ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΑΓΡΟΝΟΜΩΝ ΚΑΙ ΤΟΠΟΓΡΑΦΩΝ ΜΗΧΑΝΙΚΩΝ
ΤΟΜΕΑΣ ΤΟΠΟΓΡΑΦΙΑΣ

«Ανάπτυξη Υπηρεσιών Προστιθέμενης Αξίας με Αξιοποίηση Web Services σε Περιβάλλον Ανοικτού Κώδικα στο Διαδίκτυο»

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΜΑΡΙΑ Γ.ΣΤΡΙΛΙΓΚΑ

ΟΚΤΩΒΡΙΟΣ 2011

Επιβλέπων Καθηγητής: Βεσκούκης Βασίλειος

ΠΕΡΙΛΗΨΗ

Ο σκοπός της διπλωματικής εργασίας είναι ο συνδυασμός υπηρεσιών που παρέχονται στο διαδίκτυο ως web services με υπολογισμούς που αφορούν περιβαλλοντικές παραμέτρους, προκειμένου να παραχθούν νέα δεδομένα με χωρική αναφορά, χωρίς την προαπαίτηση ύπαρξης ψηφιακών χαρτών, δηλαδή με χρήση μόνο υπηρεσιών που παρέχονται στο internet. Με τον τρόπο αυτό τεκμηριώνεται η δυνατότητα αξιοποίησης των υπηρεσιών που παρέχονται ως υπηρεσίες web για τη δημιουργία νέων δεδομένων με χωρική αναφορά, χωρίς άλλες απαιτήσεις σε χωρικά δεδομένα, πέραν της αξιοποίησης των συγκεκριμένων υπηρεσιών.

Συγκεκριμένα, επιλέχθηκαν δύο υπηρεσίες που προσφέρονται δωρεάν από την Google και αφορούν τον υπολογισμό υψομέτρων και τη δρομολόγηση μεταξύ δύο σημείων, οι οποίες συνδυάστηκαν με μαθηματικό τύπο υπολογισμού θερμιδικής κατανάλωσης ποδηλατών προκειμένου να εμπλουτιστούν οι χάρτες που διατίθενται στο περιβάλλον Google maps με δεδομένα θερμιδικής κατανάλωσης και ευκολίας ποδηλατικών διαδρομών. Για την επίτευξη του σκοπού που αναφέρεται πιο πάνω, αντικείμενο της εργασίας ήταν η ανάπτυξη αλγορίθμου για την απόδοση υψομετρικής πληροφορίας και κατ' επέκταση τον υπολογισμό κλίσεων σε διαδρομές επί οδικών δικτύων σε μία δεδομένη περιοχή με τη χρήση της γλώσσας προγραμματισμού PHP.

Τα αποτελέσματα της εφαρμογής αυτής μπορούν να αξιοποιηθούν από εργαλεία προσομοίωσης περιβαλλοντικής επιβάρυνσης, υπολογισμού έκθεσης σε συγκεντρώσεις ρύπων, καθώς και αξιολόγησης δυσκολίας διαδρομών με διάφορα μέσα (ποδήλατο, πεζοπορία, κ.ά.). Στη συγκεκριμένη περίπτωση η εξαγόμενη πληροφορία (κλίσεις διαδρομών οδικού δικτύου) επιλέχθηκε να αξιοποιηθεί για τον υπολογισμό της κατανάλωσης θερμίδων ενός μέσου ποδηλάτη, επιλογή η οποία έγινε μόνο για λόγους επίδειξης των δυνατοτήτων αξιοποίησης των web services, οι οποίες προφανώς είναι απεριόριστες.

Συγκεκριμένα, δημιουργήθηκαν τυχαία σημεία στα όρια μίας τυχαία επιλεγμένης περιοχής, σχεδιάστηκαν τυχαίες δρομολογήσεις μεταξύ των σημείων αυτών και υπολογίστηκαν οι κλίσεις των τμημάτων των παραγμένων διαδρομών. Η πληροφορία αυτή τροφοδοτεί ένα μοντέλο υπολογισμού της θερμιδοκατανάλωσης ενός μέσου ποδηλάτη για κάθε τμήμα της εκάστοτε διαδρομής. Τα αποτελέσματα απεικονίστηκαν σε διαγράμματα καθώς επίσης και στο Google Earth.

Ο αλγόριθμος αυτός μπορεί να αποτελέσει τη βάση για την υλοποίηση εφαρμογών προσομοίωσης περιβαλλοντικής επιβάρυνσης, υπολογισμού έκθεσης σε συγκεντρώσεις ρύπων, καθώς και αξιολόγησης δυσκολίας διαδρομών με διάφορα μέσα όπως αναφέρθηκε και παραπάνω, αλλά και πολλές άλλες εφαρμογές όπως την αξιολόγηση της ροής και της συγκέντρωσης των βρόχινων υδάτων και την απεικόνιση αυτών.

Λέξεις κλειδιά:
Υπηρεσίες Δικτύου, γλώσσες περιγραφής δεδομένων, τυχαίες διαδρομές, υπολογισμός υψομέτρων, υπολογισμός κλίσεων, θερμιδική κατανάλωση ποδηλάτη, παραγωγή καυσαερίου, Υπολογιστική Ρευστοδυναμική.

ABSTRACT

The purpose of this thesis is the combination of services provided on the internet as Web Services including calculations involving environmental parameters, in order to produce new data with spatial reference, without any prerequisite for digital maps, requiring just services provided on the internet. This is how the possibility of utilizing the services provided as web services to create new data with spatial reference is documented, without any other requirements to spatial data beyond the use of such services.

In particular, two services were selected, which are offered for free by Google and are calculating altitude and routing between two points, which were combined with a mathematical formula for calculating cyclists calorie consumption in order to enrich maps available to Google maps with calorie consumption data and cycling routes convenience. To achieve the purpose mentioned above, the subject of the project was the development of an algorithm in order to determine elevation data and therefore calculate slopes of routes on a road network in a given area using the PHP programming language.

The results of this application can be exploited by simulation tools of environmental burden, calculating the exposure to pollutant concentrations, as well as the evaluation of the path access by various means (cycling, hiking etc.). In this particular case, the exported information (slopes) is being used to calculate the calorie consumption of an average cyclist, an option which was only for the demonstration of the capabilities of web services, which are apparently unlimited.

Specifically, random points were generated near the boundaries of a randomly selected area, random routes were created between these points and finally slopes were calculated for each segment of the generated routes. This information feeds a calorie consumption calculator for an average cyclist for every part of each path. The results were reflected in diagrams and also in Google Earth.

This algorithm can be the basis for the implementation of simulation applications of environmental burden, calculating the exposure to pollutant concentrations, as well as the evaluation of the path access by various meansas mentioned above, but also many other applications, such as the evaluation of rain water flow and concentration, and their illustration.

Keywords:
Web Services, XML, KML, PHP, JSON, Google APIs, Google Directions API, Google Elevation API, random directions, altitude calculation, slope calculation, cycling calorie consumption, exhaust gas production, Computational Fluid Dynamics.

ΕΥΧΑΡΙΣΤΙΕΣ

Πριν την παρουσίαση της παρούσας διπλωματικής εργασίας, αισθάνομαι την υποχρέωση να ευχαριστήσω τους ανθρώπους που κατείχαν πολύ σημαντικό ρόλο κατά υλοποίησή της.

Πρώτο από όλους θα ήθελα να ευχαριστήσω τον επιβλέποντα της διπλωματικής εργασίας, Επίκουρο Καθηγητή Βασίλειο Βεσκούκη για την πολύτιμη συμβολή και καθοδήγησή του, την εμπιστοσύνη και εκτίμηση που μου έδειξε, αλλά και τη συνέπεια και την άψογη συνεργασία που επίσης έδειξε καθ' όλη τη διάρκεια της εκπόνησης της διπλωματικής εργασίας.

Στη συνέχεια θα ήθελα να ευχαριστήσω τον τοπογράφο μηχανικό Ιωάννη Σοφό, ο οποίος βοήθησε και συνέβαλε και αυτός στην περάτωση αυτής της εργασίας.

Τέλος, θέλω να ευχαριστήσω την οικογένειά μου για την ηθική συμπαράσταση που μου παρείχε όλο αυτόν τον καιρό για την ολοκλήρωση της διπλωματικής μου εργασίας.

ΚΑΤΑΛΟΓΟΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΕΙΣΑΓΩΓΗ

Προσέγγιση του Προβλήματος

Το Google Earth καθώς επίσης και άλλα παρεμφερή προγράμματα είναι πολύ χρήσιμα για την εξαγωγή γεωγραφικής πληροφορίας για ένα μέσο χρήστη του Η/Υ. Ωστόσο, η πληροφορία που δίνουν τα προγράμματα αυτά δεν είναι επαρκής για τον σκοπό ενός προγραμματιστή ή ενός ατόμου που αναζητά πιο συγκεκριμένη πληροφορία. Αυτό συμβαίνει διότι λείπει από αυτά η συστηματοποίηση της εξαγωγής δεδομένων αλλά και η αξιοποίηση αυτών σε άλλες πιο εξειδικευμένες εφαρμογές. Αυτός ήταν και ο στόχος της διπλωματικής εργασίας, η δημιουργία ενός αλγορίθμου ο οποίος θα δίνει μαζική γεωγραφική πληροφορία, και θα μπορεί να χρησιμοποιηθεί ως βάση σε άλλες εφαρμογές.

Αντικείμενο της Διπλωματικής Εργασίας

Το αντικείμενο της διπλωματικής εργασίας είναι η απόδοση και η αξιοποίηση υψομετρικής πληροφορίας σε διαδρομές οδικών δικτύων. Στόχος ήταν ο υπολογισμός υψομέτρων και κλίσεων των τμημάτων των διαδρομών ενός δεδομένου οδικού δικτύου. Η εφαρμογή αυτή αποτελεί εφαλτήριο για την αξιοποίηση των αποτελεσμάτων της σε πιο εξειδικευμένες εφαρμογές.

Η εργασία αυτή προσπαθεί να λύσει το πρόβλημα της αυτοματοποίησης της υψομετρικής πληροφορίας για οδικά δίκτυα. Πιο συγκεκριμένα, αφού οριστεί μία περιοχή στο Google Earth, δημιουργούνται τυχαίες διαδρομές που καλύπτουν όλη την περιοχή και στη συνέχεια αφού προσδιοριστούν τα υψόμετρα των σημείων των διαδρομών υπολογίζονται οι κλίσεις των τμημάτων που ορίζουν τα σημεία αυτά σε κάθε διαδρομή. Με τη μέθοδο αυτή, λαμβάνεται η υψομετρική πληροφορία για το σύνολο των δρόμων μίας περιοχής, και όχι για μεμονωμένα σημεία όπως συμβαίνει στο Google Earth, με αυτοματοποιημένο τρόπο. Στο Google Earth τα υψόμετρα συλλέγονται χειροκίνητα και μεμονωμένα. Αποτέλεσε όμως το βασικό εργαλείο μαζί με τις υπηρεσίες της Google (Google Direction και Elevations API) για την περάτωση της εφαρμογής.

Η εφαρμογή αυτή μπορεί να αξιοποιηθεί σε ένα ευρύ φάσμα άλλων εφαρμογών. Στην εργασία αυτή, χρησιμοποιείται για τον υπολογισμό της κατανάλωσης θερμίδων ενός ποδηλάτη σε κάθε τμήμα της διαδρομής ενός οδικού δικτύου συναρτήσει της κλίσης αυτού.

Συνεισφορά

Η συνεισφορά της διπλωματικής εργασίας συνοψίζεται ως εξής:

1. Μελετήθηκε η σχεδίαση της υλοποίησης των αλγορίθμων
2. Υλοποιήθηκαν ο αλγόριθμος, ο οποίος στηρίζεται σε δύο βασικές διεργασίες: στη δημιουργία τυχαίων σημείων εντός της περιοχής που ορίζει ο ενδιαφερόμενος στο Google Earth, και τον υπολογισμό των κλίσεων των τμηματων των διαδρομών και της θερμιδοκατανάλωσης.

3. Αξιολογήθηκε η επίδοση των αλγορίθμων.
4. Εξήχθησαν ανάλογα συμπεράσματα.

Οργάνωση Κειμένου

Συνοπτικά, η διπλωματική εργασία περιλαμβάνει τέσσερα μέρη και συνολικά την εισαγωγή, δώδεκα κεφάλαια, τη βιβλιογραφία και το παράρτημα. Τα κεφάλαια της εργασίας αναπτύσσονται ως εξής:

✓ **Μέρος 1: Web services και συναφείς τεχνολογίες**

Στο πρώτο κεφάλαιο, γίνεται μια γενική περιγραφή των web services.

Στο δεύτερο κεφάλαιο, γίνεται μια πλήρης περιγραφή των γλωσσών περιγραφής δεδομένων XML.

Στο τρίτο κεφάλαιο, περιγράφεται πλήρως η γλώσσα προγραμματισμού PHP και παρατίθενται παραδείγματα για την βαθύτερη κατανόηση της γλώσσας.

Στο τέταρτο κεφάλαιο, γίνεται μια λεπτομερής περιγραφή του πρωτοκόλλου JSON και παρατίθενται αντίστοιχα παραδείγματα.

Στο πέμπτο κεφάλαιο, αναλύονται τα Google APIs και συγκεκριμένα το Directions και το Elevations API, τα οποία και χρησιμοποιούνται για την υλοποίηση της εφαρμογής.

✓ **Μέρος 2: Σχεδίαση και υλοποίηση**

Στο έκτο κεφάλαιο περιγράφονται οι απαιτήσεις από την εφαρμογή που επρόκειτο να υλοποιηθεί.

Στο έβδομο κεφάλαιο πραγματοποιείται περιγραφή της εφαρμογής που υλοποιήθηκε.

✓ **Μέρος 3: Εφαρμογές - Μελέτες περίπτωσης**

Στο όγδοο κεφάλαιο αναλύονται οι τύποι υπολογισμού θερμιδικής κατανάλωσης ποδηλάτη και παραδοχές που έπρεπε να γίνουν.

.

Στο ένατο κεφάλαιο πραγματοποιείται υπολογισμός κλίσεων τυχαίων διαδρομών γύρω από την περιοχή της Πολυτεχνειούπολης Ζωγράφου.

Στο δέκατο κεφάλαιο πραγματοποιείται υπολογισμός θερμιδικής κατανάλωσης τυχαίων ποδηλατικών διαδρομών στην περιοχή του λεκανοπεδίου της Αθήνας.

Στο ενδέκατο κεφάλαιο πραγματοποιείται υπολογισμός θερμιδικής κατανάλωσης τυχαίων ποδηλατικών διαδρομών στην περιοχή του San Francisco.

✓ **Μέρος 4: Συμπεράσματα - συζήτηση**

Στο δωδέκατο κεφάλαιο περιγράφονται προβλήματα και παραδοχές που προέκυψαν κατά την υλοποίηση της εργασίας.

Στο δέκατο τρίτο κεφάλαιο περιγράφονται οι δυνατότητες της εργασίας για άλλες εφαρμογές.

Μέρος 1

Web Services και συναφείς τεχνολογίες

1

WEB SERVICES

Μέχρι πρόσφατα η δημιουργία και η παροχή υπηρεσιών από επιχειρήσεις στο Internet γίνονταν με ακαθόριστο τρόπο ο οποίος διέφερε από επιχείρηση σε επιχείρηση. Έτσι, ενώ υπήρχε ένα αρκετά μεγάλο σύνολο από παρεχόμενες υπηρεσίες στο Internet, για να μπορούσε κάποιος να τις χρησιμοποιήσει θα έπρεπε για κάθε μία υπηρεσία να μελετήσει τον τρόπο με τον οποίο θα την καλέσει, να ελέγξει αν χρησιμοποιούν το ίδιο πρωτόκολλο επικοινωνίας (TCP/IP, Http, κλπ) και γενικά να προσαρμόσει όλο το σύστημά του έτσι ώστε να γίνει συμβατό με αυτό του παροχέα της υπηρεσίας.

Για παράδειγμα ας υποθέσουμε ότι κάποια επιχείρηση ενδιαφερόταν να χρησιμοποιήσει μία υποτιθέμενη υπηρεσία που παρείχε το Εθνικό Κέντρο Βιβλίου και η οποία παρουσίαζε όλες τις συνοδευτικές πληροφορίες (τίτλο, εκδοτικό οίκο, τιμή κλπ) για κάποιο βιβλίο δοθέντος του κωδικού του (ISBN).

Σε αυτή την περίπτωση ο προγραμματιστής της επιχείρησης θα έπρεπε στην ουσία να δημιουργήσει ένα σύστημα συμβατό με αυτό του Εθνικού Κέντρου Βιβλίου και ως προς το πρωτόκολλο επικοινωνίας αλλά και ως προς τον τρόπο κλήσης των ερωτημάτων και κατόπιν να το προσαρμόσει στις ανάγκες του συστήματος της επιχείρησης.

Πολλές φορές αυτό ήταν πολύ δύσκολο – αν όχι ακατόρθωτο – και ακόμα περισσότερες φορές οι επιχειρήσεις σχεδίαζαν τα συστήματά τους έτσι ώστε να αποφεύγουν τέτοιου είδους συνεργασίες με ξένες πηγές για λόγους πολυπλοκότητας και γενικότερα για λόγους κόστους.

Τα πράγματα όμως τα τελευταία τρία χρόνια φαίνεται να παίρνουν διαφορετική τροπή αφού πλέον σχεδόν όλες οι επιχειρήσεις που δημιουργούν υπηρεσίες στο Internet βασίζονται σε μία κοινή αρχιτεκτονική ανάπτυξης, δημοσίευσης και εκμετάλλευσης των υπηρεσιών τους, όπως αυτή καθορίζεται από το W3C[1] και που ορίζεται ως η αρχιτεκτονική των Web Services.

1.1 Πλεονεκτήματα των Web Services

Η αρχιτεκτονική των Web Services παρέχει αρκετά πλεονεκτήματα μερικά από τα οποία αναφέρονται παρακάτω.

[1] Το World Wide Web Consortium (W3C) είναι ο κύριος διεθνής οργανισμός προτύπων για το Παγκόσμιο Διαδίκτυο (World Wide Web).

- ✓ Ευκολότερος χειρισμός δεδομένων. Παραδοσιακά το κυριότερο πρόβλημα στις κατανεμημένες τεχνολογίες ήταν το λεγόμενο tight-coupling ή στα ελληνικά η ισχυρή συνδεδιμότητα. Μια εφαρμογή που καλούσε μια άλλη απομακρυσμένη ήταν αυστηρά δεμένη με αυτή από την κλήση λειτουργίας (function call) που εκτελούσε και τις παραμέτρους που περνούσε. Στα περισσότερα συστήματα πριν από την έλευση των Web Services ο τρόπος επικοινωνίας ήταν μια σταθερή διεπαφή με λίγη έως καθόλου ευελεξία ή προσαρμοστικότητα στα περιβάλλοντα ή τις ανάγκες που μεταβάλλονται συνεχώς.
  Τα Web Services χρησιμοποιούν τη γλώσσα XML η οποία μπορεί να περιγράψει οποιαδήποτε δεδομένα σε ένα πραγματικά ανεξάρτητο από πλατφόρμα τρόπο για ανταλλαγή αυτών των δεδομένων μεταξύ συστημάτων. Με αυτόν τον τρόπο οδηγούμαστε σε εφαρμογές με χαλαρή συνδεσιμότητα (loosely-coupled). Επιπλέον τα Web Services μπορούν να λειτουργήσουν σε πιο αφηρημένο επίπεδο στο οποίο μπορούν να επαναξιολογήσουν, να τροποποιήσουν ή να χειριστούν τύπους δεδομένων δυναμικά κατά περίπτωση. Έτσι σε τεχνικό επίπεδο τα Web Services μπορούν να χειριστούν δεδομένα πολύ ευκολότερα και να επιτρέψουν στο λογισμικό να επικοινωνεί πιο ελεύθερα.

- ✓ Απλότητα πρωτοκόλλου επικοινωνίας. Τα Web Services χρησιμοποιούν ως πρωτόκολλο επικοινωνίας το SOAP. Το πρωτόκολλο αυτό είναι πολύ πιο απλό από πρωτόκολλα παλαιότερων τεχνολογιών όπως αυτά που χρησιμοποιούνταν από τα κατανεμημένα περιβάλλοντα CORBA , DCOM, RPC. Έτσι το να δημιουργήσει κανείς μια υλοποίηση SOAP που υπόκειται στα πρότυπα (standards-compliant) είναι πολύ πιο εύκολο. Σήμερα μπορεί να βρει κανείς υλοποιήσεις του SOAP από τις μεγαλύτερες εταιρίες πληροφορικής αλλά ακόμη και από μεμονωμένους προγραμματιστές, πράγμα αδιανόητο για παλαιότερες κατανεμημένες τεχνολογίες.

- ✓ Απλότητα υποδομής. Τα Web Services λειτουργούν με πρότυπες γλώσσες και πρωτόκολλα όπως η XML , το HTTP και το TCP/IP. Η πλειονότητα των εταιριών έχουν ήδη την δικτυακή υποδομή και τους ανθρώπους με γνώσεις και εμπειρία που τη συντηρούν. Έτσι το κόστος για την εφαρμογή των Web Services είναι σημαντικά μικρότερο από αυτό των προηγούμενων τεχνολογιών.

- ✓ Ευκολία στην επικοινωνία. Με τις προηγούμενες τεχνολογίες η συνεργασία μεταξύ εταιριών ήταν ένα θέμα, διότι κατανεμημένες τεχνολογίες όπως CORBA και DCOM χρησιμοποιούσαν μη πρότυπες πόρτες. Σαν αποτέλεσμα η συνεργασία σήμαινε άνοιγμα "οπών" στα τείχη προστασίας (firewalls) κάτι που πολλές φορές δεν ήταν αποδεκτό από τους ανθρώπους της πληροφορικής σε μια εταιρία αφού έθετε σε κίνδυνο στην ασφάλεια των συστημάτων. Το γεγονός αυτό δεν επέτρεπε δυναμική συνεργασία λόγω του ότι απαιτούσε μια χειροκίνητη διαδικασία για τη συνεργασία μιας εταιρίας με τους συνεργάτες της. Τα Web Services μπορούν να χρησιμοποιήσουν (μεταξύ άλλων) το HTTP ως πρωτόκολλο μεταφοράς και τα περισσότερα τείχη προστασίας επιτρέπουν την πρόσβαση μέσω της θύρας 80 (πρότυπη θύρα για το HTTP). Με αυτόν τον τρόπο οδηγούμαστε σε ευκολότερες και δυναμικές συνεργασίες μεταξύ των συστημάτων των εταιριών.

- ✓ Διαλειτουργικότητα. Ένα Web Service παρέχει ανεξαρτησία τόσο από το λειτουργικό σύστημα όσο και από το hardware. Οποιοδήποτε πρόγραμμα που συμβαδίζει με αυτή τη τεχνολογία μπορεί πολύ εύκολα να προσπελάσει μία τέτοια υπηρεσία.

- ✓ Ενσωμάτωση. Σε ένα υπάρχον λογισμικό σύστημα που λειτουργεί μέσα στο Internet η δημιουργία ενός Web Service δεν απαιτεί αλλαγές στον μηχανισμό του συστήματος.

- ✓ Διαθεσιμότητα και δημοσίευση. Οι πληροφορίες για τα Web Services δημοσιεύονται οπότε η εύρεση και η χρήση τους μπορεί να είναι ταχύτατες.

- ✓ Επέκταση. Ένα έτοιμο Web Service είναι δυνατό να ανανεωθεί με εύκολο τρόπο παρέχοντας έτσι επιπρόσθετες υπηρεσίες στους χρήστες του.

- ✓ Μικρό κόστος δημιουργίας και χρήσης. Εφόσον σε ένα λογισμικό σύστημα υπάρχει ήδη κάποια διαδικασία που χρειάζεται να επεκταθεί σε on-line υπηρεσία, η δημιουργία του Web Service κοστίζει ελάχιστα. Επίσης το κόστος ενσωμάτωσης ενός Web Service σε κάποιο website ή σε δικτυακή εφαρμογή είναι πάρα πολύ μικρό. Ακόμα και στις περιπτώσεις που η χρήση κάποιου Web Service γίνεται με ενοικίαση σίγουρα το συνολικό κόστος της χρήσης είναι αρκετά πιο μικρό από το κόστος δημιουργίας της υπηρεσίας αυτής.

- ✓ Χρήση λογισμικών συστημάτων. Όλα τα λογισμικά συστήματα και ειδικότερα τα websites που χρησιμοποιούν έτοιμες υπηρεσίες γίνονται πιο λειτουργικά και πιο φιλικά αφού παρέχουν περισσότερες υπηρεσίες στους χρήστες.

1.2 Παραδείγματα των Web Services

Υπάρχει μία μεγάλη λίστα από έτοιμα web services που θα μπορούσε να χρησιμοποιήσει κανείς, ακόμα και εντελώς δωρεάν.

Ψάχνοντας στη διεύθυνση http://www.webservicelist.com μπορούμε να βρούμε μία πληθώρα από web services συνοδευόμενα από το wsdl file καθώς και με on-line demo της λειτουργίας τους.
Στον πίνακα που ακολουθεί αναγράφονται κάποια ενδεικτικά web services.

Web Service	**URL**
Country Population Lookup Service Δέχεται το όνομα μίας χώρας και επιστρέφει τον πληθυσμό της	http://www.cs.uga.edu/~sent
BN Quote Service Δέχεται το ISBN ενός βιβλίου και επιστρέφει την τιμή του	http://www.xmethods.com/help/addspecs.html
Rich Credit Card Validator Ελέγχει τα στοιχεία μίας πιστωτικής κάρτας αν	http://www.richsolutions.com

αντιστοιχούν σε υπάρχουσα πιστωτική.	
SOAP SMS Στέλνει κείμενα των χρηστών ως sms σε κινητά.	http://www.redcoal.com
Google Web Service Απαντά σε ερωτήσεις εύρεσης σελίδων και επιστρέφει τα αποτελέσματα σε μορφή επεξεργάσιμη.	http://www.google.com/apis
FreshScore Δίνει τα αποτελέσματα των αγώνων σε πραγματικό χρόνο.	http://www.freshscore.com/ service/FreshScoreLiveScores.asmx

Πίνακας 1.1: Τα δημοφιλέστερα Web Services και η λειτουργία τους

1.3 Υλοποίηση των Web Services

Οι περισσότερες από τις Ελληνικές επιχειρήσεις που έχουν συνεχή παρουσία στο Internet είτε μέσω ενός δυναμικού website είτε μέσω εξειδικευμένων δικτυακών εφαρμογών μπορούν με σχετικά απλό τρόπο να δημιουργήσουν το δικό τους web service.

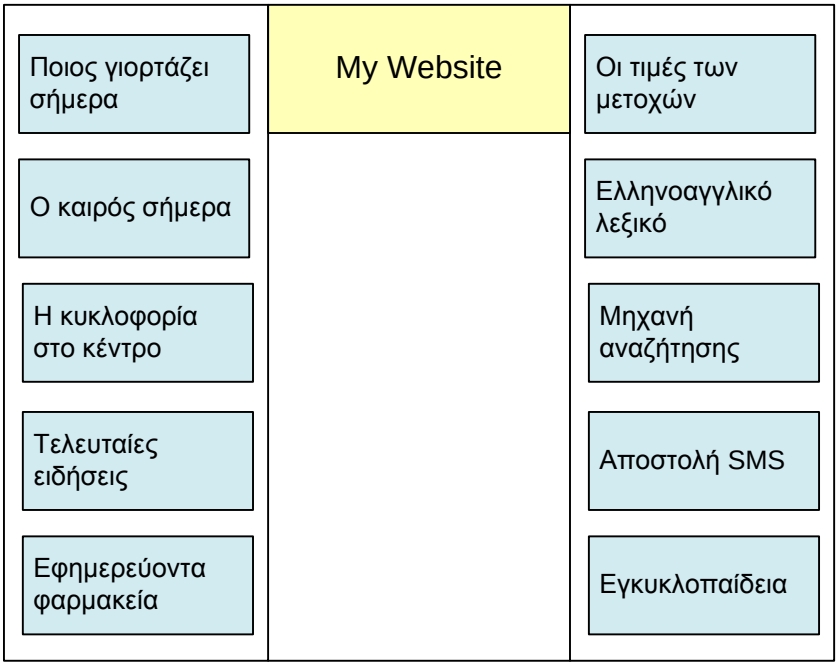
Σίγουρα η δημιουργία ενός web service έχει κάποιο κόστος αφού είναι δουλειά για προγραμματιστές και όχι για απλούς χειριστές. Δημιουργώντας κάποια επιχείρηση ένα web service, αυτό σημαίνει πως η επιχείρηση αυτή στοχεύει είτε στην πώληση των υπηρεσιών της σε άλλες επιχειρήσεις είτε στη δημοσιοποίηση και διαφήμιση των παρεχόμενων υπηρεσιών της. Είναι λοιπόν στα χέρια κάθε επιχειρηματία να αποφασίσει αν πρέπει να επενδύσει χρόνο και χρήμα σε δημιουργία υπηρεσιών που θα παρέχονται μέσω Internet είτε δωρεάν είτε επί πληρωμή.

Από την άλλη μεριά, αν σκεφτεί κανείς το κόστος που μπορεί να έχει για μία εταιρεία η δημιουργία από το μηδέν ενός συστήματος που θα της καλύπτει κάποιες ανάγκες σε σχέση με την επιλογή της χρήσης έτοιμης υπηρεσίας από το Internet, μπορεί εύκολα να δικαιολογήσει γιατί μέρα με τη μέρα η λίστα των διαθέσιμων web services συνεχώς μεγαλώνει.

Τα δύο βήματα για την χρησιμοποίηση ενός Web Service είναι:
1. Αναζήτηση του Web Service που ενδιαφέρει το χρήστη
2. Αναπροσαρμογή της ιστοσελίδας ώστε να εμφανίζεται η σελίδα που επέλεξε ο χρήστης στο σημείο που επιθυμεί.

1.4 Τι αλλάζει αν χρησιμοποιήσουμε Web Services

Η υλοποίηση ενός website το οποίο περιέχει τις παρακάτω υπηρεσίες είναι πολύ απλή και εύκολη με τη χρήση των Web Services.



Εικόνα 1.1: Παροχή υπηρεσιών ενός website με χρήση των web services

Το μόνο που πρέπει να κάνει εδώ ο κατασκευαστής του website είναι να ψάξει για να βρει Internet Web Services δωρεάν ή και επί πληρωμή που προσφέρουν αυτές τις υπηρεσίες. Κατόπιν θα πρέπει να συμπληρώσει για κάθε κουτάκι στην σελίδα του κάποιες γραμμές εντολών οι οποίες επικοινωνούν με τον αντίστοιχο SOAP server (ο οποίος είναι ένας server που υλοποιείται από τη χρήση ενός πρωτοκόλλου SOAP), παίρνουν τις πολύτιμες πληροφορίες και τέλος τις εμφανίζουν με ωραίο, γραφικό τρόπο. Το αποτέλεσμα θα είναι η δημιουργία ενός website με υπηρεσίες οι οποίες φαίνονται στους χρήστες σαν να λειτουργούν στο ίδιο το site.

Αν και τελικά δημιουργείται μία ιστοσελίδα η οποία στην ουσία συντάσσεται τμηματικά από κλήσεις σε ξένες υπηρεσίες, ο τελικός χρήστης θα βλέπει πάντα το καλύτερο δυνατό αποτέλεσμα και ο κατασκευαστής θα έχει ξοδέψει ελάχιστο χρόνο και χρήμα κάνοντας ταυτόχρονα την ιστοσελίδα πιο ελκυστική.

1.5 Τεχνικά Στοιχεία των Web Services

1.5.1 Ορισμός

Το Web Service είναι ένα λογισμικό σύστημα που αναγνωρίζεται από ένα URI και που το περιβάλλον διεπαφής (interface) του καθώς και οι δράσεις του ορίζονται πλήρως και περιγράφονται σε eXtensble Markup Language (XML) μορφή.

Το Web Service μπορεί να βρεθεί και να χρησιμοποιηθεί εύκολα από άλλα λογισμικά συστήματα. Αυτά τα συστήματα μπορούν να αλληλεπιδρούν με το Web Service χρησιμοποιώντας υποχρεωτικά τρόπους επικοινωνίας μέσω του Internet και αλλάζοντας πληροφορίες σε μορφή XML.

Παρόλο που ο ορισμός φαίνεται αρκετά γενικός και δεν περιορίζει τη χρήση των Web

Services με συγκεκριμένα πρωτόκολλα, μετά από μία κοινή αποδοχή των μεγαλύτερων εταιρειών λογισμικού στον κόσμο (Microsoft, IBM, Sun κ.α.) της αρχιτεκτονικής αυτής, έχει πλέον καθοριστεί ένα πιο συγκεκριμένο μοντέλο σύμφωνα με το οποίο θα πρέπει οι εταιρείες να παράγουν και να χρησιμοποιούν τα Web Services.

1.5.2 Μοντέλο

Το πιο συνηθισμένο μοντέλο για τα Web Services προδιαγράφεται ως εξής:

1. Για την επικοινωνία χρησιμοποιείται συνήθως το πρωτόκολλο HTTP, το ίδιο δηλαδή πρωτόκολλο που χρησιμοποιούν και οι κοινοί browsers για την πλοήγηση στο Internet. Πιο απλά τα δεδομένα μεταφέρονται όπως ακριβώς μεταφέρονται και οι ιστοσελίδες.

2. Βασιζόμενο πάνω στο HTTP πρωτόκολλο (όχι αναγκαστικά) χρησιμοποιείται ένα άλλο πρωτόκολλο που ονομάζεται SOAP (Simple Object Access Protocol). Με τη χρήση του πρωτοκόλλου αυτού υλοποιείται ένας εξυπηρετητής (server), ο SOAP server.

3. Για κάθε μέθοδο της υπηρεσίας που θέλουμε να προσφέρουμε, ορίζουμε μία αντιστοιχία με μία λειτουργία του SOAP Server. Κατόπιν, το σύνολο των λειτουργιών του SOAP Server καθώς και τα υπόλοιπα χαρακτηριστικά του, περιγράφονται σε ένα αρχείο που ονομάζεται WSDL (Web Service Description Language) file και το οποίο δημοσιεύεται στο Internet έτσι ώστε να δίνεται η δυνατότητα σε κάθε ενδιαφερόμενο χρήστη να μπορεί άμεσα να χρησιμοποιήσει την υπηρεσία.

Το SOAP (Simple Object Access Protocol) είναι επίσης ένα άλλο standard της W3C και χρησιμοποιείται αρκετά με στόχο την αποστολή απλών αντικειμένων (αρχείων, εφαρμογών, κλπ.) σε XML μορφή.

Γι' αυτό και κάθε Web Service που χρησιμοποιεί SOAP μπορεί να λάβει αιτήσεις για συγκεκριμένες λειτουργίες απλά δεχόμενο αντικείμενα σε XML. Η χρήση του SOAP πρωτοκόλλου γίνεται συνήθως πάνω από το πρωτόκολλο HTTP αλλά μπορεί να λειτουργήσει και με άλλα πρωτόκολλα όπως FTP, SMTP κ.ά.

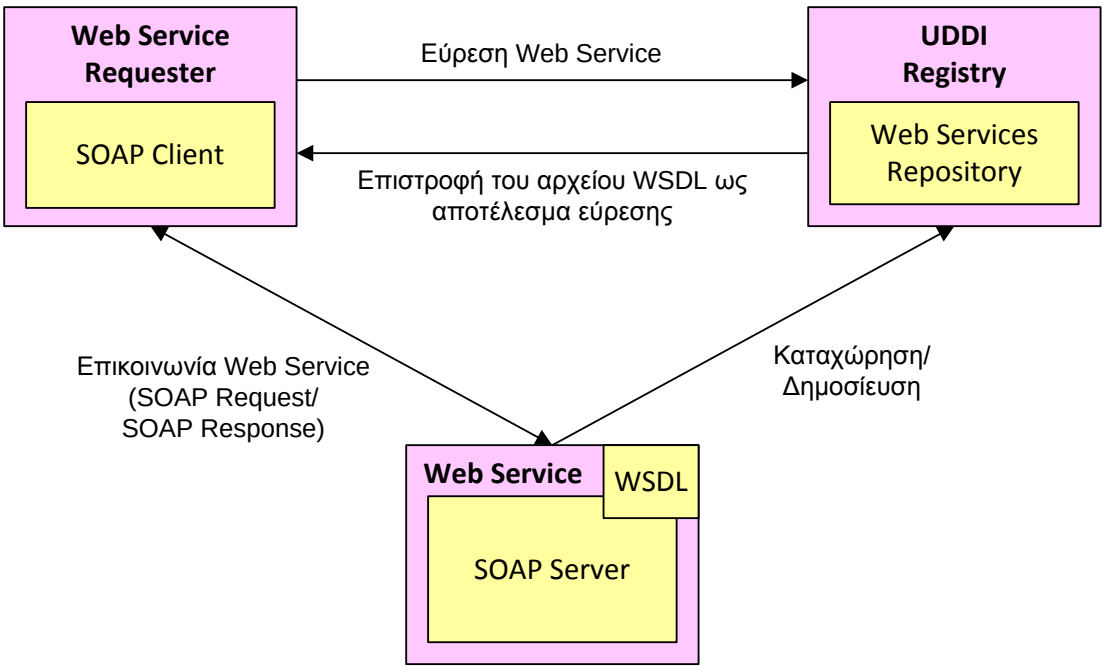
Πλέον, οι περισσότερες γλώσσες προγραμματισμού από την Delphi v7 μέχρι το Visual Studio.net, την PHP, την JSP και άλλες πολλές, υποστηρίζουν τη δημιουργία SOAP Servers με πάρα πολύ απλό τρόπο. Το μόνο που πρέπει να κάνει κανείς είναι να καθορίσει τις λειτουργίες που πρέπει να γίνουν όταν ο server δεχθεί κάποιο αίτημα προς εξυπηρέτηση.

Η WSDL (Web Service Description Language) είναι μία γλώσσα σε XML μορφή η οποία περιγράφει απόλυτα ένα Web Service. Έτσι για κάθε ένα Web Service που δημιουργείται, αντίστοιχα πρέπει να δημιουργείται ένα αρχείο WSDL στο οποίο θα καταγράφονται όλες οι πληροφορίες για το ίδιο το service.

Πιο συγκεκριμένα εκεί καταγράφεται το πού βρίσκεται ο server (σε ποια διεύθυνση),

ποιες λειτουργίες υποστηρίζει καθώς και πως δέχεται και πως επιστρέφει τα δεδομένα για κάθε λειτουργία. Παραδείγματα αρχείων WSDL μπορούν να βρεθούν σε κάθε διεύθυνση που αναφέρεται σε Web Service.

Οι προγραμματιστές και εδώ δεν θα πρέπει να ανησυχούν πολύ αφού υπάρχουν πάρα πολλά διαθέσιμα εργαλεία που δημιουργούν αυτόματα ένα wsdl αρχείο παράλληλα με τη δημιουργία του SOAP server.



Εικόνα 1.2: Το μοντέλο και η χρήση των Web Services
Πηγή: http://www.go-online.gr

Υπάρχουν διαφορετικοί τύποι καταχωρήσεων μίας υπηρεσίας. Πιο συγκεκριμένα υπάρχουν καταχωρήσεις που μπορούν να γίνουν για υπηρεσίες από όλο τον κόσμο και που απευθύνονται σε όλο τον κόσμο, αλλά και καταχωρίσεις που απευθύνονται μόνο σε εξειδικευμένες επιχειρήσεις προωθώντας έτσι και το B2B μοντέλο συνεργασίας. Τέλος υπάρχουν και καταχωρήσεις υπηρεσιών για πιο εξειδικευμένες περιπτώσεις.

1.5.3 Δημιουργία των Web Services

Υπάρχουν αρκετές διαφορετικές πλατφόρμες στις οποίες μπορεί να βασιστεί κανείς για τη δημιουργία ενός web service. Από τη μεριά της Microsoft, οι έτοιμες λύσεις που δίνει το περιβάλλον Visual Studio .net έχουν προσελκύσει πολλούς προγραμματιστές για να δημιουργούν τέτοιες υπηρεσίες.

Επίσης άλλες μεγάλες εταιρίες όπως η IBM και η ORACLE χρησιμοποιούν τα δικά τους προγραμματιστικά εργαλεία. Τέλος, ακόμα και οι περισσότερες γλώσσες προγραμματισμού έχουν ενσωματώσει στις δυνατότητές τους την αυτόματη δημιουργία SOAP servers και την υποστήριξη των Web Services.

Μία δωρεάν λύση προσφέρεται και στους προγραμματιστές δυναμικών ιστοσελίδων που χρησιμοποιούν την γλώσσα PHP. Υπάρχουν έτοιμες βιβλιοθήκες που μπορούν να χρησιμοποιηθούν από οποιονδήποτε για να δημιουργήσει απλά – προσθέτοντας μόνο 5 γραμμές εντολών ένα Web Service.

Ακολουθεί ένα παράδειγμα στην γλώσσα PHP:

Ας υποθέσουμε πως η εκκλησία της Ελλάδος επιθυμεί να υλοποιήσει ένα Web Service το οποίο θα παρέχει υπηρεσίες εορτολογίου. Δηλαδή για κάθε ημερομηνία που θα δέχεται θα επιστρέφει τα ονόματα των Αγίων που γιορτάζουν εκείνη την ημέρα.

Έχοντας δημιουργήσει μία απλή βάση δεδομένων με τις ημερομηνίες και τις αντίστοιχες εορτές το υπόλοιπο κομμάτι της υλοποίησης περιγράφεται παρακάτω:

1. Δημιουργούμε τη διαδικασία ερώτησης στην βάση και επιστροφής αποτελεσμάτων. Αυτό γίνεται με μία συνάρτηση σε PHP.
2. Δημιουργούμε τον SOAP server. Χρησιμοποιώντας την έτοιμη βιβλιοθήκη nusoap, με 5 εντολές δημιουργούμε τον server μας σε ένα αρχείο.
3. Ορίζουμε στον server τη λειτουργία που επιθυμούμε να κάνει
4. Δημιουργούμε το wsdl αρχείο με τις πληροφορίες για τον server μας την λειτουργία τους και τα δεδομένα που δέχεται και επιστρέφει.

2

ΓΛΩΣΣΕΣ ΠΕΡΙΓΡΑΦΗΣ ΔΕΔΟΜΕΝΩΝ XML

2.1 XML

Η γλώσσα XML, που προέρχεται από τα ακρωνύμια των λέξεων Extensive Markup Language, αποτελεί μια περιγραφική γλώσσα δομών δεδομένων που καλούνται XML documents και μερικώς περιγράφει τη συμπεριφορά των προγραμμάτων που επεξεργάζεται. Η XML ουσιαστικά προέρχεται από την SGML και αποτελεί μια προσαρμοσμένη έκδοσή της με πρωτεύων σκοπό την εκτενή χρησιμοποίησή της σε κατανεμημένες εφαρμογές.

Η XML αποτελεί μία γλώσσα σήμανσης, που περιέχει ένα σύνολο κανόνων για την ηλεκτρονική κωδικοποίηση κειμένων. Ορίζεται, κυρίως, στην προδιαγραφή XML 1.0 (XML 1.0 Specification), που δημιούργησε ο διεθνής οργανισμός προτύπων W3C (World Wide Web Consortium), αλλά και σε διάφορες άλλες σχετικές προδιαγραφές ανοιχτών προτύπων. Η XML σχεδιάστηκε δίνοντας έμφαση στην απλότητα, τη γενικότητα και τη χρησιμότητα στο Διαδίκτυο. Είναι μία μορφοποίηση δεδομένων κειμένου, με ισχυρή υποστήριξη Unicode για όλες τις γλώσσες του κόσμου. Αν και η σχεδίαση της XML εστιάζει στα κείμενα, χρησιμοποιείται ευρέως για την αναπαράσταση αυθαίρετων δομών δεδομένων, που προκύπτουν για παράδειγμα στις υπηρεσίες ιστού.

Υπάρχει μία ποικιλία διεπαφών προγραμματισμού εφαρμογών, που μπορούν να χρησιμοποιούν οι προγραμματιστές, για να προσπελαύνουν δεδομένα XML, αλλά και διάφορα συστήματα σχημάτων XML, τα οποία είναι σχεδιασμένα για να βοηθούν στον ορισμό γλωσσών, που προκύπτουν από την XML.

Έως το 2009, έχουν αναπτυχθεί εκατοντάδες γλώσσες που βασίζονται στην XML, συμπεριλαμβανομένων του RSS, του SOAP και της XHTML.

Παρακάτω ερμηνεύονται τα ακρωνύμια των λέξεων Extensible Markup Language.

eXtensible. Η XML είναι επεκτάσιμη. Αυτό επιτρέπει να ορίσουμε ετικέτες, τη σειρά που παρουσιάζονται, και πώς θα πρέπει να εμφανίζονται. Ένας άλλος τρόπος να ορίσουμε την επεκτασιμότητα είναι να θεωρήσουμε ότι η XML μας επιτρέπει να επεκτείνουμε την αντίληψη του τι είναι ένα έγγραφο: μπορεί να είναι ένα αρχείο που ζει σε ένα διακομιστή αρχείων, ή μπορεί να είναι ένα μεταβατικό τμήμα δεδομένων

που ρέει μεταξύ δύο συστημάτων υπολογιστών, όπως στην περίπτωση των Web Services.

Markup. Το πιο αναγνωρίσιμο χαρακτηριστικό της XML είναι οι ετικέτες (tags) ή στοιχεία (elements). Στην πραγματικότητα, τα στοιχεία που δημιουργούνται σε XML είναι πολύ παρόμοια με τα στοιχεία που έχουν ήδη δημιουργηθεί σε έγγραφα HTML. Ωστόσο, η XML επιτρέπει στον ενδιαφερόμενο να ορίσει το δικό του σύνολο ετικετών.

Language. Η XML είναι μια γλώσσα που είναι πολύ παρόμοια με την HTML. Είναι πολύ πιο ευέλικτη επειδή επιτρέπει τη δημιουργία προσαρμοσμένων ετικετών από τον εκάστοτε χρήστη. Ωστόσο, είναι σημαντικό να συνειδητοποιήσει ο ενδιαφερόμενος ότι η XML δεν είναι απλώς μια γλώσσα. Η XML είναι μια μετα-γλώσσα, δηλαδή μια γλώσσα που μας επιτρέπει να δημιουργήσουμε ή να ορίσουμε άλλες γλώσσες. Με την XML μπορούμε να δημιουργήσουμε άλλες γλώσσες, όπως τις RSS, MathML και ακόμη και εργαλεία, όπως η XSLT.

2.1.1 Βασική ορολογία

Χαρακτήρας Unicode
Εξ ορισμού, ένα κείμενο XML είναι μία ακολουθία χαρακτήρων. Σχεδόν κάθε χαρακτήρας Unicode μπορεί να εμφανίζεται σε ένα κείμενο XML.

Επεξεργαστής και Εφαρμογή
Είναι το λογισμικό που επεξεργάζεται ένα κείμενο XML. Ένας επεξεργαστής δουλεύει για μία εφαρμογή. Υπάρχουν μερικές πολύ συγκεκριμένες απαιτήσεις, σχετικά με το τι μπορεί και τι δεν μπορεί να κάνει ένας επεξεργαστής XML, αλλά καμία όσον αφορά στη συμπεριφορά της εφαρμογής. Ο επεξεργαστής (όπως ονοματίζεται από την προδιαγραφή), αναφέρεται συχνά, με τον αγγλικό όρο XML parser.

Σήμανση και Περιεχόμενο
Οι χαρακτήρες που απαρτίζουν ένα κείμενο XML, αποτελούν είτε τη σήμανση είτε το περιεχόμενό του. Η σήμανση και το περιεχόμενο, μπορούν να επισημανθούν και να διακριθούν, ύστερα από την εφαρμογή κάποιων απλών συντακτικών κανόνων. Όλα τα αλφαριθμητικά που συνιστούν τη σήμανση, είτε ξεκινούν με το χαρακτήρα "<" και καταλήγουν στο χαρακτήρα ">", είτε ξεκινούν με το χαρακτήρα "&" και καταλήγουν στο χαρακτήρα ";". Ακολουθίες χαρακτήρων που δε συνιστούν τη σήμανση, αποτελούν το περιεχόμενο ενός κειμένου XML.

Ετικέτα
Είναι ένα στοιχείο σήμανσης που ξεκινά με το χαρακτήρα "<" και καταλήγει στο χαρακτήρα ">". Υπάρχουν τρία είδη ετικέτας: ετικέτες-αρχής, για παράδειγμα `<section>`, ετικέτες-τέλους, για παράδειγμα `</section>`, και ετικέτες-χωρίς-περιεχόμενο, για παράδειγμα `<line-break/>`.

Στοιχείο
Είναι ένα λογικό απόσπασμα ενός κειμένου, που είτε ξεκινά με μία ετικέτα-αρχής και καταλήγει σε μία ετικέτα-τέλους, είτε αποτελείται μόνο από μία ετικέτα-χωρίς-

περιεχόμενο. Οι χαρακτήρες που υπάρχουν, αν υπάρχουν, μεταξύ μιας ετικέτας-αρχής και μιας ετικέτας-τέλους, συνιστούν το περιεχόμενο του στοιχείου, το οποίο μπορεί να περιέχει σήμανση, συμπεριλαμβανομένων και άλλων στοιχείων, που ονομάζονται στοιχεία-παιδιά. Ένα παράδειγμα ενός στοιχείου είναι το `<Greeting>Hello, world.</Greeting>`. Ένα άλλο είναι το `<line-break/>`.

