoadsDerived::SelectOne::Error
occured.", ex);
            }
            finally
            {
                cmdToExecute.Dispose();
            }


        }

        private string GetWidthDesc(int value)
        {
            string desc;

            desc = string.Empty;

            if (value == 1)
            {
                desc = "Όσο μία μηχανή";
            }
            else if (value == 2)
            {
                desc = "Όσο ένα αυτοκίνητο";
            }
            else if (value == 3)
            {
                desc = "Όσο δύο αυτοκίνητα ή ένα μεγάλο όχημα";
            }
            else if (value == 4)
            {
                desc = "Όσο ένα μεγάλο όχημα και ένα αυτοκίνητο";
            }
            else if (value == 5)
            {
                desc = "Όσο δύο μεγάλα οχήματα";
            }


            return desc;
        }

        private int GetWidthValue(string value)
        {
            int desc=0;
```

```
            if (value == "Όσο μία μηχανή")
            {
                desc = 1;
            }
            else if (value == "Όσο ένα αυτοκίνητο")
            {
                desc = 2;
            }
            else if (value == "Όσο δύο αυτοκίνητα ή ένα μεγάλο όχημα")
            {
                desc = 3;
            }
            else if (value == "Όσο ένα μεγάλο όχημα και ένα
αυτοκίνητο")
            {
                desc = 4;
            }
            else if (value == "Όσο δύο μεγάλα οχήματα")
            {
                desc = 5;
            }


            return desc;
        }

        private string GetSlopeDesc(int value)
        {
            string desc;

            desc = string.Empty;

            if (value == 1)
            {
                desc = "Μεγάλη κλίση";
            }
            else if (value == 2)
            {
                desc = "Μέτρια κλίση";
            }
            else if (value == 3)
            {
                desc = "Μικρή κλίση";
            }


            return desc;
        }

        private int GetSlopeValue(string value)
        {
            int desc;

            desc = 0;
```

```
            if (value == "Μεγάλη κλίση")
            {
                desc =1 ;
            }
            else if (value == "Μέτρια κλίση")
            {
                desc = 2;
            }
            else if (value == "Μικρή κλίση")
            {
                desc = 3;
            }


            return desc;
        }

        private string GetSurfaceDesc(int value)
        {
            string desc;

            desc = string.Empty;

            if (value == 1)
            {
                desc = "Χωματόδρομος";
            }
            else if (value == 2)
            {
                desc = "Δρόμος με χαλίκι";
            }
            else if (value == 3)
            {
                desc = "Άσφαλτος";
            }


            return desc;
        }

        private int GetSurfaceValue(string value)
        {
            int desc;

            desc = 0;

            if (value == "Χωματόδρομος")
            {
                desc = 1;
            }
            else if (value == "Δρόμος με χαλίκι")
            {
                desc = 2;
            }
            else if (value == "Άσφαλτος")
            {
                desc = 3;
```

```
            }


            return desc;
        }

        private string GetSurfaceConditionDesc(int value)
        {
            string desc;

            desc = string.Empty;

            if (value == 1)
            {
                desc = "Πολύ κακή";
            }
            else if (value == 2)
            {
                desc = "Κακή";
            }
            else if (value == 3)
            {
                desc = "Μέτρια";
            }
            else if (value == 4)
            {
                desc = "Καλή";
            }
            else if (value == 5)
            {
                desc = "Πολύ Καλή";
            }


            return desc;
        }

        private int GetSurfaceConditionValue(string value)
        {
            int desc;

            desc = 0;

            if (value == "Πολύ κακή")
            {
                desc = 1;
            }
            else if (value == "Κακή")
            {
                desc = 2;
            }
            else if (value == "Μέτρια")
            {
                desc = 3;
            }
            else if (value == "Καλή")
            {
```

```
                desc = 4;
            }
            else if (value == "Πολύ Καλή")
            {
                desc = 5;
            }


            return desc;
        }

        public bool UpdateStreet(string fWidth, string fSurface, string
fSurfaceCon, string fSlope)
        {
            int _rowsAffected;
            OleDbCommand cmdToExecute = new OleDbCommand();
            cmdToExecute.CommandText = "UPDATE RoadsDerived SET " +
                "RoadsDerived.Width = @fWidth," +
                "RoadsDerived.Surface = @fSurface," +
                "RoadsDerived.SurfaceCon = @fSurfaceCon," +
                "RoadsDerived.Slope = @fSlope" +
                " where StreetBel=@fStreetBel";

            cmdToExecute.CommandType = CommandType.Text;


