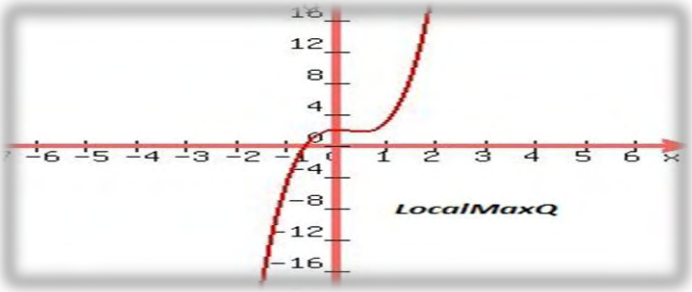
Στον Τάσο και τον Χριστόφορο...

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

Τμήμα Μηχανικών Ηλεκτρονικών Υπολογιστών Τηλεπικοινωνιών και Δικτύων

Διπλωματική Εργασία

Εφαρμογή LocalMaxQuadratic σε iphone



Σωτήρχου Σωτηρία –Τριανταφυλλιά

Επιβλέποντες καθηγητές:

Ακρίτας Αλκιβιάδης,

Ασπασσία Δασκαλοπούλου

Φλεβάρης 2011

Στον Τάσο και τον Χριστόφορο...

ΠΡΟΛΟΓΟΣ

Η παρούσα διπλωματική εργασία με θέμα «Υλοποίηση του LocalMaxQuantratic αλγορίθμου σε εφαρμογή iphone/ipad» αποτελεί τον επίλογο των προπτυχιακών μου σπουδών στο τμήμα Μηχανικών Ηλεκτρονικών Υπολογιστών Τηλεπικοινωνιών και Δικτύων του Πανεπιστημίου Θεσσαλίας.Το περιεχόμενο της εργασίας σχετίζεται με τη δημιουργία μιας μαθηματικής εφαρμογής σε κινητό iphone .Η εργασία αυτή αποτελεί το επιστέγασμα των πολύτιμων γνώσεων που είχα την τύχη και τη χαρά να αποκτήσω σε όλο αυτό το διάστημα.

Πριν την παρουσίαση της εργασίας μου αισθάνομαι την ανάγκη να ευχαριστήσω όλους εκείνους που μου συμπαραστάθηκαν και με στήριξαν σε όλη τη διάρκεια των προπτυχιακών μου σπουδών.

Αρχικά θέλω να ευχαριστήσω τον Καθηγητή του τμήματος Μηχανικών Ηλεκτρονικών Υπολογιστών Τηλεπικοινωνιών και Δικτύων και επιβλέπων καθηγητή της διπλωματικής μου, κύριο Αλκιβιάδη Ακρίτα.Τον ευχαριστώ θερμά για τις συμβουλές και την καθοδήγησή του καθώς επίσης και για την ευκαιρία που μου έδωσε να ασχοληθώ με ένα τόσο ενδιαφέρον και σύγχρονο θέμα.

Αισθάνομαι την ανάγκη να ευχαριστήσω και την καθηγήτρια του τμήματος, κυρία Δασκαλοπούλου Ασπασσία για όλες τις γνώσεις που μου προσέφερε όλα αυτά τα χρόνια καθώς επίσης και για την τιμή που μου έκανε να είναι η δεύτερη επιβλέπων καθηγήτρια της διπλωματικής μου.

Θέλω ακόμα να ευχαριστήσω θερμά τον φίλο και συμφοιτητή του τμήματος, Μητσιγιώργη Πέτρο για τη βοήθεια , τις παρατηρήσεις και τις συμβουλές που μου προσέφερε χωρίς τα οποία αυτή η διπλωματική δεν θα είχε ολοκληρωθεί επιτυχώς.

Κλείνοντας θέλω να ευχαριστήσω τους φίλους μου Αναστάσιο Τσογγίδη,Αναστάσιο Χιδερίδη και Χριστόφορο Κρόνη, για όλη την υλική και ψυχική υποστήριξή τους σε όλα αυτά τα χρόνια της προπτυχιακής μου πορείας.

Στον Τάσο και τον Χριστόφορο...

Περιεχόμενα

Στον Τάσο και τον Χριστόφορο...

Περίληψη

Η διπλωματική αυτή έχει ως στόχο την ανάπτυξη μιας μαθηματικής εφαρμογής για ένα iphone.Η μαθηματική αυτή εφαρμογή είναι η LocalMaxQuadratic η οποία βρίσκει τα μέγιστα και ελάχιστα ενός πολυωνύμου που ο ίδιος ο χρήστης εισάγει και τη γραφική παράσταση αυτού.Στην αρχή γίνεται μια εισαγωγή για το πως θα εγκατασταθεί το λειτουργικό της apple(mac) σε windows υπολογιστή,είτε κανονικά είτε εικονικά,στη συνέχεια εξηγείται ο κώδικας του αλγορίθμου και τέλος η γραφική παράσταση του πολυωνύμου, κάνοντας και μια εισαγωγή για το πως θα μπορούσαμε να χρησιμοποιήσουμε έτοιμες βιβλιοθήκες για την δημιουργία της γραφικής μας παράστασης.

Στον Τάσο και τον Χριστόφορο...

1 Εισαγωγή

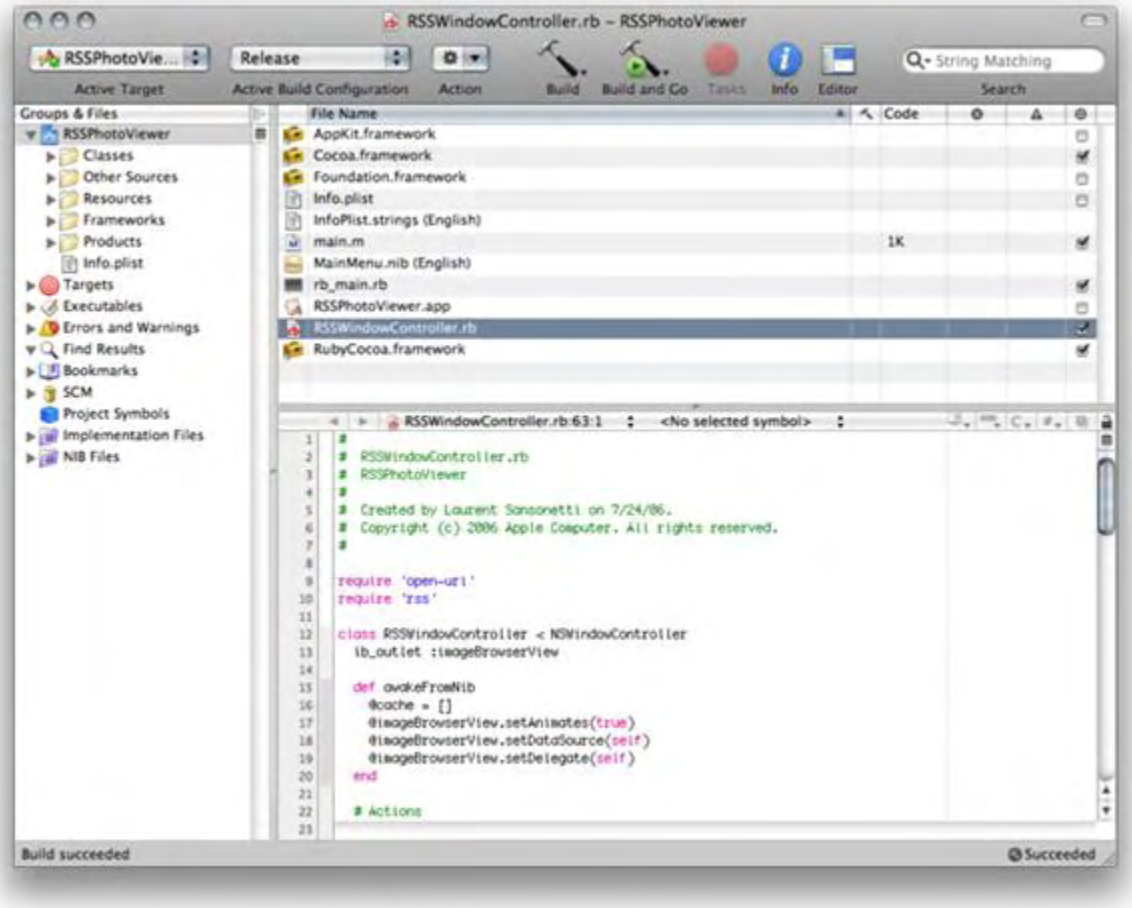
1.1 X Code the developer tool for building Mac OS X and iOS apps

Το Xcode είναι ένα εργαλείο για την ανάπτυξη Mac OS X και IOS εφαρμογών , απόλυτα συνδεδεμένο με Cocoa και Cocoa touch frameworks και αποτελεί μια απίστευτα παραγωγική πλατφόρμα ανάπτυξης εφαρμογών. Αποτελείται από τον Interface Builder , τον IOS simulator και τον IDE.

Στον Τάσο και τον Χριστόφορο...

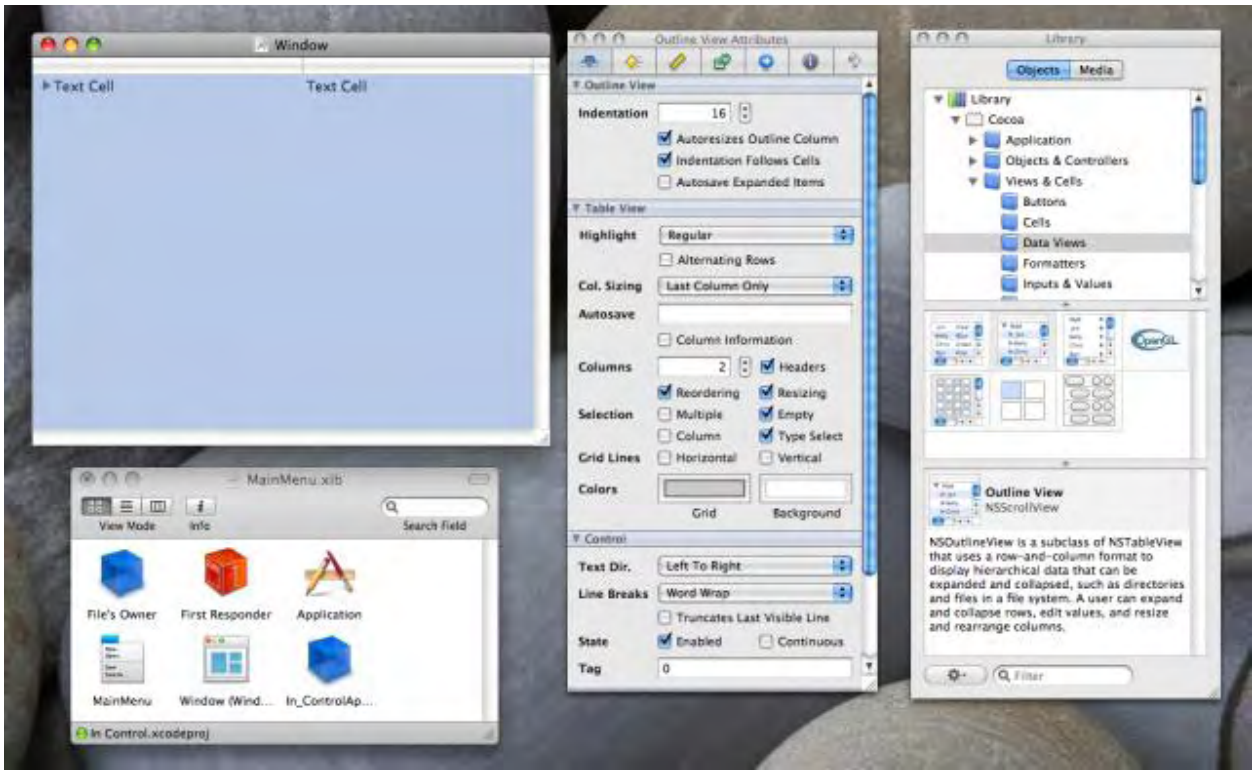
Το IDE είναι πλήρες εξοπλισμένο χτισμένο γύρω απο μια ομαλή ροή εργασίας που ενσωματώνει την επεξεργασία του πηγαίου κώδικα με την μεταγλώτιση και την εκτέλεση μέσα απο μία γραφική εμπειρία εντοπισμού σφαλμάτων – όλα χωρίς την έξοδο απο την προβολή του πηγαίου κώδικα μας. Με την έλευση του IOS SDK, το Xcode μπορεί να διαχειριστεί πλέον όλες τις συσκευές ελέγχου μας, την αυτόματη φόρτωση των iPhone εφαρμογών με τα κατάλληλα πιστοποιητικά, καθώς και την εγκατάσταση εφαρμογών για το iPhone. Ο απομακρυσμένος debugger συνδέεται με τη συσκευή σε πραγματικό χρόνο, και διαχειρίζεται τα breakpoints καθώς η εφαρμογή ελέγχεται από τη συσκευή.

Στον Τάσο και τον Χριστόφορο...



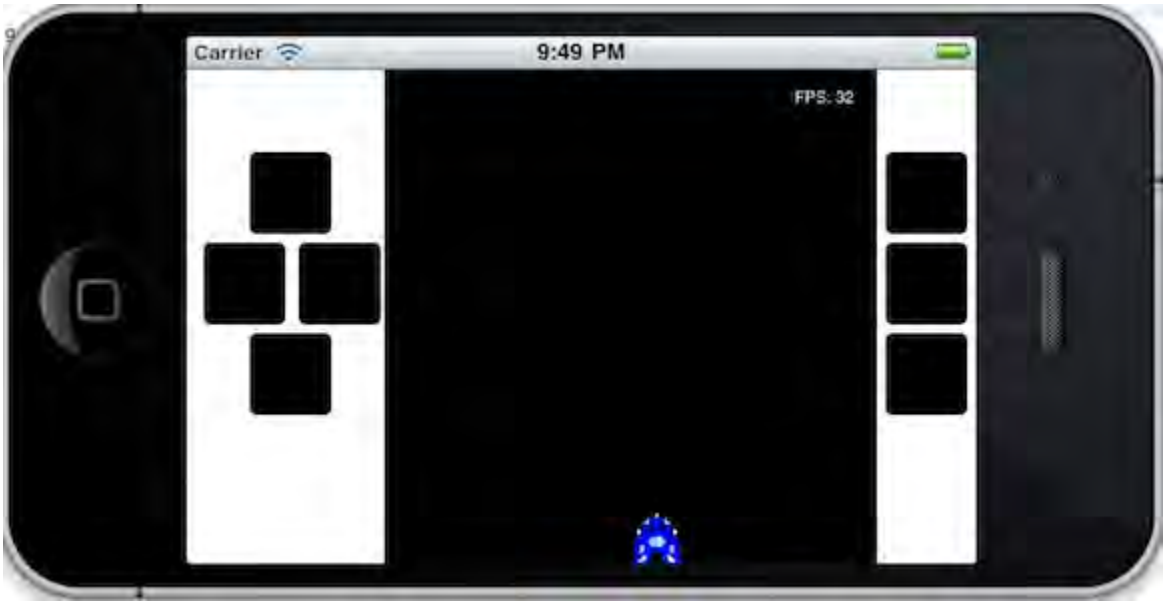
Ο Interface Builder είναι ένας εύκολος στη χρήση, γραφικός συντάκτης για τον σχεδιασμό κάθε πτυχής του γραφικού περιβάλλοντος της iphone εφαρμογής μας. Ο Interface Builder αποθηκεύει τη σχεδίαση της διεπαφής του χρήστη σε περισσότερα απο ένα αρχεία εφαρμογών ως ένα σύνολο απο τα αντικείμενα διεπαφής και τις σχέσεις τους.’Ο,τι αλλαγές γίνονται στον Interface Builder συγχρονίζονται αυτόματα με το Xcode.

Στον Τάσο και τον Χριστόφορο...



Τέλος, ο IOS Simulator τρέχει την εφαρμογή όπως ακριβώς μια IOS συσκευή.Έτσι μας δίνει την δυνατότητα να ελέξουμε αν το γραφικό περιβάλλον της εφαρμογής «δουλεύει» με τον τρόπο που θέλουμε, αν οι κλήσεις του δικτύου μας είναι σωστές και

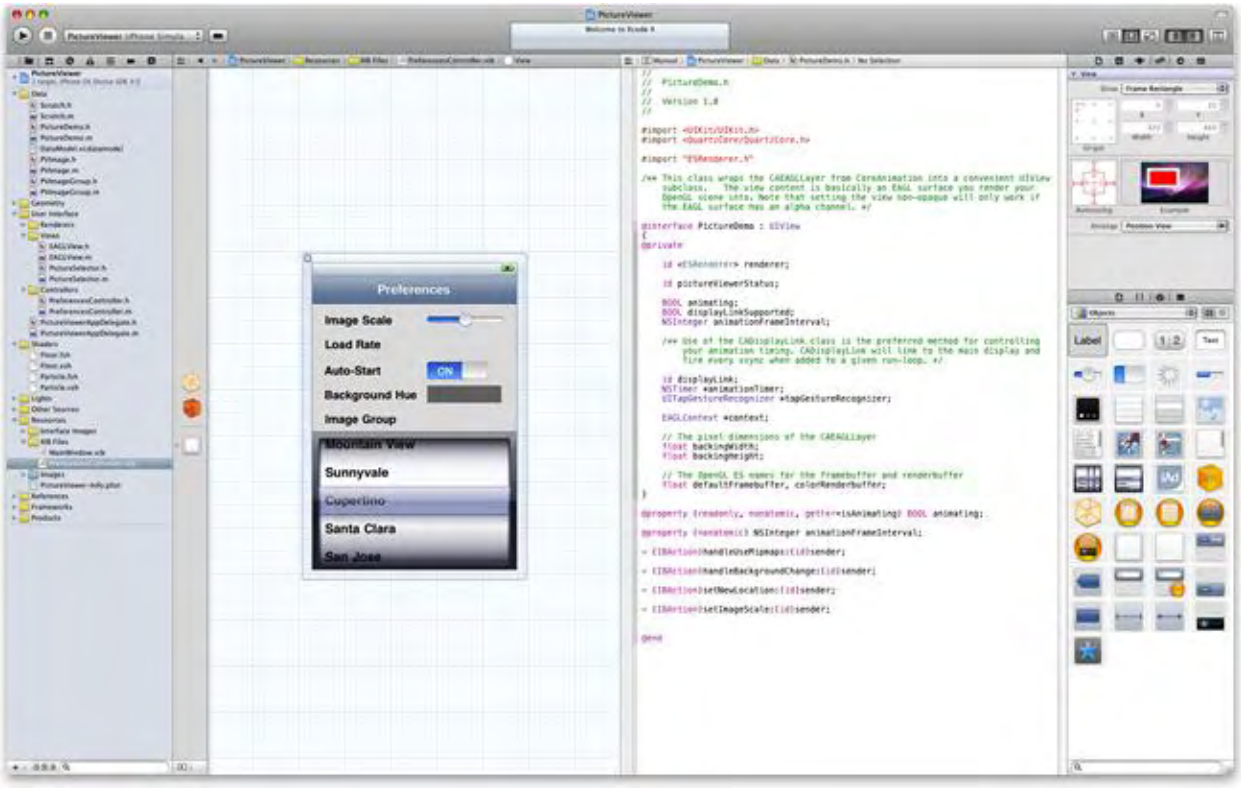
αν το γραφικό περιβάλλον αλλάζει σωστά καθώς περιστρέφεται το τηλέφωνο.



1.1.1 What's new..Xcode 4..

Στο Xcode 4, έχουν επανασχεδιαστεί τα εργαλεία ώστε να είναι ταχύτερο, πιο εύκολο στη χρήση, και πιο χρήσιμο από ποτέ. Το Xcode IDE καταλαβαίνει κάθε λεπτομέρεια του έργου μας, εντοπίζει λάθη τόσο συντακτικά όσο και λογικά, και θα μας διορθώσει ακόμη και τον κωδικό για εμάς. Πολύ απλά, το Xcode 4 θα μας βοηθήσει να γράψουμε καλύτερο κώδικα.Έχει ένα ολοκαίνουργιο περιβάλλον χρήστη, που χτίστηκε πάνω σε αποδεδειγμένες τεχνολογίες που η ίδια η Apple χρησιμοποιεί για να χτίσει το Mac OS X και IOS, και τα οποία έχουν παράξει πάνω από μισό εκατομμύριο Mac OS X και IOS εφαρμογές.