Χαρακτηριστικό
Είναι ένα στοιχείο σήμανσης που αποτελείται από ένα ζευγάρι όνομα/τιμή, το οποίο υπάρχει μέσα σε μία ετικέτα-αρχής ή σε μία ετικέτα-χωρίς-περιεχόμενο.

Δήλωση XML
Τα κείμενα XML μπορούν να αρχίζουν, με τη δήλωση κάποιων πληροφοριών σχετικών με αυτά, όπως στο ακόλουθο παράδειγμα:
<?xml version="1.0" encoding="UTF-8"?>

2.1.2 Η ιστορία της XML

Οι ρίζες της XML μπορούν να αναζητηθούν στην εκρηκτική ανάπτυξη του Παγκόσμιου Ιστού στα μέσα της δεκαετίας του 1990 και στους πολέμους των browser που έλαβαν χώρα μεταξύ της Microsoft Corporation και της Netscape Corporation, όπου καθεμιά από αυτές πάλευε για την απόλυτη κυριαρχία.

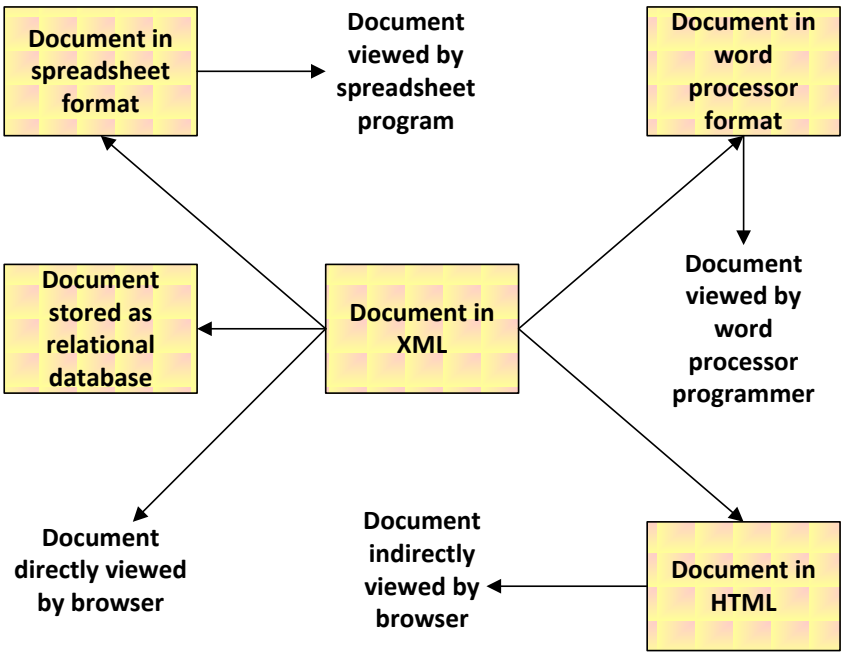
Καθώς το Διαδίκτυο γινόταν όλο και πιο μεγάλο, και όλο και περισσότεροι χρήστες το χρησιμοποιούσαν, άρχισαν να ανακαλύπτονται από τους προγραμματιστές που χρησιμοποιούσαν HTML, διάφορα προβλήματα:

- Ο ίδιος πόρος HTML εμφανιζόταν με διάφορες μορφές, ανάλογα με τον browser που χρησιμοποιούνταν. Αυτό σήμαινε ότι οι σχεδιαστές των ιστοσελίδων έπρεπε το λιγότερο να διπλασιάσουν τις προσπάθειές τους.
- Ορισμένοι κατασκευαστές browser ανέπτυξαν εργαλεία HTML που δεν ήταν αναγνωρίσιμα από άλλους browser.
- Ήταν σχεδόν αδύνατον να διακριθεί οποιαδήποτε σημαντική αλλαγή μέσα σε μια ιστοσελίδα: πουθενά αλλού αυτό δεν ήταν πιο προφανές από ότι στην χρήση των μηχανών αναζήτησης. Ακόμη και στα μέσα της δεκαετίας του 1990 πολλοί χρήστες εξέφραζαν την απογοήτευσή τους όσον αφορούσε αυτά τα προγράμματα εξαιτίας του όγκου των ανακτημένων αρχείων που επέστρεφαν οι μηχανές, τα οποία, στην καλύτερη περίπτωση, σχετίζονταν οριακά με την αναζήτηση. Η αιτία αυτής της "φτωχής" απόδοσης δεν ήταν η τεχνολογία των μηχανών αναζήτησης - στην πραγματικότητα επρόκειτο για εξεζητημένα προγράμματα που αποτελούν αποδεικτικό της εξυπνάδας των προγραμματιστών - αλλά το ότι τα δεδομένα που επεξεργάζονταν οι σελίδες HTML δεν έδιναν πολλά στοιχεία για το περιεχόμενό τους.

Εξαιτίας αυτών των προβλημάτων η Κοινοπραξία Παγκόσμιου Ιστού, η ομάδα που ελέγχει την διαδικασία τυποποίησης του Ιστού, αποφάσισε το 1996 να αναπτύξει μία σημειακή γλώσσα που μελλοντικά θα υποσκέλιζε την HTML. Οι στόχοι αυτής της γλώσσας ήταν:

- Να χρησιμοποιείται εύκολα στο Internet.
- Να μπορεί να υποστηρίζει πολλές εφαρμογές οι οποίες θα κυμαίνονται από browser μέχρι βάσεις δεδομένων μηχανών αναζήτησης.
- Να είναι συμβατή με την SGML, την γλώσσα επεξεργασίας κειμένου που αποτέλεσε την έμπνευση για την HTML.
- Να μην αποτελεί πολύπλοκη διαδικασία η ανάπτυξη επεξεργαστών κειμένων γραμμένων σε γλώσσες που θα βασίζονταν σε XML, για παράδειγμα θα έπρεπε να είναι εύκολη η εγγραφή ενός προγράμματος για τον έλεγχο της σαφήνειας ενός κειμένου πόρου.
- Ο αριθμός των προαιρετικών εργαλείων της γλώσσας να είναι χαμηλός.
- Να είναι εύκολη η ανάγνωση και κατανόηση των αρχείων XML.
- Να είναι εύκολο να αναπτυχθούν με την χρήση απλών συντακτών, αρχεία γραμμένα σε γλώσσα βασιζόμενη σε XML.

Το 1998 η γλώσσα XML παρουσιάστηκε παγκόσμια ως μία τελευταία υπόδειξη της Κοινοπραξίας Παγκόσμιου Ιστού. Σαν αποτέλεσμα, έγινε μία σταθερά για το Internet. Βασιζόταν στην γλώσσα επεξεργασίας κειμένων SGML, η οποία αποτέλεσε την έμπνευση για την HTML. Ο ρόλος της XML συνοψίζεται στο παρακάτω σχήμα, το οποίο παρουσιάζει την σχέση της με άλλες μορφές αποθήκευσης και παρουσίασης.



Εικόνα 2.1: Σχέση της XML με άλλους τρόπους αποθήκευσης και παρουσίασης
Πηγή: http://users.uom.gr

2.1.3 10 χαρακτηριστικές ιδιότητες της XML

Η ακόλουθη συνοπτική παρουσίαση των 10 χαρακτηριστικών ιδιοτήτων της XML συγκεντρώνει βασικές έννοιες και προσφέρει μία ολοκληρωμένη εικόνα για τη γλώσσα.

1. Η XML είναι μία γλώσσα για τη δόμηση δεδομένων

Με την έννοια δομημένα δεδομένα εννοούμε μία συλλογή στοιχείων δεδομένων όπως είναι για παράδειγμα τα λογιστικά φύλλα, οι κατάλογοι διευθύνσεων, οι παράμετροι διαμόρφωσης, οι οικονομικές συναλλαγές και τα τεχνικά σχέδια. Η XML είναι, δηλαδή, ένα σύνολο κανόνων (ή διαφορετικά ένα πακέτο κατευθυντήριων γραμμών ή συμβάσεων) για το σχεδιασμό μορφών κειμένου οι οποίες διευκολύνουν τη δόμηση των δεδομένων σας. Η XML διευκολύνει τον υπολογιστή να παράγει δεδομένα, να διαβάζει δεδομένα και να εξασφαλίζει τη σαφήνεια της δομής των δεδομένων. Η XML αποφεύγει τις συνήθεις παγίδες του σχεδιασμού γλωσσών: είναι επεκτάσιμη, ανεξάρτητη συστήματος υλικού και μπορεί να υποστηρίξει διεθνείς και τοπικές προσαρμογές. Η XML είναι πλήρως συμβατή με Unicode όπως αναφέρθηκε και παραπάνω.

2. Η XML θυμίζει την HTML
Η XML, όπως η HTML, χρησιμοποιεί ετικέτες (tags) (λέξεις μέσα σε γωνιακές αγκύλες '<' και '>') και γνωρίσματα (τύπου όνομα = "τιμή"). Σε αντίθεση με την HTML η οποία διευκρινίζει τη σημασία κάθε ετικέτας και γνωρίσματος και συχνά προσδιορίζει πως θα εμφανίζεται σε φυλλομετρητή το κείμενο το οποίο περιλαμβάνεται σε αυτά, η XML χρησιμοποιεί ετικέτες μόνο για να οριοθετήσει κομμάτια δεδομένων και αφήνει την ερμηνεία των δεδομένων στην εφαρμογή που τα διαβάζει.

3. Η XML είναι κείμενο αλλά δεν προορίζεται για ανάγνωση
Τα προγράμματα που παράγουν λογιστικά φύλλα, καταλόγους διευθύνσεων και άλλα δομημένα δεδομένα αποθηκεύουν, συχνά, τα εν λόγω δεδομένα στο σκληρό δίσκο, χρησιμοποιώντας δυαδική μορφή ή μορφή κειμένου. Ένα από τα πλεονεκτήματα της μορφής κειμένου είναι ότι επιτρέπει στο χρήστη, εάν είναι αναγκαίο, να δει τα δεδομένα χωρίς το πρόγραμμα που τα παρήγαγε. Οι μορφές κειμένου επιτρέπουν, επίσης, στους κατασκευαστές λογισμικού να εκσφαλματώνουν εφαρμογές με μεγαλύτερη ευκολία. Όπως και τα αρχεία HTML, τα αρχεία XML είναι αρχεία κειμένου τα οποία δεν προορίζονται για ανάγνωση αλλά προσφέρουν αυτή τη δυνατότητα στο χρήστη εάν προκύψει ανάγκη. Ωστόσο, οι κανόνες των αρχείων XML είναι αυστηροί σε αντίθεση με τα αρχεία HTML. Η παράληψη μίας ετικέτας ή ένα γνώρισμα δίχως αγκύλες καθιστά άχρηστο το αρχείο XML ενώ η HTML ανέχεται τέτοιου είδους παραλήψεις και συχνά τις επιτρέπει εξολοκλήρου. Η επίσημη προδιαγραφή της XML δεν επιτρέπει σε εφαρμογές να προσπαθούν να μαντέψουν ποιο είναι το πρόγραμμα δημιουργός ενός αρχείου XML με χαμένο σύνδεσμο. Εάν ο σύνδεσμος του αρχείου παρουσιάζει πρόβλημα, η εφαρμογή πρέπει να σταματήσει και να αναφέρει το σφάλμα.

4. Η XML είναι "φλύαρη" γλώσσα
Η XML εμφανίζεται υπό μορφή κειμένου και χρησιμοποιεί ετικέτες για την οριοθέτηση των δεδομένων και για τον λόγο αυτό τα αρχεία XML είναι σχεδόν πάντα μεγαλύτερα σε έκταση από συγκρίσιμα αρχεία σε δυαδική μορφή. Πρόκειται για συνειδητή επιλογή των σχεδιαστών της XML. Τα πλεονεκτήματα ενός αρχείου υπό μορφή κειμένου είναι ολοφάνερα και τα μειονεκτήματα αντισταθμίζονται συνήθως σε άλλο επίπεδο. Η χωρητικότητα του σκληρού δίσκου δεν είναι τόσο ακριβή όσο παλαιότερα και προγράμματα όπως το zip και το gzip μπορούν να συμπιέσουν αρχεία αποτελεσματικά και γρήγορα. Επιπρόσθετα, πρωτόκολλα επικοινωνίας όπως τα

πρωτόκολλα μόντεμ και το HTTP/1.1, το οποίο είναι το πρωτόκολλο πυρήνας του Ιστού, μπορούν να συμπιέσουν πολύ εύκολα αρχεία με μεγάλη ταχύτητα μεταφοράς και το ίδιο αποτελεσματικά όσο και τα δυαδικά αρχεία.

5. Η XML συνδυάζει διαφορετικές τεχνολογίες

Η *XML 1.0* είναι η προδιαγραφή που ορίζει τι είναι οι "ετικέτες" και τα "γνωρίσματα". Πέρα από την XML 1.0, "η οικογένεια XML" είναι ένα διαρκώς αναπτυσσόμενο σύνολο λειτουργικών μονάδων οι οποίες προσφέρουν χρήσιμες υπηρεσίες για τη διεκπεραίωση σημαντικών έργων τα οποία ανακύπτουν συχνά. Η *Xlink* περιγράφει έναν προκαθορισμένο τρόπο εισαγωγής υπερσυνδέσμων σε αρχεία XML. Τα *XPointer* και τα XFragments είναι συντακτικά υπό διαμόρφωση για την υπόδειξη θέσεων ενός εγγράφου XML. Το XPointer μοιάζει λίγο με URL αλλά αντί να υποδεικνύει έγγραφα στον Ιστό, υποδεικνύει κομμάτια πληροφοριών ενός εγγράφου XML. Το *CSS*, η γλώσσα μορφοποίησης σελίδων, είναι δυνατό να εφαρμοστεί σε XML όπως και σε HTML. Το *XSL* είναι προηγμένη γλώσσα (advanced language) μορφοποίησης σελίδων. Βασίζεται στο *XSLT*, μία γλώσσα μετασχηματισμού η οποία χρησιμοποιείται για την αναδιάταξη, την πρόσθεση και την διαγραφή ετικετών και γνωρισμάτων. Το *DOM* είναι ένα προκαθορισμένο σύνολο λειτουργιών για τη διαχείριση αρχείων XML (και HTML) από μία γλώσσα προγραμματισμού. Τα *XML Schemas 1* και *2* επιτρέπουν στους κατασκευαστές λογισμικού να ορίσουν με ακρίβεια τις δομές των δικών τους μορφών XML. Υπάρχουν αρκετά εργαλεία και λειτουργικές μονάδες τα οποία βρίσκονται υπό διαμόρφωση ή είναι ήδη διαθέσιμα.

6. Η XML είναι καινούρια όχι, όμως, εντελώς καινούρια

Η ανάπτυξη της XML ξεκίνησε το 1996. Από το Φεβρουάριο του 1998 η XML αποτελεί Σύσταση του W3C. Ίσως, λοιπόν να θεωρήσετε ότι η XML δεν έχει ωριμάσει ακόμα τεχνολογικά. Στην πραγματικότητα, όμως, η τεχνολογία XML δεν είναι τόσο καινούρια. Πριν από την XML υπήρχε η SGML, η οποία αναπτύχθηκε στις αρχές της δεκαετίας του '80, τυποποιήθηκε από τον ISO το 1986, και χρησιμοποιήθηκε ευρέως σε προγράμματα με εκτεταμένη τεκμηρίωση. Η ανάπτυξη της HTML ξεκίνησε το 1990. Οι σχεδιαστές της XML επέλεξαν τα καλύτερα τμήματα της SGML, χρησιμοποίησαν την εμπειρία που είχαν αποκτήσει κατά την ανάπτυξη της HTML και παρήγαγαν μία γλώσσα η οποία δεν είναι λιγότερο ισχυρή από την SGML αλλά είναι πιο κανονικοποιημένη και πολύ πιο εύχρηστη. Είναι λοιπόν δύσκολο να διακρίνει κανείς την εξελικτική από την επαναστατική πρόοδο. Αξίζει να σημειωθεί, τέλος, ότι ενώ η SGML χρησιμοποιείται κυρίως για τεχνική τεκμηρίωση, και πολύ λιγότερο για δεδομένα άλλου είδους, για την XML ισχύει ακριβώς το αντίθετο.

7. Η XML οδηγεί την HTML σε XHTML

Μία από τις εφαρμογές XML υπάρχει υπό μορφή εγγράφου: πρόκειται για την XHTML του W3C, τη διάδοχο της HTML. Η XHTML διαθέτει αρκετά κοινά στοιχεία με την HTML. Το συντακτικό, όμως, έχει αλλάξει έτσι ώστε να συμβαδίζει με τους κανόνες της XML. Τα έγγραφα με βάση την XML χρησιμοποιούν το συντακτικό της XML ,με ορισμένους, όμως, περιορισμούς και πρόσθεση σημασίας στο συντακτικό.

8. Η XML επιδέχεται συνδυασμό διαφορετικών μορφών

Η XML επιτρέπει στο χρήστη τον ορισμό νέας μορφής εγγράφου προσφέροντάς του τη δυνατότητα να συνδυάσει και να χρησιμοποιήσει άλλες μορφές. Ωστόσο, επειδή δύο διαφορετικές μορφές, οι οποίες έχουν αναπτυχθεί ανεξάρτητα, ενδέχεται να διαθέτουν στοιχεία ή γνωρίσματα με το ίδιο όνομα, πρέπει να αποδοθεί ιδιαίτερη προσοχή κατά το συνδυασμό των δύο μορφών. Για την αποφυγή σύγχυσης ονομάτων κατά το συνδυασμό μορφών, η XML παρέχει ένα μηχανισμό namespace. Παραδείγματα μορφών με βάση την XML οι οποίες χρησιμοποιούν namespaces είναι η XSL και η RDF .

9. Η XML αποτελεί τη βάση του RDF και του Σημασιολογικού Ιστού
Ο Σκελετός Περιγραφής Πόρων του W3C (Resource Description Framework) (RDF) είναι μία μορφή κειμένου XML η οποία υποστηρίζει περιγραφή πόρων και εφαρμογές μεταδεδομένων, όπως οι κατάλογοι μουσικής, οι συλλογές φωτογραφιών και οι βιβλιογραφίες. Για παράδειγμα, το RDF έχει τη δυνατότητα αναγνώρισης προσώπων σε ένα άλμπουμ φωτογραφιών του Ιστού χρησιμοποιώντας πληροφορίες από μία προσωπική λίστα επαφών. Στη συνέχεια, ο πελάτης ηλεκτρονικού ταχυδρομείου (mail client), μπορεί να αποστείλει μηνύματα σε όσους εμφανίζονται στις φωτογραφίες ειδοποιώντας τους ότι οι φωτογραφίες τους έχουν δημοσιευθεί στον Ιστό. Και, βέβαια, όπως οι άνθρωποι έχουν συμφωνήσει να χρησιμοποιούν κοινές ονομασίες για τις σημασίες των λέξεων που χρησιμοποιούν όταν επικοινωνούν, έτσι και οι υπολογιστές χρειάζονται μηχανισμούς οι οποίοι να ορίζουν κοινά ονόματα για τους όρους ώστε να είναι εφικτή η αποτελεσματική επικοινωνία. Οι επίσημες περιγραφές όρων που ανήκουν σε ένα συγκεκριμένο νοηματικό πεδίο (για παράδειγμα αυτό των αγορών ή των κατασκευών) ονομάζονται οντολογίες και συνιστούν σημαντικό τμήμα του Σημασιολογικού Ιστού.

10. Η XML δεν χρειάζεται άδεια χρήσης, λειτουργεί ανεξαρτήτως συστήματος υλικού και τυγχάνει ευρείας υποστήριξης
Η επιλογή της XML προσφέρει πρόσβαση σε μια μεγάλη και διαρκώς αναπτυσσόμενη κοινότητα εργαλείων και ειδικών με μεγάλη εμπειρία στις εν λόγω τεχνολογίες. Αν ο χρήστης διαλέξει την XML είναι σαν να διαλέγει SQL για βάσεις δεδομένων: πρέπει να δημιουργήσει τη δική του βάση δεδομένων και τα δικά του προγράμματα και διαδικασίες για τη διαχείρισή της. Και επειδή η XML δεν χρειάζεται άδεια χρήσης μπορεί να κατασκευάσει πάνω της το δικό του λογισμικό δίχως να πρέπει να πληρώσει. Επίσης, τυγχάνει ευρείας και ολοένα επεκτεινόμενης υποστήριξης που σημαίνει ότι δεν δεσμεύεται ο χρήστης σε ένα μόνο κατασκευαστή. Η XML δεν είναι πάντα η καλύτερη λύση αλλά αξίζει να ληφθεί υπόψη.

2.2 KML

Το KML (Keyhole Markup Language) είναι μία μορφή αρχείου για την απεικόνιση γεωγραφικών πληροφοριών σε έναν Earth browser, όπως τα Google Earth, Google Maps και Google Maps για κινητά. Η KML χρησιμοποιεί μία δομή βασισμένη σε ετικέτες (tags) με ένθετα στοιχεία (elements) και χαρακτηριστικά (attributes) και είναι βασισμένη στο πρότυπο XML. Όλες οι ετικέτες είναι ευαίσθητες στη διάκριση πεζών-κεφαλαίων και πρέπει να εμφανίζονται ακριβώς όπως αναφέρονται στον ορισμό KML. Ο ορισμός δηλώνει ποιες ετικέτες είναι προαιρετικές. Μέσα σε ένα

συγκεκριμένο στοιχείο, οι ετικέτες πρέπει να εμφανίζονται με τη σειρά που εμφανίζονται στον ορισμό.

2.2.1 Βασικά αρχεία KML

Το απλούστερο είδος KML αρχείων είναι αυτά που μπορούν να συνταχθούν άμεσα στο Google Earth, δηλαδή ο ενδιαφερόμενος δεν χρειάζεται να επεξεργαστεί ή να δημιουργήσει κάθε KML σε ένα πρόγραμμα επεξεργασίας κειμένου. Σύνταξη σημάνσεων μερών (Placemarks), επικαλύψεις εδάφους (ground overlays), διαδρομές (paths) και πολύγωνα (polygons) μπορούν όλα να συνταχθούν άμεσα στο Google Earth. Ένα παράδειγμα ενός KML αρχείου είναι το ακόλουθο:

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2"
xmlns:gx="http://www.google.com/kml/ext/2.2"
xmlns:kml="http://www.opengis.net/kml/2.2"
xmlns:atom="http://www.w3.org/2005/Atom">
<Document>
    <name>test_simeio.kml</name>
    <Style id="sn_ylw-pushpin">
        <IconStyle>
            <scale>1.1</scale>
            <Icon>
                        <href>http://maps.google.com/mapfiles/kml/
pushpin/ylw-pushpin.png</href>
            </Icon>
            <hotSpot x="20" y="2" xunits="pixels" yunits="pixels"/>
        </IconStyle>
    </Style>
    <StyleMap id="msn_ylw-pushpin">
        <Pair>
            <key>normal</key>
            <styleUrl>#sn_ylw-pushpin</styleUrl>
        </Pair>
        <Pair>
            <key>highlight</key>
            <styleUrl>#sh_ylw-pushpin</styleUrl>
        </Pair>
    </StyleMap>
    <Style id="sh_ylw-pushpin">
        <IconStyle>
            <scale>1.3</scale>
            <Icon>
                <href>http://maps.google.com/mapfiles/kml/
pushpin/ylw-pushpin.png</href>
            </Icon>
            <hotSpot x="20" y="2" xunits="pixels" yunits="pixels"/>
        </IconStyle>
    </Style>
      <Placemark>
```

```
        <name>Point 1</name>
        <LookAt>
            <longitude>-122.444743</longitude>
            <latitude>37.7526445</latitude>
            <altitude>0</altitude>
            <heading>0.2554718671115674</heading>
            <tilt>0</tilt>
            <range>193836.0400797448</range>
            <altitudeMode>relativeToGround</altitudeMode>
                <gx:altitudeMode>relativeToSeaFloor</gx:altitudeMode>
        </LookAt>
        <styleUrl>#msn_ylw-pushpin</styleUrl>
        <Point>
            <altitudeMode>clampToGround</altitudeMode>
            <gx:altitudeMode>clampToSeaFloor</gx:altitudeMode>
            <coordinates>-
122.50678013203,37.769149792429,0</coordinates>
        </Point>
    </Placemark>
    <Placemark>
        <name>OUT</name>
        <styleUrl>#msn_ylw-pushpin</styleUrl>
        <Polygon>
            <tessellate>1</tessellate>
            <outerBoundaryIs>
                <LinearRing>
                    <coordinates>
                        -122.50854,37.777099,0 -
122.380946,37.777099,0 -122.380946,37.72819,0 -122.50854,37.72819,0
                    </coordinates>
                </LinearRing>
            </outerBoundaryIs>
        </Polygon>
    </Placemark>
  </Document>
</kml>
```

2.2.2 Συστατικά μέρη ενός KML αρχείου

Κάθε KML αρχείο ξεκινάει με την εξής δομή:

- Μία κεφαλίδα XML. Αυτή βρίσκεται στη γραμμή 1 σε κάθε KML αρχείο. Κανένα κενό ούτε άλλοι χαρακτήρες μπορούν να εμφανιστούν πριν από αυτή τη γραμμή.
- Μια δήλωση χώρου ονομάτων KML. Αυτό πραγματοποιείται στη γραμμή 2 σε κάθε KML αρχείο.

Σε ένα KML αρχείο μπορεί να εμφανίζονται τα εξής στοιχεία:

kml	Document	name				
		style id	IconStyle	scale		
				Icon	href	
				hotspot		
		StyleMap	Pair	key		
				StyleUrl		
		Placemark	name			
			LookAt	longitude		
				latitude		
				altitude		
				heading		
				tilt		
				range		
				altitudeMode		
				gx:altitudeMode		
			styleUrl			
			Point	altitudeMode		
				gx:altitudeMode		
				coordinates		
		Placemark	name			
			styleUrl			
			Polygon	tesselate		
				outerBoundaryIs	LinearRing	coordinates

Πίνακας 2.1: Συστατικά μέρη ενός KML αρχείου

Αναλυτικότερα τα βασικά στοιχεία που αποτελούν ένα KML αρχείο είναι τα εξής:

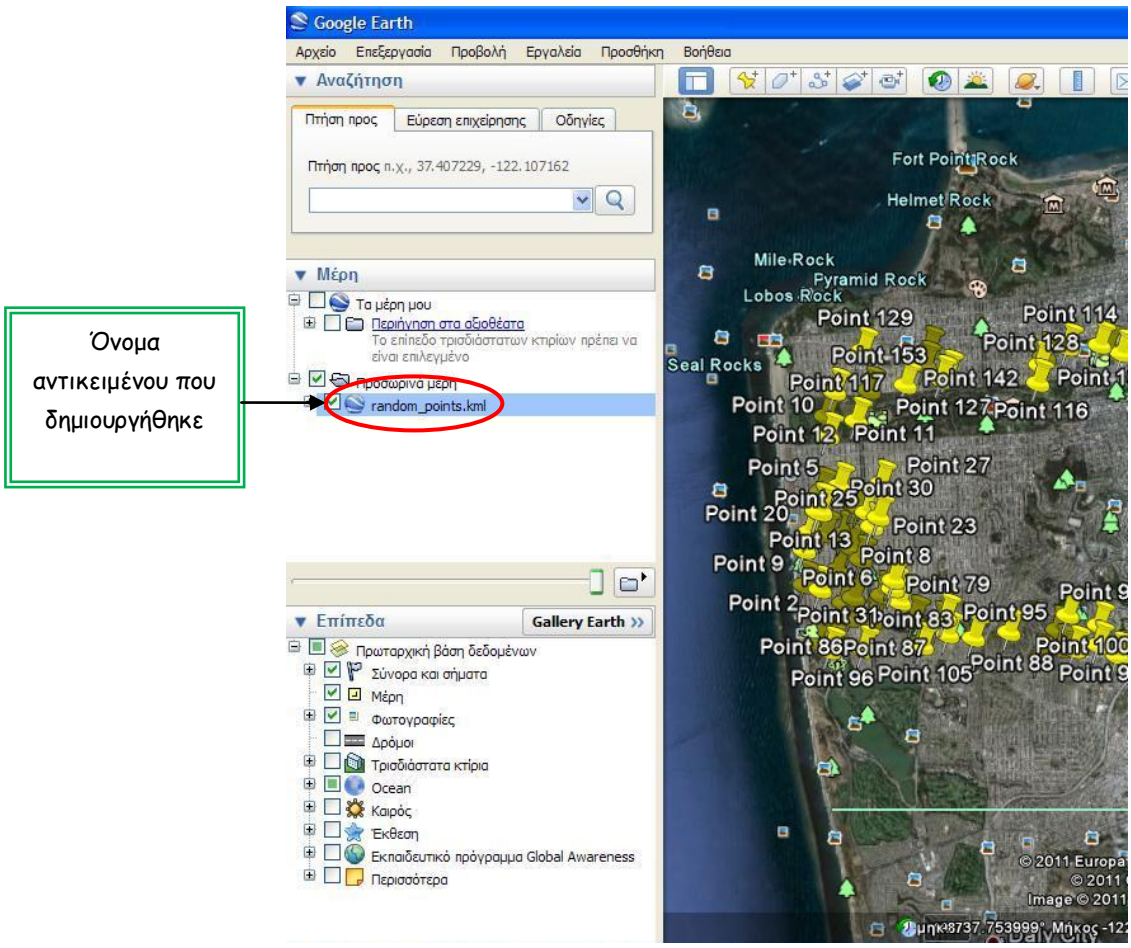
✓ **Έγγραφο – <Document>**

Ένα στοιχείο <Document> περιέχει ένα σύνολο από χαρακτηριστικά και στυλ. Το στοιχείο αυτό απαιτείται εάν το KML αρχείο χρησιμοποιεί κοινόχρηστα στυλ (s*hared styles)*. Ενδείκνυται η χρήση κοινόχρηστων στυλ, τα οποία απαιτούν τα παρακάτω βήματα:

- Προσδιορισμό όλων των στυλ μέσα σε ένα <Document>
- Απόδοση μοναδικής ταυτότητας ID για κάθε στυλ
- Εντός ενός δοσμένου στοιχείου Feature ή StyleMap, αναφορά στην ταυτότητα ID του στυλ χρησιμοποιώντας ένα <styleUrl> στοιχείο.

✓ **Όνομα - <name>**

Το πεδίο αυτό καθορίζεται από τον χρήστη και απεικονίζεται στην τρισδιάστατη απεικόνιση ως ετικέτα για το αντικείμενο (για παράδειγμα για ένα σημείο σήμανσης μέρους, ένα φάκελο ή ένα διαδικτυακό σύνδεσμο).



Εικόνα 2.2: Απεικόνιση αντικειμένου στο Google Earth και εμφάνιση ονόματος αντικειμένου

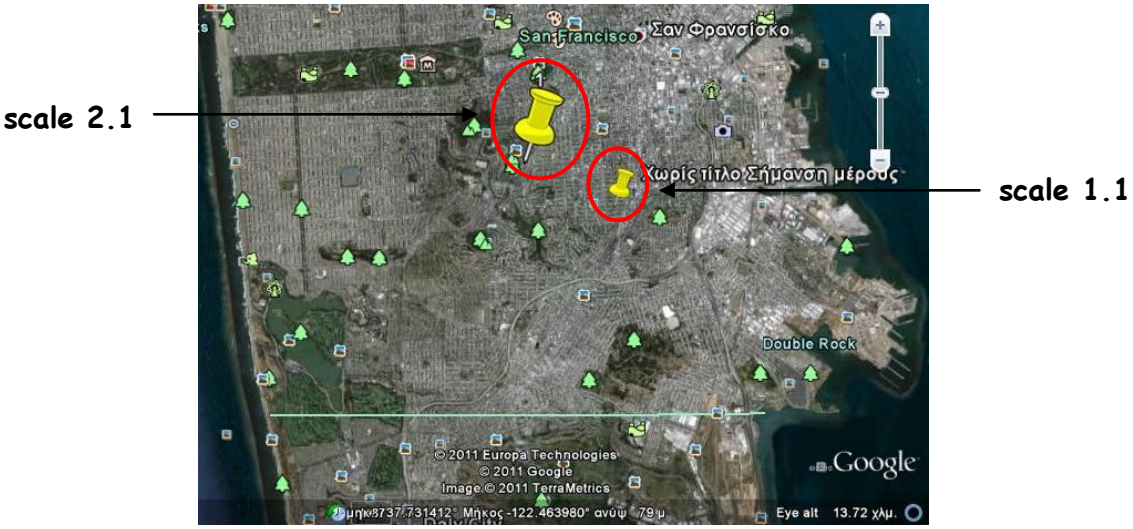
✓ **<u>Στυλ εικονιδίων - <IconStyle></u>**

Καθορίζει πως απεικονίζονται τα εικονίδια για τα Placemarks. Το στοιχείο <Icon> καθορίζει την εμφάνιση του εικονιδίου. Το στοιχείο <scale> καθορίζει την κατά x και y κλίμακα του εικονιδίου. Το χρώμα που έχει ορίζεται στο πεδίο <color> του <IconStyle> αναμιγνύεται με το χρώμα του πεδίου <Icon>.

Στοιχεία σχετικά με το στυλ των εικονιδίων - <IconStyle>

<scale>

Επαναπροσδιορίζει το μέγεθος του εικονιδίου.



Εικόνα 2.3: Απεικόνιση εικονιδίων στο Google Earth με διαφορετικά μεγέθη

<heading>

Κατεύθυνση (Βοράς, Νότος, Ανατολή, Δύση), σε μοίρες. Από προεπιλογή είναι μηδέν (Βοράς). Η διακύμανση των τιμών είναι από 0 έως 360 μοίρες.

<Icon>

Στο <IconStyle>, το μόνο θυγατρικό στοιχείο του <Icon> είναι το <href>:

<href>

Εδώ βρίσκεται μία HTTP διεύθυνση ή ο εντοπισμός ενός τοπικού αρχείου που χρησιμοποιείται για να φορτωθεί ένα εικονίδιο (icon).

<hotSpot x="0.5" **y="0.**5" **xunits="**fraction" **yunits="**fraction"**>**

Η γραμμή αυτή του κώδικα προσδιορίζει τη θέση στο εικονίδιο που είναι «συνδεδεμένη» με το στοιχείο <Point> που προσδιορίζεται στο <Placemark>. Οι τιμές ***x*** και ***y*** μπορούν να προσδιοριστούν με τρεις διαφορετικούς τρόπους: ως pixels, ως fractions (κλάσματα) του εικονιδίου, ή ως inset pixels, το οποίο είναι μία μετατόπιση σε pixels από την ανώτερη δεξιά γωνία του εικονιδίου. Οι ***x και y*** θέσεις μπορούν να προσδιοριστούν με διάφορους τρόπους – για παράδειγμα, το ***x*** μπορεί να είναι σε pixels και το ***y*** να είναι κλάσμα. Η αρχή του συστήματος συντεταγμένων βρίσκεται στην χαμηλότερη αριστερή γωνία του εικονιδίου.

- **x** – είτε ο αριθμός των pixels, ένα κλασματικό συστατικό της εικόνας, ή ένα ένθετο pixel (pixel inset) που δείχνει το x στοιχείο ενός σημείου στο εικονίδιο
- **y** - είτε ο αριθμός των pixels, ένα κλασματικό συστατικό της εικόνας, ή ένα ένθετο pixel (pixel inset) που δείχνει το y στοιχείο ενός σημείου στο εικονίδιο
- **xunits** – μονάδες στις οποίες προσδιορίζεται η τιμή x. Μία τιμή ενός κλάσματος δείχνει ότι η τιμή x είναι κλάσμα του εικονιδίου. Μια τιμή των pixels δείχνει την

τιμή x σε pixels. Μια τιμή insetPixels δείχνει την γραμμή από το δεξί άκρο του εικονιδίου.

- **yunits** - μονάδες στις οποίες προσδιορίζεται η τιμή y. Μία τιμή ενός κλάσματος δείχνει ότι η τιμή y είναι κλάσμα του εικονιδίου. Μια τιμή των pixels δείχνει την τιμή y σε pixels. Μια τιμή insetPixels δείχνει την γραμμή από το πάνω άκρο του εικονιδίου.

<StyleMap>

Το στοιχείο <StyleMap> σχεδιάζει μεταξύ δύο διαφορετικών στυλ. Τυπικά ένα <StyleMap> στοιχείο χρησιμοποιείται για να δώσει ξεχωριστά φυσιολογικά και τονισμένα στυλ για μία σήμανση Placemark, έτσι ώστε η τονισμένη εκδοχή να εμφανίζεται όταν ο χρήστης τοποθετήσει τον κέρσορα του ποντικού επάνω στο εικονίδιο στο Google Earth.

<Pair>

Καθορίζει ένα ζευγάρι κλειδιού/τιμής το οποίο χαρτογραφεί μία λειτουργία (κανονική ή τονισμένη) στο προκαθορισμένο <styleUrl>. Το στοιχείο <Pair> περιέχει δύο στοιχεία, που είναι απαραίτητα και τα δύο:

- **<key>.** Αναγνωρίζει το κλειδί.
- **<styleUrl>** ή **<Style>**. Σχετίζεται με το στυλ. Στο <styleUrl>, για σχετικά στοιχεία στυλ που βρίσκονται τοπικά στο έγγραφο KML, χρησιμοποιείται ένα απλό σύμβολο αναφοράς #. Για στυλ που περιέχονται σε εξωτερικά αρχεία, χρησιμοποιείται ένα πλήρες URL μαζί με το σύμβολο #.

✓ **Σήμανση μέρους - Placemarks**

Τα Placemarks είναι μία από τις πιο συχνά χρησιμοποιούμενες δυνατότητες του Google Earth. Σηματοδοτούν μια θέση στην επιφάνεια της γης, χρησιμοποιώντας μία κίτρινη πινέζα ως εικονίδιο. Η απλούστερη σήμανση μέρους περιλαμβάνει μόνο ένα στοιχείο <point> , το οποίο καθορίζει τη θέση της σημάνσεως. Το Google Earth δίνει τη δυνατότητα στον χρήστη να καθορίσει ένα όνομα και ένα προσαρμοσμένο εικονίδιο για την σήμανση μέρους, και επίσης να προσθέσει και άλλα στοιχεία γεωμετρίας σε αυτό. Ανοίγοντας ένα αρχείο KML στο Google Earth, διακρίνουμε διαφορετικούς τύπους σημάνσεως μέρους, οι οποίοι φαίνονται στο παρακάτω τμήμα κώδικα:

```
<Placemark>
    <name>Point 1</name>
    <LookAt>
        <longitude>-122.444743</longitude>
        <latitude>37.7526445</latitude>
        <altitude>0</altitude>
        <heading>0.2554718671115674</heading>
        <tilt>0</tilt>
```

