ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ &
ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Ανάπτυξη Διαδικτυακής Εφαρμογής Δημιουργίας και Προβολής Ανυσματικών Χαρτών Πλοήγησης με Χρήση Ανοικτού Λογισμικού

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Γεώργιος Δ. Σταματούκος

ΕΠΙΒΛΕΠΩΝ: Νικόλαος Μήτρου
Καθηγητής ΕΜΠ

Αθήνα, Μάρτιος 2012

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ &
ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Ανάπτυξη Διαδικτυακής Εφαρμογής Δημιουργίας και Προβολής Ανυσματικών Χαρτών Πλοήγησης με Χρήση Ανοικτού Λογισμικού

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Γεώργιος Δ. Σταματούκος

ΕΠΙΒΛΕΠΩΝ: Νικόλαος Μήτρου
Καθηγητής ΕΜΠ

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 28 Μαρτίου 2012

.....	.....	.....
Νικόλαος Μήτρου	Μιχαήλ Θεολόγου	Συμεών Παπαβασιλείου
Καθηγητής ΕΜΠ	Καθηγητής ΕΜΠ	Καθηγητής ΕΜΠ

Αθήνα, Μάρτιος 2012

Γεώργιος Δ. Σταματούκος

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Copyright © Γεώργιος Δ. Σταματούκος.

Με επιφύλαξη παντός δικαιώματος. All rights reserved.

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα.

Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και δεν πρέπει να ερμηνευθεί ότι αντιπροσωπεύουν τις επίσημες θέσεις του Εθνικού Μετσόβιου Πολυτεχνείου.

Περίληψη

Ο σκοπός της διπλωματικής εργασίας ήταν η ανάπτυξη μιας εφαρμογής για τη διαδραστική (on the fly) απεικόνιση/οπτικοποίηση διανυσματικών εικόνων χαρτών μικρής κλίμακας, χρησιμοποιώντας ανοικτό λογισμικό και υλοποιώντας το μοντέλο client/server.

Συγκεκριμένα, έγινε μελέτη των σύγχρονων τάσεων και τεχνολογιών για την ανάπτυξη συστημάτων πλοήγησης και ειδικότερα του τρόπου μετατροπής και οπτικοποίησης των χωρικών δεδομένων, που είναι αποθηκευμένες στον server, σε εικόνες χαρτών στην πλευρά του client. Αναζητήθηκαν και χρησιμοποιήθηκαν εργαλεία ανοικτού λογισμικού για τη δημιουργία της απαραίτητης χωρικής βάσης δεδομένων και συγκεράζοντας σύγχρονες τεχνολογίες (AJAX, DOM, php, XML, javascript) διαδικτυακού λογισμικού υλοποιήθηκε μια εφαρμογή η οποία επιτρέπει την απεικόνιση διανυσματικών πολυεπίπεδων χαρτών πλοήγησης στον client με βάση επιλογές που τίθονται δυναμικά (on the fly) από τον χρήστη.

Με δεδομένο ότι ο client μπορεί να είναι μια φορητή συσκευή (smartphone, tablet, κλπ), η παρούσα εργασία θα μπορούσε να αποτελέσει τη βάση για τη δημιουργία μιας ενιαίας πλατφόρμας παροχής ανθρωποκεντρικής υπηρεσίας πλοήγησης εντός εσωτερικών χώρων (π.χ. συνεδριακά και εμπορικά κέντρα).

Λέξεις Κλειδιά

Δημιουργία Χαρτών, Σύστημα Συντεταγμένων, Χωρικές Βάσεις Δεδομένων, Διανυσματικοί Χάρτες, Δρομολόγηση

Abstract

The scope of this thesis was the development of an application for interactive (on the fly) imaging/visualization of small-scale maps as vector type images, by using open source software platform and implementing the client/server model.

Specifically, the latest trends and technologies for the development of navigation systems were studied and in particular how the spatial data which are stored in databases in the server side are being converted/transformed and visualized in map images in the client side. Open source tools for creating the necessary spatial database were sought and used. By bringing together modern web technologies (AJAX, DOM, php, XML, javascript), an application was implemented that allows the visualization of multi-level vector map images on the client side, based on the options that are being dynamically set (on the fly) by the user.

Given that the client could be a mobile device (smartphone, tablet, etc), this thesis could form the basis for the creation of a unified platform for providing a human based service for indoor navigation (e.g. in conference and shopping centers.

Key Words

Map Creation, Coordinate System, Spatial Databases, Vector Maps, Routing

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΕΦΑΛΑΙΟ 1

ΕΙΣΑΓΩΓΗ

1.1. Γενικά

Η τεχνολογική πρόοδος που σημειώθηκε τα τελευταία χρόνια στο πεδίο των κινητών συσκευών, των ασύρματων δικτύων και των εφαρμογών, σε συνάρτηση με τις ανάγκες της αγοράς, οδήγησαν σε μια ραγδαία εξάπλωση των ασύρματων προσωπικών επικοινωνιών (wireless personal communications). Αυτό μπορεί να παρατηρηθεί και από την εξέλιξη της $2^{ης}$, $3^{ης}$ και $4^{ης}$ γενιάς των κινητών τηλεπικοινωνιών αλλά και από την ευρεία εγκατάσταση ασύρματων τοπικών δικτύων (Wireless Local Area Networks - WLAN). Όλα αυτά τα σύγχρονα δίκτυα, με τη σειρά τους, μπορούν να υποστηρίξουν την υλοποίηση του οράματος για διάχυτες υπηρεσίες (ubiquitous services), οι οποίες θα βοηθούν τους χρήστες στις καθημερινές τους δραστηριότητες με έναν «ευφυή» αλλά και διακριτικό (unobtrusive) τρόπο. Τέτοιες ακριβώς υπηρεσίες έχουν περιγραφεί και από το ISTAG (Information Society Technologies Advisory Group) στα πλαίσια του οράματος για «Ambient Intelligence» (το οποίο είναι ευρύτερα γνωστό με τον όρο Διάχυτος Υπολογισμός ή Pervasive Computing). Μερικά από τα κύρια στοιχεία του Διάχυτου Υπολογισμού (ΔΥ) είναι η αξιοποίηση της πληροφορίας πλαισίου (contextual information), οι διάσπαρτες στο χώρο, και πάντα δικτυωμένες, υπολογιστικές διατάξεις, οι προηγμένες διεπαφές χρήστη κ.α.

Εκτός βέβαια από τις δικτυακές ή άλλες υποδομές, η υλοποίηση του ΔΥ απαιτεί και ένα είδος «ευφυΐας» από τις εφαρμογές και τις υπηρεσίες. Ο σημασιολογικός «εμπλουτισμός» των διαφόρων συστημάτων που εμπλέκονται σε ένα περιβάλλον διάχυτου υπολογισμού μπορεί να προσδώσει στο περιβάλλον αυτό κάποια μορφή «ευφυΐας». Με τον όρο «ευφυΐα» εννοούμε την ικανότητα που έχει κάποιο υπολογιστικό σύστημα να κάνει εξαγωγή συμπερασμάτων, να προβλέπει μελλοντικές καταστάσεις ή να αντιλαμβάνεται συγκεκριμένες καταστάσεις του περιβάλλοντος με αυτόματο ή τουλάχιστον ημι-αυτόματο τρόπο. Μια μοντελοποίηση συστημάτων που περιλαμβάνει και τη σημασιολογία τους μπορεί να οδηγήσει στην ανάπτυξη εφαρμογών με προηγμένη λειτουργικότητα που θα είναι πολύ πιο αποτελεσματικές και φιλικές προς το χρήστη. Ο κύριος τρόπος αναπαράστασης και χρήσης της σημασιολογίας σήμερα είναι η μοντελοποίηση με τη χρήση οντολογιών (ontologies). Οι οντολογίες είναι εννοιολογικά μοντέλα που μπορούν να αναπαραστήσουν με ικανή εκφραστικότητα τη γνώση ενός συγκεκριμένου πεδίου (domain). Η γνώση αυτή αποτελείται από τις έννοιες του πεδίου,

τις συσχετίσεις μεταξύ των εννοιών, τα αξιώματα που ισχύουν κλπ. Μια τέτοια αναπαράσταση της γνώσης σε συνδυασμό με μηχανισμούς εξαγωγής συμπερασμάτων (reasoning - inference) δίνει τη δυνατότητα στις υπηρεσίες να ανακαλύψουν «κρυφές» συσχετίσεις μεταξύ των οντοτήτων του συστήματος και να παρέχουν λύσεις σε δυσεπίλυτα προβλήματα.

Η παρούσα εργασία μελετά πώς μπορούν να υλοποιηθούν πιο ανθρωποκεντρικές υπηρεσίες θέσης για τα μελλοντικά περιβάλλοντα, εστιάζοντας στην πλοήγηση σε εσωτερικούς χώρους. Ο χαρακτηρισμός «ανθρωποκεντρικές υπηρεσίες θέσης», στα πλαίσια αυτής της εργασίας, ερμηνεύεται σαν εξατομικευμένες, αποδοτικές, ευφυείς και φιλικές υπηρεσίες θέσης. Ο λόγος που μελετάται η πλοήγηση είναι ότι έχει μεγαλύτερη πολυπλοκότητα αλλά και χρηστική αξία από τις υπόλοιπες υπηρεσίες θέσης.

Ένα δημοφιλές παράδειγμα για τις υπηρεσίες πλοήγησης εξαρτώμενες απ' τη θέση είναι: «Βρες μου το κοντινότερο φαρμακείο». Η ερώτηση αυτή περιλαμβάνει πολύ περισσότερα από αυτά που φαίνονται με τη πρώτη ματιά. Σε πρώτη φάση απαιτείται ο εντοπισμός της θέσης του ατόμου. Αν βρίσκεται σε εξωτερικό χώρο έχουμε τη γνωστή λύση του GPS (Global Positioning System). Ο εντοπισμός της θέσης σε εσωτερικό χώρο ή σε υπόγειες διαβάσεις ή τούνελ είναι μια άλλη υπόθεση. Εδώ δε γίνεται να χρησιμοποιηθεί GPS, αλλά έχουν προταθεί κάποιες ιδιαίτερα ενδιαφέρουσες λύσεις όπως τα RFID tags ή οι infrared beacons. Οι λύσεις αυτές είναι απόλυτα λειτουργικές και υλοποιήσιμες.

Αφού εντοπιστεί η θέση του ατόμου, δηλαδή οι (x, y) συντεταγμένες, θα πρέπει να γίνει μια κατάλληλη επερώτηση σε χωρική βάση δεδομένων η οποία να ανακτήσει το φαρμακείο το οποίο έχει τη μικρότερη απόσταση από το άτομο. Αυτή είναι μια απλή επερώτηση για τις χωρικά εκτεταμένες σύγχρονες βάσεις δεδομένων και η οποία μάλιστα υλοποιείται με τη χρήση μίας μόνο έτοιμης συνάρτησης. Όμως η Ευκλείδεια απόσταση μεταξύ δύο σημείων είναι αυτό που μας ενδιαφέρει; Φυσικά μια εταιρεία τηλεπικοινωνιών δε μπορεί να έχει την απαίτηση οι πελάτες της να περνάνε μέσα από τοίχους και άρα αυτό που μας ενδιαφέρει, η διαδοχή δηλαδή των δρόμων ή αλλιώς το μονοπάτι, δε μπορούμε να το εξάγουμε με μια απλή επερώτηση από τη βάση. Βέβαια και γι' αυτό το θέμα έχει γίνει αρκετή έρευνα και υπάρχει πληθώρα αλγορίθμων για την εύρεση μονοπατιών και άρα τη δρομολόγηση του χρήστη σ' έναν αυτοσχέδιο γράφο. Ως τώρα, η ερώτηση μας είναι μάλλον απλή. «Βρες το κοντινότερο φαρμακείο». Η απάντηση της είναι σαφώς σημαντική αλλά τι θα γινόταν αν ενδιαφερόμασταν να την εμπλουτίσουμε έτσι ώστε να μας εξυπηρετεί σε περισσότερες από μία εργασίες. Αν για

παράδειγμα, θέλαμε να δρομολογηθούμε στο κοντινότερο φαρμακείο περνώντας όμως και από ένα σούπερ – μάρκετ ή και να διασχίσουμε το αγαπημένο μας, βάσει ιστορικού επισκέψεων, πάρκο. Αυτές όλες είναι επιλογές που ο πελάτης πιθανά δε θα τις εκφράσει αλλά αν υπάρχει ένα ιστορικό διαδρομών του χρήστη μπορούν να χρησιμοποιηθούν την κατάλληλη στιγμή.

Εδώ εισάγεται σιγά σιγά η έννοια της σημασιολογίας του χώρου και η ερώτησή μας δεν είναι πια για το συντομότερο μονοπάτι προς κάποιο φαρμακείο, αλλά για τη συντομότερη διαδρομή με ειδικό ενδιαφέρον για το χρήστη. Αυτό απαιτεί την πολύ δύσκολη, αλλά και πολύ χρήσιμη εξατομίκευση των εφαρμογών. Και ίσως να φαίνεται πολυτέλεια η δρομολόγηση του χρήστη από ένα μονοπάτι που έχει καταστήματα με εκπτώσεις, αλλά αν ο χρήστης έχει ειδικές ανάγκες (π.χ. βρίσκεται σε αναπηρική καρέκλα) τότε πρέπει να δρομολογηθεί από μία διαδρομή που δεν περιλαμβάνει σκάλες ή τα πεζοδρόμια είναι αρκετά μεγάλα.

Σε αυτό το χώρο, της ειδικής σημασιολογίας του χώρου υπάρχουν πολλά ανοιχτά θέματα. Υπάρχει επίσης και το θέμα των χρονικών δεδομένων και των τρόπων που αυτά μπορούν να χρησιμοποιηθούν για να χαρακτηρίσουν το χώρο. Για παράδειγμα, όταν ένας δρόμος παρουσιάζει κυκλοφοριακή συμφόρηση μεταξύ οκτώ και δέκα το πρωί τότε ο χρήστης δεν μπορεί να δρομολογηθεί μέσω αυτού αν η ώρα είναι π.χ. εννιάμιση. Ο όγκος των χωρικών και χρονικών δεδομένων είναι τόσο μεγάλος ώστε πληθώρα ερευνών λαμβάνει χώρα με στόχο την καλύτερη δυνατή διαχείρισή τους.

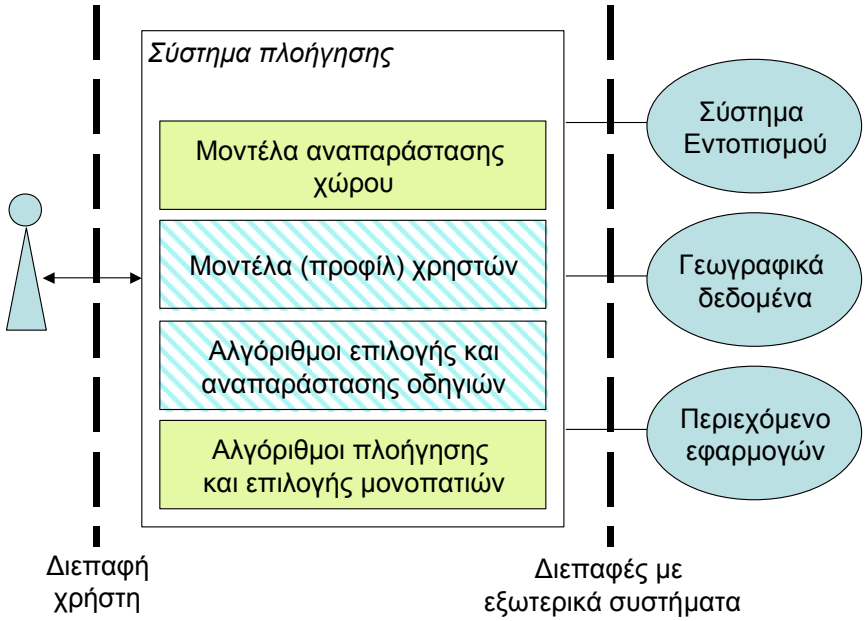
Ακόμη περισσότερο αυξάνει ο όγκος των δεδομένων αν μιλήσουμε για την παρακολούθηση κινούμενων αντικειμένων. Γίνεται προσπάθεια να αποφεύγεται η διαρκής ανανέωση των δεδομένων της βάσης σχετικά με τη θέση των αντικειμένων για κάθε χρονική στιγμή. Η δυσκολία έγκειται στο γεγονός ότι οι βάσεις δεδομένων είναι σχεδιασμένες έτσι ώστε να διαχειρίζονται δεδομένα με σταθερές τιμές. Όταν η πληροφορία του χώρου μεταβάλλεται διαρκώς η λύση φυσικά δεν είναι τα δεδομένα της βάσης να ανανεώνονται διαρκώς. Αυτό εκτός από ακριβό είναι και αδύνατο.

Ακριβώς γι’ αυτό το λόγο έχουν αναπτυχθεί αλγόριθμοι οι οποίοι υπολογίζουν μια υποθετική τροχιά του κινούμενου αντικειμένου βασισμένοι στη μέση ταχύτητά του, στον προορισμό, αν αυτός είναι γνωστός, και σε πολλά άλλα στοιχεία. Εδώ μπορεί και πάλι να περιληφθεί σημασιολογία του χώρου καθώς αν γνωρίζουμε ότι ο χρήστης διασχίζει συγκεκριμένο δρόμο με στόχο να πάρει πρωινό σε συγκεκριμένο εστιατόριο, ο υπολογισμός της τροχιάς του θα περιλαμβάνει και μια δεκάλεπτη στάση στο συγκεκριμένο σημείο. Η διατήρηση ιστορικού είτε γενικά για το δρόμο (κίνηση, μέση

ταχύτητα κτλ.) είτε συγκεκριμένα για το χρήστη (ειδικά ενδιαφέροντα σε συγκεκριμένη, επαναλαμβανόμενη διαδρομή) είναι πολύ σημαντική.

Σε αυτό το σημείο κρίνεται σκόπιμο να αναφερθεί μια γενική αρχιτεκτονική αναφοράς (reference architecture) για συστήματα πλοήγησης. Η αρχιτεκτονική αυτή απεικονίζεται στην **Εικόνα 1**. Σε αυτήν μπορούμε να εντοπίσουμε τα παρακάτω βασικά συστατικά:

- *Διεπαφή χρήστη*: περιγράφει τον τρόπο με τον οποίο επιτυγχάνεται η αμφίδρομη αλληλεπίδραση μεταξύ συστήματος και χρήστη. Μπορεί να είναι κάποια γραφική διεπαφή, διεπαφή ομιλίας κλπ.
- *Διεπαφές με εξωτερικά συστήματα:* περιγράφουν την επικοινωνία του συστήματος με εξωτερικά συστήματα όπως Γεωγραφικά Πληροφοριακά Συστήματα (Geographic Information Systems, GIS), συστήματα εντοπισμού, εξυπηρέτες θέσης και συστήματα διαχείρισης περιεχομένου (Content Management Systems, CMS).
- *Μοντέλα αναπαράστασης χώρου (ή χωρικά μοντέλα)*: αποτελούν συμβολικές, γεωμετρικές ή υβριδικές αναπαραστάσεις του χώρου πλοήγησης. Τυπικά είναι κάποιος γράφος ή κάποιο μοντέλο αναπαράστασης GIS (αν η υπηρεσία πλοήγησης υλοποιείται από το ίδιο το GIS).
- *Μοντέλα χρηστών:* αποτελούν τα μοντέλα των χρηστών που περιγράφουν τις ικανότητες ή προτιμήσεις τους όσον αφορά την επιλογή μονοπατιών και οδηγιών. Η μεγάλη πλειοψηφία των συστημάτων δεν χρησιμοποιεί ξεχωριστά μοντέλα αλλά ενσωματώνει αυτή τη πληροφορία στους αλγορίθμους πλοήγησης (π.χ. αν το σύστημα απευθύνεται σε τυφλούς χρήστες, όλα τα μονοπάτια που θα υπολογίζονται θα είναι κατάλληλα για τυφλούς αλλά ίσως όχι για άλλες κατηγορίες χρηστών).
- *Αλγόριθμοι πλοήγησης και επιλογής μονοπατιού:* είναι οι αλγόριθμοι που αποφασίζουν ποιο μονοπάτι είναι κατάλληλο για τη πλοήγηση του χρήστη από τη τρέχουσα θέση του ως τον προορισμό του.
- *Αλγόριθμοι επιλογής και αναπαράστασης οδηγιών πλοήγησης:* οι αλγόριθμοι αυτοί ουσιαστικά προσπαθούν να βρουν την καταλληλότερη αναπαράσταση των οδηγιών πλοήγησης (και τα αντίστοιχα στοιχεία της διεπαφής χρήστη) με βάση το μοντέλο του κάθε χρήστη.



Εικόνα 1: Αρχιτεκτονική ενός συστήματος πλοήγησης αναφοράς

Από τα παραπάνω συστατικά, αυτά που αποτελούν κύριο αντικείμενο της παρούσας εργασίας είναι η υλοποίηση ενός μοντέλου *αναπαράστασης χώρου* και η ανάπτυξη της κατάλληλης διεπαφής για την επικοινωνία με το χρήστη.

Γενικά, έχουν αναπτυχθεί πολλά συστήματα, τα οποία εστιάζουν στη γεωμετρική απεικόνιση του χώρου. Αυτό συμβαίνει γιατί ο άνθρωπος θέλει να έχει την αίσθηση του χώρου με μετρικά συστήματα και ακριβείς τοποθεσίες. Όμως μια πολύ ενδιαφέρουσα όψη του κόσμου που θέλουμε να αναπαραστήσουμε είναι η σημασιολογική. Σ' αυτήν την όψη όλες οι συμβολικές τοποθεσίες και οι σχέσεις μεταξύ τους έχουν μοναδικά ονόματα για να ξεχωρίζουν μεταξύ των υπολοίπων οντοτήτων.

Η χρηστική αξία ενός συστήματος πλήγησης για εσωτερικούς χώρους μπορεί να αυξηθεί σημαντικά εάν συνδυάσει τη γεωμετρική με τη σημασιολογική απεικόνιση του χώρου. Στην περίπτωση αυτή η μοντελοποίηση των χώρων βασίζεται τόσο στη γεωμετρική/γεωγραφική αναπαράστασή τους όσο και στη σημασιολογική-συμβολική περιγραφή τους.

Μέσω της υβριδικής αυτής μοντελοποίησης τόσο τα μονοπάτια πλοήγησης (paths) όσο και οι οδηγίες που καθοδηγούν τους χρήστες πάνω σε αυτά μπορούν να παρασχεθούν σε αυτούς ανάλογα με τις φυσικές και αντιληπτικές (perceptual) τους ικανότητες, αλλά και τις ενδεχόμενες προτιμήσεις τους. Οι φυσικές ικανότητες των χρηστών αναφέρονται στην ικανότητά τους να περπατήσουν, να δουν, να ακούσουν κλπ. Ο συνυπολογισμός αυτών των ικανοτήτων κατά την διαδικασία της επιλογής ενός μονοπατιού είναι πολύ

σημαντικός για τα άτομα με αναπηρίες (ΑμεΑ) και είναι κάτι που λείπει από τα σημερινά συστήματα πλοήγησης. Οι φυσικές ικανότητες των χρηστών μπορεί να αποτελούν παράγοντες αποκλεισμού συγκεκριμένων μονοπατιών. Για παράδειγμα ένας τυφλός θα πρέπει να πλοηγείται από μέρη χωρίς πολυκοσμία, ενώ ένας ανάπηρος δεν είναι δυνατόν να ανεβοκατεβαίνει σκάλες. Οι αντιληπτικές ικανότητες των χρηστών έχουν σχέση με το πόσο εύκολα ή δύσκολα μπορεί να καθοδηγηθεί/πλοηγηθεί κάποιος μέσα σε ένα άγνωστο περιβάλλον. Αυτές οι ικανότητες εξαρτώνται κυρίως από την ηλικία του χρήστη και τη νοητική του κατάσταση. Για παράδειγμα ένα παιδί θα μπορούσε ευκολότερα να πλοηγηθεί σε ένα χώρο αν έβλεπε πολλά σημάδια (landmarks) της διαδρομής σε φωτογραφίες, ενώ ένας ενήλικας χωρίς νοητικά προβλήματα πιθανόν θα προτιμούσε να του δοθούν λίγες και σαφείς γραπτές οδηγίες στην αρχή της διαδικασίας πλοήγησης. Τέλος οι προτιμήσεις διαδρομής (routing preferences), που ενδεχομένως να έχει κάποιος χρήστης, συμπεριλαμβάνουν σημεία ενδιαφέροντος (Points of Interest, POIs) που θα ήθελε ο χρήστης να βρίσκονται στο μονοπάτι πλοήγησης και άλλες προτιμήσεις που εξαρτώνται από τις ιδιότητες του μονοπατιού. Για παράδειγμα, ένας γιατρός θα ήθελε πιθανότατα να χρησιμοποιήσει την υπηρεσία πλοήγησης σε ένα νοσοκομείο για να βρει το μονοπάτι προς ένα προορισμό, το οποίο περνά από τους περισσότερους ασθενείς που πρέπει, σύμφωνα με το πρόγραμμά του, να εξετάσει εκείνη την ώρα. Έτσι θα μπορούσε η υπηρεσία αυτή να συνδυαστεί με κάποιο χρονοπρογραμματιστή εργασιών (job scheduler), για να αυξηθεί η παραγωγικότητα ή η αποδοτικότητα των εργαζομένων. Από την άλλη πλευρά, κάποιος ηλικιωμένος χρήστης θα μπορούσε να δηλώσει ότι προτιμά τις διαδρομές που δεν έχουν σκάλες επειδή είναι πιο ξεκούραστες για αυτόν (παρόλο που ίσως να μην είναι πιο σύντομες).

ΚΕΦΑΛΑΙΟ 2
ΓΝΩΣΤΙΚΟ ΚΑΙ ΤΕΧΝΟΛΟΓΙΚΟ ΥΠΟΒΑΘΡΟ

2.1. Γενικά

Η παρούσα εργασία σχετίζεται με την υλοποίηση μιας εφαρμογής πλοήγησης σε ένα εσωτερικό χώρο (indoor navigation), οπότε σε αυτήν την ενότητα θα αναφερθούμε στις τεχνολογίες με τις οποίες μπορεί να εντοπισθεί η θέση κάποιας κινητής συσκευής (χρήστη) μέσα σε ένα τέτοιο χώρο.

2.2. Σημασιολογικός Ιστός (Semantic Web)

Η αξία και η σημασία του Παγκόσμιου Ιστού (World Wide Web ή WWW ή απλώς Web) στην σχέση του ανθρώπου με τον υπολογιστή είναι πάρα πολύ μεγάλη. Το WWW έδωσε τη δυνατότητα σε εκατομμύρια ανθρώπους να επικοινωνούν και να αλληλεπιδρούν, ανεξαρτήτως που βρίσκονται, τι εθνικότητας είναι και τι ενδιαφέροντα έχουν. Ένα πολύ μεγάλο μέρος της ανθρώπινης επικοινωνίας μέσω υπολογιστή καλύπτεται σήμερα από το Διαδίκτυο και τη βασικότερη εφαρμογή του, το WWW.

Παρόλα αυτά το WWW έχει ένα βασικό μειονέκτημα: βασίζεται στην συντακτική περιγραφή του περιεχομένου (content), κύριος στόχος της οποίας είναι η καταληπτότητά του από τον άνθρωπο. Έτσι οι ιστοσελίδες περιέχουν κείμενα ή άλλου είδους δημοσιευμένα περιεχόμενα τα οποία είναι κατανοητά μόνο από τους ανθρώπους. Το χαρακτηριστικό αυτό του WWW θεωρείται σαν ένα από τα μειονεκτήματά του, αφού δεν επιτρέπει την αυτόματη επεξεργασία περιεχομένου στο Διαδίκτυο. Τέτοια επεξεργασία προϋποθέτει μια καταλληλότερη περιγραφή του δημοσιευμένου περιεχομένου και αλγόριθμους που να προσδίδουν την επιθυμητή «ευφυΐα» στο υπολογιστικό περιβάλλον. Μια τέτοια περιγραφή θα πρέπει εκτός από το βασικό περιεχόμενο των ιστοσελίδων να περιλαμβάνει και δεδομένα που προορίζονται αποκλειστικά για τους υπολογιστές και που θα περιγράφουν την σημασιολογία (semantics) του περιεχομένου. Όσον αφορά στους αλγόριθμους, εφόσον υπάρχει η σωστή περιγραφή θα αναπτύσσονται ανάλογα με τη περίσταση και τη χρήση της πληροφορίας.

Το παραπάνω περιορισμό του WWW καθώς και τη λύση του οραματίστηκε πρώτος ο ιδρυτής του, Tim Berners-Lee[i], ο οποίος ονόμασε αυτό το εξελικτικό στάδιο του WWW σαν «Σημασιολογικό Ιστό» (Semantic Web). Ο Σημασιολογικός Ιστός έχει σαν στόχο να προσδώσει δομή στο νόημα του περιεχομένου των ιστοσελίδων, δημιουργώντας ένα περιβάλλον όπου οι πράκτορες λογισμικού (software agents) περιπλανώμενοι από

σελίδα σε σελίδα θα μπορούν να εκτελούν προηγμένες εργασίες για τους χρήστες. Ο Σημασιολογικός Ιστός δεν είναι ένας διαφορετικός ιστός αλλά μια επέκταση του υπάρχοντος Παγκόσμιου Ιστού, στην οποία η πληροφορία αποκτά καλά ορισμένο νόημα, δίνοντας τη δυνατότητα για πιο αποτελεσματική συνεργασία ανάμεσα στον άνθρωπο και στον υπολογιστή, αφού πλέον θα υπάρχει μια κοινή γλώσσα επικοινωνίας ανάμεσά τους (η σημασιολογική περιγραφή).

Γενικά ο Σημασιολογικός Ιστός αποτελεί τη βάση για μια πλήρως κατανεμημένη μορφή τεχνητής νοημοσύνης. Όπως είναι γνωστό, η τεχνητή νοημοσύνη ασχολείται κυρίως με δύο θέματα: την αναπαράσταση γνώσης (knowledge representation) και τις μεθόδους αναζήτησης και συμπερασμού (reasoning). Έτσι, για να λειτουργήσει ο Σημασιολογικός Ιστός θα πρέπει οι υπολογιστές να έχουν πρόσβαση σε δομημένες βάσεις γνώσης και σε κανόνες συμπερασμού τους οποίους να μπορούν να χρησιμοποιήσουν για να διενεργήσουν αυτόματη συλλογιστική.

Οι υπάρχουσες τεχνολογίες αναπαράστασης γνώσης έχουν σημειώσει κάποια πρόοδο αλλά δεν είναι ακόμα ικανές να υλοποιήσουν τις προσδοκίες της τεχνητής νοημοσύνης. Οι κλασικές μέθοδοι αναπαράστασης γνώσης προϋποθέτουν μια κοινή, παγκόσμια βάση γνώσης και κανόνων έτσι ώστε όλοι να χρησιμοποιούν τους ίδιους κανόνες. Από την άλλη πλευρά ο Σημασιολογικός Ιστός δεν επιβάλει την απόλυτη συμφωνία μεταξύ των διάφορων συστημάτων που θα διασυνδέει τηρώντας έτσι την αρχή του συμβατικού WWW: δεν είναι σίγουρο ότι μπορείς να βρεις κάτι που ψάχνεις, ακόμη κι αν υπάρχει. Με αυτό το τίμημα όμως παρέχει ευελιξία και εκφραστικότητα αφού δεν περιορίζει την διαδικασία της αναπαράστασης γνώσης. Έτσι μια πρόκληση για τον Σημασιολογικό Ιστό είναι να παρέχει μια γλώσσα που θα εκφράζει και δεδομένα και κανόνες συλλογιστικής και θα επιτρέπει την εξαγωγή στο WWW κάθε υπάρχοντος συστήματος αναπαράστασης γνώσης.

Η μεγαλύτερη συμβολή της πρωτοβουλίας του Σημασιολογικού Ιστού μέχρι σήμερα είναι η προτυποποίηση που παρείχε σε γλώσσες και τεχνολογίες. Πιο συγκεκριμένα σήμερα η πιο διαδεδομένη γλώσσα για δημιουργία οντολογιών είναι η Web Ontology Language (OWL)[ii]. Το συντακτικό της γλώσσας αυτής βασίζεται στην XML (eXtensible Markup Language)[iii] και στην RDF/RDF Schema (Resource Description Framework)[iv]. Η εκφραστικότητα και η σημασιολογία της OWL καθορίζεται κυρίως από τις Περιγραφικές Λογικές (Description Logics)[v].

2.3. Υπηρεσίες Θέσης

Οι Υπηρεσίες Θέσης (Location Based Services) μπορούν να περιγραφούν σαν «εφαρμογές» που εκτελούνται και παρέχουν κάποια λειτουργικότητα (εξ' ου και ο όρος «υπηρεσίες») στον κινητό χρήστη, πάντα λαμβάνοντας υπόψη τη γεωγραφική θέση του. Η εκτέλεσή τους μπορεί να ξεκινήσει είτε μετά από συγκεκριμένες αλλαγές της θέσης του χρήστη είτε μετά από ρητή αίτησή του.

Σήμερα έχουν υλοποιηθεί αρκετές υπηρεσίες θέσης και έχουν προταθεί ακόμα περισσότερες. Οι κυριότερες είναι:

- Διαφημίσεις (Advertising)
- Ιχνηλασία (Tracking)
- Πλοήγηση (Navigation)
- Εύρεση (Πλησιέστερων) Σημείων Ενδιαφέροντος (Points of Interest)
- Υπηρεσίες Έκτακτης Ανάγκης (Emergency Services)
- Διασκέδαση (Entertainment)
- Χρέωση και Διαχείριση (Billing and Management)

Στα πλαίσια της παρούσας πτυχιακής εργασίας θα μας απασχολήσει μόνο η πλοήγηση. Η πλοήγηση γίνεται συνήθως ανάμεσα σε δύο σημεία (την αφετηρία και τον προορισμό) και έχει σαν στόχο την εύρεση και χρήση της καλύτερης διαδρομής ανάμεσα σε αυτά τα σημεία. Ο όρος «καλύτερη διαδρομή» από μόνος του είναι κάπως ασαφής. Γενικά, ο ορισμός της ποιότητας της διαδρομής/μονοπατιού εξαρτάται από την εκάστοτε εφαρμογή. Για παράδειγμα, για μια εφαρμογή πλοήγησης η καλύτερη διαδρομή μπορεί να είναι η συντομότερη ενώ για κάποιο άλλο η πιο γρήγορη (αν θεωρήσουμε ότι το μήκος ενός μονοπατιού δεν είναι ανάλογο της ταχύτητάς διέλευσής του).

Ένα σύγχρονο σύστημα πλοήγησης περιλαμβάνει κατ' ελάχιστον τις ακόλουθες οντότητες:

- *Μηχανισμός εντοπισμού του χρήστη*. Ο μηχανισμός αυτός μπορεί να βασίζεται σε διάφορες τεχνολογίες: υπέρυθρους και υπερηχητικούς φάρους (infrared and ultrasound beacons), RFID tags, τριγωνοποίηση λαμβανόμενων ισχύων σήματος (RSS fingerprinting and triangulation) από σταθμούς βάσης ασύρματων τοπικών δικτύων, κλπ. Σε κάθε περίπτωση όμως, η διαδικασία πλοήγησης απαιτεί μεγάλη ακρίβεια εντοπισμού των χρηστών.