Στον Τάσο και τον Χριστόφορο...



Όπως φαίνεται και απο την εικόνα , όλα τα βασικά παράθυρα που χρησιμοποιούνται για την ανάπτυξη της εφαρμογής , βρίσκονται μαζεμένα σε ένα παράθυρο.

1.2 Εισαγωγή στην Objective C

Η γλώσσα που χρησιμοποεί το Xcode είναι η Objective C ,μια αντικειμενοστραφής γλώσσα με ένα δυναμικό σύστημα κλάσεων το οποίο είναι εξαιρετικά γρήγορο και ευέλικτο, σχεδιασμένο ως ένα υπερσύνολο πάνω απο την γνωστή και τυποποιημένη γλώσσα C.Η Objective C χρησιμοποιώντας cocoa και Cocoa Touch frameworks , παρέχει υψηλού επιπέδου APIs τα οποία απελευθερώνουν μια απίστευτη λειτουργικότητα απο λίγες και κατανοητές γραμμές κώδικα. Τέλος περιλαμβάνει τέτοια μοντέρνα χαρακτηριστικά όπως για παράδειγμα γρήγορη επανάληψη και garbage collection συντονισμένα για τα σημερινά Mac.

Βασικές εντολές της Objective C :

Για να ορίσετε ένα αντικείμενο απλά πρέπει να γράψετε το όνομα της κλάσης και ένα όνομα που θα χρησιμεύσει ως το τοπικό δείκτη στο αντικείμενο:

```
NSNumber numberObject* ?
```

Το «NSNumber» είναι μια κοινά χρησιμοποιούμενη κλάση στην Objective-C και εδώ το αντικείμενό μας είναι το numberObject. Ο αστερίσκος χρησιμοποιείται για να δείξει ότι το όνομα του αντικειμένου είναι ένας δείκτης.

Instantiating Αντικείμενα σε Objective-C

Πριν να χρησιμοποιήσουμε ένα αντικείμενο σε Objective-C θα χρειαστεί να δημιουργήσουμε ένα τοπικό στιγμιότυπο του αντικειμένου. Για να δημιουργήσουμε το ίδιο το αντικείμενο, θα χρησιμοποιήσουμε ένα constructor που έχει συνήθως το πρόθεμα τη λέξη "init" ή απλά init.

```
myClass * myobject = [[myClass Alloc] init];
```

Στον Τάσο και τον Χριστόφορο...

Απελευθέρωση Αντικειμένων

Όταν χρησιμοποιείτε Objective-C για το iPhone θα πρέπει να απελευθερώσουμε τα αντικείμενα που έχουμε δημιουργήσει χρησιμοποιώντας alloc, new ή copy. Αυτό που πρέπει να κάνουμε είναι να«στέλνουμε» το μήνυμα «release» προς το αντικείμενο.

```
[myObject release];
```

Γενικά το μοτίβο που πρέπει να ακολουθήσουμε όταν χρησιμοποιούμε αντικείμενα σε Objective-C είναι :

```
/ / Εκχώρηση μνήμης και τη δημιουργία αντικειμένου
myClass  *myobject = [[myClass alloc] init]?
/ / Χρήση αντικειμένου
[myObject doStuff];
//Απελευθέρωση αντικειμένου
[myObject release];
```

Διαχείριση μνήμης και Καταμέτρηση αναφοράς

Όταν εργαζόμαστε με Objective-C για το iPhone θα πρέπει να διαχειριστούμε τη μνήμη χειροκίνητα. Αυτό είναι ένα πολύ λεπτομερές και σημαντικό θέμα και, όταν η διαχείριση μνήμης δεν έχει γίνει σωστά τότε θα μπορούσε να οδηγήσει σε διαρροές μνήμης ή σε « πάγωμα» των εφαρμογών . Η διαχείριση μνήμης είναι ένα απλό σύστημα που ονομάζεται «καταμέτρηση αναφοράς».Η ιδέα είναι ότι το σύστημα θα παρακολουθεί κατά πόσον πρέπει να διατηρηθεί η μνήμη για ένα αντικείμενο που διατίθενται με βάση τον αριθμό των άλλων στοιχείων που δείχνουν ότι θέλουν το αντικείμενο για να μείνει. Κάθε στοιχείο πρέπει να δηλώσει το ενδιαφέρον του για το αντικείμενο με την προσθήκη μιας αναφοράς (μερικές φορές καλείται «retain») που προσμετρούνται στο αντικείμενο. Έτσι, αν για παράδειγμα 5 συστατικά ενδιαφέρονται για το « myobject», τότε το «myobject» μετρά 5 αναφορές. Το σύστημα θα διατηρήσει τη μνήμη του αντικείμενου μας στη θέση του για όσο διάστημα ο αριθμός αναφοράς είναι πάνω από 0.Όταν τα στοιχεία μας δεν χρειάζονται πλέον ένα αντικείμενο , θα

αφαιρέσει το ενδιαφέρον τους για το αντικείμενο, στέλνοντας ένα μήνυμα «release» προς το αντικείμενο. Κάθε φορά που ένα μήνυμα απελευθέρωσης αποστέλλεται σε ένα αντικείμενο , μειώνεται η αναφορά κατά 1. Μόλις ο αριθμός αναφοράς φτάσει το 0, το σύστημα μπορεί να καταστρέψει το αντικείμενο και την εκ νέου χρήση της μνήμης, ανά πάσα στιγμή.

Κλάσεις που χρησιμοποιούνται συχνά στην Objective-C:

Αρχικά είναι η **NSString** η οποία χρησιμοποιείται για την δήλωση αντικειμένου string, η **NSLog** η οποία εκτυπώνει στην οθόνη ή στην κονσόλα ένα μήνυμα,για παράδειγμα:

```
// Δήλωση/Δέσμευση αντικειμένου String
  NSString * myString = [[NSString alloc] initWithString: @ "A String"];
 //Εκτύπωση αντικειμένου
  NSLog (myString);
 //Απελευθέρωση (αντικειμένου)String
  [myString  release];
```

Η **NSNumber** η οποία χρησιμοποιείται όταν διαπραγματευόμαστε αριθμούς με ένα συγκεκριμένο τρόπο, η **NSMutableArray** για τη δήλωση πινάκων **.** Οι πίνακες είναι απλά λίστες από αντικείμενα και στην ObjectiveC μπορούμε να βάλουμε οτιδήποτε σε έναν πίνακα και μπορούμε ακόμη και να αναμίξουμε και να ταιριάξουμε αντικείμενα. Ένας τρόπος για να χρησιμοποιήσουμε έναν πίνακα είναι:

```
    // Αρχικοποίηση ενός πίνακα
NSMutableArray * myArray = [[NSMutableArray Alloc] init];
   // Προσθήκη στοιχείων σε έναν πίνακα
[myArray addObject: @ "Στοιχείο 1"];
[myArray addObject:  @ "Στοιχείο 2"];
[myArray addObject:  @ "Στοιχείο 3"];
  //Ανάκτηση του πρώτου αντικειμένου απο έναν πίνακα
NSLog(myArray objectAtIndex:0);
  //Ανάκτηση του τελευταίου αντικειμένου απο έναν πίνακα
```

```
NSLog(myArray lastObject);
```

Φυσικά, στο τέλος θα πρέπει να απελευθερώσουμε το αντικείμενο πίνακα, όταν

τελειώσουμε..

```
[release myArray];
```

Properties

Τα **Properties** είναι ένα χαρακτηριστικό της Objective-C που θα μας επιτρέψουν να δημιουργήσουμε αυτόματα «accessors».Ο τρόπος που χρησιμοποιούμε τα **Properties** στον κώδικα είναι :

Στο header file:

```
@interface Photo : NSObject {

       NSString* caption;
      NSString* photographer;

}

   @property (retain) NSString* caption;
   @property (retain) NSString* photographer;


@end
```

Το ***@ property*** είναι μια εντολή της γλώσσας που δηλώνει το **property**. Το «retain» στην παρένθεση διευκρινίζει ότι ο «setter» θα πρέπει να διατηρήσει την αξία των

αρχικών τιμών , καθώς και το υπόλοιπο της γραμμής και απλά διευκρινίζει το είδος και το όνομα της ιδιότητας. Τώρα ας ρίξουμε μια ματιά στην εφαρμογή της κλάσης:

```
#import "Photo.h"
@implementation Photo

 @synthesize caption;
 @synthesize photographer;


- (void) dealloc
{
 [caption release];
 [photographer release];
[super dealloc];
 }

 @end
```

Η **@ synthesize** δημιουργεί αυτόματα τους setters και getters για εμάς, έτσι το μόνο που έχουμε να κάνουμε είναι να εφαρμόσουμε για αυτή την κλάση την μέθοδο dealloc.

Οι Accessors μόνο θα δημιουργηθούν εάν δεν υπάρχουν ήδη, έτσι καθορίζουμε @ synthesize για ένα@property , στη συνέχεια, εφαρμόζουμε setter ή getter εάν θέλουμε. Ο compiler θα συμπληρώσει όποια μέθοδος λείπει.

1.3 Εγκατάσταση του VMware και προβλήματα

Πολλοί αγοραστές υπολογιστών επιλέγουν να αγοράσουν ένα Mac όχι μόνο στη χρήση του OS X αλλά και αντί των windows.Αυτό είναι ένα ισχυρό πλεονέκτημα που έχουν Macs στον πόλεμο Mac vs PC. Οι υπολογιστές της Apple είναι σε θέση να εκτελούν τα Windows, είτε μέσω Bootcamp ή εικονικές μηχανές. Έτσι, γιατί δεν είναι σε θέση το Windows PC να τρέχει OS X; Λοιπόν, τα Windows PC είναι σε θέση να Τρεξουν OS X. Αλλά, είναι ενάντια στην συμφωνία αδειοδότησης της Apple να το πράξει. Εάν θέλoyme να εκτελέσoyme OS X με τον καλύτερο δυνατό τρόπο, καλύτερα να πάρουμε ένα Mac(!!).Για να λειτουργήσει

αυτό, ο υπολογιστής μας πρέπει να διαθέτει έναν επεξεργαστή που υποστηρίζει virtualization. Αν δεν εχει, μπορεί να λάβουμε σφάλματα.Οποιοδήποτε νεότερη γραμμή επεξεργαστων: Core 2 Duo, Quad Core, Core i7 θα πρέπει όλα να δουλεύουν μια χαρά.

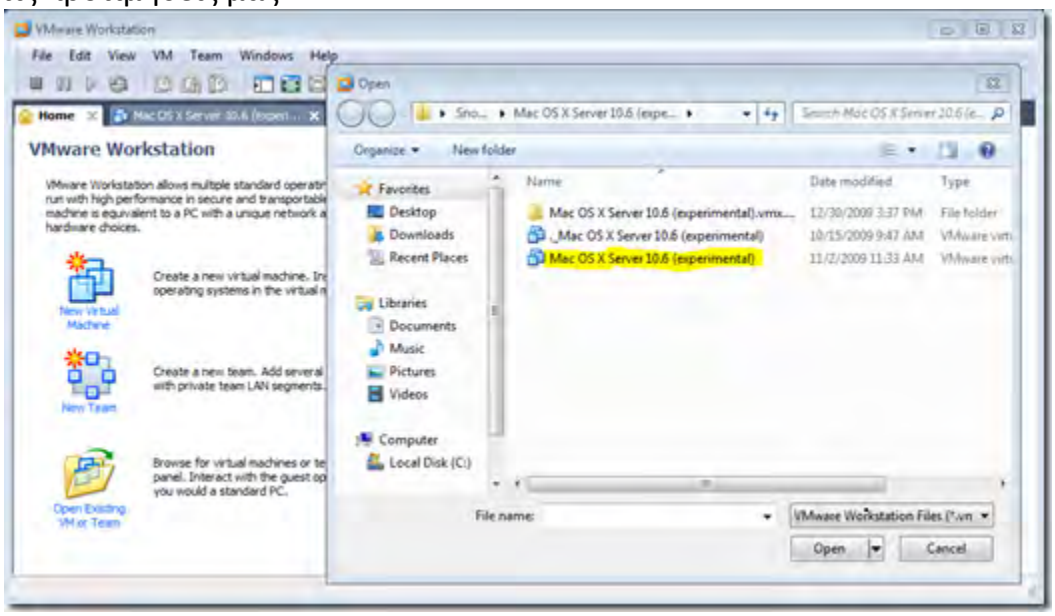
Αλλά, μην περιμένουμε να μην υπάρχουν σφαλματα, αν χρησιμοποιούμε παλιότερο Pentium 4 μηχάνημα. Ο επεξεργαστής πρέπει να είναι Intel και όχι amd.

Επειδή εμείς θα εγκαταστήσουμε το OS X στην κορυφή των Windows μέσω ενός εικονικού μηχανήματος, το πρώτο βήμα είναι να πάρουμε κάποιο λογισμικό εικονικής μηχανής. Για αυτό το demo, θα χρησιμοποιήσουμε VMware_v7.1.3_Lite μόνο 92 megabyte.(Στην αρχή βγάζει Ρώσσικα αλλά μετά ειναι OK)

Ώρα να ξεκινήσουμε την εγκατάσταση του Snow Leopard. Τρέχουμε το VMware Workstation χρησιμοποιώντας τη διαδρομή Έναρξη> Όλα τα προγράμματα> VMware> VMware Workstation.

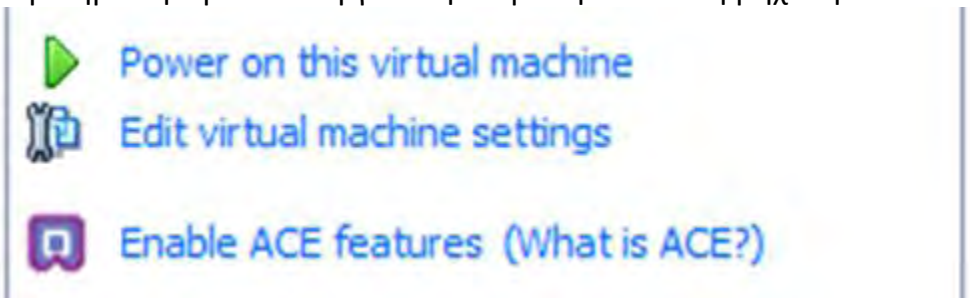
Επιλέγουμε file> Open. Πλοηγηγούμαστε στο Snowy_VM φάκελο, ανοίγουμε το επόμενο φάκελο που λέει "Mac OS X 10.6 Server (expiramental), και επιλέγουμε το 2KB αρχείο που ονομάζεται Mac OS X Server 10,6 (expiramental). Κάνουμε κλικ στο κουμπί ανοίγματος για να φορτώσει την εικονική μηχανή.

Από την καρτέλα Mac OS X Server στο παράθυρο του VMware Workstation, κάνουμε κλικ στο εικονίδιο Επεξεργασία ρυθμίσεων εικονικής μηχανής. Μπορούμε να προσαρμόσουμε το μέγεθος του σκληρού δίσκου, μνήμης, κτλ, με τις προτιμήσεις μας.



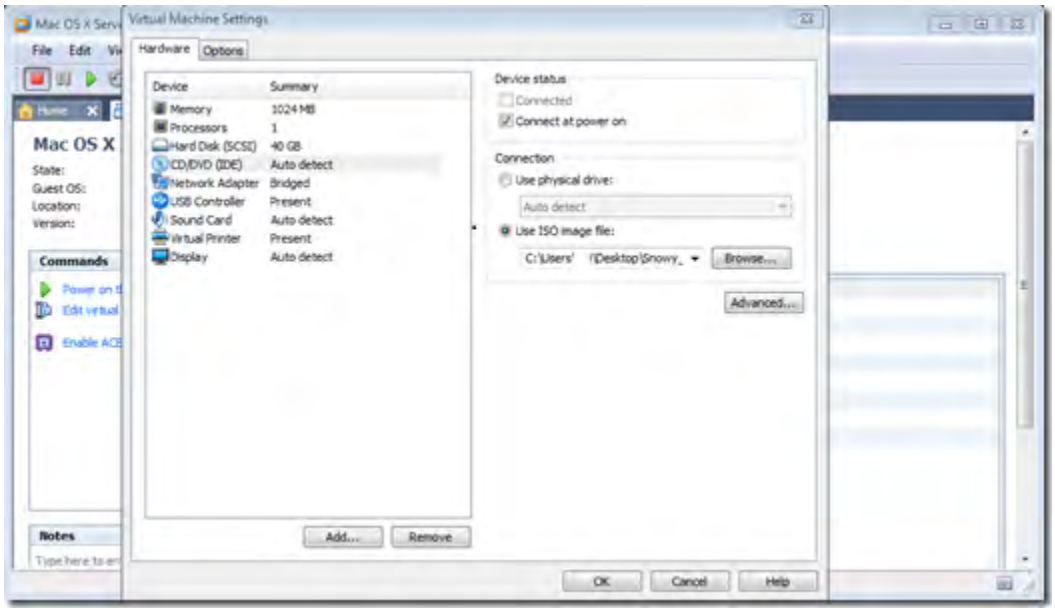
Επιλέγουμε το CD / DVD (IDE) και υπό τις Ρυθμίσεις σύνδεσης, επιλέγουμε χρήση ISO αρχείο εικόνας. Κάνουμε κλικ στο κουμπί Αναζήτηση και πηγαίνουμε στο Darwin_Snow.iso το οποίο βρίσκεται στο Snowy_VM φάκελό μας. Κάνουμε κλικ στο κουμπί Άνοιγμα για να επιλέξουμε την εικονική μηχανή.

Τώρα ήρθε η ώρα να ενεργοποιήσουμε την εικονική μηχανή.



Κάνουμε κλικ στο "Power on this virtual machine". Μόλις ξεκινήσει η διαδικασία, κρατάμε πατημένο το πλήκτρο F8 στο πληκτρολόγιό μας. Θα πρέπει να δούμε μαύρη οθόνη με μενού εκκίνησης.

Τώρα ήρθε η ώρα για την εγκατάσταση του πραγματικά Snow Leopard επάνω σε εκείνη την εικονική εικόνα δισκέτα μηχανής. Θα πρέπει να δούμε ορισμένες ρυθμίσεις εικονίδια που βρίσκονται στην κάτω δεξιά γωνία του παραθύρου VMware. Κάνουμε δεξί κλικ στο εικονίδιο που μοιάζει με ένα CD και επιλέγουμε τις ρυθμίσεις από το αναδυόμενο μενού.



Σύμφωνα με τη λίστα των συνδέσεων, αλλάζουμε τις επιλογές από την εικόνα ISO στην φυσική μονάδα δίσκου που θα βάλουμε το snow leopard. Αφού έχουμε κάνει αυτό, κάνουμε κλικ στο κουμπί Αποθήκευση και Έξοδος.

Τώρα ήρθε η ώρα να βάλουμε το dvd εγκατάστασης στην μονάδα δίσκου αν το κάψαμε,αλλιώς μπορούμε να το μπουτάρουμε με το daemon toolls.