```
        <range>193836.0400797448</range>
        <altitudeMode>relativeToGround</altitudeMode>
        <gx:altitudeMode>relativeToSeaFloor</gx:altitudeMode>
    </LookAt>
    <styleUrl>#msn_ylw-pushpin</styleUrl>
    <Point>
        <altitudeMode>clampToGround</altitudeMode>
        <gx:altitudeMode>clampToSeaFloor</gx:altitudeMode>
        <coordinates>-122.49054825514,37.750183714057,0
</coordinates>
    </Point>
</Placemark>>
</kml>
```

Η δομή του αρχείου αυτού αναλύεται ως εξής:

- Ένα όνομα (<name>) που χρησιμοποιείται ως ετικέτα για την σήμανση μέρους.
- Μια περιγραφή (<description>) που εμφανίζεται στο παράθυρο "Βοήθειας" που επισυνάπτεται στο σημείο σήμανσης.
- Ένα σημείο (<Point>) που καθορίζει τη θέση του σημείου σημάνσεως στην επιφάνεια της γης. (γεωγραφικό μήκος- ***longitude***, γεωγραφικό πλάτος- ***latitude*** και προαιρετικά υψόμετρο- ***altitude***).

Ένα σημείο σήμανσης μέρους είναι ο μόνος τρόπος για να σχεδιαστεί ένα εικονίδιο και ετικέτα σε 3D Viewer του Google Earth. Από προεπιλογή, το εικονίδιο είναι η κίτρινη πινέζα. Στην KML, ένα <placemark> μπορεί να περιέχει ένα ή περισσότερα στοιχεία γεωμετρίας, όπως ένα LineString, ένα Πολύγωνο (Polygon), ή ένα μοντέλο (model). Αλλά μόνο ένα <Placemark> με ένα σημείο (Point) μπορεί να έχει εικονίδιο και ετικέτα. Το <Point> χρησιμοποιείται για να τοποθετηθεί το εικονίδιο, αλλά δεν υπάρχει καμία γραφική αναπαράσταση του σημείου του ίδιου.

<LookAt>

Καθορίζει μία εικονική κάμερα που σχετίζεται με οποιοδήποτε στοιχείο που προέρχεται από το πεδίο <Feature>. Το στοιχείο LookAt τοποθετεί την κάμερα σε σχέση με το αντικείμενο που παρατηρείται. Στο Google Earth η εικόνα "πετάει" σε αυτό το σημείο this LookAt όταν ο χρήστης πατήσει διπλό κλικ σε ένα αντικείμενο στο Places panel (παράθυρο μερών) ή σε ένα εικονίδιο στο 3D viewer (3D προβολή).

Στοιχεία σχετικά με το <LookAt>

<longitude>

Γεωγραφικό μήκος του σημείου από το οποίο κοιτάζει η κάμερα. Γωνιακή απόσταση σε μοίρες, ως προς τον Πρώτο Μεσημβρινό. Οι τιμές δυτικά του κεντρικού Μεσημβρινού κυμαίνονται από -180 έως 0 μοίρες. Οι τιμές ανατολικά του κεντρικού Μεσημβρινού κυμαίνονται από 0 έως 180 μοίρες.

<latitude>

Γεωγραφικό πλάτος του σημείου από το οποίο κοιτάζει η κάμερα. Σε μοίρες βόρεια και νότια του Ισημερινού (0 μοίρες). Οι τιμές κυμαίνονται από -90 έως 90 μοίρες.

<altitude>

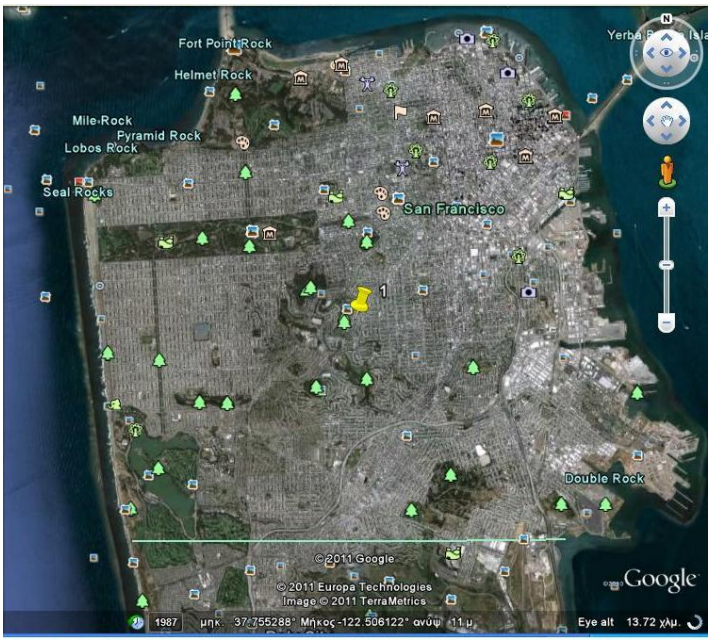
Απόσταση από την επιφάνεια της γης σε μέτρα. Distance from the earth's surface, in meters. Ερμηνεύεται σύμφωνα με το <altitudeMode> που περιγράφεται παρακάτω.

<heading>

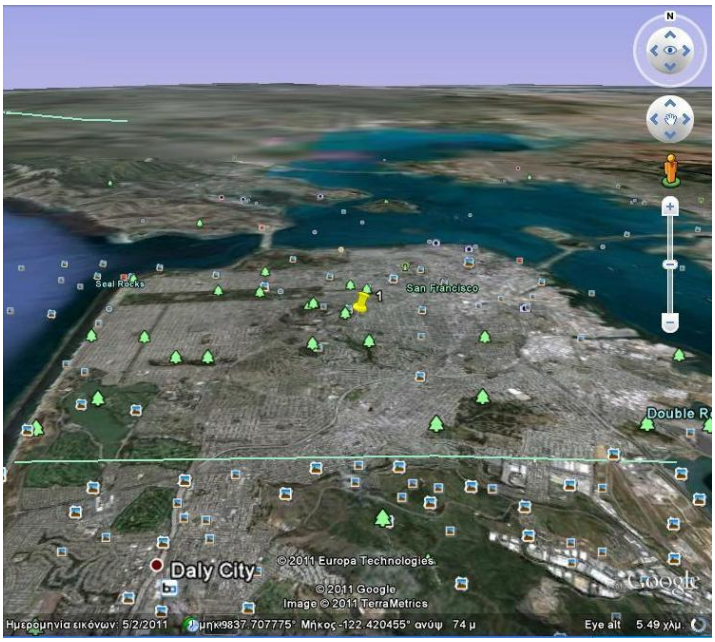
Κατεύθυνση (Βόρεια, Νότια, Ανατολικά, Δυτικά), σε μοίρες. Από προεπιλογή είναι 0 (Βόρεια). Οι τιμές κυμαίνονται από 0 έως 360 μοίρες.

<tilt>

Η γωνία μεταξύ της κατεύθυνσης της LookAt θέσης και της κάθετης στην επιφάνεια της γης. Οι τιμές κυμαίνονται από 0 έως 90 μοίρες. Οι τιμές για το στοιχείο <tilt> δε μπορούν να είναι αρνητικές. Μία τιμή <tilt> των 0 μοιρών δείχνει την προβολή ακριβώς από πάνω. Μία τιμή <tilt> των 90 μοιρών δείχνει την προβολή κατά μήκος του ορίζοντα.



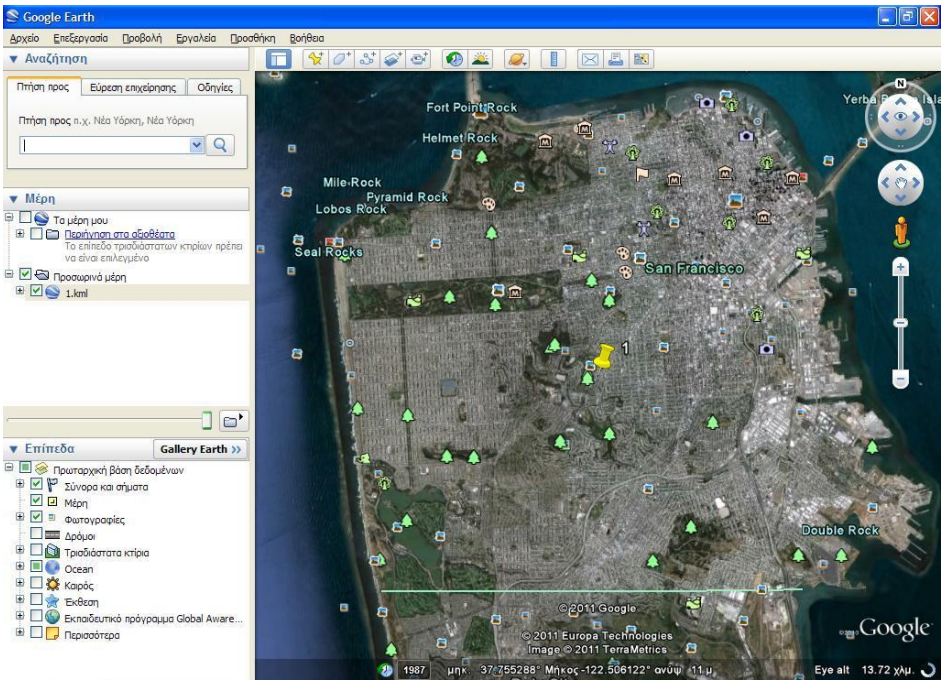
Εικόνα 2.4: Απεικόνιση San Fransisco στο Google Earth με tilt=0



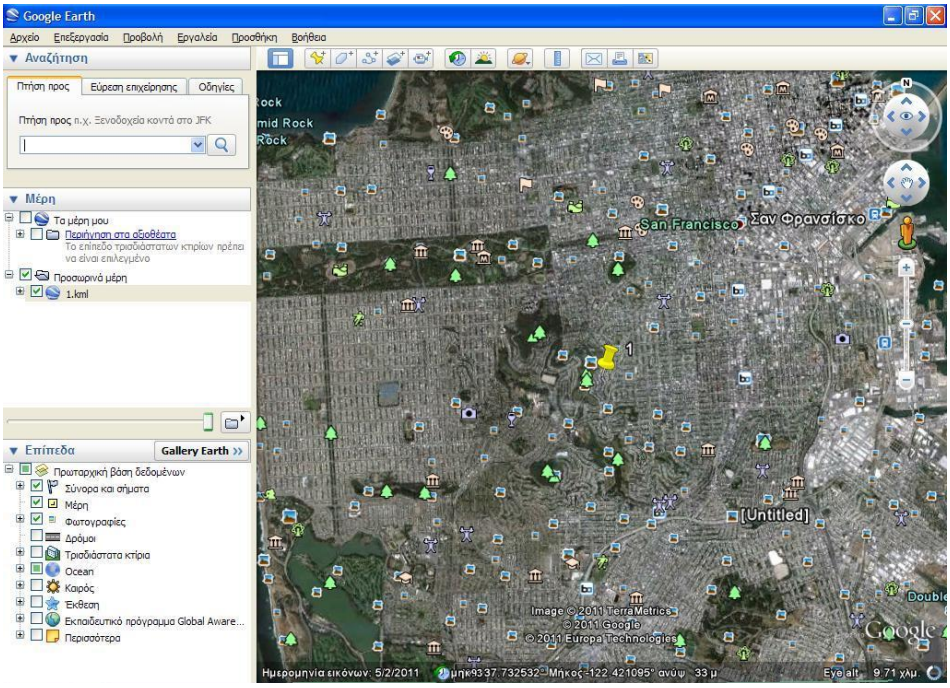
Εικόνα 2.5: Απεικόνιση San Fransisco στο Google Earth με tilt=67

<range>

Απόσταση σε μέτρα από το σημείο που προσδιορίζεται από το γεωγραφικό μήκος, το γεωγραφικό πλάτος και το υψόμετρο στην LookAt θέση.



Εικόνα 2.6: Απεικόνιση San Fransisco στο Google Earth με range=9530.85758459179



Εικόνα 2.7: Απεικόνιση San Fransisco στο Google Earth με range=13530.85758459179

<altitudeMode>

Καθορίζει πως ερμηνεύεται το στοιχείο <altitude> που προσδιορίζεται για ο σημείο LookAt. Πιθανές τιμές είναι οι ακόλουθες:

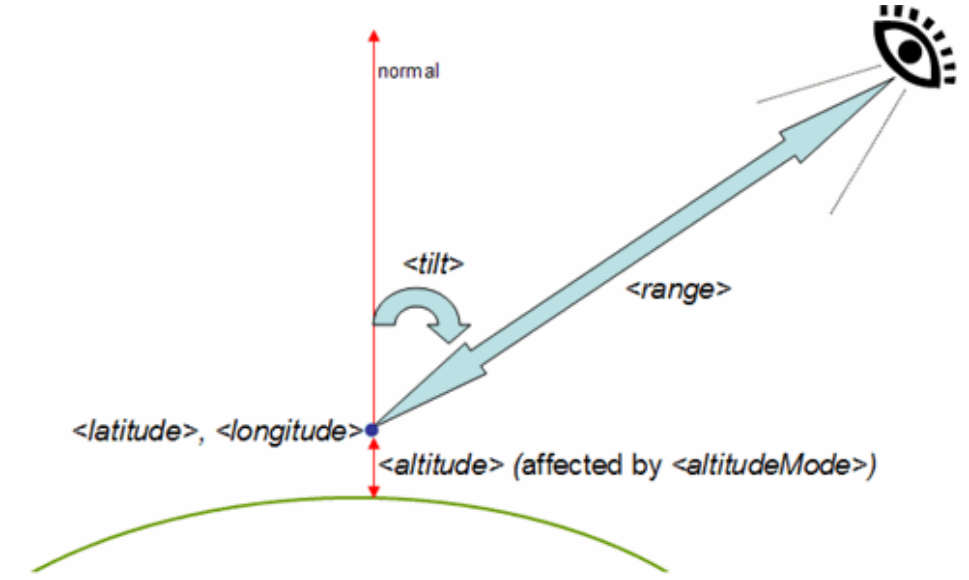
- **clampToGround** - (προεπιλογή) αγνοεί τον προσδιορισμό υψομέτρου και τοποθετεί τη θέση LookAt στο έδαφος.
- **relativeToGround** – ερμηνεύει το υψόμετρο ως μία τιμή σε μέτρα πάνω από το έδαφος.
- **absolute** – ερμηνεύει το υψόμετρο ως μία τιμή σε μέτρα πάνω από την επιφάνεια της θάλασσας.

<gx:altitudeMode>

Αποτελεί μία επέκταση του KML, που επιτρέπει υψόμετρα ως προς τον πυθμένα της θάλασσας. Οι πιθανές τιμές είναι:

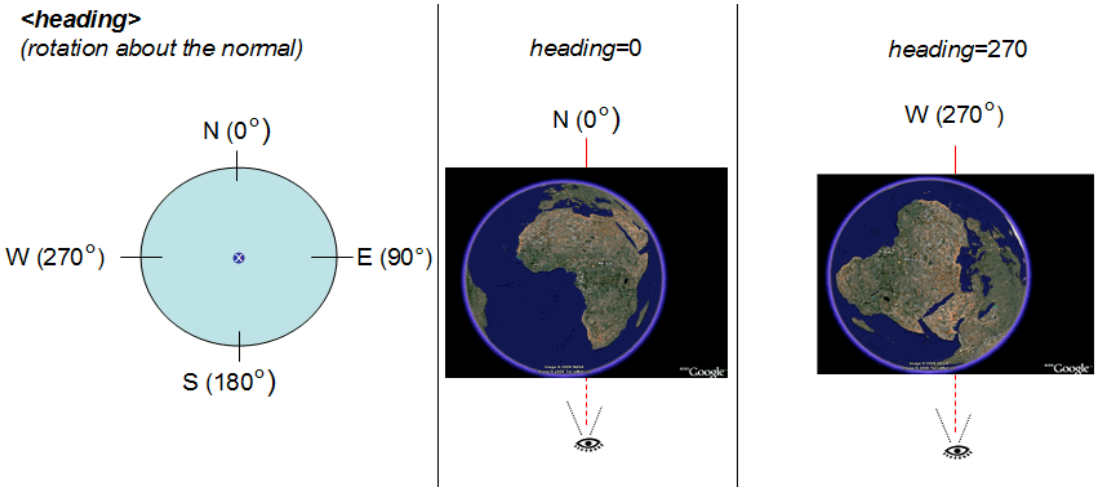
- **relativeToSeaFloor (ως προς τον πυθμένα της θάλασσας)** – ερμηνεύει το υψόμετρο ως μία τιμή σε μέτρα πάνω από τον πυθμένα της θάλασσας. Αν το σημείο είναι πάνω από ξηρά αντί για θάλασσα, το υψόμετρο θα ερμηνευθεί σα να είναι πάνω από το έδαφος.
- **clampToSeaFloor (κολλημένο στο βυθό της θάλασσας)** – ο προσδιορισμός υψομέτρου αγνοείται, και το σημείο θα τοποθετηθεί στον πυθμένα της θάλασσας. Αν το σημείο βρίσκεται σε ξηρά και όχι σε θάλασσα, το σημείο θα τοποθετηθεί στο έδαφος.

Το παρακάτω διάγραμμα απεικονίζει τα στοιχεία <range>, <tilt>, και <altitude>:



Εικόνα 2.8: Απεικόνιση των στοιχείων <range>, <tilt>, <altitude>

Το ακόλουθο διάγραμμα απεικονίζει το στοιχείο <heading>:



Εικόνα 2.9: Απεικόνιση στοιχείου <heading>

<Point>

Αποτελεί την γεωγραφική θέση, η οποία καθορίζεται από το γεωγραφικό μήκος και πλάτος και προαιρετικά το ύψος. Όταν ένα σημείο περιέχεται στο στοιχείο Placemark, το σημείο το ίδιο καθορίζει τη θέση του ονόματος και του εικονιδίου της σημάνσεως Placemark. Όταν εξάγεται ένα σημείο, συνδέεται με το έδαφος με μία γραμμή. Αυτή η «πρόσδεση» χρησιμοποιεί το τρέχον στυλ γραμμής LineStyle.

Στοιχεία σχετικά με το στοιχείο <Point>

<extrude>

Τελεστές boolean. Προσδιορίζει αν το σημείο θα συνδεθεί με το έδαφος με γραμμή. Για να εξαχθεί ένα σημείο, η τιμή του στοιχείου <altitudeMode> πρέπει να είναι είτε

σε σχέση με το έδαφος (relativeToGround), είτε σε σχέση με τη στάθμη της θάλασσας (relativeToSeaFloor), ή απόλυτη (absolute). Το σημείο τοποθετείται με φορά προς το κέντρο της γήινης σφαίρας.

<altitudeMode>

Καθορίζει πως ερμηνεύονται τα συστατικά του υψομέτρου στο στοιχείο <coordinates>. Πιθανές τιμές είναι:

- **clampToGround (κολλημένο στο έδαφος)** - (προεπιλεγμένο) υποδεικνύει την αγνόηση του προσδιορισμού υψομέτρου.
- **relativeToGround (ως προς το έδαφος)** – ορίζει το υψόμετρο του στοιχείου σε σχέση με το πραγματικό υψόμετρο του εδάφους σε μία συγκεκριμένη περιοχή. Για παράδειγμα, αν το υψόμετρο του εδάφους μιας περιοχής είναι ακριβώς στην επιφάνεια της θάλασσας και το υψόμετρο ενός σημείου έχει οριστεί στα 9 μέτρα, τότε το υψόμετρο για το εικονίδιο μιας σημάνσεως μέρους σημείου (point placemark) είναι 9 μέτρα με αυτή τη λειτουργία. Παρόλα αυτά, εάν η ίδια συντεταγμένη οριστεί σε μία περιοχή όπου η ανύψωση του εδάφους είναι 10 μέτρα από την επιφάνεια της θάλασσας, τότε η ανύψωση της συντεταγμένης είναι 19 μέτρα. Μία συνηθισμένη χρήση αυτού του τρόπου είναι για την τοποθέτηση στύλων τηλεφώνου ή τελεφερίκ.
- **absolute (απόλυτο)** – ορίζει το υψόμετρο του σημείου σε σχέση με την επιφάνεια της θάλασσας, ανεξάρτητα από την πραγματική ανύψωση της γης κάτω από το στοιχείο. Για παράδειγμα, αν το υψόμετρο μιας συντεταγμένης οριστεί στα 10 μέτρα με την επιλογή του απόλυτου υψομέτρου, το εικονίδιο του σημείου σημάνσεως μέρους θα εμφανιστεί στο επίπεδο του εδάφους αν το έδαφος κάτω από αυτό είναι επίσης 10 μέτρα πάνω από την επιφάνεια της θάλασσας. Αν το έδαφος είναι 3 μέτρα πάνω από την επιφάνεια της θάλασσας, η σήμανση μέρους θα εμφανιστεί ανυψωμένη κατά 7 μέτρα πάνω από το έδαφος. Μία συνηθισμένη χρήση αυτής της επιλογής είναι για την τοποθέτηση αεροσκαφών.

<gx:altitudeMode> (Όμοια με παραπάνω)

<coordinates>

Πρόκειται για ένα σύνολο που αποτελείται από τιμές κινητής υποδιαστολής του γεωγραφικού μήκους (*longitude*), του γεωγραφικού πλάτους (*latitude*) και του υψομέτρου (*altitude*) κατά σειρά. Το γεωγραφικό μήκος και πλάτος είναι σε μοίρες, όπου:

- -180≤ *longitude* ≤ 180
- -90≤ *latitude* ≤ 90
- *altitude* (προαιρετικά) σε μέτρα ως προς την επιφάνεια της θάλασσας.

<Polygon>

Ένα πολύγωνο καθορίζεται από ένα εξωτερικό όριο και κανένα ή περισσότερα εσωτερικά όρια. Τα όρια με τη σειρά τους καθορίζονται με το στοιχείο <LinearRings>. Όταν εξάγεται ένα πολύγωνο, τα όριά του συνδέονται στο έδαφος για να σχηματιστούν πρόσθετα πολύγωνα, γεγονός που του δίνει την εμφάνιση κτιρίου ή κουτιού. Τα εξαγόμενα πολύγωνα χρησιμοποιούν το στοιχείο <PolyStyle> για το χρώμα, τις χρωματικές λειτουργίες και το γέμισμά τους.

Το στοιχείο <coordinates> για τα πολύγωνα πρέπει να καθοριστεί με φορά αντίθετη του ρολογιού. Τα πολύγωνα ακολουθούν τον κανόνα του δεξιού χεριού, ο οποίος ορίζει ότι αν τοποθετηθούν τα δάχτυλα του δεξιού χεριού προς την κατεύθυνση στην οποία καθορίζονται οι συντεταγμένες, ο αντίχειρας δείχνει στη γενική κατεύθυνση της γεωμετρικής καθέτου του πολυγώνου. Η Google Earth γεμίζει μόνο την εμπρόσθια επιφάνεια των πολυγώνων κι έτσι το επιθυμητό αποτέλεσμα επιτυγχάνεται μόνο όταν οι συντεταγμένες προσδιορίζονται με την σωστή σειρά. Διαφορετικά το πολύγωνο θα εμφανιστεί γκρι.

Στοιχεία σχετικά με το <Polygon>

<extrude>

Τελεστές boolean. Προσδιορίζει αν το σημείο θα συνδεθεί με το έδαφος με γραμμή. Για να εξαχθεί ένα σημείο, η τιμή του στοιχείου <altitudeMode> πρέπει να είναι είτε σε σχέση με το έδαφος (relativeToGround), είτε σε σχέση με τη στάθμη της θάλασσας (relativeToSeaFloor), ή απόλυτη (absolute). Το σημείο τοποθετείται με φορά προς το κέντρο της γήινης σφαίρας.

<tessellate>

Αυτό το πεδίο δε χρησιμοποιείται από το πολύγωνο. Για να ακολουθήσει ένα πολύγωνο το έδαφος ώστε να ενεργοποιηθεί η δημιουργία τρισδιάστατου μοντέλου με πολλά πολύγωνα τα οποία συνδέονται μεταξύ τους (tessellation), πρέπει να προσδιοριστεί μία υψομετρική λειτουργία clampToGround ή clampToSeaFloor.

<altitudeMode> (όπως αναλύθηκε παραπάνω)

<gx:altitudeMode> (όπως αναλύθηκε παραπάνω)

<outerBoundaryIs>

Περιέχει ένα στοιχείο <LinearRing>.

<innerBoundaryIs>

Περιέχει ένα στοιχείο <LinearRing>. Ένα πολύγωνο μπορεί να περιέχει πολλαπλά <innerBoundaryIs> στοιχεία, τα οποία δημιουργούν πολλαπλές εγκοπές στο πολύγωνο.

<LinearRing>

Καθορίζει μία κλειστή γραμμική χορδή, συνήθως το εξωτερικό όριο του πολυγώνου. Προαιρετικά, το στοιχείο αυτό μπορεί να χρησιμοποιηθεί ως το εσωτερικό όριο ενός πολυγώνου για να δημιουργηθούν τρύπες στο πολύγωνο. Ένα πολύγωνο μπορεί να περιέχει πολλαπλά <LinearRing> στοιχεία που χρησιμοποιούνται ως εσωτερικά όρια.

Στοιχεία σχετικά με το <LinearRing>

<gx:altitudeOffset>

Μία επέκταση KML η οποία τροποποιεί τον τρόπο απόδοσης υψομετρικών τιμών. Αυτό το αντιστάθμισμα επιτρέπει να μετακινηθεί ένα ολόκληρο LinearRing πάνω ή κάτω ως μία μονάδα χωρίς να τροποποιεί όλες τις συντεταγμένες χωριστά που συγκροτούν το LinearRing. Οι μονάδες είναι σε μέτρα.

<extrude> (όπως αναλύθηκε παραπάνω)

<tessellate> (όπως αναλύθηκε παραπάνω)

<altitudeMode> (όπως αναλύθηκε παραπάνω)

<gx:altitudeMode> (όπως αναλύθηκε παραπάνω)

<coordinates>

Τέσσερις ή περισσότερες ομάδες, η κάθε μία αποτελούμενη από δεκαδικούς αριθμούς που αντιστοιχούν στο γεωγραφικό μήκος, το γεωγραφικό πλάτος και το υψόμετρο. Το υψόμετρο είναι προαιρετικό. Δεν πρέπει να παρεμβάλλονται κενά μέσα στις ομάδες συντεταγμένων. Η τελευταία συντεταγμένη πρέπει να είναι ίδια με την πρώτη συντεταγμένη. Οι συντεταγμένες εκφράζονται μόνο σε δεκαδικές μοίρες μόνο.

2.2.3 Περιγραφική HTML στα Placemarks

Στη σήμανση μέρους κειμένου μπορεί ο χρήστης να προσθέσει συνδέσεις, μεγέθη γραμματοσειράς, στυλ και χρώματα, καθώς και να ορίσει στοίχιση κειμένου και πίνακες.

Εάν ο χρήστης θέλει να γράψει ένα πρότυπο HTML μέσα σε ένα tag <description> , μπορεί να το βάλει μέσα σε ένα tag CDATA.

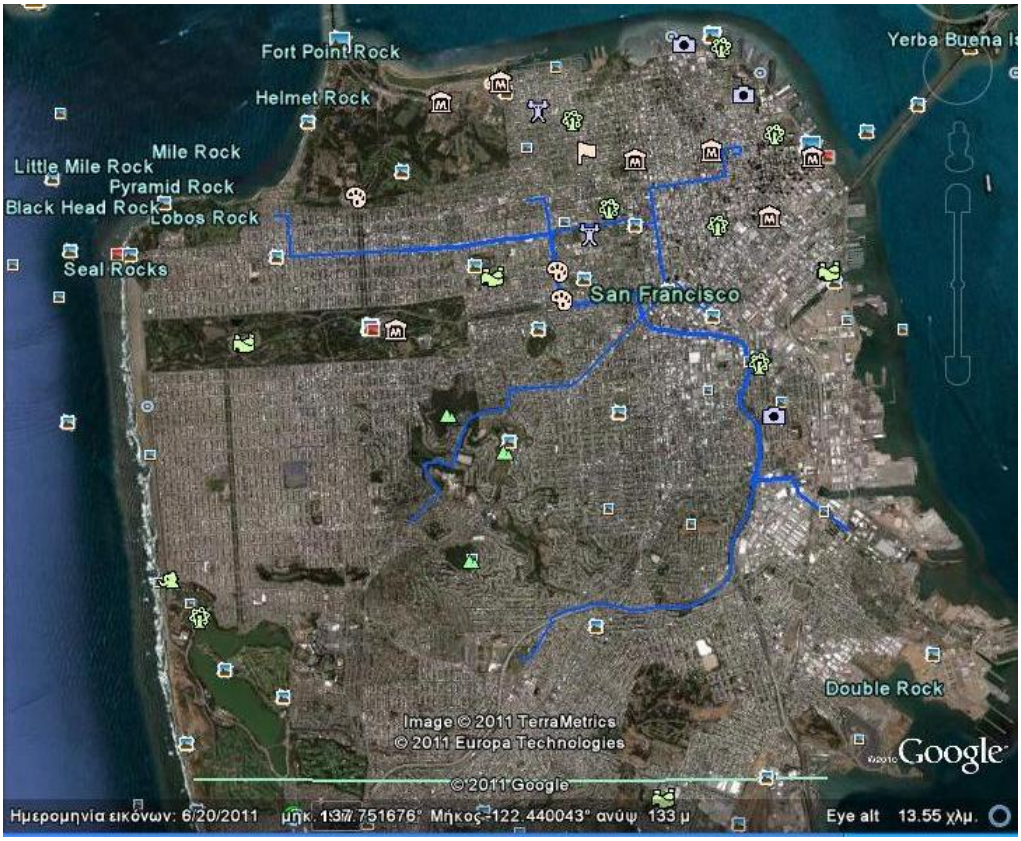
2.2.4 Διαδρομές - Paths

Πολλοί διαφορετικοί τύποι διαδρομών μπορούν να δημιουργηθούν στο Google Earth. Στο KML, δημιουργείται μια διαδρομή με ένα στοιχείο <LineString> .

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
  <Document>
    <name>Paths</name>
    <description>Examples of paths. Note that the tessellate tag is
by default
      set to 0. If you want to create tessellated lines, they must be
authored
      (or edited) directly in KML.</description>
    <Style id="yellowLineGreenPoly">
      <LineStyle>
        <color>7f00ffff</color>
        <width>4</width>
      </LineStyle>
      <PolyStyle>
        <color>7f00ff00</color>
      </PolyStyle>
    </Style>
    <Placemark>
      <name>Absolute Extruded</name>
      <description>Transparent green wall with yellow
outlines</description>
      <styleUrl>#yellowLineGreenPoly</styleUrl>
      <LineString>
        <extrude>1</extrude>
        <tessellate>1</tessellate>
        <altitudeMode>absolute</altitudeMode>
        <coordinates> -112.2550785337791,36.07954952145647,2357
          -112.2552505069063,36.08260761307279,2357
          -112.2564540158376,36.08395660588506,2357
          -112.2580238976449,36.08511401044813,2357
          -112.2595218489022,36.08584355239394,2357
          -112.2608216347552,36.08612634548589,2357
          -112.262073428656,36.08626019085147,2357
          -112.2633204928495,36.08621519860091,2357
          -112.2644963846444,36.08627897945274,2357
          -112.2656969554589,36.08649599090644,2357
        </coordinates>
      </LineString>
    </Placemark>
  </Document>
</kml>
```

Η <tessellate> ετικέτα διασπά τη γραμμή σε μικρότερα κομμάτια, και η <extrude> ετικέτα επεκτείνει τη γραμμή προς το έδαφος.

Ακολουθεί ένα παράδειγμα απεικόνισης διαδρομών στην ευρύτερη περιοχή του San Fransisco.



Εικόνα 2.10: Απεικόνιση διαδρομών στην περιοσή του San Fransisco στο Google Earth

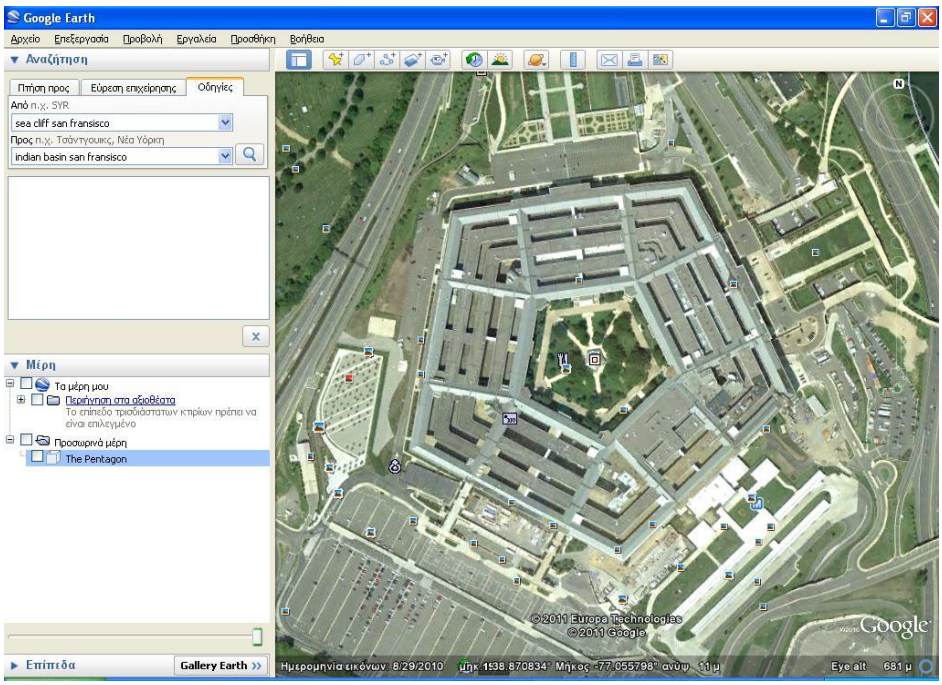
2.2.5 Πολύγωνα - Polygons

Τα πολύγωνα είναι χρήσιμα για τη δημιουργία απλών κτιρίων και άλλων σχημάτων. Ακολουθεί ένα παράδειγμα.

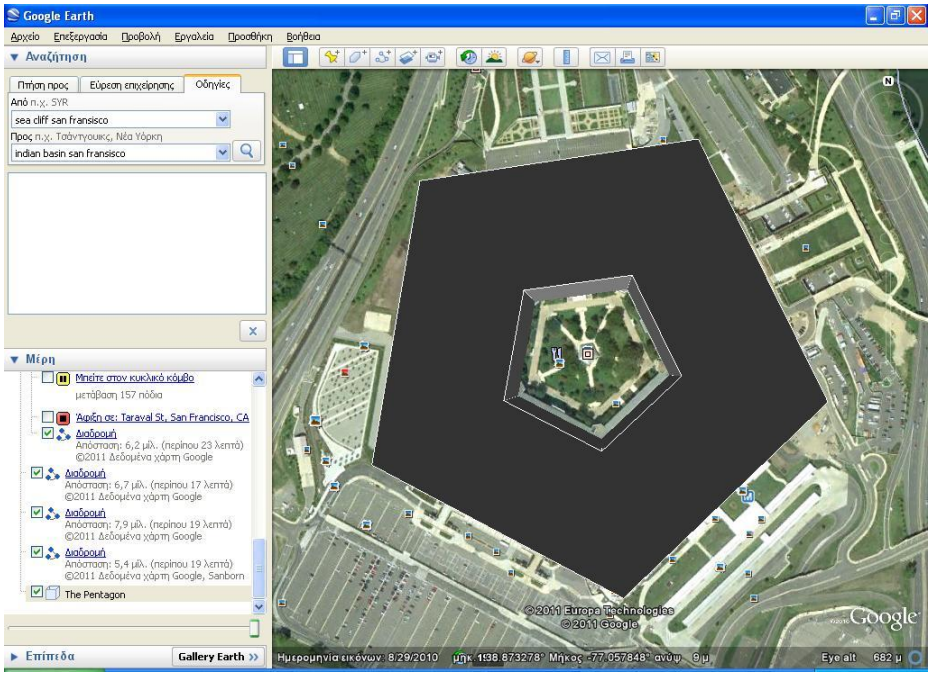
```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
  <Placemark>
    <name>The Pentagon</name>
    <Polygon>
      <extrude>1</extrude>
      <altitudeMode>relativeToGround</altitudeMode>
      <outerBoundaryIs>
        <LinearRing>
          <coordinates>
            -77.05788457660967,38.87253259892824,100
            -77.05465973756702,38.87291016281703,100
            -77.05315536854791,38.87053267794386,100
            -77.05552622493516,38.868757801256,100
            -77.05844056290393,38.86996206506943,100
            -77.05788457660967,38.87253259892824,100
          </coordinates>
        </LinearRing>
      </outerBoundaryIs>
      <innerBoundaryIs>
        <LinearRing>
```

```
          <coordinates>
            -77.05668055019126,38.87154239798456,100
            -77.05542625960818,38.87167890344077,100
            -77.05485125901024,38.87076535397792,100
            -77.05577677433152,38.87008686581446,100
            -77.05691162017543,38.87054446963351,100
            -77.05668055019126,38.87154239798456,100
          </coordinates>
        </LinearRing>
      </innerBoundaryIs>
    </Polygon>
  </Placemark>
</kml>
```

Ακολουθεί η απεικόνιση του παραπάνω παραδείγματος στο Google Earth.



Εικόνα 2.11: Απεικόνιση Πενταγώνου στο Google Earth πριν την δημιουργία πολυγώνου



Εικόνα 2.12: : Απεικόνιση Πενταγώνου στο Google Earth μετά την δημιουργία πολυγώνου

2.2.6 Στυλ για Γεωμετρία

Το στυλ είναι ένα σημαντικό μέρος στον τρόπο εμφάνισης των δεδομένων. Ορίζοντας ένα στυλ στην αρχή ενός εγγράφου KML και επίσης καθορίζοντας μία ταυτότητα ID για αυτό, μπορεί ο χρήστης να χρησιμοποιήσει αυτό το στυλ στη γεωμετρία, τις σημάνσεις μερών και τις επικαλύψεις που έχουν οριστεί αλλού στο έγγραφο. Επειδή περισσότερα από ένα στοιχείο μπορούν να χρησιμοποιήσουν το ίδιο στυλ, στυλ που ορίζονται και χρησιμοποιούνται με αυτόν τον τρόπο αναφέρονται ως κοινόχρηστα στυλ. Ο χρήστης ορίζει ένα δεδομένο στυλ μία φορά, και στη συνέχεια κάνει αναφορά σε αυτό πολλές φορές, χρησιμοποιώντας το στοιχείο <styleUrl>. Εάν ο ορισμός στυλ είναι μέσα στο ίδιο αρχείο, προηγείται το Αναγνωριστικό στυλ ID με ένα σύμβολο #. Εάν ο ορισμός στυλ είναι σε ένα εξωτερικό αρχείο, περιλαμβάνει την πλήρη διεύθυνση στο στοιχείο <styleUrl>.
Εδώ πρέπει να σημειωθεί ότι το στοιχείο <Style> είναι θυγατρικό του <Placemark>.

3

PHP

3.1 Τι είναι η PHP

Ο όρος PHP αποτελεί ακρωνύμιο των λέξεων *Hypertext Preprocessor* και είναι μία ευρέως διαδεδομένη, ανοικτού κώδικα, γενικής χρήσης χειρόγραφη γλώσσα η οποία είναι ειδικά κατάλληλη για διαδικτυακό προγραμματισμό και μπορεί να ενσωματωθεί στην HTML. Η μεγάλη διαφορά της με πανομοιότυπες γλώσσες προγραμματισμού (π.χ. C, Perl, Python) είναι ότι ο κώδικας εκτελείται στον διακομιστή, παράγοντας και κατά συνέπεια συνδυάζοντας HTML το οποίο αποστέλλεται στη συνέχεια στον πελάτη. Ο πελάτης θα λάβει τα αποτελέσματα από την εφαρμογή αυτών των ενεργειών, αλλά δε θα ξέρει τον κώδικα που κρύβεται από πίσω. Επίσης, πρέπει να σημειωθεί ότι η χρήση της είναι ιδιαίτερα εύκολη για τη δημιουργία εφαρμογών.

Οι σελίδες PHP περιέχουν HTML με ενσωματωμένο κώδικα. Ο κώδικας PHP περικλείεται σε ειδικές οδηγίες επεξεργασίας αρχής και τέλους <?php και ?> αντίστοιχα που επιτρέπουν στον ενδιαφερόμενο να μεταπηδήσει προς και από την PHP λειτουργία.

Το μεγαλύτερο πλεονέκτημα στη χρήση της PHP είναι ότι είναι πάρα πολύ απλή για έναν αρχάριο, αλλά ταυτόχρονα προσφέρει πολλά προηγμένα χαρακτηριστικά για έναν επαγγελματία προγραμματιστή.

3.2 Πριν από την Εμφάνιση της PHP

Πριν ακόμα την εμφάνιση της PHP, υπήρχε μεγάλη δραστηριότητα στο Web programming. Παλαιότερα, ο κώδικας για την ευκολότερη επεξεργασία των δεδομένων μιας φόρμας γραφόταν στη γλώσσα C. Αλλά, ενώ ο κώδικας αυτός ήταν πολύ γρήγορος στην εκτέλεσή του, ήταν πολύ δύσκολος και πολύπλοκος στο γράψιμό του. Ο κύριος λόγος γι' αυτό ήταν ότι η C δεν είχε σχεδιασθεί ειδικά για το Web και έτσι δεν υπήρχε έτοιμος κώδικας για την εκτέλεση κάποιων κοινών εργασιών και ο προγραμματιστής (Web developer) έπρεπε να τα κάνει όλα μόνος του.

Ο προγραμματισμός στην C βέβαια ήταν πολύ καλύτερος από την κατάσταση που υπήρχε μέχρι τότε, καθώς το πρωτόκολλο HTTP είναι ένα stateless σύστημα, που σημαίνει ότι δεν αποθηκεύει καθόλου δεδομένα ανάμεσα στις σελίδες και έτσι και

αυτό ακόμα το γράψιμο κώδικα σε C για την αποστολή δεδομένων ανάμεσα στις σελίδες ήταν ένα σημαντικό βήμα μπροστά.

Αυτό το πρόβλημα αντιμετωπίστηκε κάπως με μια ευκολότερη γλώσσα υψηλού επιπέδου, την Perl, όπου τα αρχικά προέρχονται από τις λέξεις "Practical Extraction and Report Language". Η Perl, αν και αρχικά δημιουργήθηκε ως μια γλώσσα επεξεργασίας κειμένου, είχε δυνατότητες για επεξεργασία των καταχωρήσεων μιας HTML φόρμας και όχι μόνο. Η σχεδίαση της Perl ήταν απλή : ένα script της Perl μπορούσε να κάνει όλες τις απαραίτητες λειτουργίες μιας ιστοσελίδας και μπορούσαμε να ενσωματώσουμε σ' αυτό όποιον κώδικα της HTML θέλαμε. Η Perl διέθετε ακόμη πάρα πολλές συναρτήσεις (functions) για να γίνονται εύκολα πολλές εργασίες και έτσι έγινε σύντομα πολύ δημοφιλής ανάμεσα στους Web developers.

Αν και αποτέλεσε ένα σημαντικό βήμα μπροστά για το Web development, η Perl απείχε ακόμα πολύ από το ιδανικό. Ο τρόπος εργασίας της, δηλαδή το ότι «μια γλώσσα μπορεί να τα κάνει όλα» σήμαινε ότι δεν ήταν σχεδιασμένη για το Web και πολλοί προγραμματιστές της Perl προτιμούσαν τον δομημένο, εύκολο στην ανάγνωση προγραμματισμό από τον προγραμματισμό της «μίας γραμμής», όπου υπήρχε συμπυκνωμένος και δύσκολος στην κατανόηση κώδικας. Ίσως το μεγαλύτερο ελάττωμά της ήταν ότι η Perl ήταν Perl-centric, που σημαίνει ότι για να δημιουργηθεί HTML κώδικας, έπρεπε να ενσωματώσουμε τον HTML κώδικα μέσα στην Perl.

3.3 Οι Πρώτες Εκδόσεις της PHP

Η αρχική έκδοση της PHP σχεδιάσθηκε και δημιουργήθηκε από τον Rasmus Lerdorf στα μέσα της δεκαετίας του 1990 ως ένας τρόπος για να μπορούν να γίνονται διάφορες κοινές εργασίες στο Web ευκολότερα και με λιγότερες επαναλήψεις. Τότε, ο κύριος σκοπός ήταν να υπάρχει η λιγότερη δυνατή ποσότητα λογικής στην επίτευξη του αποτελέσματος και αυτό οδήγησε την PHP στο να γίνει HTML-centric, δηλαδή ο κώδικας της PHP ήταν ενσωματωμένος μέσα στον κώδικα της HTML.

Η πρώτη δημοφιλής έκδοση της PHP ονομάσθηκε PHP/FI 2.0, από τα αρχικά Personal Home Page/Form Interpreter. Το κύριο χαρακτηριστικό αυτής της έκδοσης ήταν ότι ο PHP/FI parser ήταν γραμμένος κυρίως με το χέρι και έτσι δημιουργούνταν συχνά λάθη. Ο όρος parser (αναλυτής/επεξεργαστής) αναφέρεται στον μηχανισμό ο οποίος δέχεται ένα script (κώδικα) και το μετατρέπει σε κάτι που μπορεί να κατανοήσει ο υπολογιστής.

Μερικά από αυτά τα προβλήματα επιλύθηκαν στην έκδοση 3, όταν ο Zeev Suraski και ο Andi Gutmans ξαναέγραψαν την PHP από την αρχή χρησιμοποιώντας καινούργια εργαλεία. Η PHP απέκτησε πολλούς οπαδούς και όταν εμφανίσθηκε η καινούργια έκδοση στα μέσα του 2000, είχε ήδη εγκατασταθεί σε περισσότερα από 2,5 εκατομμύρια Web-site domains, σε σύγκριση με τα 250.000 μόλις 18 μήνες νωρίτερα.

Στα μέσα του έτους 2000, εμφανίσθηκε η έκδοση PHP 4, που είχε μεγάλες διαφορές από την PHP 3. Πολύ δουλειά έγινε στο να διασφαλισθεί η προς τα πίσω

συμβατότητα του κώδικα με τα παλιά scripts της PHP και έτσι η αναβάθμιση από την PHP 3 στην PHP 4 ήταν πολύ πιο ομαλή απ' ό,τι ήταν η αναβάθμιση από την PHP/FI στην PHP 3.

Η σημαντικότερη ίσως αλλαγή που έγινε στην PHP 4 ήταν η καθιέρωση της Μηχανής Zend (Zend Engine), η οποία δημιουργήθηκε από την εταιρεία Zend, των Zeev Suraski και Andi Gutmans. Το όνομα Zend προέρχεται από τις λέξεις ZEev και aNDi και ο σκοπός της μηχανής ήταν να προωθήσει την PHP στο εταιρικό περιβάλλον, ώστε να υπάρχει πολύ περισσότερη ευελιξία στη γλώσσα απ' ό,τι παλαιότερα.

Μια άλλη σημαντική καινοτομία ήταν ότι η PHP μπορούσε τώρα να εκτελεστεί σε πολλούς Web servers, όπως Apache 1.3.x, Apache 2, Microsoft's IIS, Zeus, AOLServer κ.ά. Επίσης, η απόδοση της γλώσσας έκανε ένα πολύ μεγάλο άλμα μπροστά εξαιτίας δύο παραγόντων. Πρώτα, ενώ η PHP 3 χρησιμοποιούσε τη λογική "εκτέλεση ενώ γίνεται διερμήνευση", που σήμαινε ότι η PHP διάβαζε μια γραμμή πηγαίου κώδικα, τον διερμήνευε, τον εκτελούσε, διάβαζε μια άλλη γραμμή κώδικα, τον διερμήνευε, τον εκτελούσε, διάβαζε την επόμενη γραμμή κοκ. Αυτό σήμαινε ότι ο κώδικας διαβαζόταν και διερμηνευόταν πολλές φορές, χωρίς να υπάρχει κανένας απολύτως λόγος.

Η PHP 4 υιοθέτησε τη λογική "μεταγλώττιση πρώτα, εκτέλεση αργότερα", όπου πρώτα διάβαζε ολόκληρο το script και το μεταγλώττιζε σε ενδιάμεσο κώδικα (byte code) πριν το εκτελέσει. Αυτό είχε ως αποτέλεσμα μεγάλη αύξηση στην ταχύτητα. Ο κώδικας "byte code" αποτελεί μια εσωτερική αναπαράσταση ενός script που η PHP μπορεί να κατανοήσει εύκολα και είναι συνήθως πολύ μεγαλύτερος σε μήκος από το ίδιο το script καθώς η κάθε εντολή της PHP διασπάται (αναλύεται) σε πολλές άλλες πιο απλές εντολές.

Επίσης, η PHP 4 εισήγαγε την πολυεπεξεργασία (multi-threading), όπου μπορούν κάποιες συναρτήσεις να εκτελούνται ανεξάρτητα από το κυρίως script. Η PHP συνεχίζει να προχωράει ατάραχη και η τρέχουσα έκδοσή της είναι η 4.3.9, είναι δε εγκατεστημένη σε 9,5 εκατομμύρια περίπου Web servers σ' όλον τον κόσμο.

3.4 Η Τρέχουσα Έκδοση της PHP

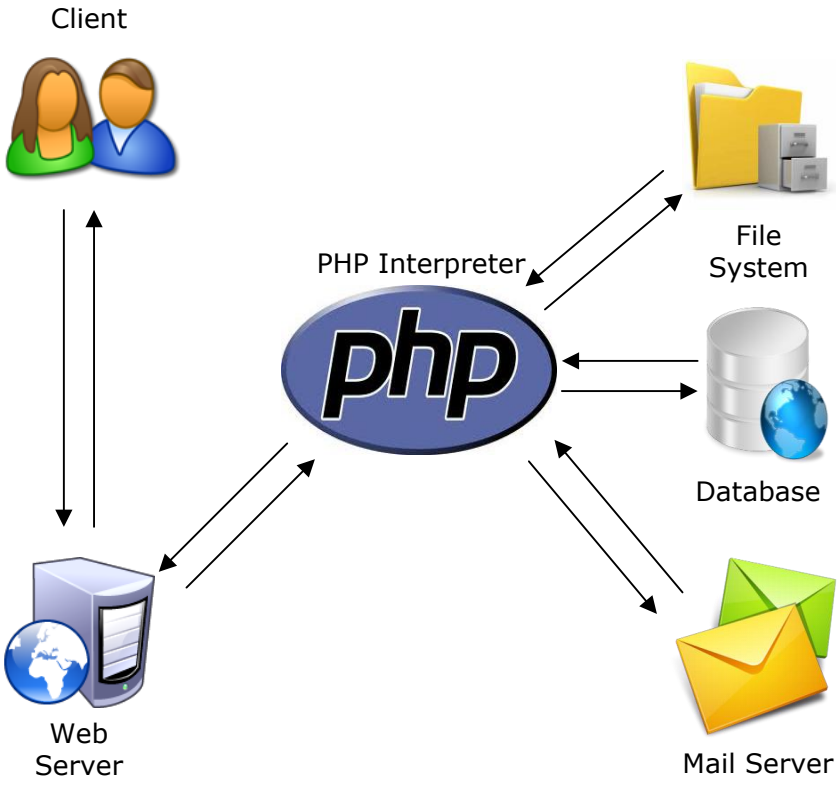
Η PHP 5 ήταν ένα μεγάλο βήμα μπροστά για τη γλώσσα, αν και όχι τόσο μεγάλο όσο η μετάβαση από την PHP 3 στην PHP 4. Η PHP 5 προσφέρει scripts για αντικειμενοστραφή προγραμματισμό (object-oriented). Επίσης, υπάρχει μια μεγάλη ποικιλία από συναρτήσεις για αντικείμενα (objects) που τα κάνει πολύ πιο ευέλικτα και εύκολα στη χρήση τους. Ακόμη, τα αντικείμενα αντιμετωπίζονται πάντα ως αναφορές (references) ώστε να βοηθηθούν οι προγραμματιστές που δυσκολεύονται να εργαστούν με τα αντικείμενα.

3.5 Η Σύνδεση με την HTML

Ο κώδικας της PHP ενσωματώνεται μέσα στον κώδικα της HTML σε ειδικά μπλοκ κώδικα, που περικλείονται από τα σύμβολα <?php και ?>, όπως φαίνεται παρακάτω:

```
<html>
<body>
<p>Καλώς ήρθες,<?php print $Name; ?></p>
</body>
</html>
```

Δεν θα πρέπει να ξεχνάμε ότι ο κώδικας της PHP εκτελείται εξ ολοκλήρου στον διακομιστή (server) και έτσι ο πελάτης (client) λαμβάνει μόνο το τελικό αποτέλεσμα από την εκτέλεση του script. Αυτό σημαίνει με απλά λόγια ότι οι τελικοί χρήστες δεν μπορούν ποτέ να δουν τον πηγαίο κώδικα (source code) της PHP.



Εικόνα 3.1: Τρόπος λειτουργίας της PHP

3.6 Η Διερμήνευση και η Μεταγλώττιση

Η PHP χρησιμοποιεί μια μίξη από διερμήνευση (interpretation) και μεταγλώττιση (compilation) έτσι ώστε να μπορέσει να δώσει στους προγραμματιστές τον καλύτερο δυνατό συνδυασμό απόδοσης και ευελιξίας. Στο παρασκήνιο, η PHP μεταγλωττίζει το script σε μια σειρά από εντολές (instructions), που είναι γνωστές με τον όρο opcodes, οι οποίες εντολές εκτελούνται μία-μία μέχρι να τελειώσει το script.

Αυτό είναι κάτι διαφορετικό από τις παραδοσιακές γλώσσες που μεταγλωττίζονται, όπως είναι η C++, όπου ο κώδικας μεταγλωττίζεται σε εκτελέσιμο κώδικα μηχανής, ενώ η PHP μεταγλωττίζει εκ νέου το script κάθε φορά που αυτό απαιτείται. Αυτή η συνεχής μεταγλώττιση μπορεί να φαίνεται ως απώλεια χρόνου, αλλά δεν είναι καθόλου κακή καθώς δεν χρειάζεται να κάνουμε συνέχεια εμείς τη μεταγλώττιση των scripts όταν γίνονται κάποιες αλλαγές σ' αυτά.

3.7 Πώς γράφεται η PHP

Όπως είδαμε και νωρίτερα, ο κώδικας της PHP ενσωματώνεται μέσα στην HTML και τα στοιχεία της PHP που υπάρχουν σ' ένα script είναι σαν νησίδες κώδικα (code islands), δηλαδή αυτόνομα κομμάτια κώδικα που μπορούν να εκτελεστούν ανεξάρτητα από την περιβάλλουσα HTML. Αυτό βέβαια δεν σημαίνει ότι ο κώδικας της PHP δεν μπορεί να επηρεάσει την HTML, κάθε άλλο μάλιστα.

Τα scripts της PHP αποθηκεύονται συνήθως με την επέκταση .php και κάθε φορά που ο Web server πρέπει να στείλει ένα αρχείο που τελειώνει σε .php, πρώτα το στέλνει στον διερμηνευτή (interpreter) της PHP, ο οποίος εκτελεί τον κώδικα της PHP που υπάρχει στο script πριν επιστρέψει το παραγόμενο αρχείο στον τελικό χρήστη. Η κάθε γραμμή του PHP κώδικα είναι γνωστή ως εντολή (statement) και τελειώνει με το σύμβολο ;.

Οι μεταβλητές (variables) της PHP ξεκινούν με το σύμβολο $ ακολουθούμενο από ένα γράμμα ή τον χαρακτήρα _ (underscore) και μετά από έναν συνδυασμό από γράμματα, ψηφία και τον χαρακτήρα _. Αυτό σημαίνει ότι δεν μπορούμε να ξεκινήσουμε το όνομα μιας μεταβλητής με ψηφίο.

3.8 Πώς δουλεύει η PHP

Η PHP είναι μια γλώσσα "server-side". Αυτό σημαίνει ότι ο κώδικας PHP που περιέχει μια σελίδα εκτελείται στον server (όπου είναι αποθηκευμένη η σελίδα), ενώ τα αποτελέσματα εμφανίζονται με μορφή HTML στον τελικό χρήστη.

Ο τρόπος με τον οποίο δουλεύει ένας web server (απαραίτητο λογισμικό για την επεξεργασία και τη λειτουργία μιας ιστοσελίδας) στον οποίο υπάρχει εγκατεστημένη η PHP είναι ο εξής: ο χρήστης "καλεί" μια σελίδα και ο server κάνει τις αντίστοιχες διεργασίες, για να παρουσιάσει το επιθυμητό αποτέλεσμα πίσω στο χρήστη. Μια απλή σελίδα HTML παρακάμπτει το εγκατεστημένο λογισμικό της PHP στον web server και εμφανίζεται όπως ακριβώς είναι στο χρήστη.

3.9 Τι μπορεί να κάνει η PHP

Η PHP κυρίως επικεντρώνεται στον προγραμματισμό και τις ενέργειες του διακομιστή (server-side scripting), επομένως μπορεί ο χρήστης να κάνει οτιδήποτε, μπορεί να κάνει οποιοδήποτε άλλο CGI πρόγραμμα. Αλλά η PHP μπορεί να κάνει πολλά περισσότερα.

Υπάρχουν τρεις βασικές περιπτώσεις στις οποίες χρησιμοποιούνται PHP σενάρια:

- Server-side scripting. Είναι το πιο συνηθισμένο. Χρειάζονται τρία πράγματα για να δουλέψει αυτό, ο PHP parser, ένας web server κι ένας web browser. Ο χρήστης πρέπει να "τρέξει" το διακομιστή ιστοσελίδων. Ο χρήστης μπορεί να έχει πρόσβαση στο αποτέλεσμα του προγράμματος PHP μέσω ενός προγράμματος περιήγησης. Όλα αυτά μπορούν να εκτελούνται στο σταθερό υπολογιστή καθώς ο ενδιαφερόμενος θα πειραματίζεται με τον προγραμματισμό σε PHP.

- Command line scripting. Μπορεί να τρέξει ένα PHP πρόγραμμα χωρίς κάποιον διακομιστή ή πρόγραμμα περιήγησης. Χρειάζεται μόνο ένα πρόγραμμα ανάλυσης PHP (PHP parser).
- Writing desktop applications. Η PHP δεν είναι η καλύτερη γλώσσα για τη δημιουργία εφαρμογών επιφάνειας εργασίας με ένα γραφικό περιβάλλον χρήστη, αλλά ο χρήστης που γνωρίζει καλά την PHP μπορεί να χρησιμοποιήσει κάποια προηγμένα PHP χαρακτηριστικά στις εφαρμογές του καθώς επίσης και PHP-GTK για να γράψει κάποια προγράμματα. Η PHP-GTK είναι μία επέκταση της PHP, μη διαθέσιμη όμως στην κεντρική διανομή.

Η PHP μπορεί να χρησιμοποιηθεί σε όλα τα βασικά λειτουργικά συστήματα, όπως Linux, Microsoft Windows, Mac OS X, RISC OS. Η PHP υποστηρίζει επίσης τους περισσότερους διακομιστές δικτύου (web servers), μεταξύ των οποίων Apache, IIS, και πολλοί άλλοι. Η PHP λειτουργεί είτε ως μία λειτουργική μονάδα είτε ως CGI επεξεργαστής.

Επομένως με την PHP, ο προγραμματιστής έχει την ελευθερία να διαλέξει ένα λειτουργικό σύστημα κι έναν διακομιστή δικτύου. Επιπλέον, έχει την ευκαιρία να χρησιμοποιήσει δομημένο προγραμματισμό ή αντικειμενοστραφή προγραμματισμό, ή ένα μίγμα και των δύο.