- *Ασύρματο δίκτυο*. Η δικτύωση είναι πλέον απαραίτητο συστατικό κάθε σύγχρονης υπολογιστικής εφαρμογής. Στην περίπτωση της πλοήγησης χρησιμοποιείται κυρίως για να μεταφέρει στον χρήστη τις οδηγίες πλοήγησης. Τα ασύρματα δίκτυα για εσωτερικούς χώρους βασίζονται στο πρότυπο IEEE 802.11. Ο εξοπλισμός που είναι συμβατός με αυτό το πρότυπο (σταθμοί βάσης, κάρτες δικτύου, κα.) είναι πλέον αρκετά φτηνός και υψηλής ποιότητας. Επίσης ένα ασύρματο δίκτυο IEEE 802.11 μπορεί να χρησιμοποιηθεί και για τον εντοπισμό του χρήστη που περιγράφηκε στην προηγούμενη παράγραφο.
- *Χωρική βάση δεδομένων (spatial database)*. Η χωρική βάση δεδομένων χρησιμοποιείται για την αποθήκευση της γεωμετρίας του χώρου στον οποίο θέλουμε να υποστηρίξουμε πλοήγηση. Εκτός από τις κατάλληλες δομές αναπαράστασης γεωμετρίας, παρέχει και ειδικούς τελεστές που μπορούν να προσδώσουν πολλή μεγάλη εκφραστικότητα στις διάφορες επερωτήσεις που μπορούν να τεθούν στη βάση. Παραδείγματα χωρικών βάσεων δεδομένων είναι η Oracle Spatial[vi] και η PostGIS[vii] (μια spatial επέκταση της PostgreSQL[viii]).
- *Συσκευή παρουσίασης/αναπαραγωγής οδηγιών πλοήγησης*. Κάθε χρήστης θα πρέπει να χρησιμοποιεί μια συσκευή για τις αιτήσεις του προς την υπηρεσία πλοήγησης αλλά και για τον εντοπισμό του. Η συσκευή αυτή μπορεί να είναι φορητή ή όχι. Μια φορητή συσκευή (π.χ. Personal Digital Assistant – PDA, smartphone), είναι προτιμότερη καθώς μπορεί να κατευθύνει με μεγαλύτερη ευελιξία τον χρήστη στον προορισμό του (π.χ. να τον οδηγεί βήμα-βήμα, να εντοπίζει και να του επισημαίνει τυχόν παρεκκλίσεις του από το μονοπάτι κλπ.). Η συσκευή αύτή θα πρέπει να είναι ικανή να αναπαραστήσει (display) τις διάφορες οδηγίες πλοήγησης μέσω κατάλληλων διεπαφών (εικόνα, ήχος, κείμενο).

2.4. GEOGRAPHIC INFORMATION SYSTEMS (GIS)

Τα συστήματα χαρτογράφησης στην αγορά σήμερα ποικίλουν από απλά display-on συστήματα όπως οι ηλεκτρονικοί άτλαντες ως τα πλήρη συστήματα γεωγραφικών πληροφοριών (GIS). Τα συστήματα βέβαια διαφέρουν σημαντικά σε έναν αριθμό θεμάτων: το πώς συνδέουν τη γεωγραφική θέση με την πληροφορία για τη θέση αυτή, την ακρίβεια με την οποία καθορίζουν γεωγραφικές τοποθεσίες, το επίπεδο ανάλυσης που εφαρμόζουν, τον τρόπο με τον οποίο παρουσιάζουν την πληροφορία σε γραφικά σχέδια κλπ.

Οι ηλεκτρονικοί άτλαντες, για παράδειγμα, υποστηρίζουν την παρουσίαση εικόνων γεωγραφικών περιοχών στην οθόνη του υπολογιστή. Παρέχουν περιορισμένες πληροφορίες για τις περιοχές και μικρή δυνατότητα αλλαγής των γραφικών. Χωρίς εργαλεία για την ανάλυση της πληροφορίας, αυτά τα συστήματα είναι περισσότερο χρήσιμα για την παροχή γραφικών που μπορούν να χρησιμοποιηθούν σε αναφορές ή παρουσιάσεις.

Το GIS έχει τη δυνατότητα της παρουσίασης εικόνων γεωγραφικών περιοχών χρησιμοποιώντας πληροφορίες αποθηκευμένες σε μια βάση δεδομένων συνδέοντας τη γεωγραφική θέση με τις πληροφορίες αυτές. Επιπλέον παρέχει αφενός τη δυνατότητα ανάλυσης των γεωγραφικών θέσεων και της διασυνδεδεμένης πληροφορίας και αφετέρου πρόσβαση από την πληροφορία στο χάρτη και από το χάρτη στην πληροφορία. Πολλές από τις λειτουργίες, υπηρεσίες και εφαρμογές ενός GIS παρέχονται από τα συστήματα υπηρεσιών θέσης.

ΟΡΙΣΜΟΣ

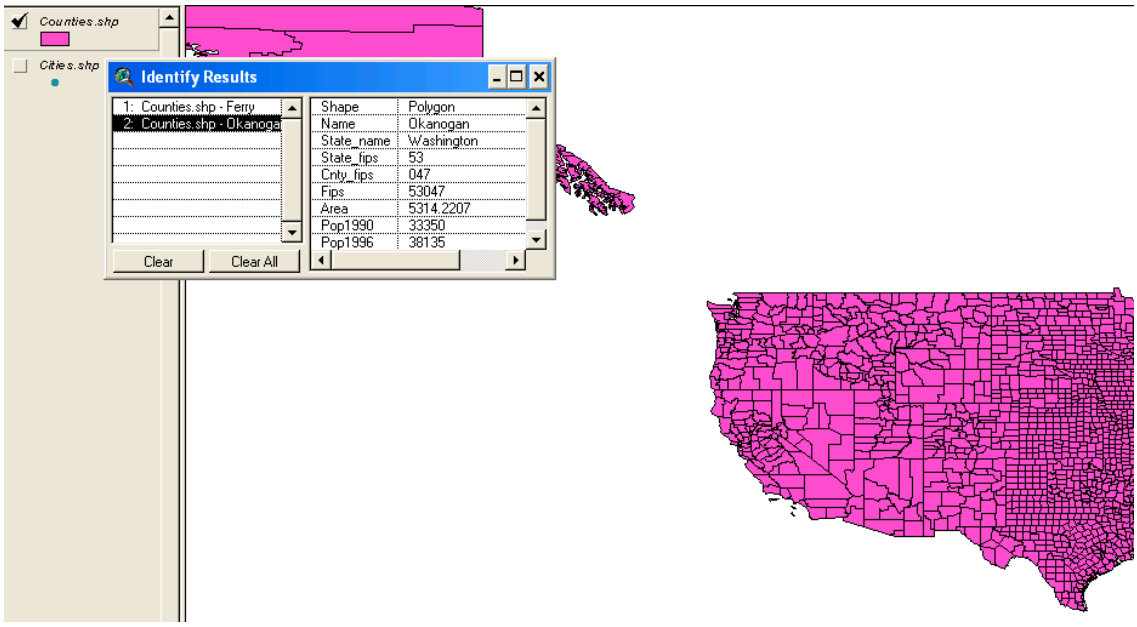
Σύμφωνα με τους Huxhold & Levinsohn (1995), «ένα σύστημα γεωγραφικών πληροφοριών (GIS) είναι μια συλλογή τεχνολογικών πληροφοριών, δεδομένων, και διαδικασιών για συλλογή, αποθήκευση, διαχείριση, ανάλυση και παρουσίαση χαρτών και περιγραφικών πληροφοριών για στοιχεία που μπορούν να παρασταθούν στους χάρτες.» Πιο απλά το GIS είναι μια τεχνολογία διαχείρισης και ανάλυσης γεωγραφικής γνώσης.

Η τεχνολογία που απαιτείται για να κατασκευαστεί ένα GIS δεν είναι απλώς ένα κομμάτι λογισμικού, είναι ένας συνδυασμός πληροφοριακών τεχνολογιών που περιλαμβάνουν: GIS, συστήματα διαχείρισης βάσεων δεδομένων, GPS - για εξωτερικούς χώρους- ή άλλες τεχνικές εντοπισμού για εσωτερικούς χώρους.

Τα GIS αναπτύχθηκαν λόγω της αντικειμενικής δυσκολίας που υπάρχει στην μετάφραση και διαχείριση δεδομένων αποθηκευμένων μόνο σε μια βάση δεδομένων. Έτσι, η οπτικοποίηση δεδομένων σε χάρτες είναι ο καλύτερος τρόπος να κατανοηθούν, επερωτηθούν ή να μεταβληθούν τέτοια δεδομένα, ειδικά δεδομένα που σχετίζονται με τη γεωγραφία του χώρου. Παραδείγματα χρήσης GIS βρίσκονται σε πολλά διαφορετικά πεδία όπως: καταγραφή περιβαλλοντολογικών αλλαγών, μοντελοποίηση, βιομηχανικές εφαρμογές (π.χ. οπτικοποίηση υπόγειων σωλήνων. Ουσιαστικά το GIS μπορεί να χρησιμοποιηθεί οπουδήποτε απαιτείται ανάλυση χωρικών δεδομένων.

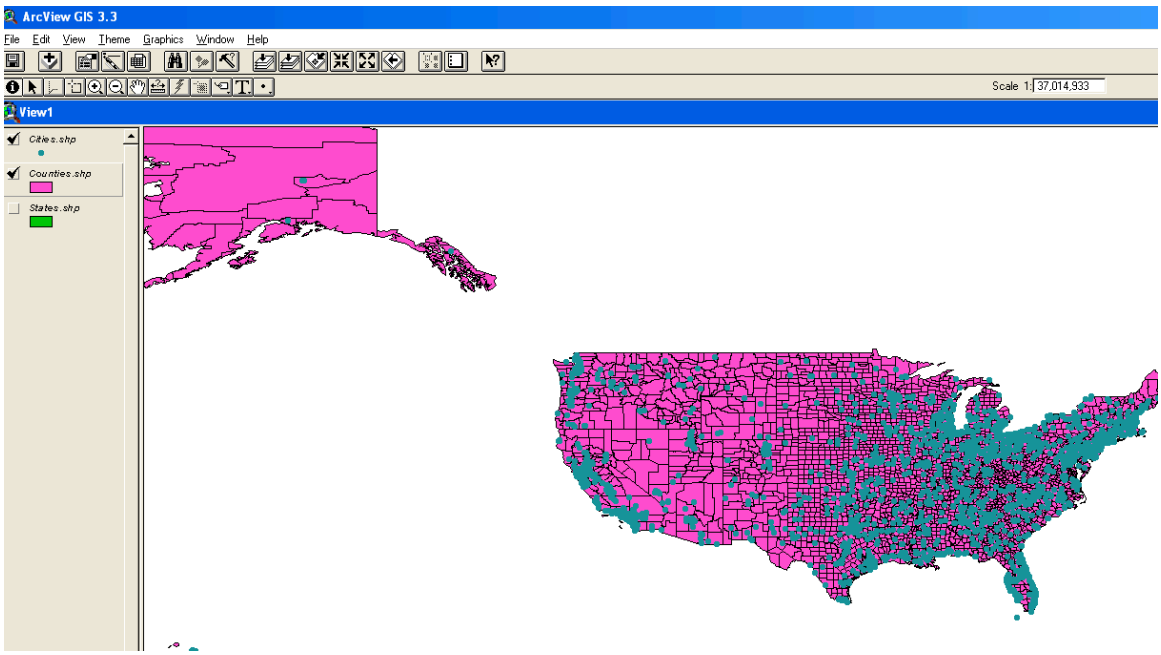
Το GIS συνδέει τη θέση με την πληροφορία (π.χ. ανθρώπους με διευθύνσεις, κτίρια με οικοδομικά τετράγωνα, ή τους δρόμους σε ένα δίκτυο) και τοποθετεί την πληροφορία σε θεματικά επίπεδα (layers) ώστε να παρέχει μια καλύτερη εικόνα του πως όλα διασυνδέονται. Αν και συχνά στο GIS δουλεύουμε με χάρτες, ο χάρτης είναι ο ένας μόνο από τους τρεις τρόπους εργασίας με γεωγραφική πληροφορία στο GIS.

1. Η όψη της βάσης δεδομένων: Ένα GIS είναι ένα μοναδικό είδος βάσης δεδομένων του κόσμου – μια γεωγραφική βάση δεδομένων (geodatabase). Είναι ένα «Πληροφοριακό σύστημα για τη γεωγραφία». Πρωταρχικά, το GIS βασίζεται σε μια δομημένη βάση δεδομένων που περιγράφει τον κόσμο με γεωγραφικούς όρους. Στην **Εικόνα 2** φαίνεται ένα σύστημα γεωγραφικών πληροφοριών από την οπτική της βάσης δεδομένων.



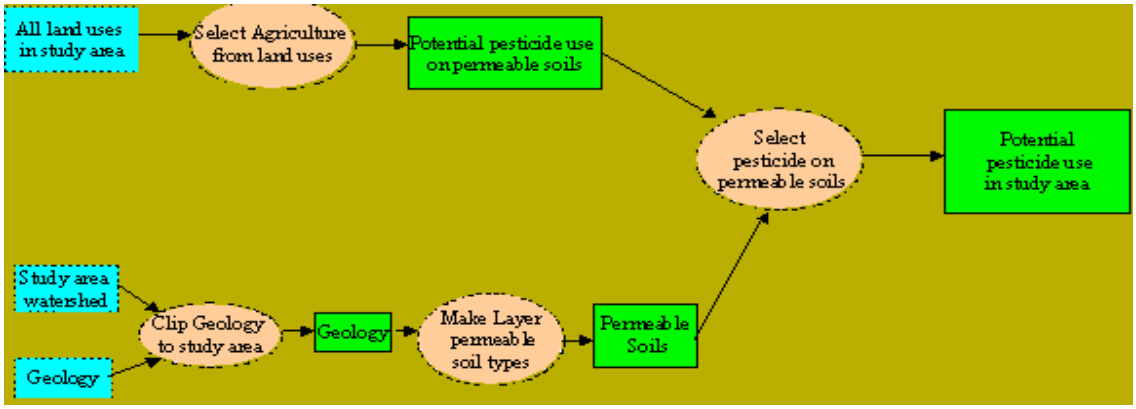
Εικόνα 2: Η όψη της βάσης δεδομένων ενός GIS

2. Η όψη του χάρτη (βλ. **Εικόνα 3**): Ένα GIS είναι ένα σύνολο έξυπνων χαρτών και άλλων όψεων που παρουσιάζουν στοιχεία και σχέσεις μεταξύ στοιχείων στην επιφάνεια της γης. Ακόμη, χάρτες της υποβόσκουσας γεωγραφικής πληροφορίας μπορούν να κατασκευαστούν και να χρησιμοποιηθούν σαν παράθυρα στη βάση δεδομένων ώστε να υποστηρίξουν επερωτήσεις, ανάλυση και τροποποίηση της πληροφορίας. Αυτό καλείται geovisualization.



Εικόνα 3: Η όψη χάρτη του GIS

3. Η όψη του μοντέλου (βλ. **Εικόνα 4**): Ένα GIS είναι ένα σύνολο εργαλείων τροποποίησης της πληροφορίας τα οποία ανακτούν νέα σύνολα γεωγραφικών δεδομένων από τα ήδη υπάρχοντα. Αυτές οι λειτουργίες επεξεργασίας γεωγραφικών δεδομένων (geoprocessing functions) παίρνουν πληροφορίες από υπάρχοντα σύνολα δεδομένων, εφαρμόζουν αναλυτικές διαδικασίες και γράφουν τα αποτελέσματα σε νέα σύνολα δεδομένων.



Εικόνα 4: Η όψη μοντέλου GIS

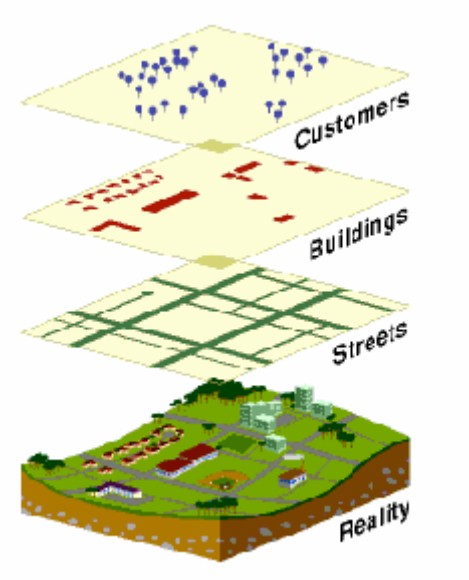
Τέλος, το GIS διευκολύνει και συνεπώς επιταχύνει τη λήψη αποφάσεων χάρη στους εσωτερικούς του μηχανισμούς επερωτήσεων και παρουσίασης των αποτελεσμάτων (charts, maps). Για παράδειγμα, το GIS μπορεί να χρησιμοποιηθεί ώστε να πάρουμε μια απόφαση για την τοποθεσία κατασκευής ενός αριθμού κατοικιών. Θέλουμε οι κατοικίες να βρίσκονται αρκετά κοντά σε κάποια μεγάλη πόλη, αλλά σε περιοχή με την ελάχιστη δυνατή μόλυνση περιβάλλοντος. Η πληροφορία αυτή μπορεί να παρασταθεί

γρήγορα και ξεκάθαρα με τη μορφή ενός χάρτη. Έτσι, το χρόνο που θα σπαταλούσαμε προσπαθώντας να κατανοήσουμε τα δεδομένα τον αξιοποιούμε συγκεντρωμένοι στα πραγματικά προβλήματα. Ακόμη, αφού τα αποτελέσματα στο GIS μπορούν να παραχθούν ταχύτατα, εναλλακτικά σενάρια μπορούν να αξιολογηθούν με ταχύτητα και συνέπεια.

2.5. SPATIAL DATA (ΧΩΡΙΚΑ ΔΕΔΟΜΕΝΑ)

Σε αντίθεση με τις παραδοσιακές εφαρμογές βάσεων δεδομένων, οι GIS εφαρμογές απαιτούν τόσο χωρικά όσο και μη χωρικά αντικείμενα σα δεδομένα. Ένα χωρικό αντικείμενο έχει διαστάσεις όπως μήκος (για γραμμές) ή εμβαδό (για επιφάνειες). Ακόμη, τα χωρικά δεδομένα έχουν συγκεκριμένη θέση όσον αφορά κάποιο κεντρικό σύστημα συντεταγμένων. Αυτό το χαρακτηριστικό της θέσης των χωρικών αντικειμένων εκκινεί μια σειρά ερωτημάτων και έρευνας για τα συστήματα συντεταγμένων και τη σχέση των χωρικών αντικειμένων με αυτά.

Τα χωρικά δεδομένα είναι πληροφορία για το σχήμα, τη θέση και τη σχέση μεταξύ γεωγραφικών στοιχείων. Παρουσιάζονται σε πολλές μορφές. Από ψηφιακά αρχεία μέχρι παραδοσιακούς χάρτες. Ένα GIS απαιτεί οι κλασικοί χάρτες να σκαναριστούν ή να ψηφιοποιηθούν ώστε να τους έχουμε τελικά σε κάποια ψηφιακή μορφή. Η ανάκτηση, διαχείριση και συντήρηση των δεδομένων είναι από τις πιο ακριβές εργασίες σε ένα GIS. Ένα παράδειγμα βρίσκεται στην **Εικόνα 5** παρακάτω. Τα χωρικά ή γεωγραφικά δεδομένα μπορούν να διαχωριστούν σε στοιχεία του χώρου (σημεία, γραμμές, πολύγωνα) κάθε ένα από τα οποία εμφανίζεται σε ξεχωριστό ομογενές επίπεδο (layer).



Εικόνα 5: Απεικόνιση Χαρτών σε Επίπεδα

2.5.1. Τύποι Αποθήκευσης Χωρικών Δεδομένων

Υπάρχουν τρεις τύποι απεικόνισης δεδομένων (data mapping) και μορφοποίησης GIS δεδομένων. Οι τύποι αυτοί μαζί με κάποια παραδείγματα υλοποίησης τους φαίνονται παρακάτω:

- βασισμένοι σε αρχεία (file-based) – Shapefiles, GeoTIFF images, Microstation Design Files (DGN)
- βασισμένοι σε καταλόγους (directory-based): ESRI ArcInfo Coverages, US Census Tiger
- με βάσεις δεδομένων (DB-based): PostGIS, ESRI ArcSDE, MySQL

Κάθε τύπος δεδομένων αποτελείται από την πηγή δεδομένων (data source) και από ένα ή περισσότερα επίπεδα (layers). Για την πηγή δεδομένων και το layer δίδονται οι εξής ορισμοί:

Πηγή Δεδομένων (Data Source): Μία ομάδα layers που είναι αποθηκευμένη σε ένα κοινό χώρο. Ο χώρος αυτός μπορεί να είναι ένα αρχείο το οποίο περιλαμβάνει τα layers ή ένα φάκελος (folder) που έχει ένα σύνολο αρχεία που χαρακτηρίζουν τα layers.

Επίπεδο (Layer): Ένα υποσύνολο του data source που συχνά περιέχει πληροφορία ενός τύπου διανυσματικής μορφής (σημείο, γραμμή, πολύγωνο).

2.5.1.1 File-based Δεδομένα

Τα File-based δεδομένα αποτελούνται από ένα ή περισσότερα αρχεία που είναι αποθηκευμένα σε κάποιο φάκελο (folder). Σε πολλές περιπτώσεις χρησιμοποιείται ένα μόνο αρχείο (π.χ. DGN), αλλά δεν αποκλείεται να χρησιμοποιούνται και περισσότερα. Για παράδειγμα στην περίπτωση των ESRI Shapefiles χρησιμοποιούνται τρία διαφορετικά αρχεία (με επεκτάσεις SHP, DBF, SHX). Η ύπαρξη και των τριών αρχείων είναι απαραίτητη καθώς το καθένα επιτελεί διαφορετικό ρόλο.

Τα ονόματα αρχείων συνήθως αποτελούν και το όνομα των πηγών δεδομένων. Τα αρχεία αυτά περιλαμβάνουν layers τα οποία ενδέχεται να είναι προφανή από το όνομα του αρχείου χωρίς αυτό να είναι απαραίτητο. Για παράδειγμα στα Shapefiles, υπάρχει ένα data source για κάθε shapefile και ένα layer που έχει το ίδιο όνομα με αυτό του αρχείου shapefile. Στην παρούσα εργασία για την υλοποίηση της εφαρμογής χρησιμοποιήθηκαν ανυσματιικά δεδομένα τύπου Shapefiles.

Το shapefile είναι μορφή χωρικών δεδομένων της ESRI. Αποθηκεύει μη τοπολογική γεωμετρία και πληροφορίες χωρικών γνωρισμάτων σε ένα σύνολο

δεδομένων. Υποστηρίζουν σα στοιχεία γραμμές, σημεία και περιοχές. Η έννοια του shapefile περιλαμβάνει τρεις τουλάχιστον τύπους αρχείων οι οποίοι φαίνονται στον **Πίνακας 1** και έχουν καθορισμένες επεκτάσεις. Αυτά τα αρχεία πρέπει να αποθηκεύονται στον ίδιο χώρο εργασίας (workspace).

.shp	Αποθηκεύει τη γεωμετρία των στοιχείων
.shx	Ευρετήριο για τη γεωμετρία των στοιχείων
.dbf	Το αρχείο βάσης δεδομένων για τις πληροφορίες των γνωρισμάτων των στοιχείων

Πίνακας 1: Επεκτάσεις Shapefile Αρχείων και η Περιγραφή Λειτουργίας τους

Το βασικό αρχείο (.shp) περιέχει μια επικεφαλίδα αρχείου συγκεκριμένου μήκους η οποία ακολουθείται από μεταβλητού μήκους εγγραφές. Κάθε μεταβλητού μήκους εγγραφή αποτελείται από συγκεκριμένου μήκους εγγραφή επικεφαλίδας η οποία ακολουθείται από μεταβλητού μήκους περιεχόμενα εγγραφών.

2.5.1.2 Directory-based Δεδομένα

Τα directory-based δεδομένα αποτελούνται από ένα ή περισσότερα αρχεία τα οποία είναι αποθηκευμένα με συγκεκριμένο τρόπο μέσα σε ένα φάκελο γονέα. Σε κάποιες περιπτώσεις (π.χ. Coverages) μπορεί να είναι επίσης απαραίτητη η ύπαρξη επιπλέον φακέλων σε άλλα σημεία του δένδρου αρχείων (file tree) προκειμένου να είναι δυνατή η πρόσβαση στα directory-based δεδομένα. Το ίδιο το directory ενδέχεται να είναι το data source. Διαφορετικά αρχεία μέσα στο directory συχνά αντιπροσωπεύουν τα διαθέσιμα layers δεδομένων.

Για παράδειγμα τα ESRI ArcInfo Coverages αποτελούνται από περισσότερα του ενός αρχεία με επέκταση αρχείου ADF μέσα σε ένα φάκελο. Το αρχείο «PAL.ADF» αντιπροσωπεύει δεδομένα τύπου πολύγωνο. Το «ARC.ADF» περιλαμβάνει δεδομένα τύπου τόξου (arc) ή γραμμής. Ο φάκελος αποτελεί το data source και κάθε αρχείο ADF είναι ένα layer.

2.5.1.3 DB-based Δεδομένα

Τα δεδομένα αυτά είναι παρόμοια με τις δομές file-based και directory-based, που αναφέρονται παραπάνω, υπό την έννοια ότι παρέχουν δεδομένα γεωγραφικών

συντεταγμένων στον server που δημιουργεί τους χάρτες, ώστε να χρησιμοποιηθούν για τη δημιουργία των διανυσματικών συνόλων δεδομένων.

Οι database connections παρέχουν μια ροή δεδομένων γεωγραφικών συντεταγμένων η οποία αποθηκεύεται προσωρινά (π.χ. στη μνήμη) και στη συνέχεια διαβάζεται από τον Mapserver προκειμένου να δημιουργηθεί ο χάρτης. Είναι δυνατόν να απαιτούνται και επιπρόσθετα δεδομένα (π.χ. attribute) αλλά ο πυρήνας είναι τα δεδομένα γεωγραφικών συντεταγμένων.

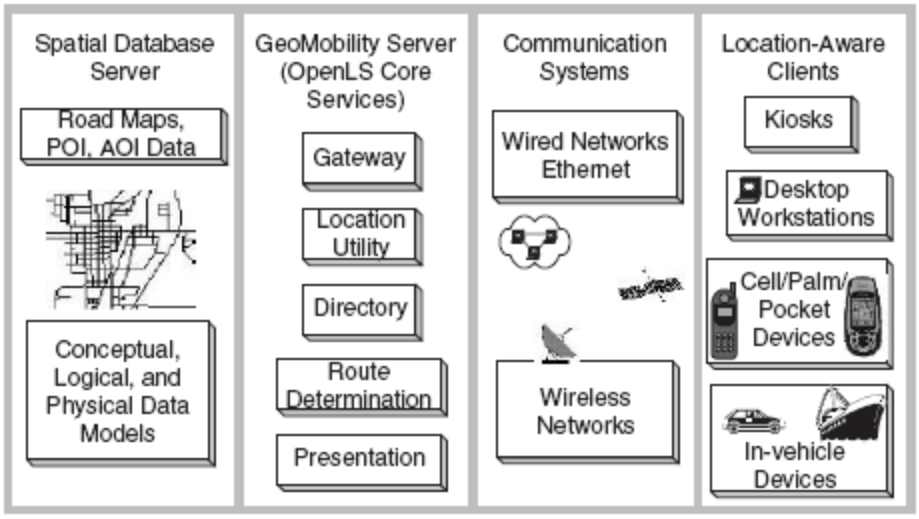
Θα πρέπει εδώ να επισημανθεί μια σημαντική διάκριση μεταξύ των βάσεων δεδομένων. Οι βάσεις δεδομένων στις οποίες αναφερόμαστε εδώ είναι οι χωρικές (spatial) οι οποίες περιλαμβάνουν γεωγραφικά δεδομένα σε μια συγκεκριμένη δική τους δομή. Αυτό διαφέρει από τις βάσεις δεδομένων που είναι αυστηρά με τη μορφή πινάκων (tabular databases) οι οποίες δεν μπορούν να περιλαμβάνουν γεωγραφικές συντεταγμένες με τον ίδιο τρόπο. Κάποια πολύ απλά γεωγραφικά δεδομένα είναι δυνατόν να περιληφθούν σε απλές βάσεις δεδομένων αλλά για οτιδήποτε πιο σύνθετο απαιτείται η χρήση χωρικών βάσεων δεδομένων. Υπάρχουν πολλές επεκτάσεις (extensions) βάσεων δεδομένων για απεικόνιση χωρικών δεδομένων, τόσο εμπορικού τύπου όσο και ανοικτού κώδικα. Από τις πλέον ισχυρές επεκτάσεις είναι η PostGIS η οποία παρέχει δυνατότητα αποθήκευσης χωρικών δεδομένων στη βάση δεδομένων PostgreSQL. Με τη χρήση της PostGIS μπορούμε όχι μόνο να αποθηκεύσουμε χωρική πληροφορία αλλά και να χειριστούμε τα χωρικά δεδομένα με SQL εντολές. Μια άλλη βάση δεδομένων με χωρικές δυνατότητες είναι η MySQL.

2.5.2. Spatial Databases and GIS

Όπως αναφέρθηκε και στις προηγούμενες ενότητες τα συστήματα υπηρεσιών θέσης παρέχουν τη δυνατότητα εύρεσης της γεωγραφικής θέσης μιας κινητής συσκευής και τότε παρέχουν υπηρεσίες βασισμένες σ' αυτή την θέση (OpenLS)[ix]. Αυτές οι βασισμένες στη θέση υπηρεσίες απαιτούν ένα Spatial Database (SDB) server, ο οποίος παρέχει την αποτελεσματική και αποδοτική ανάκτηση και διαχείριση των geospatial δεδομένων. Τα χωρικά συστήματα βάσεων δεδομένων παραδίδουν διάφορα χωρικά δεδομένα (π.χ. ψηφιακοί οδικοί χάρτες) και μη χωρικές πληροφορίες (π.χ., οδηγίες καθοδήγησης διαδρομών) στον πελάτη μετά από αίτησή του. Οι SDB servers παρέχουν αποδοτικές geospatial ικανότητες επεξεργασίας ερωτημάτων όπως η εύρεση του κοντινότερου σε μια συγκεκριμένη θέση βενζινάδικου και εύρεσης της κοντινότερης πορείας στον προορισμό. Το SDB σύστημα ενεργεί ως back-end server στον GeoMobility server.

Κατά συνέπεια οι SDB servers διαδραματίζουν έναν κρίσιμο ρόλο στην υλοποίηση αποδοτικών και περίπλοκων εφαρμογών συστημάτων πλοήγησης.

Η γενική αρχιτεκτονική ενός σύγχρονου συστήματος πλοήγησης παρουσιάζεται στην **Εικόνα 6**. Τα τμήματα μπορούν να ταξινομηθούν ευρέως σε τέσσερα υποσυστήματα: στον SDB server, στον GeoMobility server, στα συστήματα επικοινωνίας, και στους location-aware clients.



Εικόνα 6: Αρχιτεκτονική ενός σύγχρονου συστήματος πλοήγησης

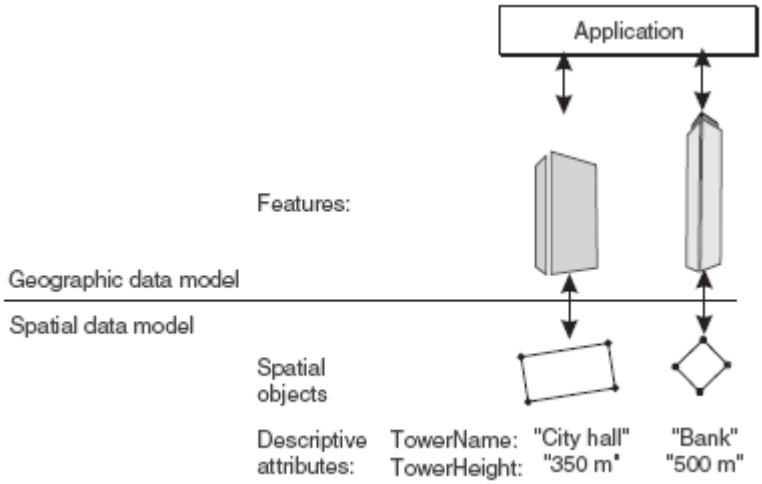
2.5.2.1 Σύστημα Διαχείρισης Χωρικών Βάσεων Δεδομένων

Γενικά, μια βάση δεδομένων δημιουργείται, οργανώνεται, και διατηρείται από ένα *Σύστημα Διαχείρισης Βάσεων Δεδομένων (Database Management System - DBMS*), που είναι ένα λογισμικό που εκτελείται σ' έναν server για την πραγματοποίηση τέτοιων εργασιών. Σε αυτή τη λογική, μια χωρική DBMS εστιάζει στην αποδοτική αποθήκευση και τη βελτιστοποίηση των χωρικών δεδομένων.

Μια χωρική DBMS αποτελεί ένα αναπόσπαστο τμήμα ενός GIS, που χρησιμοποιείται για τη διατήρηση των χωρικών δεδομένων, ενώ ένα GIS προσφέρει αρκετά διαφορετικές λειτουργίες από μια χωρική DBMS. Στην αγορά, υπάρχουν αρκετά προϊόντα GIS διαθέσιμα, όπως το ArcInfo και το ArcView (και τα δύο από την ESRI). Κάποια από αυτά έχουν ενσωματωμένη μια χωρική DBMS, η οποία συχνά μπορεί να αντικατασταθεί από άλλα συστήματα ενώ υπάρχουν περιπτώσεις όπου οι χωρικές βάσεις είναι διαθέσιμες ως επεκτάσεις των σχεσιακών DBMS (για παράδειγμα η postgis στην postgreSQL).