Πηγαίνουμε πίσω στην μαύρη οθόνη εκκίνησης και επιλέγουμε την επιλογή «Εκκίνηση» από το DVD. Πατάμε το κουμπί F8 στο πληκτρολόγιό μας για τις προηγμένες επιλογές και πληκτρολογούμε "-v" χωρίς τα εισαγωγικά και πατάμε Enter. Τώρα, το VMware Workstation πρέπει να εκκίνησει το Snow Leopard

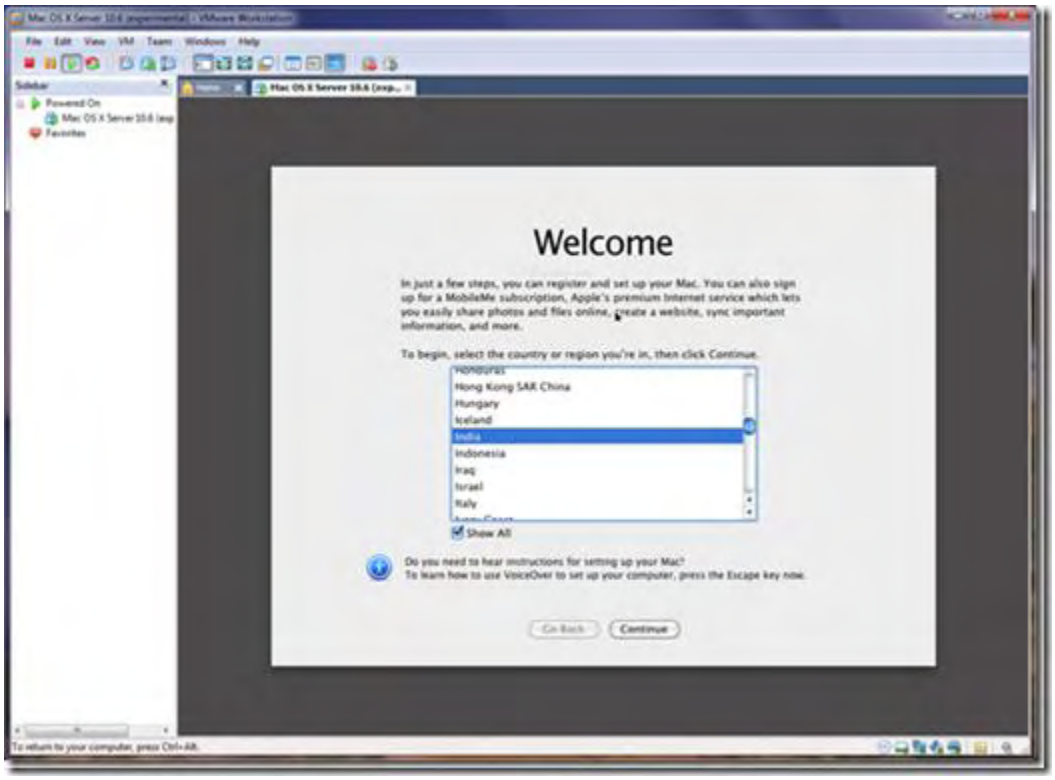
δίσκο εγκατάστασης. Μπορεί να χρειαστούν μερικά λεπτά για να φορτώσει.

Θα πρέπει τώρα να είναι στην κανονική οθόνη εγκατάστασης το Snow Leopard. Κάνουμε κλικ στο "continue" στην πρώτη οθόνη που εμφανίζεται. Στη δεύτερη οθόνη, κάνουμε κλικ στην επιλογή utilities στο παράθυρο της εγκατάστασης και να επιλέγουμε disk utility. Χρησιμοποιούμε το βοηθητικό δίσκο για να αλλάξουμε τον τύπο μορφής του σκληρού δίσκου σε OS X Extended (Journaled).Κάνουμε Έξοδο απο το disk utility και αφήνουμε το OS X να εγκατασταθεί πλήρως.

Μετά την εγκατάσταση του Snow Leopard, η εικονική μηχανή μας θα κάνει επανεκκίνηση. Μετά την επανεκκίνηση, θα χρειαστεί να κάνουμε κάποια πράγματα για να μπούμε στο Snow Leopard. Επιλέγουμε το CD / DVD options, όπως κάναμε πριν και επιλέγουμε την "Use ISO Image File" επιλογή.

Ανοίγουμε το the Darwin_Snow.iso και το επιλέγουμε.Τώρα κατά το άνοιγμα της εικονικής μηχανής,θα πρέπει να Bootαρει σε Snow Leopard.Μπορεί να χρειαστεί

να κάνουμε κάποια restarts μέχρι το Snow Leopard να φορτωθεί σωστά . Αφού φορτώσει, προσαρμόζουμε τις ρυθμίσεις μας μέσα από το Snow Leopard.Κάνουμε Save και απολαμβάνουμε το καινούριο μας Mac.



Χαρακτηριστικά:
Το snow Leopard 6.22 Gygabyte
Εκδοση:10a4110(Multilanguage)
Και μονο για INTEL Επεξεργαστές

1.4 Εγκατάσταση των Mac (apple) σε windows υπολογιστή

Η εγκατάσταση των Mac σε windows υπολογιστή δεν είναι τόσο δύσκολη όσο φαίνεται.Αυτό που χρειάζεστε είναι ένα cd με τα Snow Leopard τα οποία και εύκολα μπορείτε να κατεβάσετε απο οποιοδήποτε site με torrents γνωρίζουμε.Αν και τα Snow Leopard έχουν απο μόνα τους boot loader , εγώ χρησιμοποίησα τον Acronis boot loader τον οποίο και εγκατέστησα στα windows πριν ξεκινήσω οποιαδήποτε απόπειρα να εγκαταστήσω τα Snow Leopard.Σημαντικό επίσης είναι να έχουμε και ένα «αρκετά» μεγάλο partition στον σκληρό δίσκο ώστε να χωρέσουν τα Snow Leopard αλλά και όποια άλλα προγράμματα θελήσουμε να εγκαταστήσουμε.Τα Snow Leopard χρειάστηκαν γύρω στα 6gb αλλά μαζί με την εγκατάσταση του Xcode έφτασαν τα 13gb.Εγώ έκανα ένα partition 20gb για να είμαι σίγουρη.Έτσι λοιπόν, τοποθετούμε το cd με τα Snow Leopard και πατάμε boot from cd, αυτό που εμφανίζεται στην οθόνη είναι:

Στον Τάσο και τον Χριστόφορο...

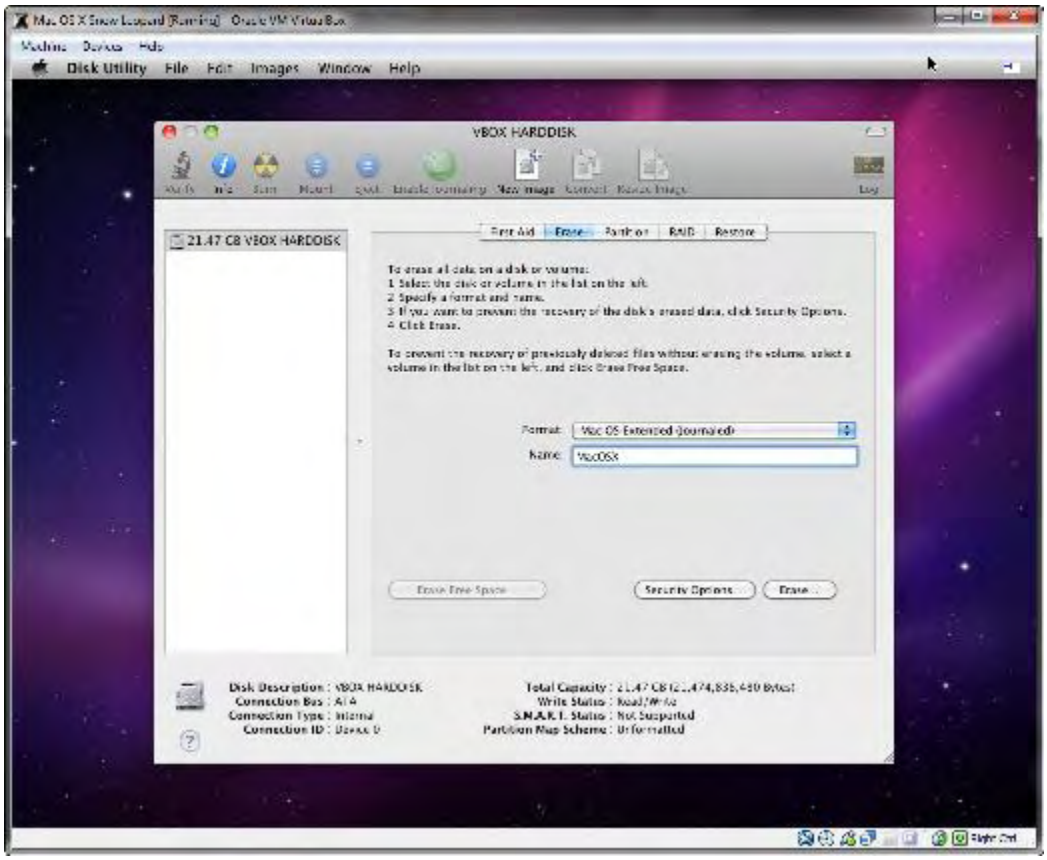
Πρώτα απο όλα μας ζητείται να διαλέξουμε την γλώσσα στην οποία θα εγκατασταθούν τα Mac .Όπως βλέπετε στην οθόνη δεν υπάρχει η ελληνική γλώσσα(και γιατί να υπάρχει άλλωστε! !!)οπότε πατάμε το «**Use English for the Mac Language»** και το βελάκι για να συνεχίσουμε.Στην συνέχεια εμφανίζεται στην οθόνη το :

Οπότε και πατάμε «**continue**» και στη συνέχεια :**Utilities<Disk Utility**

Στον Τάσο και τον Χριστόφορο...

και θα εμφανιστεί στην οθόνη μας :

Στον Τάσο και τον Χριστόφορο...



Όπου και θα πρέπει να επιλέξουμε Erase<Format(Mac OS Extended journalized) και Name(Mac OS) και τέλος «Erase» για να σβηστούν ότι έχουμε στο partition που δημιουργήσαμε πιο πριν για να εγκαταστήσουμε τα Mac .Εμφανίζεται το εικονίδιο του partition μας :

Στον Τάσο και τον Χριστόφορο...

Το πατάμε και **προτού** πατήσουμε install πρέπει να πατήσουμε customize για να εγκαταστήσουμε τα απαραίτητα kexts για να δουλέψουν καλά τα mac.Τα kexts που χρησιμοποίησα για τον acer7720g είναι:

PC EFI 10,5 - Chameleon 2 RC3 (R658) PC EFI
- NullCPUPM - IntelCPUPowerManagement
- PlatformUUID - Erreur 35 fix
- FakeSMC
- IOATA Kernel Panic fix - IOATAFamily.kext 32/64 de Leo 10.5.8
- EvOreboot - Redémarrez et fixer l'arrêt
- EvOSpeedStep - για Core i7
- EvOenabler - vidéo ATI
- SMBIOSResolver - για PC EFI 9.4!
- Legacy AHCI - 32/64 kext Legacy Intel AHCI (SATA)
- Legacy ATA - 32/64 kext Legacy ATA processeur Intel (IDE)
- Legacy JMicron ATA - 32/64 kext Legacy ATA JMicron (IDE)

Στον Τάσο και τον Χριστόφορο...

- PS / 2 - kext pour PS / 2
- Rosetta
- ATools - outils

Για την κάρτα γραφικών και την κάρτα Ethernet εγκατέστησα kexts αλλά δεν δούλεψε τίποτα. Ίσως εσείς να είστε πιο τυχεροί.Τέλος πατάμε install και επιτέλους έχουμε τα mac σε windows υπολογιστή.Κάνουμε restart και θα εμφανιστεί και ο acronis loader.Ένα άλλο πρόβλημα που είχα στον acer είναι ότι μόλις έβαλα το xcode μου χάλασε το πληκτρολόγιο και το ποντίκι.Ίσως έφταιγαν τα kexts που άλλαξαν. Καλή δουλειά !

.

Στον Τάσο και τον Χριστόφορο...

2 Ο Αλγόριθμος LocalMaxQuadratic

2.1 Περιγραφή αλγορίθμου

Ο αλγόριθμος LocalMaxQuadratic χρησιμοποιείται για να βρίσκουμε τα τοπικά ελάχιστα και μέγιστα μιας συνάρτησης.Όπως είναι γνωστό για να τα βρούμε ακολουθούμε τα εξής βήματα:

1)Βρίσκουμε την παράγωγο της συνάρτησης

2)Λύνουμε την συνάρτηση με το 0 και βρίσκουμε για ποιο χ μηδενίζεται

3)Για να βρούμε αν είναι τοπικό μέγιστο ή ελάχιστο, βρίσκουμε την δεύτερη παράγωγο και παρατηρούμε που αλλάζει το πρόσημο της συνάρτησης στο χ που βρήκαμε προηγουμένως.Αν είναι αρνητική έχουμε τοπικό ελάχιστο, ενώ αν είναι θετική είναι τοπικό μέγιστο.

Στον Τάσο και τον Χριστόφορο...

Ο αλγόριθμος του Local Max σε mathematica είναι ο εξής:

```
LMQ[p_] := Module[{cl = CoefficientList[p, Variables[p]], timesused, len, m, n, q = -1, tempmin, tempmax = 0,
   timestart = TimeUsed[]},
  If[Last[cl] < 0, cl = -cl];
  len = Length[cl];

  If[len <= 1, Return[0]];
  timesused = Table[1, {len}];

  Do[
   If[cl[[m]] < 0,
    (*then*)
    tempmin = ∞;
    Do[
     If[cl[[n]] > 0,
      q = N[(2^(timesused[[n]]++) -cl[[m]]/cl[[n]])^(1/(n-m))];
      If[q < tempmin, tempmin = q]
     ],
     {n, len, m+1, -1}
    ];
    If[tempmax < tempmin, tempmax = tempmin]
   ],
   {m, len-1, 1, -1}
  ];
  {tempmax, TimeUsed[] - timestart}]
```

Στην αρχή λοιπόν στον πίνακα cl αποθηκεύονται όλοι οι συντελεστές του πολυωνύμου.Αν ο συντελεστής της μεγαλύτερης δύναμης είναι αρνητικός, αλλάζουν τα

πρόσημα στον πίνακα cl (cl=- cl).Στη συνέχεια ελέγχουμε το πρόσημο καθενός απο τους συντελεστές, αν είναι αρνητικός το τοπικό ελάχιστο γίνεται άπειρο, αν είναι θετικός τότε $q = N\left[\left(2^{\text{timesused}[[n]]++\frac{-\text{cl}[[m]]}{\text{cl}[[n]]}}\right)^{1/(n-m)}\right]$,όπου timesused ένας πίνακας με μέγεθος ίσο με την δύναμη του πολυωνύμου κι αρχικοποιημένος με ένα για την κάθε θέση του. Αν q<tempmin,tempmin=q και αν tempax<tempin,τότε tempmax=tempin.

Ο αλγόριθμος για το τοπικό ελάχιστο είναι :

```
posLbdLMQ[p_, x_] :=
 Module[{cl = CoefficientList[p, x], timesused, len, m, n, q = -1
  If[First[cl] < 0, cl = -cl];
  len = Length[cl];

  If[len <= 1, Return[0]];
  timesused = Table[1, {len}];

  Do[
   If[cl[[m]] < 0,
    (*then*)
    tempmin = ∞;
    Do[
     If[cl[[n]] > 0,
      q = N[(2^(timesused[[n]]++) (-cl[[m]])/cl[[n]])^(1/(m-n))];
      If[q < tempmin, tempmin = q]
     ],
     {n, 1, m - 1, 1}
    ];
    If[tempmax < tempmin, tempmax = tempmin]
   ],
   {m, 2, len, 1}
  ];
  2^(-Ceiling[Log[2, tempmax]])]
```

Στον οποίο χρησιμοποιούμε τους ίδιους πίνακες για τους συντελεστές και τις ίδιες μεταβλητές, μόνο που τώρα ελέγχουμε αν ο συντελεστής της μηδενικής δύναμης του χ είναι αρνητικός για να αλλάξουμε τα πρόσημα στον πίνακα συντελεστών, ενώ για να βρούμε το tempmin χρησιμοποιούμε την εντολή

`2^(-Ceiling[Log[2, tempmax]])` η οποία παίρνει τον δυαδικό λογάριθμο του **tempax** και τον υψώνει στο τετράγωνο.

2.2 Περιγραφή του κώδικα

Για να φτιάξουμε τον αλγόριθμο του LocalMaxQuadratic σε Objective C δημιουργήσαμε 4 αρχεία,LMQBrain.h,LMQBrain.m,LocalMaxQuadraticViewController.h και LocalMaxQuadraticViewController.m(2 αρχεία header και δύο .m).Το αρχείο LMQBrain.h είναι το header file της κλάσης μας.

```
//
//  LMQBrain.h
//  LocalMaxQuadratic
//
//  Created by filioui on 12/23/10.
//  Copyright 2010 __MyCompanyName__. All rights reserved.
//

#import <Foundation/Foundation.h>


@interface LMQBrain : NSObject {
    int *cl,*clu,*cll,*timesused;
    int i,flag,len,cmax,ctemp,y,m,coeff,n,temp1,temp2,powCoef;
    double temp3,temp4,q,tempmax,tempmin,ub,lb;
    char chr,chrr;
}

@property (readonly) int *cl;
@property (readonly) int cmax;

- (void)getCoeffs: (NSString *)equation;
- (double)getUB;
- (double)getLB;

@end
```

Στο αρχείο αυτό κάνουμε :

Στον Τάσο και τον Χριστόφορο...

#import <Foundation/Foundation.h> που είναι το header file που περιέχει όλες τις κλάσεις της γλώσσας μας(υπερκλάση).Στη συνέχεια γράφουμε @interface LMQBrain(το όνομα της κλάσης μας):NSObject(η υπερκλάση μας) και δηλώνουμε όλες τις μεταβλητές(cl,clu,cll κλπ) που θα χρησιμοποιήσουμε στο model αρχείο μας για να δημιουργήσουμε τον αλγόριθμο.Επίσης δηλώνουμε και τις συναρτήσεις που θα υλοποιήσουμε στην συνέχεια,την getCoeffs η οποία επιστρέφει τους συντελεστές του πολυωνύμου,την getUB η οποία επιστρέφει το τοπικό μέγιστο και την getLB η οποία επιστρέφει το τοπικό ελάχιστο.Οι πιο σημαντικές μεταβλητές(instance variables) για να αναφερθούν είναι οι :

cl:πίνακας με τους συντελεστές του πολυωνύμου

clu: πίνακας με τους συντελεστές του πολυωνύμου για το τοπικό μέγιστο

cll: πίνακας με τους συντελεστές του πολυωνύμου για το τοπικό ελάχιστο

timesused: πίνακας αρχικοποιημένος με 1 για όλους τους συντελεστές

cmax:μεταβλητή που «κρατάει» την μεγαλύτερη δύναμη

len:το μέγεθος των πινάκων(στην ουσία η τιμή της μεγαλύτερης δύναμης)

tempmin:προσωρινή μεταβλητή για το τοπικό ελαχιστο

tempmax: προσωρινή μεταβλητή για το τοπικό μέγιστο

ub:μεταβλητή που αποθηκεύει το τοπικό μέγιστο(upper bound)

lib: μεταβλητή που αποθηκεύει το τοπικό μέγιστο(lower bound)

powCoef:η δύναμη του πολυωνύμου

Στο implementation αρχείο LMQBrain.m έχουμε:

```
//
//  LMQBrain.m
//  LocalMaxQuadratic
//
//  Created by filioui on 12/23/10.
//  Copyright 2010 __MyCompanyName__. All rights reserved.
//

#import "LMQBrain.h"

@implementation LMQBrain

@synthesize cl;
@synthesize cmax;

- (void)brainInit {
    i=0;flag=1;len=0;cmax=0;ctemp=0;y=0;
    m=-1;coeff=0;n=0;temp1=0;temp2=0;powCoef=0;
    temp3=0.0;temp4=0.0;q=-1.0;tempmax=0.0;
    tempmin=0.0;ub=0.0;lb=0.0;
}

- (void)getCoeffs: (NSString *)equation {

    self.brainInit;

    NSString *tempString1 = [[NSString alloc] initWithFormat:equation];
    NSString *tempString = [[tempString1
componentsSeparatedByCharactersInSet: [NSCharacterSet
whitespaceCharacterSet]] componentsJoinedByString: @""];
    NSCharacterSet *doNotWant = [NSCharacterSet
characterSetWithCharactersInString:@"~!@#$%&*()"];
    tempString = [[tempString componentsSeparatedByCharactersInSet:
doNotWant] componentsJoinedByString: @""];

    len= [tempString length];

    char tempMatrix[len];

    for(i=0,y=i+1; (i<len); i++,y++)
    {
      chr=(char )[tempString characterAtIndex:i];

      if(chr =='^')
      {
            chrr=(char)[tempString characterAtIndex:y];

            flag=0;
            ctemp= chrr-'0';
```

```
        if(cmax<ctemp)
      {
            cmax = ctemp;
        }
  }
  else if(flag==1 && chr=='x')
  {
        cmax=1;
        flag=0;
  }
}

cl=(int *)malloc((cmax+1)*sizeof(int));
clu=(int *)malloc((cmax+1)*sizeof(int));
cll=(int *)malloc((cmax+1)*sizeof(int));
timesused=(int *)malloc((cmax+1)*sizeof(int));

for(i=0;i<=cmax;i++)
{
  cl[i]=0;
  timesused[i]=1;
}

i=0;

flag=1;

while(flag)
{
  if(i<len)
  {
        chr=(char )[tempString characterAtIndex:i];
  }
  else if(i==len && chr=='x' )
  {
        break;
  }
  else
  {
        flag=0;
  }
  if(chr=='x' || i==len || ((chr=='+' || chr=='-') && i>0 && m!=-1))
  {
        if(chr=='x' && m==-1)
        {
              coeff=1;
        }
        else if(chr=='x' && m==0 && tempMatrix[0]=='+')
        {
              coeff=1;
        }
        else if(chr=='x' && m==0 && tempMatrix[0]=='-')
        {
              coeff=-1;
        }
```

```
            else
            {
                  for(y=0;m>=0;m--,y++)
                  {
                        if(tempMatrix[m]!='+' && tempMatrix[m]!='-')
                        {
                              coeff= coeff + (tempMatrix[m] -'0')* pow(10,y);
                        }
                        else if(tempMatrix[m]=='-')
                        {
                              coeff=-coeff;
                        }
                  }
            }

            if(chr=='x' && i<len-1 && (char )[tempString
characterAtIndex:i+1]=='^')
            {
                  powCoef=(char )[tempString characterAtIndex:i+2] -'0';
                  i=i+2;
                  cl[powCoef]=coeff;
                  coeff=0;
            }
            else if(chr=='x')
            {
                  cl[1]=coeff;
                  coeff=0;
            }
            else
            {
                  cl[0]=coeff;
                  coeff=0;
                  i--;
            }
            m=-1;
      }
      else
      {
            m++;
            tempMatrix[m]=chr;
      }

      i++;
   }

   flag=1;

   if(cl[cmax]<0)
   {
      for(i=0; i<=cmax; i++)
      {
            clu[i]=-cl[i];
      }
   }
   else
```

```
    {
      for(i=0; i<=cmax; i++)
      {
            clu[i]=cl[i];
      }
    }

    if(cl[0]<0)
    {
      for(i=0; i<=cmax; i++)
      {
            cll[i]=-cl[i];
      }
    }
    else
    {
      for(i=0; i<=cmax; i++)
      {
            cll[i]=cl[i];
      }
    }

    len = cmax+1;

}
```