Με την PHP ο χρήστης δεν περιορίζεται στην εξαγωγή HTML. Οι δυνατότητες της PHP περιλαμβάνουν την εξαγωγή εικόνων, PDF αρχείων, ακόμη και ταινίες Flash. Μπορεί επίσης να εξάγει εύκολα ένα κείμενο, όπως για παράδειγμα XHTML και οποιοδήποτε άλλο XML αρχείο. Η PHP μπορεί να παράγει μόνη της αυτά τα αρχεία, και να τα αποθηκεύσει στο σύστημα αρχείων (file system), αντί να τα εκτυπώσει, δημιουργώντας μία κρυφή μνήμη για το δυναμικό περιεχόμενο.

Ένα από τα πιο δυνατά και σημαντικά χαρακτηριστικά της PHP είναι ότι υποστηρίζει ένα ευρύ φάσμα βάσεων δεδομένων. Η δημιουργία μίας ιστοσελίδας στηριζόμενη σε βάσεις δεδομένων είναι πολύ απλή με τη χρήση συγκεκριμένων επεκτάσεων των βάσεων δεδομένων (π.χ. mysql), ή με τη σύνδεση σε οποιαδήποτε βάση δεδομένων υποστηρίζοντας την Open Database Connection μέσω της ODBC επέκτασης.

Η PHP υποστηρίζει επίσης την επικοινωνία με άλλες υπηρεσίες χρησιμοποιώντας πρωτόκολλα όπως LDAP, IMAP, SNMP, NNTP, POP3, HTTP, COM (στα Windows) και πολλά άλλα. Όσον αφορά τη διασύνδεση, η PHP υποστηρίζει επίσης τη δημιουργία αντικειμένων και τη χρήση αυτών ως PHP αντικείμενα.

Η PHP έχει χρήσιμες δυνατότητες επεξεργασίας κειμένου, οι οποίες περιλαμβάνουν επεκτάσεις και εργαλεία για την ανάλυση και την πρόσβαση του χρήστη σε XML έγγραφα. Η PHP τυποποιεί όλες τις XML επεκτάσεις με σταθερή βάση την libxml2, και επεκτείνει τη δυνατότητα του χρήστη να ορίσει την προσθήκη SimpleXML, XMLReader και XMLWriter υποστήριξης.

Τα χαρακτηριστικά και τα προτερήματα της PHP είναι πολλά περισσότερα από αυτά που αναφέρθηκαν παραπάνω. Αυτός είναι και ο λόγος που είναι τόσο διαδεδομένη και τόσο συνηθισμένη γλώσσα προγραμματισμού.

3.10 Παραδείγματα PHP

Στη συνέχεια ακολουθούν ορισμένα χαρακτηριστικά παραδείγματα που αφορούν βασικές λειτουργίες της PHP και μπορούν να αποτελέσουν αρωγό για έναν αρχάριο της PHP για την ευκολότερη εξοικείωσή του με τη γλώσσα .

Writing text to a file

```
<?php
$handle = fopen("myfile.txt", 'w+');

if($handle)
{

    if(!fwrite($handle, "Student Name: Mark Fendisen"))
        die("couldn't write to file.");

    echo "success writing to file";
}

?>
<pre>
```

Open file for reading

```
<?php
$fh = fopen("myfile.txt", "r");

if($fh==false)
    die("unable to create file");
?>
```

Create new file for writing

```
<?php
$fh = fopen("myfile.txt", "w");

if($fh==false)
    die("unable to create file");
?>
```

Creating and Calling a PHP Function

```
<html>
<body>

<?php
//we create a function name my_function
function my_function()
{
    echo "Hello! How are you?";
}

//we call our function like this when we want to use it
my_function();

?>
```

```
</body>
</html>
```

PHP Functions with Parameters

```
<html>
<body>

<?php
//we create a function named my_function
function my_function($first_name, $last_name, $message)
{
    echo "$first_name $last_name once said " . $message;
}

//we call our function like this when we want to use it
my_function("Duke", "Nukem", '"It\'s time to kick some ass and chew
bubble gum."');

?>

</body>
</html>
```

Creating an Array in PHP

```
<?php
$employee_names[0] = "Dana";
$employee_names[1] = "Matt";
$employee_names[2] = "Susan";

echo "The first employee's name is ".$employee_names[0];
echo "<br>";
echo "The second employee's name is ".$employee_names[1];
echo "<br>";
echo "The third employee's name is ".$employee_names[2];
?>
```

Concatenating PHP Strings

```
<?php
$str1 = "I Love PHP.";
$str2 = "PHP is fun to learn.";
echo $str1." ".$str2;
?>
```

4

ΠΡΩΤΟΚΟΛΛΟ JSON

Το JSON (JavaScript Object Notation) είναι ένα ελαφρύ πρότυπο ανταλλαγής δεδομένων. Είναι εύκολο για τους ανθρώπους να το διαβάσουν και να το γράψουν. Είναι εύκολο για τις μηχανές να το αναλύσουν (parse) και να το παράγουν (generate). Είναι βασισμένο πάνω σε ένα υποσύνολο της γλώσσας προγραμματισμού JavaScript, Standard ECMA-262 Έκδοση 3η - Δεκέμβριος 1999. Το JSON είναι ένα πρότυπο κειμένου το οποίο είναι τελείως ανεξάρτητο από γλώσσες προγραμματισμού αλλά χρησιμοποιεί πρακτικές (conventions) οι οποίες είναι γνωστές στους προγραμματιστές της οικογένειας προγραμματισμού C, συμπεριλαμβανομένων των C, C++, C#, Java, JavaScript, Perl, Python, και πολλών άλλων. Αυτές οι ιδιότητες κάνουν το JSON μια ιδανική γλώσσα προγραμματισμού ανταλλαγής δεδομένων.

Η JSON προσφέρεται ως εναλλακτική λύση στην XML. Φυσικά να αντικαταστήσει εξ ολοκλήρου την XML δεν μπορεί διότι δεν υποστηρίζει schema validation, δεν μπορεί από μόνη της να ενημερώσει σχετικά με την κωδικοποίησή της και δεν έχει την έννοια των attributes, αλλά όπου αυτά τα ελαττώματα μπορούν να αγνοηθούν η αντικατάσταση θα είναι πολύ εύκολη. Όπως και η XML, η JSON έχει self-documentated format που περιγράφει την δομή των δεδομένων και δεν ασχολείται με την παρουσίασή τους.

Το JSON είναι χτισμένο σε δύο δομές:

- Μια συλλογή από ζευγάρια ονομάτων/τιμών. Σε διάφορες γλώσσες προγραμματισμού, αυτό αντιλαμβάνεται ως ένα object, καταχώρηση, δομή, λεξικό, πίνακα hash (hash table), λίστα κλειδιών, ή associative πίνακα.
- Μία ταξινομημένη λίστα τιμών. Στις περισσότερες γλώσσες προγραμματισμού, αυτό αντιλαμβάνεται ως ένας πίνακας (array), διάνυσμα, λίστα, ή ακολουθία.

Αυτά είναι τα universal data structures. Ουσιαστικά όλες οι μοντέρνες γλώσσες προγραμματισμού τα υποστηρίζουν με τον έναν ή τον άλλον τρόπο.

Ο λόγος που η JSON είναι απαραίτητη παρά την ύπαρξη της XML είναι ότι η JSON είναι συμπαγής. Στα πλαίσια του web έχει άλλο ένα πλεονέκτημα: έχει 100% valid JavaScript κώδικα και είναι πολύ εύκολο να μετατραπεί από κείμενο σε δεδομένα JavaScript, και οι εργασίες με τις δομές είναι πιο εύκολες και άνετες από την αδέξια, αν και καθολική DOM (Document Object Mode).

Τα μειονεκτήματα της JSON είναι τα εξής:

- Δεν υποστηρίζει schema validation σε αντίθεση με την XML
- Δεν μπορεί από μόνη της να ενημερώσει σχετικά με την κωδικοποίησή της
- Δεν έχει την έννοια των attributes

Ένα XML schema προσδιορίζει τη δομή των στοιχείων και των χαρακτηριστικών σε ένα έγγραφο XML. Για ένα έγγραφο XML προκειμένου να είναι έγκυρο βασισμένο σε ένα XML schema, πρέπει να επικυρωθεί (validation) σε σχέση με το XML schema.

4.1 Παράδειγμα JSON

Οι βασικοί τύποι JSON είναι:

- Number (δεν προσδιορίζεται ο τύπος των αριθμών αλλά στην πράξη είναι διπλής ακρίβειας σε δεκαδική μορφή)
- String (με διπλά εισαγωγικά, κλείνει με ανάποδη κάθετο)
- Boolean (`true` ή `false`)
- Array (μία διατεταγμένη σειρά τιμών, διαχωρισμένες με κόμμα και περικλειόμενες από αγκύλες- οι τιμές δε χρειάζεται να είναι της ίδιας μορφής)
- Object (μία μη διατεταγμένη συλλογή από ζεύγη κλειδιά-τιμές, διαχωρισμένα με κόμμα και περικλειόμενα από άγκιστρα- το κλειδί πρέπει να είναι συμβολοσειρά)
- null (κενό)

Το ακόλουθο παράδειγμα δείχνει την αναπαράσταση JSON ενός αντικειμένου που περιγράφει έναν άνθρωπο. Το αντικείμενο έχει πεδία συμβολοσειρών για το όνομα, το επώνυμο, ένα πεδίο αριθμού για την ηλικία, περιέχει ένα αντικείμενο αντιπροσωπεύοντας τη διεύθυνση του ατόμου, και περιέχει μία λίστα (πίνακας) αντικειμένων τηλεφωνικών αριθμών.

```
{
     "firstName": "John",
     "lastName": "Smith",
     "age": 25,
     "address":
     {
         "streetAddress": "21 2nd Street",
         "city": "New York",
         "state": "NY",
         "postalCode": "10021"
     },
     "phoneNumber":
     [
         {
           "type": "home",
           "number": "212 555-1234"
         },
         {
           "type": "fax",
           "number": "646 555-4567"
         }
     ]
 }
```

Το ακόλουθο παράδειγμα δείχνει την αναπαράσταση JSON ενός αντικειμένου που περιγράφει έναν προϊόν.

```
{
        "name":"Product",
        "properties":
        {
                "id":
                {
                        "type":"number",
                        "description":"Product identifier",
                        "required":true
                },
                "name":
                {
                        "description":"Name of the product",
                        "type":"string",
                        "required":true
                },
                "price":
                {
                        "type":"number",
                        "minimum":0,
                        "required":true
                },
                "tags":
                {
                        "type":"array",
                        "items":
                        {
                                "type":"string"
                        }
                }
        }
```

4.2 Παραλληλισμός της JSON με τα struct της C++

Ακολουθεί ένα παράδειγμα της δομής struct στη γλώσσα προγραμματισμού C++.

```
struct Product {
    char mfg_id[4];    // 4 char code for the manufacturer.
    char prod_id[8];   // 8-char code for the product
    int  price;        // price of the product in dollars.
    int  qty_on_hand;  // quantity on hand in inventory
};
```

Τα structs δεν είναι τίποτα άλλο από ένα σύνθετο τύπο δεδομένων. Θα ήταν χρήσιμο όταν έχουμε να αντιμετωπίσουμε ένα struct, να το σκεφτόμαστε σαν κουτιά το ένα μέσα στο άλλο. Γραφικά, είναι η αποθήκευση όλων των μεταβλητών που περιέχονται σε κάποιο ξεχωριστό χώρο ώστε να μπορούν να είναι προσβάσιμες οπουδήποτε. Η διαφορά με τον "κλασικό" τρόπο ορισμού των μεταβλητών είναι ότι με πολλές μεταβλητές που κάνουν την ίδια δουλειά χρησιμοποιώντας structs μπορούμε να εξοικονομήσουμε χρόνο-χώρο και να κάνουμε πιο κατανοητό και ευκολοδιάβαστο το πρόγραμμά μας.

Όπως ήδη ειπώθηκε, η δομή JSON βασίζεται σε μια συλλογή από ζευγάρια ονομάτων/τιμών, αλλά και μία ταξινομημένη λίστα τιμών. Παρατηρούμε στο παραπάνω παράδειγμα ότι η δομή των structs στη C++ είναι πολύ παρόμοια με τη δομή JSON και έχουν αντίστοιχες λειτουργίες.

Παρακάτω ακολουθεί ένα παράδειγμα με δομή JSON και με δομή struct στην C++ αντίστοιχα. Ο ορισμός των τιμών των μελών της δομής struct πραγματοποιείται μέσα στη συνάρτηση main().

Δομή JSON	**Δομή Struct C++**

`{`
`"firstName": "John",`
`"lastName": "Smith",`
`"age": 25,`
`"address":`
`{`
`"streetAddress": "21 2nd Street",`
`"city": "New York",`
`"state": "NY",`
`"postalCode": "10021"`
`},`
`"phoneNumber":`
`[`
`{`
`"type": "home",`
`"number": "212 555-1234"`
`},`
`{`
`"type": "fax",`
`"number": "646 555-4567"`
`}`
`]`
`}`
	`struct Person {`
	`char firstname[8];`
	`char lastname[8];`
	`int age;`
	`struct address{`
	`char streetAddress[8];`
	`char city[8];`
	`char state[8];`
	`int postalCode[8];`
	`};`
	`struct phoneNumber{`
	`char type[8];`
	`int number;`
	`char type[8];`
	`int number;`
	`};`
	`};`

Πίνακας 4.1: Σύγκριση παραδείγματος δομής JSON με δομή struct C++

4.3 Συναρτήσεις της Βιβλιοθήκης της PHP που υποστηρίζουν το Πρωτόκολλο JSON

Οι συναρτήσεις της βιβλιοθήκης της PHP που υποστηρίζουν το πρωτόκολλο JSON είναι η json_encode και η json_decode. Χρησιμοποιούνται για την κωδικοποίηση και αποκωδικοποίηση αντίστοιχα του κώδικα JSON σε PHP.

Για το κατέβασμα των δεδομένων από το διαδίκτυο χρησιμοποιείται κωδικοποίηση JSON (JavaScript Object Notation) που αποτελεί πρότυπο ανταλλαγής δεδομένων.

Για τη επίτευξη της σύνδεσης χρησιμοποιείται η βιβλιοθήκη touchJSON που διανέμεται δωρεάν στο διαδίκτυο. Η touchJSON είναι γραμμένη εξ ολοκλήρου σε Objective-C και δημιουργήθηκε με σκοπό την:

- κωδικοποίηση δεδομένων σε μορφή JSON (**json_encode**)

- αποκωδικοποίηση δεδομένων από μορφή JSON σε μορφή συμβατή με το NSDictonary που χρησιμοποιείται για αποθήκευση δεδομένων και παρέχεται στο SDK του iOS 4 (**json_decode**)

Για την επίτευξη της σύνδεσης πρέπει πρώτα να γίνει εισαγωγή της κατάλληλης κλάσης της βιβλιοθήκης touchJSON στην κλάση του προγράμματος που πρόκειται να χρησιμοποιηθεί. Εφόσον πραγματοποιείται το κατέβασμα δεδομένων από το διαδίκτυο, χρησιμοποιείται κλάση αποκωδικοποίησης.

Ο κώδικας που χρησιμοποιείται πρέπει να είναι έτσι διαμορφωμένος ώστε τα δεδομένα που δέχεται η εφαρμογή από τα PHP αρχεία, που επικοινωνούν με τη βάση δεδομένων, να είναι αξιοποιήσιμα από το πρόγραμμα.

Για την εκμετάλλευση της παραπάνω διαδικασίας επιβάλλεται η παράγωγη δεδομένων σε μορφή JSON από τα PHP αρχεία. Για την επίτευξη αυτού του στόχου τα δεδομένα που στέλνονται στην εφαρμογή αποθηκεύονται στα αρχεία PHP, σε έναν πίνακα και κωδικοποιούνται σε μορφή JSON.

Δημιουργώντας αυτές τις συνθήκες επιτυγχάνεται η αποστολή δεδομένων από τη βάση δεδομένων που βρίσκεται στο διαδίκτυο στην εφαρμογή που εδρεύει στο iPhone για παράδειγμα.

5

GOOGLE APIS

Το ακρωνύμιο API προέρχεται από τα αρχικά των λέξεων Application Programming Interface.

Το Maps API Web Services αποτελεί μία συλλογή από HTTP διεπαφές με τις υπηρεσίες της Google προσφέροντας γεωγραφικές πληροφορίες χρήσιμες για την εφαρμογή τους σε χάρτες. Οι υπηρεσίες αυτές είναι οι εξής:

- Directions API
- Distance Matrix API
- Elevation API
- Geocoding API
- Places API

Οι υπηρεσίες που χρησιμοποιήθηκαν στη διπλωματική εργασία και οι οποίες αναλύονται παρακάτω είναι οι εξής:

- Directions API
- Elevation API

5.1 Υπηρεσία Δικτύου- Web Service

Το Google Maps API προσφέρει διαδικτυακές υπηρεσίες (web services) ως μία διεπαφή για την αναζήτηση Maps API πληροφοριών από εξωτερικές υπηρεσίες και την χρήση αυτών σε διάφορες εφαρμογές. Αυτές οι υπηρεσίες είναι σχεδιασμένες να χρησιμοποιούνται σε συνδυασμό με ένα χάρτη.

Αυτές οι διαδικτυακές υπηρεσίες χρησιμοποιούν HTTP αιτήσεις σε συγκεκριμένες διευθύνσεις URLs, εισάγοντας URL παραμέτρους στις υπηρεσίες αυτές. Γενικά, οι υπηρεσίες αυτές επιστρέφουν δεδομένα είτε ως JSON είτε ως XML για ανάλυση ή/και επεξεργασία από την εφαρμογή.

Ένα σύνηθες Web Service request ακολουθεί γενικά την παρακάτω μορφή:
`http://maps.googleapis.com/maps/api/`*`service`*`/`*`output`*`?`*`parameters`*

όπου η λέξη `service` δείχνει τη συγκεκριμένη υπηρεσία που ζητήθηκε και η λέξη `output` δείχνει τη μορφή της απάντησης (συνήθως `json` ή `xml`).

5.2 Google Directions API

Το Google Directions API αποτελεί μία υπηρεσία η οποία υπολογίζει διαδρομές μεταξύ τοποθεσιών χρησιμοποιώντας μία κλήση HTTP. Οι διαδρομές μπορεί να καθορίζουν προελεύσεις, προορισμούς, και ενδιάμεσα σημεία είτε ως συμβολοσειρές κειμένου (text strings) – π.χ. "Chicago, IL" ή "Darwin, NT, Australia" – ή ως γεωγραφικές συντεταγμένες. Το Directions API μπορεί να επιστρέψει πολυτμηματικές διαδρομές χρησιμοποιώντας μία σειρά ενδιάμεσων σημείων.

Αυτή η υπηρεσία έχει σχεδιαστεί γενικά για τον υπολογισμό κατευθύνσεων – διαδρομών για γνωστές εκ των προτέρων διευθύνσεις για την τοποθέτηση του περιεχομένου της εφαρμογής στο χάρτη. Αυτή η υπηρεσία δε σχεδιάστηκε για να ανταποκρίνεται σε πραγματικό χρόνο στα δεδομένα εισόδου του χρήστη, για παράδειγμα.

Η υπηρεσία αυτή παρέχει τη δυνατότητα υπολογισμού διευθύνσεων σε πολύ λίγο χρόνο και αποθηκεύει τα αποτελέσματα σε μία προσωρινή κρυφή μνήμη της επιλογής του χρήστη.

5.2.1 Περιορισμοί Χρήσης

Η χρήση του Google Directions API υπόκειται στον περιορισμό των 2500 αιτήσεων διαδρομών ανά ημέρα. Οι διαδρομές μεμονωμένα μπορεί να περιέχουν μέχρι και 8 ενδιάμεσα σημεία στην κλήση.

Επίσης, οι διευθύνσεις Directions API URLs περιορίζονται στους 2048 χαρακτήρες, πριν την κωδικοποίηση URL.

5.2.2 Directions Requests

Μία Directions API κλήση παίρνει την παρακάτω μορφή:

`http://maps.googleapis.com/maps/api/directions/`*`output`*`?`*`parameters`*

όπου το `output` μπορεί να πάρει κάποια από τις ακόλουθες τιμές:

- `json` (προτείνεται) παρουσιάζει το αποτέλεσμα σε JavaScript Object Notation (JSON)
- `xml` παρουσιάζει το αποτέλεσμα σε XML

Για να αποκτήσει ο χρήστης πρόσβαση στο Directions API μέσω HTTPS, πρέπει να χρησιμοποιήσει:

`https``://maps.googleapis.com/maps/api/directions/`*`output`*`?`*`parameters`*

Το HTTPS συνιστάται για εφαρμογές που περιέχουν ευαίσθητες πληροφορίες χρήστη, όπως την τοποθεσία του χρήστη, στις κλήσεις.

5.2.3 Παράμετροι αιτήματος - Request Parameters

Απαιτούνται συγκεκριμένες παράμετροι ενώ κάποιες άλλες παράμετροι είναι προαιρετικές. Όλες οι παράμετροι στις διευθύνσεις URLs χωρίζονται από το σύμβολο `&`. Η λίστα των παραμέτρων και των πιθανών τιμών τους απαριθμούνται παρακάτω.

Το Directions API καθορίζει μία κλήση διαδρομών (directions request) χρησιμοποιώντας τις ακόλουθες URL παραμέτρους:

- `origin` (*απαιτείται*) — η διεύθυνση ή η τιμή γεωγραφικών συντεταγμένων από τις οποίες ο χρήστης θέλει να υπολογίσει τις διαδρομές.
- `destination` (*απαιτείται*) — η διεύθυνση ή η τιμή γεωγραφικών συντεταγμένων από τις οποίες ο χρήστης θέλει να υπολογίσει τις διαδρομές.
- `mode` (*προαιρετικό*, από προεπιλογή οδηγώντας `driving`) — ορίζει τον τρόπο μεταφοράς που επιθυμεί ο χρήστης όταν θα υπολογιστούν οι διαδρομές.
- `waypoints` *(προαιρετικό*) – ορίζει ένα πίνακα ενδιάμεσων σημείων. Τα ενδιάμεσα σημεία μεταβάλλουν μία διαδρομή με τη δρομολόγησή της μέσω συγκεκριμένης(-ων) περιοχής(-ων). Ένα τέτοιο σημείο ορίζεται είτε με τις γεωγραφικές του συντεταγμένες είτε ως μία κωδικοποιημένη διεύθυνση.
- `alternatives` *(προαιρετικό*), αν έχει οριστεί ως `true`, καθορίζει ότι η Υπηρεσία Διευθύνσεων (Directions Service) μπορεί να παρέχει περισσότερες από μία εναλλακτική διαδρομή. Εδώ πρέπει να σημειωθεί ότι η παροχή εναλλακτικών διαδρομών μπορεί να αυξήσει το χρόνο απάντησης από τον διακομιστή.
- `avoid` (*προαιρετικό*) δείχνει ότι οι υπολογισμένες διαδρομές πρέπει να αποφεύγουν τα προτεινόμενα χαρακτηριστικά. Επί του παρόντος, αυτή η παράμετρος υποστηρίζει τα παρακάτω δύο επιχειρήματα:
  - `tolls`, δείχνει ότι η υπολογισμένη διαδρομή πρέπει να αποφύγει δρόμους/γέφυρες με σταθμούς διοδίων.
  - `Highways`, δείχνει ότι η υπολογισμένη διαδρομή πρέπει να αποφεύγει δρόμους ταχείας κυκλοφορίας.
- `units` (*προαιρετικό*) — καθορίζει ποιο σύστημα μονάδων να χρησιμοποιηθεί όταν εμφανίζονται τα αποτελέσματα.
- `region` (*προαιρετικό*) — ο κώδικας περιοχής καθορίζεται ως μία ccTLD ("Top-Level Domain", δηλαδή ανώτατου επιπέδου περιοχή) τιμή δύο χαρακτήρων.
- `language` (*προαιρετικό*) — η γλώσσα στην οποία επιστρέφονται τα αποτελέσματα. Αν η γλώσσα δεν διατίθεται στο κατάλογο, η υπηρεσία προσπαθεί να χρησιμοποιήσει τη μητρική γλώσσα του προγράμματος περιήγησης όπου αυτό είναι δυνατό.
- `sensor` (*απαιτείται*) — δείχνει αν η αίτηση διευθύνσεων προέρχεται από μία συσκευή με αισθητήρα θέσης. Αυτή η τιμή πρέπει να είναι είτε `true` είτε `false`.

5.2.4 Επιλογές Ταξιδιού

Όταν υπολογίζονται διευθύνσεις, ο χρήστης μπορεί να επιλέξει ποιο τρόπο μεταφοράς επιθυμεί να χρησιμοποιήσει. Από προεπιλογή, οι διευθύνσεις υπολογίζονται ως διευθύνσεις με οδήγηση. Υποστηρίζονται οι ακόλουθες μορφές πλοήγησης:

- `driving` (προεπιλογή) δείχνει τις βασικές διευθύνσεις οδήγησης χρησιμοποιώντας το οδικό δίκτυο.
- `walking` ζητάει διευθύνσεις για πεζούς μέσω πεζοδρομίων
- `bicycling` ζητάει διευθύνσεις για ποδηλάτες μέσω ποδηλατοδρόμων.

5.2.5 Χρήση ενδιάμεσων σημείων

Όταν υπολογίζονται οι διαδρομές με τη βοήθεια του Directions API, ο χρήστης μπορεί να προσδιορίσει και τα ενδιάμεσα σημεία μίας διαδρομής. Τα σημεία αυτά επιτρέπουν στο χρήστη να υπολογίζουν διαδρομές μέσω συγκεκριμένων περιοχών, και η διαδρομή που επιστρέφεται ως αποτέλεσμα περνάει από τα δοσμένα ενδιάμεσα σημεία.

Τα ενδιάμεσα σημεία προσδιορίζονται μέσα στην παράμετρο `waypoints` και αποτελούνται από μία ή περισσότερες διευθύνσεις ή τοποθεσίες και χωρίζονται από το σύμβολο (`|`).

Από προεπιλογή, η υπηρεσία Διαδρομών υπολογίζει μία διαδρομή μέσω των συνιστώμενων ενδιάμεσων σημείων με τη δοσμένη σειρά τους.

5.2.6 Περιορισμοί

Οι περιορισμοί κατά τον υπολογισμό μίας διαδρομής εκφράζονται με τη χρήση της παραμέτρου `avoid`, και του επιχειρήματος δείχνοντας τον περιορισμό. Προς το παρόν, υποστηρίζονται δύο περιορισμοί όπως αναφέρθηκε και παραπάνω:

- `avoid=tolls`
- `avoid=highways`

5.2.7 Σύστημα Μονάδων

Τα αποτελέσματα των διαδρομών περιέχουν κείμενο μέσα στα πεδία των αποστάσεων, τα οποία μπορεί να εμφανιστούν στο χρήστη για να δείξουν την απόσταση σε ένα συγκεκριμένο τμήμα μιας διαδρομής. Από προεπιλογή, αυτό το κείμενο χρησιμοποιεί το σύστημα μονάδων της χώρας ή περιοχής προέλευσης.
Ο χρήστης μπορεί να παρακάμψει αυτό το σύστημα μονάδων και να ορίσει ο ίδιος ένα εντός της παραμέτρου `units`, εισάγοντας μία από τις ακόλουθες τιμές:

- `metric`. Οι αποστάσεις που επιστρέφονται είναι σε χιλιόμετρα και μέτρα.
- `Imperial`. Οι αποστάσεις που επιστρέφονται είναι σε μίλια και πόδια.

5.2.8 Περιοχή Πόλωσης

Ο χρήστης μπορεί να ρυθμίσει την Υπηρεσία Διαδρομών να επιστρέφει αποτελέσματα σύμφωνα με μία συγκεκριμένη περιοχή, με τη χρήση της παραμέτρου

`region`. Η παράμετρος αυτή δέχεται μία ccTLD (country code Top-Level Domain) μεταβλητή καθορίζοντας τη χώρα. Οι περισσότεροι ccTLD κώδικες είναι όμοιοι με τους κώδικες ISO 3166-1, εκτός από μερικές εξαιρέσεις. Για παράδειγμα, ο ccTLD κώδικας του Ηνωμένου Βασιλείου είναι "uk" ενώ ο ISO 3166-1 κώδικάς του είναι "gb".

5.2.9 Directions Responses

Τα αποτελέσματα των διαδρομών επιστρέφονται στη μορφή που υποδεικνύεται στη διεύθυνση URL, και μπορεί να είναι είτε σε μορφή JSON είτε σε μορφή XML.

- JSON Output

Ένα παράδειγμα κλήσης HTTP είναι το παρακάτω:

```
http://maps.googleapis.com/maps/api/directions/json?origin=Chicago,IL
&destination=Los+Angeles,CA&waypoints=Joplin,MO|Oklahoma+City,OK&sens
or=false
```

Στο παράδειγμα αυτό υπολογίζεται η διαδρομή από το IL, Σικάγο στο CA , Λος Άντζελες μέσω δύο ενδιάμεσων σημείων και επιστρέφεται η απάντηση σε μορφή JSON.

Εδώ θα πρέπει να σημειωθεί ότι τα αποτελέσματα που προκύπτουν πρέπει να αναλύονται (*parse*) εάν ο χρήστης επιθυμεί να εξάγει τις τιμές που προέκυψαν. Συνήθως χρησιμοποιείται το πρωτόκολλο JSON διότι η διαδικασία εξαγωγής των αποτελεσμάτων καθώς επίσης και η αποκωδικοποίηση σε PHP είναι σχετικά πιο εύκολη.

- XML Output

Στη συνέχεια ακολουθεί ένα πανομοιότυπο με το παραπάνω παράδειγμα, με τη διαφορά ότι επιστρέφει μία `xml` απάντηση και όχι `json`.

```
http://maps.googleapis.com/maps/api/directions/xml?origin=Chicago,IL&
destination=Los+Angeles,CA&waypoints=Joplin,MO|Oklahoma+City,OK&senso
r=false
```

Είναι προτιμότερη χρήση `json` για την εξαγωγή των διαδρομών εκτός εάν η `xml` απάντηση προορίζεται για συγκεκριμένο σκοπό. Η επεξεργασία ενός XML αρχείου χρειάζεται περισσότερη προσοχή και είναι πιο πολύπλοκη, ιδιαίτερα όσον αφορά στη σωστή αναφορά στους κόμβους και τα στοιχεία.

Οι διαφορές στη χρήση XML και JSON είναι οι εξής:

- Τα XML αποτελέσματα περιλαμβάνονται σε ένα `<DirectionsResponse>` στοιχείο.
- Το αποτέλεσμα JSON δηλώνει καταχωρήσεις με πολλαπλά στοιχεία μέσω πολυδιάστατων πινάκων (`steps`), ενώ το αντίστοιχο XML δηλώνει τις καταχωρήσεις χρησιμοποιώντας πολλαπλά μοναδικά στοιχεία (`<step>`).
- Τα κενά στοιχεία παρουσιάζονται μέσω άδειων πινάκων σε JSON, ενώ χαρακτηρίζονται από την απουσία οποιουδήποτε στοιχείου σε XML. Μία απάντηση που δεν παράγει κανένα αποτέλεσμα θα επιστρέψει έναν άδειο

πίνακα διαδρομών σε JSON, αλλά κανένα στοιχείο `<route>` σε XML, για παράδειγμα.

5.2.10 Στοιχεία του Directions Response

Οι απαντήσεις των διαδρομών περιέχουν δύο στοιχεία αφετηρίας:

- `"status"` – περιέχει μεταδεδομένα στην αίτηση.
- `"routes"` – περιέχει έναν πίνακα διαδρομών από το σημείο εκκίνησης στο σημείο προορισμού.

Οι διαδρομές αποτελούνται από ένθετα τμήματα, τα Legs και τα Steps.

5.2.11 Κωδικοί κατάστασης - Status Codes

Το πεδίο `"status"` στην Απάντηση Διαδρομών περιέχει την κατάσταση της κλήσης και μπορεί να περιέχει πληροφορίες εντοπισμού σφαλμάτων που βοηθούν το χρήστη να παρακολουθήσει το λόγο που απέτυχε η υπηρεσία Διαδρομών. Το πεδίο `"status"` μπορεί να περιέχει τις ακόλουθες τιμές:

- `OK` δείχνει ότι η απάντηση φέρει ένα έγκυρο αποτέλεσμα.
- `NOT_FOUND` δείχνει ότι τουλάχιστον μία από τις περιοχές που ορίστηκαν ως αρχή, προορισμός και ενδιάμεσα σημεία δεν μπόρεσαν να αποκωδικοποιηθούν.
- `ZERO_RESULTS` δείχνει ότι δε βρέθηκε διαδρομή.
- `MAX_WAYPOINTS_EXCEEDED` δείχνει ότι εισήχθησαν υπερβολικά πολλά ενδιάμεσα σημεία στο αίτημα. Ο μέγιστος αριθμός ενδιάμεσων σημείων είναι 8, εκτός από τα σημεία αρχής και τέλους.
- `INVALID_REQUEST` δείχνει ότι το αίτημα δεν ήταν έγκυρο.
- `OVER_QUERY_LIMIT` δείχνει ότι η υπηρεσία έχει λάβει πολλά αιτήματα από την εφαρμογή ενός συγκεκριμένου χρήστη μέσα σε πολύ μικρό χρονικό διάστημα.
- `REQUEST_DENIED` δείχνει ότι η υπηρεσία αρνήθηκε τη χρήση της υπηρεσίας διαδρομών για κάποιον χρήστη.
- `UNKNOWN_ERROR` δείχνει ότι η κλήση της υπηρεσίας δε μπόρεσε να πραγματοποιηθεί εξαιτίας κάποιου σφάλματος του διακομιστή.

5.2.12 Πεδίο «routes»

Όταν η υπηρεσία Directions API επιστρέφει αποτελέσματα, τα τοποθετεί σε έναν JSON `routes` πίνακα.

Κάθε στοιχείο του πίνακα περιέχει ένα μοναδικό αποτέλεσμα από τα ορισμένα σημεία προέλευσης και προορισμού. Αυτή η διαδρομή μπορεί να αποτελείται από ένα ή περισσότερα τμήματα `legs` αν ορίστηκαν ή όχι ενδιάμεσα σημεία.

Κάθε διαδρομή μέσα στο πεδίο `routes` μπορεί να περιλαμβάνει τα ακόλουθα πεδία:

- `summary` περιέχει μία μικρή λεκτική περιγραφή της διαδρομής.

- `legs[]` περιέχει έναν πίνακα με πληροφορίες για ένα τμήμα leg της διαδρομής, μεταξύ δύο περιοχών εντός της δοσμένης διαδρομής.
- `waypoint_order` περιέχει έναν πίνακα που δείχνει την σειρά των ενδιάμεσων σημείων της διαδρομής.
- `overview_polyline` περιέχει ένα αντικείμενο με έναν πίνακα κωδικοποιημένων σημείων και επιπέδων που αντιπροσωπεύουν μία προσεγγιστική διαδρομή.
- `bounds` περιέχει ένα πλαίσιο οριοθέτησης της διαδρομής.
- `copyrights` περιέχει τα πνευματικά δικαιώματα που θα εμφανίζονται για τη διαδρομή.
- `warnings[]` περιέχει έναν πίνακα προειδοποιήσεων που θα εμφανίζονται όταν θα απεικονίζονται οι διαδρομές.

5.2.13 Πεδίο «legs»

Κάθε στοιχείο στον πίνακα `legs` καθορίζει ένα μοναδικό τμήμα leg της διαδρομής. Για τις διαδρομές που δεν περιέχουν ενδιάμεσα σημεία, θα αποτελούνται από ένα μόνο τμήμα "leg" .

Κάθε τμήμα leg εντός του πεδίου `legs` μπορεί να περιλαμβάνει τα ακόλουθα πεδία:

- `steps[]` περιέχει ένα πίνακα με τμήματα steps δηλώνοντας πληροφορίες για το κάθε τμήμα step του τμήματος leg της διαδρομής.
- `distance` δείχνει τη συνολική απόσταση που διανύεται σε αυτό το τμήμα leg.
- `duration` δείχνει τη συνολική διάρκεια του τμήματος leg.
- `start_location` περιέχει τις γεωγραφικές συντεταγμένες του σημείου προελεύσεως αυτού του τμήματος leg.
- `end_location` περιέχει τις γεωγραφικές συντεταγμένες του σημείου προορισμού αυτού του τμήματος leg.
- `start_address` περιέχει την αναγνώσιμη διεύθυνση που αντιστοιχεί στο `start_location` του ποδιού αυτού.
- `end_addresss` περιέχει την αναγνώσιμη διεύθυνση που αντιστοιχεί στο `end_location` του ποδιού αυτού.

5.2.14 Πεδίο «steps»

Κάθε στοιχείο στα τμήματα `steps` καθορίζει ένα μοναδικό τμήμα στις υπολογισμένες διευθύνσεις. Το step είναι η πιο ατομική μονάδα μίας διαδρομής, που περιέχει steps που περιγράφουν μία συγκεκριμένη οδηγία ενός ταξιδιού, καθώς επίσης και άλλες πληροφορίες που περιγράφουν το step.

Κάθε τμήμα step εντός του πεδίου `steps` μπορεί να περιέχει τα ακόλουθα πεδία:

- `html_instructions` περιέχει μορφοποιημένες οδηγίες για το step αυτό.
- `distance` περιέχει την απόσταση που καλύπτεται σε αυτό το step μέχρι το επόμενο step.
- `duration` περιέχει τον τυπικό χρόνο που χρειάζεται για να πραγματοποιηθεί το step, μέχρι το επόμενο step.

- `start_location` περιέχει την θέση του σημείου ενάρξεως του step, ως γεωγραφικές συντεταγμένες.
- `end_location` περιέχει την θέση του σημείου τέλους του step, ως γεωγραφικές συντεταγμένες.

5.2.15 Παράδειγμα κλήσης της υπηρεσίας Directions API

Ακολουθεί ένα παράδειγμα κλήσης του Directions API για τον υπολογισμό της διαδρομής από το Τολέδο της Ισπανίας στη Μαδρίτη.

http://maps.googleapis.com/maps/api/directions/json?origin=Toledo&destination=Madrid&region=es&sensor=false

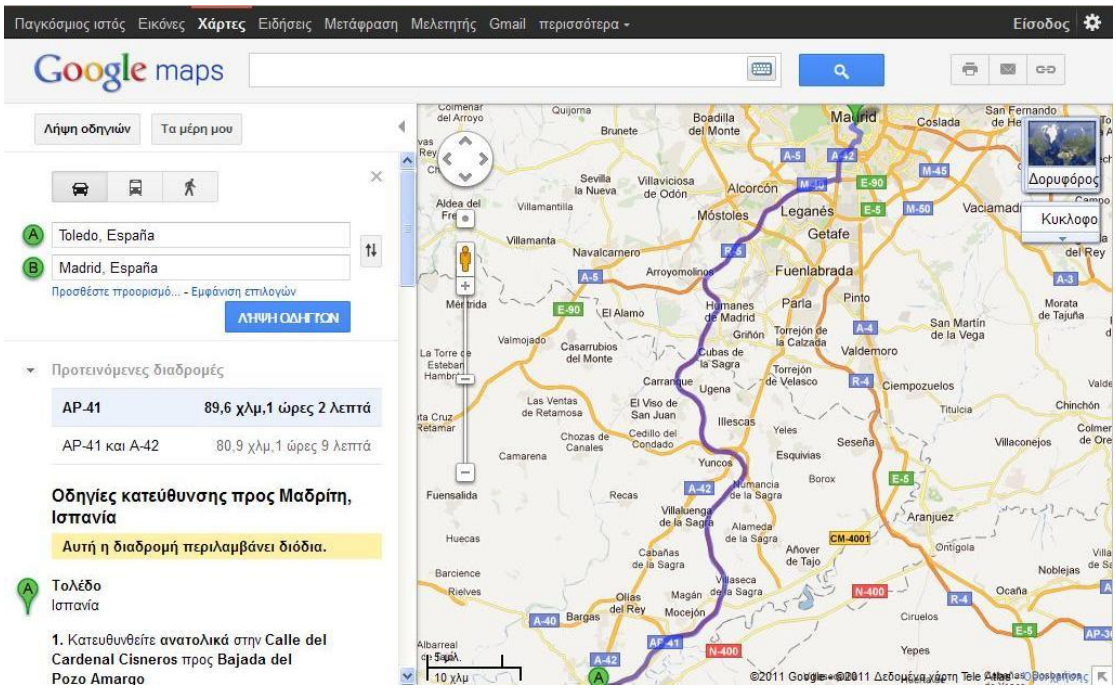
Παρακάτω παρατίθεται το αποτέλεσμα που προκύπτει από το παραπάνω αίτημα.

```
{
  "status": "OK",
  "routes": [ {
    "summary": "AP-41",
    "legs": [ {
        ...
    } ],
    "copyrights": "Map data ©2010 Europa Technologies, Tele Atlas",
    "warnings": [ ],
    "waypoint_order": [ ]
  } ]
}
```

Εδώ πρέπει να σημειωθεί ότι σε περίπτωση που δε διευκρινιστεί η παράμετρος `region` που να προσδιορίζει την πόλη Τολέδο, μπορεί να ερμηνευτεί ως η πόλη του Οχάιο και όχι της Ισπανίας. Έτσι η κλήση δε θα επιστρέψει αποτέλεσμα.

http://maps.googleapis.com/maps/api/directions/json?origin=Toledo&destination=Madrid&sensor=false

```
{
  "status": "ZERO_RESULTS",
  "routes": [ ]
}
```



Εικόνα 5.1: Υπολογισμός της διαδρομής από το Τολέδο της Ισπανίας στη Μαδρίτη στο Google Maps

5.3 Google Elevation API

Το Google Elevation API αποτελεί μία υπηρεσία η οποία προσφέρει μία απλή διασύνδεση με ερωτήματα θέσεως στη γη για υψομετρικές πληροφορίες. Επιπλέον, ένας χρήστης της υπηρεσίας αυτής μπορεί να ζητήσει δείγμα υψομετρικών πληροφοριών κατά μήκος διαδρομών, επιτρέποντάς του έτσι να υπολογίσει υψομετρικές αλλαγές κατά μήκος διαδρομών.

5.3.1 Τι μπορεί να κάνει το Elevation API

Η υπηρεσία Elevation API παρέχει υψομετρική πληροφορία για όλες τις περιοχές στην επιφάνεια της γης, περιλαμβάνοντας και τις περιοχές κάτω από την επιφάνεια της θάλασσας στον πυθμένα των ωκεανών επιστρέφοντας αρνητικές τιμές. Στις περιπτώσεις αυτές στις οποίες η Google δεν κατέχει ακριβείς υψομετρικές μετρήσεις στη συγκεκριμένη περιοχή που ζητάει ο χρήστης, η υπηρεσία θα εφαρμόσει παρεμβολή και θα επιστρέψει μία μέση τιμή χρησιμοποιώντας τις τέσσερις πλησιέστερες περιοχές.
Το Elevation API είναι μία νέα υπηρεσία. Η πρόσβαση στο Elevation API επιτυγχάνεται μέσω μίας HTTP διασύνδεσης.

5.3.2 Χρησιμότητα

Το Elevation API είναι χρήσιμο σε προγραμματιστές ιστοσελίδων και κινητών τηλεφώνων οι οποίοι θέλουν να χρησιμοποιήσουν υψομετρική πληροφορία σε χάρτες.

5.3.3 Περιορισμοί χρήσης

Η χρήση του Google Elevation API υπόκειται σε ένα όριο 2.500 αιτημάτων ανά ημέρα. Σε κάθε κλήση – αίτημα ο ενδιαφερόμενος μπορεί να ζητήσει το υψόμετρο μέχρι και 512 περιοχών, αλλά δε μπορεί να ξεπεράσει τις 25.000 περιοχές συνολικά ανά ημέρα. Αυτό το όριο επιβάλλεται ώστε να αποφευχθεί η κατάχρηση ή/και η αλλαγή του στόχου της υπηρεσίας. Το όριο αυτό μπορεί να αλλάξει χωρίς προειδοποίηση στο μέλλον. Εάν ο ενδιαφερόμενος ξεπεράσει το 24ωρο όριο ή καταχραστεί την υπηρεσία, το Elevation API ενδέχεται να σταματήσει την εργασία για τον χρήστη προσωρινά. Εάν αυτός συνεχίσει να υπερβαίνει το όριο, η πρόσβασή του στο Elevation API μπορεί να αποκλειστεί.
Οι διευθύνσεις URLs του Elevation API περιορίζονται στους 2048 χαρακτήρες, πριν την κωδικοποίηση του URL.

5.3.4 Αίτημα υψομέτρου - Elevation Request

Το Elevation API επιστρέφει υψομετρική πληροφορία για περιοχές πάνω στη γη.
Η αναγνώριση των κορυφών διαδρομών ή περιοχών πραγματοποιείται με τη χρήση των γεωγραφικών συντεταγμένων.
Μία URL διεύθυνση του Google Elevation API πρέπει να έχει την ακόλουθη μορφή:

`http://maps.googleapis.com/maps/api/elevation/`*`outputFormat`*`?`*`parameters`*

5.3.5 Μορφή Αποτελεσμάτων

Το Elevation API προς το παρόν υποστηρίζει τις ακόλουθες μορφές εξόδου:

- `/json` επιστρέφει αποτελέσματα σε JavaScript Object Notation (JSON).
- `/xml` επιστρέφει αποτελέσματα σε XML.

5.3.6 Χρήση παραμέτρων

Τα αιτήματα προς το Elevation API χρησιμοποιούν διαφορετικές παραμέτρους βασισμένες στο αν το αίτημα είναι για διακριτές περιοχές ή για ταξινομημένη διαδρομή. Για τις διακριτές περιοχές, τα αιτήματα για υψόμετρα επιστρέφουν πληροφορίες στις συγκεκριμένες περιοχές που εισήχθησαν στο αίτημα. Για τις διαδρομές, τα αιτήματα υψομέτρων λαμβάνονται δειγματοληπτικά κατά μήκος της δοσμένης διαδρομής.

Όπως συμβαίνει σε όλες τις URL διευθύνσεις, οι παράμετροι χωρίζονται με τη χρήση του συμβόλου (`&`) . Η λίστα των παραμέτρων και οι πιθανές τους τιμές δηλώνονται παρακάτω:

Αιτήματα Περιοχής (Positional Requests):

- `locations` (απαιτείται) καθορίζει την περιοχή(ές) στη γη από τις οποίες θα επιστραφούν οι υψομετρικές πληροφορίες. Αυτή η παράμετρος δέχεται είτε μία

τοποθεσία ως ένα ζευγάρι γεωγραφικές συντεταγμένες είτε πολλαπλά ζεύγη γεωγραφικών συντεταγμένων που έχουν εισαχθεί ως πίνακας ή ως κωδικοποιημένη πολλαπλή γραμμή (polyline).

ή

Δειγματοληπτικά Αιτήματα Διαδρομής (Sampled Path Requests):

- `path` (απαιτείται) καθορίζει μία διαδρομή επάνω στη γη για την οποία θα επιστραφούν υψομετρικές πληροφορίες. Αυτή η παράμετρος καθορίζει ένα σύνολο δύο ή περισσοτέρων διατεταγμένων ζευγαριών (γεωγραφικό μήκος, γεωγραφικό πλάτος) ορίζοντας μία διαδρομή κατά μήκος της επιφάνειας της γης.
- `samples` (απαιτείται) προσδιορίζει τον αριθμό των δειγματοληπτικών σημείων κατά μήκος μιας διαδρομής για τα οποία επιστρέφεται η υψομετρική πληροφορία.

Παράμετροι Αναφοράς (Reporting Parameters):

- `sensor` (απαιτείται) προσδιορίζει εάν τα δεδομένα της εφαρμογής χρησιμοποιούν αισθητήρα για τον καθορισμό της θέσης του χρήστη.

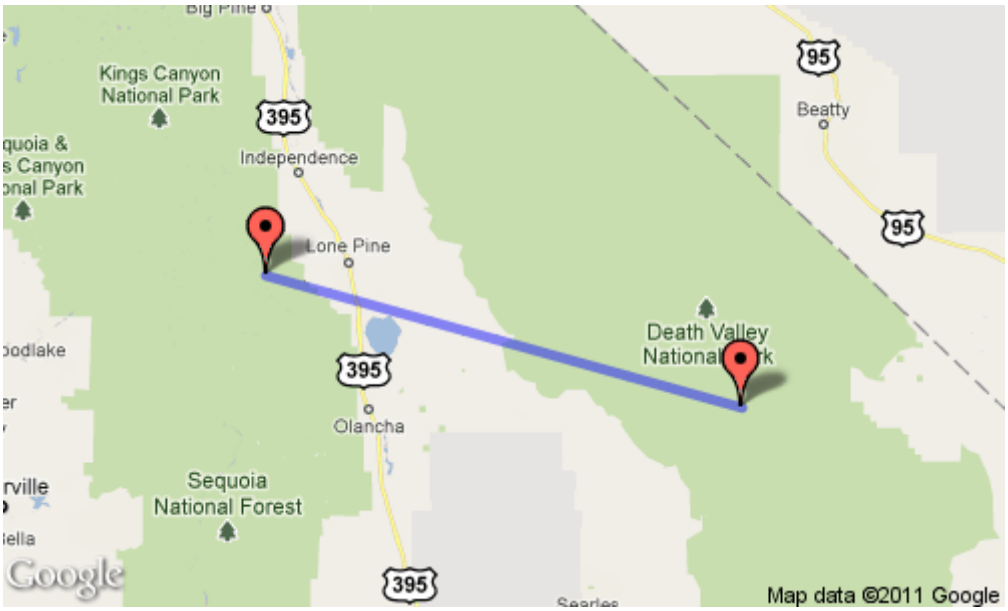
Οι εφαρμογές που καθορίζουν τη θέση του χρήστη μέσω αισθητήρα πρέπει να εισάγουν `sensor=true` στην κλήση της URL διεύθυνσης του Elevation API. Εάν η εφαρμογή δε χρησιμοποιεί αισθητήρα ο χρήστης εισάγει `sensor=false`.