Για την κατανόηση των GIS, είναι απαραίτητο να γίνει διάκριση μεταξύ δύο αφαιρετικών επιπέδων, τα οποία απεικονίζονται στην **Εικόνα 7**. Το ανώτερο στρώμα σε ένα GIS, το

αποκαλούμενο *γεωγραφικό μοντέλο δεδομένων*, παρέχει μια εννοιολογική άποψη του γεωγραφικού περιεχομένου από την άποψη μονάδων, οι οποίες αποκαλούνται *features*. Ένα feature αντιπροσωπεύει μια πραγματική οντότητα, για παράδειγμα, ένα κτήριο, ένα δρόμο, ένα ποτάμι, μια χώρα, ή μια πόλη. Αποτελείται από ένα *χωρικό στοιχείο*, το οποίο καθορίζει τη θέση του, τη μορφή του και την τοπολογική σχέση του με άλλες οντότητες, και μια *περιγραφή*, η οποία παρέχει μη χωρικές πληροφορίες για την οντότητα, για παράδειγμα, το όνομα μιας πόλης ή ενός δρόμου, ή του πληθυσμού μιας χώρας. Κάθε feature έχει ένα καθορισμένο με σαφήνεια σύνολο διαδικασιών, το οποίο προσαρμόζεται στον τύπο που η πραγματική οντότητα αντιπροσωπεύει. Παραδείγματος χάριν, ένα feature που αντιπροσωπεύει ένα δρόμο μπορεί να παρέχει μια λειτουργία για να ζητήσει το μήκος του, ενώ ένα feature για μια πόλη μπορεί να προσφέρει μια λειτουργία για τη λήψη του μεγέθους της περιοχής. Οι διαδικασίες χρησιμοποιούνται από τις εφαρμογές για το χειρισμό ή την αναζήτηση της περιγραφικής πληροφορίας ενός feature και της συσχέτισης του με άλλα. Κατά συνέπεια, το γεωγραφικό μοντέλο δεδομένων αντιπροσωπεύει τη διεπαφή μεταξύ του GIS και της εφαρμογής.



Εικόνα 7: Επίπεδα GIS

Το χαμηλότερο layer εξετάζει όλες τις πτυχές της φυσικής διαχείρισης δεδομένων, συμπεριλαμβανομένης της αποθήκευσης, της επεξεργασίας του αιτήματος, και της βελτιστοποίησης, καθώς επίσης και της αποκατάστασης. Ένα ιδιαίτερο ζήτημα όσον αφορά τα GIS είναι η αναπαράσταση των χωρικών τμημάτων των features, τα οποία καλούνται *χωρικά αντικείμενα (spatial objects)*, καθώς επίσης και ο συνδυασμός τους με περιγραφικές ιδιότητες (*descriptive attributes)* για τη διατήρηση των περιγραφών τους. Τα χωρικά αντικείμενα μπορούν να αναπαρασταθούν με πολλούς τρόπους. Γενικά, οι τρόποι αναπαράστασης είναι δύο: *raster mode* και *vector mode*. Αυτοί οι τρόποι

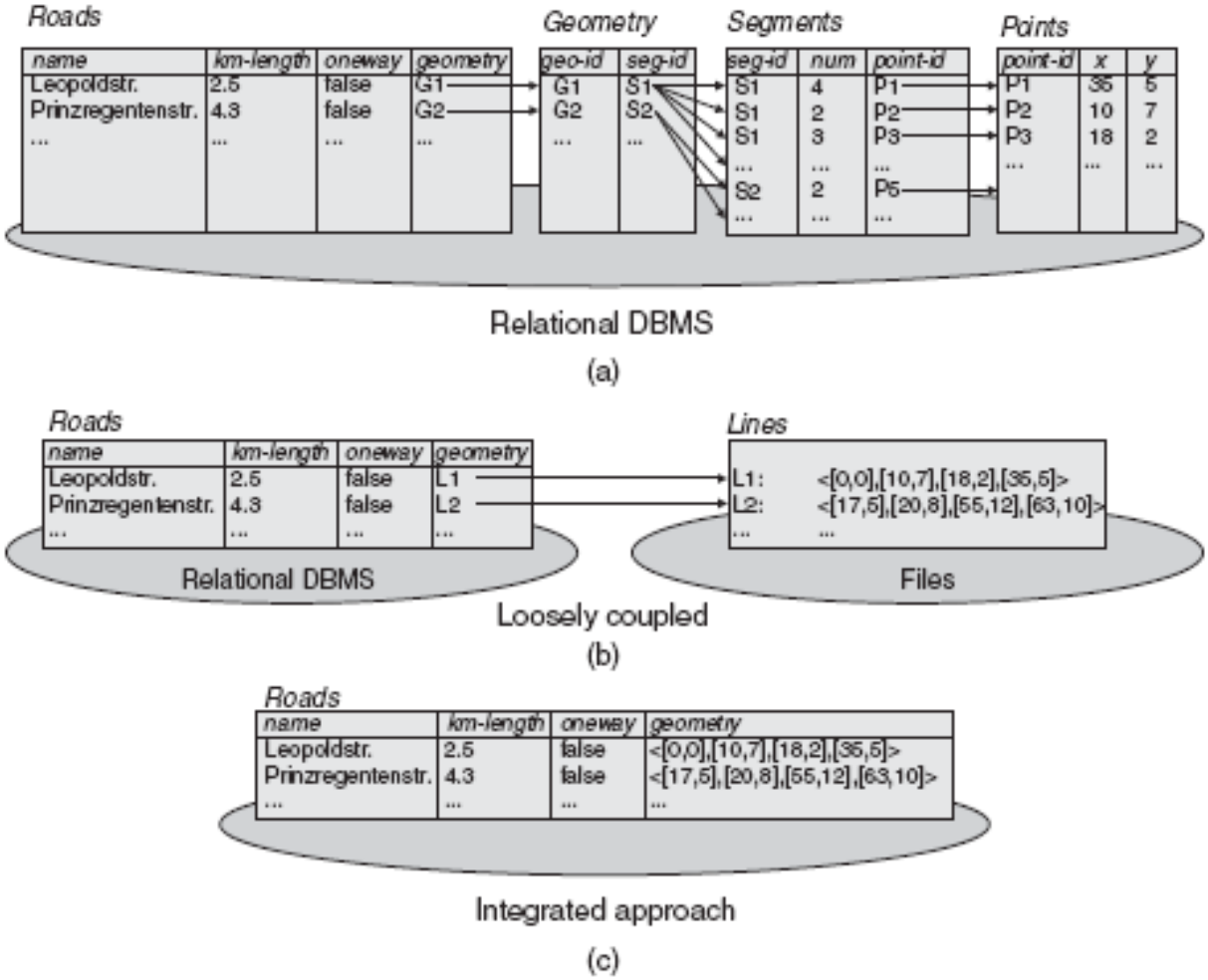
αναπαράστασης παρουσιάζονται αναλυτικά στο Κεφάλαιο 4. Στα πλαίσια των GIS, το χαμηλότερο layer που αντιμετωπίζει αυτά τα ζητήματα αναφέρεται ως *μοντέλο χωρικών δεδομένων*.

2.5.2.2 Κατηγορίες Βάσεων Δεδομένων για Χωρικά Αντικείμενα

Γενικά, υπάρχει μια διάκριση μεταξύ *σχεσιακών (relational)* και *αντικειμενοστραφών (object-oriented) DBMS* για την αποθήκευση συμβατικών, μη χωρικών δεδομένων. Σε μια σχεσιακή βάση δεδομένων, τα δεδομένα οργανώνονται σε πίνακες όπου κάθε σειρά ενός πίνακα αντιπροσωπεύει μια εγγραφή (record) και κάθε στήλη αντιπροσωπεύει ένα γνώρισμα (attribute). Μια εγγραφή πρέπει να είναι καλά ορισμένη δεδομένου ότι πρέπει να περιέχει έναν σταθερό αριθμό attributes, και κάθε attribute πρέπει να είναι ενός ορισμένου, απλού τύπου δεδομένων όπως *integer, float, character ή string.* Αυτό είναι δυνατόν να συσχετίζει εγγραφές διαφορετικών πινάκων έτσι ώστε κάθε εγγραφή ενός πίνακα να προσδιορίζεται από ένα μοναδικό κλειδί που διαμορφώνει ένα attribute της εγγραφής, και για να χρησιμοποιήσει αυτό το κλειδί ως αναφορά μέσα σε έναν άλλο πίνακα.

Οι αντικειμενοστραφείς DBMS, από την άλλη μεριά, υιοθετούν τους βασικούς μηχανισμούς του αντικειμενοστραφούς μοντέλου, όπως οι κλάσεις, οι μέθοδοι, η ενθυλάκωση (encapsulation), και η κληρονομικότητα για την οργάνωση των δεδομένων. Το αντίστοιχο μιας εγγραφής είναι ένα αντικείμενο που τοποθετεί τα δεδομένα των απλών ή σύνθετων τύπων και παρέχει διάφορες μεθόδους για τον χειρισμό τους. Οι αντικειμενοστραφείς DBMS παρέχουν πολύ καλύτερη απόδοση από τις σχεσιακές, αλλά χρειάζεται περισσότερος χρόνος και δεξιότητες για να σχεδιαστούν. Επομένως μερικά DBMS προϊόντα είναι βασισμένα σε μια υβριδική προσέγγιση που υιοθετεί τα οφέλη και των σχεσιακών και των αντικειμενοστραφών DBMS αποφεύγοντας τα μειονεκτήματά τους. Αυτά τα συστήματα είναι γνωστά υπό τον όρο *object-oriented relational DBMS*. Τα περισσότερα GIS λειτουργούν σε βασικές ή εκτεταμένες σχεσιακές DBMS για την αναπαράσταση των χωρικών αντικειμένων και των attributes οπότε δεν θα αναφερθούμε περαιτέρω στις αντικειμενοστραφής DBMS.

Για την αναπαράσταση των χωρικών αντικειμένων από σχεσιακές DBMS (βλ. **Εικόνα 8**), έχουν προσδιοριστεί οι ακόλουθες προσεγγίσεις:



Εικόνα 8: Βάσεις Δεδομένων σε GIS

2.5.2.3 Αναπαράσταση από σχεσιακές DBMS

Η πιο στοιχειώδης προσέγγιση είναι να διαμορφωθούν τα χωρικά αντικείμενα από μία συμβατική σχεσιακή DBMS όπως φαίνεται στην **8.a**. Το μειονέκτημα αυτής της προσέγγισης είναι ότι κανένας αποκλειστικός (dedicated) τύπος δεδομένων για την αντιπροσώπευση των χωρικών αντικειμένων δεν είναι διαθέσιμος, αλλά μόνο απλοί τύποι όπως *integer* και *string*. Κατά συνέπεια, τα χωρικά αντικείμενα πρέπει να συντεθούν από τις σχέσεις των αρχείων μεταξύ των πολυάριθμων πινάκων. Ένα άλλο πρόβλημα προκύπτει από το γεγονός ότι ο αριθμός των σημείων στα χωρικά αντικείμενα συνήθως δεν καθορίζεται, και ως εκ τούτου είναι απαραίτητο να αντιπροσωπευθεί κάθε ένα από αυτά από διάφορα αρχεία αμέσως. Το όφελος αυτής της προσέγγισης είναι ότι οι συμβατικές γλώσσες DBMS και οι query γλώσσες όπως η SQL μπορούν να χρησιμοποιηθούν, αλλά πάσχει από δυσκολίες στη διαχείριση και την επεξεργασία των δεδομένων καθώς επίσης και από την κακή απόδοση λόγω της σύνδεσης πολλών εγγραφών.

2.5.3. Προσέγγιση χαλαρής σύζευξης (Loosely coupled)

Η loosely coupled προσέγγιση αποτελείται από δύο υποσυστήματα, ένα για την αποθήκευση των attributes και ένα άλλο για τη διατήρηση των χωρικών αντικειμένων (**Εικόνα 8b**). Το πρώτο δίνεται από μια σχεσιακή DBMS, ενώ για το δεύτερο χρησιμοποιείται ένα σύστημα αρχείων. Τα χωρικά αντικείμενα αποθηκεύονται με ένα ορισμένο σχήμα στα αρχεία, και συνδέονται με τα αντίστοιχα περιγραφικά attributes στη σχεσιακή DBMS μέσω εσωτερικών προσδιοριστικών. Αν και αυτή η προσέγγιση είναι καταλληλότερη για την αποθήκευση και την επεξεργασία χωρικών αντικειμένων από την αποκλειστική χρήση μιας σχεσιακής DBMS, πάσχει όμως από δυσκολίες σχετικά με τις τεχνικές αποκατάστασης, την αναζήτηση, και τη βελτιστοποίηση.

2.5.4. Integrated προσέγγιση

Οι εκτεταμένες σχεσιακές DBMS παρέχουν μια ολοκληρωμένη προσέγγιση, με την εισαγωγή νέων χωρικών τύπων δεδομένων όπως σημεία, γραμμές (polylines) και πολύγωνα, καθώς επίσης και τροποποιημένες γλώσσες ερωταπαντήσεων (querying). Με τον τρόπο αυτό διευκολύνεται η υποβολή ερωτημάτων με κριτήριο τα attributes των χωρικών αντικειμένων και η μετέπειτα αποθήκευση των αντίστοιχων αποτελεσμάτων. Έτσι τα χωρικά στοιχεία αντιμετωπίζονται πολύ αποτελεσματικά και μπορεί να βελτιστοποιηθεί η διαδικασία αναζήτησης και επιλογής τους. Αυτή η ολοκληρωμένη προσέγγιση παρουσιάζεται στην **Εικόνα 8c**.

2.6. Τύποι Απεικόνισης-Προβολής Χαρτών για Εφαρμογές Διαδικτύου

Οποιαδήποτε ανάπτυξη εφαρμογής πλοήγησης στο διαδίκτυο έχει ως τελικό προιόν τη δημιουργία ενός χάρτη. Επομένως απαιτείται η επιλογή του κατάλληλου τύπου απεικόνισης εικόνας για την παρουσίαση του χάρτη αυτού. Οι τύποι (format) εικόνων που μπορούν να χρησιμοποιηθούν για την εμφάνιση ενός χάρτη παρουσιάζονται παρακάτω. Στην υλοποίηση της παρούσας εφαρμογής χρησιμοποιείται το τύπος svg.

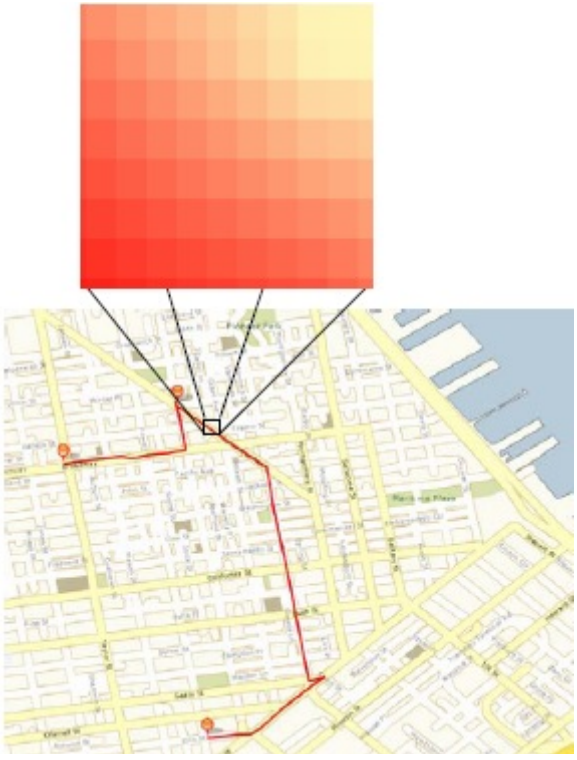
2.6.1. Raster Εικόνες

Η raster εικόνα αποτελείται από συγκεκριμένο αριθμό pixels ο οποίος περιγράφεται από τον όρο «ανάλυση». Επειδή κάθε raster εικόνα έχει συγκεκριμένη ανάλυση υπάρχουν όρια στον τρόπο χρήσης τους. Για παράδειγμα έστω μια εικόνα με ανάλυση 2000x1600=3,2 Mpixels την οποία θέλουμε να εκτυπώσουμε με μια καλή ανάλυση της τάξης των 200ppi (pixels per inch). Αυτό σημαίνει ότι οι διαστάσεις της εικόνας που θα εκτυπωθεί θα είναι:

H=2000/200=10 inches.

W=1600/200=8 inches.

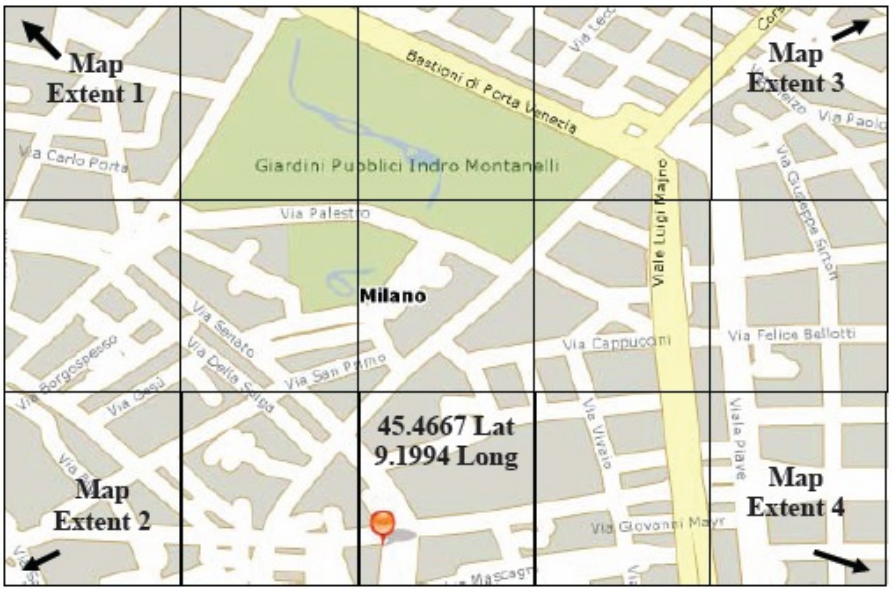
Όπως φαίνεται στην **Εικόνα 9**, οι raster χάρτες αποτελούνται από ένα δυσδιάστατο πίνακα (γραμμές και στήλες) χρωματισμένων pixels τα οποία σχηματίζουν την εικόνα του χάρτη. Σε υψηλά επίπεδα μεγέθυνσης οι raster εικόνες εμφανίζουν ακανόνιστες ακμές οι οποίες είναι οι άκρες των μεμονωμένων pixel. Αυτή η κοκκοποίηση συχνά αναφέρεται ως “pixelation” και εμφανίζεται όταν τα μεμονωμένα στοιχεία από τα οποία αποτελείται η εικόνα γίνονται ορατά στο μάτι. Οι εικόνες αυτές εμφανίζονται κανονικά όταν παρατηρούνται στην αιτηθείσα κλίμακα αλλά σε περίπτωση που η εικόνα μεγεθυνθεί τα μεμονωμένα pixels γίνονται εμφανή.



Εικόνα 9: Raster γραφικά που αποτελούνται από pixels με χρώμα

Το 2005 ξεκίνησε η παροχή χαρτών στη μορφή των tiled raster εικόνων. Ολόκληρη η εικόνα αποτελείτο από πολλές μικρότερες εικόνες οι οποίες δημιουργούσαν τον χάρτη που έβλεπε κανείς στον browser. Όλες οι εικόνες είχαν δημιουργηθεί ή προ-παραχθεί σε κάποια στιγμή πρωθύστερη του αιτήματος, από τον server που δημιουργούσε τους χάρτες, για προβολή τους. Το αποτέλεσμα ήταν να επιτυγχάνεται υψηλή απόδοση, καθώς ο server δε χρειαζόταν χρόνο για τη δημιουργία του χάρτη αλλά απλώς ανακαλούσε τα κατάλληλα αρχεία raster και τα έστελνε στον client. Επιπρόσθετα, αυτή η τεχνική επέτρεπε ομαλή πλοήγηση (panning) στο χάρτη. Από την άλλη όμως, επειδή οι εικόνες ήταν συγκεκριμένες υπήρχε περιορισμός στον τρόπο χειρισμού της

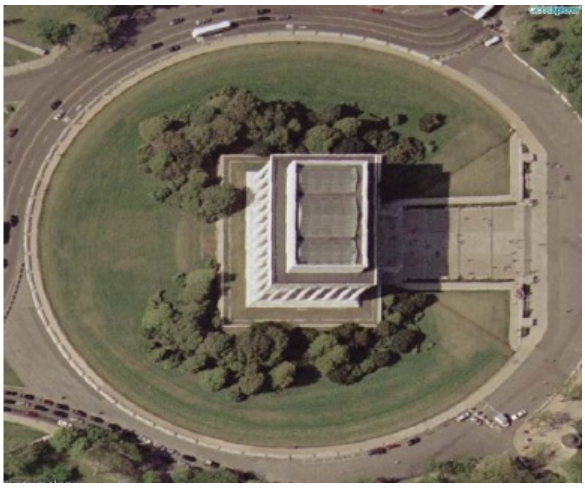
απεικόνισης του χάρτη. Για παράδειγμα τα map styles που καθορίζουν τα χρωματικά μοτίβα που θα χρησιμοποιηθούν για το χάρτη δεν μπορούσαν να μεταβληθούν. Τυπικά τα μόνα μετα-δεδομένα που είναι διαθέσιμα στον δημιουργό χαρτών raster είναι τα όρια (extents) γεωγραφικού μήκους και πλάτους όπως φαίνεται στην **Εικόνα 10**. Τα όρια αυτά είναι το ελάχιστο ορθογώνιο παραλληλόγραμμο που περιέχει το χάρτη και ορίζεται από 4 ζεύγη (ένα για κάθε γωνία του παραλληλογράμμου) γεωγραφικών συντεταγμένων μήκους και πλάτους. Αν ο σχεδιαστής γνωρίζει τα όρια αυτά μπορεί να χρησιμοποιήσει την εικόνα του χάρτη ως στατικό background καμβά για να τοποθετήσει αντικείμενα όπως σημεία ένδειξης πάνω από την εικόνα-χάρτη. Τα σημεία ένδειξης ευθυγραμμίζονται στο χάρτη μέσω κάποιας συνάρτησης της εφαρμογής η οποία μετατρέπει τις γεωγραφικές συντεταγμένες μήκους και πλάτους σε pixels της εικόνας του χάρτη.



Εικόνα 10: Προσομοίωση tiled Raster Χάρτη με τα όρια των Tiles και του Χάρτη να διακρίνονται

Η χρήση raster εικόνων σε χάρτες γίνεται για διάφορους λόγους. Καθένας μπορεί να έχει πρόσβαση μέσω internet σε raster γραφικά χωρίς να υπάρχει απαίτηση για ειδικό λογισμικό. Ο τελικός χρήστης μπορεί να έχει πρόσβαση σε εφαρμογές που περιέχουν raster εικόνες χωρίς να απαιτείται να εγκαταστήσει στο σύστημά του κάποιο ειδικό λογισμικό. Υπάρχουν χρήστες που για λόγους ασφαλείας αποφεύγουν την εγκατάσταση plug-ins και έτσι μόνο εφαρμογές που χρησιμοποιούν raster εικόνες είναι προσβάσιμες από όλους τους χρήστες του διαδικτύου. Επιπρόσθετα η δημιουργία των raster εικόνων γίνεται αποκλειστικά στον mapping server με αποτέλεσμα να μειώνεται ο κίνδυνος της μη απεικόνισης του χάρτη λόγω της επεξεργαστικής δυνατότητας του client.

Καθένα από τα είδη raster (GIF, JPEG και PNG) έχει τα πλεονεκτήματά του. Το GIF είναι ο βέλτιστος τύπος για απεικόνιση δρόμων ή χαρτών με γραμμές και επιπρόσθετα υποστηρίζει διαφάνεια (transparency). Το JPEG δημιουργεί το μικρότερο σε όγκο αρχείο όταν χρησιμοποιείται σε χάρτες με εικόνες και υποστηρίζει πάνω από 16 εκατομμύρια αποχρώσεις (Η **Εικόνα 11** παρέχει ένα παράδειγμα ενός χάρτη με εικόνες). Το PNG που έχει δημιουργηθεί ως ένας τύπος, χωρίς απαιτήσεις για πνευματικά δικαιώματα, ισοδύναμος με τον GIF, όταν ο GIF ήταν πατενταρισμένος, υποστηρίζει διαφάνεια και είναι μη απωλεστικός, που σημαίνει ότι δεν υπάρχει απώλεια πληροφορίας κατά τη συμπίεση.



Εικόνα 11: Ο JPEG είναι καλή επιλογή για απεικόνιση εικόνων από δορυφόρο

2.6.2. Διανυσματικές Εικόνες

Αντιθέτως με τις raster εικόνες οι διανυσματικές (vector graphics) είναι ανεξάρτητες από την ανάλυση. Ένα διανυσματικό γραφικό ορίζεται με μαθηματικούς όρους που σημαίνει ότι είναι απείρως αυξομειώσιμο. Στις εφαρμογές της Adobe, για παράδειγμα, οι μαθηματικές γραμμές ορίζονται ως καμπύλες Bezier. Τα ευθύγραμμα τμήματα σε μια Bezier ενώνονται με anchor σημεία. Κάθε σημείο anchor μπορεί να έχει μια επιπρόσθετη εφαπτόμενη γραμμή που ονομάζεται χειριστής διεύθυνσης και ορίζει πώς το ευθύγραμμο τμήμα καμπυλώνει.

Τα vector γραφικά δημιουργούνται με αυτόματο τρόπο αντίθετα με τα raster που δημιουργούνται μέσω σάρωσης ή από μια ψηφιακή μηχανή. Υπάρχουν εργαλεία μετατροπής raster σε vector αλλά συνήθως τα vector δημιουργούνται εξ αρχής.

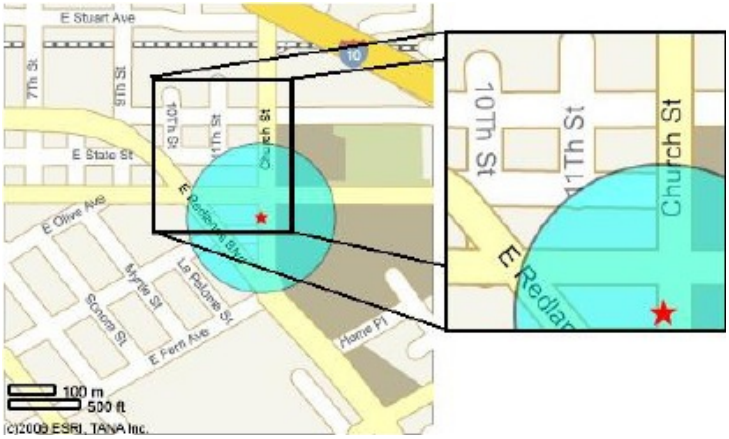
Η χρήση διανυσματικών γραφικών οδηγεί τις εφαρμογές πέραν της απλής απεικόνισης χαρτών παρέχοντας μια πλούσια διαδραστική εμπειρία στον client ο οποίος χρησιμοποιεί χαρακτηριστικά (features) μέσα στα δεδομένα του χάρτη. Οι πιο γνωστοί

και ευρέως χρησιμοποιούμενοι τύποι διανυσματικών γραφικών είναι ο Adobe Flash (SWF) και ο προτυποποιημένος Scalable Vector Graphics (SVG). Λόγω του ότι τα διανυσματικά γραφικά αποτελούνται από οδηγίες που προσδιορίζουν μαθηματικά σχήματα, οι σχεδιαστές μπορούν να δημιουργήσουν εφαρμογές που χειρίζονται ολόκληρα αντικείμενα του χάρτη. Τα διανυσματικά γραφικά εμφανίζονται με την ίδια μορφή ανεξάρτητα εάν ζητηθεί μεγέθυνση ή σμίκρυνσή τους και επιτρέπουν το δυναμικό επανασχεδιασμό τους χωρίς να απαιτείται κλήση για νέα εικόνα από τον map server.

Για παράδειγμα για τον ορισμό μιας ακτίνας γύρω από συγκεκριμένη γεωγραφική συντεταγμένη στο χάρτη, η εφαρμογή θα πρέπει να εισάγει μια γραμμή κώδικα όμοια με την παρακάτω μέσα σε ένα αρχείο svg. Όταν ο client εμφανίσει την εικόνα ο κύκλος θα εμφανιστεί στο χάρτη:

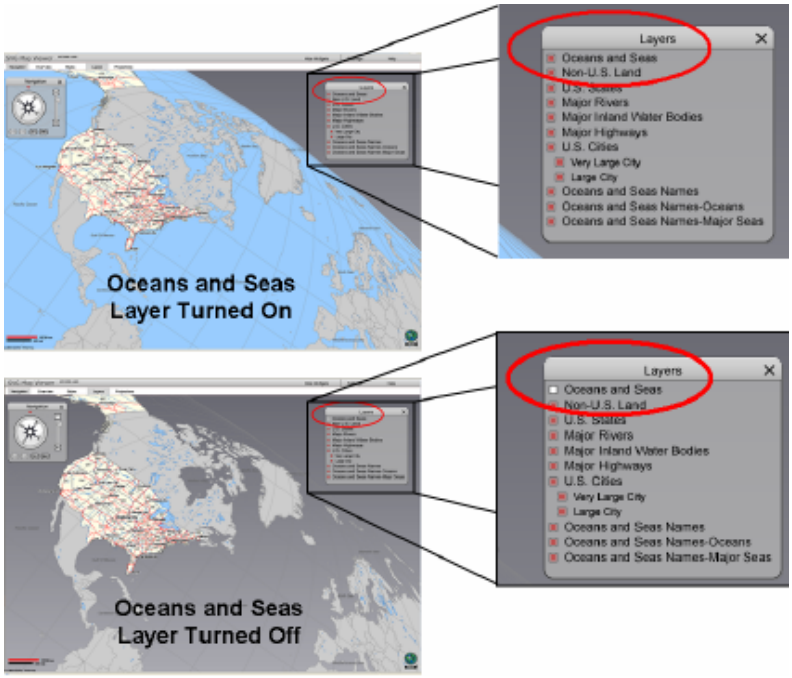
<circle cx=’103’ cy=’103’ r=’50’ style=’fill:cyan; opacity:0.3; stroke:black’ />

Ο παραπάνω κώδικας προσθέτει ένα ημιδιάφανο κυανό κύκλο ακτίνας 50 με κέντρο τη θέση (103,103) περιγράμματος χρώματος μαύρου, όπως φαίνεται στην **Εικόνα 12**.



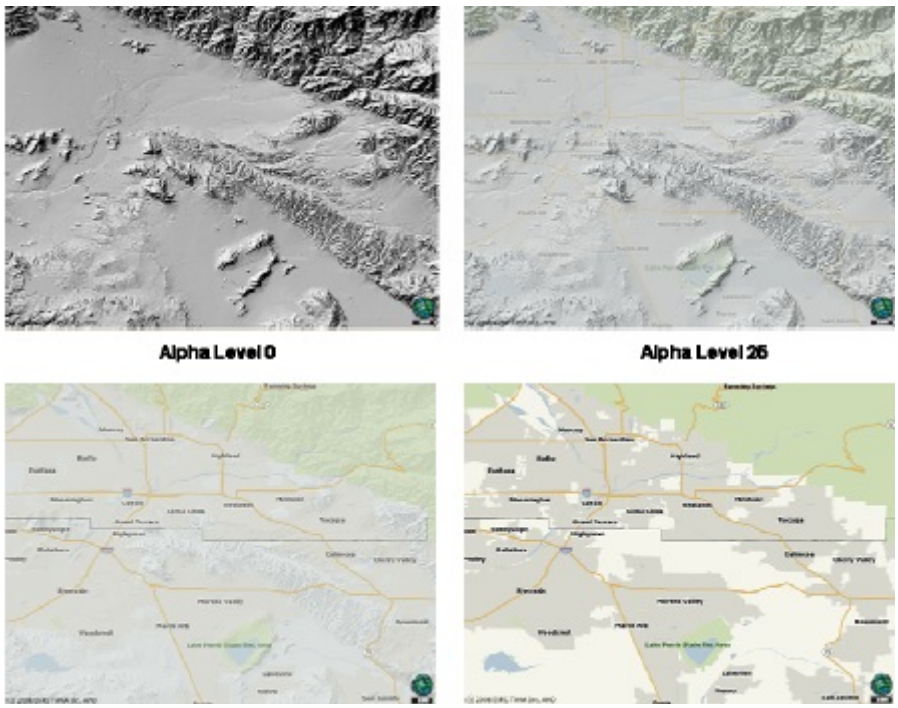
Εικόνα 12: Τα διανυσματικά γραφικά αποτελούνται από εντολές γεωμετρίας

Η **Εικόνα 13** απεικονίζει τον SVG Map Viewer των ArcWeb Services όπου φαίνεται μια εφαρμογή που χρησιμοποιεί τη λειτουργικότητα των SVG για να βελτιώσει το αποτέλεσμα που λαμβάνει ο τελικός χρήστης. Ανάμεσα σε άλλες λειτουργίες ο χρήστης μπορεί να θέσει on ή off τα layers, να μετακινήσει τη γη υπό διαφορετικές γωνίες και να αλλάξει την προβολή του χάρτη.



Εικόνα 13: Εικόνες πριν και μετά: Ο χρήστης θέτει layer και σχήματα on και off

Οι εφαρμογές που χρησιμοποιούν διανυσματικά γραφικά είναι γρήγορες για πολλούς λόγους. Οι πιο βασικές λειτουργίες που είναι η εμφάνιση ή απόκρυψη των επιπέδων με γεωγραφικά δεδομένα καθώς και η μεγέθυνση ή σμίκρυνση, μπορούν να γίνουν τοπικά χωρίς την ανάγκη να γίνει νέα κλήση στον server. Όπως φαίνεται στην **Εικόνα 14**, με το που θα ανακληθούν τα δεδομένα από την εφαρμογή όπως ο ArcWeb Service Explorer, ο τελικός χρήστης μπορεί να αλλάξει τη μορφή του χάρτη ή άλλα χαρακτηριστικά του «on the fly». Για αυξημένη λειτουργικότητα όπως η χωρική αναζήτηση, η πληροφορία μπορεί να προστεθεί εύκολα ως ένα layer αναζήτησης στην εφαρμογή του χάρτη το οποίο θα παρέχεται από το API της Web service.



Εικόνα 14: Αλλάζοντας τη διαφάνεια των layers on the fly

Ο **Πίνακας 2** δίνει μια επισκόπηση των διαφορών ανάμεσα στους τύπους SWF και SVG. Ο SWF ο οποίος μερικές φορές αναφέρεται και ως Flash είναι ένας ιδιοκτησιακός (proprietary) διανυσματικός τύπος γραφικών που δημιουργήθηκε από την Adobe/Macromedia Flash. Τα αρχεία SWF εμφανίζονται από τον Adobe Flash Player ο οποίος μπορεί να είναι είτε αυτόνομος player είτε plug-in στον browser.

