ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ
ΥΠΟΛΟΓΙΣΤΩΝ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ ΚΑΙ ΔΙΚΤΥΩΝ

Προσομοίωση της υπολογιστικής μηχανής Xcas στο iPad

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Τσακμαλής Κυριάκος

Βόλος, Μάρτιος 2012

Προσομοίωση της υπολογιστικής μηχανής Xcas στο iPad

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Τσακμαλής Κυριάκος

Επιβλέπων: Αλκιβιάδης Ακρίτας
Αναπλ. Καθηγητής Π.Θ.

(Υπογραφή) (Υπογραφή)

....
Αλκιβιάδης Ακρίτας Δασκαλοπούλου Ασπασία

Αναπληρωτής Καθηγητής Επίκουρη Καθηγήτρια
Παν. Θεσσαλίας Παν. Θεσσαλίας

(Υπογραφή)

.....
Τσακμαλής Κυριάκος

Διπλωματούχος Μηχανικός Ηλεκτρονικών Υπολογιστών,
Τηλεπικοινωνιών και Δικτύων Πανεπιστημίου Θεσσαλίας

Πρόλογος

Η πτυχιακή εργασία "Προσομοίωση της υπολογιστικής μηχανής Xcas σε iPad" αναφέρεται σε ένα σχετικά καινούργιο κομμάτι στον τομέα της ανάπτυξης εφαρμογών, που χρονολογείται από τις αρχές του 2010 (από το 2007 για εφαρμογές σε iPhone). Στη συνέχεια θα παρουσιάσουμε τον τρόπο που έγινε compile η βιβλιοθήκη Giac η οποία είναι υλοποιημένη σε C++ και έχει αναπτυχθεί από τον Γάλλο καθηγητή *Bernard Parisse* του πανεπιστημίου της *Grenoble*. Στις σελίδες που ακολουθούν θα αναφερθούμε σε άλλες αντίστοιχες εφαρμογές σε ανάλογα λειτουργικά συστήματα, στο λόγο που διαλέξαμε τη βιβλιοθήκη Giac, αλλά και πώς έγινε το compile, στον τρόπο που αναπτύξαμε το interface της εφαρμογής μας καθώς και στην ανάλυση του κώδικα.

Συγκεκριμένα στο πρώτο κεφάλαιο θα γίνει μια σύντομη αναφορά σε διάφορες αντίστοιχες εφαρμογές που υπάρχουν στην αγορά και το λόγο που διαλέξαμε να αναπτύξουμε την εφαρμογή μας στο λειτουργικό σύστημα του iOS.

Στο δεύτερο κεφάλαιο θα αναφερθούμε στη βιβλιοθήκη Giac καθώς και στον τρόπο που καταφέραμε να την κάνουμε compile και να τη χρησιμοποιήσουμε στην εφαρμογή μας.

Στο τρίτο κεφάλαιο θα αναφερθούμε στο εργαλείο του XCode που χρησιμοποιήσαμε για να αναπτύξουμε την εφαρμογή μας.

Στο τέταρτο κεφάλαιο θα γίνει εκτενέστερη ανάλυση στον κώδικα της εφαρμογής και θα αναλυθούν οι λόγοι επιλογής της συγκεκριμένης μορφοποίησης.

Κατάλογος περιεχομένων

1.Αντίστοιχες Εφαρμογές που υπάρχουν στην Αγορά

1.1 Εφαρμογές σε iPhone/iPad

Στο λειτουργικό iOS ,που χρησιμοποιούν οι συσκευές iPad και iPhone ,έχουν αναπτυχθεί πολλές εφαρμογές (από το 2007 για το iPhone και από το 2010 για το iPad) και οι οποίες σε μεγάλο ποσοστό είναι επί πληρωμή. Στον τομέα των calculators υπάρχουν γύρω στις 250 εφαρμογές για iphone και γύρω στις 200 για iPad, όμως από αυτές γύρω στο 90% είναι σαν τα"απλά" κουμπιουτεράκια που χρησιμοποιούμε καθημερινά όλοι μας και που μπορούν να κάνουν ένα μικρό σύνολο πράξεων. Το υπόλοιπο 10% αφορά "επιστημονικά" κουμπιουτεράκια των οποίων οι λειτουργικότητες διαφέρουν μεταξύ τους, άλλες εμπεριέχουν μαθηματικές συναρτήσεις, άλλες μπορούν να λειτουργήσουν σαν υπολογιστικές μηχανές και άλλες έχουν τη δυνατότητα να σχηματίσουν τρισδιάστατες ή διδιάστατες γραφικές παραστάσεις.