5.3.7 Elevation Responses

Για κάθε έγκυρο αίτημα, η υπηρεσία Υψομέτρων (Elevation Service) θα επιστρέφει μία απάντηση στη μορφή που δίνεται στη διεύθυνση URL του αιτήματος. Κάθε απάντηση θα περιέχει τα ακόλουθα στοιχεία:

- έναν κώδικα `status`, το οποίο μπορεί να είναι ένα από τα παρακάτω:
  1. `OK` δείχνοντας ότι η κλήση API ήταν επιτυχής
  2. `INVALID_REQUEST` δείχνοντας ότι η κλήση API ήταν αλλοιωμένη
  3. `OVER_QUERY_LIMIT` δείχνοντας ότι το πρόγραμμα αιτημάτων έχει υπερβεί το ενδεδειγμένο ποσοστό.
  4. `REQUEST_DENIED` δείχνοντας ότι το API δεν ολοκλήρωσε τη διαδικασία κλήσης.
  5. `UNKNOWN_ERROR` υποδεικνύοντας άγνωστης μορφής σφάλμα.
- έναν πίνακα αποτελεσμάτων `results` που περιλαμβάνει τα παρακάτω στοιχεία:
  1. Ένα στοιχείο `location` (που περιέχει `lat` και `lng` στοιχεία) της θέσης για την οποία υπολογίζονται οι υψομετρικές πληροφορίες.
  2. Ένα στοιχείο `elevation` το οποίο δείχνει το υψόμετρο της περιοχής σε μέτρα.

Η ακόλουθη εικόνα δείχνει πως απεικονίζονται οι παραπάνω πληροφορίες σε χάρτη.



Εικόνα 5.2: Απεικόνιση υψομετρικής πληροφορίας σε χάρτη

5.3.8 Παραδείγματα κλήσης της υπηρεσίας Elevation API

Παράδειγμα υπολογισμού σημειακών υψομέτρων

Το ακόλουθο παράδειγμα ζητάει το υψόμετρο στο Ντένβερ του Κολοράντο σε μορφή JSON.

```
http://maps.googleapis.com/maps/api/elevation/json?locations=39.7391536,-104.9847034&sensor=true_or_false
```

Το αποτέλεσμα που επιστρέφεται σε μορφή json από την κλήση του API είναι το παρακάτω:

```
{
  "status": "OK",
  "results": [ {
    "location": {
      "lat": 39.7391536,
      "lng": -104.9847034
    },
    "elevation": 1608.8402100
  } ]
}
```

Το επόμενο παράδειγμα δείχνει πολλαπλές απαντήσεις (για το Ντένβερ στο Κολοράντο και την κοιλάδα του Death Valley στην Καλιφόρνια) σε μορφή JSON.

```
http://maps.googleapis.com/maps/api/elevation/json?locations=39.7391536,-104.9847034|36.455556,-116.866667&sensor=true_or_false
```

Το αποτέλεσμα που επιστρέφεται σε μορφή json από την κλήση του API είναι το παρακάτω:

```
{
  "status": "OK",
  "results": [ {
    "location": {
      "lat": 39.7391536,
      "lng": -104.9847034
    },
    "elevation": 1608.8402100
  }, {
    "location": {
      "lat": 36.4555560,
      "lng": -116.8666670
    },
    "elevation": -50.7890358
  } ]
}
```


Η ακόλουθη κλήση είναι πανομοιότυπη με την παραπάνω, εκτός από το γεγονός ότι το αίτημα δείχνει ότι το αποτέλεσμα θα εξαχθεί σε μορφή XML.

```
http://maps.googleapis.com/maps/api/elevation/xml?locations=39.739153
6,-104.9847034|36.455556,-116.866667&sensor=true_or_false
```

Το αποτέλεσμα που προκύπτει σε μορφή xml από την κλήση του API είναι το παρακάτω:


```
<ElevationResponse>
 <status>OK</status>
 <result>
  <location>
   <lat>39.7391536</lat>
   <lng>-104.9847034</lng>
  </location>
  <elevation>1608.8402100</elevation>
 </result>
 <result>
  <location>
   <lat>36.4555560</lat>
   <lng>-116.8666670</lng>
  </location>
  <elevation>-50.7890358</elevation>
 </result>
</ElevationResponse>
```


Παράδειγμα υπολογισμού υψομέτρων διαδρομής

Το παράδειγμα που ακολουθεί ζητάει υψομετρική πληροφορία κατά μήκος μίας ευθείας γραμμικής διαδρομής από το Mt. Whitney της Καλιφόρνια στο Badwater της Καλιφόρνια, το χαμηλότερο και υψηλότερο σημείο στις ηπειρωτικές ΗΠΑ. Ζητούνται τρία δείγματα ώστε να περιληφθούν τα δύο ακραία σημεία και ένα ενδιάμεσο.

```
http://maps.googleapis.com/maps/api/elevation/json?path=36.578581,-
118.291994|36.23998,-116.83171&samples=3&sensor=true_or_false
```

Προκύπτει το ακόλουθο αποτέλεσμα:


```
{
  "status": "OK",
  "results": [ {
    "location": {
```

```
      "lat": 36.5785810,
      "lng": -118.2919940
    },
    "elevation": 4411.9418945
  }, {
    "location": {
      "lat": 36.4115029,
      "lng": -117.5602608
    },
    "elevation": 1381.8616943
  }, {
    "location": {
      "lat": 36.2399800,
      "lng": -116.8317100
    },
    "elevation": -84.6169968  } ]}
```

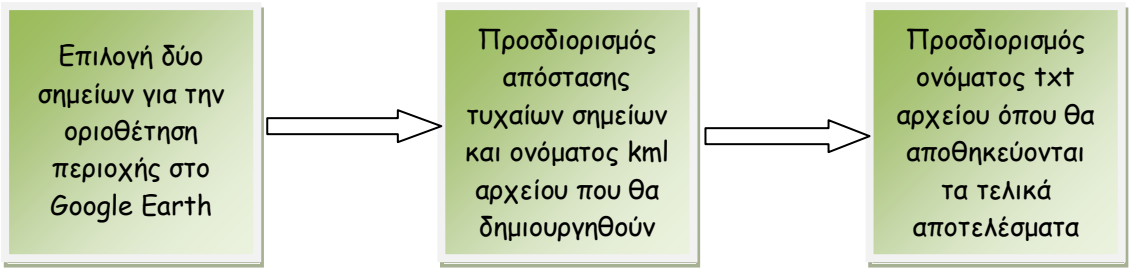
Μέρος 2

Σχεδίαση και υλοποίηση

6

ΑΠΑΙΤΗΣΕΙΣ ΑΠΟ ΤΗΝ ΕΦΑΡΜΟΓΗ ΠΟΥ ΘΑ ΥΛΟΠΟΙΗΘΕΙ

Για την υλοποίηση της εφαρμογής που επρόκειτο να κατασκευαστεί, ο χρήστης πρέπει να τηρήσει μία αλληλουχία ενεργειών. Αρχικά, το ζητούμενο είναι ο χρήστης να επιλέξει δύο σημεία στο Google Earth τα οποία θα ορίζουν την ορθογώνια περιοχή στην οποία θα πραγματοποιούνται όλοι οι υπολογισμοί. Στη συνέχεια, θα ζητείται από το χρήστη να προσδιορίσει την απόσταση που επιθυμεί να έχουν τα τυχαία σημεία που θα δημιουργηθούν, δηλαδή την πυκνότητα των σημείων, καθώς επίσης και το όνομα του kml αρχείου που θα δημιουργηθεί και θα απεικονίζει τα τυχαία σημεία. Τέλος, ο χρήστης ορίζει το όνομα του txt αρχείου που θα αποθηκεύονται όλοι οι υπολογισμοί, δηλαδή τα ζητούμενα αποτελέσματα.

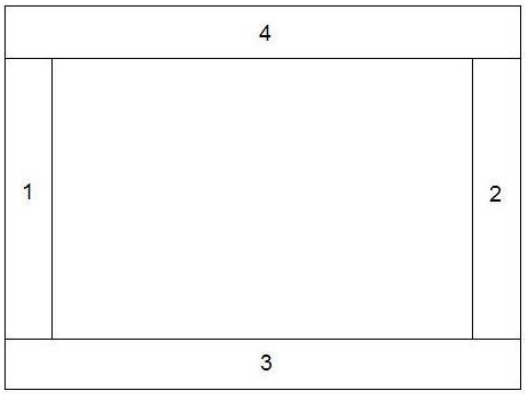


Εικόνα 6.1: Διάγραμμα με τη ροή των εργασιών που χρειάζεται να κάνει ο χρήστης

Αναλυτικότερα, η ροή των λειτουργιών που πρέπει να πραγματοποιηθούν για την περάτωση της εφαρμογής θα είναι η εξής ακόλουθη:

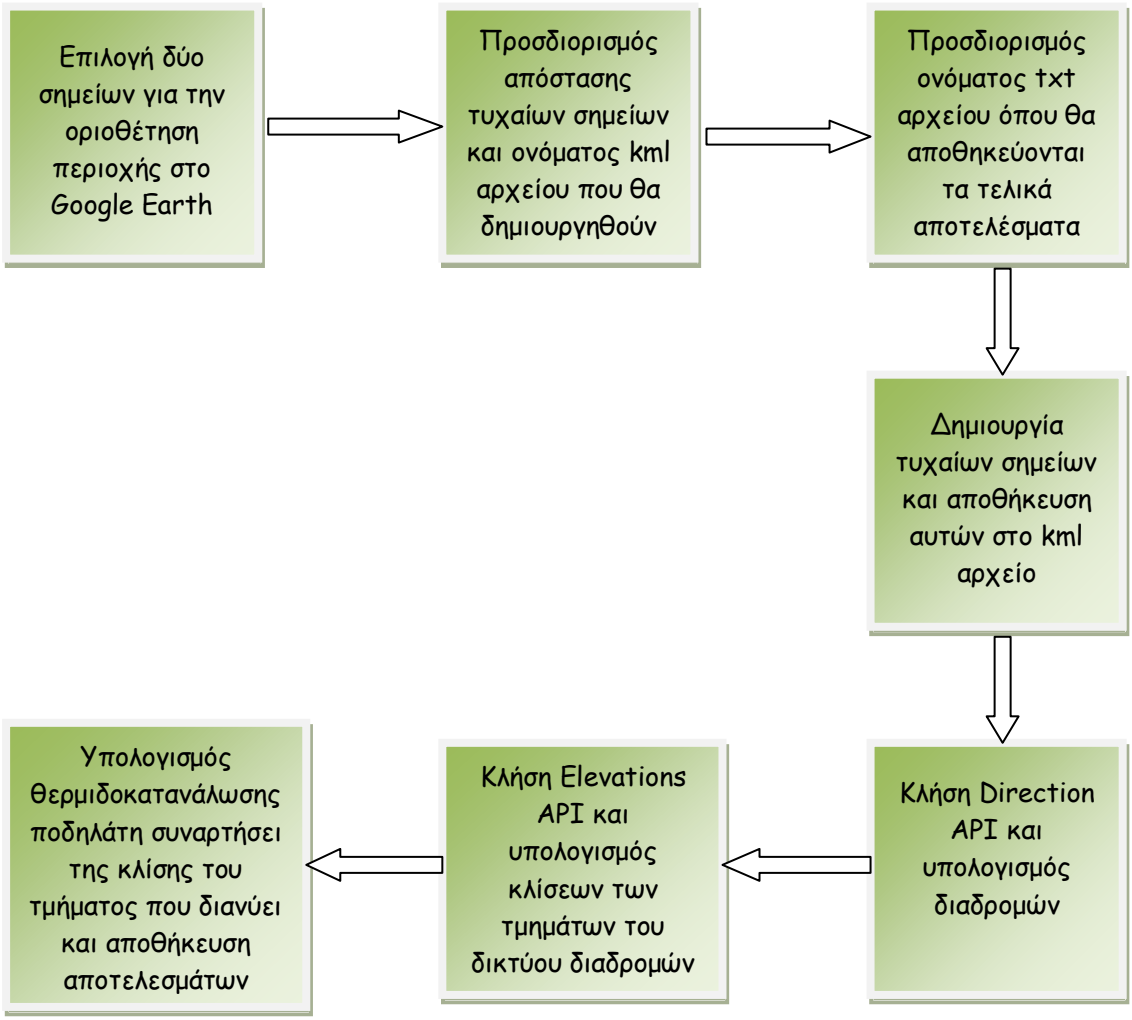
- ✓ Ο χρήστης θα δίνει αρχικά τα δύο σημεία που θα ορίζουν την περιοχή που τον ενδιαφέρει όπως ειπώθηκε και παραπάνω. Από τα δύο αυτά αντιδιαγώνια σημεία θα ορίζονται τα όρια μίας ορθογώνιας περιοχής και στη συνέχεια τα όρια μίας υποπεριοχής εντός της αρχικής περιοχής, μετατοπισμένης εσωτερικά κατά ποσοστό 15% από την αρχική. Επίσης θα προσδιορίζει το όνομα του kml και του txt αρχείου που πρόκειται να δημιουργηθούν, που θα περιέχουν την απεικόνιση σε google earth των τυχαίων σημείων και τα αποτελέσματα που θα προκύψουν από την εφαρμογή. Οι παράμετροι του βήματος αυτού θα είναι τα δύο δοθέντα σημεία, τα ονόματα των αρχείων και η επιθυμητή απόσταση των σημείων που ορίζει ο χρήστης.

- ✓ Η περιοχή που θα δημιουργείται ανάμεσα στις δύο περιοχές που περιγράφηκαν παραπάνω θα αποτελεί την περιοχή όπου θα δημιουργούνται τυχαία σημεία, ανάλογα με την πυκνότητα σημείων που έχει ορίσει ο χρήστης.



Εικόνα 6.2: Οι τέσσερις υποπεριοχές στις οποίες υπολογίζονται τα τυχαία σημεία

- ✓ Θα δημιουργούνται στη συνέχεια τυχαίες διαδρομές οι οποίες θα προκύπτουν από την τυχαία επιλογή σημείων από τις απέναντι υποπεριοχές σε συνδυασμό με τη χρήση της υπηρεσίας Direction API. Η υπηρεσία αυτή της Google θα επιστρέφει διαδρομές έπειτα από αίτημα της εφαρμογής. Έτσι θα δημιουργείται ένα δίκτυο τυχαίων διαδρομών που θα καλύπτει ολόκληρη την υπό μελέτη περιοχή. Οι παράμετροι που θα χρησιμοποιούνται στο συγκεκριμένο βήμα θα είναι οι συντεταγμένες των τυχαίων σημείων που θα έχουν προκύψει από το παραπάνω βήμα.
- ✓ Για κάθε τμήμα της κάθε διαδρομής του δικτύου αυτού θα υπολογίζεται η κλίση, που είναι και το ζητούμενο της εφαρμογής. Αυτό θα γίνεται με τη βοήθεια της υπηρεσίας Elevations API, η οποία έπειτα από αίτημα της εφαρμογής θα επιστρέφει τα υψόμετρα των κορυφών των τμημάτων κάθε διαδρομής. Ως παράμετροι θα θεωρηθούν τα υψόμετρα που θα έχουν υπολογιστεί από την κλήση της υπηρεσίας του Elevations API.
- ✓ Τέλος, για κάθε τμήμα της κάθε διαδρομής του δικτύου θα υπολογίζεται η κατανάλωση θερμίδων ενός μέσου ποδηλάτη συναρτήσει της κλίσης του τμήματος του δρόμου που διανύει, η οποία θα αποτελεί και την παράμετρο υπολογισμού των αποτελεσμάτων του βήματος αυτού.



Εικόνα 6.3: Διάγραμμα με τη ροή των λειτουργιών που θα υλοποιηθούν

6.1 Παραδοχές

Κατά την διεκπεραίωση των αλγορίθμων, έγιναν κάποιες παραδοχές, οι οποίες ήταν απαραίτητες για την απλοποίηση του προβλήματος και δεν είχαν κάποια ουσιώδη επιρροή στο αποτέλεσμα. Οι παραδοχές αυτές περιγράφονται παρακάτω:

- Για τους υπολογισμούς λήφθηκε υπόψη η μέση ακτίνα της γης. Θεωρήθηκε ότι η γη είναι σφαίρα και η ακτίνα δε μεταβάλλεται από περιοχή σε περιοχή. Η μέση ακτίνα της γης είναι ίση με R=6373 km και χρησιμοποιείται στον υπολογισμό της γωνιακής απόστασης w=S/R.
- Το Google Earth χρησιμοποιεί απλή κυλινδρική προβολή για τη βάση εικόνων του. Αυτή είναι μια απλή προβολή χάρτη όπου οι μεσημβρινοί και οι παράλληλοι είναι ευθείες παράλληλες γραμμές που ισαπέχουν μεταξύ τους, με τα δύο σύνολα να τέμνονται σε ορθές γωνίες. Αυτή η προβολή είναι επίσης γνωστή ως γεωγραφικό πλάτος/μήκος WGS84. Πιο συγκεκριμένα, το Google Earth βασίζεται στο παγκόσμιο γεωδαιτικό σύστημα WGS84. Εδώ πρέπει να σημειωθεί ότι τα

GR87 και WGS84, βασίζονται στα ελλειψοειδή GRS80 / WGS84. Οι παράμετροι των ελλειψοειδών είναι:

- Μεγάλος ημιάξονας a=6378137 m (GRS80 & WGS84)
- Μικρός ημιάξονας b=6,356,752.3141 m (GRS80), 6356752.3142 m (WGS84)
- $e^2 = 1-(1-f)^2$
- $e'^2 = 1/((1-f)*(1-f))*e^2$;
- f = 1/298.257222101 (GRS80), 1/298.257223563 (WGS84)

Το ελλειψοειδές WGS84 αποτελεί εξέλιξη του GRS80 και όπως φαίνεται έχουν πολύ μικρή διαφορά που δίνει γύρω στα 0.1 mm διαφορά στον μικρό ημιάξονα.

- Μία άλλη παραδοχή που κάνουμε είναι ότι ανάμεσα στα 2 σημεία των τμημάτων των διαδρομών που υπολογίζεται η εκάστοτε κλίση, η κλίση είναι σταθερή και δε μεταβάλλεται από αρνητική σε θετική ή αντίστροφα.
- Επίσης, θεωρούμε ότι το οδικό δίκτυο που δημιουργείται μέσω των διαδρομών αποτελείται από ευθύγραμμα τμήματα χωρίς καμπύλες. Δηλαδή στο τμήμα που υπολογίζεται η κλίση ανάμεσα σε δύο σημεία δεν παρεμβάλλονται καμπύλα τμήματα.

7

ΠΕΡΙΓΡΑΦΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ ΠΟΥ ΥΛΟΠΟΙΗΘΗΚΕ

7.1 Ανάλυση Αλγορίθμων

Για την υλοποίηση της εφαρμογής αρχικά κατασκευάστηκαν ορισμένα τμήματα αλγορίθμου στη γλώσσα προγραμματισμού PHP και στη συνέχεια συνενώθηκαν σε ένα php αρχείο. Ο κώδικας βρίσκεται αυτούσιος στο παράρτημα της εργασίας.

Το πρώτο μέρος του κώδικα αποτελεί τη βάση για την πραγματοποίηση όλων των περαιτέρω υπολογισμών. Στο τμήμα αυτό έχουν δημιουργηθεί οι ακόλουθες συναρτήσεις `make_seed()`, `myrand2()`, `myrand()`, `random_array()`, `txtoutput()`, `filesave()` και `create_random_points()` εντός των οποίων εκτελούνται οι εξής διεργασίες:

- `make_seed()`. Η συνάρτηση αυτή χρησιμοποιείται για διότι κατά την παραγωγή τυχαίων αριθμών, μετά από κάποια στιγμή, επαναλαμβάνονται οι ίδιοι αριθμοί. Για να αποφευχθεί λοιπόν αυτό και να εξασφαλιστεί η τυχαιότητα των αριθμών, λαμβάνεται υπόψη η ώρα του υπολογιστή κατά την τροφοδότηση της γεννήτριας τυχαίων αριθμών.
- `myrand2()`. Στη συνάρτηση αυτή υπολογίζονται τυχαίοι αριθμοί συναρτήσει του αριθμού των επιθυμητών δεκαδικών ψηφίων των εξαγόμενων συντεταγμένων. Οι τυχαίοι αυτοί αριθμοί βρίσκονται μεταξύ του 1 και του 10 υψωμένο στον αριθμό των δεκαδικών ψηφίων που θα δώσει ο χρήστης.
- `myrand()`. Εδώ παράγονται τυχαίοι αριθμοί που αντιστοιχούν στο γεωγραφικό μήκος και πλάτος μεταξύ ενός ελάχιστου κι ενός μέγιστου ακέραιου αριθμού και προστίθενται σε αυτούς οι τυχαίοι δεκαδικοί αριθμοί που προέκυψαν από τη συνάρτηση myrand2(). Έτσι δημιουργούνται τυχαίοι αριθμοί με συγκεκριμένο αριθμό δεκαδικών ψηφίων μεταξύ ενός ελαχίστου κι ενός μεγίστου, ο συνδυασμός των οποίων θα αποτελέσει στη συνέχεια τις γεωγραφικές συντεταγμένες των σημείων που θα δημιουργηθούν.
- `random_array()`. Δημιουργεί τυχαίους αριθμούς χρησιμοποιώντας τα αποτελέσματα της συνάρτησης `make_seed()` για την παραγωγή τυχαίων αριθμών και την αποφυγή επαναλαμβανόμενων, και καλώντας τη συνάρτηση `myrand()` για την παραγωγή τυχαίων φ και λ, τα οποία καταχωρούνται σε πίνακες για την ευκολότερη μετέπειτα χρήση τους.

- `txtoutput()`. Στο σημείο αυτό δημιουργείται η δομή του στοιχείου <Placemark> όμοια με αυτή ενός kml αρχείου, όπου τοποθετούνται σε σειρά οι συντεταγμένες που προέκυψαν παραπάνω.
- `filesave()`. Γράφει (append) σε ένα txt αρχείο τα περιεχόμενα φ, λ των πινάκων με τις συντεταγμένες που δημιουργούνται προσθέτοντας την πλευρά (Side) που βρίσκεται κάθε σημείο σε κάθε γραμμή.
- `create_random_points()`. Η συνάρτηση create_random_points() είναι η βασική συνάρτηση του κώδικα, μέσα στην οποία πραγματοποιούνται όλοι οι ουσιώδεις υπολογισμοί. Δέχεται ως παραμέτρους τις συντεταγμένες δύο αντιδιαγώνιων σημείων για τον προσδιορισμό της περιοχής που ενδιαφέρει το χρήστη, το όνομα του kml αρχείου που πρόκειται να δημιουργηθεί, το όνομα ενός βοηθητικού txt αρχείου, και την επιθυμητή από το χρήστη απόσταση μεταξύ των τυχαίων σημείων που θα παραχθούν, που θα καθορίσει την πυκνότητα αυτών. Στη συνάρτηση αυτή εκτελούνται κατά σειρά οι παρακάτω υπολογισμοί:
  - ✓ αρχικά ορίζεται ο αριθμός των δεκαδικών ψηφίων που χρειάζεται για την απαιτούμενο ακρίβεια για τον ορισμό των γεωγραφικών συντεταγμένων των σημείων. Ο αριθμός αυτός συγκεκριμένα έχει οριστεί ίσος με 6.
  - ✓ Ορίζεται ο παρονομαστής 10 υψωμένος στον αριθμό των δεκαδικών, με τον οποίο διαιρούνται οι τιμές των γεωγραφικών συντεταγμένων, οι οποίες εισάγονται στον κώδικα ως ακέραιες τιμές χωρίς την υποδιαστολή, για να δώσουν την πραγματική τους τιμή.
  - ✓ Ορίζονται οι γεωγραφικές συντεταγμένες φ, λ δύο αντιδιαμετρικών σημείων στην υπό μελέτη περιοχή, εκ των οποίων προκύπτουν οι τέσσερις κορυφές μιας ορθογώνιας περιοχής εντός της οποίας θα διεξαχθούν οι υπολογισμοί. Οι συντεταγμένες των σημείων αυτών τοποθετούνται σε πίνακες.
  - ✓ Καθορίζεται από τον χρήστη η απόσταση (μήκος τόξου) που επιθυμεί να έχουν τα τυχαία σημεία μεταξύ τους. Στη συνέχεια, υπολογίζεται η επίκεντρη γωνία w που αντιστοιχεί σε αυτό το μήκος τόξου σύμφωνα με τη σχέση $w=S/R$ σε rad, όπου S το μήκος τόξου και R η μέση ακτίνα της γης σε km. Η υπολογισμένη γωνιακή απόσταση μετατρέπεται σε μοίρες. Αυτή η προεργασία θα χρησιμεύσει αργότερα για τον υπολογισμό του αριθμού των τυχαίων σημείων στην οριοθετημένη περιοχή.
  - ✓ Εφόσον ορίστηκε η ορθογώνια περιοχή όπου θα γίνουν οι υπολογισμοί, έπρεπε να οριοθετηθεί μία υποπεριοχή εντός της αρχικής με παράλληλη μετατόπιση, η οποία επιλέχθηκε να είναι εσωτερικά κατά 15% της επίκεντρης γωνίας που αντιστοιχεί στις τέσσερις πλευρές της ορθογώνιας περιοχής.
  - ✓ Αφού υπολογίστηκε η επίκεντρη γωνία που αντιστοιχεί στην εσωτερική μετατόπιση του αρχικού ορθογωνίου, λήφθηκαν εννέα περιπτώσεις για τον υπολογισμό των συντεταγμένων των τεσσάρων σημείων του νέου ορθογωνίου. Εδώ πρέπει να σημειωθεί ότι οι γεωγραφικές συντεταγμένες στο Google Earth έχουν αρχή το μεσημβρινό του Greenwich και τον Ισημερινό. Το γεωγραφικό μήκος μετριέται από 0 έως 180 μοίρες ανατολικά, και 0 έως -180 δυτικά του μεσημβρινού του Greenwich, ενώ το γεωγραφικό πλάτος από 0 έως 90 μοίρες βόρεια, και 0 έως -90 μοίρες νότια του Ισημερινού. Για το λόγο αυτό, λαμβάνονται οι εξής περιπτώσεις: η μελετώμενη περιοχή μπορεί να βρίσκεται είτε σε ένα από τα τέσσερα

ημισφαίρια (βορειοανατολικό, βορειοδυτικό, νοτιοανατολικό, νοτιοδυτικό), είτε να βρίσκεται στα όρια δύο ημισφαιρίων, είτε να «πατάει» και στα τέσσερα ημισφαίρια. Πραγματοποιείται επομένως διερεύνηση και προκύπτουν τα τέσσερα σημεία του εσωτερικού ορθογωνίου, τα οποία επίσης τοποθετούνται σε πίνακες. Η περιοχή που θα μελετηθεί είναι η περιοχή που σχηματίζεται ανάμεσα στα δύο ορθογώνια, η οποία χωρίζεται σε τέσσερις υποπεριοχές για την καλύτερη επεξεργασία της.

- ✓ Το επόμενο βήμα είναι ο υπολογισμός του αριθμού των σημείων που θα παραχθούν με τυχαία κατανομή για κάθε μία από τις τέσσερις υποπεριοχές που σχηματίστηκαν παραπάνω, συναρτήσει του εμβαδού κάθε μίας και της ορισμένης από το χρήστη απόστασης που αναφέρθηκε παραπάνω. Ο αριθμός των σημείων για κάθε υποπεριοχή υπολογίζεται ως εξής: $n_i=(S_1*S_2)_i/(d^2)$, όπου i=1,2,3,4, S1 και S2 τα πλευρικά μήκη τόξου κάθε υποπεριοχής και d η ορισμένη από το χρήστη επιθυμητή απόσταση μεταξύ των σημείων.
- ✓ Ακολουθεί η κλήση της συνάρτησης `random_array()` για κάθε μία από τις υποπεριοχές κι επιστρέφονται ως αποτέλεσμα οι τυχαίες συντεταγμένες φ, λ σε αντίστοιχους πίνακες. Γίνεται συγχώνευση των πινάκων.
- ✓ Δημιουργήθηκε ανεξάρτητα ένα txt αρχείο (headers.txt) το οποίο έχει τη δομή ενός kml αρχείου. Γίνεται ανάγνωση του αρχείου αυτού και έπειτα από κλήση της συνάρτησης `txtoutput()` για κάθε υποπεριοχή, δημιουργείται ένα kml αρχείο με βάση το περιεχόμενο του txt αρχείου και την προσθήκη των στοιχείων <Placemark>, τα οποία είναι όσα και τα τυχαία σημεία. Προηγουμένως έχουν υπολογιστεί και οι γεωγραφικές συντεταγμένες του κέντρου της υπό μελέτη περιοχής.
- ✓ Στο kml αρχείο που δημιουργήθηκε προστέθηκαν δύο ακόμα πεδία <Placemark> για την απεικόνιση δύο πολυγώνων (<Polygon>) που προέκυψαν από τις δύο ορθογώνιες περιοχές. Επίσης, προστέθηκαν τα στοιχεία εκείνα που είναι απαραίτητα για να κλείσει ένα kml αρχείο και να είναι αναγνώσιμο και απεικονίσιμο.
- ✓ Τέλος, δημιουργείται ένα txt αρχείο, το οποίο φέρει αντίστοιχο του kml αρχείου όνομα, όπου αποθηκεύονται τα δεδομένα: αύξον αριθμός, γεωγραφικό μήκος, γεωγραφικό πλάτος και πλευρά που ανήκει το κάθε σημείο, καλώντας τη συνάρτηση `filesave()` για κάθε υποπεριοχή.
- ✓ Καλείται η συνάρτηση `slope()` στην οποία υπολογίζονται διαδρομές, κλίσεις των διαδρομών και θερμιδοκατανάλωση του ποδηλάτη συναρτήσει της υπολογισμένης κλίσης, και με βάση τις υπολογισμένες κλίσεις δημιουργούνται πέντε κλάσεις, κατατάσσονται οι κλίσεις στις κατηγορίες που ανήκουν και στη συνέχεια απεικονίζονται στο google earth με το αντίστοιχο χρώμα.

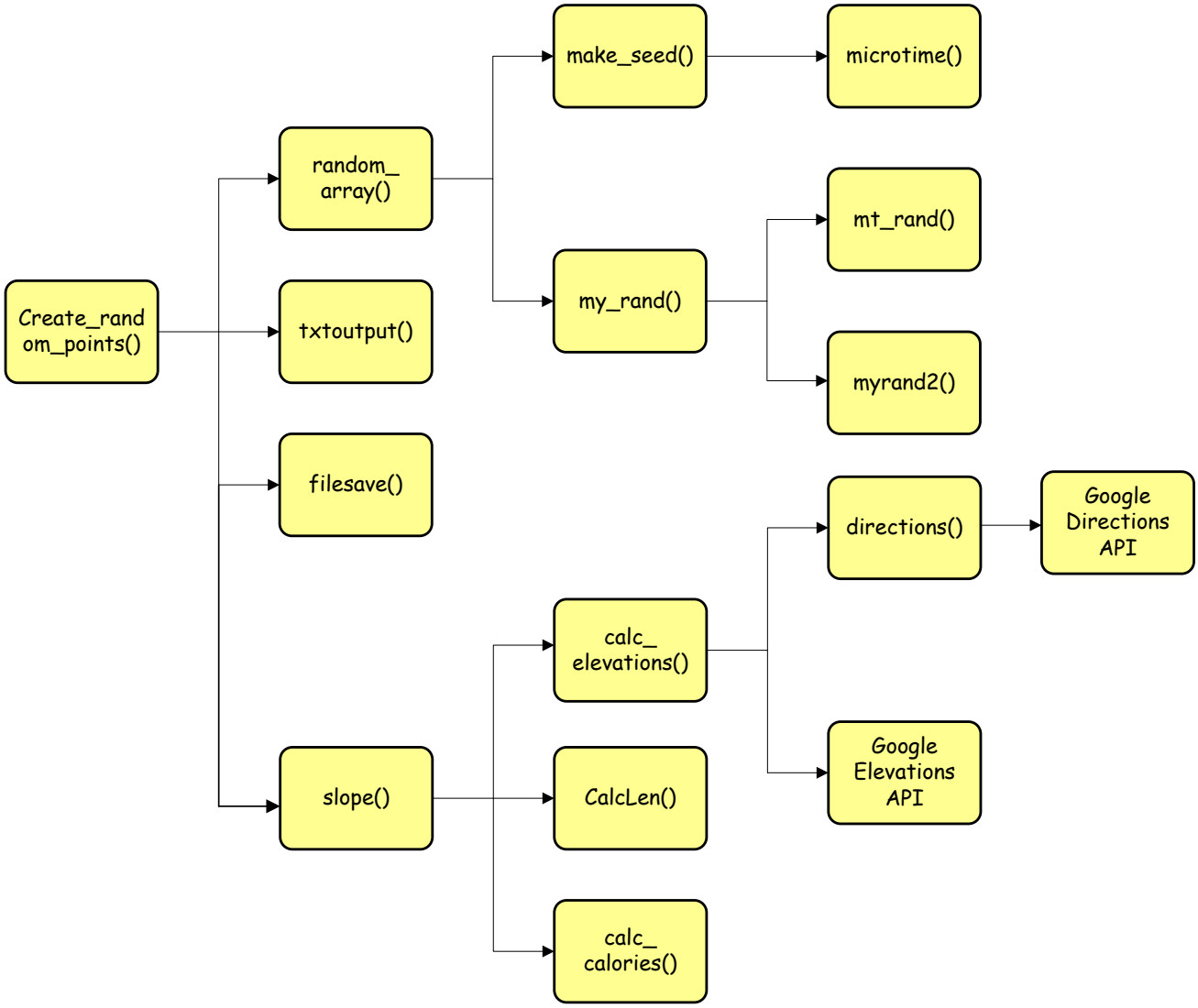
Ο κώδικας συνεχίζεται με την αξιοποίηση των αποτελεσμάτων που προέκυψαν παραπάνω με σκοπό τον υπολογισμό κλίσεων διαδρομών καθώς επίσης και της κατανάλωσης θερμίδων ενός μέσου ποδηλάτη συναρτήσει της κλίσεως των διαδρομών αυτών καθώς τις διασχίζουν. Οι συναρτήσεις που λαμβάνουν μέρος

στους υπολογισμούς αυτούς είναι οι εξής: `directions()`, `calc_elevations()`, `CalcLen()`, `calc_calories()` και `slope()`.

- `directions()`. Σκοπός της συνάρτησης αυτής είναι ο υπολογισμός πιθανής διαδρομής από δύο δοθέντα σημεία. Εντός της συνάρτησης αυτής, αρχικά δημιουργείται η συμβολοσειρά που πρόκειται να περάσει στην κλήση του Google Direction API για τον υπολογισμό των πιθανών διαδρομών. Στη συνέχεια, πραγματοποιείται κλήση του Google Direction API, επιστρέφεται το ζητούμενο αποτέλεσμα και γίνεται αποκωδικοποίηση της συμβολοσειράς JSON σε ένα αντικείμενο PHP. Έτσι, από δύο σημεία που επιλέχθηκαν από δύο απέναντι υποπεριοχές από τον πρώτο αλγόριθμο, προέκυψε μία διαδρομή με συγκεκριμένα ενδιάμεση σημεία, των οποίων οι γεωγραφικές συντεταγμένες είναι το εξαγόμενο αποτέλεσμα από το PHP αντικείμενο. Οι συντεταγμένες αυτές τοποθετούνται σε πίνακες.
- `calc_elevations()`. Εδώ συλλέγεται η υψομετρική πληροφορία της μελετώμενης διαδρομής. Καλείται η συνάρτηση `directions()`. Δημιουργείται αντίστοιχα με παραπάνω η συμβολοσειρά που θα περάσει στην κλήση του Google Elevation API. Έπειτα ακολουθεί η κλήση του Google Elevation API, επιστρέφεται το ζητούμενο αποτέλεσμα και γίνεται αποκωδικοποίηση της συμβολοσειράς JSON σε ένα αντικείμενο PHP. Έτσι, ο χρήστης δίνει στο Google Elevation API τα σημεία της διαδρομής που προέκυψαν από την κλήση του Google Direction API, και του επιστρέφονται οι γεωγραφικές συντεταγμένες των σημείων αυτών συμπληρωμένες με το υψόμετρο καθενός σημείου, που τοποθετούνται στη συνέχεια σε πίνακα και αποθηκεύονται σε ένα txt αρχείο.
- `CalcLen()`. Στο σημείο αυτό πραγματοποιούνται όλοι οι απαραίτητοι υπολογισμοί για τη μετατροπή των γεωγραφικών συντεταγμένων φ, λ, h όλων των ενδιάμεσων σημείων της διαδρομής σε τρισδιάστατες γεωδαιτικές συντεταγμένες X, Y, Z. έπειτα, υπολογίζεται η κεκλιμένη απόσταση για κάθε τμήμα της διαδρομής (ανά δύο διαδοχικά ενδιάμεσα σημεία).
- `calc_calories()`. Η συνάρτηση αυτή υπολογίζει πόσες θερμίδες καταναλώνει ένας ποδηλάτης μάζας 75 kg σε κάθε τμήμα της διαδρομής που διανύει, ο οποίος πηγαίνει με σταθερή μέση ταχύτητα 10 km/h σε ιδανικό δρόμο χωρίς την επίδραση μετωπικού ή πλάγιου ανέμου.
- `slope()`. Καλείται η συνάρτηση `calc_elevations()` μέσω της οποίας υπολογίστηκαν τα υψόμετρα των ενδιάμεσων σημείων της διαδρομής, και υπολογίζονται οι διαδοχικές υψομετρικές διαφορές μεταξύ των σημείων αυτών. Καλείται στη συνέχεια η συνάρτηση CalcLen() όπου υπολογίστηκαν οι αποστάσεις μεταξύ των σημείων, και υπολογίζεται η κλίση w=arcsin(dh/L) μεταξύ των διαδοχικών σημείων της διαδρομής. Τέλος, τα αποτελέσματα φ_i, λ_i, H_i, φ_{i+1}, λ_{i+1}, H_{i+1}, L, dH, Slope% και Cal/min για τα σημεία της διαδρομής τοποθετούνται συγκεντρωτικά σε πίνακες και αποθηκεύονται σε ένα txt αρχείο.
- Εκτός όλων αυτών των συναρτήσεων, δίνονται οι συντεταγμένες των σημείων αρχής και τέλους της διαδρομής και καλείται η συνάρτηση `slope()` για την πραγματοποίηση των υπολογισμών.
- Εκτός όλων των συναρτήσεων, ζητούνται από το χρήστη το επιθυμητό όνομα του kml αρχείου, και η απόσταση μεταξύ των τυχαίων σημείων ενώ δίνεται εξ ορισμού το όνομα του υπάρχοντος βοηθητικού αρχείου. Γίνεται κλήση της

συνάρτησης `create_random_points()` και με διαδοχικές κλήσεις των λοιπών συναρτήσεων εκτελείται ο αλγόριθμος.

Στο διάγραμμα που ακολουθεί φαίνονται οι μονάδες κώδικα (συναρτήσεις) και η σειρά με την οποία η μία καλεί την επόμενη.



Εικόνα 7.1: Μονάδες κώδικα (συναρτήσεις) και απεικόνιση της αλληλουχίας κλήσης τους

ΣΥΝΑΡΤΗΣΗ	ΠΑΡΑΜΕΤΡΟΙ ΕΙΣΟΔΟΥ	ΚΑΘΟΛΙΚΕΣ ΜΕΤΑΒΛΗΤΕΣ	ΧΡΗΣΙΜΟΤΗΤΑ ΣΥΝΑΡΤΗΣΗΣ	ΤΙ ΕΠΙΣΤΡΕΦΕΙ Η ΣΥΝΑΡΤΗΣΗ	ΚΑΘΟΛΙΚΕΣ ΜΕΤΑΒΛΗΤΕΣ ΠΟΥ ΜΕΤΑΒΑΛΛΕΙ
	άμεση χρήση (αποτελούν ταυτόχρονα και παράμετροι των συναρτήσεων)				
create_random_points()	Γεωγραφικές συντεταγμένες φ και λ δύο σημείων, όνομα kml αρχείου, όνομα υπάρχοντος txt αρχείου, απόσταση μεταξύ τυχαίων σημείων	Γεωγραφικές συντεταγμένες φ και λ δύο σημείων, όνομα kml αρχείου, όνομα υπάρχοντος txt αρχείου, απόσταση μεταξύ τυχαίων σημείων	Υπολογίζει τις κορυφές της περιοχής που επιλέχθηκε και μίας υποπεριοχής μετατοπισμένης εσωτερικά κατά 15%, καλώντας την random_array() δημιουργεί τυχαίες συντεταγμένες φ και λ και επομένως τυχαία σημεία, δημιουργεί ένα kml αρχείο προσθέτοντας τις νέες συντεταγμένες, αποθηκεύει τα αποτελέσματα σε ένα txt αρχείο	-	-
random_array()	Αριθμός τυχαίων σημείων, μέγιστα κι ελάχιστα φ και λ υποπεριοχής, αριθμός δεκαδικών ψηφίων	-	Δημιουργεί σημεία με τυχαίους συνδυασμούς φ και λ	Πίνακα με τις συντεταγμένες τυχαίων σημείων	-
txtoutput()	Όνομα kml αρχείου, φ και λ του κέντρου της περιοχής, φ και λ τυχαίου σημείου, μετρητής σημείων	-	Δημιουργεί την τυπική δομή των Placemarks ενός kml αρχείου συμπεριλαμβάνοντας τα νέα τυχαία σημεία	-	-
filesave()	Όνομα txt αρχείου που δημιουργείται, φ και λ τυχαίου σημείου, μετρητής σημείων	-	Γράφει τα αποτελέσματα (Α/Α, φ, λ, Πλευρά) σε txt αρχείο	-	-

make_seed()	-	-	Δημιουργεί αριθμούς με βάση το ρολόι του υπολογιστή	Αριθμούς με βάση το ρολόι του υπολογιστή	-
my_rand()	Ελάχιστος και μέγιστος αριθμός (για τον προσδιορισμό του εύρους των τυχαίων σημείων), αριθμός δεκαδικών ψηφίων συντεταγμένων	-	Δημιουργεί τυχαίους δεκαδικούς αριθμούς	Τυχαίους δεκαδικούς αριθμούς εντός κάποιων ορίων	-
myrand2()	Αριθμός δεκαδικών ψηφίων συντεταγμένων	-	Δημιουργεί τυχαίους ακέραιους αριθμούς	Τυχαίους ακέραιους αριθμούς εντός κάποιων ορίων	-
slope()	Γεωγραφικές συντεταγμένες φ και λ αρχής και τέλους διαδρομής, όνομα txt αρχείου υψομέτρων, όνομα txt αρχείου με την προσθήκη θερμοδοκατανάλωσης	Γεωγραφικές συντεταγμένες φ και λ αρχής και τέλους διαδρομής, όνομα txt αρχείου υψομέτρων, όνομα txt αρχείου με την προσθήκη θερμοδοκατανάλωσης	Υπολογίζει υψομετρικές διαφορές, κλίσεις και κατανάλωση θερμίδων ενός ποδηλάτη με την κλήση των αντίστοιχων συναρτήσεων για κάθε τμήμα της ζητούμενης διαδρομής και αποθηκεύει τα αποτελέσματα σε ένα txt αρχείο	-	-
calc_elevations()	φ και λ σημείων αρχής και τέλους τμήματος διαδρομής, όνομα txt αρχείου	-	Καλεί το Google Elevations API επιστρέφοντας υψόμετρα για τα σημεία της διαδρομής που προέκυψαν από την κλήση της συνάρτησης directions()	-	-

	υψομέτρων				
CalcLen()	φ , λ και h σημείων αρχής και τέλους τμήματος διαδρομής	-	Υπολογίζει το κεκλιμένο μήκος των τμημάτων της ζητούμενης διαδρομής	Κεκλιμένο μήκος τμήματος διαδρομής	-
calc_calories()	Κλίση τμήματος διαδρομής	-	Υπολογίζει την κατανάλωση θερμίδων ενός μέσου ποδηλάτη σε κάθε τμήμα της ζητούμενης διαδρομής	Κατανάλωση θερμίδων ποδηλάτη	-
directions()	φ και λ σημείων αρχής και τέλους διαδρομής, πίνακας σημείων μιας διαδρομής	-	Καλεί το Google Directions API το οποίο επιστρέφει σημεία της δοσμένης διαδρομής	Μετρητή αριθμού σημείων που αποτελούν μία διαδρομή	-

Πίνακας 7.1: Οι συναρτήσεις που χρησιμοποιήθηκαν

Μέρος 3

Εφαρμογές - Μελέτες Περίπτωσης

8

ΤΥΠΟΙ ΥΠΟΛΟΓΙΣΜΟΥ ΘΕΡΜΙΔΙΚΗΣ ΚΑΤΑΝΑΛΩΣΗΣ ΠΟΔΗΛΑΤΗ ΚΑΙ ΠΑΡΑΔΟΧΕΣ

Ο γενικός τύπος που υπολογίζει την κατανάλωση των θερμίδων ενός ποδηλάτη είναι ο παρακάτω:

W = Cv [K1 + {K2(Cv+Cw)(Cv+Cw)} + {10.32Em(s/100 + 1.01a/g)}]

Όπου:

- W = δύναμη σε watts
  - 1 W = 1 joule/sec
  - 69.78W = 1000 calories/min = 1 kilocal/min = 1 Calorie/min
  - 1 Calorie = 4186 joules
- Cv = ταχύτητα ποδηλάτη σε meters/sec
  - 1 mph = .447 meters/sec
  - 1 mph = 1.609 kilometeres/hr
- Τα K1 και K2 είναι σταθερές (βλ. πίνακα παρακάτω)
- Cw = μετωπικός άνεμος σε meters/sec
- Em = μάζα ποδηλάτη και ποδηλάτου σε kg
  - 1 pound = .4536 kg
- s = κλίση σε %
- a = επιτάχυνση ποδηλάτου σε meters/(sec)(sec)
- g = επιτάχυνση της βαρύτητας = 9.806 m/sec-sec στο επίπεδο της θάλασσας

ΣΤΑΘΕΡΕΣ K1 ΚΑΙ K2:

ASSUMPTIONS	MTN BIKE	ROAD BIKE
BICYCLE WT	**15 kg**	**10 kg**
RIDER + GEAR	**80 kg**	**75 kg**
K1	7.845	3.509
K2	0.3872	0.2581

*Το Cv είναι η ταχύτητα του αέρα του ποδηλάτη (δηλαδή η αντίσταση στο πετάλι είναι η αντίσταση του αέρα στο σώμα του ποδηλάτη και του ποδηλάτου καθώς ποδηλατεί) και δεν είναι η ταχύτητα εδάφους
** Οι σταθερές K1 και K2 υπολογίζονται για road bike/ποδήλατο/εξοπλισμό των 85 kg ή mountain bike/ποδήλατο/εξοπλισμό των 95 kg.

Για τον υπολογισμό της ενέργειας που καταναλώνεται στο πετάλι του ποδηλάτου χρησιμοποιείται ο παρακάτω τύπος:

Cal/min (expended at the pedal) = [(K1)(Cv) + (K2)(Cv)(Cv)(Cv)]/69.78

Ο αριθμός των θερμίδων που θα πρέπει να καταναλώνονται ανά λεπτό για να διατηρηθεί η ταχύτητα Cv μίλια/ώρα θα ήταν:

Ingested Cal/min = {[(K1)(Cv x .497) + (K2)(Cv x .497)(Cv x .497)(Cv x .497)]/69.78}/.25

Εάν γνωρίζουμε τη μέση ταχύτητα του ποδηλάτη και το συνολικό χρόνο ποδηλασίας μπορούμε να υπολογίσουμε των αριθμό των θερμίδων που καταναλώθηκαν. Ακολουθούν μερικά παραδείγματα:

- 5 mph - 7 Cal/mile - 37 Cal/hr
- 10 mph - 13 Cal/mile - 133 Cal/hr
- 15 mph - 23 Cal/mile - 349 Cal/hr
- 20 mph - 37 Cal/mile - 742 Cal/hr
- 25 mph - 55 Cal/mile - 1374 Cal/hr
- 30 mph - 77 Cal/mile - 2303 Cal/hr

Κατά τον υπολογισμό της κατανάλωσης των θερμίδων ενός ποδηλάτη, γίνονται οι εξής παραδοχές:

- Ανάλογα με το είδος της ποδηλασίας (mountain bike ή road bike), γίνονται διαφορετικές υποθέσεις. Στην περίπτωση αυτή, επιλέχθηκε ο υπολογισμός της θερμιδοκατανάλωσης για ποδηλάτες δρόμου. Έτσι, το βάρος του ποδηλάτου θεωρείται ίσο με 10 kg, του ποδηλάτη μαζί με τον εξοπλισμό ίσο με 75 kg, η σταθερά K1=3.509 και η σταθερά K2=0.2581. Άρα $\mathbf{E_m=85\ kg}$.
- Ο αντίθετος άνεμος είναι μηδενικός $\mathbf{C_w=0\ m/sec}$.
- Η ταχύτητα του ποδηλάτη είναι σταθερή (μηδενική επιτάχυνση - επιβράδυνση).
- Ο δρόμος είναι ιδανικός, δηλαδή σταθερές κλίσεις, και το οδικό δίκτυο αποτελείται από ευθύγραμμα τμήματα.
- Η επιτάχυνση της βαρύτητας g θεωρείται σταθερή και ίση με 9.806 m/s^2 στην επιφάνεια της θάλασσας.
- Η κλίση ενός τμήματος μιας διαδρομής είναι αρνητική ή μηδενική τότε η θερμιδοκατανάλωση του ποδηλάτη είναι ίση με μηδέν.