	Flash (SWF) Maps	SVG Maps
Διανυσματικοί	ΝΑΙ	ΝΑΙ
Browser Plug-in	Οι περισσότεροι χρήστες ήδη χρησιμοποιούν το plug-in Flash player	Ελάχιστοι χρήστες έχουν ήδη το plug-in
Αδειοδότηση	Adobe proprietary	Non-proprietary
Client-Side Programming Control	Adobe Flash Player ή ενσωματωμένο στο API	Οι περισσότεροι XML parsers
Δυνατότητα τροποποίησης του map file	ΟΧΙ (δυαδικό αρχείο)	ΝΑΙ (XML αρχείο)

Πίνακας 2: Σύγκριση μεταξύ SWF και SVG

Ο τύπος SVG είναι ανοικτού κώδικα προδιαγραφών μη ιδιοκτησιακού καθεστώτος που δημιουργήθηκε από το World Wide Web Consortium, το οποίο έχει δημιουργήσει και τα HTML, CSS και XML. Επειδή το SVG είναι ανοικτού κώδικα και βασίζεται στην XML οι περισσότεροι από τους γνωστούς XML parsers μπορούν να διαβάσουν τον SVG κώδικα (αντίθετα με τα flash αρχεία). Σαν αποτέλεσμα το περιεχόμενο των SVG αρχείων είναι αναγνώσιμο και μπορεί να διαμορφωθεί από γλώσσες προγραμματισμού

που αλληλεπιδρούν με XML. Τα SVG αρχεία μπορούν επίσης να ενσωματωθούν σε Adobe PDF έγγραφα. Το χαρακτηριστικό αυτό είναι ιδιαίτερα σημαντικό στις περιπτώσεις που υπάρχει απαίτηση ποιοτικής εκτύπωσης χαρτών ενσωματωμένων σε αναφορές. Επιπρόσθετα ένα αρχείο SVG μπορεί να γραφεί και διαμορφωθεί από οποιοδήποτε text editor καθώς ουσιαστικά αποτελεί ένα XML αρχείο ενώ έχει τη δυνατότητα χρήσης CSS όσον αφορά τον έλεγχο του layout, στυλ, χρωμάτων και γραμματοσειρών. Επισημαίνεται όμως ότι δεν είναι όλοι οι SVG renderers συμβατοί με CSS.

2.6.3. Σύγκριση Raster και Vector Εικόνων για Απεικόνιση Χαρτών

Παραδοσιακά οι χάρτες μέσω διαδικτύου βασίζονταν σε raster γραφικά τα οποία δημιουργούνταν από έναν Web Server και παραδίδονταν στον φυλλομετρητή του εξυπηρετούμενου (client browser). Στις μέρες μας η τάση είναι η δημιουργία εφαρμογών χαρτογραφίας με χρήση διανυσματικών γραφικών.

Οι raster χάρτες δεν απαιτούν κάποιο ειδικό plug-in προκειμένου να απεικονιστούν στον client. Από την άλλη όμως αυτού του είδους οι χάρτες έχουν μικρή λειτουργικότητα. Από τη στιγμή που δημιουργηθεί η εικόνα παραμένει αμετάβλητη. Για την τροποποίησή της απαιτείται επιπλέον κλήση στον map server. Για την επίτευξη γρήγορης απεικόνισης τέτοιων εικόνων καθώς και για ομαλή πλοήγηση (panning) συνήθως ο server έχει δημιουργήσει εκ των προτέρων τις εικόνες και τις αποδίδει αμέσως μετά την κάθε κλήση. Έτσι από τη στιγμή που οι εικόνες δημιουργηθούν στον server μπορούν να χρησιμοποιηθούν μεν πολλές φορές αλλά από την άλλη δεν είναι δυνατόν να τροποποιηθούν. Αντιθέτως οι διανυσματικοί χάρτες μπορούν να προσαρμοστούν ανά πάσα στιγμή επιτρέποντας τη δημιουργία δυναμικών εφαρμογών διαδικτύου όπου οι χρήστες αλληλεπιδρούν με τα χαρτογραφικά δεδομένα.

Στον **Πίνακας 3** δίδονται περιληπτικά τα χαρακτηριστικά και οι δυνατότητες των χαρτών μορφής raster, tiled raster και ανύσματος (vector).

Μορφή Χάρτη	Δυνατότητα εναλλαγής layer, style, projection	Κατάλληλο για click, Drag and Pan	Προσδιορισμός των Features	Highlight, Animate Features	Απαίτηση για χρήση ειδικού plug-in
Raster (JPEG, GIF, PNG)	ΝΑΙ	ΟΧΙ	Απαιτεί νέα κλήση (request) σε server	ΟΧΙ	ΟΧΙ

Tiled Raster (overlaid JPEG, GIF, PNG)	ΟΧΙ	ΝΑΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ
Vector (SWF, SVG)	ΝΑΙ	ΝΑΙ	Δεν απαιτεί νέα κλήση (request) σε server	ΝΑΙ	ΝΑΙ

Πίνακας 3: Σύγκριση λειτουργικότητας Raster και Vector Χαρτών

Παραδοσιακά οι χάρτες στο διαδίκτυο δημιουργούνταν στον server και παραδίδονταν στον client με τη μορφή γραφικών τύπου μορφής JPEG ή GIF. Εάν ο χρήστης επιθυμούσε να χειριστεί το χάρτη (για παράδειγμα ζητούσε επιπλέον layer) η client εφαρμογή έκανε νέα κλήση στον server ο οποίος δημιουργούσε ένα νέο χάρτη.

ΚΕΦΑΛΑΙΟ 3

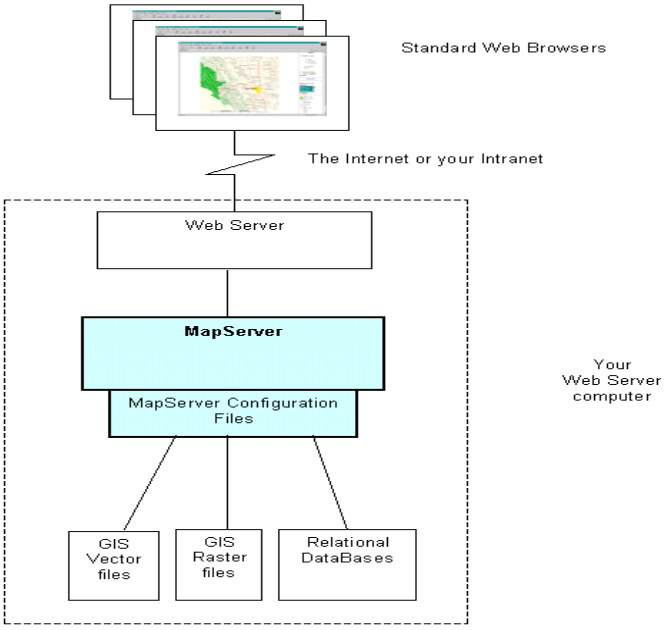
ΚΥΡΙΟ ΛΟΓΙΣΜΙΚΟ ΑΝΑΠΤΥΞΗΣ ΕΦΑΡΜΟΓΗΣ

3.1. Γενικά

Οποιαδήποτε ανάπτυξη εφαρμογής συστήματος υπηρεσιών θέσης απαιτεί τη χρήση ενός server ο οποίος θα δημιουργεί και θα αποδίδει τους χάρτες κατόπιν σχετικού αιτήματος του χρήστη. Ο server που χρησιμοποιήθηκε στην εφαρμογή μας είναι ο Mapserver.[x],[xi]

Ο Mapserver είναι ένα περιβάλλον ανάπτυξης open source για τη δημιουργία χωρικών εφαρμογών στο internet. Ο Mapserver δεν αποτελεί ένα ολοκληρωμένο GIS σύστημα αλλά είναι αποτελεί ένα open source, ολοκληρωμένο και αποδοτικό σύστημα για την απεικόνιση χωρικών δεδομένων (χάρτες, εικόνες, και διανυσματικά στοιχεία) μέσω του Διαδικτύου. Εκτός από το browsing των GIS δεδομένων, ο MapServer επιτρέπει τη δημιουργία "γεωγραφικών χαρτών", δηλαδή χαρτών που μπορούν να κατευθύνουν τους χρήστες σε κάποιο σημείο ενδιαφέροντος.

Ο MapServer αναπτύχθηκε αρχικά από το πανεπιστήμιο της Μινεσότα (UMN)[xii] στα πλαίσια του προγράμματος Fornet σε συνεργασία με τη NASA και το Τμήμα Φυσικών Πόρων της Μινεσότα (MNDNR). Τώρα, το MapServer project φιλοξενείται από το TerraSIP πρόγραμμα, το οποίο υποστηρίζεται από τη NASA μεταξύ του UMN και μιας κοινοπραξίας διαχείρισης γης. Η αρχιτεκτονική του Mapserver φαίνεται στην **Εικόνα 15**.



Εικόνα 15: Αρχιτεκτονική του Mapserver

3.2. Πλεονεκτήματα Mapserver

Η χρήση του Mapserver έχει αρκετά πλεονεκτήματα. Ένα από αυτά είναι η δυνατότητα που προσφέρει για ευρεία πρόσβαση στην χαρτογραφημένη πληροφορία από πολλούς χρήστες, ιδιαίτερα μέσω του Διαδικτύου. Ο MapServer, δίνει τη δυνατότητα σε απλούς χρήστες να δημιουργήσουν χάρτες χωρίς να χρειάζονται κάποια ιδιαίτερα εργαλεία ή τη βοήθεια των προγραμματιστών. Επιπλέον αποτελεί μια από τις λίγες διαθέσιμες λύσεις για εκείνους που διαχειρίζονται διαφορετικά data formats. Ο MapServer, μέσω της χρήσης των βιβλιοθηκών όπως το GDAL/OGR, μπορεί να έχει πρόσβαση σε διάφορα data formats χωρίς μετατροπή δεδομένων. Ένας άλλος σημαντικός λόγος χρήσης του είναι το γεγονός ότι είναι ανοικτό λογισμικό (open source).

Θεωρήστε ότι θα μπορούσατε να έχετε μια συλλογή 10 διαφορετικών συνόλων δεδομένων χαρτογράφησης, τα οποία πρέπει να εμφανιστούν στον ίδιο χάρτη ταυτόχρονα χωρίς οποιαδήποτε από αυτά τα δεδομένα να μετατραπούν από το αρχικό format τους. Τα αρχικά format περιλαμβάνουν εκείνα που χρησιμοποιούνται από τους διαφορετικούς εμπορικούς προμηθευτές. Τα ESRI shapefiles, τα αρχεία σχεδίου Intergraph Microstation (DGN), τα MapInfo TAB αρχεία, και οι Oracle χωρικές βάσεις δεδομένων μπορούν όλες να χαρτογραφηθούν μαζί χωρίς μετατροπή. Άλλα μη εξειδικευμένα formats μπορούν να χρησιμοποιηθούν επίσης, συμπεριλαμβανομένων των προτύπων OGC για τη γλώσσα GML, τον Web Map Server (WMS), τον Web Feature Server (WFS), την PostGIS και άλλες βάσεις δεδομένων. Η δυνατότητα να υπάρξει ταυτόχρονη πρόσβαση στα διαφορετικά format δεδομένων χωρίς μετατροπή κάνει τον MapServer μια από τις λίγες επιλογές για εκείνους που δεν μπορούν (ή δεν θέλουν) να κάνουν μια μετατροπή σε ένα συγκεκριμένο format.

3.2.1. Τύποι Δεδομένων

Ο MapServer υποστηρίζει αρκετά format. Μερικά format υπάρχουν στην εγκατάσταση του MapServer, ενώ τα άλλα υποστηρίζονται μέσω των βιβλιοθηκών GDAL/OGR. Η πρόσβαση μέσω των βιβλιοθηκών προσθέτει ένα πρόσθετο επίπεδο επικοινωνίας μεταξύ του MapServer και της πηγής δεδομένων (που μπορούν να προκαλέσουν την κακή απόδοση σε μερικές περιπτώσεις).

Γενικά, τα format που υποστηρίζονται εγγενώς από τον MapServer τρέχουν γρηγορότερα από εκείνα που χρησιμοποιούν τις βιβλιοθήκες GDAL/OGR. Παραδείγματος χάριν, το πιο βασικό format του MapServer είναι το ESRI shapefile ή GeoTiff. Το OGR υποστηρίζει το format αρχείων U.S. Census TIGER. Η διαφορά απόδοσης όταν φορτώνεται ένα TIGER αρχείο ή ένα shapefile μπορεί να είναι

σημαντική. Ο τύπος μορφοποίησης των δεδομένων μπορεί να αποτελέσει συχνά πρόβλημα. Εάν τα δεδομένα σε ένα αρχείο είναι δομημένα με έναν τρόπο που δεν τα καθιστά εύκολα προσπελάσιμα ή απαιτεί πολυάριθμα επίπεδα μετάφρασης, επηρεάζεται η ταχύτητα φόρτωσης των χαρτών.

Ο γενικός εμπειροτεχνικός κανόνας για την καλύτερη απόδοση είναι να χρησιμοποιηθεί το ESRI shapefile format ή το format εικόνας GeoTiff. Εάν όμως υπάρχει πρόσβαση στα δεδομένα μέσω ενός δικτύου, τότε αντί της χρήσης shapefile ή GeoTiff ενδέχεται η καλύτερη επιλογή να είναι η αποθήκευση των στοιχείων στη βάση δεδομένων PostGIS. Επειδή η PostGIS επεξεργάζεται τα ερωτήματα για τα δεδομένα στον server, μόνο τα επιθυμητά αποτελέσματα στέλνονται από το δίκτυο. Η server-side επεξεργασία σε μια βάση δεδομένων PostGIS μπορεί να βελτιώσει σημαντικά την απόδοση των εφαρμογών του Mapserver.

3.2.2. Ανοικτή Αρχιτεκτονική και Διαλειτουργικότητα

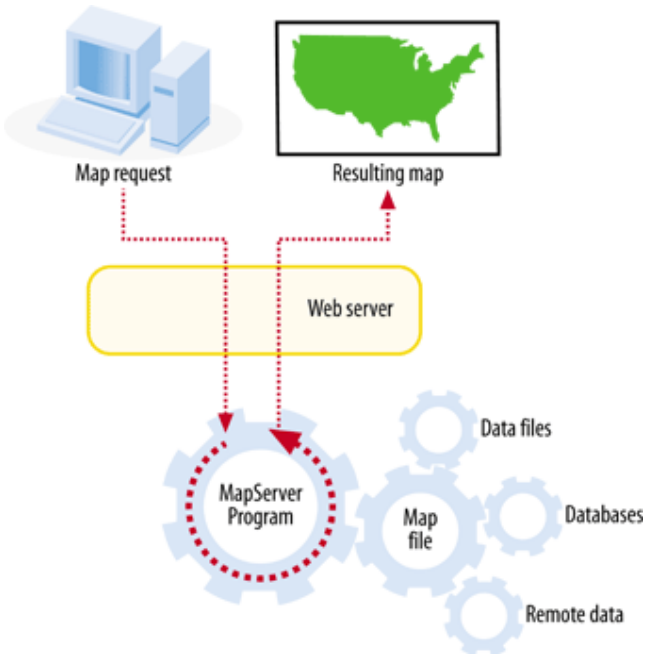
Ο MapServer και τα ενισχυτικά εργαλεία υποστήριξης του είναι διαθέσιμα για διάφορα λειτουργικά συστήματα. Επιπλέον, η λειτουργία του MapServer μπορεί να προσεγγιστεί από πολλές γλώσσες προγραμματισμού, καθιστώντας το πιθανό να ενσωματώσει τη λειτουργία του MapServer σε συνηθισμένα προγράμματα. Ο MapServer μπορεί να χρησιμοποιηθεί σε συνηθισμένα περιβάλλοντα όπου άλλοι web mapping servers ίσως να μην μπορούν να εκτελεστούν.

Επειδή ο MapServer είναι λογισμικό open source, οι προγραμματιστές μπορούν να βελτιώσουν, να διορθώσουν, να τροποποιήσουν τον πραγματικό κώδικα του MapServer, όπως επίσης και να τον προσαρμόσουν σε νέα λειτουργικά συστήματα ή πλατφόρμες.

Ο MapServer είναι πρωτίστως μια εφαρμογή παρακολούθησης και δημιουργίας χαρτών. Οι χρήστες έχουν πρόσβαση στους χάρτες μέσω ενός web browser ή άλλων data sharing πρωτοκόλλων Διαδικτύου. Αυτό επιτρέπει την οπτική διανομή των πληροφοριών χαρτογράφησης και τη διανομή δεδομένων με άλλες εφαρμογές σε πραγματικό χρόνο χρησιμοποιώντας τις προδιαγραφές του OGC. Ο MapServer μπορεί να εκτελέσει μετατροπή δεδομένων διαβάζοντας διαφορετικά format και να παρέξει πρόσβαση σε άλλους server ή εφαρμογές χρησιμοποιώντας κοινά πρωτόκολλα. Ο MapServer δεν είναι μόνο ένα εργαλείο ανάλυσης, αλλά μπορεί να παρουσιάσει πληροφορίες χαρτογράφησης χρησιμοποιώντας διαφορετικές χαρτογραφικές τεχνικές για να απεικονίσει τα αποτελέσματα.

3.3. Περιγραφή Λειτουργίας Server

Ο MapServer συνήθως λειτουργεί πίσω από μια web server εφαρμογή. Ο web server λαμβάνει τα αιτήματα για πρόσβαση στους χάρτες και τα περνά στον MapServer ο οποίος αναλαμβάνει τη δημιουργία του ζητούμενου χάρτη. Ο MapServer παράγει τη ζητούμενη εικόνα χαρτών και την παραδίδει στο web server, ο οποίος την διαβιβάζει πίσω στο χρήστη.Η **Εικόνα 16** δείχνει πως ο χρήστης αλληλεπιδρά με το web server, ο οποίος στη συνέχεια, υποβάλλει τα αιτήματα στο MapServer πρόγραμμα.



Εικόνα 16: Βασική λειτουργία μιας Mapserver εφαρμογής

Η αρχική λειτουργία του MapServer είναι να διαβάζει τα δεδομένα από τις διάφορες πηγές και να τοποθετεί αυτά τα layers μαζί σε ένα γραφικό αρχείο, επίσης γνωστό ως *map image*. Κάθε layer «κάθεται» ή σχεδιάζεται πάνω από άλλα και τότε αποτυπώνεται σε ένα web-friendly γραφικό για να το δει ο τελικός χρήστης. Ένα καλό παράδειγμα των αποτελεσμάτων της διαδικασίας επικάλυψης και χαρτογράφησης μπορεί να φανεί στην **Εικόνα 17**. Φαίνεται μια δορυφορική εικόνα (από έναν μακρινό server), οι γραμμές των δρόμων και οι θέσεις των πόλεων. Οι ετικέτες των πόλεων παράγονται δυναμικά από τον MapServer.

Εικόνα 17: Χάρτης που απεικονίζει διάφορα layers πληροφορίας

Αυτή η διαδικασία της σχεδίασης (rendering) εμφανίζεται κάθε φορά που υποβάλλεται ένα αίτημα στον MapServer για να δημιουργήσει έναν νέο χάρτη, παραδείγματος χάριν, όταν ένας χρήστης μεγεθύνει ένα χάρτη (zoom in). Αυτή η διαδικασία ακολουθείται επίσης όταν ένας χρήστης ζητά manually την επανασχεδίαση ενός χάρτη, όπως όταν αλλάξει το περιεχόμενο ενός layer δεδομένων και ο χρήστης θέλει να δει την αλλαγή.

Ο MapServer μπορεί να λειτουργήσει με δύο διαφορετικούς τρόπους: το CGI και το MapScript mode. Στο CGI mode, ο MapServer λειτουργεί σε ένα web server περιβάλλον σαν CGI script. Σ' αυτή την περίπτωση είναι εύκολο να «στηθεί» και παράγει μια γρήγορη και απλή εφαρμογή. Στο MapScript mode, το MapServer API είναι προσπελάσιμο από την Perl, την Python, την PHP ή την Java. Η MapScript διεπαφή επιτρέπει μια ευέλικτη, πλούσια σε features εφαρμογή που έχει τη δυνατότητα να εκμεταλλευθεί τα πρότυπα απεικόνισης (templates) του MapServer.

Ο Mapserver είναι ένα πρόγραμμα που βασίζεται σε templates. Όταν εκτελείται πρώτη φορά σαν απάντηση σε ένα web request, διαβάζει ένα configuration file (το οποίο αποκαλείται mapfile) που περιγράφει τα layers και τα άλλα συστατικά του χάρτη. Αφού διαβάσει το mapfile, σχεδιάζει και αποθηκεύει το χάρτη. Στη συνέχεια, διαβάζει ένα ή περισσότερα HTML templates, τα οποία προσδιορίζονται στο mapfile. Κάθε template αποτελείται από συμβατικά HTML markup tags και τα MapServer substitution strings. Αυτά τα strings χρησιμοποιούνται, παραδείγματος χάριν, για να διευκρινίσουν τα μονοπάτια για την εικόνα του χάρτη, που ο MapServer έχει δημιουργήσει, για να προσδιορίσει ποια layers πρόκειται να αποδοθούν, και για να διευκρινίσουν το επίπεδο

του zoom και την κατεύθυνση. Ο MapServer αντικαθιστά τις τρέχουσες τιμές για αυτά τα strings και στέλνει έπειτα το data stream στον web server, ο οποίος το διαβιβάζει έπειτα στον browser. Όταν ένας χρήστης που πραγματοποιεί ένα αίτημα αλλάζει οποιαδήποτε δεδομένα στη φόρμα της σελίδας (αλλάζοντας για παράδειγμα την κατεύθυνση του ζουμ ή το επίπεδο του ζουμ) και κάνει click στο κουμπί υποβολής (submit), ο MapServer λαμβάνει ένα αίτημα από τον web server με αυτές τις νέες τιμές. Κατόπιν ο κύκλος αρχίζει πάλι.

Ο MapServer εκτελεί αυτόματα διάφορες εργασίες κατά την παραγωγή ενός χάρτη. Ονομάζει τα features και αποτρέπει τις συγκρούσεις (collisions) μεταξύ των γειτονικών ετικετών. Επιτρέπει τη χρήση και bitmapped και TrueType γραμματοσειρών. Τα μεγέθη των ετικετών μπορούν να καθοριστούν ή να διαμορφωθούν στην ίδια κλίμακα με την κλίμακα του χάρτη. Η επιλογή να μην τυπωθούν οι ετικέτες για τις διευκρινισμένες σειρές κλίμακας χαρτών παρέχεται επίσης.

Επίσης ο MapServer δημιουργεί λεζάντες και scale bars (τα οποία μπορούν να διαμορφωθούν στο mapfile) και παράγει τους χάρτες αναφοράς. Ένας χάρτης αναφοράς παρουσιάζει το περιεχόμενο του χάρτη που απεικονίζεται εκείνη τη στιγμή. Παραδείγματος χάριν, εάν η περιοχή ενδιαφέροντος είναι η βόρεια Ελλάδα, ο χάρτης αναφοράς θα παρουσίαζε ένα μικρό χάρτη της βόρειας Ελλάδας (με τη μορφή επισκόπησης), με τα περιεχόμενα του τρέχοντος χάρτη να αποτυπώνονται μέσα σε αυτόν. Τα zoom in, zoom out και pan είναι υπό τον έλεγχο των χρηστών.

Ο MapServer δημιουργεί τους χάρτες τοποθετώντας τα layers το ένα πάνω από το άλλο. Κάθε ένα layer που αποδίδεται, τοποθετείται στην κορυφή του σωρού. Κάθε layer επιδεικνύει τα features που επιλέγονται από ένα σύνολο δεδομένων. Τα features που επιδεικνύονται μπορούν να επιλεγούν με τη χρησιμοποίηση κανονικών εκφράσεων Unix, συγκρίσεων strings και λογικών εκφράσεων. Λόγω της ομοιότητας των δεδομένων και της ομοιότητας των παραμέτρων απεικόνισης (όπως κλίμακα, χρώματα, και ετικέτες), μπορούμε να θεωρήσουμε ένα layer σαν ένα θέμα. Η απεικόνιση των layers είναι υπό interactive έλεγχο, που επιτρέπει στο χρήστη να επιλέξει ποια layers πρόκειται να αποδοθούν. Ο MapServer διαθέτει ισχυρές και περίπλοκες ικανότητες ερώτησης, αλλά στο CGI mode στερείται τα εργαλεία που επιτρέπουν το είδος ανάλυσης που παρέχεται από ένα GIS.

Αυτή η επισκόπηση έχει περιγράψει μερικά από τα χαρακτηριστικά γνωρίσματα του MapServer και έχει παρουσιάσει γιατί δεν είναι ένα πλήρες GIS: δεν παρέχει κανένα ενσωματωμένο εργαλείο βάσης δεδομένων DBMS (database management system), οι

αναλυτικές δυνατότητές της είναι περιορισμένες, και δεν έχει κανένα εργαλείο για γεω-αναφορές (georeferencing).

Δεδομένου ότι οι λειτουργίες του MapServer μπορούν να προσεγγιστούν μέσω ενός API από διάφορες γλώσσες προγραμματισμού (όπως η PHP, η Perl, η Java και η Python), μπορεί να χρησιμεύσει ως μια ισχυρή χωρική εφαρμογή που έχει πολλές από τις λειτουργίες ανάλυσης και αναφοράς ενός αληθινού GIS. Επιπλέον, ενώ δεν υπάρχει κανένα ενσωματωμένο εργαλείο για τον χειρισμό χωρικών δεδομένων, υπάρχουν σύνολα εργαλείων τρίτων που εκτελούν πολλές (αν και όχι όλες) από αυτές τις λειτουργίες.

Όταν ο MapServer «τρέχει» ως CGI σε ένα web περιβάλλον, μπορεί να αποδώσει χάρτες, να παρουσιάσει τα δεδομένα των attributes, και να πραγματοποιήσει τις στοιχειώδεις χωρικές ερωτήσεις. Όταν προσεγγίζεται μέσω του API, η εφαρμογή γίνεται σημαντικά ισχυρότερη. Σε αυτό το περιβάλλον, ο MapServer μπορεί να εκτελέσει τις ίδιες εργασίες που θα μπορούσε να κάνει ως CGI, αλλά έχει επίσης πρόσβαση στις εξωτερικές βάσεις δεδομένων μέσω του προγράμματος ελέγχου, καθώς επίσης και την πιο σύνθετη λογική και ένα μεγαλύτερο ρεπερτόριο των πιθανών συμπεριφορών.

3.3.1. CGI Εφαρμογή

Η απλούστερη μορφή του MapServer τρέχει σαν μία εκτελέσιμη CGI εφαρμογή σε έναν web server. Τεχνικά, ο MapServer θεωρείται μια HTTP-based stateless διαδικασία. Μια stateless διαδικασία σημαίνει ότι γίνεται η επεξεργασία ένα αιτήματος και μετά σταματά να τρέχει. Μια CGI εφαρμογή λαμβάνει τα αιτήματα από ένα web server, τα επεξεργάζεται, και μετά επιστρέφει μια απάντηση ή δεδομένα στο web server. Η CGI εφαρμογή είναι ο πιο δημοφιλής τρόπος χρησιμοποίησης του Mapserver και αυτό οφείλεται στην απλότητα του, που προκύπτει από το γεγονός ότι καμιά προγραμματιστική εργασία δεν απαιτείται για να λειτουργήσει Οι ενέργειες που απαιτούνται για να τρέξει η εφαρμογή είναι να δημιουργηθεί ένα text-based, ένα runtime configuration file και να δημιουργηθεί μια ιστοσελίδα, με τέτοιο τρόπο ώστε να μπορεί να εξυπηρετηθεί από ένα web server.

Το CGI MapServer εκτελέσιμο επιδρά σαν ενδιάμεσο μεταξύ των χαρτογραφικών δεδομένων των αρχείων και του web server προγράμματος που ζητά το χάρτη. Τα αιτήματα περνούν υπό μορφή CGI παραμέτρων από τον web server στον MapServer. Οι εικόνες που δημιουργούνται από τον MapServer ανατροφοδοτούνται έπειτα στον web server και τελικά στον web browser του χρήστη.

Ο αρχικός στόχος είναι να παράγονται χάρτες σε ένα CGI περιβάλλον, στο οποίο ένας χρήστης έχει πρόσβαση σε έναν Apache web server από έναν web browser. Σε αυτό το περιβάλλον, ο Apache καλεί τον MapServer, περνώντας του τις μεταβλητές της φόρμας από τον web browser. Χρησιμοποιώντας αυτές τις πληροφορίες, ο MapServer παράγει τις εικόνες και μία ιστοσελίδα, την οποία ο Apache προωθεί πίσω στον browser. Φυσικά, ο MapServer χρειάζεται κάποιες παραμέτρους για να δημιουργήσει έναν χάρτη. Στην πραγματικότητα, μια CGI MapServer web εφαρμογή έχει τέσσερα συστατικά: το mapfile, μια HTML φόρμα αρχικοποίησης, ένα ή περισσότερα HTML template αρχεία, και μια χωρική βάση δεδομένων.

Ο MapServer, όπως όλες οι web εφαρμογές είναι βασισμένες σε ένα stateless πρωτόκολλο, πράγμα που σημαίνει ότι σε κάθε κλήση γνωρίζει μόνο ό,τι του έχει δηλώσει ο browser μόλις πριν. Η έλλειψη της προηγούμενης κατάστασης αποκλείει τη χρήση των εφαρμογών που πρέπει να κάνουν περισσότερα από το να απαντούν στην τελευταία ερώτηση που τους υποβλήθηκε. Εντούτοις, κάποια έξυπνη κωδικοποίηση μπορεί να προσφέρει ένα statefull server περιβάλλον και να δώσει στις web εφαρμογές τη δυνατότητα να εκτελέσουν πιο σύνθετες εργασίες.Παραδείγματος χάριν, η κατάσταση (state) μπορεί να διατηρηθεί μεταξύ των κλήσεων με την αποθήκευση των πληροφοριών κατάστασης σε κρυμμένες μεταβλητές, στο URL, ή σε cookies. Απαιτείται όμως κάποια μέθοδος για την προσαρμογή της εφαρμογής έτσι ώστε να έχει τις πληροφορίες που απαιτούνται στην πρώτη κλήση του Mapserver. Αυτό υλοποιείται από το αρχείο αρχικοποίησης. Σε μια CGI MapServer εφαρμογή, το αρχείο αρχικοποίησης είναι μια συμβατική HTML φόρμα με τις πληροφορίες έναρξης κωδικοποιημένες σε μεταβλητές μορφής. Σχεδόν οποιαδήποτε τιμή που χρησιμοποιεί ο MapServer μπορεί να μπει στο αρχείο έναρξης.

Όταν ο MapServer καλείται αρχικά από τον Apache μέσω της HTML φόρμας αρχικοποίησης, διαβάζει το κατάλληλο αρχείο mapfile (αρχείο αρχικοποίησης με μια επέκταση .map, που έχει φτιαχτεί προγραμματιστικά από τον deleloper του συστήματος) για να εντοπίσει τις γραμματοσειρές, τα σύμβολα, τα templates και τα χωρικά δεδομένα που θα χρησιμοποιήσει. Το mapfile διευκρινίζει επίσης το μέγεθος του προκύπτοντος χάρτη, τη γεωγραφική έκτασή του, και εάν είναι σε GIF, JPEG, ή PNG format. Έχοντας διαβάσει το mapfile, ο MapServer αποδίδει έπειτα μια ή περισσότερες εικόνες: τον ίδιο το χάρτη, τις λεζάντες και τη scalebar και ένα χάρτη αναφοράς (εφόσον έχει οριστεί προγραμματιστικά). Οι εικόνες αυτές αποθηκεύονται σε μια θέση (folder), η οποία διευκρινίζεται στο mapfile.

Προκειμένου να παρουσιάσει τα αποτελέσματά του, ο MapServer πρέπει να μορφοποιήσει το χάρτη και τα σχετικά elements σαν μια ιστοσελίδα. Το ίδιο το πρόγραμμα δεν δημιουργεί την HTML. Αυτό που κάνει είναι να ανιχνεύει ένα HTML template για *τα* κατάλληλα *substitution strings*. Τα substitution strings μπορούν να είναι αναφορές αρχείων, λεπτομέρειες της γεωμετρίας του χάρτη, προδιαγραφές των layers, ή παράγοντες ζουμ. Μπορούν επίσης να είναι οι τρέχουσες τιμές των CGI μεταβλητών όπως το μέγεθος εικόνας, το όνομα του mapfile, η έκταση του χάρτη, κλπ. Ο MapServer αντικαθιστά τα substitution strings με τις κατάλληλες τιμές και επιστρέφει την τροποποιημένη HTML σελίδα στον browser.

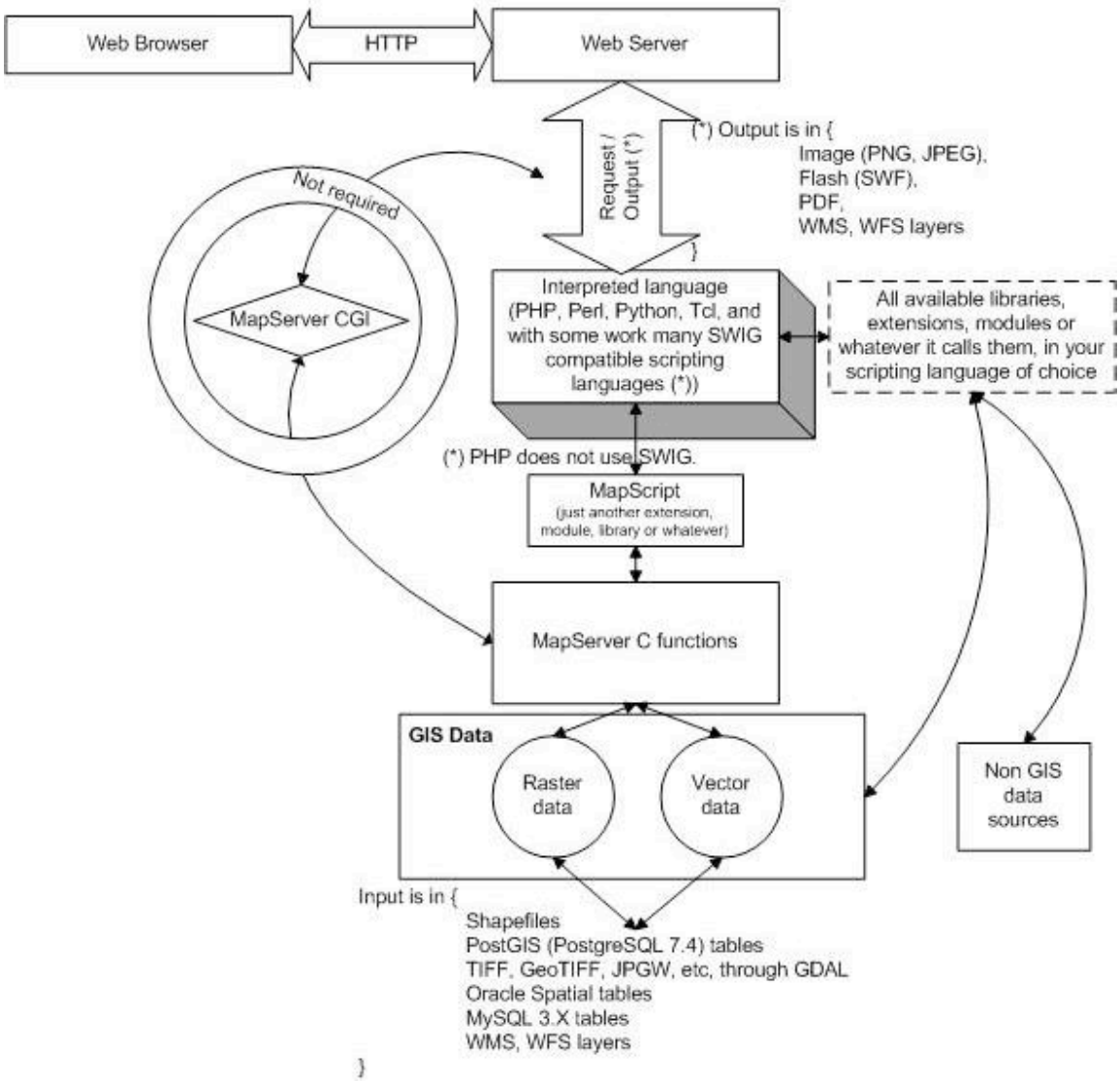
3.3.2. Mapscript

Όπως αναφέρθηκε παραπάνω ο CGI Mapserver μπορεί να κάνει αρκετά πράγματα αλλά έχει περιορισμένες δυνατότητες όσον αφορά τις απαντήσεις σε χωρικά ερωτήματα σε μια βάση δεδομένων. Υπάρχει, όμως και ένας άλλος πιο ισχυρός τρόπος χρήσης – κλήσης του Mapserver, η mapscript. Με τη mapscript δίνεται η δυνατότητα να πραγματοποιήσουμε πράγματα τα οποία δεν μπορούν να γίνουν μέσω του Mapserver CGI mode, όπως δυναμική δημιουργία layers, παραμετροποιήσιμη πλοήγηση, on the fly classification, προηγμένο έλεγχο των layers legends, καθώς και χωρικών αναζητήσεων. Η MapScript ενσωματώνει τη λειτουργικότητα του MapServer στις διάφορες scripting γλώσσες. Αυτό μειώνει το χρόνο προγραμματισμού για τους προγραμματιστές που θέλουν να προσθέσουν ικανότητες χαρτογράφησης σε μια εφαρμογή. Αντί να δημιουργεί μια εξειδικευμένη μέθοδο για τη χαρτογράφηση, το MapScript API παρέχει μερικά ισχυρά, έτοιμα προς χρήση εργαλεία. Παρέχει επίσης έναν κατάλληλο τρόπο αλληλεπίδρασης με τα δεδομένα χαρτογράφησης μέσω της επιθυμητής από το χρήστη γλώσσα προγραμματισμού.