Κάνουμε #import “LMQBrain.h” για να κληρονομήσουμε το header file της εφαρμογής μας και στη συνέχεια υλοποιούμε ό,τι συναρτήσεις δηλώσαμε στο header file «LMQBrain.h» Για παράδειγμα, υλοποιούμε την συνάρτηση getCoeffs η οποία όπως προαναφέρθηκε επιστρέφει τους συντελεστές του πολυωνύμου.Στην συνάρτηση αυτή σκανάρουμε το πολυώνυμο το οποίο το εισάγουμε με την μορφή String για να βρούμε τα «^» τα οποία υποδεικνύουν την ύπαρξη δύναμης του χ και συνεπώς συντελεστή.Με το που βρούμε το «^» βρίσκουμε την θέση της δύναμης του χ(y+1) και την αποθηκεύουμε σε μια μεταβλητή (chrr) την οποία και μετατρέπουμε σε αριθμό ctemp=chrr-‘0’.Έτσι βρίσκουμε και την μεγαλύτερη δύναμη του πολυωνύμου η οποία στην ουσία αποτελεί και το μέγεθος των πινάκων μας(cmax==len).Δηλώνουμε στην συνέχεια τους πίνακες των συντελεστών μας που πλέον γνωρίζουμε το μέγεθός τους(cl[cmax],xll[cmax],clu[cmax]) και αρχικοποιούμε τον πίνακα timesused με 1.Στην συνέχεια πάμε να τους αρχικοποιήσουμε με το να βρούμε τους συντελεστές του πολυωνύμου.Σαρώνουμε πάλι το πολυώνυμο μέχρι να βρούμε αυτή τη φορά το «χ» και πριν φτάσουμε στο τέλος του πολυωνύμου μας.Μόλις βρούμε το «χ» μετράμε μία θέση πριν για τον συντελεστή του και δύο θέσεις πρίν το «χ» για να βρούμε το πρόσημο του συντελεστή.Μόλις τους βρούμε τους αποθηκεύουμε στον πίνακα cl.Αν ο συντελεστής

της τελευταίας δύναμης είναι αρνητικός αλλάζουμε τα πρόσημα όλων των συντελεστών και τους αποθηκεύμε στον πίνακα clu και αν ο συντελεστής της πρώτης δύναμης είναι αρνητικός αλλάζουμε όλα τα πρόσημα των συντελεστών και τους αποθηκεύουμε στον πίνακα cll .Τέλος υλοποιούμε τις getUB και getLB οι οποίες και υλοποιούν τους αλγορίθμους μας για το τοπικό μέγιστο και ελάχιστο αντίστοιχα.

```
- (double)getUB {
    if(len<=1)
    {
      ub = 0;
    }
    else
    {
      for(m=len-1; m>=1; m--)
      {
            if(clu[m-1]<0)
            {
                  tempmin = 999999999*1.0;

                  for(n=len; n>=m+1; n--)
                  {
                        if(clu[n-1]>0)
                        {
                              temp1=(timesused[n-1]++);
                              temp2=pow(2,temp1);
                              temp3=-clu[m-1]*1.0/clu[n-1]*1.0;
                              temp4=(temp2*1.0)*temp3;
                              temp3=1.0/(n-m);
                              q=powf(temp4,temp3);
                              if(tempmin>q)
                              {
                                    tempmin=q;
                              }
                        }
                  }
                  if(tempmax<tempmin)
                  {
                        tempmax=tempmin;
                  }
            }
      }
      ub=ceil((65*1.0/64*1.0)*tempmax);
    }
    return ub;
}
- (double)getLB {
    temp1=0;
    temp2=0;
    temp3=0.0;
    temp4=0.0;
    q=-1.0;
    tempmax=0.0;
```

```
    tempmin=0.0;
    len=cmax+1;

    for(i=0; i<=cmax; i++)
    {
      timesused[i]=1;
    }

    if(len<=1)
    {
      lb = 0;
    }
    else
    {
      for(m=2; m<=len; m++)
      {
            if(cll[m-1]<0)
            {
                  tempmin = 999999999*1.0;

                  for(n=1; n<=m-1; n++)
                  {
                        if(cll[n-1]>0)
                        {

                              temp1=(timesused[n-1]++);
                              temp2=pow(2,temp1);
                              temp3=-cll[m-1]*1.0/cll[n-1]*1.0;
                              temp4=(temp2*1.0)*temp3;
                              temp3=1.0/(m-n);
                              q=powf(temp4,temp3);
                              if(tempmin>q)
                              {
                                    tempmin=q;
                              }
                        }
                  }
                  if(tempmax<tempmin)
                  {
                        tempmax=tempmin;
                  }
            }
      }
      lb=powf(2,-ceil(log2f(tempmax)));
    }
    return lb;
}

@end
```

Στον Τάσο και τον Χριστόφορο...

Στο αρχείο LocalMaxQuadraticViewController υλοποιούμε όλα τα κουμπιά του interface builder μας για την εμφάνιση των τοπικών ακρότατων.Πρώτα , στο LocalMaxQuadraticViewController.h δηλώνουμε τα κουμπιά που θα χρησιμοποιήσουμε και θα υλοποιήσουμε στη συνέχεια.Πιο αναλυτικά στην αρχη του LocalMaxQuadraticViewController.h κάνουμε « #import <UIKit/UIKit.h>» γιατί ο controller μας κληρονομεί από το **UIViewController.To UIKit** υποστηρίζει το Model View Controller κυρίως μέσα απο αυτή την κλάση.Στη συνέχεια κάνουμε **#import "LMQBrain.h"** το header file της υπολογιστικής μας και τα «#import <AVFoundation/AVAudioPlayer.h>» και **«#import <AVFoundation/AVFoundation.h>»** για τον ήχο και την εικόνα αντίστοιχα της εφαρμογής μας.Μέσα στο @interface δηλώνουμε το upper(up) και lower bound(lb) το txtEquation το οποίο είναι της κλάσης UITextField και αποτελεί το πολυώνυμο που δίνουμε ,το brain το οποίο είναι αντικείμενο της κλάσης LMQBrain και δείχνει στο model μας, το display το οποίο «δείχνει» στο **UIView του interface builder** μας και τέλος το player το οποίο αποτελεί τον ήχο της εφαρμογής μας.Στη συνέχεια δηλώνουμε τα κουμπιά της εφαρμογής μας:το computerBounds το οποίο με το που το πατάμε καλεί την μέθοδο για τον υπολογισμό των τοπικών ακρότατων ,το makeyKeboardGoAway το οποίο αν το πατήσουμε εξαφανίζει το πληκτρολόγιο απο την οθόνη, το κουμπί graph το οποίο καλεί την μέθοδο για την εμφάνιση της γραφικής παράστασης του πολυωνύμου(θα το συζητήσουμε στο τρίτο κεφάλαιο) και τέλος το κουμπί help το οποίο εμφανίζει οδηγίες στον χρήστη για το πως θα εισαγάγει το πολυώνυμο στην εφαρμογή.

```
//  LocalMaxQuadraticViewController.h
//  LocalMaxQuadratic
//
//  Created by filioui on 12/23/10.
//  Copyright __MyCompanyName__ 2010. All rights reserved.
//

#import <UIKit/UIKit.h>
#import "LMQBrain.h"
#import <AVFoundation/AVAudioPlayer.h>
#import <AVFoundation/AVFoundation.h>
```

```
@interface LocalMaxQuadraticViewController : UIViewController
<AVAudioPlayerDelegate> {
    IBOutlet UILabel *display;
    IBOutlet UITextField *txtEquation;
    LMQBrain *brain;
    double ub;
    double lb;
    AVAudioPlayer *player;
}

@property (nonatomic,retain) IBOutlet UITextField *txtEquation;
@property (nonatomic,retain) IBOutlet  UILabel *display;
@property (nonatomic,retain) AVAudioPlayer *player;

- (IBAction) computeBounds;
- (IBAction) makeKeyboardGoAway;
- (IBAction) graph;
- (IBAction) help;

@end
```

Στο LocalMaxQuadraticVieController.m υλοποιούμε όσα δηλώσαμε(και προαναφέραμε) στο LocalMaxQuadraticVieController.h.Φυσικά κάνουμε και «#import "GraphViewController.h"» μιας και καλούμε τον GraphViewController για την εμφάνιση της γραφικής μας παράστασης :

```
//  LocalMaxQuadraticViewController.m
//  LocalMaxQuadratic
//
//  Created by filioui on 12/23/10.
//  Copyright __MyCompanyName__ 2010. All rights reserved.
//

#import "LocalMaxQuadraticViewController.h"
#import "GraphViewController.h"

@implementation LocalMaxQuadraticViewController

@synthesize txtEquation;
@synthesize display;
@synthesize player;
```

```
/*
- (IBAction) play {
    [self.player play];
}
*/

- (LMQBrain *) brain {
    if (!brain) {
      brain = [[LMQBrain alloc] init];
    }
    return brain;
}

- (IBAction) help {
    NSString *helpString = @"Insert an equation in the textfield and press
either BoundIt to get Upper and Lower Bounds, or press Graph to plot its
graph.\nEquation example: 3x^2+2x^3+x-7\nNo need to use the * sign for
multiplication, no need to enter the powers of x in a specific set.";

    UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"Help"

    message:helpString

delegate:nil

cancelButtonTitle:@"Done"

otherButtonTitles:nil];
    [alert show];
    [alert release];
}
```

```
- (IBAction) graph {
    [[self brain] getCoeffs:[txtEquation text]];
    GraphViewController *gvc = [[GraphViewController alloc] init];
    gvc.expression=brain.cl;
    gvc.size=brain.cmax+1;
    //gvc.scale = (float)self.view.contentScaleFactor;
    gvc.scale=50;
    gvc.title=[txtEquation text];
    [self.navigationController pushViewController:gvc animated:YES];
    [gvc release];
}
```

```
- (IBAction) computeBounds {
    [txtEquation resignFirstResponder];

    [self.player play];
    [[self brain] getCoeffs:[txtEquation text]];
    ub = [[self brain] getUB];
    lb = [[self brain] getLB];

    NSString *msg=[[NSString alloc] initWithFormat:@"Upper bound: %f\nLower
bound: %f",ub,lb];

    display.lineBreakMode = UILineBreakModeWordWrap;
    display.numberOfLines = 0;
    display.textAlignment = UITextAlignmentRight;

    [display setText:msg];
    [msg release];
}
```

```
- (IBAction) makeKeyboardGoAway
{
    [txtEquation resignFirstResponder];
}

- (BOOL)textFieldShouldReturn:(UITextField *)textField
{
    [textField  resignFirstResponder];
    return YES;
}

-(BOOL)shouldAutorotateToInterfaceOrientation:
(UIInterfaceOrientation)interfaceOrientation
{
    return YES;
}

/*
// The designated initializer. Override to perform setup that is required
before the view is loaded.
- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle
*)nibBundleOrNil {
    if (self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil]) {
        // Custom initialization
    }
    return self;
}
```

```
*/

/*
// Implement loadView to create a view hierarchy programmatically, without
using a nib.
- (void)loadView {
}
*/

// Implement viewDidLoad to do additional setup after loading the view,
typically from a nib.









- (void)viewDidLoad {

    NSString *soundPath = [[NSBundle mainBundle] pathForResource:@"sound"
ofType:@"wav"];
    self.player =[[AVAudioPlayer alloc] initWithContentsOfURL:[NSURL
fileURLWithPath:soundPath] error:NULL];
    [player setDelegate: self];
    [player prepareToPlay];

    [super viewDidLoad];
}

/*
// Override to allow orientations other than the default portrait
orientation.
-
(BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)interfac
eOrientation {
    // Return YES for supported orientations
    return (interfaceOrientation == UIInterfaceOrientationPortrait);
}
*/

- (void)didReceiveMemoryWarning {
    // Releases the view if it doesn't have a superview.
    [super didReceiveMemoryWarning];

    // Release any cached data, images, etc that aren't in use.
}

- (void)viewDidUnload {
    // Release any retained subviews of the main view.
    // e.g. self.myOutlet = nil;
```

```
}


- (void)dealloc {
    [super dealloc];
}

@end
```

Πιο αναλυτικά, το κουμπί help καλεί την alert η οποία είναι UIAlertView και το αρχικοποεί με τον τίτλο «help».Με το που πατάμε το κουμπί εμφανίζεται το μήνυμα στην οθόνη μας το οποίο είναι NSString.

Το κουμπί graph καλεί την getCoeffs για να του επιστρέψει τους συντελεστές του πολυωνύμου και το gvc το οποίο είναι αντικείμενο του GraphViewController για την εμφάνιση της γραφικής παράστασης.

Το κουμπί του computerBounds καλεί την resignFirstResponder ώστε αφού γράψουμε το πολυώνυμο να εξαφανιστεί το πληκτρολόγιο, στη συνέχεια καλεί την getCoeffs για να του επιστρέψει τους συντελεστές του πολυωνύμου και τις getUB και getLB για να του επιστρέψουν τα upper και lower bounds αντίστοιχα.Το msg της κλάσης NSString είναι εκείνο που χρησιμοποιούμε για να εμφανιστούν τα τα upper και lower bounds στην οθόνη(display setText:msg).

Η εντολή «-(BOOL)shouldAutorotateToInterfaceOrientation: (UIInterfaceOrientation)interfaceOrientation{return YES;}» χρησιμεύει στο όταν περιστρέφεται το κινητό να περιστρέφεται και ο interface builder.
Στο viewDidLoad καλούμε την AVAudioPlayer για να παίξει τον ήχο καθώς υπολογίζονται τα τοπικά ακρότατα απο την computerBounds.

Στο αρχείο LocalMaxQuadraticAppDelegate.h έχουμε μια τοπική μεταβλητή για το UIWindow στο LocalMaxQuadraticViewController.xib το οποίο ονομάζεται window .Έχει μια τοπική μεταβλητή για LocalMaxQuadraticViewController που ονομάζεται viewController.Αυτή είναι η μέθοδος όπου η view του LocalMaxQuadraticViewController, προστίθεται στο UIWindow αλλά και το

UIWindow γίνεται ορατό [window makeKeyAndVisible] αλλά και window addSubview:navcon.view:

```
//  LocalMaxQuadraticAppDelegate.h
//  LocalMaxQuadratic
//
//  Created by Peter on 2/13/11.
//  Copyright __MyCompanyName__ 2011. All rights reserved.
//

#import <UIKit/UIKit.h>

@interface LocalMaxQuadraticAppDelegate : NSObject <UIApplicationDelegate> {
    UIWindow *window;
}

@property (nonatomic, retain) IBOutlet UIWindow *window;

@end
```

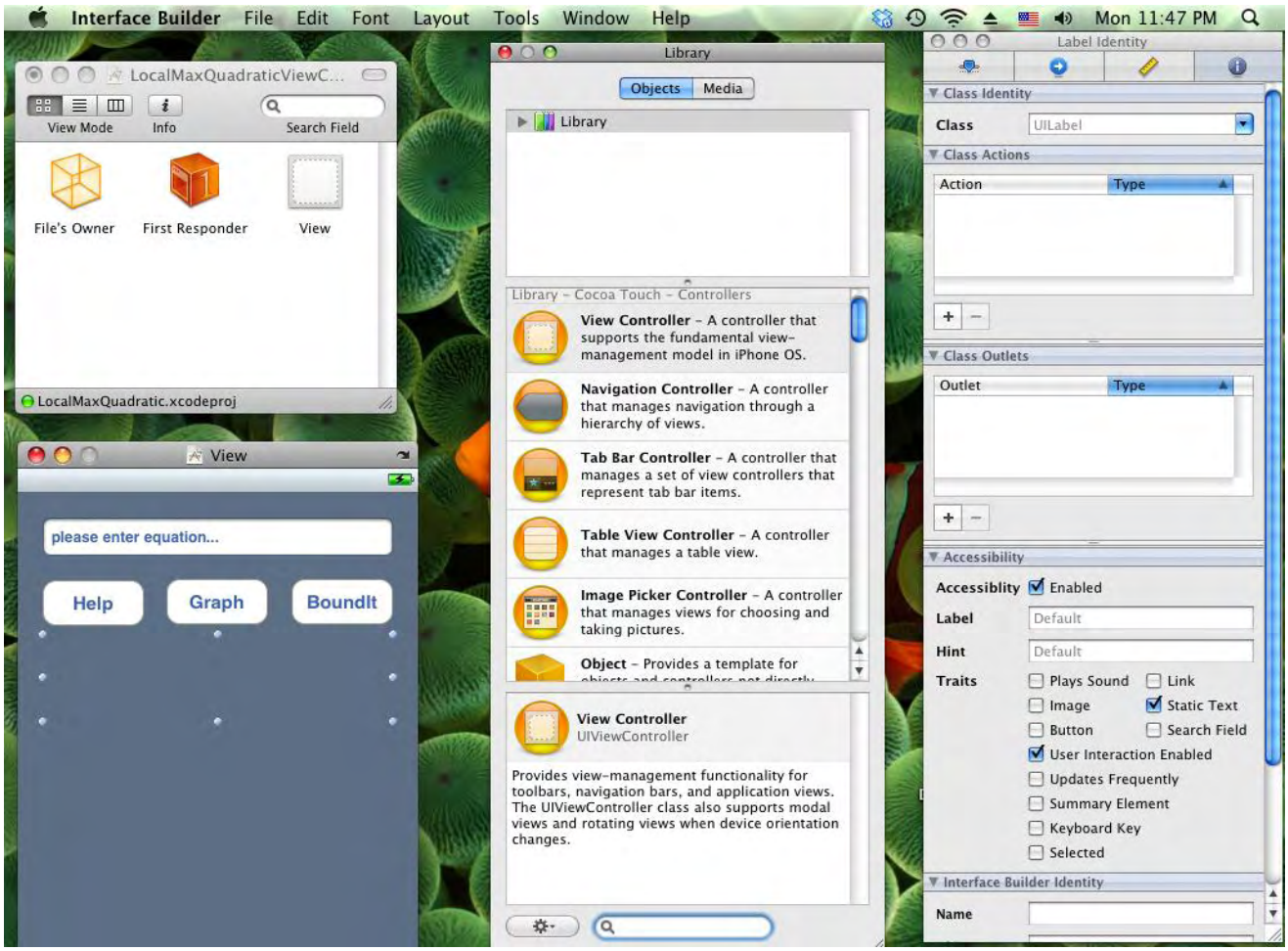
Στο οποίο δηλώνουμε το window της κλάσης UIWindow και στο αρχείο LocalMaxQuadraticAppDelegate.m το υλοποιούμε προσθέτοντας το navcon του UINavigationController στο addSubview αλλά και εδώ προσθέτουμε τον τίτλο της εφαρμογής μας «LocalMaxQuadratic» στο αντικείμενο lvc του LocalMaxQuadraticViewController:

```
//  LocalMaxQuadraticAppDelegate.m
//  LocalMaxQuadratic
//
//  Created by Peter on 2/13/11.
//  Copyright __MyCompanyName__ 2011. All rights reserved.
//

#import "LocalMaxQuadraticAppDelegate.h"
#import "LocalMaxQuadraticViewController.h"

@implementation LocalMaxQuadraticAppDelegate
```

```
@synthesize window;


- (void)applicationDidFinishLaunching:(UIApplication *)application {

    // Override point for customization after application launch
    [window makeKeyAndVisible];
}

- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
    // Override point for customization after application launch.
    UINavigationController *navcon = [[UINavigationController alloc] init];
    LocalMaxQuadraticViewController *lvc = [[LocalMaxQuadraticViewController
alloc] init];
    lvc.title=@"LocalMaxQuadratic";
    [navcon pushViewController:lvc animated:NO];
    [lvc release];
    [window addSubview:navcon.view];
    [window makeKeyAndVisible];

    return YES;
}

- (void)dealloc {
    [window release];
    [super dealloc];
}
@end
```