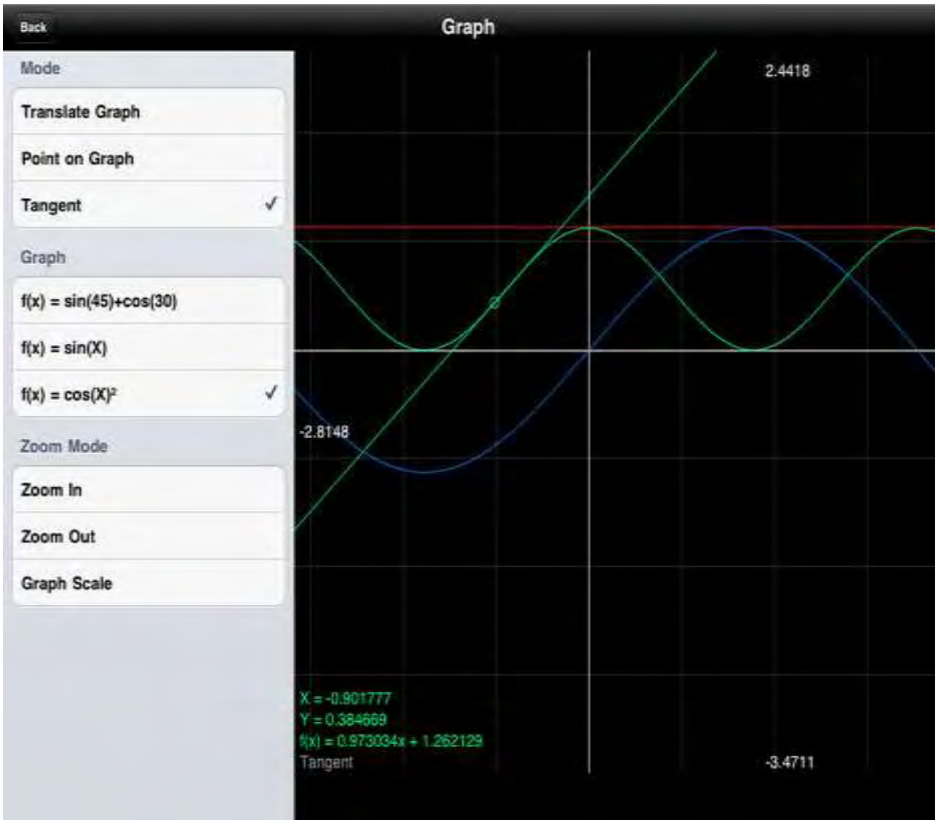
Οι βιβλιοθήκες που χρησιμοποιούν οι εφαρμογές αυτές είναι είτε ελεύθερα προσβασιμες, είτε δημιούργημα των ίδιων των developer. Τη βιβλιοθήκη Giac που χρησιμοποιούμε στην εφαρμογή μας τη χρησιμοποιεί και μία άλλη εφαρμογή με ονομασία "PocketCAS". Εμείς χρησιμοποιούμε ορισμένες συναρτήσεις της βιβλιοθήκης και όχι όλες τις δυνατότητες της (π.χ. για κατασκευή γραφικών παραστάσεων).

Η εφαρμογή PocketCAS είναι μια απλή υπολογιστική μηχανή η οποία έχει ένα διαφορετικό interface από τη δικιά μας εφαρμογή. Στην εικόνα 1.1 μπορούμε να δούμε ένα μικρό δείγμα της εφαρμογής αυτής σε περιβάλλον iPad. Σημειωτέον θα πρέπει να αναφέρουμε ότι το *PockeCAS* είναι μία από τις δημοφιλέστερες και τις πληρέστερες εφαρμογές τέτοιου είδους.

Εικόνα 1.1

Μία άλλη εφαρμογή αυτού του είδους είναι και ιδιαίτερα δημοφιλής είναι η *CalcPro* η οποία έχει παρόμοιες δυνατότητες με την προηγούμενη, αλλά δεν έχει συναρτήσεις για απαιτητικούς χρήστες. Ένα δείγμα αυτής μπορούμε να δούμε στην εικόνα 1.2.

Κάτι το οποίο θα πρέπει να επισημάνουμε σε αυτό το σημείο είναι ότι η οικονομική πολιτική που ακολουθούν οι developers είναι να προωθούν μια *free demo* έκδοση στο ***apple store*** προκειμένου να προωθήσουν το προϊόν και για να συλλέξουν πληροφορίες αν εξυπηρετεί τους χρήστες.



Εικόνα 1.2

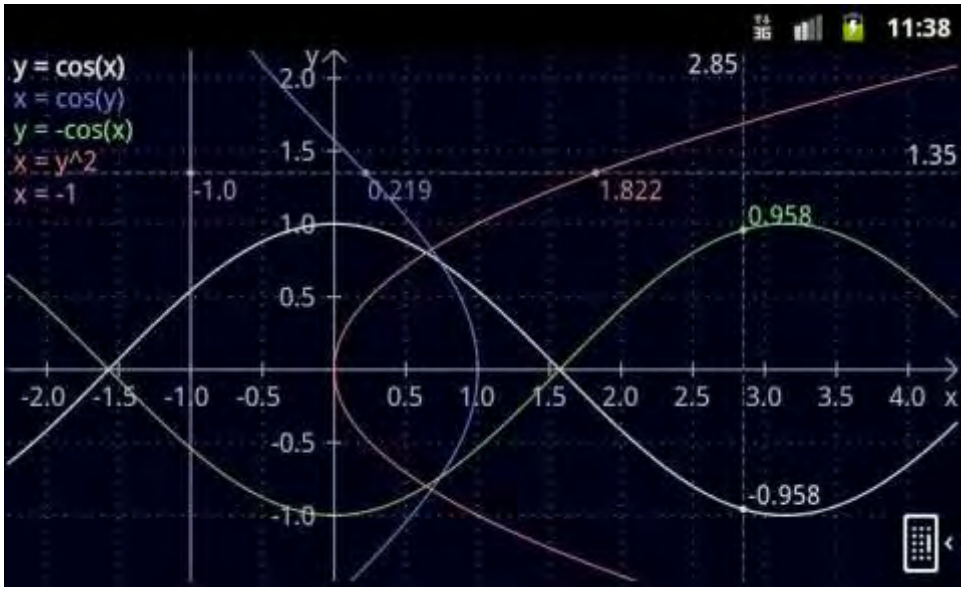
Τέλος, η επόμενη εφαρμογή είναι ένας ισάξιος “αντίπαλος” του PocketCAS στις προτιμήσεις των χρηστών αλλά και στις δυνατότητες. Στην εικόνα 1.3 φαίνεται η εφαρμογή *HiCalcPro*.