Η MapScript επιτρέπει στους χρήστες να φορτώσουν, να χειριστούν, και να δημιουργήσουν αρχεία χαρτών. Παραδείγματος χάριν, μπορεί κάποιος να αλλάξει τα settings των layers, να χειριστεί τις κλάσεις των mapfiles, να παράγει εικόνες εξόδου, να εξαγάγει τα χωρικά δεδομένα, κλπ. Επειδή χρησιμοποιεί κοινά scripting περιβάλλοντα, οι λειτουργίες της MapScript μπορούν να συνδυαστούν με άλλες λειτουργίες της script γλώσσας προγραμματισμού που επιθυμούμε.

Η MapScript υποστηρίζεται από διάφορες γλώσσες. Στην παρούσα φάση στις διαθέσιμες γλώσσες συγκαταλέγονται η PHP, η Python, η Perl, η Java, η C#, η TCL και η Ruby.

Η αρχιτεκτονική της Mapscript φαίνεται στην **Εικόνα 18**.



Εικόνα 18: Αρχιτεκτονική της Mapscript

Η Mapscript στηρίζεται στο αντικειμενοστραφή προγραμματισμό. Έτσι το API της είναι μια συλλογή από κλάσεις με μεθόδους και χαρακτηριστικά (attributes). Με δεδομένο ότι κάθε γλώσσα προγραμματισμού έχει διαφορετική δομή και συντακτικό, το API της διαφοροποιείται κάπως, ανάλογα με τη προγραμματιστική γλώσσα που χρησιμοποιούμε. Έτσι για παράδειγμα υπάρχουν οι Perl Mapscript, Python Mapscript, Java, TCL, Ruby και η PHP Mapscript. Στην εργασία μας χρησιμοποιήθηκε η PHP Mapscript [10].

3.3.2.1 PHP/Mapscript

Ο πηγαίος κώδικας της PHP/Mapscript περιέχεται μέσα στη διανομή του Mapserver αλλά δεν εγκαθίσταται με τον mapserver παρά μόνο εάν υπάρξει τέτοια απαίτηση από το χρήστη.

3.4. Εγκατάσταση

Ο Mapserver μπορεί να εκτελείται σε μια πληθώρα από λειτουργικά συστήματα και ο τρόπος εγκατάστασής του ποικίλει ανάλογα με την αντίστοιχη πλατφόρμα [9].

Ο πιο εύκολος τρόπος προκειμένου να χρησιμοποιήσουμε τον Mapserver στα Windows είναι να κατεβάσουμε τα τυποποιημένα πακέτα για Windows τα οποία περιλαμβάνουν οτιδήποτε χρειαζόμαστε για τον Mapserver.[xiii]

Στη δικτυακή τοποθεσία http://maptools.org/ms4w/ υπάρχει το πιο εύκολο στη χρήση πακέτο εγκατάστασης, το MS4W (MapServer For Windows). Το βασικό πακέτο MS4W εγκαθιστά ένα προρυθμισμένο Web Server περιβάλλον που περιλαμβάνει τα ακόλουθα στοιχεία:

Apache HTTP Server

PHP

MapServer CGI

MapScript (CSharp, Java, PHP, Python)

Περιλαμβάνει υποστήριξη για την Oracle 10g, και SDE data

MrSID support built-in

GDAL/OGR και Utilities

MapServer Utilities

PROJ Utilities

Shapelib Utilities

Shp2tile Utility

Shpdiff Utility

AVCE00 Utilities

OGR/PHP Extension

OWTChart

DEMtools Utilities

Το πακέτο αυτό είναι ένα zip αρχείο το οποίο απλά αποσυμπιέζουμε στο root του σκληρού (C:\). Δημιουργείται έτσι ένα πλήθος υποφακέλων κάτω από το φάκελο C:\ms4w\, συμπεριλαμβανομένου του φακέλου Apache όπου βρίσκεται το αντίστοιχος

server. Στη συνέχεια εκτελούμε το αρχείο apache-install.bat. Ο Apache τρέχει ως service και ξεκινά σε κάθε εκκίνηση του υπολογιστή. Για να ελέγξουμε τη λειτουργία του server καλούμε στον browser http://localhost/ ή http://127.0.0.1/ οπότε εμφανίζεται η welcome screen του MS4W.

Προκειμένου να μπορέσουμε να εκτελέσουμε την php/mapscript θα πρέπει να τροποποιήσουμε το αρχείο php.ini ώστε να φορτώνεται η αντίστοιχη βιβλιοθήκη. Έτσι στο php.ini προσθέτουμε τη γραμμή extension=php_mapscript.dll. Στη συνέχεια εκτελούμε το αρχείο apache-restart.bat.

3.5. Στατικό Αρχείο Οπτικής Αρχικοποίησης

Το mapfile καθορίζει πώς θα φαίνεται ο χάρτης, ποιο θα είναι το μέγεθος του, και ποια layers θα εμφανίζονται. Υπάρχουν πολλές διαφορετικές προσεγγίσεις διαμόρφωσης του mapfile. Τα έγγραφα αναφοράς του mapfile που βρίσκονται στο web site του MapServer είναι μια απαραίτητη αναφορά για την εκμάθηση των νέων features. Μπορείτε να βρείτε τα έγγραφα αυτά στη διεύθυνση http://mapserver.gis.umn.edu/doc/mapfile-reference.html.

Τα mapfiles είναι απλά αρχεία κειμένου που χρησιμοποιούν μια ιεραρχική, ή εμφωλευμένη (nested), δομή αντικειμένων για να καθορίσουν τα χαρακτηριστικά για το χάρτη. Το mapfile αποτελείται από αντικείμενα. Κάθε αντικείμενο έχει μια λέξη κλειδί για να αρχίζει και τη λέξη END για να κλείνει. Τα αντικείμενα μπορούν να έχουν μέσα τους άλλα αντικείμενα, όπου αυτό χρειάζεται. Πιο συγκεκριμένα, ένα layer θα περιλαμβάνει αντικείμενα που καθορίζουν πως το αντικείμενο layer θα σχεδιαστεί, παραδείγματος χάριν πώς θα δημιουργούνται οι ετικέτες για αυτό το layer. Η **Εικόνα 19** δείχνει ένα απλό mapfile, που χρησιμοποιεί μόνο 15 γραμμές κειμένου.

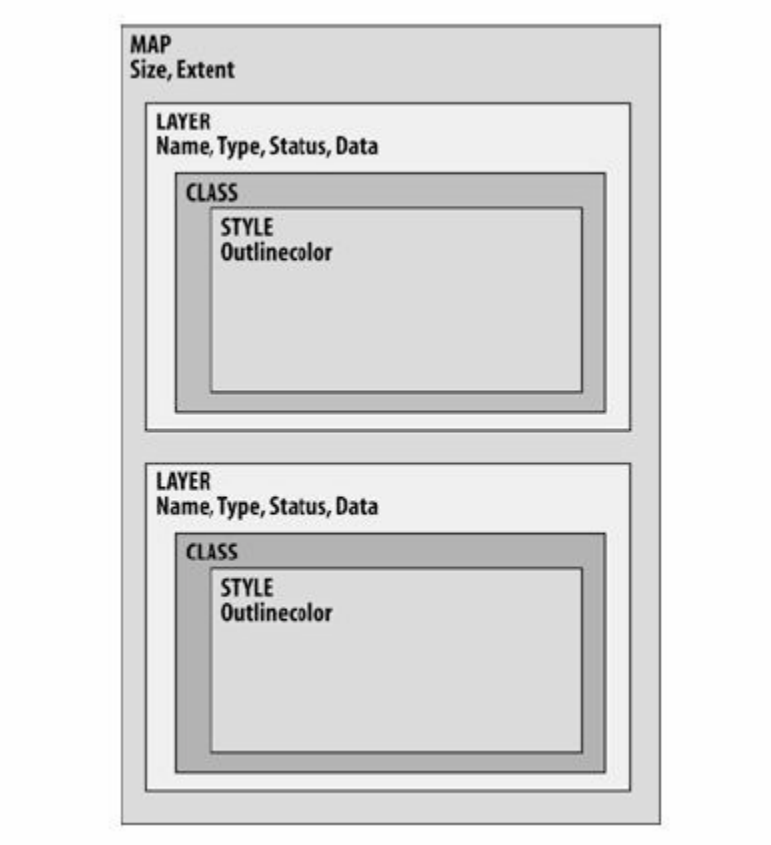
```
MAP                          # Start of MAP object
  SIZE 600 300
  EXTENT -180 -90 180 90
  LAYER                      # Start of LAYER object
    NAME countries
    TYPE POLYGON
    STATUS DEFAULT
    DATA countries_simpl
    CLASS                    # Start of CLASS object
      STYLE                  # Start of STYLE object
        OUTLINECOLOR 100 100 100
      END                    # End of STYLE object
    END                      # End of CLASS object
  END                        # End of LAYER object
END                          # End of MAP object and map file
```

Εικόνα 19: Ένα απλό mapfile με ένα layer

Υπάρχουν μόνο τέσσερα αντικείμενα σε αυτό το mapfile:

- το MAP αντικείμενο, που καθορίζει τα χαρακτηριστικά του χάρτη για όλο το αρχείο και μέσα στο οποίο περιλαμβάνονται όλα τα υπόλοιπα αντικείμενα
- το LAYER αντικείμενο, που καθορίζει το layer που θα σχεδιάσει στο χάρτη
- το CLASS αντικείμενο, που καθορίζει τις κλάσεις των χαρακτηριστικών για το layer
- το STYLE αντικείμενο, το οποίο καθορίζει πώς τα features θα σχεδιαστούν στην κλάση

Τα υπόλοιπα αντικείμενα, όπως και τα παραπάνω, θα αναλυθούν στη συνέχεια. Η **Εικόνα 20** δείχνει τη δομή ενός mapfile.



Εικόνα 20: Δομή ενός mapfile

Ακολουθεί ανάλυση των κυριότερων αντικειμένων και των features του κάθε αντικειμένου του mapfile.

3.5.1. Map Object

Το map object δεν ορίζεται απλά μέσα στο mapfile - *είναι* το mapfile [10]. Είναι ο πατέρας όλων των αντικειμένων του mapfile και καθορίζει τα χαρακτηριστικά της εφαρμογής που είναι καθολικά (global).

FONTSET [filename]

Καθορίζει το μονοπάτι (απόλυτο ή σχετικό με τη θέση του mapfile) στο αρχείο που ορίζει τη χαρτογράφηση από τα αρχεία aliases γραμματοσειρών στα αρχεία γραμματοσειρών True Type.

IMAGECOLOR [int r][int g][int b]

Καθορίζει το χρώμα υποβάθρου της εικόνας του χάρτη. Αυτό το χρώμα γίνεται διαφανές εάν το TRANSPARENT έχει επιλεγεί. Οι τιμές είναι ακέραιοι αριθμοί ενός byte στη σειρά 0 έως 255, αναπαριστώντας σχετικά ποσά κόκκινου, πράσινου, και μπλε.

IMAGETYPE [gif | png | jpeg | wbmp | gtiff | swf | userdefined]

Καθορίζει τη μορφή της εικόνας εξόδου. Ο τύπος εικόνας μπορεί να είναι ένας από τους OUTPUTFORMAT τύπους που αναγνωρίζονται από τον MapServer (που περιγράφονται αργότερα σ' αυτή την ενότητα) ή αυτό μπορεί να είναι ένας καθορισμένος από το χρήστη τύπος προσδιορισμένος από το όνομά του όπως ορίζεται από τη λέξη κλειδί NAME στην κατάλληλη OUTPUTFORMAT δήλωση.

LEGEND

Δείχνει την έναρξη ενός LEGEND αντικειμένου.

NAME [name]

Καθορίζει το όνομα που χρησιμοποιείται για να προσδιορίσει το χάρτη εξόδου. Ένας αριθμός ID (που παράγεται από σύνδεση της ταυτότητας χρόνου και της διαδικασίας ID) επισυνάπτεται σε αυτό το όνομα για να παρέχει μια μοναδική ID ταυτότητα.

SHAPEPATH [path]

Καθορίζει το directory στο οποίο είναι αποθηκευμένα τα shapefiles. Η τιμή που ορίζεται στο SHAPEPATH προτάσσεται στο σύνολο των δεδομένων που καθορίζεται από το keyword του layer DATA.

SIZE [int x][int y]

Καθορίζει το πλάτος και το ύψος της εικόνας του χάρτη σε pixels.

STATUS [on | off]

Default: on

Καθορίζει εάν η εικόνα του χάρτη δημιουργείται.

UNITS [feet | inches | kilometers | meters | miles | dd]

Default: n/a

Καθορίζει τις μονάδες μέτρησης των αποστάσεων στο χάρτη. Αυτή η λέξη κλειδί επηρεάζει τους υπολογισμούς της κλίμακας και τις scale bars. Το dd δείχνει τους δεκαδικούς βαθμούς.

EXTENT [minx][miny][maxx][maxy]

Default: n/a

Καθορίζει την έκταση (extent) του χάρτη. Το extent toy χάρτη καθορίζεται από τις συντεταγμένες του κάτω αριστερά σημείου του χάρτη (minx, miny) και του πάνω δεξιά σημείου (maxx, maxy). Λάθη στον καθορισμό του extent μπορούν να οδηγήσουν σε

κενούς χάρτες ή σε διαστρεβλωμένους χάρτες. Το extent πρέπει να καθοριστεί, και πρέπει να είναι στο ίδιο σύστημα συντεταγμένων με το αντικείμενο PROJECTION του χάρτη (εάν υπάρχει). Υπάρχει η δυνατότητα να καθοριστεί από τον χρήστη και αν αυτό δεν γίνει τότε θα ορίσει κάποιο ο mapserver.

3.5.2. Layer Object

Τα στοιχεία του LAYER αντικειμένου καθορίζουν τα χωρικά δεδομένα που πρόκειται να μεταδοθούν και πώς αυτά πρόκειται να ταξινομηθούν. Αρχίζει με τη λέξη κλειδί LAYER και ολοκληρώνεται με τη λέξη κλειδί END. Τα layers σχεδιάζονται στη σειρά που βρίσκονται στο mapfile, και τα επόμενα στρώματα δίνονται πάνω από εκείνα που αποδόθηκαν πρώτα.

CLASSITEM [attribute]

Default: none

Καθορίζει το όνομα των attributes που χρησιμοποιούνται για να ταξινομήσουν τα features όταν η λέξη κλειδί EXPRESSION χρησιμοποιείται για να εκτελέσει κανονικές συγκρίσεις ή συγκρίσεις σε strings

LABELITEM [attribute]

Default: n/a

Καθορίζει το όνομα των attributes των οποίων η τιμή χρησιμοποιείται για να ονομάσει ένα feature.

NAME [string]

Default: none

Καθορίζει το όνομα του layer (μέγιστο 20 χαρακτήρες). Αυτό το όνομα χρησιμοποιείται ως τιμή του layer της φόρμας της CGI μεταβλητής για να επιτρέψει στο layer να είναι on και off αμφίδρομα.

SIZEUNITS [pixels | feet | inches | kilometers | meters | miles]

Default: pixels

Θέτει τις μονάδες του CLASS αντικειμένου SIZE, το οποίο καθορίζει το μέγεθος των συμβόλων που χρησιμοποιούνται για να για να σχεδιάσουν τα features.

STATUS [on | off | default]

Default: n/a

Καθορίζει εάν ένα layer θα εμφανίζεται και εάν μπορεί να επιλεχτεί στη θέση on ή στη θέση off. Η επιλογή on καθορίζει ότι ένα layer θα εμφανίζεται πάντα, η επιλογή default καθορίζει ότι ένα layer θα εμφανίζεται, αλλά μπορούμε κάποια στιγμή να επιλέξουμε να μην εμφανίζεται ενώ η επιλογή off καθορίζει ότι ένα layer δεν θα εμφανίζεται, αλλά μπορούμε και κάποια στιγμή να επιλέξουμε να εμφανίζεται.

3.5.3. Class Object

Το CLASS αντικείμενο καθορίζει τις ιδιότητες εμφάνισης των features. Κάθε layer πρέπει να περιέχει μια ή περισσότερες κλάσεις. Ένα CLASS αντικείμενο εισάγεται με τη λέξη κλειδί CLASS και ολοκληρώνεται με τη λέξη κλειδί END.

COLOR [int r][int g][int b]

Default: 0 0 0

Καθορίζει το χρώμα που θα χρησιμοποιηθεί για να δώσει στα features. Οι τιμές είναι ακέραιοι αριθμοί 1-byte από 0 έως 255, αντιπροσωπεύοντας σχετικά ποσά κόκκινου, πράσινου, και μπλε.

EXPRESSION [string]

Default: n/a

Καθορίζει την έκφραση που αξιολογεί για να καθορίσει εάν ένα feature πρέπει να περιληφθεί στην κλάση. Η τιμή που ορίζεται στο EXPRESSION αντιπροσωπεύει έναν από τους ακόλουθους τύπους εκφράσεων: μια σύγκριση string, μια κανονική έκφραση, μια λογική έκφραση, ή μια string λειτουργία. Μια σύγκριση string ταιριάζει με το περιεχόμενο των attributes που προσδιορίζονται από τη λέξη κλειδί CLASSITEM με ένα string. Η σύγκριση αυτή είναι case sensitive. Αν οι δύο εκφράσεις ταιριάζουν, το feature συμπεριλαμβάνεται στην κλάση. Εάν το string περιέχει κενά ή τους ειδικούς χαρακτήρες (όπως tabs ή τις νέες γραμμές), πρέπει να μπει σε αποσιωπητικά. Μια κανονική έκφραση ταιριάζει με το περιεχόμενο του attribute που προσδιορίζεται από τη λέξη κλειδί CLASSITEM με την κανονική έκφραση που διευκρινίζεται από το string. Η κανονική έκφραση πρέπει να οριοθετηθεί από κάθετους, (π.χ., / κανονική έκφραση/). Μια λογική έκφραση αποτελείται από παρενθέσεις, στις οποίες υπάρχει ένας συνδυασμός ονομάτων των attributes (που οριοθετούνται από αγκύλες), οι Boolean operators AND και OR, οι αριθμητικοί operators σύγκρισης =, >, <, <=, >=, και !=, οι operators σύγκρισης string eq. lt, gt, le, ge, eq, and ne και τιμές σύγκρισης. Παραδείγματος χάριν, EXPRESSION ([area]>100) θα επιλέξει τα features με area *μεγαλύτερη* από 100, και EXPRESSION ([status]!=0) θα επιλέξει τα αρχεία με τη

κατάσταση *μη ίση* με 0. Τα attributes, των οποίων η τιμή είναι string και οι τιμές σύγκρισης πρέπει να μπουν σε αποσιωπητικά. Παραδείγματος χάριν, η EXPRESSION ('[type]' ne 'river') θα επιλέξει μόνο εκείνα τα features με type μη ίσο με river. Αυτήν την περίοδο, η μόνη string λειτουργία που υποστηρίζεται από τον MapServer είναι το length(), το οποίο υπολογίζει το μήκος ενός attribute, του οποίου η τιμή είναι string. Η EXPRESSION (length('[name]") < 2) επιλέγει τα features με σύντομα ονόματα.

MAXSIZE [int N]

Default: 50

Καθορίζει το μέγιστο μέγεθος (σε pixels) στο οποίο ένα σύμβολο θα σχεδιασθεί.

MINSCALE [double N]

Default: n/a

Καθορίζει την ελάχιστη κλίμακα.

MINSIZE [int N]

Default: 0

Καθορίζει το ελάχιστο μέγεθος (σε pixels) στο οποίο ένα σύμβολο θα σχεδιασθεί.

NAME [string]

Default: n/a

Καθορίζει το όνομα για την κλάση για χρήση στη λεζάντα. Εάν ένα όνομα δεν καθορίζεται, το feature θα σχεδιασθεί, αλλά δεν θα φαίνεται στη λεζάντα.

OUTLINECOLOR [int r][int g][int b]

Default: n/a

Καθορίζει το χρώμα περιγράμματος (για τα πολύγωνα μόνο). Οι τιμές είναι ακέραιοι αριθμοί ενός byte από 0 έως 255, αντιπροσωπεύοντας σχετικά ποσά κόκκινου, πράσινου, και μπλε.

MAXSIZE [int N]

Default: 50

Καθορίζει το μέγιστο μέγεθος (σε pixels) στο οποίο ένα σύμβολο θα σχεδιασθεί.

MINSIZE [int N]

Default: 0

Διευκρινίζει το ελάχιστο μέγεθος (σε pixels) στο οποίο ένα σύμβολο θα σχεδιασθεί.

SIZE [int N]

Καθορίζει το ύψος ενός συμβόλου (σε pixels).

3.5.4. Outputformat Object

Το OUTPUTFORMAT αντικείμενο χρησιμοποιείται για να καθορίσει το format του τελικού αρχείου που θα δώσει ο Mapserver. Αρχίζει με τη λέξη κλειδί OUTPUTFORMAT και ολοκληρώνεται με τη λέξη κλειδί END. Επεκτείνει την έννοια του format της εικόνας (όπως το GIF ή JPEG) για να περιλάβει λεπτομέρειες για τη δομή της εικόνας, του χρώματος και ποιοι drivers γραφικών πρέπει να χρησιμοποιηθούν για να παραχθεί μια εικόνα.

DRIVER ["name"]

Default: n/a

Καθορίζει το όνομα του driver γραφικών που χρησιμοποιείται για να παράγει την εικόνα. Οι παρακάτω GD drivers υποστηρίζονται: "GD/Gif", "GD/PNG", "GD/WBMP", and "GD/JPEG". Το όνομα του flash driver είναι "SWF". Για την έξοδο που υποστηρίζεται από το GDAL, το όνομα αποτελείται από ένα string "GDAL" συνοδευόμενο με το GDAL shortname για το format του αρχείου εξόδου, π.χ. "GDAL/GTiff".

MIMETYPE [mimetype]

Default: n/a

Καθορίζει τον τύπο mime που χρησιμοποιείται για το αποτέλεσμα.

NAME [name]

Default: n/a

Καθορίζει το όνομα που χρησιμοποιείται από τη λέξη κλειδί του mapfile IMAGETYPE για να παραπέμψει στο format που θα αποδώσει ο mapserver.

Label Object

Το Label αντικείμενο καθορίζει ένα κείμενο ή ένα σύμβολο που χρησιμοποιείται για να ονομάσει ένα feature. Αρχίζει με τη λέξη κλειδί LABEL και ολοκληρώνεται με τη λέξη κλειδί END.

ANGLE [auto | double N]

Default: n/a

Καθορίζει τη γωνία σχεδίασης μια ετικέτας. Μια γωνία 0 θα προκαλέσει τη σχεδίαση μιας ετικέτας παράλληλης με το κάτω μέρος του χάρτη. Για τα layers γραμμών μόνο, η τιμή auto προκαλεί τη σχεδίαση μιας ετικέτας ευθυγραμμισμένης με το feature.

COLOR [int r][int g][int b]

Default: 0 0 0

Καθορίζει το χρώμα που θα χρησιμοποιηθεί για να αποδώσει το κείμενο των ετικετών. Οι τιμές είναι ακέραιοι αριθμοί ενός byte από 0 έως 255, αντιπροσωπεύοντας τα σχετικά ποσά κόκκινου, πράσινου, και μπλε.

FONT [name]

Default: none

Διευκρινίζει το ψευδώνυμο της γραμματοσειράς που χρησιμοποιείται για να ονομάσει ένα feature. Ο MapServer μεταφράζει το ψευδώνυμο στο μονοπάτι που βρίσκεται στο αρχείο που καθορίζεται από τη λέξη κλειδί FONTSET.

OUTLINECOLOR [int r][int g][int b]

Default: n/a

Καθορίζει το χρώμα που χρησιμοποιείται για να δημιουργήσει ένα περίγραμμα εύρους 1 pixel γύρω από το κείμενο της ετικέτας. Οι τιμές είναι ακέραιοι αριθμοί ενός byte από 0 έως 255, αντιπροσωπεύοντας τα σχετικά ποσά κόκκινου, πράσινου, και μπλε.

SIZE [int N] | [tiny | small | medium | large | giant]

Default: n/a

Καθορίζει το μέγεθος του κειμένου της ετικέτας. Το μέγεθος των ετικετών True Type καθορίζεται σε pixels με μια ακέραια τιμή.Το μέγεθος μιας bitmapped ετικέτας ορίζεται με μια τιμή tiny, small, medium, large ή giant.

Ο mapserver εκτός από ετικέτες μπορεί να σχεδιάσει με τον ίδιο τρόπο λεζάντες, χάρτες αναφοράς, scalebars κτλ. Για κάθε ένα από τα παραπάνω που θα χρειαστεί να σχεδιάσει ο mapserver υπάρχουν και ξεχωριστά objects, π.χ. Legend Object, Reference Map Object, Scalebar Object, με παρόμοια λειτουργία με αυτή του Label object και παρόμοια features με κάποιες μικρές διαφοροποιήσεις. Οπότε κρίνεται σκόπιμο να μην γίνει εκτενέστερη αναφορά σε αυτά.

3.5.5. WEB Object

Το WEB αντικείμενο καθορίζει τη web διεπαφή, συμπεριλαμβανομένων των μονοπατιών, των URLs, των template αρχείων, και άλλες λεπτομέρειες που επηρεάζουν τον τρόπο που η εφαρμογή αποκρίνεται στο χρήστη. Οι ιδιότητες ενός Web αντικειμένου είναι οι ακόλουθες:

HEADER [filename]

Default: n/a

Καθορίζει το όνομα του HTML template αρχείου που εμφανίζεται πριν την παρουσίαση οποιοδήποτε αποτελεσμάτων ερωτήσεων. Χρησιμοποιείται για τρόπους ερώτησης που δίνουν πολλαπλά αποτελέσματα.

IMAGEPATH [path]

Default: n/a

Καθορίζει το μονοπάτι στο directory όπου οι εικόνες και τα προσωρινά αρχεία αποθηκεύονται. Το μονοπάτι πρέπει να τελειώνει με ένα / ή \ (ανάλογα με την πλατφόρμα).

IMAGEURL [path]

Default: n/a

Καθορίζει το URL που δείχνει στο directory που οι εικόνες αποθηκεύονται. Ο browser χρησιμοποιεί αυτό το μονοπάτι για να ανακτήσει τις εικόνες.

TEMPLATE [filename | url]

Default: n/a

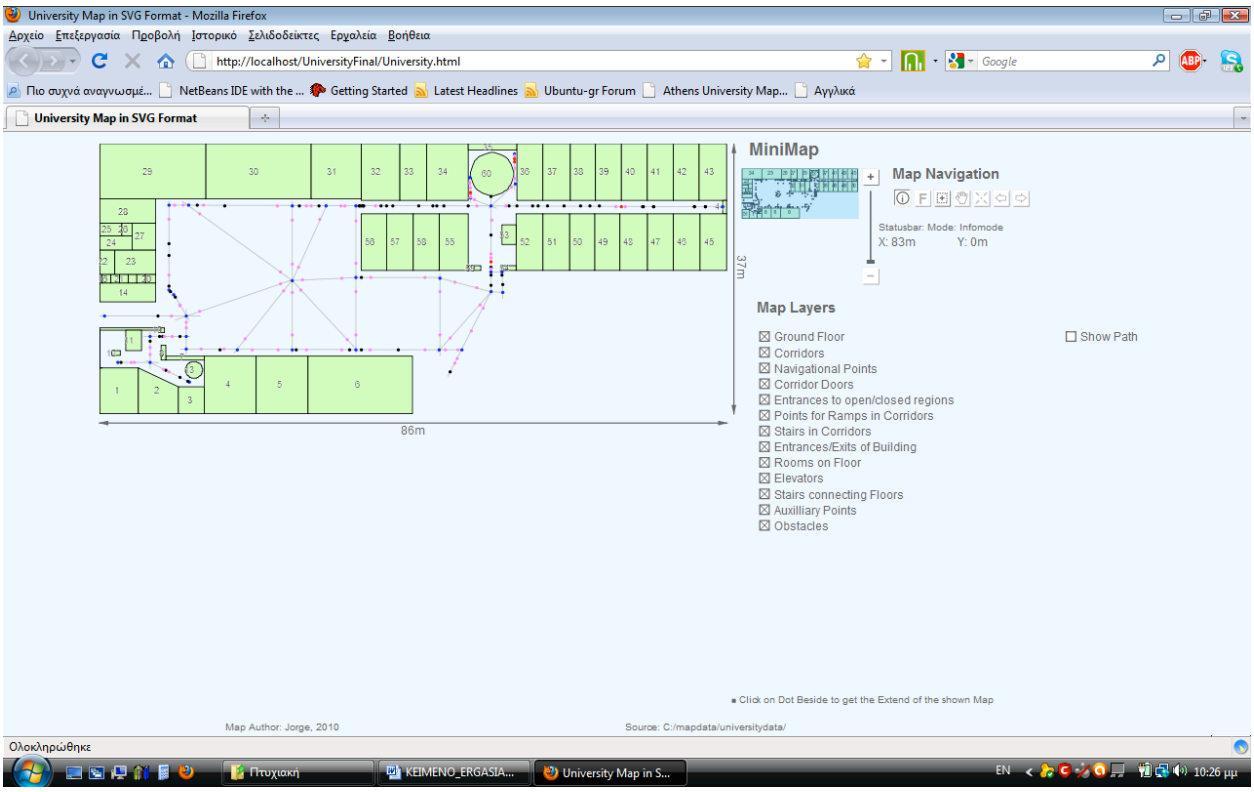
Καθορίζει το κύριο HTML template αρχείο που χρησιμοποιείται για να εμφανίσει ένα χάρτη με αμφίδρομο τρόπο.

ΚΕΦΑΛΑΙΟ 4

ΠΕΡΙΓΡΑΦΗ ΕΦΑΡΜΟΓΗΣ

4.1. Γενικά

Η υλοποίηση της απεικόνισης των χαρτών που δημιουργήσαμε στηρίζεται στο μοντέλο server-client και κάνει χρήση του mapserver, ο οποίος, για την λειτουργία του ως web browser, χρησιμοποιεί τον Apache server. Η εφαρμογή αποτελείται από δύο κύρια τμήματα. Το ένα υλοποιείται στον server και το άλλο στον client. Στην πλευρά του server αναπτύχθηκε κώδικας σε php/mapscript ο οποίος σε συνδυασμό με χρήση ενός αρχείου mapfile δημιουργεί τις απαιτούμενες διανυσματικές εικόνες χαρτών τύπου SVG που τελικά αποστέλλονται στον χρήστη. Χρησιμοποιήθηκε ως χάρτης αναφοράς η θεωρητική κάτοψη ενός κτηρίου διαστάσεων 86m x 37m. Η πλευρά του client ουσιαστικά αποτελεί τη διεπαφή (interface) του χρήστη με τον server. Δηλαδή είναι το κομμάτι της εφαρμογής όπου γίνεται η απεικόνιση των χαρτών που έχει αποστείλει ο server και περιλαμβάνει και το διαδραστικό κομμάτι με το χρήστη (π.χ. επιλογή zoom, pan, layers, κλπ). Επισημαίνεται ότι το υπόψη interface υλοποιήθηκε με χρήση του προτύπου SVG και δεν είναι απλά μια HTML σελίδα. Η επικοινωνία του server με τον client γίνεται μέσω ασύγχρονων κλήσεων (τεχνολογία Asynchronous Javascript and XML – AJAX) γεγονός που επιτρέπει την παρουσίαση των αποτελεσμάτων των αιτημάτων απεικόνισης χαρτών με έναν πολύ αποτελεσματικό και εργονομικό τρόπο. Η τεχνολογία AJAX λόγω των πλεονεκτημάτων που προσφέρει χρησιμοποιείται από πολύ γνωστές διαδικτυακές εφαρμογές με πιο χαρακτηριστική αυτή του Gmail (http://gmail.com). Η διεπαφή της εφαρμογής φαίνεται στην **Εικόνα 21**.



Εικόνα 21: Διεπαφή εφαρμογής

Γενικά για την υλοποίηση της εφαρμογής απαιτήθηκε πέραν από τη μελέτη του τρόπου λειτουργίας του mapserver, η εξοικείωση και χρήση των διαδικτυακών τεχνολογιών PHP, mapscript, javascript, AJAX, XML, DOM).[xiv],[xv],[xvi],[xvii],[xviii],[xix]

4.2. Κώδικας Εφαρμογής στον Εξυπηρετητή (Server)

Το κομμάτι της εφαρμογής που υλοποιείται στον server αφορά στη δημιουργία των αρχείων «UniversityFinal.map» και «getUniversityMap.php». Το πρώτο είναι το αρχείο mapfile που σχετίζεται με τη μορφή απεικόνισης των χαρτών, ενώ το δεύτερο δημιουργεί, κάνοντας χρήση php/mapscript, το χάρτη που έχει αιτηθεί ο χρήστης. Επισημαίνεται ότι θα μπορούσαμε να φτιάξουμε το ζητούμενο χάρτη χωρίς τη χρήση του mapfile αρχείου χρησιμοποιώντας μόνο php/mapscript κώδικα, αλλά προτιμήθηκε ο διαχωρισμός του στατικού κομματιού που σχετίζεται με τον τρόπο απεικόνισης των χαρτών και του δυναμικού που έχει να κάνει με το τι θα απεικονιστεί.