            try
            {

                cmdToExecute.Parameters.Add(new
OleDbParameter("@fWidth", OleDbType.Integer, 4,
ParameterDirection.Input, true, 10, 0, "", DataRowVersion.Proposed,
GetWidthValue(fWidth)));
                cmdToExecute.Parameters.Add(new
OleDbParameter("@fSurface", OleDbType.Integer, 4,
ParameterDirection.Input, true, 10, 0, "", DataRowVersion.Proposed,
GetSurfaceValue(fSurface)));
                cmdToExecute.Parameters.Add(new
OleDbParameter("@fSurfaceCon", OleDbType.Integer, 4,
ParameterDirection.Input, true, 10, 0, "", DataRowVersion.Proposed,
GetSurfaceConditionValue(fSurfaceCon)));
                cmdToExecute.Parameters.Add(new
OleDbParameter("@fSlope", OleDbType.Integer, 4,
ParameterDirection.Input, true, 10, 0, "", DataRowVersion.Proposed,
GetSlopeValue(fSlope)));
                cmdToExecute.Parameters.Add(new
OleDbParameter("@fStreetBel", OleDbType.Integer, 4,
ParameterDirection.Input, true, 10, 0, "", DataRowVersion.Proposed,
_streetBel));


                // Execute query.
                _rowsAffected =
WinGateway.AccessExecuteNonQuery(cmdToExecute);
```

```
                return true;
            }
            catch (Exception ex)
            {
                throw new Exception("RoadsDerived::Insert::Error
occured.", ex);
            }
            finally
            {
                cmdToExecute.Dispose();
            }
        }




    }
}
```

8.1.2. Database Layer

```
using System.Data.SqlClient;
using System.Data.OleDb;
using System.Data;
using System;

namespace XGPSMap
{
    namespace Xenophon
    {
        public class WinGateway
        {

            private static string _AccessConnectionString =
"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=GPSData.mdb;User
Id=admin;Password=;";
            //private static SqlConnection _cn = new SqlConnection();
            private static OleDbConnection _cn = new
OleDbConnection();


            #region "Access"
            public static OleDbConnection AccessGetConnection()
            {
                 if (_cn.State == ConnectionState.Closed)
                {
                     _cn.ConnectionString = _AccessConnectionString;
                }

                return _cn;
            }

            public static DataTable AccessGetDataTable(string strSQL)
            {
                return AccessGetDataTable(new OleDbCommand(strSQL));
            }

            public static DataTable AccessGetDataTable(OleDbCommand
cmd)
            {
                OleDbConnection cn = AccessGetConnection();
                OleDbDataAdapter da = new OleDbDataAdapter(cmd);
                DataTable dt = new DataTable();

                cmd.Connection = cn;
                if (cn.State == ConnectionState.Closed)
                {
                 cn.Open();
                }

                try
                {
```

```
                    da.Fill(dt);
                }
                catch (Exception ex)
                {
                    throw ex;
                }
                return dt;
            }
            public static void AccessFill(DataTable dt, OleDbCommand
cmd)
            {
                OleDbConnection cn = AccessGetConnection();
                OleDbDataAdapter da = new OleDbDataAdapter(cmd);
                cmd.Connection = cn;
                if (cn.State == ConnectionState.Closed)
                {
                    cn.Open();
                }

                try
                {
                    da.Fill(dt);
                }
                catch (Exception ex)
                {
                    throw ex;
                }

            }

            public static object AccessGetLastId(OleDbConnection conn)
            {
                OleDbCommand cmd = new OleDbCommand();
                cmd.CommandText = "SELECT @@IDENTITY";
                cmd.Connection = conn;
                return cmd.ExecuteScalar();
            }

            public static object AccessExecuteScalar(string strSQL)
            {
                return AccessExecuteScalar(new OleDbCommand(strSQL));
            }

            public static object AccessExecuteScalar(OleDbCommand cmd)
            {
                OleDbConnection cn = AccessGetConnection();
                object res;
                cmd.Connection = cn;

                if (cn.State == ConnectionState.Closed)
                {
                    cn.Open();
                }

                try
                {
                    //res = cmd.ExecuteNonQuery
```

```
                    res = cmd.ExecuteScalar();
                }
                catch (Exception ex)
                {
                    throw ex;
                }
                return res;
            }

            public static int AccessExecuteNonQuery(string strSQL)
            {
                OleDbConnection cn = AccessGetConnection();
                int res;
                OleDbCommand cmd = new OleDbCommand();
                cmd.CommandText = strSQL;
                cmd.Connection = cn;
                if (cn.State == ConnectionState.Closed)
                {
                    cn.Open();
                }

                try
                {
                    res = cmd.ExecuteNonQuery();
                }
                catch (Exception ex)
                {
                    throw ex;
                }
                return res;
            }
            public static int AccessExecuteNonQuery(OleDbCommand cmd)
            {
                OleDbConnection cn = AccessGetConnection();
                int res;
                cmd.Connection = cn;
                if (cn.State == ConnectionState.Closed)
                {
                    cn.Open();
                }