Εικόνα 1.3

1.2 Εφαρμογές σε λειτουργικό Android

Αντίστοιχα στο λειτουργικό **android** έχουμε επίσης μεγάλη ποικιλία εφαρμογών. Επίσης, αξιοσημείωτο είναι ότι το μεγαλύτερο μέρος των εφαρμογών είναι δωρεάν και ότι έτσι ο χρήστης έχει τη δυνατότητα να κάνει όποια επιλογή επιθυμεί. Όμως, το γεγονός ότι μεγάλο μέρος της αγοράς στο ***android market*** είναι free εφαρμογές, είναι ένα αρνητικό στοιχείο που επηρεάζει έμμεσα τις εφαρμογές. Εννοούμε ότι επηρεάζει έμμεσα την ποιότητα των εφαρμογών, αλλά χωρις να δυσαρεστεί αρκετά τον χρήστη.

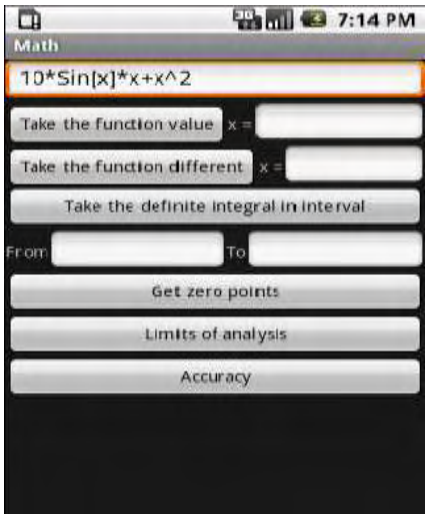
Μια πολύ δημοφιλής εφαρμογής σε android είναι το *Mathlab Caclulator.* Είναι μία από τις 350 εφαρμογές Calculator που υπάρχουν σε περιβάλλον android. Υπολογίζει γραφικές παραστάσεις πολυονύμων και ένα στιγμιότυπο της εφαρμογής φαίνεται στην εικόνα 1.4.



Εικόνα 1.4

Άλλες 2 εφαρμογές που μπορούμε να πούμε ότι συγκαταλλέγονται στην ίδια κατηγορία με την παραπάνω εφαρμογή είναι η Calc-rGraphPlotter-Analysis και η Scientific-Calculator που φαίνονται στην εικόνα 1.5 και 1.6 αντίστοιχα.

Εικόνα 1.5



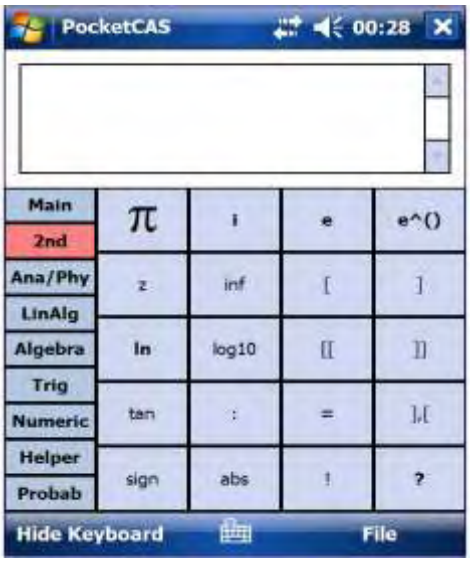
Εικόνα 1.6

1.3 Εφαρμογές σε λειτουργικό Windows Mobile

Το λειτουγικό ***window mobile*** ήταν το πρώτο λειτουργικό που ξεκίνησε να κυκλοφορεί στην αγορά, πολύ νωρίτερα από τα υπόλοιπα λειτουργικά συστήματα για κινητά τηλέφωνα. Βέβαια η πρώτη του έκδοση δεν αφορούσε κινητά τηλέφωνα και ήταν η *Pocket PC 2000* και κυκλοφόρησε το 2000. Η έκδοση που προσπάθησε να "αντιμετωπίσει" με ισάξιες προσδοκίες το iOS ήταν η έκδοση Windows Mobile 6 που βγήκε στην αγορά το 2007.