Στις περιπτώσεις που δημιουργούμε ένα map αντικείμενο με τη mapscript χρησιμοποιώντας (διαβάζοντας) ένα mapfile, όλες οι παράμετροι του χάρτη, τα επίπεδα (layers), οι κλάσεις (classes) και οι ιδιότητες (attributes) του mapfile μεταφράζονται σε mapscript αντικείμενα. Για τις ιδιότητες που δεν έχουν προσδιοριστεί επιλέγονται αυτόματα οι εξ ορισμού τιμές και εφόσον στο mapfile προσδιορίζεται ο τύπος του χάρτη (SVG, jpg, κλπ) η mapscript θα παράγει τον τύπο αυτό. Επιπρόσθετα, βέβαια, όλα τα

αντικείμενα της mapscript μπορούν να τροποποιηθούν προγραμματιστικά, κάτι που δε μπορεί να γίνει στο mapfile. Με τη χρήση του αρχείου mapfile, μπορούμε να επικεντρωθούμε στις δυνατότητες που παρέχουν οι συναρτήσεις της mapscript για τη δυναμική δημιουργία του χάρτη, χωρίς να ασχολούμαστε με τις λεπτομέρειες του τρόπου απεικόνισης, καθώς αυτές καθορίζονται στο mapfile.

4.2.1. Τρόπος Προβολής Χάρτη

Για να αναπαρασταθεί με πιστότητα η πραγματική μορφή της επιφάνειας της γης στο χαρτί απαιτούνται σύνθετοι μαθηματικοί υπολογισμοί. Ακόμα και αν η γη ήταν μια τέλεια σφαίρα, η προβολή της επιφάνειάς της σε ένα επίπεδο δεν θα ήταν απλή υπόθεση. Ωστόσο, η επιφάνεια της γης είναι ακανόνιστη και δεν μπορεί να παρασταθεί από κάποιο γεωμετρικό σχήμα. Έτσι, για τους μαθηματικούς υπολογισμούς που απαιτούνται στη δημιουργία των υποβάθρων των χαρτών χρησιμοποιείται ένα θεωρητικό (γεωμετρικό) σχήμα, το ελλειψοειδές (το σχήμα που προκύπτει από την περιστροφή μιας έλλειψης γύρω από τον άξονα των πόλων). Τα γεωμετρικά στοιχεία του ελλειψοειδούς επιλέγονται έτσι ώστε η επιφάνεια που προκύπτει να προσεγγίζει το γεωειδές, δηλαδή μια επίσης θεωρητική επιφάνεια που ταυτίζεται με το μέσο επίπεδο της θαλάσσιας επιφάνειας και τη θεωρητική προέκτασή της κάτω από τις ηπείρους και η οποία προσδιορίζεται με μετρήσεις του πεδίου βαρύτητας. Το γεωειδές είναι μια "πραγματική" επιφάνεια αναφοράς, στο βαθμό που προσεγγίζει αδρά τη μορφή της επιφάνειας της γης.

Επειδή η μετατροπή μιας σφαιρικής επιφάνειας (όπως η γη) σε επίπεδη δεν μπορεί να γίνει χωρίς παραμορφώσεις, έχουν δημιουργηθεί διάφορα συστήματα προβολής, από τα οποία, άλλα διατηρούν τις αναλογίες των εμβαδών, άλλα των μηκών και άλλα τις γωνίες ανάμεσα στο χάρτη και το πεδίο. Το προβολικό σύστημα το οποίο έχει καθιερωθεί και χρησιμοποιείται διεθνώς (εκτός από τις πολικές περιοχές) είναι η Παγκόσμια Εγκάρσια Μερκατορική των 6 μοιρών, πιο γνωστή ως UTM (Universal Transverse Mercator).

Ένα γεωδετικό σύστημα αναφοράς ορίζεται με την επιλογή ενός ελλειψοειδούς (που προσεγγίζει όσο το δυνατόν καλύτερα το γεωειδές) και ενός προβολικού συστήματος για την απεικόνιση της επιφάνειας του ελλειψοειδούς στο επίπεδο. Ο mapserver έχει τη δυνατότητα να χρησιμοποιήσει διάφορα γεωδαιτικά συστήματα αναφοράς (π.χ. ED50, WG84 κλπ).

Με δεδομένο ότι στην εργασία μας αναφερόμαστε σε μικρής έκτασης χάρτη (χάρτης κτηρίου) είναι σαφές ότι δεν έχει νόημα η χρήση γεωδαιτικών συστημάτων. Από την

άλλη βέβαια είναι απαραίτητο να υπάρχει ένα σύστημα συντεταγμένων μοναδικά ορισμένο προκειμένου να δίνεται η δυνατότητα πλοήγησης στο χάρτη. Για το σκοπό αυτό ορίστηκε ένα σύστημα συντεταγμένων (x,y) με αρχή την κάτω αριστερή γωνία του χάρτη. Όλα τα άλλα σημεία είναι ορισμένα με βάση την απόσταση σε μέτρα από την αρχή των αξόνων. Οι αποστάσεις αυξάνονται κινούμενοι προς τα πάνω και δεξιά. Έτσι το πιο απομακρυσμένο σημείο του χάρτη είναι η πάνω δεξιά γωνία του η οποία έχει συντεταγμένες (86,37). Επισημαίνεται ότι οι αποστάσεις αυτές είναι πραγματικές με βάση τα σχέδια της κάτοψης του κτηρίου. Με βάση τα παραπάνω, όταν αναφερόμαστε σε πραγματικές συντεταγμένες θα εννοούμε τις συντεταγμένες σε μέτρα όπως προκύπτουν με βάση την ανωτέρω ανάλυση. Αντίθετα οι συντεταγμένες εικόνας αναφέρονται στα pixels της εικόνας. Επισημαίνεται ότι για να μην υπάρχουν παραμορφώσεις της εικόνας και προβλήματα απεικόνισης θα πρέπει να διατηρείται η αναλογία ανάμεσα στις δύο διαστάσεις μεταξύ των πραγματικών συντεταγμένων και των συντεταγμένων εικόνας. Έτσι αφού οι διαστάσεις του κτηρίου είναι 86m μήκος και 37m πλάτος (λόγος 86/37=2,32), επιλέγοντας 800 pixels για μήκος εικόνας προκύπτει 344 pixels για το πλάτος. Για αυτό και στο mapfile έχει οριστεί ως μέγεθος της εικόνας το 800x344.

Για τη δημιουργία των διανυσματικών δεδομένων του χάρτη μπορούμε να χρησιμοποιήσουμε πραγματικές κατόψεις κτηρίων (από αντίστοιχα αρχιτεκτονικά σχέδια) από τις οποίες μέσω σάρωσης θα παραχθούν αντίστοιχες εικόνες raster (π.χ. jpeg). Με αυτό τον τρόπο μπορούμε να έχουμε την πραγματική κάτοψη ενός κτηρίου. Στη συνέχεια, έχοντας ως βάση την jpeg εικόνα, μπορούμε, κάνοντας χρήση ενός σχεδιαστικού προγράμματος (π.χ. Autocad) να σχηματίσουμε τα βασικά σχήματα της κάτοψης (αίθουσες, γραφεία, φρεάτια, διαδρόμους κλπ) σε δυανυσματική μορφή (dxf αρχεία). Επισημαίνεται ότι κατά την ανωτέρω διαδικασία θα πρέπει να ληφθούν υπόψη – ορισθούν οι πραγματικές διαστάσεις των χώρων (αιθουσών, γραφείων, διαδρόμων, κλπ) με βάση την κάτοψη. Στη συνέχεια στη δυανυσματική εικόνα που έχουμε δημιουργήσει (dxf αρχείο) ορίζουμε ένα σύστημα συντεταγμένων. Στη συγκεκριμένη υλοποίηση ορίσαμε ένα σύστημα συντεταγμένων με αρχή αξόνων (σημείο 0,0) την κάτω αριστερή γωνία του σχεδίου. Στην εφαρμογή μας λόγω του ότι δεν διαθέτουμε πραγματικά δεδομένα (κάτοψη κτηρίου) απλά σχεδιάστηκαν τα βασικά σχήματα στο autocad και ορίστηκε το συγκεκριμένο σύστημα συντεταγμένων. Έπειτα χρησιμοποιώντας το πρόγραμμα CAD2Shape, το οποίο διατίθεται δωρεάν ως demo, μετατρέψαμε το dxf αρχείο σε shapefile. Ο μετέπειτα χειρισμός των shapefiles έγινε με το δωρεάν πρόγραμμα Quantum GIS (http://www.qgis.org). Επειδή οι συντεταγμένες

και οι διαστάσεις είχαν επακριβώς οριστεί στο autocad με ακρίβεια, τα αντίστοιχα shapefiles αναγνωρίστηκαν με το ίδιο σύστημα συντεταγμένων και τις ίδιες διαστάσεις. Μέσω του Quantum GIS υπάρχει η δυνατότητα ορισμού attributes στα shapefiles δημιουργίας επιπλέον layers και γενικότερα περαιτέρω ανάπτυξης διανυσματικών χαρτών μορφής shapefile. Έχοντας καλά ορισμένο από πλευράς διαστάσεων το βασικό σχήμα (layer κάτοψης) το οποίο ονομάσαμε Ground_shp, μπορέσαμε να δημιουργήσουμε επιπλέον layers, τύπου σημείου (με εξαίρεση το layer grCor το οποίο είναι τύπου γραμμής). Σημειώνεται ότι κατά τη δημιουργία του layer της κάτοψης ορίστηκε ότι αυτό είναι τύπου πολυγώνου. Τα layers αυτά έχουν συγκεκριμένα attributes με πιο σημαντικά τα X, Y που παίρνουν ως τιμές τις πραγματικές συντεταγμένες του σημείου, το FLOOR που αναφέρεται στον όροφο του κτηρίου που βρίσκεται το συγκεκριμένο σημείο, το KIND που σχετίζεται με το είδος του σημείου (π.χ. Turn point, junction, end point) και το id του σημείου το οποίο είναι ένας αύξων αριθμός μοναδικός για κάθε σημείο. Το σύνολο των layers όπως τελικά ορίστηκαν φαίνονται στον **Πίνακας 4** παρακάτω:

Όνομα layer	Περιγραφή	Είδος	ID
Ground_shp	Βασικό σχήμα κάτοψης	Polygon	-
grCor	Διάδρομοι	Line	-
grAll	Navigational Point	Point	1 - 33
grCD	Πόρτες Διαδρόμων	Point	34 – 36
grCOA	Είσοδοι σε κλειστές ή ανοικτές περιοχές	Point	37 – 43
grCR	Σημεία για ράμπες στους διαδρόμους	Point	44 - 47
grCS	Σκάλες σε διαδρόμους	Point	48 – 51
grEE	Είσοδοι – Έξοδοι κτηρίου	Point	52 – 54
grRo	Δωμάτια Ορόφου	Point	55 – 91
grEl	Ασανσέρ	Point	92
grS	Σκάλες που συνδέουν ορόφους	Point	93 – 96

grale	Βοηθητικά σημεία	Point	140 – 222
grObP	Εμπόδια	Point	223 - 227

Πίνακας 4: Layers Χάρτη

Αναλυτικά τα attributes του κάθε layer μπορεί κάποιος να τα δει κάποιος είτε μέσω του Quantum GIS είτε ανοίγοντας με το Excel το αρχείο της μορφής «όνομα».dbf, όπου «όνομα» το όνομα του αντίστοιχου layer. Επισημαίνεται ότι τα αρχεία αυτά αποτελούν τμήμα των shapefiles τα οποία διαβάζει ο mapserver προκειμένου να σχηματίσει τους αντίστοιχους χάρτες. Το σύνολο των shapefiles παραδίδεται μαζί με την παρούσα πτυχιακή σε ηλεκτρονική μορφή.

Η λογική των επιπλέον αυτών layers φαίνεται στην αντίστοιχη περιγραφή τους του Πίνακα 3. Για παράδειγμα το grCor που είναι τύπου Line περιλαμβάνει τους διαδρόμους πάνω στους οποίου μπορεί κάποιος να κινηθεί ενώ το grRo είναι τα σημεία εισόδου στα δωμάτια του ορόφου. Χρησιμοποιώντας αυτά τα σημεία και τους διαδρόμους ένα αλγόριθμος δρομολόγησης μπορεί να υπολογίσει μια συγκεκριμένη ζητούμενη διαδρομή η οποία μπορεί δυναμικά να αποτυπωθεί στο χάρτη. Στην παρούσα εφαρμογή δεν έχει υλοποιηθεί ο αλγόριθμος δρομολόγησης αλλά για λόγους επίδειξης χρησιμοποιούμε ένα txt αρχείο με όνομα path στο οποίο περιλαμβάνονται σημεία διαδρομής τα οποία διαβάζονται από την εφαρμογή και τελικά αποτυπώνονται στο χάρτη. Το αρχείο path θα μπορούσε να είναι το αποτέλεσμα της εξόδου του αλγορίθμου δρομολόγησης.

4.2.2. Επεξήγηση Κώδικα Στατικού Αρχείου Mapfile

Το mapfile αρχείο που δημιουργήθηκε για τις ανάγκες της εφαρμογής είναι το «UniversityFinal.map». Πριν ξεκινήσουμε την ανάλυση του κώδικα του UniversityFinal.map κρίνεται σκόπιμο να αναφερθεί ότι οι λέξεις κλειδιά του mapfile, που έχουν αναπτυχθεί στο προηγούμενο κεφάλαιο, και οι τιμές τους δεν είναι case sensitive. Σ’ αυτή την εργασία όμως επιλέξαμε για λόγους σαφήνειας να γράφουμε τις λέξεις κλειδιά με κεφαλαία και τις τιμές τους με μικρά. Πρέπει όμως να τονιστεί ότι η κατάσταση των γραμμάτων είναι σημαντική κατά την αλληλεπίδραση με χωρικές βάσεις δεδομένων.

Προχωρώντας με την ανάλυση του κώδικα, στη γραμμή 1 η λέξη κλειδί NAME ορίζει ότι η βάση για όλες τις εικόνες που θα δημιουργήσει ο Mapserver θα είναι UniversityFinal. Η λέξη κλειδί UNITS ορίζει τις μονάδες μέτρησης του χάρτη σε μέτρα. Η γεωγραφική

έκταση (η ορθογώνια περιοχή που καλύπτεται από τον χάρτη) ορίζεται από την λέξη κλειδί EXTENT στη γραμμή 3. Η ορθογώνια περιοχή καθορίζεται από τις συντεταγμένες των σημείων των αντίθετων γωνιών (της κάτω αριστερής και της πάνω δεξιάς). Οι διαστάσεις του τελικού χάρτη θα είναι 800 X 344 pixels, όπως ορίζεται από την λέξη κλειδί SIZE. Στη γραμμή 5, το IMAGECOLOR καθορίζει ότι το χρώμα του φόντου για την map εικόνα θα είναι το λευκό και στη γραμμή 6 το IMAGETYPE καθορίζει ότι η μορφή της δημιουργούμενης εικόνας θα είναι SVG. Τέλος η θέση των shapefiles καθορίζεται στη γραμμή από τη λέξη κλειδί SHAPEPATH στη γραμμή 7 και στη γραμμή 8 καθορίζεται η θέση του αρχείου fontset.txt, που κάνει την αντιστοίχιση της ονομασίας (alias) της γραμματοσειράς με τη θέση της.

```
NAME "UniversityFinal"
UNITS meters
EXTENT 0 0 86 37
SIZE 800 344
IMAGECOLOR 255 255 255
IMAGETYPE svg
SHAPEPATH "C:/Mapdata/finalUniversityData/Ground/"
FONTSET "C:/ms4w/Apache/htdocs/fontset.txt"
```

Στις γραμμές 10 έως και 16 ορίζεται και ονομάζεται η μορφή του παραγόμενου αρχείου με τη χρησιμοποίηση του OUTPUTFORMAT object, το οποίο επεκτείνει την έννοια της μορφής της εικόνας svg για να περιλάβει λεπτομέρειες για τη δομή της εικόνας και ακόμα ποιος driver γραφικών πρέπει να χρησιμοποιηθεί για να παραχθεί μια εικόνα.

```
OUTPUTFORMAT
  NAME svg
  MIMETYPE "image/svg+xml"
  DRIVER "svg"
  FORMATOPTION "COMPRESSED_OUTPUT=FALSE"
  FORMATOPTION "FULL_RESOLUTION=FALSE"
END
```

Εκτός από τον ορισμό του OUTPUTFORMAT object για να παραχθεί μια SVG εικόνα πρέπει να δοθούν έγκυρες τιμές στις λέξεις κλειδιά IMAGEPATH και IMAGEURL στο WEB object του mapfile. Το WEB object στο συγκεκριμένο mapfile βρίσκεται στις γραμμές 18 έως και 21. Η λέξη κλειδί IMAGEPATH υποδεικνύει στον Mapserver που να τοποθετήσει τις εικόνες χαρτών που παράγει και η λέξη κλειδί IMAGEURL καθορίζει ένα URL που λέει στον browser που να κοιτάξει για να ανακτήσει την εικόνα. Ο Mapserver

θα συμπεριλάβει το URL στην σελίδα πριν σταλεί πίσω στον browser. Σ' αυτό το σημείο πρέπει να τονιστεί ότι το IMAGEPATH string είναι ένα απόλυτο μονοπάτι στον τοπικό εξυπηρέτη (localhost) καθώς το IMAGEURL καθορίζει τη θέση λαμβάνοντας υπόψη το DocumentRoot του web server.

```
WEB
  IMAGEPATH "/ms4w/tmp/ms_tmp/"
  IMAGEURL "/ms_tmp/"
END
```

Με βάση τις παραπάνω πληροφορίες ο Mapserver ξέρει τι είδος εικόνας να παράγει, τι μέγεθος και χρώμα φόντου να δώσει σε αυτή και πώς να απεικονίσει το χάρτη που δημιουργεί σε μια ιστοσελίδα. Αυτό που δεν γνωρίζει ακόμη είναι τι να σχεδιάσει και πώς να το σχεδιάσει. Υπεύθυνο για αυτές τις εργασίες του Mapserver είναι το LAYER object, το οποίο εισάγει τα layers του χάρτη. Ένα layer, όπως έχει αναφερθεί και παραπάνω, αναφέρεται σε μια μοναδική ομάδα δεδομένων και περιέχει μια ομάδα στοιχείων που θα αποδοθούν μαζί σε μια συγκεκριμένη κλίμακα χρησιμοποιώντας μια συγκεκριμένη προβολή Στη συνέχεια του κώδικα αναπτύσσονται τα layers του χάρτη.

Στις γραμμές 26 έως και 53 ορίζεται το πρώτο layer του χάρτη, που αποτελεί το σχήμα του 1^{ου} ορόφου. Η λέξη κλειδί NAME στη γραμμή 27 καθορίζει ένα όνομα για το layer. Στην επόμενη γραμμή, η λέξη κλειδί DATA προσδιορίζει το όνομα του shapefile που θα αποδοθεί σ' αυτό το layer. Η τιμή που συνδέεται με αυτή τη λέξη κλειδί είναι στην πραγματικότητα το σχετικό μονοπάτι από το SHAPEPATH. Στη συνέχεια του κώδικα, στη γραμμή 29, εμφανίζεται η λέξη κλειδί STATUS, η οποία αποφασίζει αν ένα layer θα αποδοθεί ή όχι και αν η κατάσταση του μπορεί να αλλάξει. Στην προκειμένη περίπτωση το STATUS είναι on, που σημαίνει ότι το layer αρχικά θα απεικονίζεται αλλά σε μεταγενέστερη κατάσταση να μην εμφανίζεται αν αυτό επιλέξει ο χρήστης. Η λέξη κλειδί TYPE καθορίζει ότι ο τύπος του layer "Ground_shp" θα είναι polygon.

```
LAYER
 NAME "Ground_shp"
 DATA "Ground_shp"
 STATUS on
 TYPE polygon
```

Στις γραμμές 31 έως 33 εμφανίζονται οι λέξεις κλειδιά LABELCACHE, LABELITEM και CLASSITEM. Η λέξη κλειδί LABELCACHE δίνει την δυνατότητα στον Mapserver να αποθηκεύει τις ετικέτες των layers, των οποίων το LABELCACHE είναι on, και να τις παραδίδει ταυτόχρονα αφού αυτά έχουν σχηματιστεί. Εκτός από το χρόνο που θα

παραδοθεί μια ετικέτα πρέπει και να ονομαστεί. Αυτό ακριβώς κάνει η λέξη κλειδί LABELITEM, δίνει όνομα στην κάθε ετικέτα του layer. Το LABELITEM έχει την τιμή “id” πράγμα που σημαίνει ότι η ετικέτα κάθε πολυγώνου θα πάρει το όνομα της από το attribute “id” του shapefile (layer). Η λέξη κλειδί CLASSITEM αναγνωρίζει το όνομα του attribute που θα χρησιμοποιηθεί για να ταξινομήσει τα features.

```
LABELCACHE on
    LABELITEM "id"
    CLASSITEM "id"
```

Στη συνέχεια του κώδικα ορίζεται το class object του layer, στο οποίο περιέχονται οι οδηγίες, που καθοδηγούν τον Mapserver, ως προς τον τρόπο παράδοσης των features σ’ ένα layer. Στο layer “Ground_shp” υπάρχει μία κλάση, όμως ο αριθμός των κλάσεων μπορεί να είναι και μεγαλύτερος σ’ ένα layer. Αυτό συμβαίνει όταν για παράδειγμα σ’ ένα layer που αναπαριστά δρόμους θέλουμε να ξεχωρίσουμε τους αυτοκινητόδρομους από τους μικρούς επαρχιακούς δρόμους, οπότε και θα τους κάνουμε μεγαλύτερους και με διαφορετικό χρώμα. Η μοναδική λοιπόν κλάση του layer “Ground_shp” προκαλεί τον Mapserver να επιλέξει όλα τα feature για να τα αποδώσει στη συνέχεια.

Η λέξη κλειδί NAME της κλάσης είναι η ετικέτα που θα εμφανιστεί στη λεζάντα που σχετίζεται με τον χάρτη. Το style object, που εμφανίζεται στις γραμμές 36 έως 40, περιέχει τις παραμέτρους που περιγράφουν τον τρόπο με τον οποίο θα σχεδιασθούν τα πολύγωνα της κλάσης. Με τη λέξη κλειδί SIZE καθορίζεται το ύψος του πολυγώνου, με την COLOR καθορίζεται το χρώμα το οποίο θα γεμίσει το πολύγωνο και η OUTLINECOLOR καθορίζει το χρώμα του περιγράμματος των πολυγώνων.

```
CLASS
        NAME "id"
        STYLE
            SIZE 1
            COLOR 211 255 190
            OUTLINECOLOR 0 0 0
        END
```

Επίσης μέσα στο class object υπάρχει και το label object, το οποίο παραδίδεται με την κλάση και καθορίζει τον τύπο της γραμματοσειράς, το μέγεθος, την θέση και το χρώμα της ετικέτας. Το TYPE, στη γραμμή 42, καθορίζει τον τύπο της γραμματοσειράς που χρησιμοποιείται για να αποδώσει την ετικέτα. Οι επιλογές που υπάρχουν είναι δύο: οι bitmapped και οι TrueType γραμματοσειρές. Οι bitmapped δημιουργούνται εσωτερικά και είναι οι default γραμματοσειρές του mapserver και οι TrueType πρέπει να

εγκατασταθούν. Στον κώδικα μας ο τύπος της γραμματοσειράς που έχει επιλεγεί είναι TrueType, οπότε είναι απαραίτητο να καθοριστεί και η γραμματοσειρά που θα χρησιμοποιηθεί. Στην επόμενη γραμμή η λέξη κλειδί FONT παίρνει την τιμή “arial”, που σημαίνει ότι ο Mapserver ξέρει ότι η γραμματοσειρά που εμφανίζεται με το string “arial” στο FONTSET αρχείο θα χρησιμοποιηθεί γι’ αυτή την ετικέτα. Με το SIZE ορίζεται το μέγεθος της ετικέτας σε 8 pixels, με το COLOR ορίζεται το χρώμα σε μαύρο και με το OUTLINECOLOR το χρώμα περιγράμματος σε λευκό. Στις επόμενες γραμμές η λέξη κλειδί MINDISTANCE ορίζει τον αριθμό τον pixels μεταξύ ίδιων ετικετών σε 5, η λέξη κλειδί POSITION ορίζει τη θέση της ετικέτας στο κέντρο του καθορισμένου τετραγώνου γι’ αυτή (cc – center center) και το MINFEATURESIZE ορίζει ότι το μέγεθος (σε pixels) της μικρότερης ετικέτας που θα σχεδιασθεί θα είναι 5. Στη γραμμή 50 η λέξη κλειδί ορίζει σαν χαρακτήρα αλλαγής γραμμής μιας ετικέτας το space (“ “).

```
LABEL
        TYPE truetype
        FONT "arial"
        SIZE 8
        OUTLINECOLOR 255 255 255
        COLOR 0 0 0
        MINDISTANCE 5
        POSITION cc
        MINFEATURESIZE 5
        WRAP ' '
END
```

Στη συνέχεια του κώδικα περιγράφονται με τον ίδιο τρόπο τα υπόλοιπα layers που αναφέρονται στον Πίνακα 3. Δηλαδή ξεκινάει πάλι ένα layer object για κάθε ένα από τα υπολειπόμενα layers με τη λέξη κλειδί LAYER και κλείνει με τη λέξη κλειδί END. Πάλι μέσα στο layer object περιέχεται ένα class object, το οποίο περιέχει ένα label object για το σχεδιασμό των ετικετών.

4.2.3. Επεξήγηση Κώδικα PHP

Ο κώδικας σε php/mapscript που τρέχει στον server περιέχεται στο αρχείο «getUniversityMap.php». Η γραμμή 1 ξεκινά το php script. Στη γραμμή 2 καθορίζεται ότι ο server θα επιστρέψει στον client ένα αντικείμενο το οποίο θα είναι τύπου SVG. Στις επόμενες γραμμές καθορίζεται η διαδρομή και το όνομα του mapfile το οποίο θα πρέπει να διαβάσει η mapscript για τη δημιουργία του χάρτη.

```
<?php
header("Content-type: image/svg+xml");
$map_path = "C:/ms4w/Apache/htdocs/UniversityFinal/";
$map_file = "UniversityFinal.map";
$img_path = "/ms4w/tmp/ms_tmp/";
```

Στις γραμμές 15 έως και 21 αποθηκεύονται σε αντίστοιχες μεταβλητές οι παράμετροι κλήσης του server.

```
//get URL parameters
$layername = $_GET['layername'];
$xmin = intval($_GET['xmin']);
$xmax = intval($_GET['xmax']);
$ymin = intval($_GET['ymin']);
$ymax = intval($_GET['ymax']);
$timestamp = $_GET['timestamp'];
```

Ουσιαστικά είναι οι παράμετροι που έχουν αποσταλεί από τον client και αφορούν το layer που έχει ζητηθεί να απεικονισθεί, το παράθυρο θέασης του layer και μια μεταβλητή σχετική με το χρόνο που έγινε η κλήση από τον client. Το παράθυρο θέασης αναφέρεται στις γεωγραφικές συντεταγμένες (x,y), που στην περίπτωσή μας εκφράζονται σε μέτρα, της κάτω αριστερής και της πάνω δεξιάς γωνίας του παραλληλογράμου τμήματος που έχει ζητηθεί να προβληθεί από τον client (περιπτώσεις zoom in/out και pan). Οι αποστάσεις μεταξύ των συντεταγμένων του παραθύρου θέασης αποτελούν και το extent του χάρτη που θα αποστείλει ο server. Το μέγιστο extent είναι 86m μήκος και 37m πλάτος και είναι οι εξωτερικές διαστάσεις του χώρου που απεικονίζουμε. Το σημείο (0,0) θεωρείται η κάτω αριστερή γωνία του χάρτη όταν αυτός είναι στο μέγιστο extent (πλήρης χάρτης χωρίς zoom).

Στις γραμμές 33-52 ορίζεται η συνάρτηση img2map(), ενώ στις γραμμές 56-69 η συνάρτηση map2img(). Οι συναρτήσεις αυτές χρησιμοποιούνται όταν απαιτείται μετατροπή των συντεταγμένων εικόνας (pixels στην οθόνη) σε πραγματικές συντεταγμένες και αντίστροφα. Η mapscript συνάρτηση zoomrectangle() απαιτεί ως παράμετρο να δοθεί το παράθυρο που θα γίνει zoom, σε συντεταγμένες εικόνας. Από την άλλη ο client αποστέλλει το παράθυρο θέασης (στο οποίο θέλει να γίνει zoom) σε

πραγματικές συντεταγμένες. Έτσι θα πρέπει να υπάρχουν οι αντίστοιχες συναρτήσεις μετατροπής από το ένα σύστημα στο άλλο.

```
//conversion of geographical coordinates (meters) to image coordinates (pixels)
function map2img($width,$height,$point,$ext) {
  // valid point required
  if ($point->x && $point->y){
    // find pixels per degree
    $ppd_x = $width/($ext->maxx - $ext->minx);
    $ppd_y = $height/($ext->maxy - $ext->miny);
    // calculate image coordinates
    $p[0] =  $ppd_x * ($point->x - $ext->minx);
    $p[1] =  $ppd_y * ($point->y - $ext->miny);
    settype($p[0],"integer");
    settype($p[1],"integer");
  }
  return $p;
} // end map2img
```

Η μέθοδος που χρησιμοποιείται είναι απλή. Η map2img λαμβάνει ως παραμέτρους το μήκος και πλάτος του χάρτη σε pixels, τις συντεταγμένες του σημείου σε μέτρα που θέλουμε να μετατρέψουμε από μέτρα σε pixels και το extent του χάρτη σε μέτρα. Υπολογίζει τα pixel ανά μέτρο του χάρτη και στη συνέχεια πολλαπλασιάζοντας τον αριθμό αυτό με την απόσταση του σημείου που θέλουμε να μετατρέψουμε από την κάτω αριστερή γωνία του παραθύρου θέασης βρίσκουμε τα αντίστοιχα pixels. Στις γραμμές 63-64 αποθηκεύονται οι τιμές των pixels ανά διάσταση σε έναν πίνακα. Στη συνέχεια οι τιμές αυτές ορίζονται ως τύπου integer, ώστε να απαλειφθούν και τα τυχόν δεκαδικά ψηφία και στη γραμμή 66 ο πίνακας που περιέχει τις ζητούμενες τιμές επιστρέφεται στη συνάρτηση που κάλεσε την map2img. Αντίστοιχα λειτουργεί και η συνάρτηση img2map.

Στη γραμμή 72 ορίζεται ένα νέο Map object βάσει του αρχείου mapfile που περιγράφεται παραπάνω. Το extent του αντικειμένου αυτού είναι αυτό που ορίζεται στο mapfile ενώ τα layers που θα εμφανίζει είναι αυτά για τα οποία το STATUS στο mapfile έχει οριστεί ως Default ή On.

```
// Retrieve mapfile and create a map from it
$map = ms_newMapObj($map_path.$map_file);
```

Στις γραμμές 74-84 επιλέγονται τα layers που θα απεικονιστούν από τον mapserver με βάση το τι ζητήθηκε από τον client. Ουσιαστικά εξετάζει όλα τα layers και εφόσον το όνομα κάποιου από αυτά είναι ίδιο με τη μεταβλητή στην οποία είχε αποθηκευτεί το layer που είχε ζητήσει ο client ($layername) το θέτει ON διαφορετικά το layer τίθεται OFF.

```
//set the map layers to ON (will be drawn) or OFF (not drawn) depending
//on the chosen layername variable
for ($layerindex = 0; $layerindex < $map->numlayers; $layerindex++) {
        $lay = $map->getLayer($layerindex);
        if ($layername==$lay->name ) {
                $lay->set(status,MS_ON);
        } else {
                $lay->set(status,MS_OFF);
        }
}
```

Στις γραμές 86-106 τίθεται το extent σε μέτρα του χάρτη με βάση το παράθυρο θέασης που δόθηκε ως παράμετρος από τον client και γίνεται zoom στο παράθυρο αυτό. Για το σκοπό αυτό χρησιμοποιούνται οι mapscript συναρτήσεις setExtent και zoomrectangle με τις αντίστοιχες απαιτούμενες μεταβλητές.

```
// Set the map to the extent retrieved from the form
$map->setextent($extent[0],$extent[1],$extent[2],$extent[3]);
// Save this extent as a rectObj, we need it to zoom.
$old_extent->setExtent($extent[0],$extent[1],$extent[2],$extent[3]);
$my_extent=ms_newRectObj();
$my_extent->setExtent($extent[0],$extent[1],$extent[2],$extent[3]);
$LLpoint=ms_newPointObj();
$URpoint=ms_newPointObj();
$LLpoint->setXY($extent[0]+0.01,$extent[1]+0.01);
$URpoint->setXY($extent[2],$extent[3]);
list($a,$b) = map2img($map->width,$map->height,$LLpoint,$old_extent);
list($c,$d) = map2img($map->width,$map->height,$URpoint,$old_extent);
$my_extent->setExtent($a,$b,$c,$d);
```

```
$map->zoomrectangle($my_extent,$map->width,$map-
>height,$old_extent,$max_extent);
```

Επισημαίνεται ότι στη γραμμή 98 ορίζουμε το σημείο της κάτω αριστερής γωνίας. Παρατηρούμε ότι στο σημείο αυτό δίνουμε την τιμή ($extent[0]+0.01, $extent[1]+0.01). Υπενθυμίζουμε ότι η τιμή ($extent[0], $extent[1]) στην περίπτωση που δεν έχει γίνει zoom είναι (0,0). Επειδή παρατηρήθηκε ότι μέσω της mapscript δεν ήταν δυνατό να γίνει zoom εάν δινόταν αυτή η τιμή κρίθηκε αναγκαίο να προσθέτουμε μια μικρή μη μηδενική τιμή. Αυτό δεν αλλοιώνει το αποτέλεσμα, αφού στη συνέχεια με τη συνάρτηση map2img γίνεται στρογγυλοποίηση σε integer. Στην περίπτωση που ο χάρτης μας δεν ξεκινά από το (0,0) αλλά από μια μη μηδενική τιμή δεν απαιτείται να προσθέσουμε το 0.01. Στην τελική υλοποίηση της εφαρμογής τα shapefiles έγιναν με τέτοιο τρόπο ώστε η αρχή των αξόνων να μην είναι το (0,0) αλλά το (0.01,0.01), οπότε επί της ουσίας δεν χρειάζεται να προσθέσουμε κάποιον αριθμό.