Έπειτα από τις παραπάνω υποθέσεις που πραγματοποιήθηκαν ο τύπος απλοποιήθηκε σε:

W = Cv* [(K1) + (K2**)(Cv*)(Cv*)]**

Επομένως, ο παραπάνω τύπος υπολογίζει πόσες θερμίδες καταναλώνει ένας ποδηλάτης μάζας 75 kg σε κάθε τμήμα της διαδρομής που διανύει, ο οποίος πηγαίνει με σταθερή μέση ταχύτητα σε ιδανικό δρόμο χωρίς την επίδραση μετωπικού ή πλάγιου ανέμου. Η ταχύτητα επιλέχθηκε να είναι **C_v=10 km/h**.

9

ΥΠΟΛΟΓΙΣΜΟΣ ΚΛΙΣΕΩΝ ΤΥΧΑΙΩΝ ΔΙΑΔΡΟΜΩΝ ΓΥΡΩ ΑΠΟ ΤΗΝ ΠΕΡΙΟΧΗ ΤΗΣ ΠΟΛΥΤΕΧΝΕΙΟΥΠΟΛΗΣ ΖΩΓΡΑΦΟΥ

9.1 Περιοχή Υπολογισμών

Η περιοχή που επιλέχθηκε για να υλοποιηθεί η εφαρμογή είναι εκείνη γύρω από την ευρύτερη περιοχή της Πολυτεχνειούπολης Ζωγράφου. Πιο συγκεκριμένα, περιλαμβάνει το μεγαλύτερο τμήμα του Δήμου Παπάγου, την περιοχή του Ζωγράφου, την Καισαριανή και σχεδόν ολόκληρη την περιοχή του Βύρωνα.

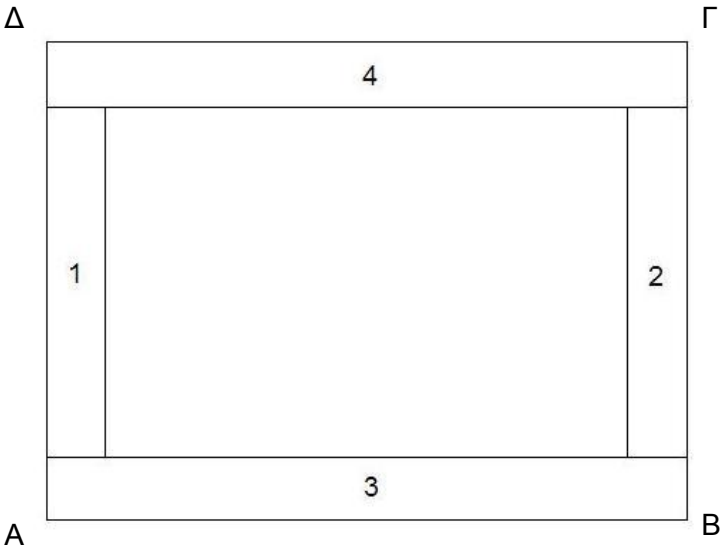
Οι περιοχές αυτές, και κυρίως ο δήμος Ζωγράφου διαθέτουν έντονο ανάγλυφο, με έντονες υψομετρικές διαφορές και μεγάλες κλίσεις δρόμων. Αυτός είναι και ο βασικός λόγος που επιλέχθηκε να μελετηθεί η περιοχή αυτή, καθώς μία περιοχή με ήπιο ανάγλυφο δε θα έδινε ποικιλία αποτελεσμάτων.

Οι συντεταγμένες των κορυφών της περιοχής εφαρμογής του αλγορίθμου είναι:

(φ, λ)	
Α (37.956382, 23.755163)	Γ (37.993854, 23.808015)
Β (37.956382, 23.808015)	Δ (37.993854, 23.755163)

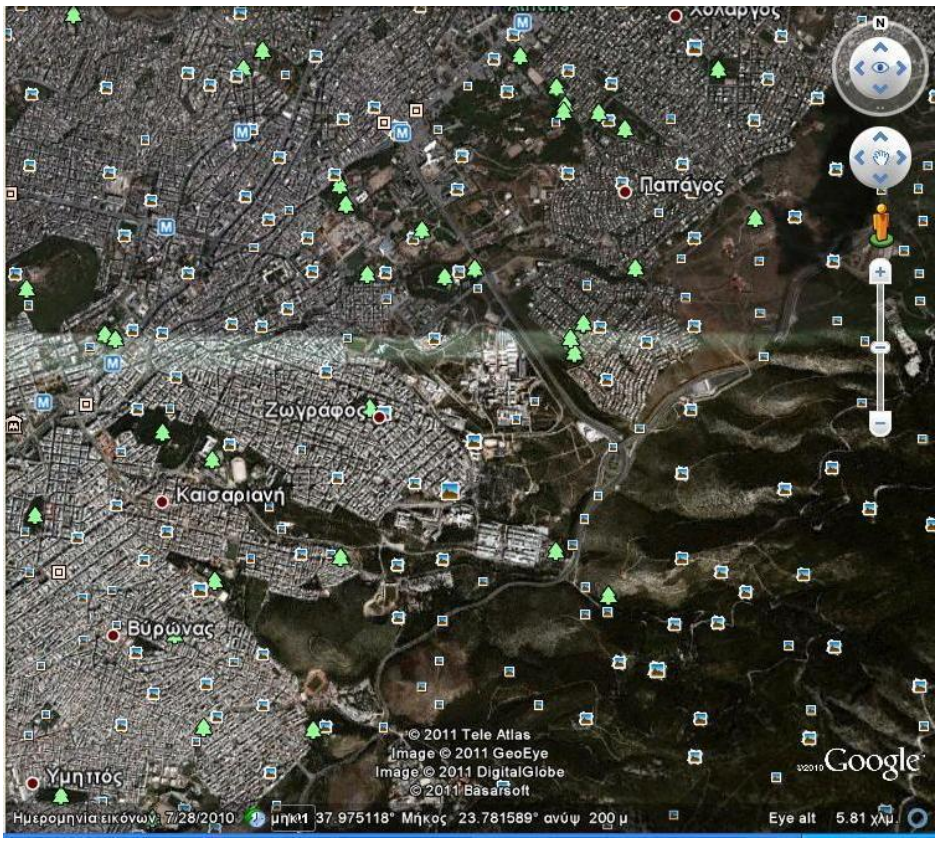
Πίνακας 9.1: Συντεταγμένες κορυφών της πρώτης περιοχής μελέτης

Στην ακόλουθη εικόνα φαίνονται οι κορυφές και οι τέσσερις υποπεριοχές της περιοχής μελέτης, με βάση τις οποίες υπολογίστηκαν τα τυχαία σημεία.

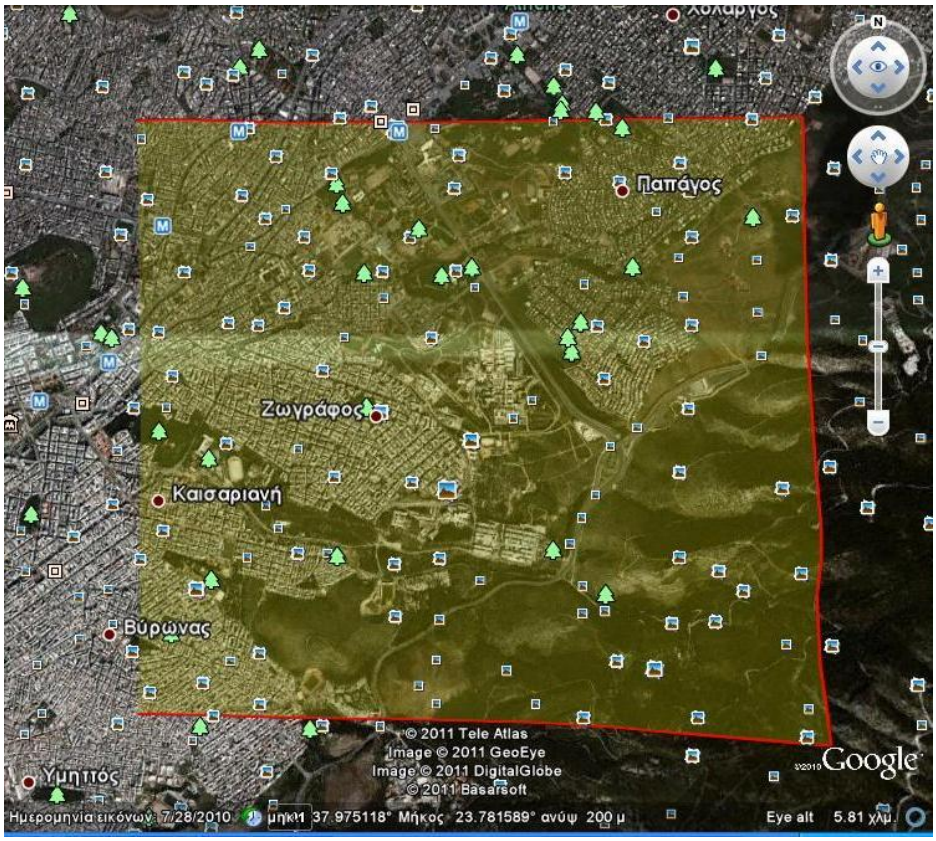


Εικόνα 9.1: Κορυφές και υποπεριοχές της πρώτης περιοχής μελέτης

Στη συνέχεια ακολουθεί χάρτης απεικόνισης στο Google Earth της υπό μελέτη περιοχής.



Εικόνα 9.2: Περιοχή μελέτης Πολυτεχνειούπολης Ζωγράφου



Εικόνα 9.3: Οριοθετημένη περιοχή μελέτης Πολυτεχνειούπολης Ζωγράφου

9.2 Παράμετροι Υπολογισμών

Οι βασικές παράμετροι υπολογισμών της εφαρμογής είναι οι εξής:

- Οι γεωγραφικές συντεταγμένες των δύο ακραίων σημείων που οριοθετούν την περιοχή ενδιαφέροντος. Αυτές μεταβάλλονται σύμφωνα με την περιοχή αλλά και την έκταση αυτής στην οποία θα επιλέξει να υλοποιήσει την εφαρμογή ο χρήστης.
- Η απόσταση που ορίζει ο χρήστης μεταξύ των τυχαίων σημείων που θα δημιουργηθούν στην περιοχή, η οποία θα δώσει την πυκνότητα και των αριθμό των τυχαίων σημείων στις τέσσερις υποπεριοχές της αρχικής περιοχής.
- Οι γεωγραφικές συντεταγμένες φ και λ αρχής και τέλους κάθε πιθανής διαδρομής αποτελούν στη συνέχεια παραμέτρους για τον υπολογισμό των δυνατών διαδρομών και των ενδιάμεσων σημείων αυτών.
- Οι γεωγραφικές συντεταγμένες φ και λ των ενδιάμεσων σημείων των διαδρομών (σημείων αρχής και τέλους των τμημάτων των διαδρομών) αποτελούν με τη σειρά τους παραμέτρους για τον υπολογισμό των υψομέτρων των σημείων αυτών.
- Η κλίση των τμημάτων κάθε διαδρομής, η οποία χρησιμοποιείται για τον υπολογισμό της θερμιδοκατανάλωσης του ποδηλάτη.

9.3 Πραγματοποίηση Υπολογισμών

Για την υλοποίηση των υπολογισμών, κρίθηκε αναγκαία η εφαρμογή τυχαίας καθυστέρησης για τους υπολογισμούς ώστε να μην υπάρχει συστηματικότητα στην κλήση των Google Direction και Elevations API και να μην διακόψει η Google την παροχή των δύο υπηρεσιών.

Αφού τελειοποιήθηκε ο αλγόριθμος, τέθηκε το σύστημα σε λειτουργία αφήνοντάς το να «τρέξει» για την ευρύτερη περιοχή της Πολυτεχνειούπολης. Μετά το τέλος του «τρεξίματος», εξήχθησαν τρία txt και δύο kml αρχεία με τα αποτελέσματα που προέκυψαν από την υλοποίηση του κώδικα για την συγκεκριμένη περιοχή. Το τελικό txt αρχείο περιέχει όλα τα αποτελέσματα του «τρεξίματος» συγκεντρωτικά. Στο πρώτο kml αρχείο απεικονίζεται η περιοχή που μελετάται με τα τυχαία σημεία που δημιουργήθηκαν σε πρώτη φάση. Στο δεύτερο kml αρχείο απεικονίζονται οι παραγμένες διαδρομές.

9.4 Αποτελέσματα

Στο κεφάλαιο αυτό παρατίθενται τα αποτελέσματα του «τρεξίματος» της εφαρμογής για την περιοχή γύρω από την Πολυτεχνειούπολη Ζωγράφου.

Από την εφαρμογή του αλγορίθμου για την περιοχή του Ζωγράφου δημιουργήθηκαν τα εξής αρχεία:

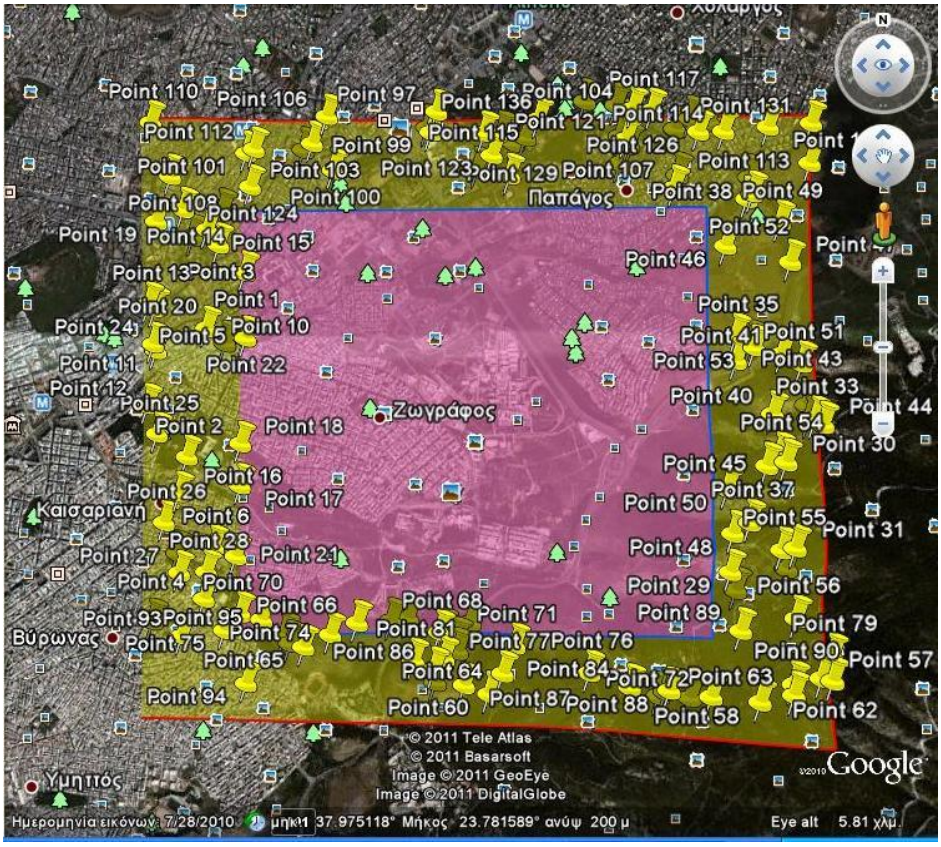
- **Zografos300.kml**, για την απεικόνιση των τυχαίων σημείων που δημιουργήθηκαν.
- **Zografos300-slopes.kml**, για την απεικόνιση των διαδρομών που προέκυψαν από τα τυχαία σημεία.
- **Zografos300-1.txt**, για την καταγραφή των συντεταγμένων των τυχαίων σημείων που δημιουργήθηκαν.
- **Zografos300-elevations.txt**, για την καταγραφή των γεωγραφικών συντεταγμένων και των υψομέτρων των ενδιάμεσων σημείων κάθε διαδρομής που έχει προκύψει με τυχαίο συνδυασμό των τυχαίων σημείων.
- **Zografos300-calories.txt**, για την καταγραφή των υπολογισμένων στοιχείων κάθε τμήματος της εκάστοτε διαδρομής (συντεταγμένες, μήκος, υψομετρική διαφορά, κλίση και θερμιδοκατανάλωση).

Στον επόμενο πίνακα, παρουσιάζονται ενδεικτικά κάποια από τα τυχαία σημεία από το αρχείο **Zografos300-1.txt**, οι γεωγραφικές τους συντεταγμένες και η υποπεριοχή που βρίσκονται - left (1), right (2), up (4), down (3) - για την περιοχή του Ζωγράφου.

N	φ (°)	λ (°)	Side
1	37.983166106073	23.763037712604	left
2	37.97160668293	23.75840702209	left
…			
29	37.964796520944	23.80100387148	right
30	37.971437726078	23.804925597688	right
…			
57	37.95899274213	23.806957782658	down
58	37.957535577019	23.802562284448	down
…			
97	37.993169715165	23.778385573556	up
98	37.992291373171	23.799795948121	up
…			

Πίνακας 9.2: Δείγμα τυχαίων σημείων από το αρχείο Zografos300-1.txt

Στον ακόλουθο χάρτη απεικονίζονται στο Google Earth τα τυχαία σημεία που δημιουργήθηκαν, όπως προέκυψαν από το αρχείο **Zografos300.kml**. Δημιουργήθηκαν 136 τυχαία σημεία, ανομοιόμορφα κατανεμημένα για κάθε υποπεριοχή, και συγκεκριμένα 28 για κάθε πλευρά αριστερά – δεξιά και 40 για κάθε πλευρά πάνω – κάτω.



Εικόνα 9.4: Απεικόνιση των τυχαίων σημείων για την περιοχή γύρω από την Πολυτεχνειούπολη Ζωγράφου στο Google Earth

Παρακάτω φαίνονται τα αποτελέσματα των διαδρομών που δημιουργήθηκαν με βάση τυχαίους συνδυασμούς των τυχαίων σημείων αντίθετων υποπεριοχών (πάνω – κάτω ή αριστερά - δεξιά) που δημιουργήθηκαν, όπως αυτά προέκυψαν από το αρχείο **Zografos300-slopes.kml**.

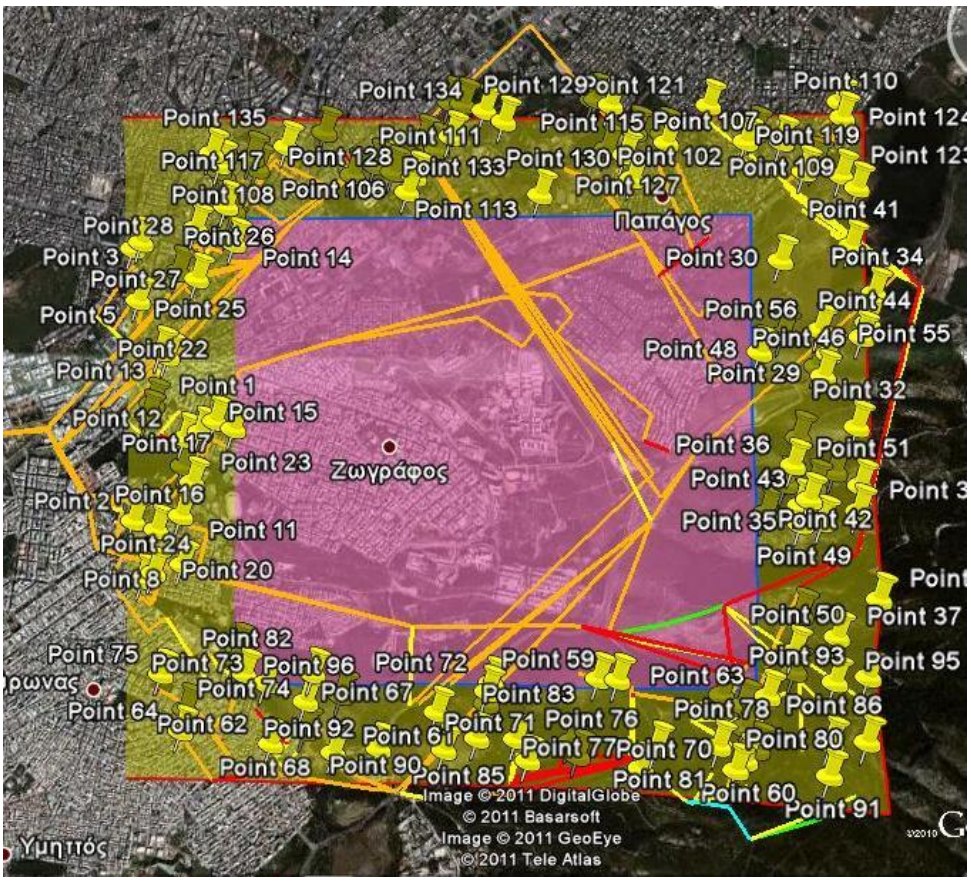
Εικόνα 9.5: Απεικόνιση των διαδρομών για την περιοχή γύρω από την Πολυτεχνειούπολη Ζωγράφου στο Google Earth

Οι διαδρομές κατατάσσονται όπως φαίνεται και στο χάρτη σε πέντε κατηγορίες, ανάλογα με την κλίση τους. Οι πέντε περιοχές κλίσεων απεικονίζονται με διαφορετικά χρώματα, όπως φαίνεται στον παρακάτω πίνακα. Για κάθε περιοχή που μελετάται οι περιοχές κλίσεων μεταβάλλονται. Αυτό συμβαίνει διότι εντοπίζεται η μέγιστη και η ελάχιστη κλίση, και με βάση αυτές τις δύο κλίσεις δημιουργούνται οι πέντε ακόλουθες κλάσεις ώστε η κατανομή να είναι σχετικά ομοιόμορφη.

ΧΡΩΜΑΤΑ	ΠΕΡΙΟΧΗ ΚΛΙΣΕΩΝ (%)
Aqua	-34,8285 έως -24,3218
Green	-24,3218 έως -13,8151
Yellow	-13,8151 έως -3,30848
Orange	-3,30848 έως 7,198187
Red	7,198187 έως 17,70485

Πίνακας 9.9.3: Περιοχές κλίσεων και χρώματα απεικόνισης

Ο επόμενος χάρτης του Google Earth απεικονίζει όλα τα αποτελέσματα (τυχαία σημεία και διαδρομές). Οι διαδρομές όπως είναι φυσικό σε κάποιες περιπτώσεις βγαίνουν και έξω από τα όρια της περιοχής μελέτης.



Εικόνα 9.6: Απεικόνιση όλων των αποτελεσμάτων για την περιοχή γύρω από την Πολυτεχνειούπολη Ζωγράφου στο Google Earth

Στη συνέχεια παρατίθενται ενδεικτικά ορισμένα τμήματα κάποιων από τις διαδρομές που δημιουργήθηκαν, οι γεωγραφικές συντετατγμένες (φ, λ, h) των δύο ακραίων σημείων των τμημάτων, το κεκλιμένο μήκος τους, η υψομετρική διαφορά των δύο ακραίων σημείων κάθε τμήματος, η κλίση του τμήματος αλλά και η κατανάλωση θερμίδων ενός ποδηλάτη στο τμήμα αυτό για την περιοχή του Ζωγράφου. Τα στοιχεία του πίνακα προέρχονται από το αρχείο **Zografos300-calories.txt.**

Όσον αφορά το tag κάθε αποτελέσματος του αρχείου, το Z300 το ορίζει ο χρήστης. Το lr ή ud (μπλε χρώμα όπως φαίνεται και στον παρακάτω πίνακα) δείχνει τη σύνδεση των δύο αρχικών τυχαίων σημείων η οποία μπορεί να είναι είτε left-right είτε up-down, δηλαδή δημιουργούνται διαδρομές μόνο από σημεία απέναντι υποπεριοχών. Με φούξια χρώμα φαίνεται η αρίθμηση του αύξοντα αριθμού της διαδρομής, ενώ με κόκκινο η αρίθμηση του αύξοντα αριθμού των ενδιάμεσων σημείων μίας συγκεκριμένης διαδρομής.

TAG	ϕ1	λ1	h1	ϕ2	λ2
Z300-lr0000-000	37,9755	23,76125	128,6185	37,97478	23,76084
Z300-lr0000-001	37,97478	23,76084	127,8617	37,97492	23,7605
Z300-lr0001-000	37,96959	23,75538	124,1146	37,96919	23,75741
Z300-lr0001-001	37,96919	23,75741	130,6672	37,96861	23,75723
Z300-ud0039-010	37,99902	23,78443	181,8585	37,99178	23,79194
Z300-ud0039-011	37,99178	23,79194	216,4063	37,99155	23,79187

h2	L	dH	Slope (%)	Calories (Calorie/min)
127,8617	87,66595	-0,75679	-0,86328	0
127,4546	33,67633	-0,40713	-1,20897	0
130,6672	183,9369	6,552589	3,563166	1,463194
130,3909	66,29401	-0,27632	-0,41681	0,073414
216,4063	1040,294	34,54779	3,321575	1,378832
217,3533	26,2772	0,947021	3,604747	1,477714

Πίνακας 9.9.4: Τα αποτελέσματα όπως προέκυψαν από την εφαρμογή του αλγορίθμου για την περιοχή του Ζωγράφου

10

ΥΠΟΛΟΓΙΣΜΟΣ ΘΕΡΜΙΔΙΚΗΣ ΚΑΤΑΝΑΛΩΣΗΣ ΤΥΧΑΙΩΝ ΠΟΔΗΛΑΤΙΚΩΝ ΔΙΑΔΡΟΜΩΝ ΣΤΟ ΛΕΚΑΝΟΠΕΔΙΟ ΤΗΣ ΑΘΗΝΑΣ

10.1 Περιοχή Υπολογισμών

Η δεύτερη περιοχή που επιλέχθηκε για να υλοποιηθεί η εφαρμογή είναι εκείνη που περικλείει το λεκανοπέδιο της Αθήνας.

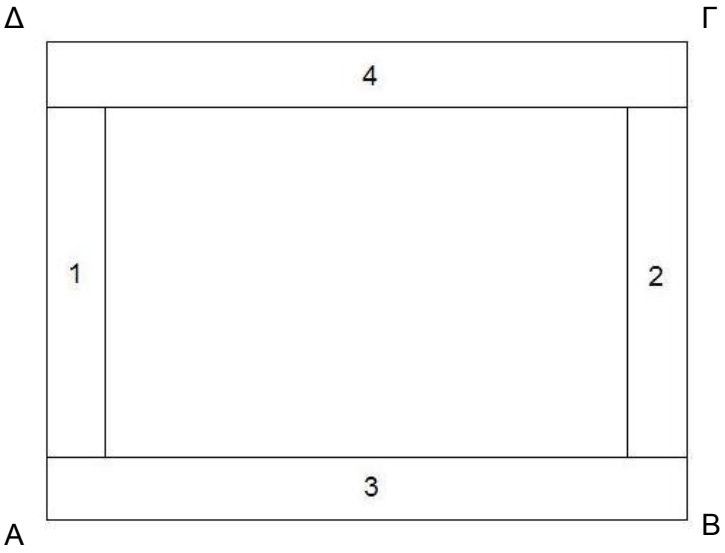
Αυτή η περιοχή μελέτης, συνδυάζει ήπιο αλλά και πιο έντονο ανάγλυφο, με έντονες υψομετρικές διαφορές και μεγάλες κλίσεις δρόμων. Αυτός είναι και ο βασικός λόγος που επιλέχθηκε να μελετηθεί η περιοχή αυτή, καθώς καλύπτει πολλές μικρότερες περιοχές με διαφορετικά γεωγραφικά και πολεοδομικά χαρακτηριστικά.

Οι συντεταγμένες των κορυφών της περιοχής εφαρμογής του αλγορίθμου είναι:

(φ, λ)	
Α (37.959873, 23.691066)	Γ (38.000941, 23.764758)
Β (37.959873, 23.764758)	Δ (38.000941, 23.691066)

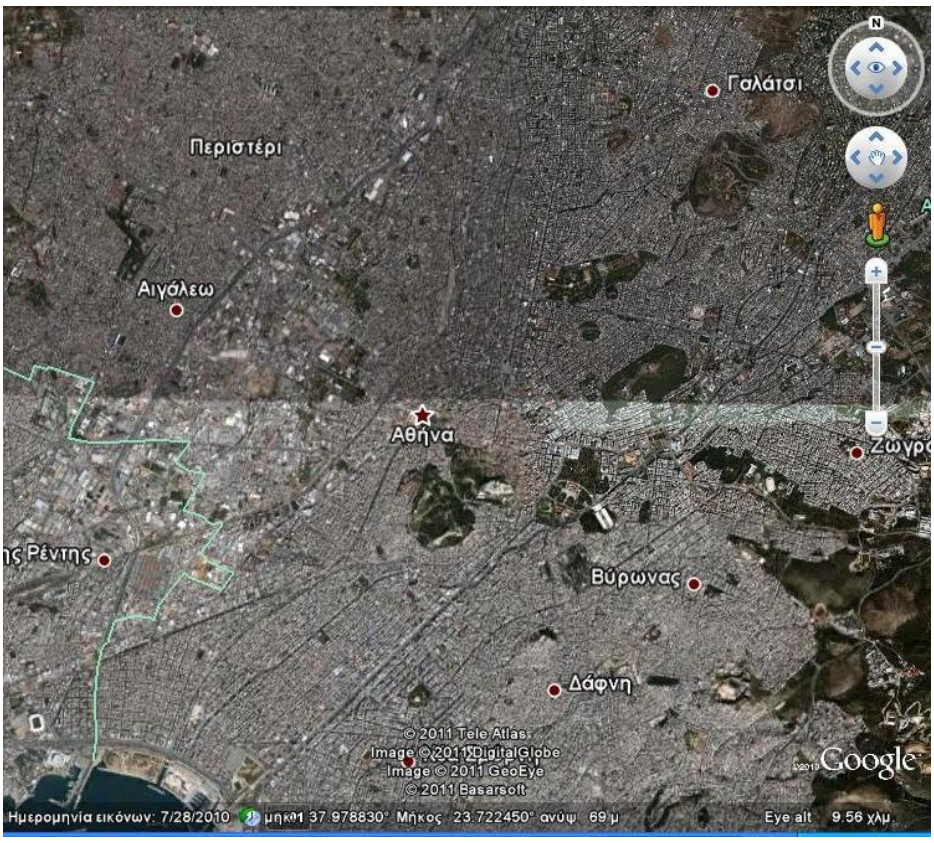
Πίνακας 10.1: Συντεταγμένες κορυφών της πρώτης περιοχής μελέτης

Στην ακόλουθη εικόνα φαίνονται οι κορυφές και οι τέσσερις υποπεριοχές της περιοχής μελέτης, με βάση τις οποίες υπολογίστηκαν τα τυχαία σημεία.

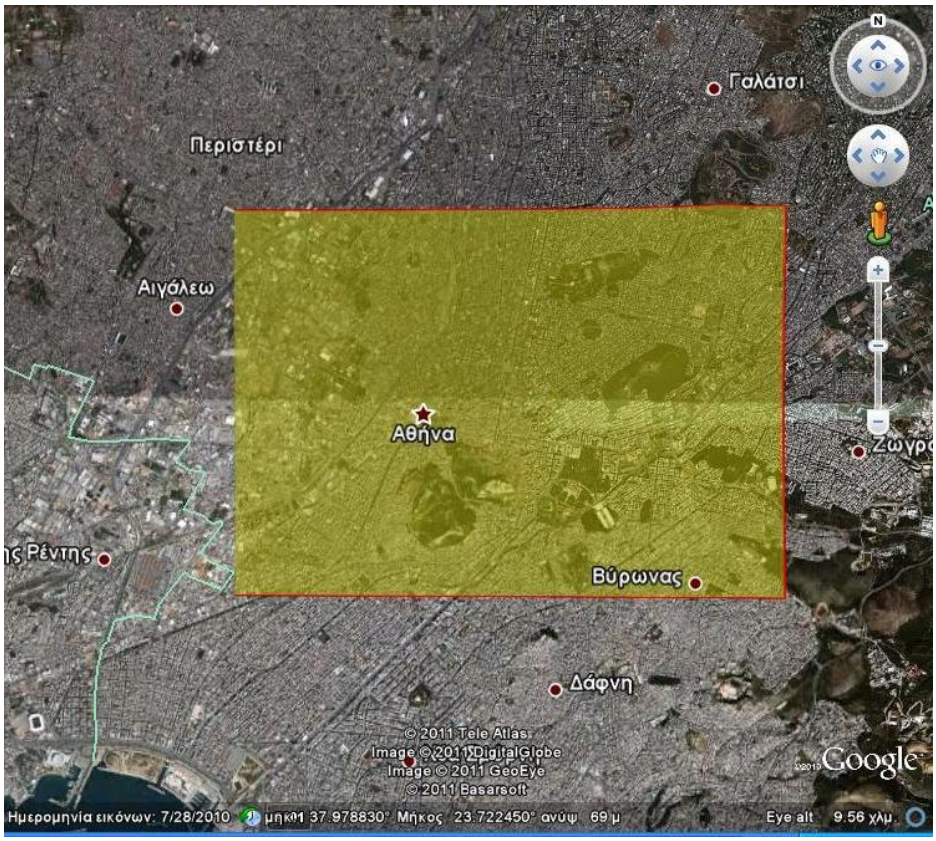


Εικόνα 10.1: Κορυφές και υποπεριοχές της πρώτης περιοχής μελέτης

Στη συνέχεια ακολουθεί χάρτης απεικόνισης στο Google Earth της υπό μελέτη περιοχής.



Εικόνα 10.2: Περιοχή μελέτης του λεκανοπεδίου της Αθήνας



Εικόνα 10.3: Οριοθετημένη περιοχή μελέτης του λεκανοπεδίου της Αθήνας

10.2 Παράμετροι Υπολογισμών

Οι βασικές παράμετροι υπολογισμών της εφαρμογής όπως αναφέρθηκαν και στο παραπάνω κεφάλαιο είναι οι εξής:

- Οι γεωγραφικές συντεταγμένες των δύο ακραίων σημείων που οριοθετούν την περιοχή ενδιαφέροντος. Αυτές μεταβάλλονται σύμφωνα με την περιοχή αλλά και την έκταση αυτής στην οποία θα επιλέξει να υλοποιήσει την εφαρμογή ο χρήστης.
- Η απόσταση που ορίζει ο χρήστης μεταξύ των τυχαίων σημείων που θα δημιουργηθούν στην περιοχή, η οποία θα δώσει την πυκνότητα και των αριθμό των τυχαίων σημείων στις τέσσερις υποπεριοχές της αρχικής περιοχής.
- Οι γεωγραφικές συντεταγμένες φ και λ αρχής και τέλους κάθε πιθανής διαδρομής αποτελούν στη συνέχεια παραμέτρους για τον υπολογισμό των δυνατών διαδρομών και των ενδιάμεσων σημείων αυτών.
- Οι γεωγραφικές συντεταγμένες φ και λ των ενδιάμεσων σημείων των διαδρομών (σημείων αρχής και τέλους των τμημάτων των διαδρομών) αποτελούν με τη σειρά τους παραμέτρους για τον υπολογισμό των υψομέτρων των σημείων αυτών.
- Η κλίση των τμημάτων κάθε διαδρομής, η οποία χρησιμοποιείται για τον υπολογισμό της θερμιδοκατανάλωσης του ποδηλάτη.

10.3 Πραγματοποίηση Υπολογισμών

Για την υλοποίηση των υπολογισμών, κρίθηκε αναγκαία η εφαρμογή τυχαίας καθυστέρησης για τους υπολογισμούς ώστε να μην υπάρχει συστηματικότητα στην κλήση των Google Direction και Elevations API και να μην διακόψει η Google την παροχή των δύο υπηρεσιών.

Αφού τελειοποιήθηκε ο αλγόριθμος, τέθηκε το σύστημα σε λειτουργία αφήνοντάς το να «τρέξει» για την ευρύτερη περιοχή της Πολυτεχνειούπολης. Μετά το τέλος του «τρεξίματος», εξήχθησαν τρία txt και δύο kml αρχεία με τα αποτελέσματα που προέκυψαν από την υλοποίηση του κώδικα για την συγκεκριμένη περιοχή. Το τελικό txt αρχείο περιέχει όλα τα αποτελέσματα του «τρεξίματος» συγκεντρωτικά. Στο πρώτο kml αρχείο απεικονίζεται η περιοχή που μελετάται με τα τυχαία σημεία που δημιουργήθηκαν σε πρώτη φάση. Στο δεύτερο kml αρχείο απεικονίζονται οι παραγμένες διαδρομές.

10.4 Αποτελέσματα

Στο κεφάλαιο αυτό παρατίθενται τα αποτελέσματα του «τρεξίματος» της εφαρμογής για την περιοχή γύρω από την Πολυτεχνειούπολη Ζωγράφου.

Από την εφαρμογή του αλγορίθμου για την περιοχή του Ζωγράφου δημιουργήθηκαν τα εξής αρχεία:

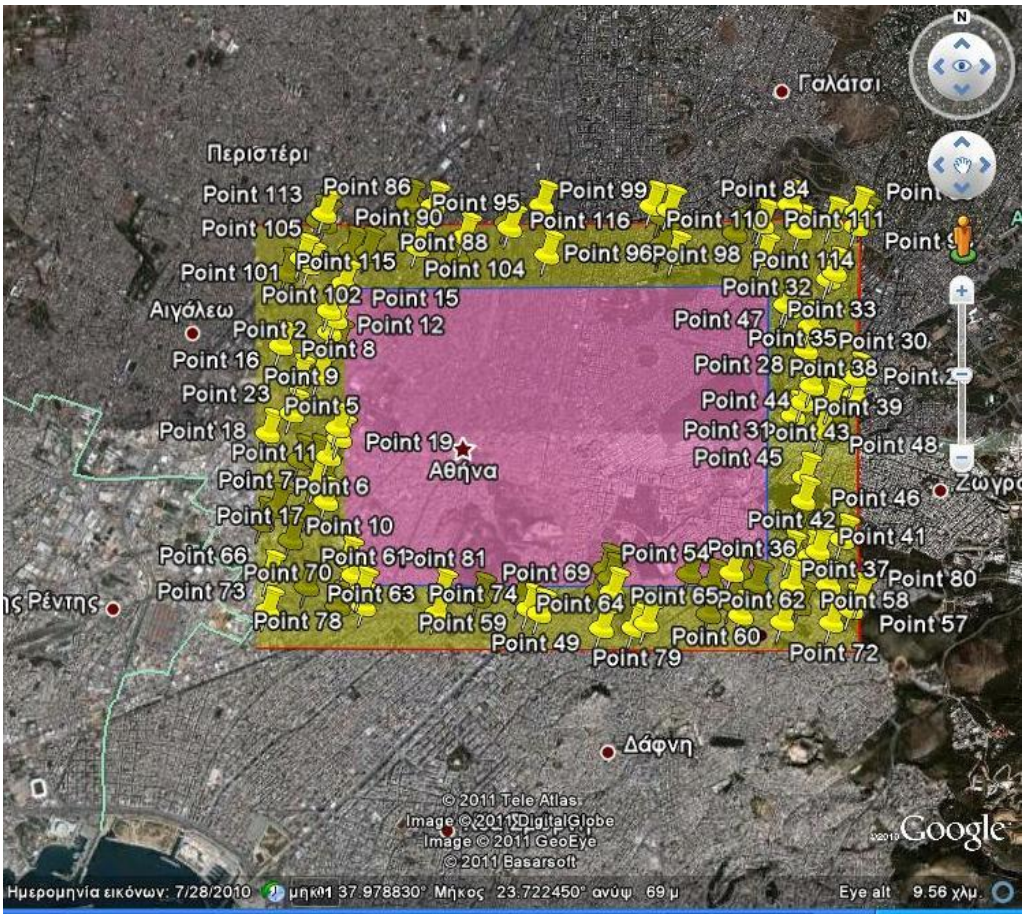
- **Athens400.kml**, για την απεικόνιση των τυχαίων σημείων που δημιουργήθηκαν.
- **Athens400-slopes.kml**, για την απεικόνιση των διαδρομών που προέκυψαν από τα τυχαία σημεία.
- **Athens400-1.txt**, για την καταγραφή των συντεταγμένων των τυχαίων σημείων που δημιουργήθηκαν.
- **Athens400-elevations.txt**, για την καταγραφή των γεωγραφικών συντεταγμένων και των υψομέτρων των ενδιάμεσων σημείων κάθε διαδρομής που έχει προκύψει με τυχαίο συνδυασμό των τυχαίων σημείων.
- **Athens400-calories.txt**, για την καταγραφή των υπολογισμένων στοιχείων κάθε τμήματος της εκάστοτε διαδρομής (συντεταγμένες, μήκος, υψομετρική διαφορά, κλίση και θερμιδοκατανάλωση).

Στον επόμενο πίνακα, παρουσιάζονται ενδεικτικά κάποια από τα τυχαία σημεία από το αρχείο **Athens400-1.txt**, οι γεωγραφικές τους συντεταγμένες και η υποπεριοχή που βρίσκονται - left (1), right (2), up (4), down (3) - για την περιοχή του Ζωγράφου.

N	φ (°)	λ (°)	Side
1	37.976202391025	23.697719601078	left
2	37.988535807268	23.699521424944	left
…			
25	37.969628218601	23.756332421161	right
26	37.978682772323	23.760298099769	right
…			
49	37.959989989069	23.732799738138	down
50	37.964835393259	23.733807114394	down
…			
84	37.998611120442	23.752057783874	up
85	37.996217038634	23.704210301144	up
…			

Πίνακας 10.2: Δείγμα τυχαίων σημείων από το αρχείο Athens400-1.txt

Στον ακόλουθο χάρτη απεικονίζονται στο Google Earth τα τυχαία σημεία που δημιουργήθηκαν, όπως προέκυψαν από το αρχείο **Athens400.kml**. Δημιουργήθηκαν 118 τυχαία σημεία, ανομοιόμορφα κατανεμημένα για κάθε υποπεριοχή, και συγκεκριμένα 24 για κάθε πλευρά αριστερά – δεξιά και 35 για κάθε πλευρά πάνω – κάτω.



Εικόνα 10.4: Απεικόνιση των τυχαίων σημείων για την περιοχή του λεκανοπεδίου της Αθήνας στο Google Earth

Παρακάτω φαίνονται τα αποτελέσματα των διαδρομών που δημιουργήθηκαν με βάση τυχαίους συνδυασμούς των τυχαίων σημείων αντίθετων υποπεριοχών (πάνω – κάτω ή αριστερά - δεξιά) που δημιουργήθηκαν, όπως αυτά προέκυψαν από το αρχείο **Athens400-slopes.kml**.

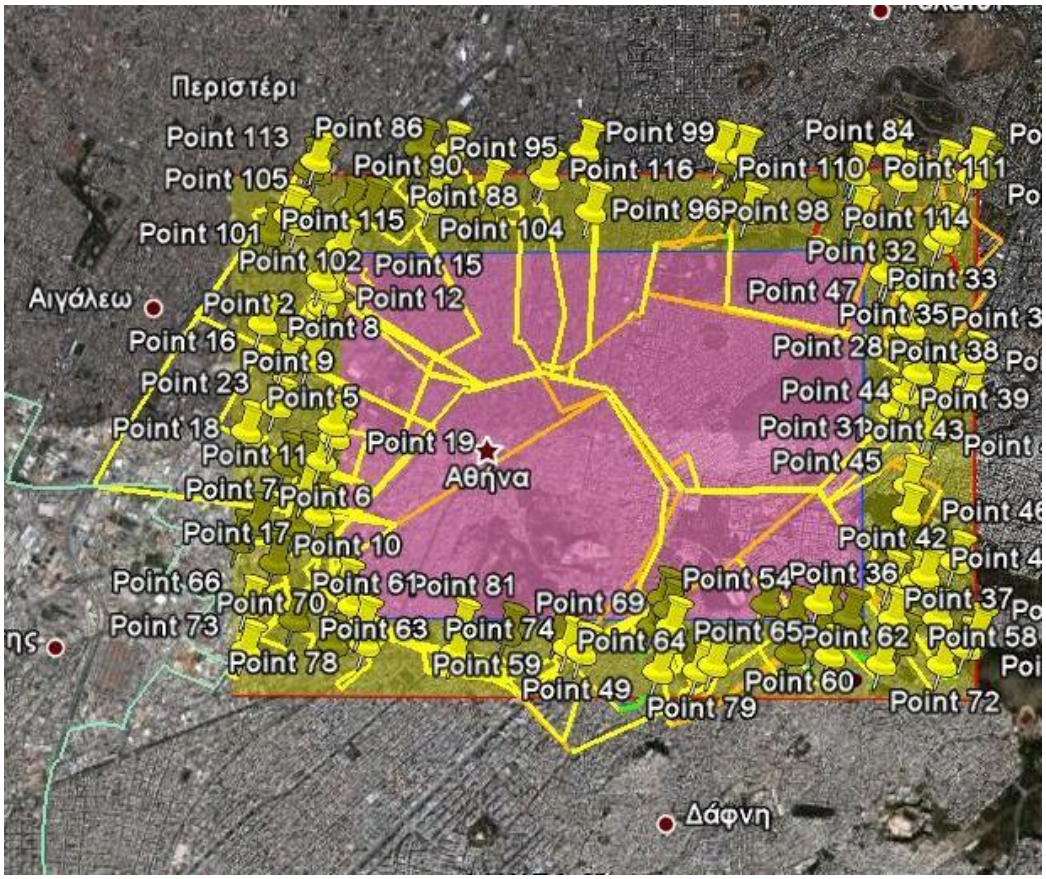
Εικόνα 10.5: Απεικόνιση των διαδρομών για την περιοχή του λεκανοπεδίου της Αθήνας στο Google Earth

Οι διαδρομές κατατάσσονται όπως φαίνεται και στο χάρτη σε πέντε κατηγορίες, ανάλογα με την κλίση τους. Οι πέντε περιοχές κλίσεων απεικονίζονται με διαφορετικά χρώματα, όπως φαίνεται στον παρακάτω πίνακα. Για κάθε περιοχή που μελετάται οι περιοχές κλίσεων μεταβάλλονται. Αυτό συμβαίνει διότι εντοπίζεται η μέγιστη και η ελάχιστη κλίση, και με βάση αυτές τις δύο κλίσεις δημιουργούνται οι πέντε ακόλουθες κλάσεις ώστε η κατανομή να είναι σχετικά ομοιόμορφη.

ΧΡΩΜΑΤΑ	ΠΕΡΙΟΧΗ ΚΛΙΣΕΩΝ (%)
Aqua	-18,3614 έως 3,0659
Green	3,0659 έως 6,1318
Yellow	6,1318 έως 9,1977
Orange	9,1977 έως 12,264
Red	12,264 έως 15,329

Πίνακας 10.3: Περιοχές κλίσεων και χρώματα απεικόνισης

Ο επόμενος χάρτης του Google Earth απεικονίζει όλα τα αποτελέσματα (τυχαία σημεία και διαδρομές). Οι διαδρομές όπως είναι φυσικό σε κάποιες περιπτώσεις βγαίνουν και έξω από τα όρια της περιοχής μελέτης.



Εικόνα 10.6: Απεικόνιση όλων των αποτελεσμάτων για την περιοχή του λεκανοπεδίου της Αθήνας στο Google Earth

Στη συνέχεια παρατίθενται ενδεικτικά ορισμένα τμήματα κάποιων από τις διαδρομές που δημιουργήθηκαν, οι γεωγραφικές συντετατγμένες (φ, λ, h) των δύο ακραίων σημείων των τμημάτων, το κεκλιμένο μήκος τους, η υψομετρική διαφορά των δύο ακραίων σημείων κάθε τμήματος, η κλίση του τμήματος αλλά και η κατανάλωση θερμίδων ενός ποδηλάτη στο τμήμα αυτό για την περιοχή της Αθήνας. Τα στοιχεία του πίνακα προέρχονται από το αρχείο **Athens400-calories.txt.**

Όσον αφορά το tag κάθε αποτελέσματος του αρχείου, το ATH400 το ορίζει ο χρήστης. Το lr ή ud (μπλε χρώμα όπως φαίνεται και στον παρακάτω πίνακα) δείχνει τη σύνδεση των δύο αρχικών τυχαίων σημείων η οποία μπορεί να είναι είτε left-right είτε up-down, δηλαδή δημιουργούνται διαδρομές μόνο από σημεία απέναντι υποπεριοχών. Με φούξια χρώμα φαίνεται η αρίθμηση του αύξοντα αριθμού της διαδρομής, ενώ με κόκκινο η αρίθμηση του αύξοντα αριθμού των ενδιάμεσων σημείων μίας συγκεκριμένης διαδρομής.