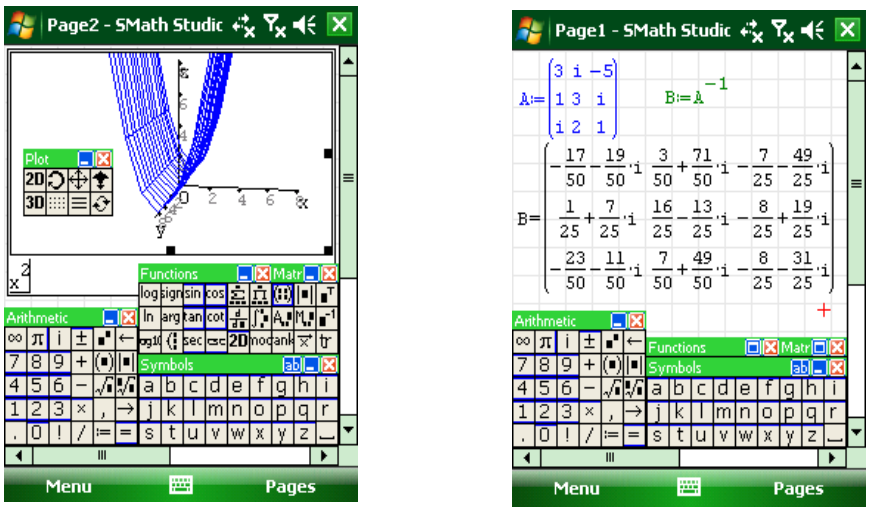
Αρκετές calculator εφαρμογές μπορεί να συναντήσει ο χρήστης στο διαδίκτυο. Ο αριθμός τους βέβαια δεν ξεπερνάει τις 100 και πολλές από αυτές είναι σαν ένα απλό κλασικό κουμπιουτεράκι. Όμως οι εφαρμογές που χρησιμοποιούν συναρτήσεις και υπολογίζουν γραφικές παραστάσεις είναι ιδιαίτερα "ελκυστικές". Δύο από αυτές τις αναφέραμε και στο λειτουργικό iOS (την CalcPro τη βλέπουμε στην εικόνα εικόνα 1.7 και την PockeCas τη βλέπουμε στην εικόνα 1.8),η δεύτερη κάνει χρήση της βιβλιοθήκης Giac που προαναφέραμε.

Εικόνα 1.7



Εικόνα 1.8

Μια τελευταία εφαρμογή που είναι και η παλαιότερη απ'ολες τις υπόλοιπες, όπως επίσης και μια από τις πιο αξιόλογες είναι η SMath η οποία υπάρχει και σε λειτουργικά συστήματα για Υπολογιστές. Η SMath υπάρχει σαν ελεύθερη εφαρμογή για Υπολογιστές και ένα δείγμα αυτής της εφαρμογής μπορούμε να δούμε στην εικόνα 1.9.



Εικόνα 1.9

1.4 Λόγοι που επιλέξαμε το λειτουργικό iOS

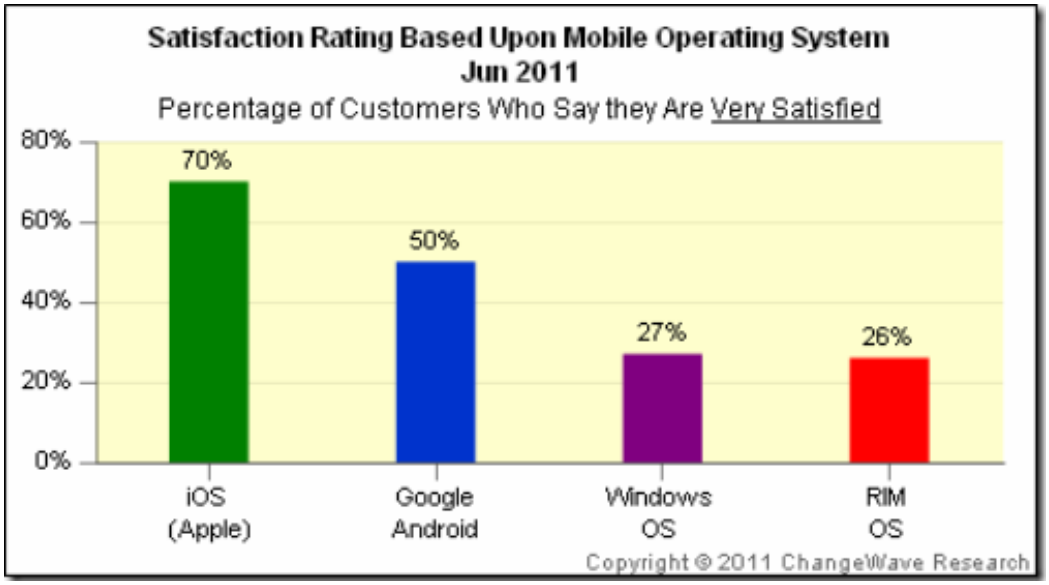
Από την αρχή της ύπαρξης και των 3 λειτουργικών συστημάτων φάνηκε πως θα υπάρξει πολύ δυνατός ανταγωνισμός μεταξύ τους.Όμως κατά τη διάρκεια της πορείας τους φάνηκε να κερδίζει και να ξεχωρίζει το λειτουργικό iOS. Πολλοί χρήστες ανυπομονούν την επόμενη έκδοσή του.

Πολλά στοιχεία των λειτουργικών Android και Windows mobile είναι επηρεασμένα από καινοτομίες που πρωτοεμφανίστηκαν στα iOS λειτουργικά. Χαρακτηριστικά μπορούμε να αναφέρουμε τις πολλές δυνατότητες που μας προσφέρει η οθόνη αφής όπως επίσης και η δυνατότητα χρήσης του *accelerometer* που πρόσφερε στις εφαρμογές πολλές δυνατότητες. Στις παρακάτω εικόνες φαίνεται ότι τα Android έχουν το μεγαλύτερο μέρος της αγοράς (εικόνα 1.10), αλλά το λειτουργικό iOS ικανοποιεί περισσότερο τους χρήστες του (εικόνα 1.11).