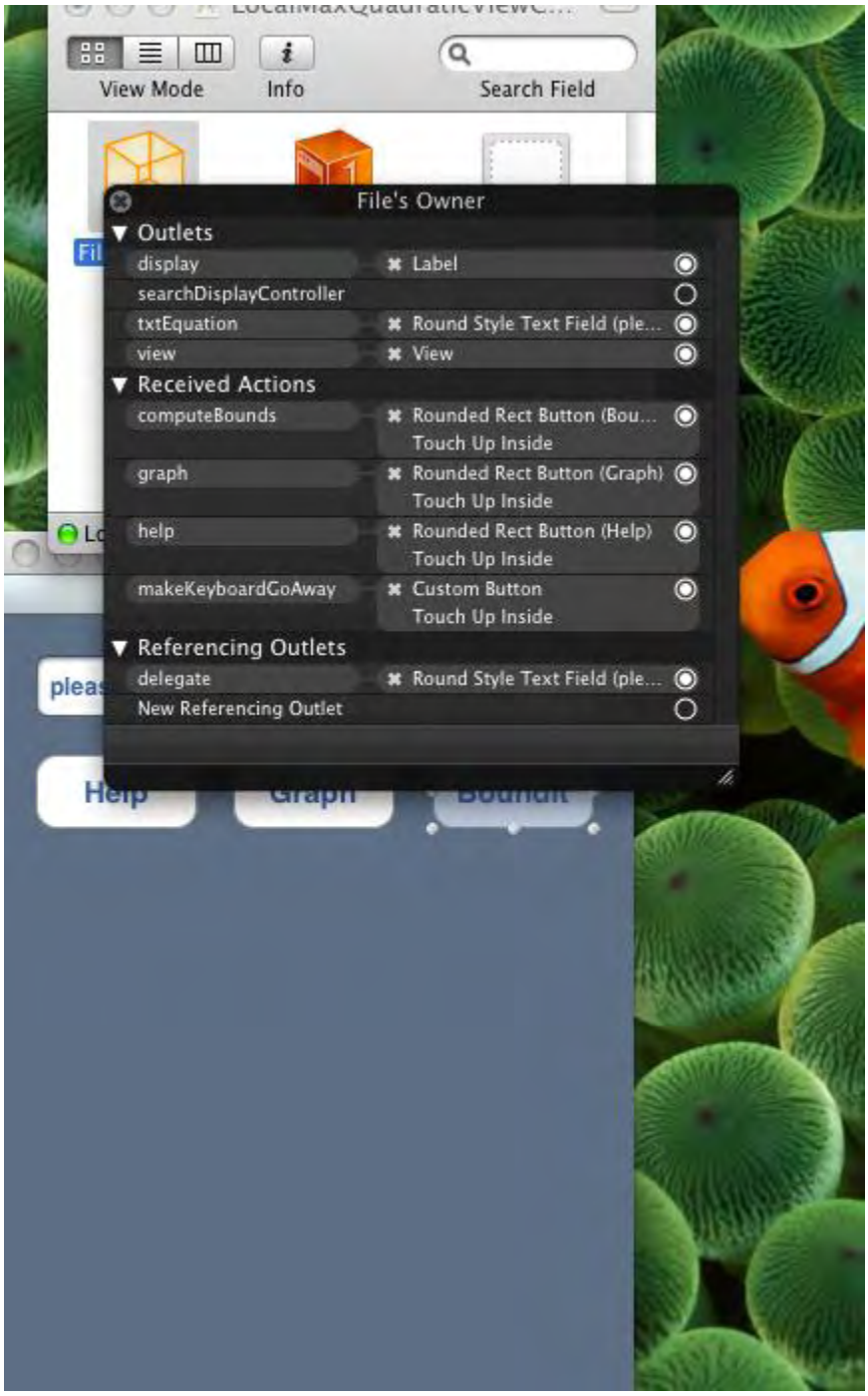
Τέλος, στο LocalMaxQuadraticViewController.xib :

Στον Τάσο και τον Χριστόφορο...



Τα κουμπιά είναι της μορφής «Round Rect Button» τα οποία βρίσκονται στο Library του interface builder.To File's Owner είναι ο Controller μας :

Στον Τάσο και τον Χριστόφορο...



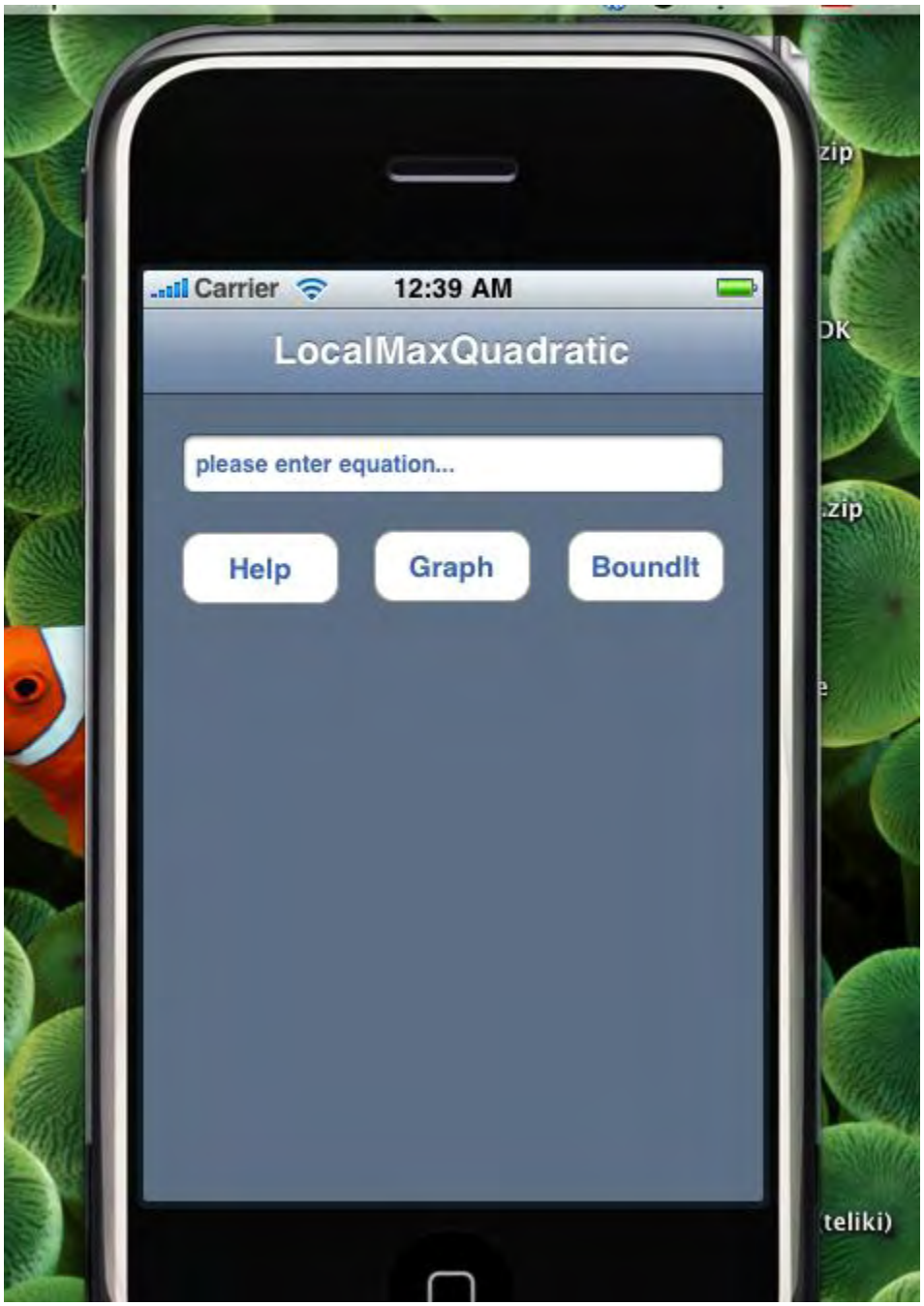
Στον Τάσο και τον Χριστόφορο...

Όπως βλέπετε και απο την εικόνα , συνδέουμε τα events (IBAction) με τα κουμπιά και τις μεθόδους που τα εκτελούν και το display με την όψη του txtEquation.

2.3 Αποτελέσματα

Πάμε λοιπόν να δούμε πως φαίνεται η εφαρμογή μας.Όπως βλέπετε έχουμε το txtEquation όπου θα «φιλοξενηθεί» το πολυώνυμό μας,τα κουμπιά Help, Graph και BoundIt και φυσικά εμφανίζεται και ο τίτλος της εφαρμογής μας.

Στον Τάσο και τον Χριστόφορο...



Στον Τάσο και τον Χριστόφορο...

Πατάμε μέσα στο txtEquation και εμφανίζεται το πληκτρολόγιο.Αν πατήσουμε έξω απο το txtEquation ή στο (μεγάλο)κουμπί ή στο return το πληκτρολόγιο θα εξαφανιστεί.

Στον Τάσο και τον Χριστόφορο...

Εισάγουμε το πολυώνυμό μας 2χ^2 και πατάμε το BoundIt.Ακούγεται ο ήχος της εφαρμογής μας και σε λίγα δευτερόλεπτα εμφανίζονται στην οθόνη τα αποτελέσματα,

Upper bound:0,00000

Lower bound:inf(άπειρο)

3 Γραφική Παράσταση

3.1 Περιγραφή του κώδικα

Όπως είδαμε προηγουμένως στον interface builder υπάρχει ένα κουμπί με το όνομα «Graph» το οποίο και αν πατήσουμε θα εμφανιστεί η γραφική παράσταση του πολυωνύμου που δώσαμε σε ένα διαφορετικό UIView window. Για να το κάνουμε αυτό δημιουργούμε 9 αρχεία,το GraphView.h, GraphView.m, AxesDrawer.h, AxesDrawer.m, GraphAppDelegate.h, GraphAppDelegate.m, GraphViewController.m, GraphViewController.m και το GraphViewController.xib.
Στο header file GraphView.h έχουμε τον εξής κώδικα :

```
//  GraphView.h
//  Graph
//
//  Created by Peter and filioui on 2/12/11.
//  Copyright 2011 __MyCompanyName__. All rights reserved.
//

#import <UIKit/UIKit.h>

@class GraphView;

@protocol GraphViewDelegate
- (float)getY:(GraphView *)requestor forX:(float)anX;
- (float)getScale:(GraphView *)requestor;
@end

@interface GraphView : UIView {
    id <GraphViewDelegate> delegate;
}

@property (assign) id <GraphViewDelegate> delegate;

@end
```

Στην αρχή λοιπόν έχουμε «@protocol GraphViewDelegate» γιατί θέλουμε να κληρονομήσουμε απο το header file του «GraphViewDelegate».Όπως βλέπετε επιστρέφουμε τα αποτελέσματα των δύο μεθόδων «getY» και «getScale».Στην συνέχεια γράφουμε « @interface GraphView : UIView» γιατί το αρχείο μας είναι UIView κλάση και φυσικά γράφουμε «id <GraphViewDelegate> delegate» στο @interface γιατί θέλουμε να ανακηρύσουμε την κλάση την οποία κληρονομούμε.

Τέλος, το delegate πάντα ορίζεται ως « assign @property», για αυτό για παράδειγμα γράφουμε «@property (assign) id <GraphViewDelegate> delegate».Αυτό που πρέπει να σημειώσουμε είναι ότι υποτίθεται πως πάντα το αντικείμενο που υπηρετεί ως delegate θα «ζει» περισσότερο από το αντικείμενο που κάνει το delegate.Συνήθως είναι αλήθεια, διότι το ένα είναι αντικείμενο View (π.χ. UIScrollView) και το άλλο είναι ένα αντικείμενο Controller. Οι Controllers συνήθως δημιουργούν και διαγράφουν τα View αντικείμενά τους.

Στη συνέχεια λοιπόν πρέπει να υλοποιήσουμε όσα δηλώσαμε στο header file «GraphView.h» ,έτσι στο «GraphView.m» θα έχουμε τον εξής κώδικα :

```
//
//  GraphView.m
//  Graph
//
//  Created by Peter and filioui on 2/12/11.
//  Copyright 2011 __MyCompanyName__. All rights reserved.
//

#import "GraphView.h"
#import "AxesDrawer.h"

@implementation GraphView

@synthesize delegate;

- (id)initWithFrame:(CGRect)frame {
    if (self = [super initWithFrame:frame]) {
        // Initialization code
    }
    return self;
}


- (void)drawRect:(CGRect)rect {

    CGPoint midPoint;
    midPoint.x = self.bounds.origin.x + self.bounds.size.width/2;
    midPoint.y = self.bounds.origin.y + self.bounds.size.height/2;

    float scale = [self.delegate getScale:self];

    CGContextRef context = UIGraphicsGetCurrentContext();
    [AxesDrawer drawAxesInRect:self.bounds originAtPoint:midPoint
scale:scale];

    CGContextBeginPath(context);

    CGFloat x = -midPoint.x;
```

```
    CGFloat y = (-[self.delegate getY:self forX:(x/scale)])*scale;


    CGContextMoveToPoint(context, x+midPoint.x, y+midPoint.y);

    [[UIColor colorWithRed:0.2431 green:0.3843 blue:0.6549 alpha:1]
setStroke];
    //[[UIColor colorWithRed:0.3686 green:0.4313 blue:0.5137 alpha:1]
setStroke];

    CGContextSetLineWidth(context, 4.0);

    for(; x<self.bounds.size.width; x++)
    {
      y = (-[self.delegate getY:self forX:(x/scale)])*scale;
      CGContextAddLineToPoint(context, x+midPoint.x, y+midPoint.y);
    }

    CGContextStrokePath(context);
}

- (void)dealloc {
    [super dealloc];
}


@end
```

Όπου το πρώτο πράγμα που πρέπει να παρατηρήσουμε είναι τα δύο header files που εισάγουμε ‘ **«#import "GraphView.h"» και « #import "AxesDrawer.h"»** όπου το πρώτο header file είναι εκείνο της κλάσης μας (**GraphView.m**) και με το δεύτερο εισάγουμε τους δύο άξονες πάνω στους οποίους ζωγραφίζεται η γραφική μας παράσταση.
Ένα View (δηλαδή μια υποκλάση της UIView) όπως αποτελεί και η «**GraphView.m**» κλάση μας αντιπροσωπεύει μια ορθογώνια περιοχή η οποία ορίζει ένα συντεταγμένο χώρο στον οποίο ζωγραφίζει και χειρίζεται γεγονότα.
Ιεραρχικά μιλώντας,μόνο μια υπερκλάση - (UIView *) superview μπορεί να έχει μία (ή καμία) υποκλάση - (NSArray *) subviews. Η ιεραρχία πιο συχνά κατασκευάζεται στον Interface Builder.Έτσι λοιπόν,πίσω στον κώδικάς μας, έχουμε τις εξής μεθόδους :Αρχικά είναι η « initWithFrame» κληρονομεί απο τον ίδιο της τον εαυτό(στην ουσία δεν κάνει τίποτα για αυτό και δεν επεκτεινόμαστε).Στην συνέχεια υπάρχει η «drawRect» μια μέθοδος της UIView την **<u>οποία προσοχή δεν μπορούμε να καλέσουμε απ’ευθείας</u>**.Η drawRect είναι μέθοδος της κλάσης CGRect την οποία χρησιμοποιούμε όταν θέλουμε να ζωγραφίσουμε πάνω στην οθόνη.Στην αρχή της drawRect αρχικοποιούμε δύο σημεία, τα οποία είναι αντικείμενα της κλάσης CGPoint και τα οποία αποτελούν το μέσο πλάτος και μήκος της οθόνης μας.Έχοντάς τα σαν

αρχικά σημεία, θα ξεκινήσουμε να ζωγραφίζουμε απο αυτά τα σημεία την γραφική μας παράσταση αλλά και τους άξονες.Η μεταβλητή scale χρησιμοποιείται για να αρχικοποιήσουμε τα pixels των αξόνων μας και συνεπώς της γραφικής μας παράστασης.Στη συνέχεια «χρωματίζουμε» την γραφική μας παράσταση(κάτι το οποίο δεν είναι απαραίτητο απλά το κάναμε για να είναι πιο ευδιάκριτη σε σχέση με του άξονες).Κάτι που είναι σημαντικό να παρατηρήσουμε είναι ο τρόπος που δημιουργείται η εκάστοτε γραφική παράσταση του πολυωνύμου μας.Αυτό που συμβαίνει στην ουσία είναι ότι δημιουργούμε δύο σημεία και μέσω της CGContextAddLineToPoint ,την οποία και χρησιμοποιούμε σε επανάληψη,τα ενώνουμε με μία γραμμή.Τέλος να προσθέσουμε ότι με την εντολή CGContextSetLineWith ορίζουμε το πλάτος που θα «φτάσει» η γραφική μας παράσταση.Τόση ώρα αναφέρομαι σε σημεία x και y τα οποία παίρνουν τιμές απο την μέθοδο getY η οποία αρχικοποείται σε άλλο αρχείο για αυτό και δεν θα επεκταθούμε ακόμα σε αυτό το σημείο.Ενδεικτικά παρουσιάζουμε και τους κώδικες των αξόνων με τους οποίους μπορεί ο καθένας να πειραματιστεί τόσο στο scale όσο και στα χρώματα των αξόνων.
Στο header file Axes.h έχουμε :

```
//
//  AxesDrawer.h
//
//  Created for Stanford University CS193p Fall 2010.
//

#import <Foundation/Foundation.h>

// AxesDrawer is a simple axis-drawing helper class.

@interface AxesDrawer : NSObject

// Draws an x axis and y axis in the specified bounds in the current context,
//   with its origin at the specified axisOrigin and scaled to the passed
pointsPerUnit,
//
// The scale is the number of points (not pixels) per unit on the axis.
// For example, if the size is 280 wide and the scale is 14, the x axis would
go from -10 to +10.
// If the size is 280 wide and the scale is 2, the x axis would go from -70
to +70.
// The graph is also drawn so that it's origin is axisOrigin.
//
// Only marks whole numbers on an axis.
//
// Does not set any graphics state, so set the colors, linewidths, etc., you
want before calling.

+ (void)drawAxesInRect:(CGRect)bounds originAtPoint:(CGPoint)axisOrigin
scale:(CGFloat)pointsPerUnit;

@end
```