                try
                {
                    res = cmd.ExecuteNonQuery();
                }
                catch (Exception ex)
                {
                    throw ex;
                }
                return res;
            }
            #endregion
        }
    }
}
```

8.1.3. Dijkstra

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

namespace XGPSMap.Dijkstra
{
    class XDijkstra
    {
        private int rank = 0;
        double [,] L;
        private int[] C;
        public double[] D;
        private int trank = 0;
        public XDijkstra(int paramRank, double[,] paramArray)
        {
            L = new double[paramRank, paramRank];
            C = new int[paramRank];
            D = new double[paramRank];
            rank = paramRank;
            for (int i = 0; i < rank; i++)
            {
                for (int j = 0; j < rank; j++)
                {
                    L[i, j] = paramArray[i, j];
                }
            }

            for (int i = 0; i < rank; i++)
            {
                C[i] = i;
            }
            C[0] = -1;
            for (int i = 1; i < rank; i++)
                D[i] = L[0, i];
        }
        public void DijkstraSolving()
        {
            double minValue = Int32.MaxValue;
            int minNode = 0;
            for (int i = 0; i < rank; i++)
            {
                if (C[i] == -1)
                    continue;
                if (D[i] > 0 && D[i] < minValue)
                {
                    minValue = D[i];
                    minNode = i;
                }
            }
            C[minNode] = -1;
            for (int i = 0; i < rank; i++)
            {
                if (L[minNode, i] < 0)
```

```
                    continue;
                if (D[i] < 0)
                {
                    D[i] = minValue + L[minNode, i];
                    continue;
                }
                if ((D[minNode] + L[minNode, i]) < D[i])
                    D[i] = minValue + L[minNode, i];
            }
        }
        public void Run()
        {
            for (trank = 1; trank < rank; trank++)
            {
                DijkstraSolving();

            }
        }
    }

}
```

8.1.4. Utilities

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Data;
using XGPSMap.Xenophon;

namespace XGPSMap.Utils
{
    class CreateData
    {
        public string restrinctions;

        public string[] GetMapNodes()
        {
            DataTable dt;

            if (restrinctions != string.Empty)
            {
                 dt = WinGateway.AccessGetDataTable("Select Distinct
XFrom from RoadsDerived where " + restrinctions + " and StreetBel<>ID
order by XFrom");

            }
            else
            {
                 dt = WinGateway.AccessGetDataTable("Select Distinct
```

```
XFrom from RoadsDerived where StreetBel<>ID order by XFrom");

            }

            string[] nodes = new string[dt.Rows.Count];

            for (int i = 0; i < dt.Rows.Count; i++)
            {
                nodes[i] = dt.Rows[i][0].ToString();
            }

            return nodes;
        }

        public double[,] GetWeights()
        {

            string[] nodes = GetMapNodes();

            double[,] weights = new double[nodes.GetUpperBound(0),
nodes.GetUpperBound(0)];

            for (int i = 0; i < nodes.GetUpperBound(0); i++)
            {
                for (int j = 0; j < nodes.GetUpperBound(0); j++)
                {
                    DataTable dt =
WinGateway.AccessGetDataTable("Select weight From RoadsDerived where
XFrom='" + nodes[i] + "' and [XTo]='" + nodes[j] + "';");

                    if (dt.Rows.Count == 0)
                    {
                        weights[i, j] = -1;

                    }
                    else
                    {
                        weights[i, j] =
Convert.ToDouble(dt.Rows[0][0]);
                    }
                }

            }

            return weights;
        }

        public DataTable nodeConnections()
        {
            DataTable dt;

            if (restrinctions != string.Empty)
            {
                dt = WinGateway.AccessGetDataTable("Select Distinct
ID,XFrom,[XTo],Weight from RoadsDerived where " + restrinctions + " and
StreetBel<>ID order by XFrom");
            }
```

```
            else
            {
                dt = WinGateway.AccessGetDataTable("Select ID,Distinct
XFrom,[XTo],Weight from RoadsDerived where StreetBel<>ID order by
XFrom");
            }


            return dt;
        }

        public DataTable nodes()
        {
            DataTable dt;
            if (restrinctions != string.Empty)
            {
                 dt = WinGateway.AccessGetDataTable("Select Distinct
XFrom from RoadsDerived where " + restrinctions + " and StreetBel<>ID
order by XFrom");
            }
            else
            {
                 dt = WinGateway.AccessGetDataTable("Select Distinct
XFrom from RoadsDerived where StreetBel<>ID order by XFrom");
            }


            return dt;
        }

        public string RoadId(string BeginPosition, string EndPos)
        {
            DataTable dt = WinGateway.AccessGetDataTable("Select top 1
ID from RoadsDerived where [XFrom]='" + BeginPosition + "' and [XTo]='"
+ EndPos + "'");

            return dt.Rows[0][0].ToString();
        }
    }
}
```