TAG	φ1	λ1	h1	φ2	λ2
ATH400-lr0000-000	37,97575	23,69768	23,03851	37,97579	23,69678
ATH400-lr0000-001	37,97579	23,69678	23,23492	37,97788	23,69842
ATH400-lr0001-000	37,98889	23,69912	35	37,98961	23,69788
ATH400-lr0001-001	37,98961	23,69788	33,94451	37,99227	23,70011
ATH400-ud0034-017	37,99601	23,74077	106,4738	37,99727	23,73985
ATH400-ud0034-018	37,99727	23,73985	105,611	37,99709	23,73966

h2	L	dH	Slope (%)	Calories (Calorie/min)
23,23492	79,20029	0,196411	0,247993	0,30556
25,21494	273,0972	1,980022	0,725031	0,472138
33,94451	135,1053	-1,05549	-0,78124	0
37,22283	354,3415	3,278316	0,925199	0,542035
105,611	161,5282	-0,86276	-0,53413	0
105,1724	26,03662	-0,43861	-1,68468	0

Πίνακας 10.4: Τα αποτελέσματα όπως προέκυψαν από την εφαρμογή του αλγορίθμου για την περιοχή του Ζωγράφου

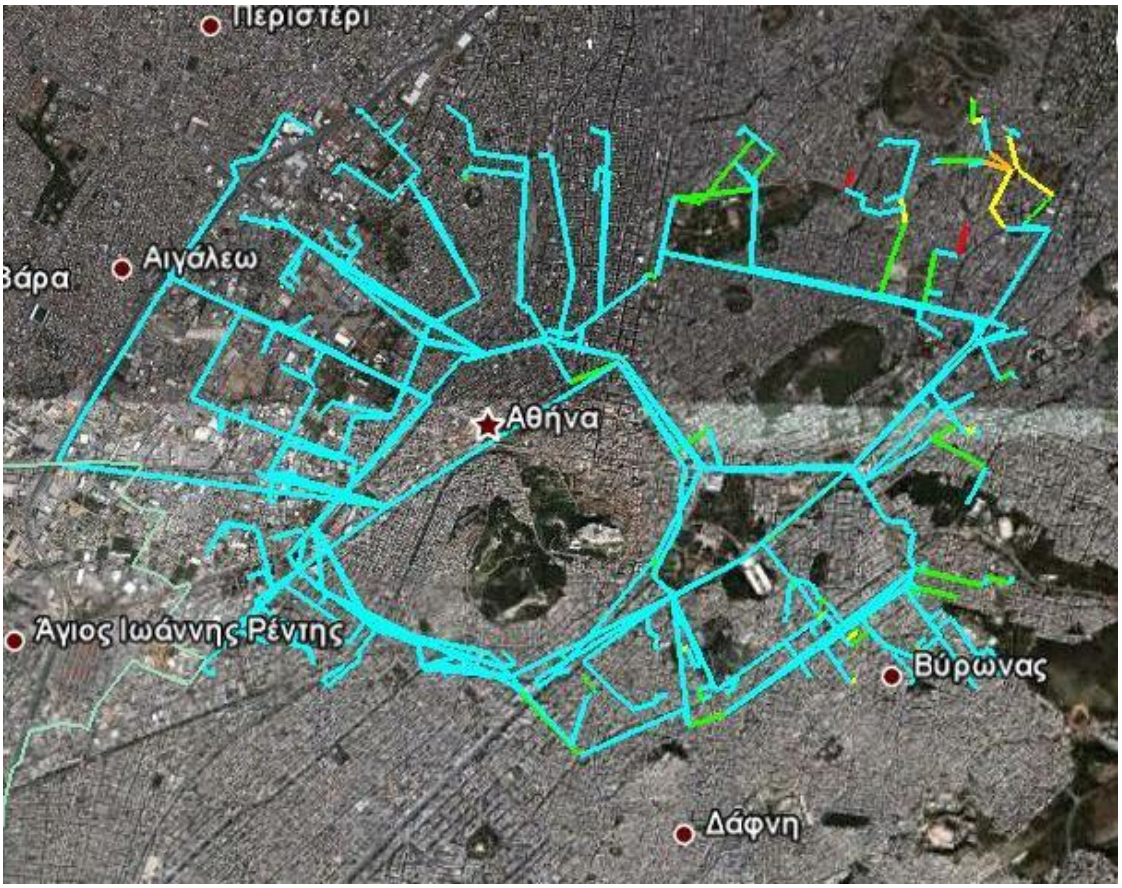
10.5 Αποτελέσματα Θερμιδικής Κατανάλωσης

Με βάση την κατανάλωση θερμίδων του ποδηλάτη που υπολογίστηκε κατά μήκος κάθε τμήματος της εκάστοτε διαδρομής, δημιουργήθηκε ένα kml αρχείο για την απεικόνισή της στο Google Earth για το **λεκανοπέδιο της Αθήνας**.

Οι διαδρομές κατατάσσονται όπως φαίνεται και στο χάρτη σε πέντε κατηγορίες, ανάλογα με την θερμιδική κατανάλωση του ποδηλάτη για κάθε τμήμα κάθε διαδρομής. Οι πέντε περιοχές θερμιδικής κατανάλωσης απεικονίζονται με διαφορετικά χρώματα, όπως φαίνεται στον παρακάτω πίνακα. Οι περιοχές θερμιδικής κατανάλωσης είναι ανεξάρτητες των αντίστοιχων των κλίσεων και δημιουργούνται εκ νέου. Συγκεκριμένα, εντοπίζεται η μέγιστη και η ελάχιστη θερμιδική κατανάλωση και με βάση αυτές τις δύο τιμές δημιουργούνται οι πέντε ακόλουθες κλάσεις ώστε η κατανομή να είναι σχετικά ομοιόμορφη.

ΧΡΩΜΑΤΑ	ΠΕΡΙΟΧΗ ΘΕΡΜΙΔΙΚΗΣ ΚΑΤΑΝΑΛΩΣΗΣ (Calorie/min)
Aqua	0 έως 1,114379777
Green	1,114379777 έως 2,228759554
Yellow	2,228759554 έως 3,343139331
Orange	3,343139331 έως 4,457519108
Red	4,457519108 έως 5,571898885

Εικόνα 10.7: Περιοχές θερμιδικής κατανάλωσης και χρώματα απεικόνισης



Εικόνα 10.8: Απεικόνιση θερμιδικής κατανάλωσης ποδηλάτη στην περιοχή της Αθήνας

11

ΥΠΟΛΟΓΙΣΜΟΣ ΘΕΡΜΙΔΙΚΗΣ ΚΑΤΑΝΑΛΩΣΗΣ ΤΥΧΑΙΩΝ ΠΟΔΗΛΑΤΙΚΩΝ ΔΙΑΔΡΟΜΩΝ ΣΤΗ ΜΗΤΡΟΠΟΛΙΤΙΚΗ ΠΕΡΙΟΧΗ ΤΟΥ SAN FRANSISCO

11.1 Περιοχή Υπολογισμών

Η δεύτερη περιοχή που επιλέχθηκε για να υλοποιηθεί η εφαρμογή είναι εκείνη της μητροπολιτικής περιοχής του San Fransisco της Καλιφόρνια.

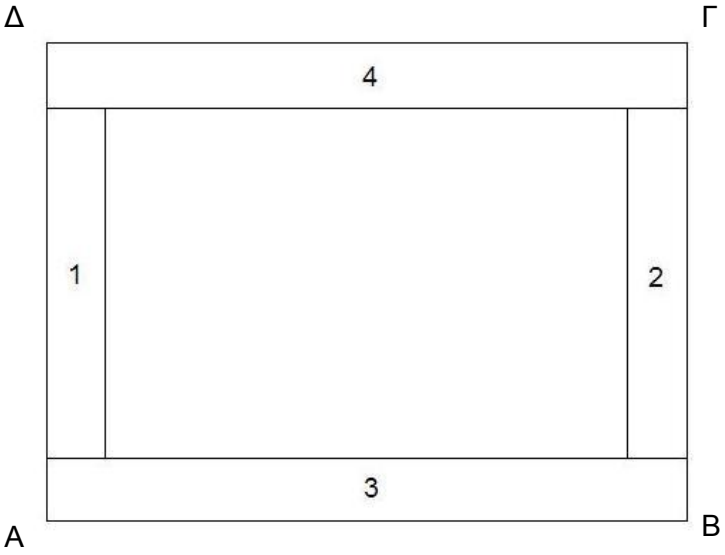
Η περιοχή αυτή διαθέτει σχετικά έντονο ανάγλυφο, με έντονες υψομετρικές διαφορές και μεγάλες κλίσεις δρόμων. Η διαφορά με την πρώτη περιοχή που επιλέχθηκε είναι ότι το San Fransisco διαθέτει ένα ορθά διατεταγμένο οδικό δίκτυο χωρίς καμπύλα τμήματα, γεγονός που παίζει ρόλο στη διεξαγωγή ρεαλιστικότερων και ορθότερων αποτελεσμάτων, καθώς κατά την υλοποίηση της εφαρμογής έχει γίνει η παραδοχή ότι οι διαδρομές που μελετώνται αποτελούνται από ευθύγραμμα και όχι καμπύλα τμήματα, γεγονός που επαληθεύει την υπόθεση στη περίπτωση του San Fransisco. Αυτός είναι και ο βασικός λόγος που επιλέχθηκε να μελετηθεί η περιοχή αυτή, καθώς συνδυάζει έντονο ανάγλυφο με ορθοκανονικό οδικό δίκτυο.

Οι συντεταγμένες των κορυφών της περιοχής εφαρμογής του αλγορίθμου είναι:

(φ, λ)	
Α (37.728190, -122.508540)	Γ (37.777099, -122.380946)
Β (37.728190, -122.380946)	Δ (37.777099, -122.508540)

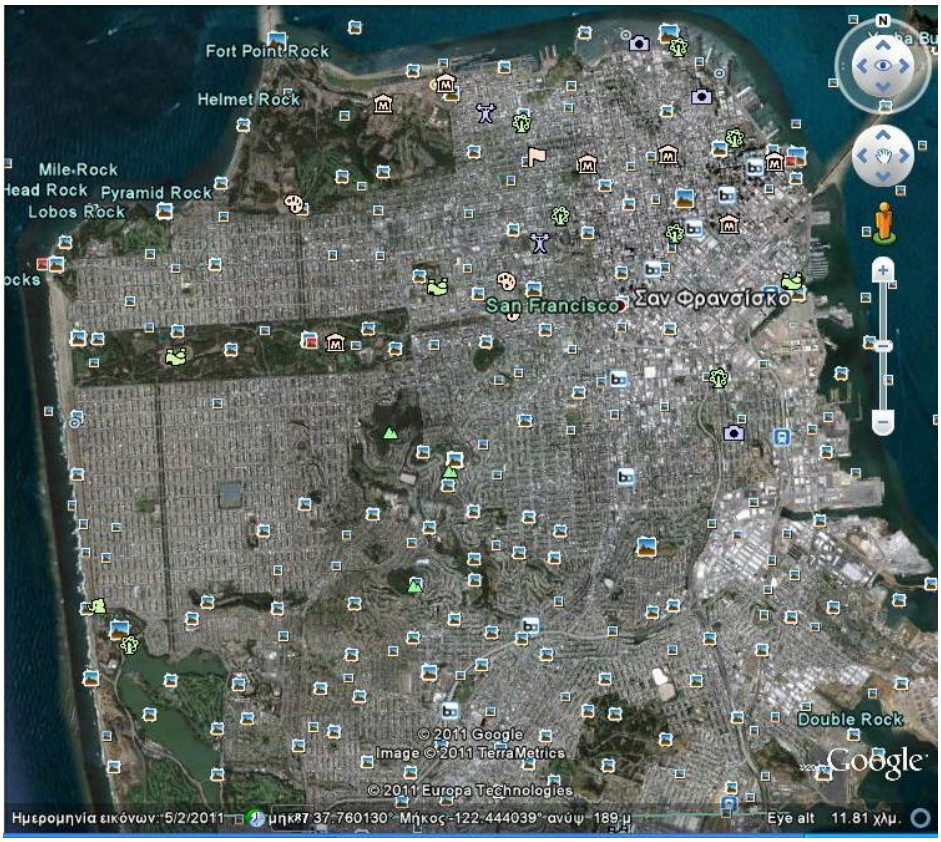
Πίνακας 11.1: Συντεταγμένες κορυφών της δεύτερης περιοχής μελέτης

Στην ακόλουθη εικόνα φαίνονται οι κορυφές και οι τέσσερις υποπεριοχές της περιοχής μελέτης, με βάση τις οποίες υπολογίστηκαν τα τυχαία σημεία.

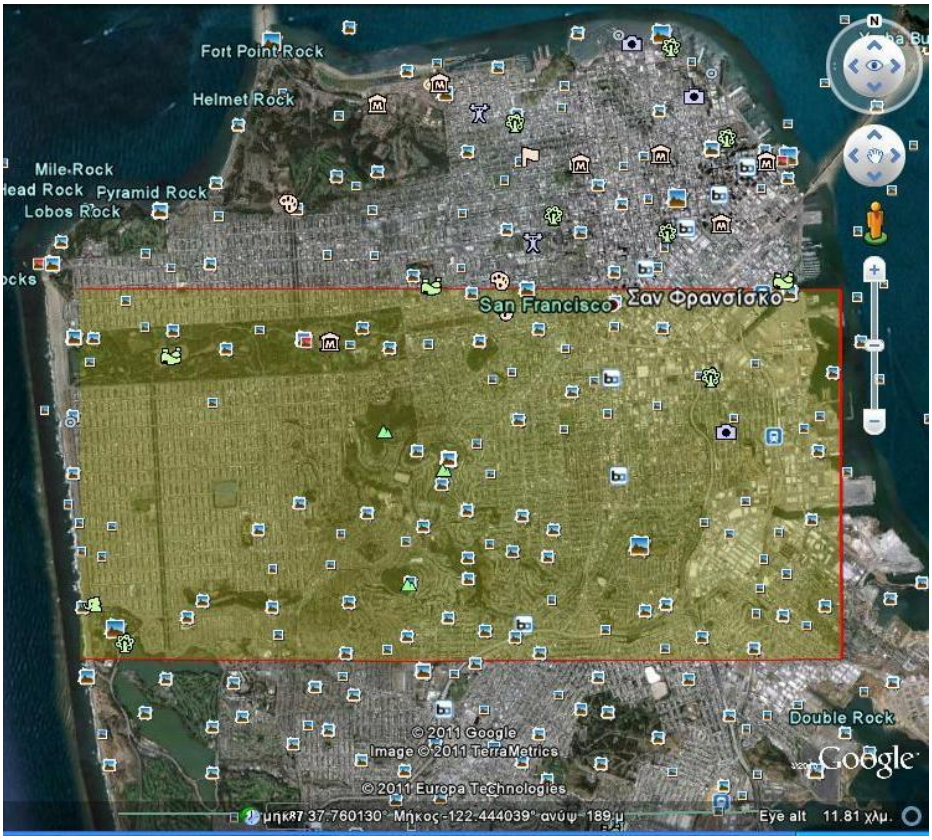


Εικόνα 11.1: Κορυφές και υποπεριοχές της δεύτερης περιοχής μελέτης

Στη συνέχεια ακολουθεί χάρτης απεικόνισης της υπό μελέτη περιοχής στο Google Earth.

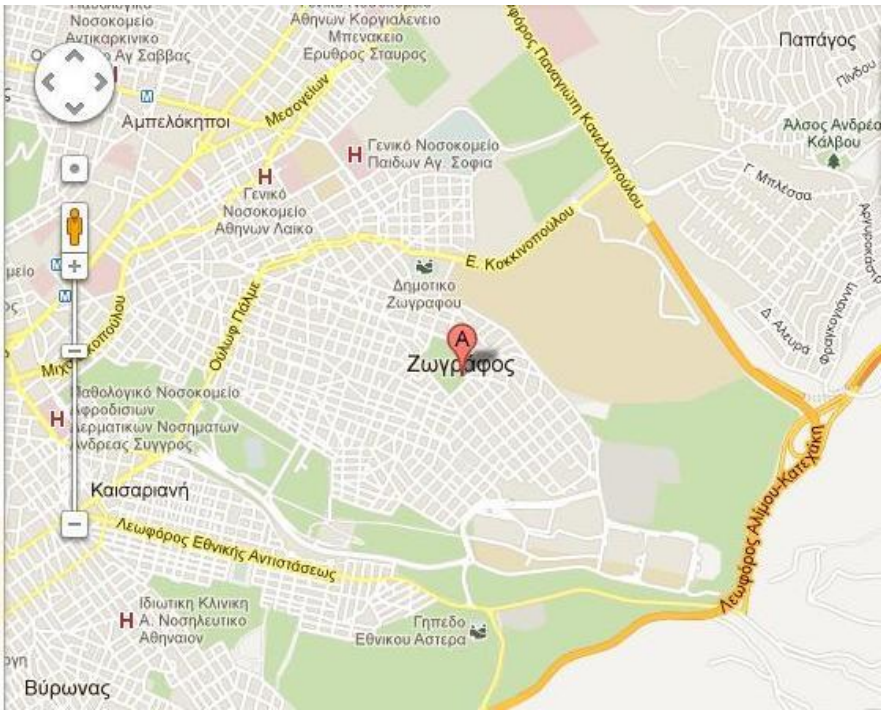


Εικόνα 11.2: Περιοχή μελέτης, San Fransisco

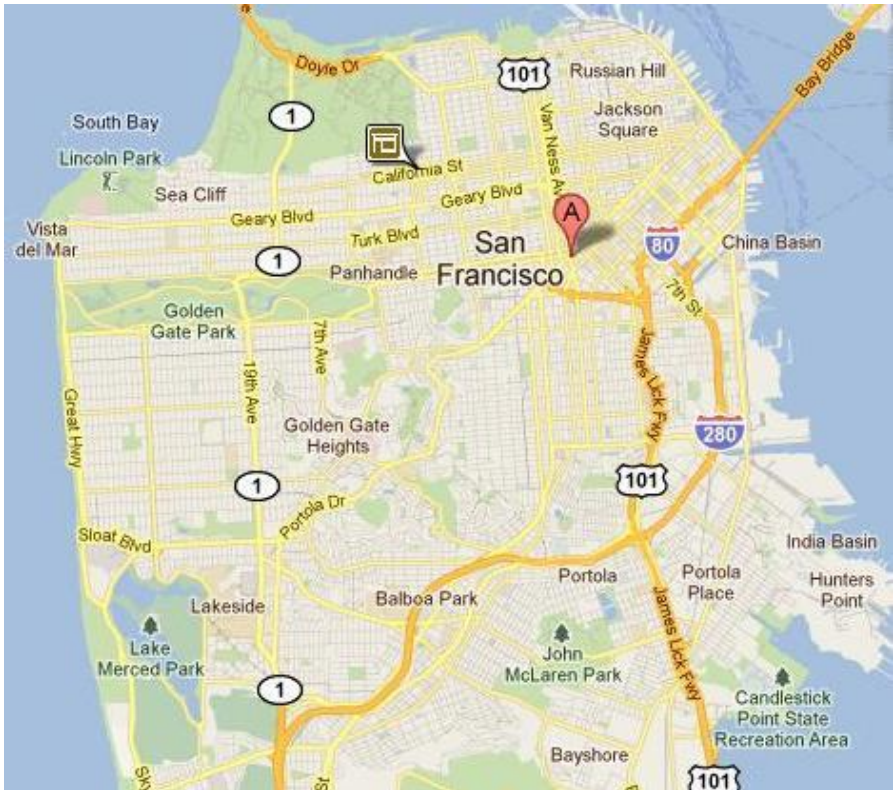


Εικόνα 11.3: Οριοθετημένη περιοχή μελέτης, San Fransisco

Σύγκριση δομής οδικού δικτύου περιοχών Ζωγράφου και San Fransisco:



Εικόνα 11.4: Οδικό δίκτυο στην ευρύτερη περιοχή του Ζωγράφου



Εικόνα 11.5: Οδικό δίκτυο στην μητροπολιτική περιοχή του San Fransisco

11.2 Παράμετροι Υπολογισμών

Οι βασικές παράμετροι υπολογισμών της εφαρμογής αντίστοιχα και για την περιοχή του San Fransisco είναι οι εξής:

- Οι γεωγραφικές συντεταγμένες των δύο ακραίων σημείων που οριοθετούν την περιοχή ενδιαφέροντος. Αυτές μεταβάλλονται σύμφωνα με την περιοχή αλλά και την έκταση αυτής στην οποία θα επιλέξει να υλοποιήσει την εφαρμογή ο χρήστης.
- Η απόσταση που ορίζει ο χρήστης μεταξύ των τυχαίων σημείων που θα δημιουργηθούν στην περιοχή, η οποία θα δώσει την πυκνότητα και των αριθμό των τυχαίων σημείων στις τέσσερις υποπεριοχές της αρχικής περιοχής.
- Οι γεωγραφικές συντεταγμένες φ και λ αρχής και τέλους κάθε πιθανής διαδρομής αποτελούν στη συνέχεια παραμέτρους για τον υπολογισμό των δυνατών διαδρομών και των ενδιάμεσων σημείων αυτών.
- Οι γεωγραφικές συντεταγμένες φ και λ των ενδιάμεσων σημείων των διαδρομών (σημείων αρχής και τέλους των τμημάτων των διαδρομών) αποτελούν με τη σειρά τους παραμέτρους για τον υπολογισμό των υψομέτρων των σημείων αυτών.
- Η κλίση των τμημάτων κάθε διαδρομής, η οποία χρησιμοποιείται για τον υπολογισμό της θερμιδοκατανάλωσης του ποδηλάτη.

11.3 Πραγματοποίηση Υπολογισμών

Για την πραγματοποίηση των αντίστοιχων υπολογισμών στο San Fransisco, εισήχθησαν οι γεωγραφικές συντεταγμένες των δύο σημείων που οριοθετούν την περιοχή.

Το σύστημα τέθηκε σε λειτουργία αφήνοντάς το να «τρέξει» για την ευρύτερη περιοχή του San Fransisco. Μετά το τέλος του «τρεξίματος», εξήχθησαν τρία txt και δύο kml αρχεία με τα αποτελέσματα που προέκυψαν από την υλοποίηση του κώδικα για την συγκεκριμένη περιοχή. Το τελικό txt αρχείο περιέχει όλα τα αποτελέσματα του «τρεξίματος» συγκεντρωτικά. Στο πρώτο kml αρχείο απεικονίζεται η περιοχή που μελετάται με τα τυχαία σημεία που δημιουργήθηκαν σε πρώτη φάση. Στο δεύτερο kml αρχείο απεικονίζονται οι παραγμένες διαδρομές.

11.4 Αποτελέσματα

Στο κεφάλαιο αυτό παρατίθενται τα αποτελέσματα του «τρεξίματος» της εφαρμογής για την περιοχή γύρω από την μητροπολιτική περιοχή του San Fransisco.

Από την εφαρμογή του αλγορίθμου δημιουργήθηκαν τα εξής αρχεία:

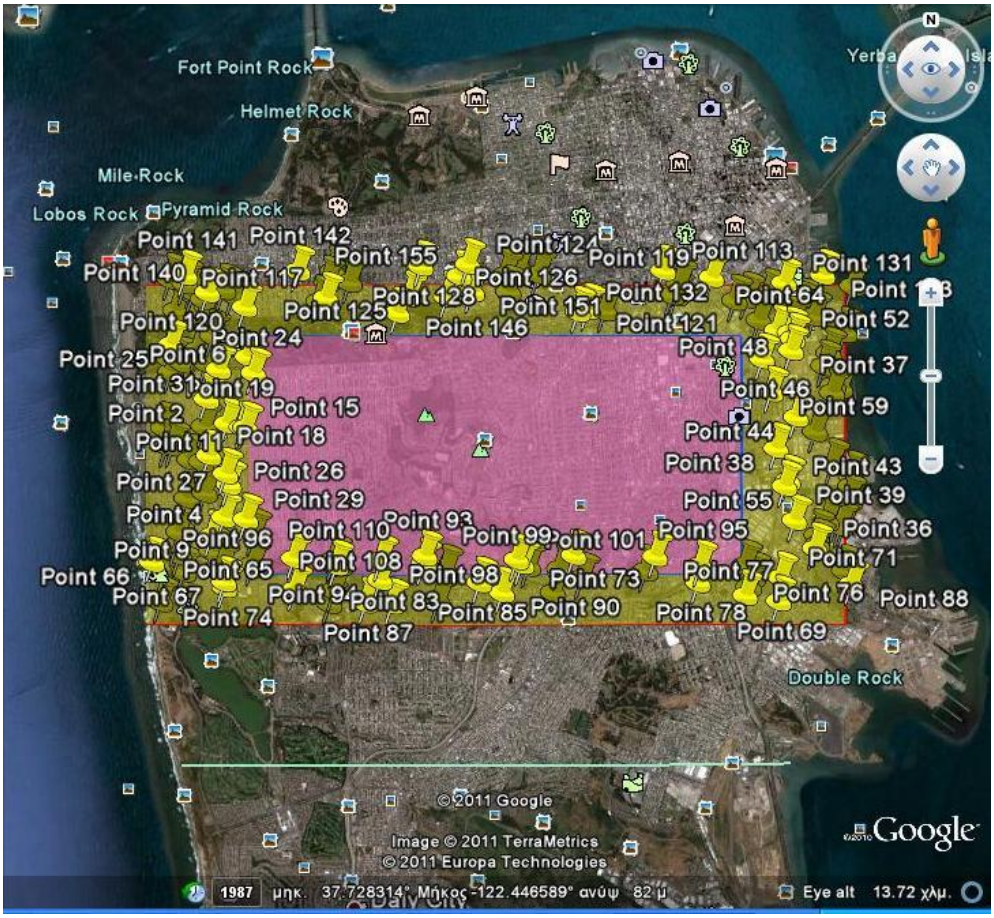
- **SanFransisco500.kml**, για την απεικόνιση των τυχαίων σημείων που δημιουργήθηκαν.
- **SanFransisco500-slopes.kml**, για την απεικόνιση των διαδρομών που προέκυψαν από τα τυχαία σημεία.
- **SanFransisco500-1.txt**, για την καταγραφή των συντεταγμένων των τυχαίων σημείων που δημιουργήθηκαν.
- **SanFransisco500-elevations.txt**, για την καταγραφή των γεωγραφικών συντεταγμένων και των υψομέτρων των ενδιάμεσων σημείων κάθε διαδρομής που έχει προκύψει με τυχαίο συνδυασμό των τυχαίων σημείων.
- **SanFransisco500-calories.txt**, για την καταγραφή των υπολογισμένων στοιχείων κάθε τμήματος της εκάστοτε διαδρομής (συντεταγμένες, μήκος, υψομετρική διαφορά, κλίση και θερμιδοκατανάλωση).

Στον επόμενο πίνακα, παρουσιάζονται ενδεικτικά κάποια από τα τυχαία σημεία, οι γεωγραφικές τους συντεταγμένες και η υποπεριοχή που βρίσκονται - left (1), right (2), up (4), down (3) - για την περιοχή του San Fransisco, από το αρχείο **SanFransisco500-1.txt**.

N	φ (°)	λ (°)	Side
1	37.748647806367	-122.49921735502	left
2	37.757618303601	-122.49829712639	left
…			
33	37.767180145971	-122.38241358856	right
34	37.764248498275	-122.38533584524	right
…			
65	37.730323638073	-122.48225994853	down
66	37.734183291115	-122.50832873148	down
…			
111	37.770864884282	-122.39057953001	up
112	37.770322182301	-122.50398263524	up
…			

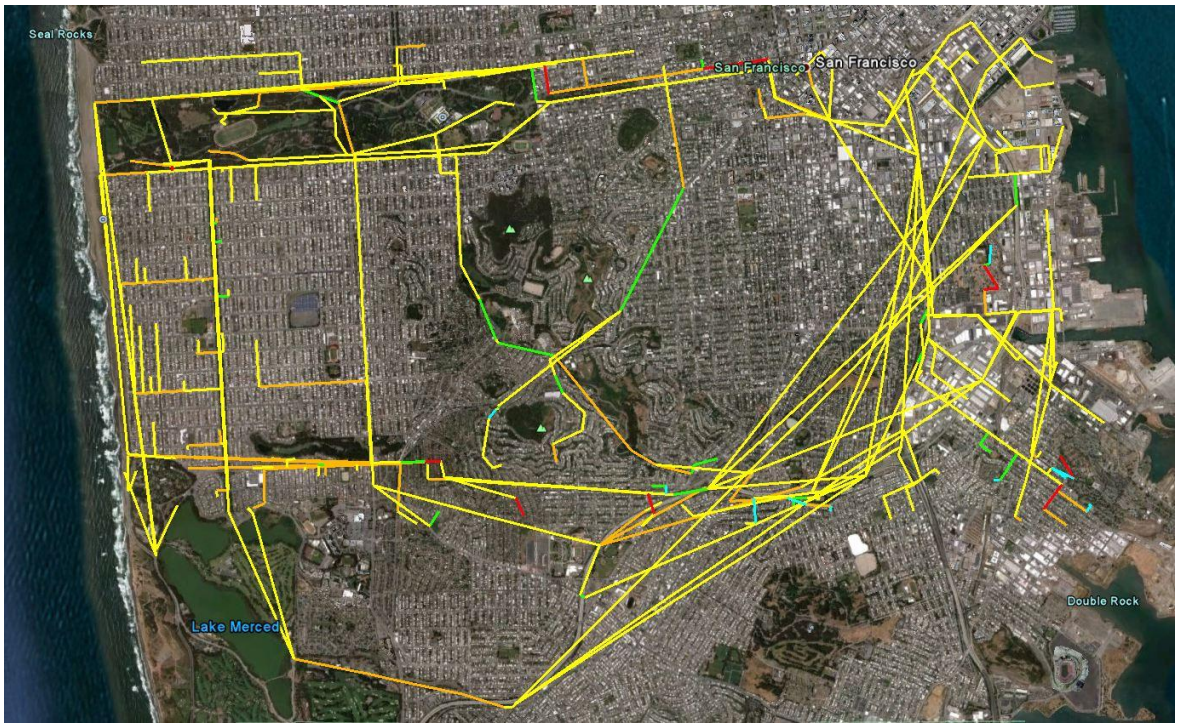
Πίνακας 11.2: Δείγμα τυχαίων σημείων από το αρχείο SanFransisco500-1.txt

Στον ακόλουθο χάρτη απεικονίζονται στο Google Earth τα τυχαία σημεία που δημιουργήθηκαν, όπως αυτά προέκυψαν από το αρχείο **SanFransisco500.kml**. Δημιουργήθηκαν 156 τυχαία σημεία, ανομοιόμορφα κατανεμημένα για κάθε υποπεριοχή, και συγκεκριμένα 36 για κάθε πλευρά αριστερά – δεξιά και 40 για κάθε πλευρά πάνω – κάτω.



Εικόνα 11.6: Απεικόνιση των τυχαίων σημείων για τη μητροποιλιτική περιοχή του San Fransisco στο Google Earth

Παρακάτω φαίνονται οι διαδρομές που δημιουργήθηκαν με βάση τυχαίους συνδυασμούς των τυχαίων σημείων αντίθετων υποπεριοχών (πάνω – κάτω ή αριστερά - δεξιά) που δημιουργήθηκαν, όπως προέκυψαν στο αρχείο **SanFransisco500-slopes.kml**.



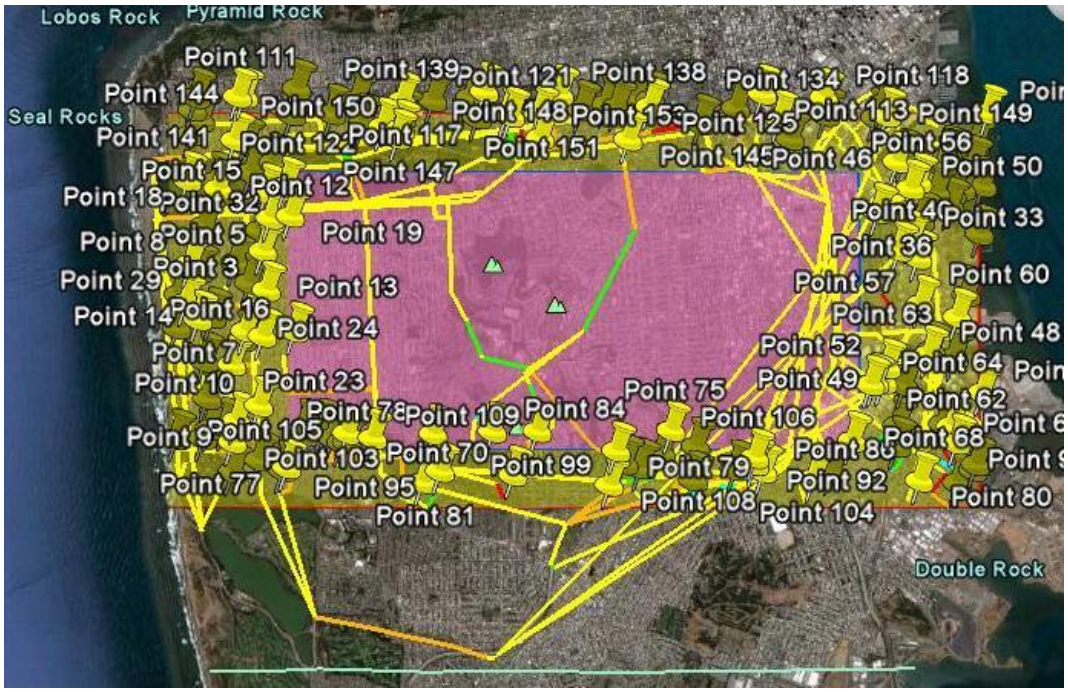
Εικόνα 11.7: Απεικόνιση των διαδρομών για τη μητροπολιτική περιοχή του San Fransisco στο Google Earth

Οι διαδρομές κατατάσσονται όπως φαίνεται και στο χάρτη σε πέντε κατηγορίες, ανάλογα με την κλίση τους. Οι πέντε περιοχές κλίσεων απεικονίζονται με διαφορετικά χρώματα, όπως φαίνεται στον παρακάτω πίνακα. Για κάθε περιοχή που μελετάται οι περιοχές κλίσεων μεταβάλλονται. Αυτό συμβαίνει διότι εντοπίζεται η μέγιστη και η ελάχιστη κλίση, και με βάση αυτές τις δύο κλίσεις δημιουργούνται οι πέντε ακόλουθες κλάσεις ώστε η κατανομή να είναι σχετικά ομοιόμορφη.

ΧΡΩΜΑΤΑ	ΠΕΡΙΟΧΗ ΚΛΙΣΕΩΝ (%)
Aqua	-20,2864 έως -13,6933
Green	-13,6933 έως -7,10006
Yellow	-7,10006 έως -0,50687
Orange	-0,50687 έως 6,08632
Red	6,08632 έως 12,67951

Πίνακας 11.3: Περιοχές κλίσεων και χρώματα απεικόνισης

Ο επόμενος χάρτης του Google Earth απεικονίζει όλα τα αποτελέσματα (τυχαία σημεία και διαδρομές). Οι διαδρομές όπως είναι φυσικό σε κάποιες περιπτώσεις βγαίνουν και έξω από τα όρια της περιοχής μελέτης.



Εικόνα 11.8: Απεικόνιση όλων των αποτελεσμάτων για τη μητροπολιτική περιοχή του San Fransisco στο Google Earth

Στη συνέχεια παρατίθενται ενδεικτικά ορισμένα τμήματα κάποιων από τις διαδρομές που δημιουργήθηκαν από το αρχείο **SanFransisco500-calories.txt**, οι γεωγραφικές συντεταγμένες (φ, λ, h) των δύο ακραίων σημείων των τμημάτων, το κεκλιμένο μήκος τους, η υψομετρική διαφορά των δύο ακραίων σημείων κάθε τμήματος, η κλίση του τμήματος αλλά και η κατανάλωση θερμίδων ενός ποδηλάτη στο τμήμα αυτό για την περιοχή του San Fransisco.

Όσον αφορά το tag κάθε αποτελέσματος του αρχείου, το SF500 το ορίζει ο χρήστης. Το lr ή ud (μπλε χρώμα όπως φαίνεται και στον παρακάτω πίνακα) δείχνει τη σύνδεση των δύο αρχικών τυχαίων σημείων η οποία μπορεί να είναι είτε left-right είτε up-down, δηλαδή δημιουργούνται διαδρομές μόνο από σημεία απέναντι υποπεριοχών. Με φούξια χρώμα φαίνεται η αρίθμηση του αύξοντα αριθμού της διαδρομής, ενώ με κόκκινο η αρίθμηση του αύξοντα αριθμού των ενδιάμεσων σημείων μίας συγκεκριμένης διαδρομής.

TAG	φ1	λ1	h1	φ2	λ2
SF500-lr0000-000	37,75559	-122,499	37,10037	37,75325	-122,499
SF500-lr0000-001	37,75325	-122,499	33,30185	37,75341	-122,495
SF500-lr0001-000	37,7602	-122,504	22,67571	37,76422	-122,504
SF500-lr0001-001	37,76422	-122,504	21,86073	37,76614	-122,46
SF500-ud0045-018	37,77049	-122,48	67,60492	37,77017	-122,495
SF500-ud0045-019	37,77017	-122,495	39,53695	37,77073	-122,496

h2	L	dH	Slope (%)	Calories (Calorie/min)
33,30185	260,234	-3,79852	-1,45971	0
51,52208	322,6662	18,22023	5,649781	2,191825
21,86073	446,9212	-0,81497	-0,18235	0,155286
82,22226	3848,964	60,36152	1,568318	0,766608
39,53695	1326,774	-28,068	-2,11566	0
39,58652	88,81184	0,049564	0,055808	0,23845

Πίνακας 11.4: Τα αποτελέσματα όπως προέκυψαν από την εφαρμογή του αλγορίθμου για την περιοχή του San Fransisco

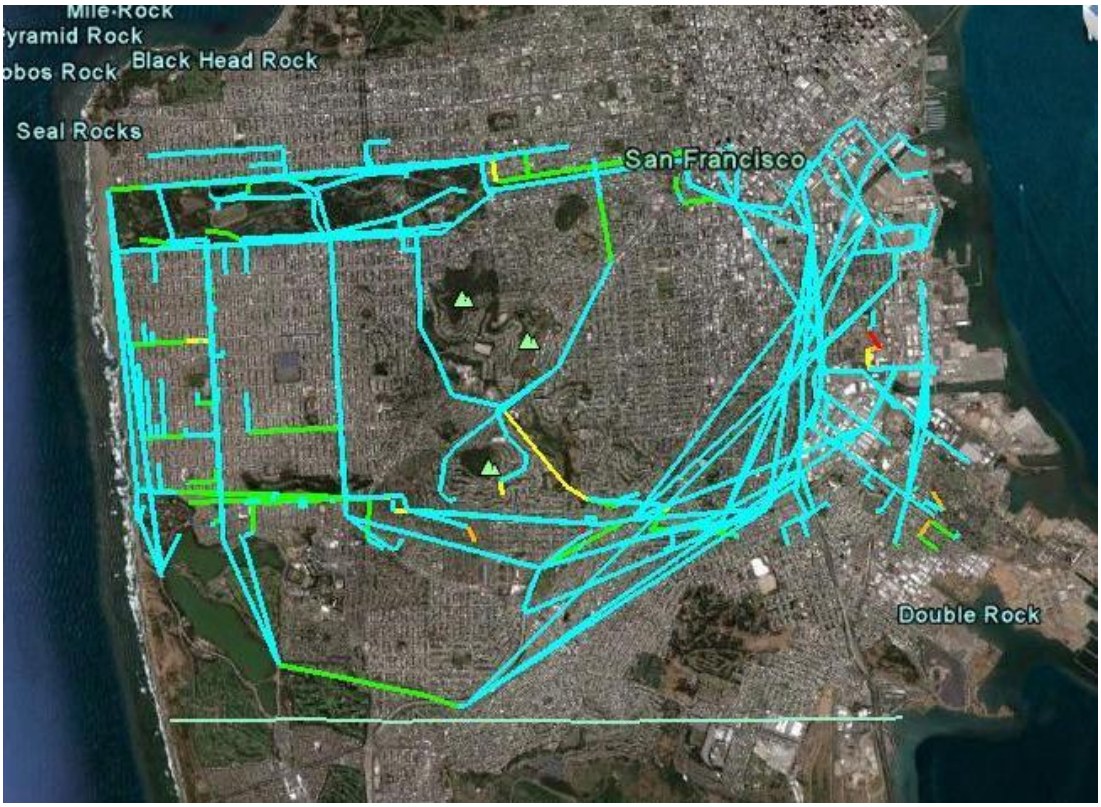
11.5 Αποτελέσματα Θερμιδικής Κατανάλωσης

Με βάση την κατανάλωση θερμίδων του ποδηλάτη που υπολογίστηκε κατά μήκος κάθε τμήματος της εκάστοτε διαδρομής, δημιουργήθηκε ένα kml αρχείο για την απεικόνισή της στο Google Earth για την περιοχή του **San Fransisco**.

Οι διαδρομές κατατάσσονται όπως φαίνεται και στο χάρτη σε πέντε κατηγορίες, ανάλογα με την θερμιδική κατανάλωση του ποδηλάτη για κάθε τμήμα κάθε διαδρομής. Οι πέντε περιοχές θερμιδικής κατανάλωσης απεικονίζονται με διαφορετικά χρώματα, όπως φαίνεται στον παρακάτω πίνακα. Οι περιοχές θερμιδικής κατανάλωσης είναι ανεξάρτητες των αντίστοιχων των κλίσεων και δημιουργούνται εκ νέου. Συγκεκριμένα, εντοπίζεται η μέγιστη και η ελάχιστη θερμιδική κατανάλωση και με βάση αυτές τις δύο τιμές δημιουργούνται οι πέντε ακόλουθες κλάσεις ώστε η κατανομή να είναι σχετικά ομοιόμορφη.

ΧΡΩΜΑΤΑ	ΠΕΡΙΟΧΗ ΘΕΡΜΙΔΙΚΗΣ ΚΑΤΑΝΑΛΩΣΗΣ (Calorie/min)
Aqua	0 έως 0,967373253
Green	0,967373253 έως 2,902119774
Yellow	2,902119774 έως 3,869493015
Orange	3,869493015 έως 4,8386629
Red	4,8386629 έως 5,804239548

Πίνακας 11.5: Περιοχές θερμιδικής κατανάλωσης και χρώματα απεικόνισης



Εικόνα 11.9: Απεικόνιση θερμιδικής κατανάλωσης ποδηλάτη στην περιοχή του San Fransisco

Μέρος 4

Συμπεράσματα - Συζήτηση

12

ΠΡΟΒΛΗΜΑΤΑ ΚΑΙ ΠΑΡΑΔΟΧΕΣ

Κατά τη διάρκεια υλοποίησης της εφαρμογής προέκυψαν διάφορα προβλήματα. Για χάριν απλοποίησης των προβλημάτων αυτών, αλλά και εξαιτίας της πίεσης του χρόνου έγιναν ορισμένες παραδοχές – υποθέσεις που διευκόλυναν την περάτωση της εφαρμογής. Αυτές περιγράφονται στη συνέχεια.

12.1 Υπολογισμός Μήκους

Το Google Earth βασίζεται στο παγκόσμιο γεωδαιτικό σύστημα WGS84 όπως έχει ήδη αναφερθεί. Εδώ πρέπει να σημειωθεί ότι το WGS84, βασίζεται στο ελλειψοειδές WGS84. Οι παράμετροι του ελλειψοειδούς είναι:

- Μεγάλος ημιάξονας a=6378137 m
- Μικρός ημιάξονας b=6356752.3142 m (WGS84)
- $e^2 = 1-(1-f)^2$
- $e'^2 = 1/((1-f)*(1-f))*e^2$;
- f = 1/298.257223563 (WGS84)

Ένα σημείο στο χώρο σε κάθε Σύστημα Αναφοράς, στη Γεωδαισία μπορεί να εκφραστεί με τα παρακάτω συστήματα συντεταγμένων.
• Τρισορθογώνιες Καρτεσιανές Συντεταγμένες X,Y,Z.
• Ελλειψοειδείς (γεωδαιτικές) συντεταγμένες φ,λ,h. Σε κρατικά (γεωδαιτικά) Συστήματα Αναφοράς χρησιμοποιείται συνήθως το υψόμετρο του κρατικού υψομετρικού συστήματος (για την Ελλάδα το ορθομετρικό υψόμετρο H^0) αντί του γεωμετρικού υψομέτρου.
• Επίπεδες ορθογώνιες συντεταγμένες x,y, (στο προβολικό σύστημα), συνδεδεμένες με το υψόμετρο H^o του σημείου.

Στη συγκεκριμένη εφαρμογή υπολογίστηκαν οι τρισορθογώνιες καρτεσιανές συντεταγμένες X, Y, Z των τμημάτων που συγκροτούν τις διαδρομές. Έτσι υπολογίστηκαν οι πραγματικές συντεταγμένες του σημείου στο χώρο. Στη συνέχεια, προκειμένου να υπολογιστεί η κλίση των τμημάτων αυτών, ήταν αναγκαίος ο υπολογισμός της απόστασης των τμημάτων των διαδρομών. Οι αποστάσεις αυτές που υπολογίστηκαν είναι τα κεκλιμένα μήκη των τμημάτων των διαδρομών στον τρισδιάστατο χώρο. Για την πραγματοποίηση των υπολογισμών θεωρήθηκε ότι τα τμήματα αυτά αποτελούν ευθύγραμμα τμήματα με σταθερή κλίση.

12.2 Παραδοχή Σταθερής Κλίσης

Κατά την υλοποίηση της εφαρμογής ήταν αναγκαίο να γίνει η υπόθεση ότι η κλίση των τμημάτων των διαδρομών που προέκυψαν από την κλήση του Google Direction API παραμένει σταθερή κατά μήκος του εκάστοτε τμήματος της κάθε διαδρομής. Αυτό δε συμβαίνει στην πραγματικότητα καθώς κατά μήκος των τμημάτων μιας διαδρομής η κλίση μπορεί να μεταβάλλεται.

Για διαδρομές σε αστικές περιοχές, όπου το οδικό δίκτυο είναι πολύ πυκνό, η παραδοχή για σταθερή κλίση κατά μήκος των τμημάτων των διαδρομών δεν επηρεάζει σημαντικά το αποτέλεσμα. Αυτό συμβαίνει διότι λόγω της πυκνότητας του οδικού δικτύου, οι διαδρομές αποτελούνται από πολλά και μικρά τμήματα, δηλαδή λαμβάνονται περισσότερα ενδιάμεσα σημεία με αποτέλεσμα την καλύτερη και ακριβέστερη απόδοση των κλίσεων των διαδρομών. Επομένως από όσο περισσότερα ενδιάμεσα σημεία αποτελείται μια διαδρομή, τόσο πιο ρεαλιστικό είναι το αποτέλεσμα.

12.3 Περιορισμοί που επιβάλλονται από το Google Elevation API

Η χρήση του Google Elevation API υπόκειται σε ένα όριο 2.500 αιτημάτων ανά ημέρα. Σε κάθε κλήση – αίτημα ο ενδιαφερόμενος μπορεί να ζητήσει το υψόμετρο μέχρι και 512 περιοχών, αλλά δε μπορεί να ξεπεράσει τις 25.000 περιοχές συνολικά ανά ημέρα. Αυτό το όριο επιβάλλεται ώστε να αποφευχθεί η κατάχρηση ή/και η αλλαγή του στόχου της υπηρεσίας. Το όριο αυτό μπορεί να αλλάξει χωρίς προειδοποίηση. Εάν ο ενδιαφερόμενος ξεπεράσει το 24ωρο όριο ή καταχραστεί την υπηρεσία, το Elevation API ενδέχεται να σταματήσει την εργασία για τον χρήστη προσωρινά. Εάν αυτός συνεχίσει να υπερβαίνει το όριο, η πρόσβασή του στο Elevation API μπορεί να αποκλειστεί.
Οι διευθύνσεις URLs του Elevation API περιορίζονται στους 2048 χαρακτήρες, πριν την κωδικοποίηση του URL.

12.4 Περιορισμοί που επιβάλλονται από το Google Directions API

Η χρήση του Google Directions API υπόκειται στον περιορισμό των 2500 αιτήσεων διαδρομών ανά ημέρα. Οι διαδρομές μεμονωμένα μπορεί να περιέχουν μέχρι και 8 ενδιάμεσα σημεία στην κλήση.

Επίσης, οι διευθύνσεις Directions API URLs περιορίζονται στους 2048 χαρακτήρες, πριν την κωδικοποίηση URL.

13

ΔΥΝΑΤΟΤΗΤΕΣ ΓΙΑ ΑΛΛΕΣ ΕΦΑΡΜΟΓΕΣ

Η εφαρμογή της διπλωματικής εργασίας μπορεί να χρησιμοποιηθεί σε μία πληθώρα άλλων εφαρμογών πέραν του υπολογισμού της θερμιδοκατανάλωσης ενός ποδηλάτη συναρτήσει των κλίσεων των διαδρομών που διανύει.

Παρακάτω περιγράφονται ενδεικτικά ορισμένες εφαρμογές στις οποίες θα μπορούσε να χρησιμοποιηθεί η συγκεκριμένη εφαρμογή.