Στις γραμμές 108-113 διαβάζεται το αρχείο path.txt στο οποίο έχουν καταγραφεί τα id των σημείων τα οποία περιλαμβάνει η διαδρομή που θέλει ο χρήστης.

```
//save to a variable the text file that contains the required navigational path
//the path is given by point ids separated commas
$path_file = "path.txt";
//read the path file and store it  to an array named $path
$path_matrix=file_get_contents($path_file);
$path=explode(",",$path_matrix);
```

Ουσιαστικά τα ids αποθηκεύονται στον πίνακα $path προκειμένου να χρησιμοποιηθούν στη συνέχεια για τη δυναμική δημιουργία ενός layer που θα περιλαμβάνει ευθύγραμμα τμήματα που θα ενώνουν τα σημεία με τα συγκεκριμένα ids. Κάθε φορά που αλλάζουμε το αρχείο path.txt, αλλάζει και το layer της διαδρομής (route). Σε επόμενη φάση υπάρχει η δυνατότητα χρησιμοποίησης κάποιου αλγόριθμου εύρεσης βέλτιστης διαδρομής μεταξύ των ζητούμενων σημείων το αποτέλεσμα του οποίου θα καταγράφεται στο path.txt ώστε να απεικονιστεί στο χάρτη.

Στις γραμμές 122-140 επεξεργαζόμαστε το layer το οποίο έχει ζητήσει για απεικόνιση ο client. Αυτό που πραγματοποιείται είναι η εύρεση όλων των σχημάτων (shapes) που περιέχονται στο παράθυρο θέασης του συγκεκριμένου layer προκειμένου να αποθηκευτούν στους πίνακες $kind, $pointsid και $ids τα attributes KIND, FLOOR και

ID των σχημάτων αυτών. Ο λόγος που αποθηκεύουμε τις τιμές αυτές είναι προκειμένου να χρησιμοποιηθούν από τις συναρτήσεις insertIDs και insertGroundIDs όπως θα εξηγηθεί παρακάτω. Δεδομένου ότι τα layers grCor και Route δεν περιέχουν τα ανωτέρω attributes γίνεται ο έλεγχος της γραμμής 122 (if ($layername !="grCor" && $layername !="Route"))*.*

```
//If $layername is a point layer then we should get the id of each point on that
//layer, store them in the array $ids[] in order to
// insert these values later in the svg file, since mapserver doesn't insert these ids in the svg by itself.
//*****************************************************************************
if ($layername !="grCor" && $layername !="Route")
{
	$i=0;
	$this_layer= $map->getLayerByName($layername);
	$status=$this_layer->open();
	$status=$this_layer->whichShapes($map->extent);
	if ($status == MS_SUCCESS)
	{
	//every shape is a point object. We read its ID feature and save it in an array
		while ( $shape=$this_layer->nextShape())
		{
			$kind[$i]=$shape->values["KIND"];
			$pointsid[$i]=$shape->values["FLOOR"];
			$ids[$i]=$shape->values["ID"];
			$i++;
		}
	}
	$this_layer->close;
}
```

Ιδιαίτερη μνεία θα πρέπει να γίνει στον έλεγχο if ($status==MS_SUCCESS) ο οποίος δε φαίνεται να χρησιμοποιείται στη βιβλιογραφία του mapserver οπότε αρχικά δεν είχε ενσωματωθεί στον κώδικά μας με αποτέλεσμα να προκαλούνται προβλήματα στη λειτουργία της εφαρμογής. Βάσει του ορισμού των συναρτήσεων του mapserver η αμέσως προηγούμενη του ελέγχου αυτού κλήση συνάρτησης (δηλαδή η $this_layer->whichShapes($map->extent)) είναι απαραίτητη προκειμένου να δουλέψει το while loop. Η συνάρτηση $this_layer->whichShapes($map->extent); Επιστρέφει

MS_SUCCESS ή MS_FALSE. Ακόμα και εάν εκτελέσουμε το while loop χωρίς να κάνουμε έλεγχο της $status η εφαρμογή δουλεύει κανονικά. Εκτελώντας την εφαρμογή υπό διάφορες συνθήκες (συνδυασμοί επιλογών layers και zoom) παρατηρήθηκε ότι κάποιες φορές η εφαρμογή εμφάνιζε μηνύματα σφάλματος που είχαν να κάνουν με αδυναμία σύνδεσης με τον server. Λόγω χρήσης της τεχνολογίας AJAX το πρόγραμμα συνέχιζε να λειτουργεί οπότε παρατηρήσαμε ότι αλλάζοντας το συνδυασμό zoom και επιλογής layer συνδεόταν κανονικά στο server και έδινε τα ζητούμενα αποτελέσματα. Τελικά αποδείχθηκε ότι η αδυναμία σύνδεσης με το server οφειλόταν στο ότι έληγε το χρονικό διάστημα αναμονής απάντησης από τον server. Αυτό συνέβαινε στην περίπτωση που είχε επιλεγεί να εμφανιστεί κάποιο layer το οποίο όμως στο συγκεκριμένο παράθυρο θέασης που είχαμε επιλέξει να κάνουμε zoom δεν διέθετε κανένα σχήμα. Το αποτέλεσμα ήταν ο βρόχος while να ψάχνει για σχήματα εκεί όπου δεν υπήρχαν με αποτέλεσμα ο server να μη δίνει καμία απάντηση στον client. Για το λόγο αυτό είναι απαραίτητος ο έλεγχος της μεταβλητής $status έτσι ώστε εφόσον το παράθυρο θέασης δεν περιλαμβάνει shapes του συγκεκριμένου layer ο while βρόχος να μην ξεκινά.

Στις γραμμές 145 – 174 ορίζονται οι συναρτήσεις insertIDs και insertGroundIDs με τις οποίες εισάγονται τα ids στα σχήματα των layer του svg αρχείου. Σαν παραμέτρους δέχονται τους πίνακες $kind, $pointsid και $ids που δημιουργήσαμε παραπάνω καθώς και το svg αρχείο που δημιουργεί ο mapserver. Η λειτουργία των δύο συναρτήσεων είναι παρόμοια οπότε παρακάτω παραθέτουμε μόνο τη μία από αυτές.

```
//inserts the ids in the svg point layers
function insertIDs($Kindarray,$PointIDArray,$IDsArray,$pointslayerFile)
{
	$data = file($pointslayerFile);
	// find the size of rows of the svg file
	$count = count($data);
	//insert ids from ids array where the point ads where stored, to svg file
	for ($i = 0; $i < $count; $i++)
	{
		//insertion is made by replacement of ellipse with ellipse+id
		$data[$i+4]=str_replace("ellipse", "ellipse id=\"".$Kindarray[$i]."_".
$PointIDArray[$i]. "_".$IDsArray[$i]."\"", $data[$i+4]);
	}
	//save the new svg with the ids
```

```
        file_put_contents($pointslayerFile, $data) or die('Could not write to file');
}
```

Ο λόγος που χρησιμοποιούνται οι συναρτήσεις αυτές είναι ο εξής: Ο mapserver όπως θα δούμε παρακάτω για κάθε layer που του ζητείται φτιάχνει ένα svg αρχείο το οποίο αποστέλλει στον client. Η μορφή του svg είναι η εξής (δίνεται το layer grAll):

```
<?xml version="1.0" encoding="UTF-8"?>
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" version="1.1" width="800" height="344" id="grAll" timestamp="1205347040480">

<!-- START LAYER grAll -->
<ellipse cx="515" cy="340" rx="2" ry="2" fill="#0000ff"/>
<ellipse cx="475" cy="341" rx="2" ry="2" fill="#0000ff"/>
<ellipse cx="569" cy="235" rx="2" ry="2" fill="#0000ff"/>
<ellipse cx="590" cy="235" rx="2" ry="2" fill="#0000ff"/>
….
</svg>
```

Παρατηρούμε ότι από τη γραμμή *<!-- START LAYER grAll -->* και κάτω κάθε σχήμα του layer (στην περίπτωσή μας είναι ένα point) αναπαρίσταται σε μια γραμμή αλλά δεν δίνεται κανένα id για το σχήμα αυτό. Έτσι όμως ο client δεν έχει τη δυνατότητα να το χειριστεί περαιτέρω εφόσον το SVG δεν περιέχει τα ids των σχημάτων. Δεδομένου ότι δεν είχαμε τη δυνατότητα να επέμβουμε στις συναρτήσεις του mapserver που καθορίζουν τον τρόπο δημιουργίας του SVG από τα shapefiles και παρατηρώντας ότι η σειρά των παραπάνω γραμμών που αναπαριστούν τα σημεία είναι ίδια με τη σειρά των ids των shapefiles που διαβάζει ο mapserver, δημιουργήσαμε τις συναρτήσεις insertIds με τις οποίες διαβάζεται το SVG που έχει δημιουργηθεί και σε κάθε γραμμή εισάγουμε το id το οποίο προκύπτει ως ένωση (concatenation) των πινάκων $kind, $pointsid και $ids που περιέχουν τα αντίστοιχα attributes των shapefiles. Έτσι το αποτέλεσμα είναι το νέο SVG αρχείο να είναι της μορφής:

```
<?xml version="1.0" encoding="UTF-8"?>
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" version="1.1" width="800" height="344" id="grAll" timestamp="1205347040480">

<!-- START LAYER grAll -->
```

```
<ellipse id="Junction_0_1" cx="515" cy="340" rx="2" ry="2" fill="#0000ff"/>
<ellipse id="Junction_0_2" cx="475" cy="341" rx="2" ry="2" fill="#0000ff"/>
<ellipse id="Junction_0_3" cx="569" cy="235" rx="2" ry="2" fill="#0000ff"/>
<ellipse id="Turn_Point_0_4" cx="590" cy="235" rx="2" ry="2" fill="#0000ff"/>
…
</svg>
```

Στις γραμμές 176-204 ελέγχεται εάν απαιτείται η απεικόνιση του layer της διαδρομής (route) και εφόσον αυτό ισχύει γίνεται εύρεση των πραγματικών συντεταγμένων των σχημάτων (points) των οποίων τα ID έχουν διαβαστεί από το αρχείο path.txt που αναφέρθηκε παραπάνω. Επισημαίνεται ότι τα IDs αυτά έχουν ήδη διαβαστεί και αποθηκευτεί στον πίνακα $path. Επίσης θα πρέπει να τονιστεί ότι η διαδικασία αυτή είναι ανεξάρτητη από την προηγούμενη με την οποία κάναμε εισαγωγή των IDs των σχημάτων στο SVG αρχείο. Αυτό σημαίνει ότι ακόμα και εάν η προηγούμενη διαδικασία δεν γινόταν η εύρεση των πραγματικών συντεταγμένων με βάση δοθέντα IDs παραμένει εφικτή. Για να γίνει αυτό κατανοητό σημειώνεται ότι η συγκεκριμένη μετατροπή γίνεται στο map object και όχι στο SVG αρχείο. Ο mapserver γνωρίζει και μέσω της mapscript μπορεί να χειριστεί τα ids του map object, απλά δεν τα περνάει στο SVG όταν το δημιουργεί.

```
//If the user chose to see the route
if ($layername == 'Route')
{
	$i=0;
//Read all the point layers
	for ($layerindex = 0; $layerindex < $map->numlayers; $layerindex++)
	{
		$lay = $map->getLayer($layerindex);
		if ($lay->name != "Ground_shp" && $lay->name !="grCor" && $lay->name !="Route")
		{
			$status=$lay->open();
//for each shape (point) get its id and if this matches with the id read by the path array in which the desired route is stored
//get the geographic X,Y coordinates (meters)  of that point. These coordinates are stored to the XYarray and they will be used
//to draw the path lines and the points of the route
```

```
                for ($i=0;$i<count($path);$i++)
                {
                        $status=$lay->whichShapes($my_extent);

                        while ($shape=$lay->nextShape())
                        {
                                $element=$shape->values["KIND"]."_".$shape-
>values["POINTID"]."_".$shape->values["ID"];
                                if ( $path[$i] == $element)
                                {
                                        $XYarray[$i]=((double)$shape-
>values["X"]).",".((double)$shape->values["Y"]).",".($shape->values["ID"])."\n";
                                }
                        }
                }
                $lay->close;
        }
    }
```

Η παραπάνω διαδικασία είναι απαραίτητη για την απεικόνιση της διαδρομής. Είπαμε ότι η διαδρομή ουσιαστικά δημιουργείται δυναμικά ως ένα επιπλέον layer. Με χρήση mapscript μπορούμε να φτιάξουμε το layer αυτό το οποίο θα είναι τύπου line. Η δημιουργία ενός ευθύγραμμου τμήματος προϋποθέτει την ύπαρξη δύο σημείων. Τα σημεία αυτά θα πρέπει να δοθούν με τη μορφή πραγματικών συντεταγμένων. Όπως είπαμε παραπάνω ανάμεσα στα attributes των layers των shapefiles είναι και οι πραγματικές τους συντεταγμένες (x,y). Με τον παραπάνω κώδικα αναζητούμε τα δοθέντα IDs (του path.txt) στα points των layers και εφόσον τα βρούμε αποθηκεύουμε τις πραγματικές συντεταγμένες των σημείων αυτών στον πίνακα XY.

Στις γραμμές 205-271 δημιουργείται ένα νέο layer το οποίο περιλαμβάνει τη γραμμή που ενώνει τα σημεία με πραγματικές συντεταγμένες αυτές του πίνακα XY. Επιπρόσθετα, επειδή η γραμμή αυτή μπορεί να τύχει να περνάει και πάνω από άλλα σημεία τα οποία όμως δεν είχαμε επιλέξει να είναι στη διαδρομή, για να αποφευχθεί τυχόν παρανόηση, τα σημεία που ανήκουν στη ζητούμενη διαδρομή χρωματίζονται εντονότερα στο χρώμα της υπόψη γραμμής (κόκκινο). Για να γίνει αυτό απλά στο layer της διαδρομής δημιουργούμε δυναμικά και σημεία (points) με πραγματικές

συντεταγμένες που λαμβάνονται από τον πίνακα XY. Δηλαδή το layer της διαδρομής περιλαμβάνει τόσο τη γραμμή της διαδρομής όσο και τα σημεία αυτής.

Μετά τα παραπάνω έχουμε δημιουργήσει το τελικό map object που ο mapserver πρέπει να κάνει διανυσματική εικόνα τύπου svg και να στείλει στον client.

Στις γραμμές 276-282 δημιουργείται και αποθηκεύεται το ζητούμενο SVG αρχείο.

```
// Drawing the map
// create unique names for map and reference images
$image_name = "University_".$layername.".svg";
//$image_url="/ms_tmp/".$image_name;
$imagefilename=$img_path.$image_name;

// Draw map image
$image=$map->draw();
$image->saveImage($imagefilename);
```

Εδώ θα πρέπει να σχολιαστεί ένα πρόβλημα που παρατηρήθηκε στη λειτουργία του mapserver κατά τη δημιουργία SVG αρχείων. Με τη συνάρτηση *$image=$map->draw();* ο mapserver δημιουργεί το χάρτη με βάση το map object ενώ με την *saveImage($imagefilename);* τον αποθηκεύει. Στις περιπτώσεις των εικόνων raster η συνάρτηση $map->draw δημιουργεί τη raster εικόνα και η saveImage την αποθηκεύει. Στην περίπτωση όμως της svg εικόνας παρατηρούμε ότι η $map->draw όχι μόνο δημιουργεί την svg εικόνα αλλά συγχρόνως την αποθηκεύει με ένα όνομα που επιλέγει τυχαία. Στη συνέχεια με την *saveImage* η SVG εικόνα αποθηκεύεται με το όνομα που έχουμε επιλέξει. Στην περίπτωση που δεν χρησιμοποιήσουμε την draw δεν δουλεύει η saveImage ενώ αν δεν χρησιμοποιήσουμε την saveImage δεν μπορούμε να εκτελέσουμε συγκεκριμένες λειτουργίες (όπως εισαγωγή ids) αφού δεν γνωρίζουμε το όνομα της SVG εικόνας ώστε να το διαβάσουμε και να το επεξεργαστούμε. Το αποτέλεσμα είναι για κάθε layer να αποθηκεύονται δύο svg αρχεία όπως για παράδειγμα τα παρακάτω που αφορούν το layer Ground_shp:

47dc408d_d94_0.svg και University_Ground_shp.svg

Σημειώνεται επίσης ότι στον client αποστέλλεται το ένα από τα 2 SVG αρχεία (αυτό του οποίου το όνομα μας είναι γνωστό).

Στις γραμμές 284-291 καλούμε τις αντίστοιχες συναρτήσεις για εισαγωγή των ids.

```
if ($layername != "Ground_shp" && $layername != "Route")
{
	//we insert  the id of the point object in the svg file
	insertIDs($kind,$pointsid,$ids,$imagefilename);
}elseif ($layername == "Ground_shp")
{
	insertGroundIDs($ids,$imagefilename);
}
```

Τέλος στις γραμμές 293 έως 318 αποστέλλουμε το svg αρχείο στον client αφού πρώτα του ενσωματώσουμε κάποια απαραίτητα στοιχεία.

```
//creation of a new xml document
$doc = new DomDocument();

//loading our svg image
$doc->Load($img_path.$image_name);

//getting the root element of the svg image which is an svg element
$root = $doc->documentElement;

//creation of id attribute with value of the $layername variable
$id_var = new DOMAttr("id", $layername);
//insertion of the id attribute to the root element
$root->appendchild($id_var);

//creation of timestamp attribute with value of the $timestamp variable
$timestamp_var = new DOMAttr("timestamp", $timestamp);

//insertion of the timestamp attribute to the root element
$root->appendchild($timestamp_var);

//creation of the XML output
$output=$doc->saveXML();
$doc->save($img_path.$image_name);
```

```
//sending the SVG/XML file to the user
echo $output;
```

Ήδη από την αρχή του php κώδικα δηλώθηκε στο header ότι ο server θα επιστρέψει ένα αρχείο τύπου svg (δηλαδή xml). Με κάθε κλήση στον server δημιουργείται ένα svg αρχείο που περιλαμβάνει ένα layer χωρίς όμως στο svg να υπάρχει ως attribute το id του layer αυτού. Επιπρόσθετα για την σωστή λειτουργία του client θα πρέπει στο svg αρχείο να εισάγουμε ως attribute και τη μεταβλητή $timestamp. Προκειμένου να υλοποιηθούν τα ανωτέρω κάνουμε χρήση των δυνατοτήτων που μας παρέχει η PHP να χειριστούμε ένα XML DOM Document. Ουσιαστικά δημιουργούμε ένα νέο DOM Document στο οποίο φορτώνουμε το SVG αρχείο (εφόσον το SVG είναι XML) και σε αυτό εισάγουμε τα επιθυμητά attributes.

Στο τέλος με τη συνάρτηση *echo $output;* το τελικό αποτέλεσμα αποστέλλεται στον client.

4.3. Κώδικας Εφαρμογής στον Client

Για την υλοποίηση του τμήματος της εφαρμογής στον client στηριχθήκαμε σε βιβλιοθήκες javascript και βασικών προγραμμάτων και τεχνικών υλοποίησης SVG που περιγράφονται στην ηλεκτρονική διεύθυνση www.carto.net[xx]. Το τμήμα αυτό της εφαρμογής αποτελεί το interface με τον χρήστη. Υλοποιείται εξ ολοκλήρου ως ένα SVG και χρησιμοποιεί τεχνικές AJAX. Για την υλοποίηση της εφαρμογής χρησιμοποιούνται τα παρακάτω αρχεία:

- University.html
- UniversityFinal.svg
- button.js
- checkbox_and_radiobutton.js
- helper_functions.js
- mapApp.js
- navigation.js
- selectionList.js
- slider.js
- timer.js

Σημειώνεται ότι τα αρχεία javascript πρέπει να αποθηκευτούν στο ίδιο directory με το UniversityFinal.svg καθώς περιλαμβάνουν τις συναρτήσεις javascript που καλούνται από το τελευταίο. Το αρχείο university.html δεν είναι απαραίτητο καθώς απλά καλεί το UniversityFinal.svg. Το σύνολο των ανωτέρω αρχείων αποθηκεύτηκαν στο directory C:\ms4w\Apache\htdocs\UniversityFinal. Λαμβάνοντας υπόψη ότι με την εγκατάσταση του maserver/apache server το αρχικό directory που «βλέπουμε» ως localhost είναι το C:\ms4w\Apache\htdocs, για να τρέξει η εφαρμογή καλούμε τη σελίδα: http://localhost/UniversityFinal/University.html. Παρακάτω αναλύεται το αρχείο UniversityFinal.svg το οποίο αποτελεί τον πυρήνα της εφαρμογής στον client.

4.3.1. Ανάλυση κώδικα SVG αρχείου στον client

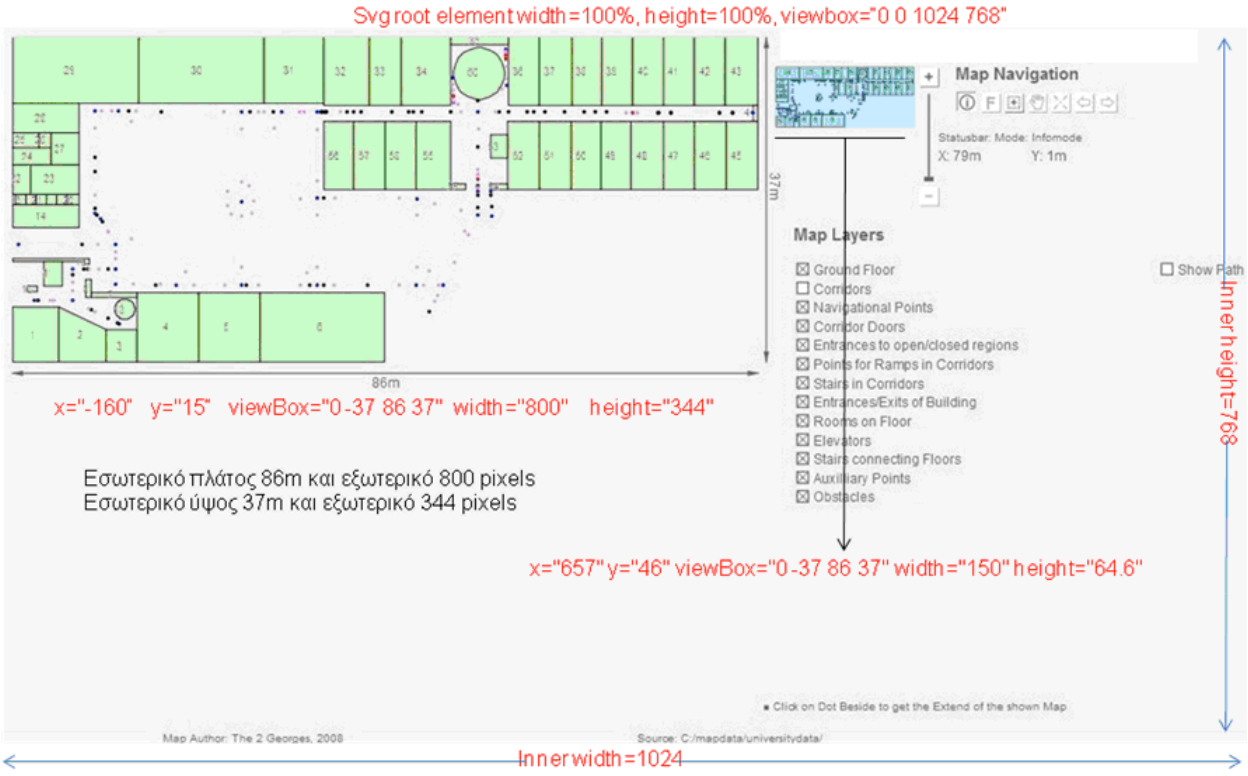
4.3.1.1 Σύστημα Συντεταγμένων

Στο αρχείο UniversityFinal.svg χρησιμοποιούνται τρία διαφορετικά συστήματα συντεταγμένων. Το πρώτο σχετίζεται με την οθόνη απεικόνισης (device oriented) και είναι ορισμένο σε pixels. Καθορίζεται στο root element του SVG αρχείου (γραμμή 3, 1024x768) στο attribute viewbox. Στο ίδιο σημείο του κώδικα ορίζεται το πλάτος και ύψος στο 100%. Στην περίπτωση που αλλαχθεί αυτό το σύστημα συντεταγμένων της οθόνης, θα πρέπει να προσαρμοστούν αντίστοιχα και τα μεγέθη των συμβόλων, που ορίζονται στο αρχείο αυτό, προκειμένου να ανταποκρίνονται στις αλλαγές των συντεταγμένων.

Το δεύτερο σύστημα συντεταγμένων είναι αυτό του παραθύρου του κυρίως χάρτη (main map window στη γραμμή 382, στο φωλιασμένο svg στοιχείο). Το σύστημα αυτό ορίζεται σε μονάδες πραγματικών συντεταγμένων (μέτρα). Σημειώνεται ότι στο σύστημα συντεταγμένων του viewbox ο άξονας y είναι ανεστραμμένος, καθώς τα συστήματα συντεταγμένων των χαρτών λειτουργούν αντίθετα από το σύστημα συντεταγμένων του SVG. Επίσης επισημαίνεται ότι, για τη σωστή και αναλλοίωτη απεικόνιση των χαρτών, θα πρέπει η αναλογία πλάτους και ύψους του εξωτερικού mainMap να είναι ίδια με αυτή του viewbox και βέβαια όμοια με την αντίστοιχη αναλογία του χάρτη που λαμβάνουμε από το server. Για το λόγο αυτό επιλέχθηκε width=800 και height = 344. Επισημαίνεται ότι ο χάρτης που λαμβάνεται από το server είναι διαστάσεων 86m μήκους και 37m πλάτους.

Το τρίτο σύστημα συντεταγμένων αφορά στο χάρτη αναφοράς και είναι το ίδιο με αυτό του main map, πράγμα που σημαίνει ότι το viewbox του referenceMap θα πρέπει να είναι ίδιο με αυτό του MainMap. Έτσι στη γραμμή 452 ορίζουμε το <svg id=”referenceMap” με viewbox ίδιο με αυτό του MainMap. Στην ίδια γραμμή ρυθμίζουμε

το πλάτος και το ύψος του χάρτη αναφοράς καθώς και τις συντεταγμένες (x,y) όπου βρίσκεται η κάτω αριστερή γωνία του.



Εικόνα 22: Διαστάσεις Χάρτη

Στη φωλιασμένη περιοχή με id=”referenceMap” μπορούμε να εισάγουμε απλά σχήματα και εικόνες είτε διανυσματικές είτε raster. Στην περίπτωσή μας τοποθετήθηκε μια raster απεικόνιση της κάτοψης του ισογείου του πανεπιστημίου. Η εικόνα αυτή δε χρειάζεται να είναι λεπτομερής καθώς χρησιμοποιείται μόνο για να δίνει συνολική αίσθηση του χώρου στο χρήστη καθώς αυτός κάνει zoom και pan πάνω στο χάρτη (ιδιαίτερα χρήσιμο χαρακτηριστικό στις περιπτώσεις που έχει γίνει μεγάλο zoom).

Στη φωλιασμένη περιοχή με id=”mainMapGroup” εισάγουμε όλα τα στοιχεία που θέλουμε να απεικονίσουμε (διανυσματικές εικόνες, σχήματα, δυναμικά layers ή στατικές εικόνες κλπ). Όλα τα παραπάνω στοιχεία θα πρέπει να εισαχθούν μέσα σε αυτό το φωλιασμένο group προκειμένου να είναι εφικτές οι λειτουργίες pan και zoom. Τονίζεται και πάλι ότι το σύστημα συντεταγμένων είναι ανεστραμμένο ως προς τον άξονα y. Για το λόγο αυτό όλα τα layers ξεκινούν με συντεταγμένες (0,-37). Το viewbox κάθε layer έχει ύψος και πλάτος όμοιο με του εξωτερικού παραθύρου (mainMap) ενώ για κάθε SVG id ορίζεται ότι θα καλύψει το 100% του width και height μέσα στο viewbox του mainMap.

4.3.1.2 Αρχικοποίηση παραμέτρων προγράμματος προβολής

Απαραίτητη προϋπόθεση για τη λειτουργία της εφαρμογής μας είναι η δήλωση των δύο καθολικών μεταβλητών myMapApp και myMainMap καθώς επίσης και ο ορισμός των μεταβλητών dynamicLayers και digiLayers στη συνάρτηση αρχικοποίησης (init) του κώδικα. Οι μεταβλητές αυτές αρχικά είναι άδειοι πίνακες, οι οποίοι χρησιμοποιούνται για να προσθέσουμε δυναμικά επίπεδα (layers). Για κάθε layer που επιθυμούμε να έχουμε, εισάγουμε και μια τιμή στον πίνακα dynamicLayers όπως φαίνεται στις γραμμές 22 έως 49. Η μεταβλητή digiLayers είναι απαραίτητο να ορισθεί, αν και δε θα χρησιμοποιηθεί. Οι πίνακες αυτοί χρησιμοποιούνται από το map αντικείμενο (mainMap) το οποίο ορίζεται στη γραμμή 59. Το αντικείμενο αυτό περιέχει διάφορα δεδομένα του χάρτη και επίσης είναι αυτό που επιτρέπει τη λειτουργία των εργαλείων zoom και pan. Ως παράμετροι του map αντικείμενου τίθενται οι εξής:

mapName (string): id του mainMap (φωλιασμένο SVG)

maxWidth (number, viewbox coordinates): μέγιστο πλάτος του χάρτη όπως ορίζεται στο σύστημα συντεταγμένων του εσωτερικού viewbox (πραγματικές μονάδες)

minWidth (number, viewbox coordinates): ελάχιστο πλάτος του χάρτη όπως ορίζεται στο σύστημα συντεταγμένων του εσωτερικού viewbox (πραγματικές μονάδες)

zoomFact (number): Συντελεστής μεταξύ 0 και 1 που ορίζει το ποσοστό zoom in και zoom out κάθε φορά που πατάμε τα κουμπιά +/-. Για παράδειγμα μια τιμή 0,6 σημαίνει ότι στο επόμενο zoom in το extent του χάρτη θα είναι το 60% του προηγούμενου extent.

nrDecimals (number, integer): ο αριθμός των δεκαδικών που θα εμφανίζεται ή θα χρησιμοποιηθεί για την ψηφιοποίηση (δεν χρησιμοποιείται)

epsg (number, integer): Η epsg προβολή του χάρτη. Με δεδομένο ότι δεν χρησιμοποιούμε προβολή του χάρτη ο αριθμός αυτός μπορεί να είναι οποιοσδήποτε.

Units (string): Περιγράφει τη μονάδα μέτρησης του χάρτη (“m” για τα μέτρα που χρησιμοποιούμε εμείς)

unitsFactor (number): Συντελεστής που χρησιμοποιείται για μετατροπή συντεταγμένων όταν αυτές προβάλλονται κατά την κίνηση του mouse. Για παράδειγμα μπορούμε να τον χρησιμοποιήσουμε για μετατροπή από πόδια σε μέτρα και αντίστροφα. Τίθεται σε 1 εφόσον δε θέλουμε να χρησιμοποιήσουμε διαφορετικές συντεταγμένες.

showCoords (Boolean): true|false, Καθορίζει εάν θα χρησιμοποιείται η συνάρτηση showCoordinates για την προβολή των συντεταγμένων με την κίνηση του mouse.

coordXId (string): id του text element για την προβολή των x τιμών

coordYId (string): id του text element για την προβολή των y τιμών

dynamicLayers (Array of strings): πίνακας που περιέχει σύνολο ids των δυναμικών επιπέδων (layers) που θα φορτώνονται.

digiLayers (Array of strings): πίνακας που περιέχει ψηφιοοποιημένα επίπεδα (κενός σε εμάς)

active DigiLayer (string): το id του τρέχοντος digitizing layer (κενό σε εμάς)

zoomRectAttribs (Πίνακας που περιλαμβάνει γνωρίσματα που αφορούν τον τρόπο προβολής): Τα γνωρίσματα αυτά καθορίζουν την εμφάνιση του παραλληλογράμμου zoom που χρησιμοποιείται στο manual zooming. Το stroke-width και το stroke-dasharray ορίζονται σε σχέση με το width του υφιστάμενου map, ενώ το συντακτικό του stroke-dasharray είναι πολύ αυστηρό. Για παράδειγμα var zoomRectstyles = { “fill”:”none”,“stroke”:0.002,“stroke-dasharray”:“0.012,0.002,"stroke-dasharray" : "0.012, 0.002"}

highlightAttribs (Πίνακας που περιέχει γνωρίσματα που αφορούν τρόπο εμφάνισης): Τα γνωρίσματα αυτά αφορούν τη μορφή των croshair γραμμών για την κατάδειξη των point features. Θα πρέπει κατ’ ελάχιστο να περιλαμβάνει τις μεταβλητές stroke και stroke-width οι οποίες ορίζονται σε σχέση με το width του υφιστάμενου map, ενώ το συντακτικό του stroke-dasharray είναι πολύ αυστηρό. Για παράδειγμα var highlightStyles = {"stroke":"crimson","stroke-width":0.002}

dragRectAttribs (Πίνακας που περιέχει γνωρίσματα που αφορούν τρόπο εμφάνισης): Τα γνωρίσματα αυτά αφορούν τη μορφή του παραλληλογράμμου που εμφανίζεται όταν κάνουμε drag στο χάρτη και δείχνει το τρέχων extent του χάρτη στον χάρτη αναφοράς. Θα πρέπει κατ’ ελάχιστον να περιλαμβάνει τις μεταβλητές stroke και stroke-width οι οποίες ορίζονται σε σχέση με το width του υφιστάμενου map, ενώ το συντακτικό του stroke-dasharray είναι πολύ αυστηρό. Για παράδειγμα var dragRectStyles = {"fill":"lightskyblue","fill-opacity":0.5}.

refnapName (string): Το id του χάρτη αναφοράς (reference map).

dragSymbol (string): Το id του στοιχείου που περιγράφει το σύμβολο που εμφανίζεται όταν ο χρήστης κάνει πολύ μεγάλο zoom in ξεπερνώντας κάποιο όριο

symbolThreshold (number, viewbox units): το όριο, οριζόμενο σε μονάδες viewbox (πλάτος του παραλληλογράμμου drag). Στην περίπτωση που το drag παραλληλόγραμμο είναι μικρότερο από αυτό το όριο τότε εμφανίζεται το dragSymbol.

Στη γραμμή 63 καλείται η zoomSlider. Το slider αντικείμενο εισάγεται ως property στο αντικείμενο myMapApp ώστε να αποφευχθεί ο ορισμός περισσότερων καθολικών μεταβλητών. Για τον ορισμό του αντικειμένου slider απαιτούνται οι παρακάτω παράμετροι:

x1: x coordinate of the slider start point

y1: y coordinate of the slider start point

value1: the value of the slider start point (the minimum width at maximal zoom in)

x2: x coordinate of the slider end point

y2: y coordinate of the slider end point

value2: the value of the slider end point (the maximum width at full view)

startVal: the slider start value (in our case myMainMap.origWidth)

sliderGroupId: the id of the group where we will place the slider (in our case mapZoomSlider, part of the group containing the navigation tools)

sliderColor: the color of the slider line (defined verbally or in rgb)

visSliderWidth: the width of the slider line

invisSliderWidth: the width of an invisible slider line receiving the slider events (for very thin slider lines, the width should be considerably bigger)

sliderSymb: the id of a slider symbol (in our case "sliderSymbol")

functionToCall: the function or object to be called when the slider value was changed (in our case the "myMapApp.refMapDragger" object instance)

mouseMoveBool: a boolean flag indicating if we want instant feedback while moving the slider or no (in our case true).