Worldwide Smartphone Operating System 2011 and 2015 Market Share and 2011-2015 CAGR

Operating System	2011 Market Share	2015 Market Share	2011-2015 CAGR
Android	39.5%	45.4%	23.8%
BlackBerry	14.9%	13.7%	17.1%
iOS	15.7%	15.3%	18.8%
Symbian	20.9%	0.2%	-65.0%
Windows Phone 7/Windows Mobile	5.5%	20.9%	67.1%
Others	3.5%	4.6%	28.0%
Total	100.0%	100.0%	19.6%

Εικόνα 1.10



Εικόνα 1.11

Ένας από τους λόγους που διαλέξα το λειτουργικό του iOS είναι ότι σαν χρήστης ,και των 3 λειτουργικών ,κατάφερε να με κερδίσει από την πρώτη στιγμή. Κανένα πρόβλημα χρηστικότητας και απόδοσης δε με έχει απασχολήσει μέχρι κι αυτή τη στιγμή.

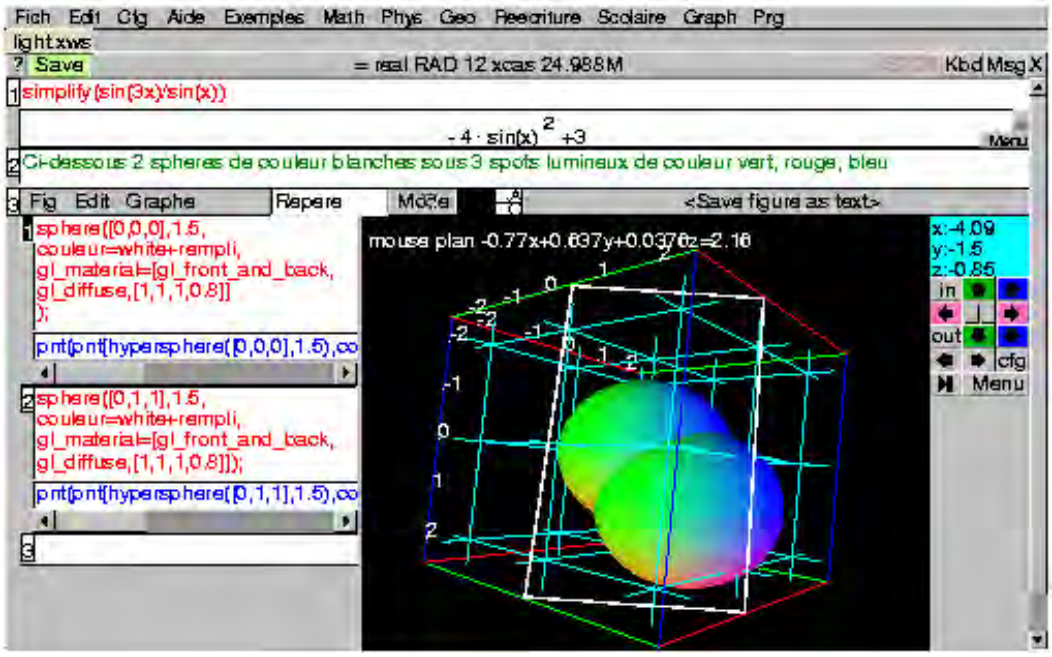
Επίσης, έχω ασχοληθεί με την ανάπτυξη εφαρμογών στα λειτουργικά Android και iOS και διέκρινα πως το Xcode ,που μου παρέχει η Apple για την ανάπτυξη εφαρμογών για τα iOS ,είναι πιο φιλικό στον προγραμματιστή και αρκετά πιο εύχρηστο σε σχέση με το αντίστοιχο για Android. Όμως ,θα πρέπει να αναφέρουμε ότι ο προγραμματιστής σε Android έχει δικαίωμα να διαλέξει ανάμεσα σε διάφορες γλώσσες προγραμματισμού όπως είναι η Java ή η C++, αλλά και θα πρέπει να γνωρίζει xml. Αντίθετα ο προγραμματιστής σε iOS θα πρέπει να ξέρει μόνο καλή

χρήση της Oblective-C και τίποτε περισσότερο (η apple παρέχει δωρεάν όσες γνώσεις χρειάζεται να έχει ο προγραμματιστής σε iOS).

Τέλος θα πρέπει να αναφερθούμε στο γεγονός ότι για την ανάπτυξη εφαρμογών στα λειτουργικά αυτά είναι δωρεάν, αλλά για να μπορέσουμε να ολοκληρώσουμε και να διαθέσουμε τις εφαρφογές στα *markets* λειτουγρικών θα πρέπει να πληρώσουμε ένα ποσό (με το οποίο θα έχουμε δικαίωμα έκδοσης εφαρμογών για 1 χρόνο) .

Οι παραπάνω λόγοι καθώς και η ενθάρρυνση του καθηγητή κ.Ακρίτα ήταν αρκετοί για να διαλέξουμε το λειτουργικό iOS.

2.Βιβλιοθήκη Giac



2.1 Στοιχεία της Giac

Η Giac είναι μια βιβλιοθήκη η οποία διατίθεται δωρεάν στους προγραμματιστές και είναι γραμμένη σε C++. Είναι ένα μεγάλο κομμάτι κώδικα το οποίο μπορεί να χρησιμοποιηθεί από άλλα προγράμματα C++ ή και άλλα προγράμματα C με τις κατάλληλες ρυθμίσεις. Μπορούμε επίσης να τη χρησιμοποιήσουμε από τη γραμμή εντολών του λειτουργικού μας.
Η βιβλιοθήκη αυτή χρησιμοποιείται από την Ελεύθερη προς χρήση Υπολογιστική Μηχανή Xcas, την οποία χρησιμοποιήσα κατά τη διάρκεια των σπουδών μου. Αποτελεί ένα πού καλό εργαλείο για Μηχανικούς με πολλές δυνατότητες, χρησιμοποιώντας ιδιαίτερα καλούς αλγορίθμους. Το Xcas αποτελεί επίσης και ένα

εργαλείο εύκολο για χρήση από μαθητές. Περιέχει tutorial που δείχνει στους μαθητές πώς μπορούν να το χρησιμοποιήσουν για τις σπουδές τους.