Και στο Axes.m έχουμε :

```
//
//  AxesDrawer.m
//
//  Created for Stanford University CS193p Fall 2010.
//

#import "AxesDrawer.h"

@implementation AxesDrawer

#define ANCHOR_CENTER 0
#define ANCHOR_TOP 1
#define ANCHOR_LEFT 2
#define ANCHOR_BOTTOM 3
#define ANCHOR_RIGHT 4

#define HASH_MARK_FONT_SIZE 12.0

#define HORIZONTAL_TEXT_MARGIN 6
#define VERTICAL_TEXT_MARGIN 3

+ (void)drawString:(NSString *)text atPoint:(CGPoint)location
withAnchor:(int)anchor
{
    if ([text length])
    {
      UIFont *font = [UIFont systemFontOfSize:HASH_MARK_FONT_SIZE];

      CGRect textRect;
      textRect.size = [text sizeWithFont:font];
      textRect.origin.x = location.x - textRect.size.width / 2;
      textRect.origin.y = location.y - textRect.size.height / 2;

      switch (anchor) {
            case ANCHOR_TOP: textRect.origin.y += textRect.size.height / 2 +
VERTICAL_TEXT_MARGIN; break;
            case ANCHOR_LEFT: textRect.origin.x += textRect.size.width / 2+
HORIZONTAL_TEXT_MARGIN; break;
            case ANCHOR_BOTTOM: textRect.origin.y -= textRect.size.height / 2
+ VERTICAL_TEXT_MARGIN; break;
            case ANCHOR_RIGHT: textRect.origin.x -= textRect.size.width / 2+
HORIZONTAL_TEXT_MARGIN; break;
      }

      [text drawInRect:textRect withFont:font];
    }
}

#define HASH_MARK_SIZE 3
#define MIN_PIXELS_PER_HASHMARK 25
```

```
+ (void)drawHashMarksInRect:(CGRect)bounds originAtPoint:(CGPoint)axisOrigin
scale:(CGFloat)pointsPerUnit
{
    if (!pointsPerUnit) return;

    if (((axisOrigin.x < bounds.origin.x) || (axisOrigin.x >
bounds.origin.x+bounds.size.width)) &&
      ((axisOrigin.y < bounds.origin.y) || (axisOrigin.y >
bounds.origin.y+bounds.size.height))) {
      return;
    }

    int unitsPerHashmark = MIN_PIXELS_PER_HASHMARK * 2 / pointsPerUnit;
    if (!unitsPerHashmark) unitsPerHashmark = 1;
    CGFloat pixelsPerHashmark = pointsPerUnit * unitsPerHashmark;

    BOOL boundsContainsOrigin = CGRectContainsPoint(bounds, axisOrigin);
    if (boundsContainsOrigin) {
      if ((axisOrigin.x - pixelsPerHashmark < bounds.origin.x) &&
            (axisOrigin.x + pixelsPerHashmark > bounds.origin.x +
bounds.size.width) &&
            (axisOrigin.y - pixelsPerHashmark < bounds.origin.y) &&
            (axisOrigin.y + pixelsPerHashmark > bounds.origin.y +
bounds.size.height)) {
            return;
      }
    } else {
      if ((axisOrigin.y >= bounds.origin.y) &&
            (axisOrigin.y <= bounds.origin.y+bounds.size.height) &&
            (bounds.size.width <= pixelsPerHashmark)) {
            return;
      }
      if ((axisOrigin.x >= bounds.origin.x) &&
            (axisOrigin.x <= bounds.origin.x+bounds.size.width) &&
            (bounds.size.height <= pixelsPerHashmark)) {
            return;
      }
    }

    CGContextRef context = UIGraphicsGetCurrentContext();
    CGContextBeginPath(context);

    int started = NO;
    int stillGoing = YES;

    for (int offset = unitsPerHashmark; !started || stillGoing; offset +=
unitsPerHashmark)
    {
      NSString *positiveLabel = nil;
      NSString *negativeLabel = nil;
      BOOL drew = NO;
      CGFloat scaledOffset = floor(offset * pointsPerUnit);
      CGPoint hashMarkPoint;
      hashMarkPoint.x = axisOrigin.x+scaledOffset;
```

```
	hashMarkPoint.y = axisOrigin.y;
	if (CGRectContainsPoint(bounds, hashMarkPoint)) {
		CGContextMoveToPoint(context, hashMarkPoint.x, hashMarkPoint.y-
HASH_MARK_SIZE);
		CGContextAddLineToPoint(context, hashMarkPoint.x,
hashMarkPoint.y+HASH_MARK_SIZE);
		if (!positiveLabel) positiveLabel = [NSString
stringWithFormat:@"%d", offset];
		[self drawString:positiveLabel atPoint:hashMarkPoint
withAnchor:ANCHOR_TOP];
		drew = YES;
	}
	hashMarkPoint.x = axisOrigin.x-scaledOffset;
	if (CGRectContainsPoint(bounds, hashMarkPoint)) {
		CGContextMoveToPoint(context, hashMarkPoint.x, hashMarkPoint.y-
HASH_MARK_SIZE);
		CGContextAddLineToPoint(context, hashMarkPoint.x,
hashMarkPoint.y+HASH_MARK_SIZE);
		if (boundsContainsOrigin) negativeLabel = positiveLabel;
		if (!negativeLabel) negativeLabel = [NSString
stringWithFormat:@"%d", (boundsContainsOrigin ? offset : -offset)];
		[self drawString:negativeLabel atPoint:hashMarkPoint
withAnchor:ANCHOR_TOP];
		drew = YES;
	}
	hashMarkPoint.x = axisOrigin.x;
	hashMarkPoint.y = axisOrigin.y-scaledOffset;
	if (CGRectContainsPoint(bounds, hashMarkPoint)) {
		CGContextMoveToPoint(context, hashMarkPoint.x-HASH_MARK_SIZE,
hashMarkPoint.y);
		CGContextAddLineToPoint(context, hashMarkPoint.x+HASH_MARK_SIZE,
hashMarkPoint.y);
		if (!positiveLabel) {
			if (boundsContainsOrigin) positiveLabel = negativeLabel;
			if (!positiveLabel) positiveLabel = [NSString
stringWithFormat:@"%d", offset];
		}
		[self drawString:positiveLabel atPoint:hashMarkPoint
withAnchor:ANCHOR_LEFT];
		drew = YES;
	}
	hashMarkPoint.y = axisOrigin.y+scaledOffset;
	if (CGRectContainsPoint(bounds, hashMarkPoint)) {
		CGContextMoveToPoint(context, hashMarkPoint.x-HASH_MARK_SIZE,
hashMarkPoint.y);
		CGContextAddLineToPoint(context, hashMarkPoint.x+HASH_MARK_SIZE,
hashMarkPoint.y);
		if (!negativeLabel) {
			if (boundsContainsOrigin) negativeLabel = positiveLabel;
			if (!negativeLabel) negativeLabel = [NSString
stringWithFormat:@"%d", (boundsContainsOrigin ? offset : -offset)];
		}
		[self drawString:negativeLabel atPoint:hashMarkPoint
withAnchor:ANCHOR_LEFT];
		drew = YES;
```

```
        }
        positiveLabel = nil;
        negativeLabel = nil;
        if (drew) started = YES;
        stillGoing = drew;
    }

    CGContextStrokePath(context);
}

+ (void)drawAxesInRect:(CGRect)bounds originAtPoint:(CGPoint)axisOrigin
scale:(CGFloat)pointsPerUnit
{
    CGContextRef context = UIGraphicsGetCurrentContext();

    UIGraphicsPushContext(context);

    CGContextBeginPath(context);
    CGContextMoveToPoint(context, bounds.origin.x, axisOrigin.y);
    CGContextAddLineToPoint(context, bounds.origin.x+bounds.size.width,
axisOrigin.y);
    CGContextMoveToPoint(context, axisOrigin.x, bounds.origin.y);
    CGContextAddLineToPoint(context, axisOrigin.x,
bounds.origin.y+bounds.size.height);
    CGContextStrokePath(context);

    [self drawHashMarksInRect:bounds originAtPoint:axisOrigin
scale:pointsPerUnit];

    UIGraphicsPopContext();
}

@end
```

Στο header file «GraphAppDelegate.h» κάνουμε ό,τι κάναμε και στο header file «LocalMaxQuadraticAppDelegate.h».Αρχικοποιούμε το παράθυρο που είναι αντικείμενο της κλάσης UIWindow αλλά και το αντικείμενο viewController της κλάσης GraphViewController:

```
//
//  GraphAppDelegate.h
//  Graph
//
//  Created by Peter on 2/12/11.
//  Copyright __MyCompanyName__ 2011. All rights reserved.
//

#import <UIKit/UIKit.h>

@class GraphViewController;

@interface GraphAppDelegate : NSObject <UIApplicationDelegate> {
    UIWindow *window;
```

```
    GraphViewController *viewController;
}

@property (nonatomic, retain) IBOutlet UIWindow *window;
@property (nonatomic, retain) IBOutlet GraphViewController *viewController;

@end
```

Ενώ στο αρχείο «GraphAppDelegate.m» προσθέτουμε το view του viewController στο window μέσω της addSubview και στη συνέχεια το κάνουμε ορατό μέσω της makeKeyVisible.Στο τέλος στη μέθοδο dealloc απελευθερώνουμε τα δύο αντικείμενα.Παρατηρούμε ότι ακόμα και αν το κάνουμε αυτό στην οθόνη παραμένει το παράθυρο με τη γραφική παράσταση.Αυτό συμβαίνει γιατί πλέον τα αντικείμενα έχουν πάει στον ViewController:

```
//
//  GraphAppDelegate.m
//  Graph
//
//  Created by Peter on 2/12/11.
//  Copyright __MyCompanyName__ 2011. All rights reserved.
//

#import "GraphAppDelegate.h"
#import "GraphViewController.h"

@implementation GraphAppDelegate

@synthesize window;
@synthesize viewController;


- (void)applicationDidFinishLaunching:(UIApplication *)application {

    // Override point for customization after app launch
    [window addSubview:viewController.view];
    [window makeKeyAndVisible];
}
```

```
- (void)dealloc {
    [viewController release];
    [window release];
    [super dealloc];
}


@end
```

Στη συνέχεια έχουμε τον GraphViewController.h στον οποίο εισάγουμε το header file του «GraphView.h», αρχικοποιούμε τις μεταβλητές scale,expression,size και δημιουργούμε το αντικείμενο graphView της ομόνυμης κλάσης και δηλώνουμε την μέθοδο zoomPressed η οποία είναι IBAction της κλάσης UIButton και στην ουσία αποτελεί τα κουμπιά zoomIn και zoomOut τα οποία βρίσκονται στο παράθυρο της GraphView(στην γραφική μας παράσταση) ο κώδικας του οποίου είναι:

```
//
//  GraphViewController.h
//  Graph
//
//  Created by Peter on 2/12/11.
//  Copyright __MyCompanyName__ 2011. All rights reserved.
//

#import <UIKit/UIKit.h>
#import "GraphView.h"

@interface GraphViewController : UIViewController <GraphViewDelegate> {
    GraphView *graphView;
    int *expression;
    int size;
    float scale;
}

@property (retain) IBOutlet GraphView *graphView;
@property int *expression;
@property float scale;
@property int size;

- (IBAction)zoomPressed:(UIButton *)sender;
```

```
@end
```

Στη συνέχεια υλοποιούμε όσα δηλώσαμε στο header file «GraphViewController.h» στο αρχείο « GraphViewController.m» αφού πρώτα το κάνουμε και εισαγωγή (#import “GraphViewController.h ”).Στην αρχή λοιπόν του κώδικα υλοποιούμε την eval η οποία υπολογίζει ανάλογα με τις τιμές του x,τις τιμές του y και μας επιστρέφει το y για να το χρησιμοποιήσουμε στη συνέχεια για να ζωγραφίσουμε τη γραφική μας παράσταση.Η μέθοδος αυτή καλεί την exression η οποία επιστρέφει τους συντελεστές του πολυωνύμου ενώ η ίδια καλείται απο την getY .Η updateUI καλείται για να αποθηκευτούν ό,τι αλλαγές έχουν γίνει στο παράθυρο του GraphView, ενώ η getScale επιστρέφει το scale που χρησιμοποιούμε.Στη συνέχεια υλοποιούμε την μέθοδο zoomPressed με την οποία δημιουργούμε τα δύο κουμπιά της GraphView μας και στα οποία δίνουμε τους τίτλους «ZoomIn» και « ZoomOut» και με τα οποία αλλάζουμε το scale τόσο των αξόνων όσο και της γραφικής μας παράστασης με το να το αυξάνουμε και να το μειώνουμε αντίστοιχα.Τέλος έχουμε τις μεθόδους viewDidLoad την οποία και εισάγουμε για να κάνει επιπλέον ρυθμίσεις μετά τη φόρτωση της view συνήθως μετά από μια αλλαγή,την didReceiveMemoryWarning στην οποία απελευθερώνουμε την view αν δεν έχει καμία υπερκλάση αλλά και ό,τι άλλα δεδομένα δεν χρησιμοποιούνται και την viewDidUnload στην οποία απελευθερώνουμε ό,τι υποκλάσεις έχουν απομείνει απο την κεντρική view.Ο κώδικας λοιπόν είναι:

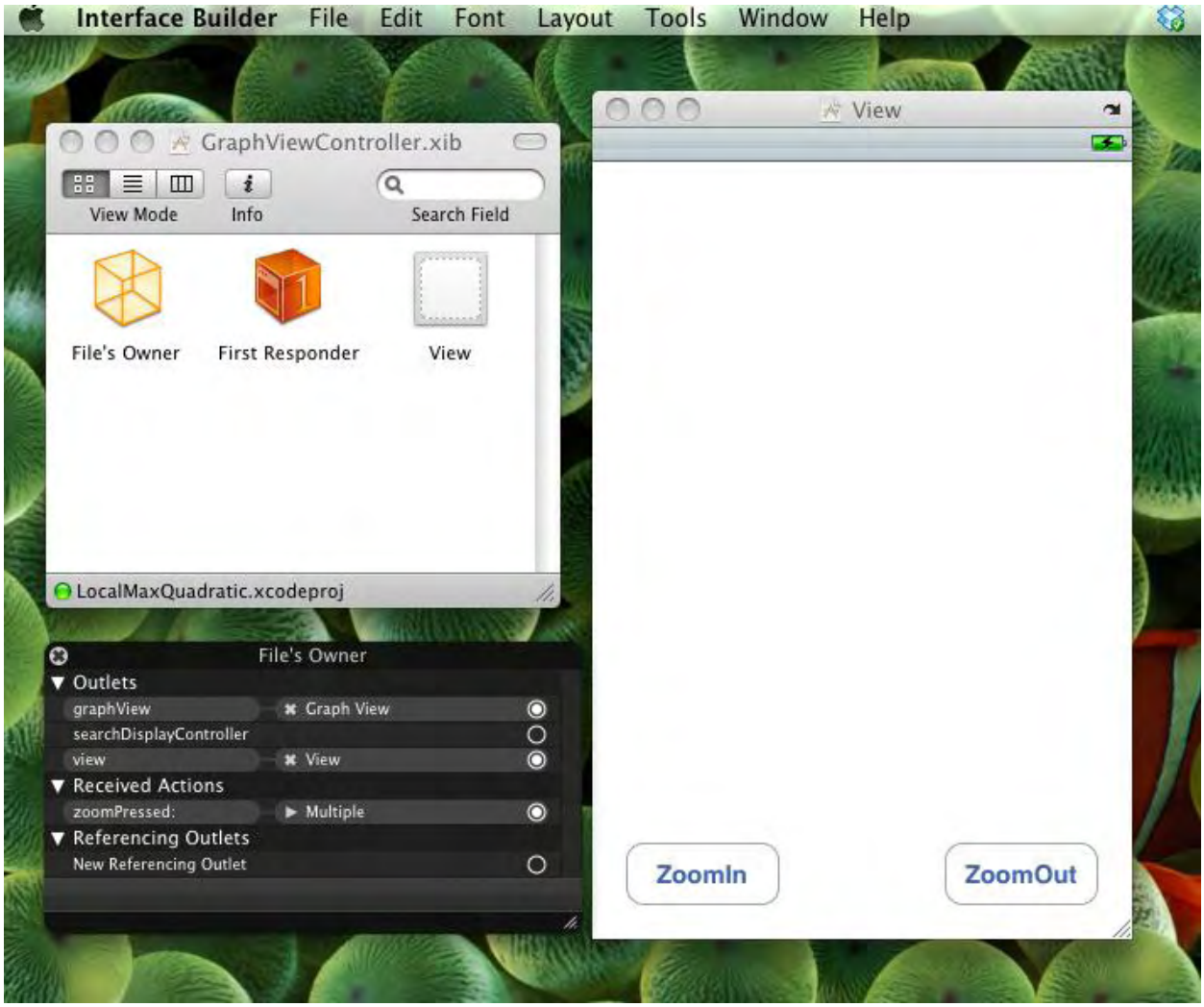
```
//
//  GraphViewController.m
//  Graph
//
//  Created by Peter on 2/12/11.
//  Copyright __MyCompanyName__ 2011. All rights reserved.
```

```
//

#import "GraphViewController.h"

@implementation GraphViewController

@synthesize graphView;
@synthesize expression;
@synthesize scale;
@synthesize size;

- (float) eval:(float)x {
    int i;
    float y=0;

    for(i=0; i<size; i++)
    {
      y = y+self.expression[i]*powf(x,i);
      NSLog(@"y=%f", y);
    }

    return y;
}

- (void) updateUI
{
    [self.graphView setNeedsDisplay];
}

- (float)getY:(GraphView *)requestor forX:(float)anX {
    float y = 0;

    if (requestor == self.graphView) {
      y = [self eval:anX];
    }
    return y;
}
- (float)getScale:(GraphView *)requestor {
    float theScale =0;

    if (requestor == self.graphView) {
      theScale = self.scale;
    }
    return theScale;
}

- (IBAction)zoomPressed:(UIButton *)sender {
    if([[sender.titleLabel text] isEqualToString:@"ZoomIn"])
      self.scale = self.scale+10;
    else if([[sender.titleLabel text] isEqualToString:@"ZoomOut"])
      self.scale = self.scale-10;
    [self updateUI];
}

-(BOOL)shouldAutorotateToInterfaceOrientation:
```

```
(UIInterfaceOrientation)interfaceOrientation
{
    return YES;
}

- (void)viewDidLoad {
    [super viewDidLoad];
    self.graphView.delegate=self;
    [self updateUI];
}

- (void)releaseOutlets
{
    self.graphView=nil;
}


- (void)didReceiveMemoryWarning {
    // Releases the view if it doesn't have a superview.
    [super didReceiveMemoryWarning];

    // Release any cached data, images, etc that aren't in use.
}

- (void)viewDidUnload {
    // Release any retained subviews of the main view.
    [self releaseOutlets];
}


- (void)dealloc {
    [super dealloc];
}

@end
```