13.1 Υπολογισμός Παραγωγής Καυσαερίου κινούμενου Οχήματος

Η κλίση του δρόμου καθώς επίσης και η ταχύτητα των οχημάτων έχουν πολύ μεγάλη επιρροή στον ρυθμό εκπομπής καυσαερίων και κατ' επέκταση ρύπων. Κατά την κίνηση των οχημάτων σε δρόμο με θετική κλίση (ανηφόρα), ο ρυθμός εκπομπής ρύπων αυξάνεται ενώ σε αρνητική κλίση δρόμου (κατηφόρα) η εκπομπή ρύπων αντίστοιχα μειώνεται. Βέβαια, σε έναν δρόμο με έντονη κατηφορική κλίση, παρατηρείται ότι ο ρυθμός εκπομπής ρύπων είναι αυξημένος, και αυτό γιατί ο οδηγός χρησιμοποιεί χαμηλές ταχύτητες και φρενάρισμα.

Ο συνδυασμός της εφαρμογής που υλοποιήθηκε με ένα πρότυπο μοντέλο υπολογισμού παραγωγής καυσαερίων ενός οχήματος εν κινήσει, μπορεί να χρησιμοποιηθεί σε περιβαλλοντικές μελέτες. Η εφαρμογή θα χρησιμοποιείται για τον υπολογισμό των εκπεμπόμενων ρύπων ενός κινούμενου οχήματος κατά μήκος μίας διαδρομής συναρτήσει της κλίσης του δρόμου στον οποίο κινείται.

13.2 Χωρική Απεικόνιση αστικών Περιοχών με βάση το Καυσαέριο

Η εφαρμογή της διπλωματικής εργασίας μπορεί επίσης να χρησιμοποιηθεί με σκοπό τη χωρική απεικόνιση αστικών περιοχών με βάση το καυσαέριο. Σε συνδυασμό με την παραπάνω εφαρμογή, θα επιλέγει την αστική περιοχή που επιθυμεί ο ενδιαφερόμενος, θα υπολογίζονται όλες οι διαδρομές και οι κλίσεις των διαδρομών της περιοχής αυτής και στη συνέχεια το παραγόμενο καυσαέριο των εν κινήσει οχημάτων. Κατ' αυτόν τον τρόπο, θα σχηματίζεται μία γενική εικόνα του καυσαερίου που παράγεται από τα οχήματα στην περιοχή αυτή και στη συνέχεια τα αποτελέσματα θα απεικονίζονται χωρικά.

13.3 Συνδυαστικές Εφαρμογές

Η εφαρμογή μπορεί επίσης να χρησιμοποιηθεί για τους σκοπούς συνδυαστικών εφαρμογών. Μία συνδυαστική εφαρμογή για παράδειγμα μπορεί να είναι ο έλεγχος καταλληλότητας ποδηλατοδρόμων με βάση τη δυσκολία διάσχισης (ανωφέρειες) και την έκθεση στο καυσαέριο.

Με βάση τις κλίσεις του οδικού δικτύου και του παραγόμενου καυσαερίου των οχημάτων συναρτήσει της κλίσης όπως περιγράφηκε και παραπάνω, μπορεί να κριθεί αν το οδικό δίκτυο μιας περιοχής είναι κατάλληλο για την κατασκευή ποδηλατοδρόμων σε σχέση με τους δύο αυτούς παράγοντες (κλίση και καυσαέριο).

13.4 Συνδυασμός με Τεχνολογίες Υπολογιστικής Ρευστοδυναμικής για την Προσομοίωση της Διάχυσης Ρύπων με ακριβέστερες αρχικές Συνθήκες

Υπολογιστική Ρευστοδυναμική (Computational Fluid Dynamics) είναι η ανάλυση συστημάτων στα οποία συμβαίνει ροή ρευστών, φαινόμενα μεταφοράς θερμότητας και φαινόμενα που έχουν σχέση με προσομοίωση χημικών αντιδράσεων.

Η επιστήμη της Υπολογιστικής Ρευστοδυναμικής με τα διάφορα υπολογιστικά μοντέλα και τις τεχνικές προσομοίωσης μπορεί να δώσει σοβαρές και με καλή ακρίβεια απαντήσεις για την εξέλιξη των φαινομένων που εμφανίζονται κατά τη διάρκεια εκπομπής ρύπων, έκρηξης, απελευθέρωσης τοξικών ουσιών, φωτιάς κ.ο.κ.

Η χρησιμοποίηση των κωδικών Υπολογιστικής Ρευστοδυναμικής στην προσομοίωση διασποράς πυκνού αερίου, π.χ. μαύρος καπνός, είναι πλέον η καλύτερη λύση περιβαλλοντικών προβλημάτων διότι υπερέχουν σε σύγκριση με τα απλούστερα ημιεμπειρικά μοντέλα, αφού έχουν την ικανότητα να προσομοιώνουν πολύπλοκα φαινόμενα όπως η τύρβη, ενώ παρέχουν τη δυνατότητα ακριβούς κατασκευής και απεικόνισης της γεωμετρίας του επιθυμητού πεδίου, προσεγγίζοντας σε μεγάλο βαθμό τις πραγματικές συνθήκες.

Σήμερα σε πολλές περιπτώσεις η προσομοίωση παραγωγής ρύπων από την κυκλοφορία αυτοκινήτων γίνεται με την παραδοχή ότι οι δρόμοι είναι επίπεδοι, δηλαδή η παραγωγή ρύπων εξαρτάται μόνο από το πλήθος και την ταχύτητα των οχημάτων. Σε συνδυασμό λοιπόν με την εφαρμογή υπολογισμού διαδρομών, κλίσεων των διαδρομών της παρούσας εργασίας και την εφαρμογή υπολογισμού των εκπεμπόμενων ρύπων από τα κινούμενα οχήματα στις διαδρομές αυτές, μπορεί να βελτιωθεί αυτό, εφόσον δεν παράγουν όλοι οι δρόμοι τους ίδιους ρύπους, οι ανηφόρες περισσότερους και οι κατηφόρες λιγότερους. Έτσι, μπορεί να σχεδιαστεί ένα μοντέλο προσομοίωσης διάχυσης ρύπων για χρήση σε τεχνολογίες της υπολογιστικής ρευστοδυναμικής.

13.5 Συνδυασμός με Αλγορίθμους Δρομολόγησης Οχημάτων για τον Υπολογισμό της πιο οικονομικής Διαδρομής

Γνωρίζουμε ότι η μεγάλη κλίση ενός δρόμου συνεπάγεται μεγάλη κατανάλωση καυσίμων για ένα κινούμενο όχημα. Επομένως μπορεί να σχεδιαστεί ένα μοντέλο κατανάλωσης

καυσίμων κατά μήκος πιθανών εναλλακτικών διαδρομών συναρτήσει της κλίσης των διαδρομών αυτών και να επιλέγεται η βέλτιστη οικονομικά διαδρομή, δηλαδή η διαδρομή με την μικρότερη κλίση. Επίσης, θα μπορούσε να συνδυαστεί και με άλλες παραμέτρους όπως το χρόνο διέλευσης του οχήματος κατά μήκος μιας διαδρομής, ώστε να επιλέγεται η οικονομικότερη συνολικά διαδρομή.

Έτσι ο οδηγός του οχήματος θα είναι σε θέση να επιλέγει τη διαδρομή που επιθυμεί η οποία θα μπορεί να συνδυάζει ενδεχομένως οικονομία σε καύσιμα αλλά και σε χρόνο.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Ullman, Larry (2005), «Εισαγωγή στην PHP για τον παγκόσμιο ιστό», εκδόσεις Κλειδάριθμος

Melonie, Julie C. (2008), «Μάθετε PHP, MySQL και Apache», εκδόσεις Γκιούρδας Μ.

Velte, Elsenpeter (2010), «Cloud Computing, μια πρακτική προσέγγιση», εκδόσεις Γκιούρδας Μ.

Danny,Brian (2006), «The Definitive Guide to Berkley DB XML», εκδόσεις Appress

Josie, Wernecke (2007), «The KML Handbook», εκδόσεις Pearson Education (US)

H.M. Deitel, P.J. Deitel (2003), «C++ Προγραμματισμός», εκδόσεις Γκιούρδας Μ.

Γ. Βέης, Χ. Μπιλλήρης, Κ. Παπαζήση (2008), «Κεφάλαια Ανώτερης Γεωδαισίας», εκδόσεις ΕΜΠ

Ασημακόπουλος, Δ. & Ν. Μαρκάτος (1995), «Υπολογιστική Ρευστοδυναμική», εκδόσεις Παπασωτηρίου

URL's

(1) http://www.php.net

(2) http://code.google.com

(3) http://www.ituts.gr

(4) http://en.wikipedia.org

(5) http://el.wikipedia.org/wiki/

(6) http://www.sitepoint.com/really-good-introduction-xml/

(7) http://www.xml.com/pub/a/98/10/guide0.html

(8) http://www.exforsys.com/tutorials/xml/xml-introduction.html#.Tg76JlsuOeY

(9) http://www.w3schools.com/xml/xml_whatis.asp

(10) http://www.garshol.priv.no/download/text/xml-intro/index-en.html

(11) http://www.filaderlis.com/ebooks/XML_tselios.pdf

(12) http://en.wikipedia.org/wiki/Keyhole_Markup_Language

(13) http://code.google.com/intl/el-GR/apis/kml/documentation/

(14) http://www.geosolution.gr/index.php?option=com_content&view=article&id=8&Itemid=15

(15) http://vivliothmmy.ee.auth.gr/909/1/alexandros.koutsonasios.pdf

(16) http://www.ituts.gr/tutorial/programming/about-json

(17) http://en.wikipedia.org/wiki/Web_service

(18) http://www.go-online.gr/ebusiness/specials/article.html?article_id=213&PHPSESSID=11e46f19aeab6950ef5eccef2890b412

(19) http://www.webopedia.com/TERM/W/Web_services.html

(20) http://searchsoa.techtarget.com/definition/Web-services

(21) http://www.it.uom.gr/project/soap/Theory/introduction.html

(22) http://users.uom.gr/~kaklaman/book/Chapters/C8/What%20is%20XML%202.htm

(23) http://office.microsoft.com/el-gr/excel-help/HA010034022.aspx

(24) http://users.uom.gr/~kaklaman/book/Chapters/C8/C8Header.htm

(25) http://www.htmlgoodies.com/beyond/xml/article.php/3473531/What-is-XML.htm

(26) http://webdesign.about.com/od/xml/a/aa091500a.htm

(27) http://communities.softwareag.com/ecosystem/communities/public/Developer/webmethods/products/tamino/faq/XMLStarter/XMLBasics.html

(28) http://www.techrepublic.com/article/what-is-php/5074693

(29) http://www.softwareprojects.org/php-what-is-01.htm

(30) http://www.wisegeek.com/what-is-php.htm

(31) http://www.peachpit.com/articles/article.aspx?p=22290

(32) http://www.cptips.com/formula.htm

ΠΑΡΑΡΤΗΜΑ

```php
<?php

//λ => gewgrafiko mikos, φ => gewgrafiko platos
//Γεωγραφικό μήκος (longitude) (λ), Γεωγραφικό πλάτος (latitude) (φ)
// ================================
// prepare random generator.
// ετοιμη από το php.net
// ================================

//
==========================================================================

function make_seed()
{
  list($usec, $sec) = explode(' ', microtime());
  return (float) $sec + ((float) $usec * 100000);
}

//
==========================================================================

// ================================
// my randoms2
// ================================
function myrand2($digits)
{
    $lowval=1;
    $maxval=pow(10, $digits);

    return mt_rand($lowval, $maxval);
}

//
==========================================================================


// ================================
// my randoms
// ================================
function myrand($from, $to, $ndec)
{
    $SM_degrees=mt_rand($from,$to);

    $SM_decpart=myrand2($ndec)/pow(10,$ndec);

    return $SM_degrees+$SM_decpart;
}

//
==========================================================================


// ========================
// my array of randoms
// ========================
function random_array($howmany, $elaxistosλ, $elaxistosφ, $megistosλ,
$megistosφ, $dekadika)
{

    // --- needed for the random generator ---
```

```
    mt_srand(make_seed());

    for ($i=0;$i<$howmany;$i++) {
      $point["λ"][$i]=myrand($elaxistosλ,$megistosλ,$dekadika);
      $point["φ"][$i]=myrand($elaxistosφ,$megistosφ,$dekadika);
    }

    return $point;
}


//
============================================================================

// ========================
// create kml into a text file
// ========================
function
txtoutput($wheretowrite,$λk,$φk,$MyPointsλ,$MyPointsφ,$curpointname)
{

     $whattowrite="";
     $whattowrite=$whattowrite.PHP_EOL."<Placemark>".PHP_EOL."
<name>Point ".$curpointname."</name>".PHP_EOL." <LookAt>".PHP_EOL."
<longitude>".$λk."</longitude>".PHP_EOL;
     $whattowrite=$whattowrite."
<latitude>".$φk."</latitude>".PHP_EOL."
<altitude>0</altitude>".PHP_EOL."
<heading>0.2554718671115674</heading>".PHP_EOL;
     $whattowrite=$whattowrite."        <tilt>0</tilt>".PHP_EOL."
<range>193836.0400797448</range>".PHP_EOL."
<altitudeMode>relativeToGround</altitudeMode>".PHP_EOL;
     $whattowrite=$whattowrite."
<gx:altitudeMode>relativeToSeaFloor</gx:altitudeMode>".PHP_EOL."
</LookAt>".PHP_EOL."    <styleUrl>#msn_ylw-pushpin</styleUrl>".PHP_EOL;
     $whattowrite=$whattowrite."    <Point>".PHP_EOL."
<altitudeMode>clampToGround</altitudeMode>".PHP_EOL."
<gx:altitudeMode>clampToSeaFloor</gx:altitudeMode>";


     $whattowrite=$whattowrite.PHP_EOL."
<coordinates>".$MyPointsλ.",".$MyPointsφ.",0 </coordinates>".PHP_EOL."
</Point>".PHP_EOL."</Placemark>".PHP_EOL;


    $myFile = $wheretowrite;
    $fh = fopen($myFile, 'a') or die("can't open file");
    fwrite($fh, $whattowrite);
    fclose($fh);


    return;
}


//
============================================================================

// ==================================
function filesave($myfilename, $myarray_f, $myarray_l, $side,
$mypointname)
{
// γράφει (append) στο αρχείο myfilename τα περιεχόμενα f l του
myarray_l, _f προσθέτοντας το side σε κάθε γραμμή
```

```
// side, N, f(N) l(N)

    $mytxtdata="";
    $mytxtdata=$mytxtdata.PHP_EOL.$mypointname."    ".$myarray_f."
".$myarray_l."  $side".PHP_EOL;

    $mytxtFile = $myfilename;
    $fh = fopen($mytxtFile, 'a') or die("can't open file");
    fwrite($fh, $mytxtdata);
    fclose($fh);

    return;
}

//
================================================================================


// ====================
// "create_random_points()"
// ====================

function create_random_points(&$lλ,&$hλ,&$lφ,&$hφ,$mykmlfilename,
$headersfilename, $density, $_elev, $_calories, $usertag,
$mykmlfilename2, $headersfilename2)

{

    // $decimals is the number of decimals
    $decimals=6;
    $denominator=pow(10,$decimals);

    $lowλ=abs($lλ);
    $hiλ=abs($hλ);
    $lowφ=abs($lφ);
    $hiφ=abs($hφ);

    //dimiourgia pinakvn  me tis 4 pragmatikes sun/nes
    $lλreal=-84171456/$denominator;
    $hλreal=-81440765/$denominator;
    $lφreal=39573987/$denominator;
    $hφreal=41109592/$denominator;

    $A=array("λa"=>$lλreal,"φa"=>$lφreal);
    $B=array("λb"=>$hλreal,"φb"=>$lφreal);
    $C=array("λc"=>$hλreal,"φc"=>$hφreal);
    $D=array("λd"=>$lλreal,"φd"=>$hφreal);

    $pinakas=array_merge((array)$A,(array)$B,(array)$C,(array)$D);


    $R=6373;
    $distance_km=$density/1000;

    $w_rad=$distance_km/$R;
    $w_deg=$w_rad*180/M_PI;


    // $inlow is the lowest valid random value for the inner rectangle
    // $inhi is the highest for the inner rectangle
    $monades_λ=abs($hiλ-$lowλ)*15/100;
    $monades_φ=abs($hiφ-$lowφ)*15/100;
```

```
    //DIAKRINW TESSERIS PERIPTWSEIS ANALOGA ME TO TETARTIMORIO POU
VRISKOMAI
    //A.
    if ($hλ>0 && $lλ>0 && $hφ>0 && $lφ>0)
    {   $inlowλ=$lowλ+$monades_λ;
        $inhiλ=$hiλ-$monades_λ;
        $inlowφ=$lowφ+$monades_φ;
        $inhiφ=$hiφ-$monades_φ;
    }

    //B.
    elseif ($hλ>0 && $lλ>0 && $hφ<0 && $lφ<0)
    {   $inlowλ=$lowλ+$monades_λ;
        $inhiλ=$hiλ-$monades_λ;
        $inlowφ=-($lowφ-$monades_φ);
        $inhiφ=-($hiφ+$monades_φ);
    }

    //C.
    elseif ($hλ<0 && $lλ<0 && $hφ<0 && $lφ<0)
    {   $inlowλ=-($lowλ-$monades_λ);
        $inhiλ=-($hiλ+$monades_λ);
        $inlowφ=-($lowφ-$monades_φ);
        $inhiφ=-($hiφ+$monades_φ);
    }

    //D.
    elseif ($hλ<0 && $lλ<0 && $hφ>0 && $lφ>0)
    {   $inlowλ=-($lowλ-$monades_λ);
        $inhiλ=-($hiλ+$monades_λ);
        $inlowφ=$lowφ+$monades_φ;
        $inhiφ=$hiφ-$monades_φ;
    }

    //E.
    elseif ($hλ>0 && $lλ>0 && $hφ>0 && $lφ<0)
    {   $inlowλ=$lowλ+$monades_λ;
        $inhiλ=$hiλ-$monades_λ;
        $inlowφ=-($lowφ-$monades_φ);
        $inhiφ=$hiφ-$monades_φ;  }

    //F.
    elseif ($hλ>0 && $lλ<0 && $hφ<0 && $lφ<0)
    {   $inlowλ=-($lowλ-$monades_λ);
        $inhiλ=$hiλ-$monades_λ;
        $inlowφ=-($lowφ-$monades_φ);
        $inhiφ=-($hiφ+$monades_φ);   }

    //G.
    elseif ($hλ<0 && $lλ<0 && $hφ>0 && $lφ<0)
    {   $inlowλ=-($lowλ-$monades_λ);
        $inhiλ=-($hiλ+$monades_λ);
        $inlowφ=-($lowφ-$monades_φ);
        $inhiφ=$hiφ-$monades_φ;  }

    //H.
    elseif ($hλ>0 && $lλ<0 && $hφ>0 && $lφ>0)
    {   $inlowλ=-($lowλ-$monades_λ);
        $inhiλ=$hiλ-$monades_λ;
        $inlowφ=$lowφ+$monades_φ;
        $inhiφ=$hiφ-$monades_φ;  }

    //I.
    elseif ($hλ>0 && $lλ<0 && $hφ>0 && $lφ<0)
```

```
    {   $inlowλ=-($lowλ-$monades_λ);
        $inhiλ=$hiλ-$monades_λ;
        $inlowφ=-($lowφ-$monades_φ);
        $inhiφ=$hiφ-$monades_φ;  }

    //sugkentrwtika oi 4 pragmatikes sun/nes (me dekadika kai prosima) se
pinaka
    $inlowrealλ=$inlowλ/$denominator;
    $inhirealλ=$inhiλ/$denominator;
    $inlowrealφ=$inlowφ/$denominator;
    $inhirealφ=$inhiφ/$denominator;

    $E=array("λe"=>$inlowrealλ,"φe"=>$inlowrealφ);
    $F=array("λf"=>$inhirealλ,"φf"=>$inlowrealφ);
    $G=array("λg"=>$inhirealλ,"φg"=>$inhirealφ);
    $H=array("λh"=>$inlowrealλ,"φh"=>$inhirealφ);

    $pinakas1=array_merge((array)$E,(array)$F,(array)$G,(array)$H);


    // DIAKRINW TESSERIS UPOPERIOXES OPOU EPANALAMVANW TIN IDIA
DIADIKASIA
    //arxika ipologizw ton arithmo twn simeiwn pou prokuptoun me vasi to
emvadon kai tin epithumiti apostasi gia kathe ipoperioxi
    $num1=(abs(abs($inhiφ)-abs($inlowφ))*abs(abs($inlowλ)-
abs($lλ)))/($w_deg*$w_deg*$denominator*$denominator);
    $n1=(int)($num1);
    echo "\n"."Number of points in the first segment:".$n1."\n";
    $num2=(abs(abs($inhiφ)-abs($inlowφ))*abs(abs($hλ)-
abs($inhiλ)))/($w_deg*$w_deg*$denominator*$denominator);
    $n2=(int)($num2);
    echo "Number of points in the second segment:".$n2."\n";
    $num3=(abs(abs($inlowφ)-abs($lφ))*abs(abs($hλ)-
abs($lλ)))/($w_deg*$w_deg*$denominator*$denominator);
    $n3=(int)($num3);
    echo "Number of points in the third segment:".$n3."\n";
    $num4=(abs(abs($hφ)-abs($inhiφ))*abs(abs($hλ)-
abs($lλ)))/($w_deg*$w_deg*$denominator*$denominator);
    $n4=(int)($num4);
    echo "Number of points in the fourth segment:".$n4."\n";

    //sunolikos arithmos simeiwn =>gia tis 4 upoperioxes
    $n=$n1+$n2+$n3+$n4;
    echo "Total number of points:".$n."\n";

    //I.

    $mypoints1=random_array($n1, $lλ, $inlowφ, $inlowλ, $inhiφ,
$decimals);


    for ($i=0;$i<$n1;$i++) {
      $mypoints1[λ][$i]=$mypoints1[λ][$i]/$denominator;
      $mypoints1[φ][$i]=$mypoints1[φ][$i]/$denominator;

    }



    //II.

    $mypoints2=random_array($n2, $inhiλ, $inlowφ, $hλ, $inhiφ,
$decimals);
```

```
    for ($i=0;$i<$n2;$i++) {
      $mypoints2[λ][$i]=$mypoints2[λ][$i]/$denominator;
      $mypoints2[φ][$i]=$mypoints2[φ][$i]/$denominator;

    }

    //III.

    $mypoints3=random_array($n3, $lλ, $lφ, $hλ, $inlowφ, $decimals);

    for ($i=0;$i<$n3;$i++) {
      $mypoints3[λ][$i]=$mypoints3[λ][$i]/$denominator;
      $mypoints3[φ][$i]=$mypoints3[φ][$i]/$denominator;

    }

    //IV.

    $mypoints4=random_array($n4, $lλ, $inhiφ, $hλ, $hφ, $decimals);

    for ($i=0;$i<$n4;$i++) {
      $mypoints4[λ][$i]=$mypoints4[λ][$i]/$denominator;
      $mypoints4[φ][$i]=$mypoints4[φ][$i]/$denominator;

    }



    //sugxwneusi twn pinakwn $mypoints[λ] kai $mypoints[φ] wste na
emfanizontai ws λ,φ oi suntetagmenes twn TUXAIWN simeiwn
    $final1 = array();
    foreach($mypoints1[λ] as $num => $value) {
    $final1[] = array($value, $mypoints1[φ][$num]);
    }
    $final2 = array();
    foreach($mypoints2[λ] as $num => $value) {
    $final2[] = array($value, $mypoints2[φ][$num]);
    }
    $final3 = array();
    foreach($mypoints3[λ] as $num => $value) {
    $final3[] = array($value, $mypoints3[φ][$num]);
    }
    $final4 = array();
    foreach($mypoints4[λ] as $num => $value) {
    $final4[] = array($value, $mypoints4[φ][$num]);
    }


$final_array=array_merge((array)$final1,(array)$final2,(array)$final3,(ar
ray)$final4);


    // use ready header file (name in $headersfilename)...
    $filedata="";


    $fh = fopen($headersfilename, 'r') or die("can't open file");
    while (!feof($fh)) {
      $filedata=$filedata.fgets($fh);
    }
    fclose($fh);

    $fh = fopen($mykmlfilename, 'a') or die("can't open file");
    fwrite($fh, $filedata);
```

```
    fclose($fh);

     $λ_k=($lλ+($hλ-$lλ)/2)/$denominator;
     $φ_k=($lφ+($hφ-$lφ)/2)/$denominator;


    // append to this file our stuff
    $currentpointname=1;

    for ($i=0; $i<$n1; $i++) {

txtoutput($mykmlfilename,$λ_k,$φ_k,$mypoints1[λ][$i],$mypoints1[φ][$i],$c
urrentpointname);
    $currentpointname=$currentpointname+1;
    }

    for ($i=0; $i<$n2; $i++) {

txtoutput($mykmlfilename,$λ_k,$φ_k,$mypoints2[λ][$i],$mypoints2[φ][$i],$c
urrentpointname);
    $currentpointname=$currentpointname+1;
    }

    for ($i=0; $i<$n3; $i++) {

txtoutput($mykmlfilename,$λ_k,$φ_k,$mypoints3[λ][$i],$mypoints3[φ][$i],$c
urrentpointname);
    $currentpointname=$currentpointname+1;
    }

    for ($i=0; $i<$n4; $i++) {

txtoutput($mykmlfilename,$λ_k,$φ_k,$mypoints4[λ][$i],$mypoints4[φ][$i],$c
urrentpointname);
    $currentpointname=$currentpointname+1;
    }


    // plus a few more lines to close tags...
    $λl=$lλ/$denominator;
    $λh=$hλ/$denominator;
    $φl=$lφ/$denominator;
    $φh=$hφ/$denominator;


    $fh = fopen($mykmlfilename, 'a') or die("can't open file");

$filedata=PHP_EOL."<Placemark>".PHP_EOL."<name>OUT</name>".PHP_EOL."<styl
eUrl>#msn_ylw-
pushpin</styleUrl>".PHP_EOL."<Polygon>".PHP_EOL."<tessellate>1</tessellat
e>".PHP_EOL;
    $filedata=$filedata.PHP_EOL."       <outerBoundaryIs>".PHP_EOL."
<LinearRing>".PHP_EOL;
    $filedata=$filedata.PHP_EOL."
<coordinates>".PHP_EOL.$λl.",".$φh.",0 ".$λh.",".$φh.",0
".$λh.",".$φl.",0 ".$λl.",".$φl.",0 ".PHP_EOL;
    $filedata=$filedata.PHP_EOL."       </coordinates>".PHP_EOL."
</LinearRing>".PHP_EOL."    </outerBoundaryIs>".PHP_EOL;
    $filedata=$filedata.PHP_EOL."
</Polygon>".PHP_EOL."</Placemark>".PHP_EOL;

    fwrite($fh, $filedata);
    fclose($fh);
```

```
    $fh = fopen($mykmlfilename, 'a') or die("can't open file");

$filedata=PHP_EOL."<Placemark>".PHP_EOL."<name>IN</name>".PHP_EOL."<style
Url>#msn_ylw-
pushpin0</styleUrl>".PHP_EOL."<Polygon>".PHP_EOL."<tessellate>1</tessella
te>".PHP_EOL;
    $filedata=$filedata.PHP_EOL."       <outerBoundaryIs>".PHP_EOL."
<LinearRing>".PHP_EOL;
    $filedata=$filedata.PHP_EOL."
<coordinates>".PHP_EOL.$inlowrealλ.",".$inhirealφ.",0
".$inhirealλ.",".$inhirealφ.",0 ".$inhirealλ.",".$inlowrealφ.",0
".$inlowrealλ.",".$inlowrealφ.",0 ".PHP_EOL;
    $filedata=$filedata.PHP_EOL."       </coordinates>".PHP_EOL."
</LinearRing>".PHP_EOL."    </outerBoundaryIs>".PHP_EOL;
    $filedata=$filedata.PHP_EOL."
</Polygon>".PHP_EOL."</Placemark>".PHP_EOL;


    $filedata.=PHP_EOL."</Document>".PHP_EOL."</kml>".PHP_EOL;

    fwrite($fh, $filedata);
    fclose($fh);


    //write the points to txt
    $pointname=1;
    $txtfilename=str_replace(".kml", "-1.txt", $mykmlfilename);

    $txtdata="";

    $fh = fopen($txtfilename, 'a') or die("can't open file");
    $txtdata=$txtdata.PHP_EOL."N    F   L   Side".PHP_EOL;
    fwrite($fh, $txtdata);
    fclose($fh);


    for ($i=0; $i<$n1; $i++) {
    filesave($txtfilename, $mypoints1[φ][$i], $mypoints1[λ][$i], "left",
$pointname);
    $pointname=$pointname+1;
    }

    for ($i=0; $i<$n2; $i++) {
    filesave($txtfilename, $mypoints2[φ][$i], $mypoints2[λ][$i], "right",
$pointname);
    $pointname=$pointname+1;
    }

    for ($i=0; $i<$n3; $i++) {
    filesave($txtfilename, $mypoints3[φ][$i], $mypoints3[λ][$i], "down",
$pointname);
    $pointname=$pointname+1;
    }

    for ($i=0; $i<$n4; $i++) {
    filesave($txtfilename, $mypoints4[φ][$i], $mypoints4[λ][$i], "up",
$pointname);
    $pointname=$pointname+1;
    }


    $m_index=0;
    $M_all=array($m_index => array("w" =>0, "lat1"=>0, "lng1" => 0, "h1"
=> 0, "lat2"=>0, "lng2" => 0, "h2" => 0));
```

```
    //
    echo PHP_EOL."ARRAY1-2".PHP_EOL;

    for ($i=0; $i<$n1; $i++) {
//  for ($i=0; $i<1; $i++) {
      $waitsecs=mt_rand(1,10);
      echo PHP_EOL."Waiting for ".$waitsecs." ms...".PHP_EOL;
      sleep($waitsecs);
      echo PHP_EOL.$mypoints1[λ][$i].", ".$mypoints1[φ][$i]." ->
".$mypoints2[λ][$i].", ".$mypoints2[φ][$i];

      $current_tag=trim($usertag)."-
lr".str_pad(trim($i),4,"0",STR_PAD_LEFT)."-";

      $M1=slope($mypoints1[φ][$i], $mypoints1[λ][$i], $mypoints2[φ][$i],
$mypoints2[λ][$i], $_elev, $_calories, $current_tag);
      $M_all = array_merge((array)$M_all, (array)$M1);
      echo "...done".PHP_EOL;

    }

    echo PHP_EOL."ARRAY3-4".PHP_EOL;

    for ($i=0; $i<$n3; $i++) {
//  for ($i=0; $i<1; $i++) {
      $waitsecs=mt_rand(1,10);
      echo PHP_EOL."Waiting for ".$waitsecs." ms...".PHP_EOL;
      sleep($waitsecs);
      echo PHP_EOL.$mypoints3[λ][$i].", ".$mypoints3[φ][$i]." ->
".$mypoints4[λ][$i].", ".$mypoints4[φ][$i];

      $current_tag=trim($usertag)."-
ud".str_pad(trim($i),4,"0",STR_PAD_LEFT)."-";

      $M2=slope($mypoints3[φ][$i], $mypoints3[λ][$i], $mypoints4[φ][$i],
$mypoints4[λ][$i], $_elev, $_calories, $current_tag);
      echo "...done".PHP_EOL;
      $M_all = array_merge((array)$M_all, (array)$M2);

    }


    echo "\n==============EXITING 2 calls of SLOPE=============\n";
    $m_all_n=count($M_all);

    $min_slope=$M_all[1]['w'];
    $max_slope=$M_all[1]['w'];

    for ($ti=1;$ti<$m_all_n;$ti++) {
        echo $ti."  = ".$M_all[$ti]['w']."  ".$M_all[$ti]['lat1']."
".$M_all[$ti]['lng1'].PHP_EOL;

        if ($M_all[$ti]['w']<$min_slope)
           $min_slope=$M_all[$ti]['w'];

        if ($M_all[$ti]['w']>$max_slope)
           $max_slope=$M_all[$ti]['w'];

    }
    echo "\n============================\n";

    echo "MIN SLOPE=".$min_slope."  MAX=".$max_slope.PHP_EOL;
```

```
    //edw vriskw ti megisti kai tin elaxisti klisi wste na ftiaksw
klaseis pou tha voithisoun stin apeikonisi tous

    $vima=($max_slope-$min_slope)/5;

    $klasi1=$min_slope+$vima;
    $klasi2=$klasi1+$vima;
    $klasi3=$klasi2+$vima;
    $klasi4=$klasi3+$vima;
    $klasi5=$klasi4+$vima;


    $filedata2="";

    $fh = fopen($headersfilename2, 'r') or die("can't open file");
    while (!feof($fh)) {
      $filedata2=$filedata2.fgets($fh);
    }
    fclose($fh);

    $fh = fopen($mykmlfilename2, 'a') or die("can't open file");
    fwrite($fh, $filedata2);
    fclose($fh);

    for ($i=1;$i<$m_all_n;$i++)
    {
        $currentlinename=1;

        if($M_all[$i]['w']<$klasi1)
        {
            $colourScale="#MyScale1";
        }
        elseif($M_all[$i]['w']>$klasi1 && $M_all[$i]['w']<$klasi2)
        {
            $colourScale="#MyScale2";
        }
        elseif($M_all[$i]['w']>$klasi2 && $M_all[$i]['w']<$klasi3)
        {
            $colourScale="#MyScale3";
        }
        elseif($M_all[$i]['w']>$klasi3 && $M_all[$i]['w']<$klasi4)
        {
            $colourScale="#MyScale4";
        }
        elseif($M_all[$i]['w']>$klasi4)
        {
            $colourScale="#MyScale5";
        }

        $fh = fopen($mykmlfilename2, 'a') or die("can't open file");
        $filedata2=$filedata2.PHP_EOL;
        $filedata2=PHP_EOL."<Placemark>".PHP_EOL."
<name>Line".$currentlinename."</name>".PHP_EOL."
<description>Line".$currentlinename."</description>".PHP_EOL."
<styleUrl>".$colourScale."</styleUrl>".PHP_EOL;
        $filedata2=$filedata2." <LineString>".PHP_EOL."
<tessellate>1</tessellate>".PHP_EOL."
<altitudemode>absolute</altitudemode>".PHP_EOL."
<coordinates>".PHP_EOL.$M_all[$i]['lng1'].",".$M_all[$i]['lat1'].",".$M_a
ll[$i]['h1']."
".$M_all[$i]['lng2'].",".$M_all[$i]['lat2'].",".$M_all[$i]['h2'].PHP_EOL.
"</coordinates>".PHP_EOL;
```

```
        $filedata2=$filedata2.PHP_EOL."
</LineString>".PHP_EOL."</Placemark>";

        fwrite($fh, $filedata2);
        fclose($fh);

        $currentlinename++;

    }


    $fh = fopen($mykmlfilename2, 'a') or die("can't open file");
    $filedata2=$filedata2.PHP_EOL;
    $filedata2=PHP_EOL."</Document>".PHP_EOL."</kml>".PHP_EOL;

    fwrite($fh, $filedata2);
    fclose($fh);



    return;
}

//
================================================================================

// --------------------------------------------------------------------
function directions($startlat, $startlng, $endlat, $endlng, &$pppoints)
{

    $serviceurl = 'http://maps.googleapis.com/maps/api/directions/json?';

    $params    = array(
    'origin'      => $startlat.",".$startlng,
    'destination' => $endlat.",".$endlng,
    'sensor'      => 'false',
    );

    $myparameters=http_build_query($params);

    $myrequest=$serviceurl.$myparameters;

    // Fetch and decode JSON string into a PHP object
    $json = file_get_contents($myrequest);

    $data = json_decode($json);

    $points_found=0; $croute=0;

    // If we got directions, output all of the coordinates
    if ($data->status === 'OK') {
    //$route = $data->routes[0];

        foreach ($data->routes as $route) {

        $cleg=0;

        foreach ($route->legs as $leg) {
        $cstep=0;

        foreach ($leg->steps as $step) {


            //apothikeusi sun/nwn se pinaka
```

```
            $start_lat[$points_found]=$step->start_location->lat;
            $start_lng[$points_found]=$step->start_location->lng;

            $pppoints[$points_found]['lat']=$start_lat[$points_found];
            $pppoints[$points_found]['lng']=$start_lng[$points_found];

            //apothikeusi sun/nwn se txt


            $points_found++;
            $cstep++;

            }

        //-------------------------------------------------
        //apothikeusi kai tou teleutaiou simeiou sto txt, to opoio den
sumperielamvanetai parapanw kathws exw parei mono ta simeia start
        $end_lat[$points_found]=$step->end_location->lat;
        $end_lng[$points_found]=$step->end_location->lng;

        $pppoints[$points_found]['lat']=$end_lat[$points_found];
        $pppoints[$points_found]['lng']=$end_lng[$points_found];


        //-------------------------------------------------

        $cleg++;
        }
        $croute++;
        }


    }

    return $points_found; // $myFile;
}

// ------------------------------------------------------------------------
// --- function calc_elevations($ff, $ll, $hh);
function calc_elevations($mystartlat, $mystartlng, $myendlat, $myendlng,
$myFile){


    $points[0]['lat']=0;
    $points[0]['lng']=0;



    $npoints=directions($mystartlat, $mystartlng, $myendlat, $myendlng,
$points);


    // --- η δικτυακή διεύθυνση της υπηρεσίας
    $serviceurl = 'http://maps.googleapis.com/maps/api/elevation/json?';

    // --- υποχρεωτικά οι παράμετροι αρχίζουν με "locations="
    $parameters="locations=";  // required

    // --- η συμβολοσειρά με τις παραμέτρους μαζί με τη διεύθυνση πρέπει
    // --- να μην ξεπερνά σε μήκος τους 2048 χαρακτήρες.
    // --- Χρησιμοποιούμε TRIM για να αφαιρέσουμε τα κενά.
    // --- Χωράνε με ασφάλεια 80 σημεία
```

```
    // *** με κατάλληλο τρόπο βάζουμε τις συντεταγμένες που έχουμε στο
string "parameters"
    for ($i=0;$i<$npoints-1;$i++)
        $parameters =
$parameters.trim($points[$i]['lat']).",".trim($points[$i]['lng'])."|";

    // *** το τελευταίο σημείο το βάζουμε μόνο του γιατι δεν θέλουμε το
"|" που χωρίζει τα σημεία
    $parameters =
$parameters.trim($points[$i]['lat']).",".trim($points[$i]['lng']);

    // --- τελική συμβολοσειρά που θα περάσει στην κλήση του API
    $myrequest=$serviceurl.$parameters."&sensor=false";

    // --- Κλήση της υπηρεσίας: Fetch and decode JSON string into a PHP
object
    $json = file_get_contents($myrequest);

    // --- Αποτελέσματα
    $data = json_decode($json);

    // *** μετρητής αποτελεσμάτων.
    $points_found1=0;

    //zitaw to onoma tou arxeiou pou thelw na dimiourgisw
    //--------------------------


    $mytxtdata="";
    $mytxtFile=$myFile;
    $fh = fopen($mytxtFile, 'a') or die("can't open file");
    $mytxtdata=$mytxtdata.PHP_EOL."N    Φ  Λ  h".PHP_EOL;
    fwrite($fh, $mytxtdata);
    fclose($fh);
    //--------------------------

    $mydata="";

    // --- διάτρεξη αποτελεσμάτων
    if ($data->status === 'OK') {

        foreach ($data->results as $fullpoint)
        {

            //-----------------------------------
            $lat[$points_found1]=$fullpoint->location->lat;
            $lon[$points_found1]=$fullpoint->location->lng;
            $h[$points_found1]=$fullpoint->elevation;

            //edw den kanw append 'a' alla aplo write 'w', dioti me to
append mou ksanagrafei ola ta simeia sto txt se kathe epanalipsi
            $fh = fopen($mytxtFile, 'w') or die("can't open file");
            $mydata=$mydata.$points_found1."    ".$lat[$points_found1]."
".$lon[$points_found1]."    ".$h[$points_found1].PHP_EOL;

            fwrite($fh, $mydata);
            fclose($fh);
            //-----------------------------------

            $points_found1++;
        }
        echo "\nFound ".$points_found1." points\n";
    }
```

```
    else
    {
        // *** οταν γίνει συνάρτηση, θα πρέπει να επιστρέφει κάτι
διαφορετικό από 0
        echo "\n\nERROR calling elevation API: ".$data->status."\n\n";
    }

return;
}

// ------------------------------------------------------------------------
// to be validated!!!
function CalcLen($LA1, $LG1, $LA2, $LG2, $HH1, $HH2)
{
        //xrisi parametrwn WGS84. to WGS84 einai ekseliksi tou GRS 80 k
exoun poli mikres diafores \
        $a=6378137; //se metra
        //1/$f=298.257223563;
        $e=sqrt(0.0066943799901);

        $N1=$a/sqrt((1-
(pow($e,2))*(sin(deg2rad($LA1)))*(sin(deg2rad($LA1)))));
        $N2=$a/sqrt((1-
(pow($e,2))*(sin(deg2rad($LA2)))*(sin(deg2rad($LA2)))));

        $X1=($N1+$HH1)*(cos(deg2rad($LA1)))*(cos(deg2rad($LG1))); //metra
        $Y1=($N1+$HH1)*(cos(deg2rad($LA1)))*(sin(deg2rad($LG1))); //metra
        $Z1=($N1*(1-(pow($e,2)))+$HH1)*(sin(deg2rad($LA1))); //metra

        $X2=($N2+$HH2)*(cos(deg2rad($LA2)))*(cos(deg2rad($LG2))); //metra
        $Y2=($N2+$HH2)*(cos(deg2rad($LA2)))*(sin(deg2rad($LG2))); //metra
        $Z2=($N2*(1-(pow($e,2)))+$HH2)*(sin(deg2rad($LA2))); //metra

        $LC=sqrt(($X2-$X1)*($X2-$X1)+($Y2-$Y1)*($Y2-$Y1)+($Z2-$Z1)*($Z2-
$Z1)); //keklimeni apostasi se metra (OXI i provoliki orizontia apostasi)

        return $LC;
}

// ------------------------------------------------------------------------
function calc_calories($s)
{

    //Assuming: no head wind, constant speed i.e no acceleration or
deceleration, ideal road or mtn. bike and rider
    /*Cv = speed of cyclist in meters/sec
    1 mph = .447 meters/sec
    1 mph = 1.609 kilometeres/hr
    K1 and K2 are constants (see table below)
    Cw = headwind in meters/sec
    Em = mass of cyclist and bicycle in kg
    1 pound = .4536 kg
    s = slope or grade in %
    a = acceleration of the bicycle in meters/(sec)(sec)
    g = gravitational accel = 9.806 m/sec-sec at sea level
    */

    $Cw=0;
    $a=0;
    $K1=3.509;
    $K2=0.2581;
    $Cv=10000/3600; //10 km/h=10000/3600 m/sec.
    $Em=85; //in kg 10+75=gear-bicycle+cyclist
```

```
    if ($s==0 || $s<0)
    {   $W=0;   }

    elseif($s>0)
    {
        $W=($Cv*($K1+($K2*($Cv)*($Cv))+(10.32*$Em*(abs($s)))))/69.78;
//in kilocal/min. isxuei 69.78W = 1000 calories/min = 1 kilocal/min = 1
Calorie/min
    }

    return $W;
}

// ------------------------------------------------------------------------
function slope($ms_startlat, $ms_startlng, $ms_endlat, $ms_endlng,
$ms_elevationsFile, $ms_caloriesFile, $ms_tag)
{

    calc_elevations($ms_startlat, $ms_startlng, $ms_endlat, $ms_endlng,
$ms_elevationsFile);

    $filename=$ms_elevationsFile;
    $lines = array();
    $file = fopen($filename, "r");
    while(!feof($file)) {

        //read file line by line into a new array element
        $lines[] = fgets($file);

    }
    fclose ($file);

    $mspoint_num=count($lines);

    for ($i=0;$i<$mspoint_num;$i++){
        $parts=explode("    ",$lines[$i]);

        $mspoints[$i]['N']=$parts[0];
        $mspoints[$i]['lat']=$parts[1];
        $mspoints[$i]['lng']=$parts[2];
        $mspoints[$i]['h']=$parts[3];
    }

    //-----ipologismos ipsometrikwn diaforwn
    $i=1;

    $fh = fopen($ms_caloriesFile, 'a') or die("can't open file");
    $txtdata="";     //"tag, f1, l1, h1, f2, l2, h2, length, dh, slope,
calories";

    $mindex=0;
    $M=array($mindex => array("w" =>0, "lat1"=>0, "lng1" => 0, "h1" => 0,
"lat2"=>0, "lng2" => 0, "h2" => 0));

    for ($j=0;$j<$mspoint_num-2;$j++){


        $h2=$mspoints[$i]['h'];
        $h1=$mspoints[$j]['h'];
        $dh=($h2-$h1);

        //Oi trisdiastates kartesianes suntetagmenes prokuptoun apo tis
elleipsoeideis
        //$L einai i keklimeni apostasi metaksi 2 simeiwn
```

```
        //φ=latitude kai λ=longtitude
        //To google earth xrisimopoiei to WGS84 ws epifaneia anaforas

        $lat1=$mspoints[$j]['lat'];
        $lng1=$mspoints[$j]['lng']; //epeidi to lng einai arnitiko
(prepei na parw periptwseis?)=> oxi dioti to apotelesma einai idio. etsi
to lng ekfrazetai se moires apo 0 ws 180 i -180 kai oxi 0 ws 360

        $lat2=$mspoints[$i]['lat'];
        $lng2=$mspoints[$i]['lng'];

        $L=CalcLen($lat1, $lng1, $lat2, $lng2, $h1, $h2);

        //uplogismos klisewn w=arcsin(dh/L);
        $w=(asin($dh/$L))*100;

        //topothetisi apotelesmatwn se pinakes
        $DH[$j]=$dh;
        $Length[$j]=$L;
        $W[$j]=$w;
        $M[$mindex]['w']=$W[$j];
        $M[$mindex]['lat1']=$mspoints[$j]['lat'];
        $M[$mindex]['lng1']=$mspoints[$j]['lng'];
        $M[$mindex]['lat2']=$mspoints[$i]['lat'];
        $M[$mindex]['lng2']=$mspoints[$i]['lng'];
        $M[$mindex]['h1']=$mspoints[$j]['h'];
        $M[$mindex]['h2']=$mspoints[$i]['h'];
        $mindex++;

        $tag=$ms_tag.str_pad(trim($j),3,"0", STR_PAD_LEFT);
        $val=calc_calories($w);
        $VAL[$j]=$val;

        $txtdata=$txtdata.PHP_EOL.$tag.", ".trim($lat1).",
".trim($lng1).", ".trim($h1).", ".trim($lat2).", ".trim($lng2).",
".trim($h2).", ".trim($L).", ".trim($dh).", ".trim($w).", ".trim($val);

        $i++;


    }


    echo "\n==============EXITING SLOPE=============\n";
    for ($ti=0;$ti<$mindex;$ti++) {
        echo $M[$ti]['w']."  ".$M[$ti]['lat1']."
".$M[$ti]['lng1'].PHP_EOL;
    }
    echo "\n============================\n";

    fwrite($fh, $txtdata);
    fclose($fh);

    //-----

    return $M;
}



// ----------------------
// EXECUTION BEGINS HERE
// ----------------------
```

```
    //$dλ,uλ to aristero kai to deksi orio gewgrafikou mikos, dφ kai uφ
to katw kai to panw orio gewgrafikou platous

    //zitaw to onoma tou arxeiou apo to opoio tha diavazw tis
sintetagmenes $dλ, $uλ, $dφ, $uφ
    echo "file to read (no extension):";
    $mytxtfilename_=fgets(STDIN);
    $mytxtfilename_=rtrim($mytxtfilename_).".txt";

    $headersfilename_="headers.txt";

    $headersfilename2_="headers2.txt";

    //diavazw tis sun/nes dφ, dλ, uφ, uλ apo txt arxeio------------
    $lines = array();
    $file = fopen($mytxtfilename_, "r");
    while(!feof($file)) {

        //read file line by line into a new array element
        $lines[] = fgets($file);

    }
    fclose ($file);

    $line_num=count($lines);

    for ($i=1;$i<$line_num-1;$i++){

        $parts=explode("    ",$lines[$i]);

        $coordinates[$i]['dλ']=$parts[0];
        $coordinates[$i]['uλ']=$parts[1];
        $coordinates[$i]['dφ']=$parts[2];
        $coordinates[$i]['uφ']=$parts[3];
        $coordinates[$i]['tag']=$parts[4];
        $coordinates[$i]['place']=$parts[5];
        }

        $usertag_=$coordinates[1]['tag'];

        $mykmlfilename_=rtrim($coordinates[1]['place']).".kml";


        $mykmlfilename2_=str_replace(".kml","-
slopes.kml",$mykmlfilename_);



        echo "MEAN distance between points (meters):";
        $distance_m=fgets(STDIN);

        $myElevationsFile=str_replace(".kml","-
elevations.txt",$mykmlfilename_);

        $myCaloriesFile=str_replace("-elevations.txt", "-calories.txt",
$myElevationsFile);


        $txtdata="";

        create_random_points($coordinates[1]['dλ'],
$coordinates[1]['uλ'], $coordinates[1]['dφ'],
```

```
&$coordinates[1]['υφ'],$mykmlfilename_,$headersfilename_, $distance_m,
$myElevationsFile, $myCaloriesFile, $usertag_, $mykmlfilename2_,
$headersfilename2_);


?>
```