Το Xcas μπορεί να εκτελέσει Αλγεβρικούς Υπολογισμούς, να σχεδιάσει γραφήματα (Τρισδιάστατα και Διδιάστατα), κάνει στατιστικές αναλύσεις και να προγραμματίσει. Εξαιτίας αυτών των δυνατοτήτων το επιλέξαμε κατά τη διάρκεια των σπουδών μας, αλλά και για την παρούσα εργασία.

2.2 Compile της Giac

Πρώτα απ' όλα θα ήθελα να ευχαριστήσω το φίλο και συνάδερφο Σπύρο Κεχαγιά για την προσπάθεια που κατέβαλε για να μεταγλωτίσει τη βιβλιοθήκη της GIAC και για τις συμβουλές που μου έδωσε πάνω στη μεταγλώτιση της βιβλιοθήκης αυτής. Στη συνέχεια θα αναφέρουμε τη διαδικασία που ακολουθήσουμε για να επιτευχθεί η μεταγλώτιση.

Cross_Compile => Η διαδικασία κατά την οποία δημιουργούμε κώδικα, ο οποίος θα μπορεί να εκτελεστεί σε διαφορετική πλατφόρμα (στην περίπτωση μας για iOS συσκευές) από αυτή που τρέχει ο μεταγλωτιστής. Για να κάνουμε αυτή την ενέργεια χρειάζεται να χρησιμοποιήσουμε το ***script*** "build_for_iphones", το οποίο βρίσκεται στο αρχείο της GIAC.

Για να μπορέσουμε να μεταγλωτίσουμε/χρησιμοποιήσουμε τη βιβλιοθήκη θα χρειαστεί να έχουμε προ-μεταγλωτίσει τις βιβλιοθήκες libiconv, ncurses, gettext, readline και μια βιβλιοθήκη μεγάλης ακριβείας (μπορεί να είναι ή η GMP ή η tommath 0.39, αλλά χρησιμοποιήσαμε τη 2η γιατί η 1η μας δημιούργησε ορισμένα προβλήματα). Αξιοσημείωτο είναι ότι θα μπορούμε μαζί με τη GIAC να χρησιμοποιήσουμε και άλλες βιβλιοθήκες όπως είναι οι εξής: NTL, Pari, GSL κλπ οι οποίες δε χρησιμοποιήθηκαν στην παρούσα εργασία.

Όπως αναφέραμε προηγουμένως η GIAC εξαρτάται από άλλες βιβλιοθήκες. Αλλά επίσης αυτές οι βιβλιοθήκες εξαρτώνται από άλλες. Βάση των εξαρτήσεων που δημιουργούνται ανάμεσα στις βιβλιοθήκες θα πρέπει να γίνει και η μεταγλώτιση των βιβλιοθηκών. Έτσι η σειρά με την οποία πρέπει να μεταγλωτιστούν είναι η εξής: libiconv, ncurses, termcap, gettext, readline, tommath, Giac. (Παραδείγματα εξαρτήσεων: Η Giac εξαρτάται από τη gettext και την readline, ενώ η readline εξαρτάται από τη gettext).

Πρέπει να σημειώσουμε επιπλέον ότι όταν εγκαθιστούμε τις βιβλιοθήκες στο σύστημα (Mac OS) παίρνουμε στατικές βιβλιοθήκες, οι οποίες είναι μία συλλογή αρχείων με την κατάληξη **".o"**. Επίσης οι βιβλιοθήκες με τη σειρά τους είναι αρχεία με κατάληξη **".α".**

<u>Μεταγλώτιση της libiconv:</u>

-Αποσυμπιέζουμε το αρχείο libiconv.

-Τοποθετούμε το script "build-for-iphoneos" στο αρχείο libiconv.

-Μέσω terminal μεταβαίνουμε στο φάκελο του libiconv.

-Πληκτρολογούμε την εξής εντολή:

./build_for_iphoneos device –prefix=installationPath (για τη συσκευή)

ή

./build_for_iphoneos simulator –prefix=installationPath (για το simulator)

(όπου installationPath είναι εκεί που εγκαθιστούμε όλες τις βιβλιοθήκες)

Μεταγλώτιση της ncurses:

-Όμοια με τη libiconv.

-Αν δε μεταγλωτιστεί επιτυχώς η tempcap.h, τη μεταγλωτίζουμε κι αυτή.

Μεταγλώτιση της termcap:

-Όμοια με τη libiconv, απλά αντικαθιστούμε στο αρχείο configure (της βιβλιοθήκης termcap) τη λέξη “gcc” με τη λέξη “llrm-gcc”.

Μεταγλώτιση της gettext:

-Όμοια με τη libiconv.

Μεταγλώτιση της readline:

-Όμοια με τη libiconv.