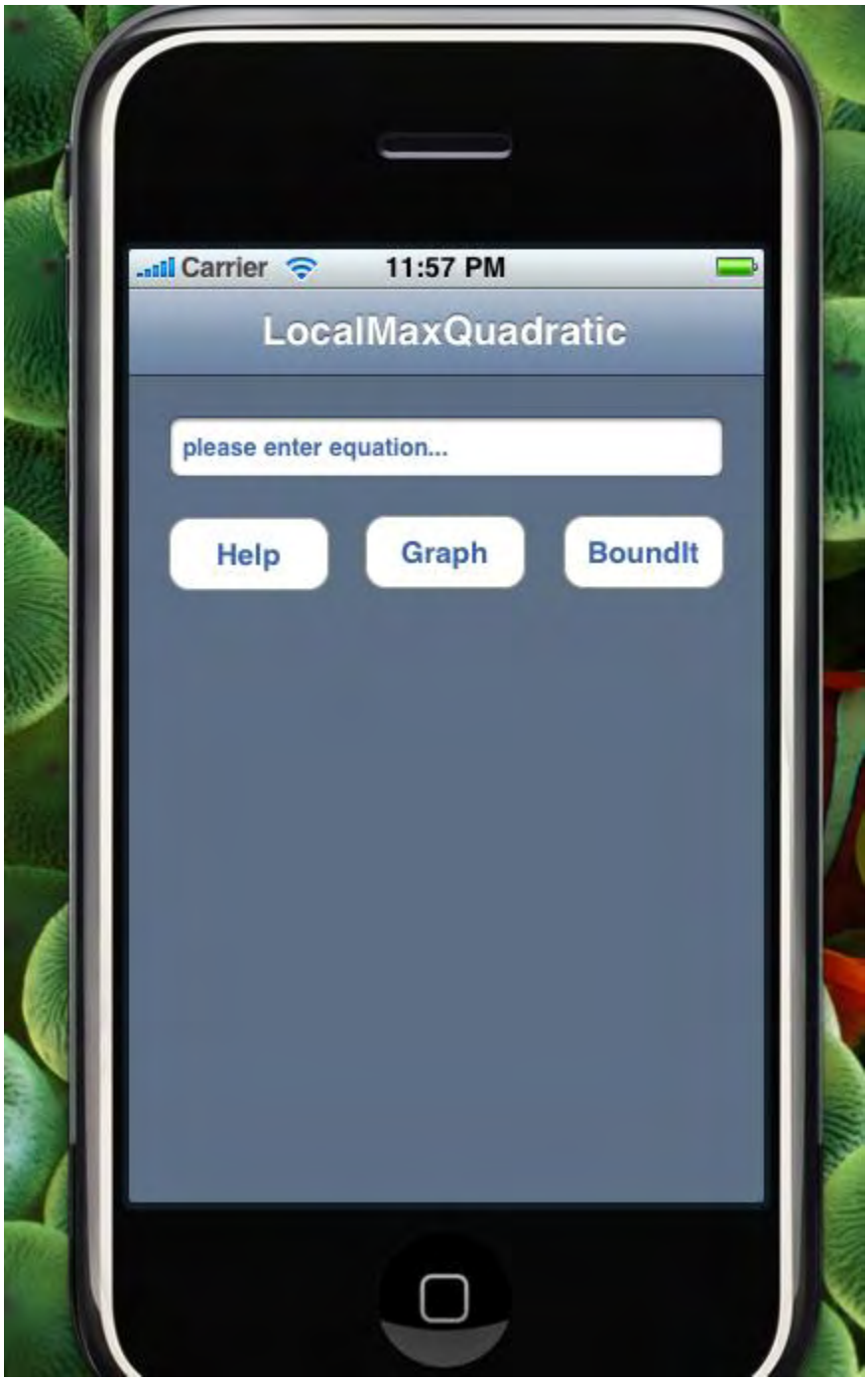
Στο GraphViewController.xib έχουμε:

Στον Τάσο και τον Χριστόφορο...



Πάμε λοιπόν να δούμε όταν εισάγουμε ένα πολυώνυμο και ζητήσουμε την γραφική του παράσταση , τι συμβαίνει.Στην οθόνη μας ζητείται λοιπόν να εισάγουμε το πολυώνυμο

Στον Τάσο και τον Χριστόφορο...



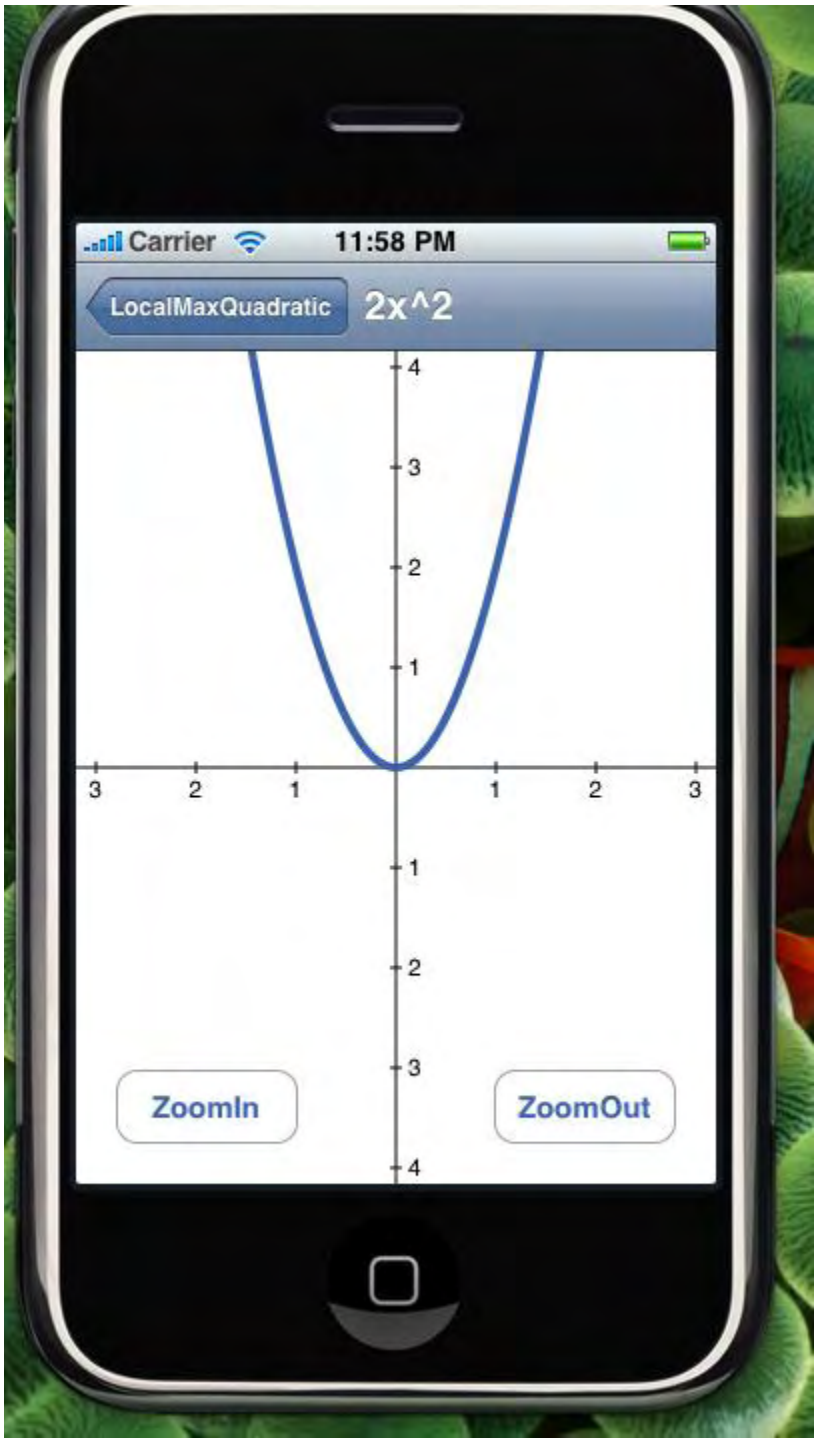
Στον Τάσο και τον Χριστόφορο...

Εισάγουμε όποιο πολυώνυμο θέλουμε,εδώ χρησιμοποιήσαμε το ίδιο πολυώνυμο με πριν για χάριν πληρότητας.Πληκτρολογούμε λοιπόν το πολυώνυμο και πατάμε το κουμπί «Graph»

Στον Τάσο και τον Χριστόφορο...

Σε λίγα δευτερόλεπτα εμφανίζεται μπροστά μας η γραφική παράσταση που πληκτρολογήσαμε.Όντως το 2χ^2 είναι μια παραβολή που περνά απο την αρχή των αξόνων.Όπως βλέπουμε το παράθυρο της GraphView αποτελείται από την γραφική παρασταση , απο τα δύο κουμπιά , απο τον τίτλο του πολυωνύμου που δώσαμε και απο ένα κουμπί με το οποίο πηγαίνουμε στην αρχή της εφαρμογής μας και φέρει τον ομώνυμο τίτλο.

Στον Τάσο και τον Χριστόφορο...

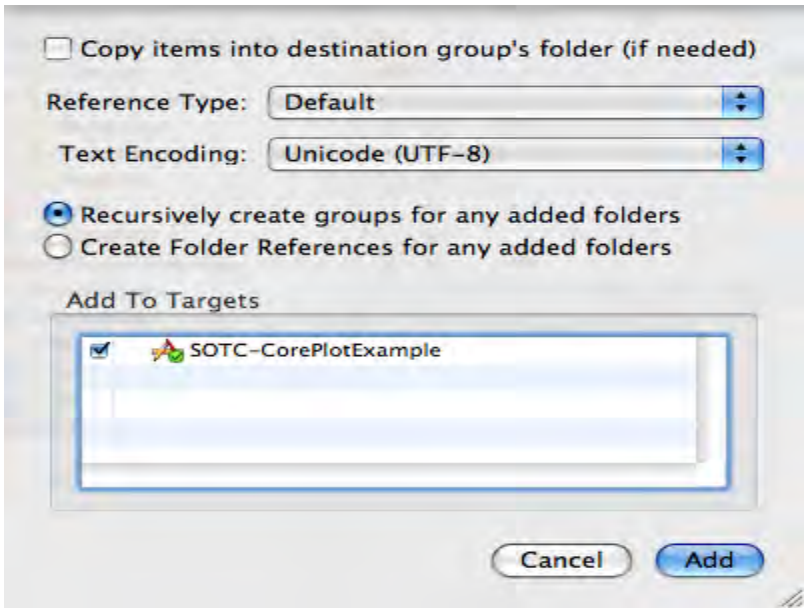


3.2 Core Plot μια εναλλακτική περίπτωση γραφικής παράστασης

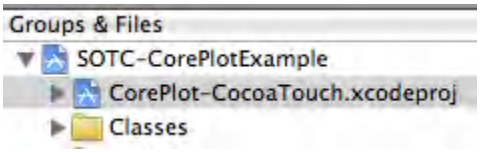
Η ανάγκη για να εμφανιστεί ένα γράφημα κάποιου είδους,είναι εξαιρετικά συχνό φαινόμενο κατά την ανάπτυξη εφαρμογών. Εξαιτίας αυτού, για όλες σχεδόν τις γλώσσες υπάρχουν κάποια γραφικά / βιβλιοθήκες (και μερικές φορές πολύ περισσότερες από μόνο μία).Αν και δεν υπάρχει ενσωματωμένη βιβλιοθήκη για γραφήματα για τις iPhone εφαρμογές, υπάρχει μια βιβλιοθήκη που ονομάζεται ***Core Plot*** . Αυτή η βιβλιοθήκη είναι υπό ανάπτυξη και μπορεί να χρησιμοποιηθεί για να δημιουργηθεί ένας αριθμός διαφορετικών τύπων γραφημάτων.Για να χρησιμοποιηθεί αυτή η βιβλιοθήκη σε εφαρμογή iPhone, επειδή είναι ακόμα σχετικά καινούρια και δεν υπάρχουν διαθέσιμα πακέτα , πρέπει να χρησιμοποιηθεί απευθείας ο κώδικας τον οποίο και αποκτούμε με την ακόλουθη εντολή:

```
hg  clone http://core-plot.googlecode.com/hg/ core-plot
```

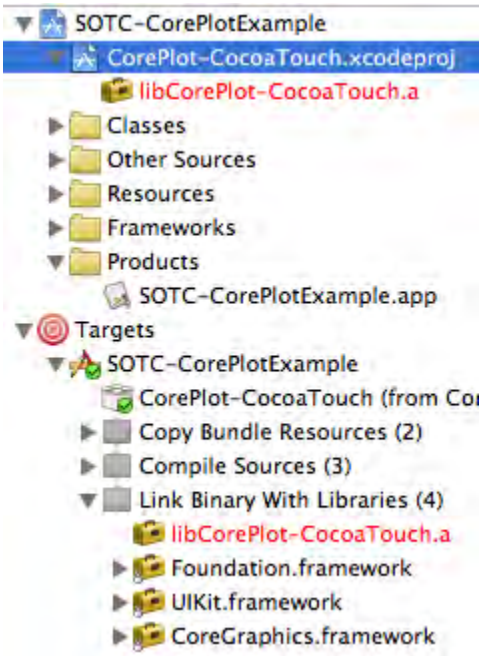
αφού πρώτα από όλα εγκατασταθεί το «Mercurial».Το τελευταίο argument (στην περίπτωσή μας εδώ το core-plot) είναι στην ουσία ο κατάλογος που θα μπει ο κώδικας.
Ας δούμε ένα παράδειγμα για να καταλάβουμε πως δουλεύει η Core Plot.Έστω λοιπόν ότι το όνομα του Xcode Project που θα χρησιμοποιήσουμε είναι "SOTC-CorePlotTest". Το πρώτο πράγμα που πρέπει να κάνουμε αφού δημιουργήσουμε ένα Xcode Project για την εφαρμογή μας, είναι να σύρουμε(drag n drop) το CorePlot-CocoaTouch.xcodeproj αρχείο από τον core-plot/framework κατάλογο στο Project μας.Όταν το κάνουμε αυτό, θα εμφανιστεί στην οθόνη:



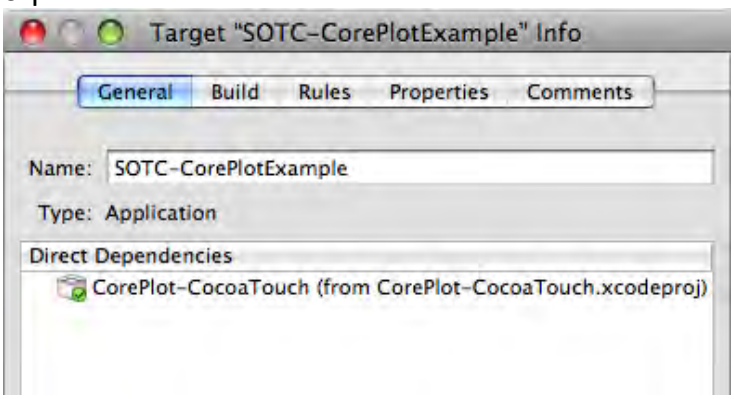
Οι προκαθορισμένες τιμές είναι εντάξει - τις περισσότερες φορές δεν θέλουμε να αντιγράψουμε ολόκληρη τη βιβλιοθήκη στο έργο μας - θέλουμε απλώς μια αναφορά για όπου και αν υπάρχει η core-plot . Αφήνουμε λοιπόν το «Copy items into destination group's folder» ως έχει. Κάνουμε κλικ στο κουμπί "Add" για αυτό το διάλογο και μια καταχώριση για το « CorePlot-CocoaTouch.xcodeproj» θα εμφανιστεί στο «Groups & Files» για το project μας :



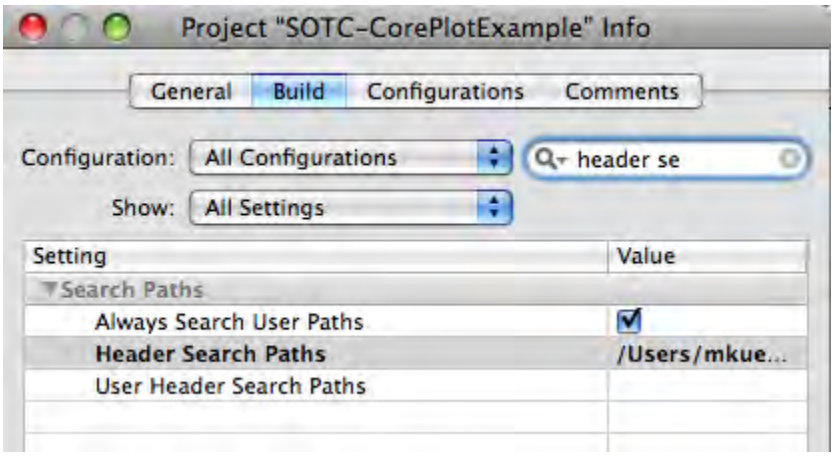
Επειδή η iPhone core-plot είναι μια στατική βιβλιοθήκη, πρέπει να συνδέεται με την εφαρμογή μας. Για να γίνει αυτό, θα πρέπει να πάρουμε την « libCorePlot-CocoaTouch.a» από το «Groups & Files» και να την σύρουμε στο «Link Binaries with Libraries»:



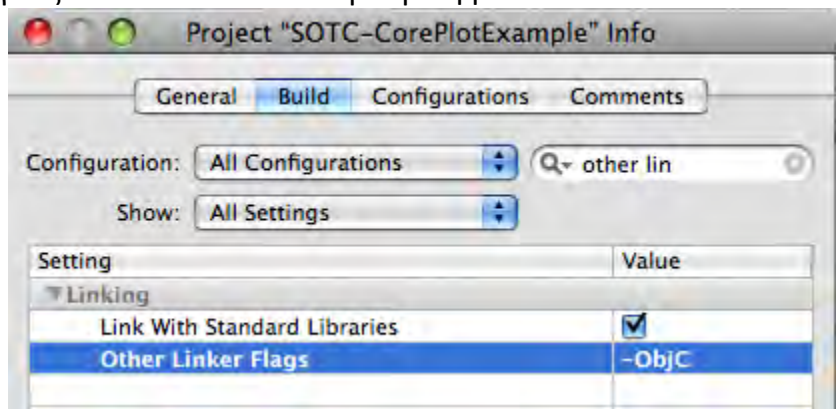
Τώρα πρέπει να «πούμε» στο Xcode ότι η εφαρμογή μας εξαρτάται από την βιβλιοθήκη core-plot. Αυτό γίνεται στο παράθυρο των ρυθμίσεων και συγκεκριμένα στην καρτέλα "General", μπορούμε να προσθέσουμε το «CorePlot-CocoaTouch» ως μια άμεση εξάρτηση:



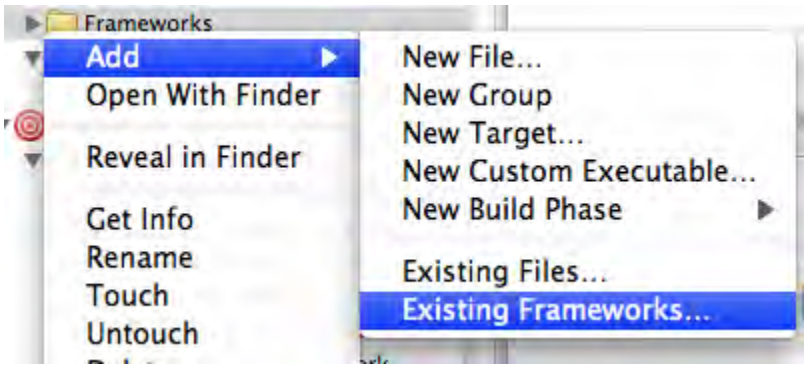
Τώρα θα πρέπει να τροποποιήσουμε ορισμένες ρυθμίσεις.Μπορούμε να το κάνουμε αυτό πηγαίνοντας στη καρτέλα "Build" που βρίσκεται στις ρυθμίσεις του Project μας.Εδώ μέσα υπάρχουν δύο ιδιότητες που πρέπει να τροποποιηθούν: «Header Search Paths» και «Other Linker Flags». Υπάρχουν πολλές ιδιότητες, αλλά ευτυχώς η Apple μας δίνει ένα μικρό κουτί φίλτρου για να βρούμε τις ιδιότητες που θέλουμε. Έτσι, πρώτα στο «Header Search Paths» πρέπει να προσθέσουμε την πλήρη διαδρομή προς το core-plot/framework κατάλογο:



Στη συνέχεια, πρέπει να προστεθεί το linker στην -ObjC , η οποία προκαλεί τη σύνδεση με το «Loads all members of static archive libraries that define an Objective C class or a category».Χρειαζόμαστε αυτή τη σημαία επειδή η Core Plot χρησιμοποιεί Objective C καταλόγους στις υπάρχουσες κλάσεις, και όταν την τρέχουμε ως στατική βιβλιοθήκη, ο κώδικας για αυτές τις κατηγορίες δεν είναι συνδεδεμένες στο τελικό εκτελέσιμο. Αυτή η σημαία αναγκάζει τον κώδικα να συμπεριληφθεί.