Στη συνέχεια ορίζονται τα απαραίτητα για το interface με το χρήστη buttons. Τα στοιχεία αυτά χρησιμοποιούν το button object. Η σχετική με τα αντικείμενα buttons βιβλιογραφία βρίσκεται στη διεύθυνση http://www.carto.net/papers/svg/gui/button/.

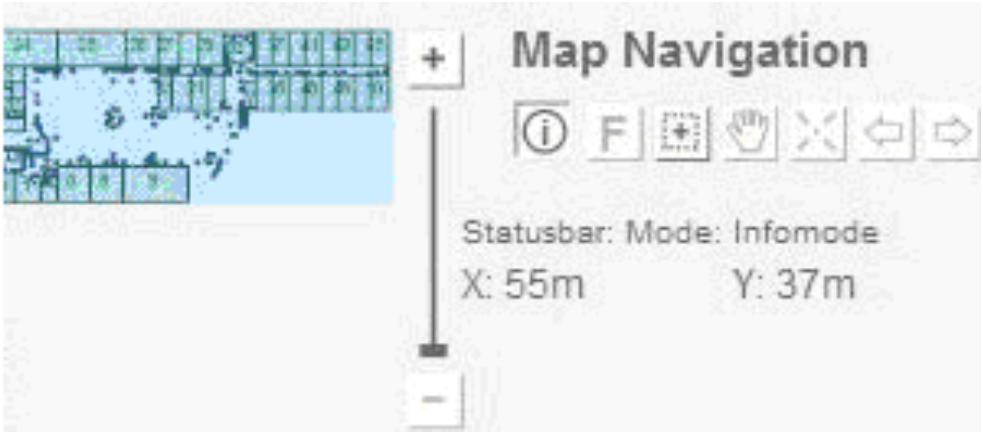
Η μορφή των buttons μπορεί να τροποποιηθεί με χρήση presentation attributes ή CSS στυλ. Οι ορισμοί των συμβόλων γίνονται στα αντίστοιχα τμήματα <symbols /> (που είναι

μέσα στο τμήμα <defs />). Επισημαίνεται ότι μπορούμε να μεταβάλλουμε το στυλ, τη μορφή και τη θέση των buttons αλλά όχι τις σχετικές με αυτά συναρτήσεις και τα id αυτών αλλιώς οι συναρτήσεις πλοήγησης δε θα λειτουργούν.

Στη γραμμή 66, με την κλήση της myMapApp.buttons = new Array();, δημιουργείται ένας πίνακας στο αντικείμενο myMapApp ο οποίος κρατά όλες τις αναφορές στα buttons zoom και pan.

Μετά τη δημιουργία των buttons καλούμε τη συνάρτηση checkButtons του αντικειμένου myMainMap. Η συνάρτηση αυτή αναλύει τα extents του υφιστάμενου χάρτη και ενεργοποιεί/απενεργοποιεί τα buttons. Σε μερικές περιπτώσεις τα buttons δε θα πρέπει να είναι ενεργά. Για παράδειγμα όταν ο χάρτης εμφανίζεται σε πλήρη προβολή (δεν έχει γίνει zoom), τα “zoomOut”, “panManual” και “recenterMap” είναι απενεργοποιημένα. Η υπόψη συνάρτηση καλείται μετά από κάθε μεταβολή των ορίων (extents) του χάρτη.

Τέλος στη γραμμή 109 καλείται η συνάρτηση loadProjectSpecific. Η ίδια συνάρτηση καλείται πάντα μετά από επανακαθορισμό του ορίων του χάρτη (π.χ. στις περιπτώσεις zoom, pan). Με αυτή τη συνάρτηση προσδιορίζουμε το τι πρέπει να γίνεται σε κάθε αλλαγή των ορίων του χάρτη. Στην περίπτωσή μας επαναπροσδιορίζουμε τους αριθμούς που δηλώνουν το μήκος και πλάτος του map extent και καλούμε τι συνάρτηση για τη δυναμική προβολή των επιθυμητών layers.



Εικόνα 23: Επιλογές Διαχείρισης Χάρτη

Περισσότερες λεπτομέρειες σχετικά με τον τρόπο λειτουργίας των ανωτέρω αντικειμένων μπορεί κάποιος να βρει στην ηλεκτρονική διεύθυνση http://www.carto.net/papers/svg/navigationTools/

4.3.1.3 Προετοιμασία του τμήματος εφαρμογής στον client για δυναμική εισαγωγή δεδομένων

Ο χάρτης ο οποίος τελικά προβάλλεται στον χρήστη αποτελείται από ένα άθροισμα επικαλυπτόμενων επιπέδων (layers). Τα επίπεδα αυτά εισάγονται δυναμικά. Υπάρχουν δύο δυνατές επιλογές για την παραγωγή και εμφάνιση των επιπέδων αυτών. Η μία είναι να ελέγχεται από το πρόγραμμα το ποια layers έχουν επιλεγεί από το χρήστη (ελέγχοντας τα αντίστοιχα checkbuttons) και να πραγματοποιείται μια κλήση στον server ζητώντας το σύνολο των επιλεγμένων κάθε φορά layers. Η άλλη επιλογή, η οποία και ακολουθήθηκε, είναι για το κάθε layer να πραγματοποιείται και μια κλήση με τις κατάλληλες παραμέτρους στο server. Όπως έχει ήδη αναφερθεί, για τις κλήσεις αυτές χρησιμοποιείται η τεχνολογία Ajax. Ένα από τα πλεονεκτήματα που παρέχει η τεχνολογία αυτή είναι και το γεγονός ότι ακόμα και εάν κάποια από τις κλήσεις για κάποιο λόγο δεν είναι δυνατόν να ολοκληρωθεί το πρόγραμμα δεν «κολλάει» περιμένοντας την απάντηση αλλά συνεχίζει να λειτουργεί (χωρίς βέβαια να απεικονίζει αποτέλεσμα από τη συγκεκριμένη κλήση). Για τον ίδιο λόγο επιλέχθηκε η πραγματοποίηση ξεχωριστών κλήσεων στο server ανά layer, έτσι ώστε αν για κάποιο λόγο ο server δεν μπορεί να δημιουργήσει κάποιο επίπεδο, να είναι δυνατή η απεικόνιση των υπολοίπων. Επιπρόσθετα με τον τρόπο αυτό, στις περιπτώσεις όπου δεν αλλάζουμε το extent του χάρτη και απλά ο χρήστης τοποθετεί ή αφαιρεί συγκεκριμένα layers από το χάρτη, ο server καλείται και επιστρέφει μόνο τα layer αυτά και όχι όλο το χάρτη εξοικονομώντας έτσι bandwidth.

Κάθε layer που επιστρέφεται από το server είναι ένα svg αρχείο, το οποίο θα πρέπει να εισαχθεί στο τμήμα της εφαρμογής που υλοποιείται στον client, προκειμένου να απεικονιστεί στο interface του χρήστη. Για το σκοπό αυτό στις γραμμές 387 – 400 ορίζουμε svg τμήματα, με IDs όμοια με αυτά των layers που θέλουμε να εισάγουμε, προκειμένου χρησιμοποιώντας javascript να κάνουμε εισαγωγή των απαντήσεων του server στις συγκεκριμένες θέσεις του svg αρχείου, το οποίο εκτελείται στον client. Με τον τρόπο αυτό μας δίνεται και η δυνατότητα να κάνουμε χρήση και μεθόδων της javascript όπως η onclick ώστε το περιεχόμενο της σελίδας μας να γίνει ακόμη πιο δυναμικό. Ας μην ξεχνάμε ότι το αποτέλεσμα που παράγεται από τον server είναι ένα svg αρχείο, το οποίο όμως δεν περιέχει συναρτήσεις όπως η onclick, onmouseover κλπ. Εάν δεν χρησιμοποιούσαμε έναν «καμβά», όπως αυτόν στις γραμμές 387-400, η εισαγωγή των ανωτέρω συναρτήσεων θα έπρεπε να είναι στον server με χρήση

συναρτήσεων php που ελέγχουν το DOM Document, κάτι που είναι αρκετά δύσχρηστο και δύσκολο.

Προκειμένου να καταστεί εφικτή η εισαγωγή των layers, που λαμβάνονται από στο server, στο SVG αρχείο του client, στη συνάρτηση αρχικοποίησης (init) για κάθε ζητούμενο layer εισάγουμε και μια αντίστοιχη τιμή στον πίνακα dynamicLayers. Το γνώρισμα “key” αποτελεί το όνομα του layer, το “value” καθορίζει εάν το layer θα « φορτώνεται» ή όχι και το “loaded” καθορίζει εάν το layer έχει ήδη φορτωθεί.

Εικόνα 24: Επιλογή Layer

Επίσης χρησιμοποιούμε και την παράμετρο timestamp, η οποία προστίθεται στο αντικείμενο myMainMap, προκειμένου να ελέγχεται καλύτερα η διαδικασία «φόρτωσης» των δεδομένων, στις περιπτώσεις που ο χρήστης κάνει zoom ή pan πολύ γρήγορα. Για την ακρίβεια η παράμετρος αυτή χρειάζεται μόνο εάν ο χρήστης εκτελεί τις παραπάνω ενέργειες πιο γρήγορα από το χρόνο που απαιτείται για να ληφθούν από τον server τα δεδομένα που ζήτησε ο client. Με κάθε κλήση προς το server στέλνεται ως παράμετρος και η μεταβλητή timestamp (η οποία τροποποιείται με κάθε κλήση). Κάθε φορά που ο χρήστης κάνει zoom ή pan η timestamp μεταβάλλεται. Εάν ο χρήστης κάνει pan ή zoom πριν ληφθούν τα δεδομένα που ζητήθηκαν προηγουμένως τότε τα δεδομένα αυτά δεν προστίθενται στο document tree αλλά απλά αγνοούνται. Τα δεδομένα αυτά προστίθενται στο DOM tree μόνο εφόσον το timestamp τους είναι ίδιο με το τρέχων timestamp. Η timestamp δημιουργείται στις γραμμές 116-117, όπου παίρνει την τρέχουσα τιμή ημερομηνίας και ώρας (εκφρασμένη σε msec από την ημερομηνία

1/1/1970). Στη γραμμή 123 καλούμε τη συνάρτηση getLayers προκειμένου να λάβουμε τα ζητούμενα layers. Επισημαίνεται ότι η loadProjectSpecific, πέραν της αρχικής κλήσης της κατά την έναρξη της εφαρμογής, καλείται με κάθε zoom ή pan του χρήστη.

4.3.1.4 Δυναμική εισαγωγή δεδομένων

Για τη δυναμική εισαγωγή των δεδομένων (layers) χρησιμοποιούνται οι συναρτήσεις getLayers(), getSingleLayer(layerId), createXmlHttpRequestObject() και η handleResults(node).

Η συνάρτηση getLayers() εκτελεί ένα βρόχο μέσα στον οποίο ελέγχει όλα τα dynamicLayers (που είναι ένα γνώρισμα τύπου πίνακα του αντικειμένου myMainMap). Εφόσον η “value” του τρέχοντος dynamicLayer είναι “yes”, καλείται η getSingleLayer(layerId) ώστε να γίνει μια κλήση στο server για ανάκτηση του συγκεκριμένου layer. Πριν γίνει αυτό όμως ελέγχουμε εάν υπάρχει καταχώρηση για το υφιστάμενο timestamp στον πίνακα myMainMap.nrLayerToLoad. Ο πίνακας αυτός κρατά τον αριθμό των layers που θα φορτωθούν με το τρέχον timestamp. Αν η ανωτέρω καταχώρηση δεν υπάρχει δημιουργείται, ενώ αν υπάρχει ήδη (στις περιπτώσεις που φορτώνονται πάνω από ένα layer) αυξάνεται η τιμή της. Για παράδειγμα εάν υπάρχει απαίτηση να φορτωθούν 3 layers η τιμή της myMainMap.nrLayerToLoad[myMainMap.timestamp.toString()] θα είναι 3.

Στη συνέχεια στη γραμμή 144 ελέγχουμε εάν η τιμή είναι 1 οπότε εμφανίζουμε ένα text element το οποίο γράφει «loading data», που σημαίνει ότι γίνεται κλήση στο server για ανάκτηση δεδομένων.

Η συνάρτηση getSingleLayer(layerId) λαμβάνει ως παράμετρο το όνομα του layer το οποίο θέλουμε να ανακτήσουμε από το server. Η μεταβλητή myUrlString, στη γραμμή 155, περιέχει την κλήση της συνάρτησης στον server καθώς και όλες τις απαιτούμενες παραμέτρους για την κλήση αυτή. Οι εν λόγω παράμετροι είναι:

layername: Το όνομα του layer που θέλουμε να λάβουμε από το server

timestamp: Το timestamp της κλήσης που κάνουμε, το οποίο στη συνέχεια ο server ενσωματώνει στο αποτέλεσμα που επιστρέφει, ώστε να ξέρουμε σε ποια κλήση αναφέρεται η απάντηση που λαμβάνουμε

xmin,ymin: Η τιμή (x,y) της κάτω αριστερής γωνίας του παραθύρου θέασης που έχει επιλέξει ο χρήστης. Το παράθυρο θέασης αρχικά είναι όλος ο χάρτης ενώ στις περιπτώσεις που έχει γίνει zoom είναι το τμήμα του χάρτη που ο χρήστης έχει ζητήσει να γίνει μεγέθυνση. Το παράθυρο αυτό δίδεται μέσω των γνωρισμάτων του myMainMap

και είναι εκφρασμένο σε πραγματικές διαστάσεις (μέτρα). Σημειώνεται ότι το σημείο (myMainMap.curxOrig, mayMainMap. curyOrig), το οποίο εφόσον δεν έχει γίνει zoom είναι το (0,0) είναι η πάνω αριστερή γωνία του παραθύρου θέασης. Όμως το σημείο (0,0) για το layer που δημιουργεί και αναγνωρίζει ο mapserver είναι η κάτω αριστερή γωνία. Για το λόγο αυτό η μεταβλητή ymin αποστέλλεται στον server ανεστραμμένη. Για τον υπολογισμό της παίρνουμε την myMainMap.curyOrig προσθέτουμε το ύψος του παραθύρου θέασης και πολλαπλασιάζουμε επί -1.

xmax,ymax: Το σημείο πάνω δεξιά του παραθύρου θέασης. Για τον υπολογισμό του xmax προσθέτουμε στο myMainMap.curxOrig το πλάτος του παραθύρου θέασης και για το ymax απλά πολλαπλασιάζουμε επί -1 το myMainMap.curyOrig.

Από τη γραμμή 156 ξεκινά η κλήση προς το server χρησιμοποιώντας τεχνολογία Ajax.

4.3.1.4.1 Κλήσεις Server με χρήση AJAX

Στη γραμμή 156 καλείται η συνάρτηση createXmlHttpRequestObject() η οποία ορίζεται στη γραμμή 189.

```
function createXmlHttpRequestObject()
        {       //αποθήκευση του δείκτη στο αντικείμενο XMLHttpRequest
                var xmlHttp;
                //για όλους τους browsers εκτός του IE6 και παλιότερων
                try
                {       //δημιουργία του XMLHttpRequest αντικειμένου
                        xmlHttp = new XMLHttpRequest();
                }
                catch(e)
                {       //σε περίπτωση του IE6 και παλιότερων
                        var XmlHttpVersions = new Array("MSXML2.XMLHTTP.6.0",
                        "MSXML2.XMLHTTP.5.0",
                        "MSXML2.XMLHTTP.4.0",
                        "MSXML2.XMLHTTP.3.0",

                        "MSXML2.XMLHTTP",
                        "Microsoft.XMLHTTP");
                        //δοκιμάζω κάθε version έως ότου κάποια δουλέψει
                        for (var i=0; i<XmlHttpVersions.length && !xmlHttp; i++)
                        {
```

```
                    try
                    {   //δοκίμασε να δημιουργήσεις το αντικείμενο XMLHttpRequest
                        xmlHttp = new ActiveXObject(XmlHttpVersions[i]);
                    }
                    catch (e) {}
                }
            }   // επιστροφή του αντικειμένου που δημιουργήθηκε ή μήνυμα λάθους
            if (!xmlHttp)
                alert("Error creating the XMLHttpRequest object.");
            else
                return xmlHttp;
        }
```

Η ανωτέρω συνάρτηση έχει δημιουργηθεί έτσι ώστε να μπορεί να λειτουργήσει ανεξαρτήτως browser. Στον Internet Explorer 6 και σε παλαιότερες από αυτόν εκδόσεις η XMLHttpRequest υλοποιείται ως ActiveX control το οποίο αρχικοποιείται ως εξής:

xmlhttp= new ActiveXObject (“Microsoft.XMLHttp”);

Για όλους του άλλους browsers η XMLHttpRequest είναι γηγενές (native) αντικείμενο οπότε μπορούμε να το δημιουργήσουμε ως εξής:

xmlhttp= new XMLHttpRequest();

Χρησιμοποιώντας τη δομή try/catch δημιουργούμε το αντικείμενο XMLHttpRequest() καλώντας την new XMLHttpRequest(); και σε περίπτωση που η Javascript επιστρέψει error (exception), το οποίο συμβαίνει όταν ο browser είναι IE 6 ή παλαιότερης έκδοσης, οδηγούμαστε στο τμήμα του κώδικα που θα χειριστεί το exception αυτό (τμήμα catch). Σε περίπτωση που δεν προκύψει exception το τμήμα catch δε θα εκτελεστεί ποτέ. Σε περίπτωση που απαιτηθεί να χειριστούμε το exception απλά δοκιμάζουμε σε κάθε version του IE να δημιουργήσουμε το αντικείμενο XMLHttpRequest(). Τελικά στη γραμμή 220 ελέγχουμε για την επιτυχή δημιουργία του αντικειμένου αυτού και είτε επιστρέφεται μήνυμα σφάλματος είτε επιστρέφεται το ίδιο το αντικείμενο.

Το παραπάνω αντικείμενο αποθηκεύεται στη μεταβλητή xmlHttp. Στη συνέχεια στη γραμμή 157 καλείται η συνάρτηση open του αντικειμένου xmlHttp. Σε κάθε κλήση προς το server χρησιμοποιούνται οι συναρτήσεις open και send. Η open αρχικοποιεί/διαμορφώνει την κλήση θέτοντας διάφορες παραμέτρους και η send εκτελεί

την κλήση (αποκτά πρόσβαση στο server). Όταν η κλήση γίνεται ασύγχρονα όπως στην περίπτωσή μας, πριν από την κλήση της send απαιτείται να αρχικοποιήσουμε το γνώρισμα onreadystatechange με μια συνάρτηση επιστροφής (callback method) η οποία εκτελείται όταν η κατάσταση (status) της κλήσης (request) προς το server αλλάξει. Η συνάρτηση αυτή σε εμάς είναι η handleHttpReturnedData() που ουσιαστικά λαμβάνει το svg αρχείο από το server. Η συνάρτηση open δέχεται απαραιτήτως ως ορίσματα δύο παραμέτρους ενώ προαιρετικά μπορεί να δεχθεί και επιπλέον ορίσματα. Τονίζεται ότι η open δεν ξεκινά τη σύνδεση με το server αλλά θέτει απλά τις παραμέτρους της σύνδεσης. Η πρώτη παράμετρος που δέχεται προσδιορίζει τη μέθοδο που θα χρησιμοποιηθεί για την αποστολή δεδομένων στο server. Η μέθοδος αυτή μπορεί να είναι μία εκ των GET, PUT ή POST. Η μέθοδος POST είναι απαραίτητη στην περίπτωση που στέλνονται δεδομένα χωρητικότητας μεγαλύτερης των 512 bytes. Η δεύτερη παράμετρος είναι η URL, η οποία καθορίζει που θα σταλεί η κλήση. Η URL μπορεί να είναι απόλυτη / πλήρης ή σχετική. Η τρίτη παράμετρος της open ονομάζεται async και καθορίζει εάν η κλήση είναι ασύγχρονη. Εφόσον οριστεί ως true σημαίνει ότι το πρόγραμμα θα συνεχίσει κανονικά την εκτέλεσή του μετά την κλήση της send χωρίς να περιμένει την απάντηση (response) από το server. Εφόσον η τιμή αυτή οριστεί ως false τότε το πρόγραμμα (η ιστοσελίδα) θα παγώσει έως ότου λάβει την απάντηση από το server. Στην περίπτωσή μας ζητάμε να εκτελεστεί η λειτουργία “get” με url όπως έχει αναφερθεί νωρίτερα και τα δεδομένα να ληφθούν από το server ασύγχρονα.

Όπως προαναφέρθηκε μετά την κλήση της send στη γραμμή 159 το πρόγραμμα δεν παγώνει αλλά συνεχίζει να εκτελείται κανονικά. Η συνάρτηση handleHttpReturnedData() αναλαμβάνει να χειριστεί τις αλλαγές της κατάσταση κλήσης. Η συνάρτηση αυτή καλείται 4 φορές, μία για κάθε αλλαγή κατάστασης. Σημειώνεται ότι το γνώρισμα readyState του αντικειμένου xmlHttprequest παίρνει τις παρακάτω τιμές:

- 0 = unintialized
- 1 = loading
- 2 = loaded
- 3 = interactive
- 4 = complete

Η κατάσταση interactive είναι μια ενδιάμεση κατάσταση όταν η απάντηση (response) έχει μερικώς ληφθεί. Οι άλλες καταστάσεις είναι προφανείς. Στην εφαρμογή μας

χρησιμοποιούμε μόνο την κατάσταση 4 που σημαίνει ότι η απάντηση από το server έχει ληφθεί.

Για το σκοπό αυτό στη γραμμή 162 ελέγχουμε εάν είμαστε στην κατάσταση 4 και στη συνέχεια εφόσον το status της Http είναι «OK» (τιμή 200) λαμβάνουμε την απάντηση του server στη μεταβλητή xmlResponse. Σημειώνεται ότι, δεδομένου ότι το svg αρχείο είναι ένα xml αρχείο, επιλέξαμε να χρησιμοποιήσουμε τη συνάρτηση xmlHttp.responseXML και όχι την xmlHttp.responseText. Επίσης αναφέρεται ότι ακριβώς επειδή θέλουμε να λάβουμε το svg αρχείο κατευθείαν από το server στη μορφή xml ορίσαμε και στην php στο server ότι η απάντηση που θα αποστείλει θα είναι xml (header('content-Type: image/svg+xml);), ενώ το output από το server αποθηκεύεται με τη συνάρτηση $output=$doc->saveXML(); και επιστρέφεται στον client με την echo $output;. Το αποτέλεσμα είναι στον client να παίρνουμε την απάντηση του server ως ένα DOM document.

Στις γραμμές 169-174 γίνεται, ανάλογα με τον browser που χρησιμοποιούμε, έλεγχος του αποτελέσματος για τυχόν σφάλματα στη δομή του και στην περίπτωση που το DOM document είναι ορθά δομημένο ως XML αρχείο, το αποθηκεύουμε στη μεταβλητή xmlRoot, η οποία πλέον κρατά το root του svg αρχείου. Τέλος καλείται η handleResults(node) η οποία πλέον θα επεξεργαστεί αυτό το svg αρχείο, εντάσσοντας το στην κατάλληλη θέση στη σελίδα του client. Στην περίπτωση που δεν ληφθεί απάντηση από τον server τότε στις γραμμές 180-184 επιστρέφεται ως μήνυμα λάθους το «αδυναμία σύνδεσης με το server.»

4.3.1.4.2 Εισαγωγή των δεδομένων του server στην εφαρμογή

Με τη συνάρτηση handleResults(node) αρχικά αποθηκεύουμε στη μεταβλητή currentDynamicLayer το όνομα του layer που έχουμε λάβει, ενώ στη μεταβλητή timestamp την τιμή timestamp αυτού. Η τιμή αυτή συγκρίνεται με το τρέχον timestamp. Στην περίπτωση που οι τιμές δεν συμφωνούν μεταξύ τους (που σημαίνει ότι ο χρήστης κάνει zoom ή pan πιο γρήγορα από ότι λαμβάνει δεδομένα από το server) το layer που απεστάλη από το server αγνοείται. Στην περίπτωση που αυτές οι τιμές συμφωνούν θα πρέπει τα δεδομένα που παραλήφθηκαν από το server να ενσωματωθούν στο svg interface του client. Για να γίνει αυτό βρίσκουμε, στη γραμμή 237, με βάση το όνομα του layer τη θέση στην οποία το νέο layer θα γίνει append. Στη γραμμές 384-402 όπως έχει ήδη αναφερθεί έχουμε δημιουργήσει τον καμβά μέσα στον οποίο τοποθετούνται οι απαντήσεις από το server. Ας πάρουμε για παράδειγμα ότι ζητήθηκε το layer με id = "grCor". Μετά την εκτέλεση της γραμμής 237 γνωρίζουμε ότι το layer αυτό θα μπει κάτω

από το < svg id=”grCor” … >. Έτσι αν το κάναμε απλά append και εξετάζαμε το νέο DOM Document που θα προέκυπτε θα βλέπαμε κάτι σαν αυτό:

```
<svg id=”grCor” …. >
        <svg id=”grCor” … >
        </svg>
</svg>
```

Μετά από διαδοχικές κλήσεις κάνοντας zoom, pan κλπ και δεδομένου ότι κάθε φορά θα λαμβάναμε και ένα svg αρχείο με id “grCor”, το DOM Document που θα βλέπαμε θα ήταν:

```
<svg id=”grCor” …. >
        <svg id=”grCor” … >
        <svg id=”grCor” … >
<svg id=”grCor” … >
<svg id=”grCor” … >
…
        </svg>
</svg>
…
</svg>
```

Από τα παραπάνω καθίσταται σαφές ότι θα πρέπει, πριν εισάγουμε το layer που ζητήσαμε στην θέση του στο DOM document, να ελέγχουμε εάν ήδη υπάρχει κάτι εκεί και να το διαγράφουμε. Αυτό ακριβώς γίνεται στις γραμμές 242-250.

```
if (myMainMap.dynamicLayers[currentDynamicLayer].loaded=="yes")
                        {
                                if (startingNodetoAppend.firstChild)
                                {

        previousDataToremove=startingNodetoAppend.firstChild;

        startingNodetoAppend.removeChild(previousDataToremove);
                                }
                        }
```

Μετά τον παραπάνω έλεγχο και τη διαγραφή προηγούμενων svg τμημάτων εισάγουμε τα νέα δεδομένα στη θέση που πρέπει στο DOM document (γραμμή 252). Οι επόμενες γραμμές (254-255) ουσιαστικά «ευθυγραμμίζουν» την αρχή των αξόνων του svg τμήματος που εισαγάγαμε με την αρχή των αξόνων του παραθύρου θέασης, έτσι ώστε

να μπορούν τα δεδομένα που λάβαμε να απεικονιστούν στο παράθυρο θέασης που έχει επιλέξει ο χρήστης (έπειτα από κάποιο zoom για παράδειγμα). Στη συνέχεια θέτουμε την παράμετρο loaded σε “yes” προκειμένου να δείξουμε ότι το συγκεκριμένο layer έχει απεικονισθεί. Τέλος, η μεταβλητή myMainMap.nrLayerToLoad [myMainMap.timestamp .toString()] μειώνεται και όταν φθάσει στην τιμή 0 θέτουμε την παράμετρο visibility του text element “loading data” σε off. Αυτό σημαίνει ότι όλα τα ζητηθέντα layers έχουν φορτωθεί. Επισημαίνεται ότι λόγω των ασύγχρονων κλήσεων προς το server δεν υπάρχει καμία εγγύηση ότι τα δεδομένα επιστρέφουν από το server με την ίδια σειρά με την οποία πραγματοποιήθηκαν και οι κλήσεις.

Προκειμένου να ενημερωθεί ο χρήστης ότι τα δεδομένα που ζήτησε φορτώνονται από το server προστέθηκε το group element με id “loadingData”, το οποίο εμφανίζεται όταν φορτώνονται τα δεδομένα και παύει να εμφανίζεται όταν φορτωθούν όλα τα δεδομένα. Το στοιχείο αυτό δημιουργείται στις γραμμές 405-409 όπως φαίνεται παρακάτω:

```
<!-- Loading Data Status -->
    <g id="loadingData" visibility="hidden">
        <rect x="200" y="300" width="150" height="35" fill="white" fill-opacity="0.8"/>
        <text x="275" y="325" font-family="sans-serif" fill="dimgray" font-size="18px" font-weight="bold" text-anchor="middle">Loading Data ...</text>
    </g>
```

4.3.1.4.3 Προσθήκη χαρακτηριστικών επιλογής layers στην εφαρμογή

Προκειμένου να μπορεί ο χρήστης να ελέγχει ποια layers θα εμφανίζονται κάθε φορά χρησιμοποιούμε checkboxes και τη συνάρτηση toggleMapLayer (id, checkStatus, labelText). Για κάθε layer προσθέτουμε και ένα checkbox στη συνάρτηση init. Πληροφορίες για τη βιβλιοθήκη των checkboxes βρίσκει κανείς στη διεύθυνση http://www.carto.net/papers/svg/gui/checkbox_and_radiobutton/. Ακόμη προσθέτουμε και ένα checkbox για την εμφάνιση του layer της διαδρομής που έχει επιλέξει ο χρήστης να ακολουθήσει στο χάρτη. Επιλέγοντας αυτό το checkbox ουσιαστικά πληροφορούμε τον mapserver να διαβάσει το αντίστοιχο text αρχείο που περιλαμβάνει τα σημεία από τα οποία απαρτίζεται η διαδρομή και να φτιάξει το σχετικό layer.

Επιπρόσθετα χρησιμοποιούμε και τη συνάρτηση toggleMapLayer(id,checkStatus, labelText) με την οποία θέτουμε το visibility ενός layer σε on ή off.

Ως ένα τελευταίο χαρακτηριστικό έχουμε προσθέσει τη συνάρτηση

```
function msclick(evt)
        {
                sth=evt.target;
                alert (sth.getAttribute("id"));
        }
```

Η συνάρτηση αυτή σχετίζεται με την onclick συνάρτηση που έχουμε ορίσει στον καμβά που έχουμε για τη δυναμική εισαγωγή των layers και χρησιμοποιείται προκειμένου να μας εμφανίζει το id του σχήματος πάνω στο χάρτη στο οποίο ο χρήστης κάνει click. Ουσιαστικά η συνάρτηση αυτή εισήχθηκε για να δείξει ότι μπορούμε να αυξήσουμε τις δυνατότητες της εφαρμογής προσθέτοντας επιπλέον δυνατότητες που η javascript μας παρέχει (π.χ. tooltips, onmouseover κλπ).

Εν κατακλείδι το σύνολο των αρχείων με τις αντίστοιχες διαδρομές τους καταγράφονται στον παρακάτω Πίνακας 5:

Αρχείο	Διαδρομή	Σχόλια
University.html	C:\ms4w\Apache\htdocs\UniversityFinal	
UniversityFinal.svg	C:\ms4w\Apache\htdocs\UniversityFinal	Αρχείο client
button.js	C:\ms4w\Apache\htdocs\UniversityFinal	Βοηθητικό αρχείο client
checkbox_and_radiobutton.js	C:\ms4w\Apache\htdocs\UniversityFinal	Βοηθητικό αρχείο client
helper_functions.js	C:\ms4w\Apache\htdocs\UniversityFinal	Βοηθητικό αρχείο client
mapApp.js	C:\ms4w\Apache\htdocs\UniversityFinal	Βοηθητικό αρχείο client
navigation.js	C:\ms4w\Apache\htdocs\UniversityFinal	Βοηθητικό αρχείο client
selectionList.js	C:\ms4w\Apache\htdocs\UniversityFinal	Βοηθητικό αρχείο client

slider.js	C:\ms4w\Apache\htdocs\UniversityFinal	Βοηθητικό αρχείο client
timer.js	C:\ms4w\Apache\htdocs\UniversityFinal	Βοηθητικό αρχείο client
getUniversityMap.php	C:\ms4w\Apache\htdocs\UniversityFinal	Αρχείο server
UniversityFinal.map	C:\ms4w\Apache\htdocs\UniversityFinal	Βοηθητικό αρχείο mapserver/server
universitysmall.png	C:\ms4w\Apache\htdocs\UniversityFinal	Εικόνα αναφοράς στον client
path.txt	C:\ms4w\Apache\htdocs\UniversityFinal	Αρχείο εισαγωγής διαδρομής (route) στο χάρτη
fontset.txt	C:\ms4w\Apache\htdocs	Ορισμός γραμματοσειρών για το mapserver
arial.ttf	C:\ms4w\Apache\htdocs	Γραμματοσειρά
arialbd.ttf	C:\ms4w\Apache\htdocs	Γραμματοσειρά
Αρχεία Shapefiles κάτοψης	C:\Mapdata\finalUniversityData\Ground	Δεδομένα εισόδου στο mapserver

Πίνακας 5: Αρχεία εφαρμογής και διαδρομές τους

ΒΙΒΛΙΟΓΡΑΦΙΚΕΣ ΑΝΑΦΟΡΕΣ

[i] T. B Lee et al., “The semantic web,” *Scientific American* 284, no. 5 (2001): 34–43; N. Shadbolt, W. Hall, and T. Berners-Lee, “The semantic web revisited,” *Intelligent Systems, IEEE* 21, no. 3 (2006): 96–101.

[ii] “OWL 2 Web Ontology Language Document Overview,” http://www.w3.org/TR/owl2-overview/.

[iii] “Extensible Markup Language (XML),” http://www.w3.org/XML/.

[iv] “Resource Description Framework (RDF): Concepts and Abstract Syntax,” http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/.

[v] Franz Baader et al., *The Description Logic Handbook. Theory, implementation and applications* (Cambridge University Press, 2003).

[vi] “Oracle Spatial, Locator, and Location-Based Services,” http://www.oracle.com/technetwork/database/options/spatial/index.html.

[vii] “PostGIS : Home,” http://postgis.refractions.net/.

[viii] “PostgreSQL: The world's most advanced open source database,” http://www.postgresql.org/.

[ix] “Location Service (OpenLS) | OGC(R),” http://www.opengeospatial.org/standards/ols.

[x] B. Kropla, *Beginning MapServer: Open Source GIS Development*, 2005th ed. (N. York: Apress).

[xi] M. Tyler, *Web Mapping Illustrated* (O'Reilly, 2005).

[xii] “Welcome to MapServer — MapServer 5.6.6 documentation,” http://mapserver.org/.

[xiii] “MS4W.MapTools.org,” http://maptools.org/ms4w/.

[xiv] Darie, C., et al., *AJAX and PHP Building Responsive Web Applications* (Birmingham, UK: Packt Publishing, 2006).

[xv] L. Babin, *Beginning Ajax with PHP From Novice to Professional* (N. York: Apress, 2007).

[xvi] C. Heilmann, *Beginning JavaScript with DOM Scripting and Ajax From Novice to Professional* (Apress, 2006).

[xvii] J. Herrington, *PHP Hacks*, 2005.

[xviii] R. Richards, *PHP XML and Web Services* (Apress, 2006).

[xix] J.D. Eisenberg, *SVG Essentials* (O'Reilly, 2002).

[xx] “carto:net - cartographers on the net,” http://www.carto.net/.