Μεταγλώτιση της GIAC:

-Όμοια με τη libiconv, απλά η εντολή στο terminal είναι η εξής:

./build_for_iphoneos device --prefix=installationPath –enable-tommath

Με τη μεταγλώτιση της GIAC στο installationPath βρίσκονται όλες οι μεταγλωτισμένες βιβλιοθήκες

3.Xcode

3.1 Η ιστορία του Xcode

Το Xcode είναι μια "σουίτα" εργαλείων, που αναπτύχθηκε από την Apple, για την ανάπτυξη εφαρμογών για λογισμικό Mac OS X και iOS. Η κύρια εφαρμογή της "σουίτας" είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης (IDE), που ονομάζεται επίσης Xcode. Περιλαμβάνει τα περισσότερα έγγραφα προγραμματιστών της Apple όπως επίσης και το *Interface Builder* που χρησιμοποιείται για την κατασκευή των γραφικών τμημάτων της εφαρμογής. Το Xcode περιλαμβάνει μια τροποποιημένη έκδοση του ελεύθερου λογισμικού *GNU* και υποστηρίζει C, C + +, Objective-C, Java, Python.

Μέχρι αυτή τη στιγμή έχουν κυκλοφορήσει 4 εκδόσεις του Xcode που σταδιακά γίνονται φιλικότερες στον προγραμματιστή και προσθέτουν επιπλέον λειτουργίες. Η 1η έκδοση κυκλοφόρησε το 2003, ενώ 2η συμπεριλαμβανόταν στο λογισμικό *"Tiger"* της APPLE. Η έκδοση 2.0 περιελάμβανε την ***Quartz Composer*** καθώς επίσης και το ***Apple Reference Library tool*** με το οποίο μπορούσε οποιοσδήποτε να διαβάσει online έγγραφα από το επίσημο website της Apple. Επίσης η έκδοση 2.1 μπορούσε να δημιουργήσει εκτελέσιμα.

Η 3η έκδοση κυκλοφόρησε μαζί με το λειτουργικό "Leopard", η οποία περιείχε εργαλείο εντοπισμού σφαλμάτων *Dtrace*. Η έκδοση 3.1 ήταν κάτι πρωτοπόρο για την ανάπτυξη εφαρμογών Mac OS X και περιελάμβανε το iPhone SDK. Ένα ακόμα χαρακτηριστικό της 3.1 είναι η SCM υποστήριξη, η οποία μπορούσε να υποστηρίξει Subversion 1.5. Η συνέχεια της 3.1 ήταν η 3.2 (η οποία χρησιμοποιήθηκε για την ανάπτυξη της εργασίας) που περιλαμβανόταν στο λειτουργικά "Snow Leopad" και η οποία υποστηρίζει *static program analysis* και άλλα επιπρόσθετα στοιχεία.Η τελευταία έκδοση του Xcode είναι η 4η η οποία κυκλοφόρησε τον Μάρτιο του 2011 και η οποία ενσωματώνει όλα τα τα στοιχεία του Xcode σε ένα περιβάλλον (εικόνα αυτού φαίνεται στην εικόνα 3.1).

3.2 Χαρακτηριστικά του Xcode

Αρχικά το Xcode μας καλοσορίζει και μας παρέχει τις εξής επιλογές:

Getting Started: Η επιλογή αυτή μας παρέχει μια λίστα από links με πόρους και έγγραφα για την "ομαλή εισαγωγή" του προγραμματιστή στο χώρο των Apple εφαρμογών.

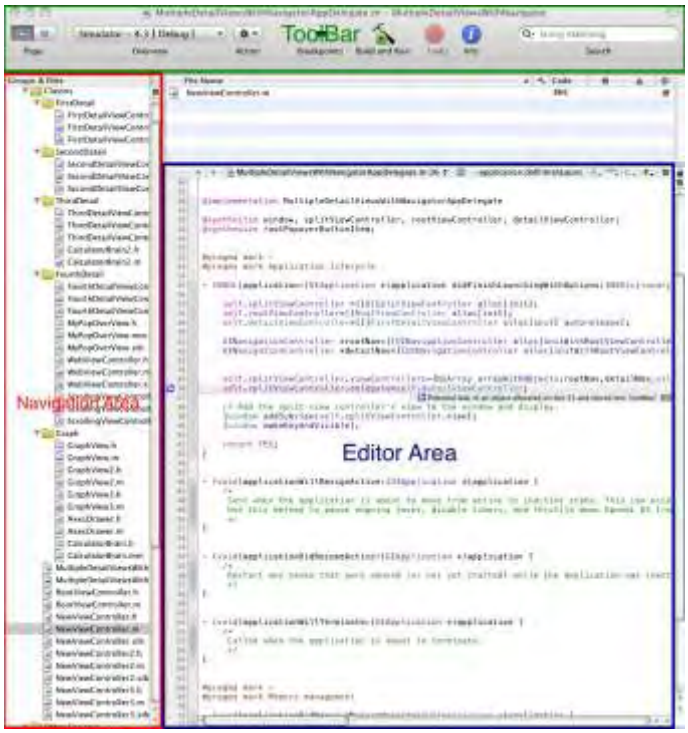
Xcode News: Είναι ένα ενσωματωμένο RSS που παρέχει στον προγραμματιστή πληροφορίες, απλά κομμάτια κώδικα και ανακοινώσεις για το development στα Mac OS X λειτουργικά.

Documentation: Γρήγορη πρόσβαση σε έγγραφα όπως επίσης και συμβουλές και γεγονότα που αφορούν το Xcode.

Mailing Lists: Γρήγορη πρόσβαση σε "mailing lists" που αφορούν την Apple.