Τέλος πρέπει να προσθέσουμε την ενσωματωμένη Core Animation βιβλιοθήκη στο project μας, γιατί η Core plot είναι βρίσκεται στην κορυφή του Core Animation. Για να το κάνουμε αυτό, προσθέτουμε το QuartzCore πλαίσιο. Κάνουμε δεξί κλικ στο φάκελο των Frameworks του "Groups & Files" και επιλέγουμε Add-> Existing Framework και αναζητούμε το QuartzCore :



Είμαστε έτοιμοι να την χρησιμοποιήσουμε. Εάν έχουμε δημιουργήσει ένα απλό παράθυρο όταν αρχίσαμε, έχουμε πιθανώς μια AppDelegate και μια τάξη ViewController στο φάκελο των κλάσεών μας (σε αυτό το παράδειγμα είναι το "CorePlotTestAppDelegate" και "CorePlotTestViewController"). Όλος ο κώδικας που πρόκειται να γράφτεί θα είναι στην ViewController. Το αρχείο για τον ελεγκτή μας (στην περίπτωσή αυτή το "CorePlotTestViewController.h") είναι :

```
#import <UIKit/UIKit.h>

@interface CorePlotTestViewController : UIViewController
{
}

@end
```

Δεν θα αλλάξουν πολλά σε αυτό το αρχείο, αλλά υπάρχουν μερικά πράγματα που θα χρειαστεί να προσθέσουμε:

```
#import <UIKit/UIKit.h>
#import "CorePlot-CocoaTouch.h"

@interface CorePlotTestViewController : UIViewController
<CPPlotDataSource>
{
  CPXYGraph *graph;
}
```

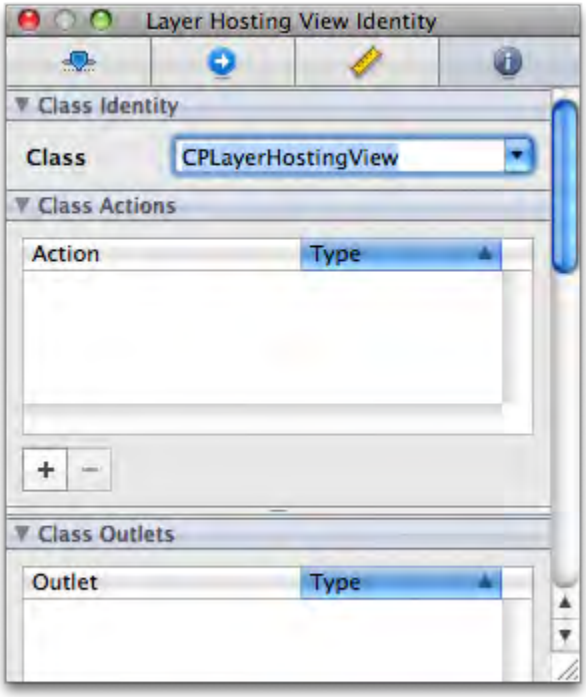
```
@end
```

Πρώτα από όλα, θα πρέπει να δείξουμε την αναφορά της Core Plot βιβλιοθήκης με μια δήλωση εισαγωγής(#import). Στη συνέχεια λοιπόν χρειαζόμαστε μια μεταβλητή (για να κρατήσει το γράφημα μας), καθώς και ένα πρωτόκολλο. Η μεταβλητή είναι αυτονόητη, αλλά το πρωτόκολλο χρειάζεται κάποια εξήγηση. Σε αυτή την περίπτωση, αυτός ο view ελεγκτής πρόκειται να είναι η πηγή δεδομένων για το γράφημά μας, και γι 'αυτό πρέπει να εφαρμόσουμε το πρωτόκολλο CPPlotDataSource . Αυτό το πρωτόκολλο ορίζεται στην Core Plot και αποτελείται από δύο συναρτήσεις:

```
@protocol CPPlotDataSource <NSObject>

-(NSUInteger)numberOfRecords;

@optional

// Implement one of the following
-(NSArray *)numbersForPlot:(CPPlot *)plot
    field:(NSUInteger)fieldEnum
    recordIndexRange:(NSRange)indexRange;

-(NSNumber *)numberForPlot:(CPPlot *)plot
    field:(NSUInteger)fieldEnum
    recordIndex:(NSUInteger)index;

-(NSRange)recordIndexRangeForPlot:(CPPlot *)plot
    plotRange:(CPPlotRange *)plotRect;

@end
```

Λοιπόν, έχουμε τέσσερις συναρτήσεις στην πραγματικότητα, αλλά που πρέπει να υλοποιηθούν μόνο δύο από αυτές. Η πρώτη είναι στην ουσία ένας getter για τον αριθμό των σημείων στη γραφική παράσταση (numberOfRecords), και οι τρεις τελευταίες (εκ των οποίων θα πρέπει να εφαρμόσουμε μία), είναι συναρτήσεις για να πάρει την ειδική τιμή για ένα συγκεκριμένο σημείο δεδομένων.
Πρέπει πριν συνεχίσουμε να προσέξουμε και κάτι ακόμα.Ένα core plot γράφημα δεν μπορεί να φιλοξενηθεί σε ένα οποιοδήποτε UIView - πρέπει να «φιλοξενηθεί» σε ένα «CPLayerHostingView». Για να συμβεί αυτό πρέπει να αλλάξουμε τον βασικό τύπο

της view μας. Έτσι, ανοίγουμε .xib αρχείο αλλάζουμε τον βασικό τύπο της view στο CPLayerHostingView στην καρτέλα «Inspector»:



Ας ρίξουμε μια ματιά και στον κώδικα. Στο παράδειγμα που θα ακολουθήσει , θα ζωγραφίσουμε δύο γραφικές παραστάσεις σε ένα γράφημα – η μία θα είναι η συνάρτηση x ^ 2, και η άλλη θα είναι η 1 / x:

```
-(NSUInteger)numberOfRecords {
  return 51;
}

-(NSNumber *)numberForPlot:(CPPlot *)plot
    field:(NSUInteger)fieldEnum
    recordIndex:(NSUInteger)index
{
```

```
  double val = (index/5.0)-5;

  if(fieldEnum == CPScatterPlotFieldX)
  { return [NSNumber numberWithDouble:val]; }
  else
  {
    if(plot.identifier == @"X Squared Plot")
    { return [NSNumber numberWithDouble:val*val]; }
    else
    { return [NSNumber numberWithDouble:1/val]; }
  }
}
```

Για αυτές τις παραστάσεις, επιστρέφουμε 51 σημεία δεδομένων που εκτείνονται σε μια περιοχή x απο -5 μέχρι 5. Όταν η numberForPlot καλείται, «παιρνάμε» τον πίνακα σημείων και το μετατρέπουμε σε σημεία του x σε μια περιοχή απο -5 μέχρι 5.Εάν η συνάρτηση κλήθηκε ζητώντας την τιμή X (όταν η fieldEnum είναι CPScatterPlotFieldX), μόνο την επιστρέφουμε. Σε αντίθετη περίπτωση, ανάλογα με το το αναγνωριστικό της γραφικής παράστασης, επιστρέφουμε την τιμή είτε στο τετράγωνο της ή ανεστραμμένη.
Έτσι, δημιουργούνται τιμές για το γράφημα.

```
- (void)viewDidLoad {
  [super viewDidLoad];

  graph = [[CPXYGraph alloc] initWithFrame: self.view.bounds];

  CPLayerHostingView *hostingView = (CPLayerHostingView *)self.view;
  hostingView.hostedLayer = graph;
  graph.paddingLeft = 20.0;
  graph.paddingTop = 20.0;
```

```
graph.paddingRight = 20.0;
graph.paddingBottom = 20.0;
```

Από το header file γνωρίζουμε, ότι η γραφική παράσταση είναι μια μεταβλητή , και εδώ την υλοποιούμε. Εδώ δημιουργούμε μια CPXYGraph (αυτή τη στιγμή το μόνο είδος γραφικής που «προσφέρει» η Core Plot), και στη συνέχεια, θέτοντας ως φιλοξενούμενο στρώμα στην view μας (η οποία είναι στην πραγματικότητα μιαCPLayerHostingView χάρη στην αλλαγή που κάναμε στο αρχείο .xib νωρίτερα). Θέσαμε, επίσης, ορισμένα όρια εδώ για να δώσουμε στο γράφημά μας λίγο χώρο στην οθόνη:

```
//////////////////////////////////////////////////////////////////////////////////
CPXYPlotSpace *plotSpace = (CPXYPlotSpace *)graph.defaultPlotSpace;
 plotSpace.xRange = [CPPlotRange plotRangeWithLocation:CPDecimalFromFloat(-6)
    length:CPDecimalFromFloat(12)];
 plotSpace.yRange = [CPPlotRange plotRangeWithLocation:CPDecimalFromFloat(-5)
    length:CPDecimalFromFloat(30)];
```

Μια Core Plot γραφική παράσταση μπορεί να έχει πολλά PlotSpaces.Ένα PlotSpace είναι εκεί όπου υπάρχουν τα plots - αλλά όλα τα plots μοιράζονται την ίδια περιοχή. Σε αυτή την περίπτωση, χρειαζόμαστε μόνο ένα PlotSpace. Εδώ καθορίζουμε την περιοχή των x y του χώρου της γραφικής - ένα πράγμα πρέπει να σημειωθεί είναι ότι οι τιμές έχουν οριστεί με βάση το μήκος και το πλάτος και όχι τις ελάχιστες και μέγιστες τιμές:

```
CPLineStyle *lineStyle = [CPLineStyle lineStyle];
 lineStyle.lineColor = [CPColor blackColor];
 lineStyle.lineWidth = 2.0f;

 axisSet.xAxis.majorIntervalLength = [NSDecimalNumber
```

```
decimalNumberWithString:@"5"];
  axisSet.xAxis.minorTicksPerInterval = 4;
  axisSet.xAxis.majorTickLineStyle = lineStyle;
  axisSet.xAxis.minorTickLineStyle = lineStyle;
  axisSet.xAxis.axisLineStyle = lineStyle;
  axisSet.xAxis.minorTickLength = 5.0f;
  axisSet.xAxis.majorTickLength = 7.0f;
  axisSet.xAxis.axisLabelOffset = 3.0f;

  axisSet.yAxis.majorIntervalLength = [NSDecimalNumber
decimalNumberWithString:@"5"];
  axisSet.yAxis.minorTicksPerInterval = 4;
  axisSet.yAxis.majorTickLineStyle = lineStyle;
  axisSet.yAxis.minorTickLineStyle = lineStyle;
  axisSet.yAxis.axisLineStyle = lineStyle;
  axisSet.yAxis.minorTickLength = 5.0f;
  axisSet.yAxis.majorTickLength = 7.0f;
  axisSet.yAxis.axisLabelOffset = 3.0f;
```

Πάμε τώρα να προσθέσουμε κάποια plots:

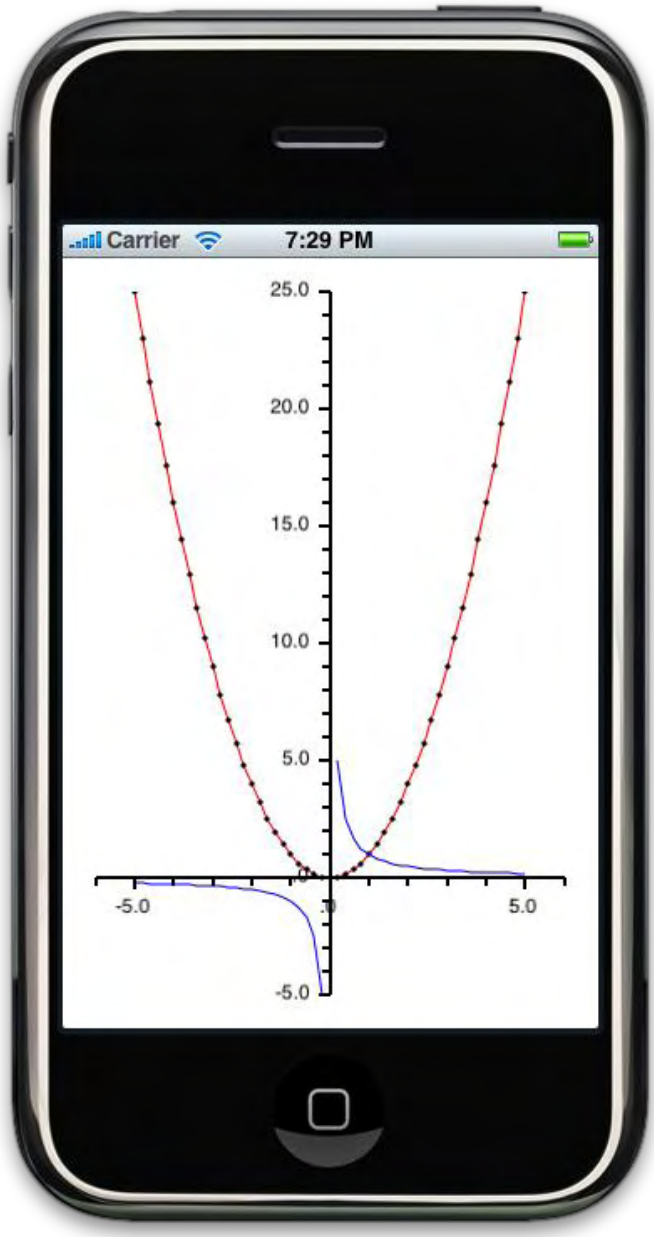
```
CPScatterPlot *xSquaredPlot = [[[CPScatterPlot alloc]
     initWithFrame:graph.defaultPlotSpace.bounds] autorelease];
  xSquaredPlot.identifier = @"X Squared Plot";
  xSquaredPlot.dataLineStyle.lineWidth = 1.0f;
  xSquaredPlot.dataLineStyle.lineColor = [CPColor redColor];
  xSquaredPlot.dataSource = self;
  [graph addPlot:xSquaredPlot];

  CPPlotSymbol *greenCirclePlotSymbol = [CPPlotSymbol ellipsePlotSymbol];
  greenCirclePlotSymbol.fill = [CPFill fillWithColor:[CPColor greenColor]];
  greenCirclePlotSymbol.size = CGSizeMake(2.0, 2.0);
  xSquaredPlot.defaultPlotSymbol = greenCirclePlotSymbol;

  CPScatterPlot *xInversePlot = [[[CPScatterPlot alloc]
     initWithFrame:graph.defaultPlotSpace.bounds] autorelease];
  xInversePlot.identifier = @"X Inverse Plot";
```

```
  xInversePlot.dataLineStyle.lineWidth = 1.0f;
  xInversePlot.dataLineStyle.lineColor = [CPColor blueColor];
  xInversePlot.dataSource = self;
  [graph addPlot:xInversePlot];
}
```

Στον Τάσο και τον Χριστόφορο...

Πάμε λοιπόν να δούμε ένα screenshot για το πως μοιάζει αυτό που φτιάξαμε:

Στον Τάσο και τον Χριστόφορο...



Τέλος εδώ δείχνουμε και το CorePlotTestViewController.m αρχείο :

```
//
//  CorePlotTestViewController.m
//  CorePlotTest
```

```
//

#import "CorePlotTestViewController.h"

@implementation CorePlotTestViewController

- (void)viewDidLoad {
  [super viewDidLoad];

  graph = [[CPXYGraph alloc] initWithFrame: self.view.bounds];

  CPLayerHostingView *hostingView = (CPLayerHostingView
*)self.view;
  hostingView.hostedLayer = graph;
  graph.paddingLeft = 20.0;
  graph.paddingTop = 20.0;
  graph.paddingRight = 20.0;
  graph.paddingBottom = 20.0;

  CPXYPlotSpace *plotSpace = (CPXYPlotSpace
*)graph.defaultPlotSpace;
  plotSpace.xRange = [CPPlotRange
plotRangeWithLocation:CPDecimalFromFloat(-6)
      length:CPDecimalFromFloat(12)];
  plotSpace.yRange = [CPPlotRange
plotRangeWithLocation:CPDecimalFromFloat(-5)
      length:CPDecimalFromFloat(30)];

  CPXYAxisSet *axisSet = (CPXYAxisSet *)graph.axisSet;

  CPLineStyle *lineStyle = [CPLineStyle lineStyle];
  lineStyle.lineColor = [CPColor blackColor];
  lineStyle.lineWidth = 2.0f;

  axisSet.xAxis.majorIntervalLength = [NSDecimalNumber
decimalNumberWithString:@"5"];
  axisSet.xAxis.minorTicksPerInterval = 4;
  axisSet.xAxis.majorTickLineStyle = lineStyle;
  axisSet.xAxis.minorTickLineStyle = lineStyle;
  axisSet.xAxis.axisLineStyle = lineStyle;
  axisSet.xAxis.minorTickLength = 5.0f;
  axisSet.xAxis.majorTickLength = 7.0f;
  axisSet.xAxis.axisLabelOffset = 3.0f;

  axisSet.yAxis.majorIntervalLength = [NSDecimalNumber
```

```
decimalNumberWithString:@"5"];
  axisSet.yAxis.minorTicksPerInterval = 4;
  axisSet.yAxis.majorTickLineStyle = lineStyle;
  axisSet.yAxis.minorTickLineStyle = lineStyle;
  axisSet.yAxis.axisLineStyle = lineStyle;
  axisSet.yAxis.minorTickLength = 5.0f;
  axisSet.yAxis.majorTickLength = 7.0f;
  axisSet.yAxis.axisLabelOffset = 3.0f;

  CPScatterPlot *xSquaredPlot = [[[CPScatterPlot alloc]
      initWithFrame:graph.defaultPlotSpace.bounds] autorelease];
  xSquaredPlot.identifier = @"X Squared Plot";
  xSquaredPlot.dataLineStyle.lineWidth = 1.0f;
  xSquaredPlot.dataLineStyle.lineColor = [CPColor redColor];
  xSquaredPlot.dataSource = self;
  [graph addPlot:xSquaredPlot];

  CPPlotSymbol *greenCirclePlotSymbol = [CPPlotSymbol
ellipsePlotSymbol];
  greenCirclePlotSymbol.fill = [CPFill fillWithColor:[CPColor
greenColor]];
  greenCirclePlotSymbol.size = CGSizeMake(2.0, 2.0);
  xSquaredPlot.defaultPlotSymbol = greenCirclePlotSymbol;

  CPScatterPlot *xInversePlot = [[[CPScatterPlot alloc]
      initWithFrame:graph.defaultPlotSpace.bounds] autorelease];
  xInversePlot.identifier = @"X Inverse Plot";
  xInversePlot.dataLineStyle.lineWidth = 1.0f;
  xInversePlot.dataLineStyle.lineColor = [CPColor blueColor];
  xInversePlot.dataSource = self;
  [graph addPlot:xInversePlot];
}

-(NSUInteger)numberOfRecords {
  return 51;
}

-(NSNumber *)numberForPlot:(CPPlot *)plot
field:(NSUInteger)fieldEnum
      recordIndex:(NSUInteger)index
{
  double val = (index/5.0)-5;
  if(fieldEnum == CPScatterPlotFieldX)
  { return [NSNumber numberWithDouble:val]; }
  else
```

```
  {
    if(plot.identifier == @"X Squared Plot")
    { return [NSNumber numberWithDouble:val*val]; }
    else
    { return [NSNumber numberWithDouble:1/val]; }
  }
}

@end
```

Στον Τάσο και τον Χριστόφορο...

Βιβλιογραφία(links)

1) **Stanford University http://itunes.apple.com/us/itunes-u/iphone-application-development/id384233225**
2) **Core Plot: http://www.switchonthecode.com/tutorials/using-core-plot-in-an-iphone-application**
3) **LocalMaxQuadratic:http://answers.yahoo.com/question/index?qid=20110130131629AAaCSqI**
4) **Lynda: http://www.lynda.com/**