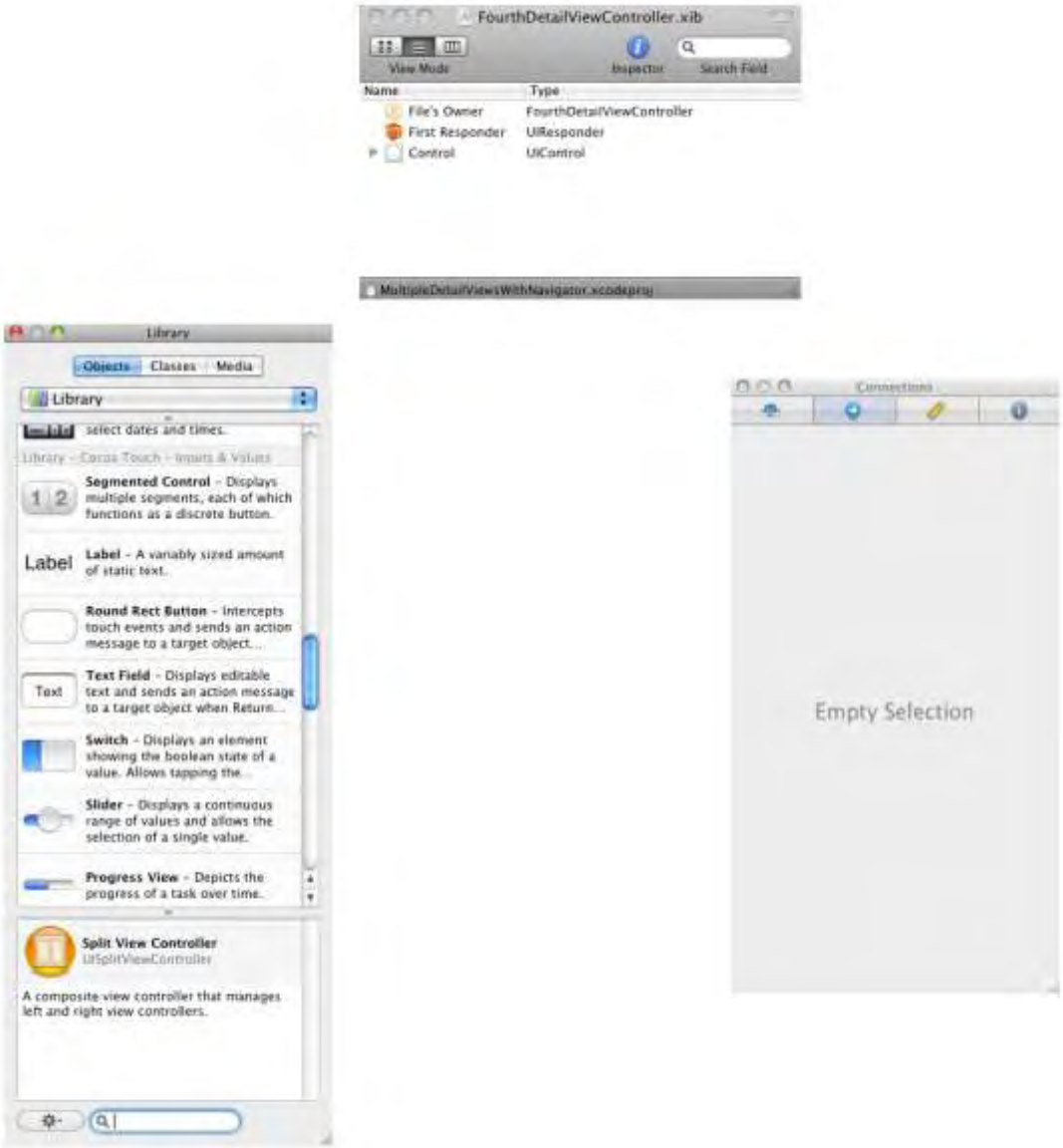
FeedBack: Γρήγορες συνδέσεις στους διάφορους μηχανισμούς ανάδρασης.

Στη συνέχεια εμφανίζεται στην οθόνη μας το κύριο παράθυρο του Xcode που αποτελείται από το ***Navigation Area***, ***Editor Area*** και το ***ToolBar*** (στην εικόνα 3.1 φαίνεται το κύριο παράθυρο).



Εικόνα 3.1

Επίσης, ένα ακόμα εργαλείο που συνοδεύεται με το Xcode είναι το *Interface Builder* με τη βοήθεια του οποίου μπορούμε να δημιουργήσουμε και να σχεδιάσουμε όπως θέλουμε το Interface των εφαρμογών μας. Στην εικόνα 3.2 φαίνεται η κονσόλα του Interface Builder (η οποία αποτελείται από 3 διαφορετικά μέρη).


FourthDetailViewController.xib
View Mode
Inspector
Search Field
Name
Type
File's Owner
FourthDetailViewController
First Responder
UIResponder
Control
UIControl
Library
Objects
Classes
Media
Library
Segmented Control - Displays multiple segments, each of which functions as a discrete button.
Label - A variably sized amount of static text.
Round Rect Button - Intercepts touch events and sends an action message to a target object...
Text Field - Displays editable text and sends an action message to a target object when Return...
Switch - Displays an element showing the boolean state of a value. Allows tapping the...
Slider - Displays a continuous range of values and allows the selection of a single value.
Progress View - Depicts the progress of a task over time.
Split View Controller
UISplitViewController
A composite view controller that manages left and right view controllers.
Connections
Empty Selection

Εικόνα 3.2

3.3 Ανάλυση στοιχείων του Xcode

Όπως αναφέραμε στο κεφάλαιο 3.2 υπάρχουν 3 επιμέρους τμήματα του Interface Builder. Αυτά τα τμήματα είναι η ***Library***, ο ***Inspector*** και το ***Κύριο Μέρος του Interface Builder***.

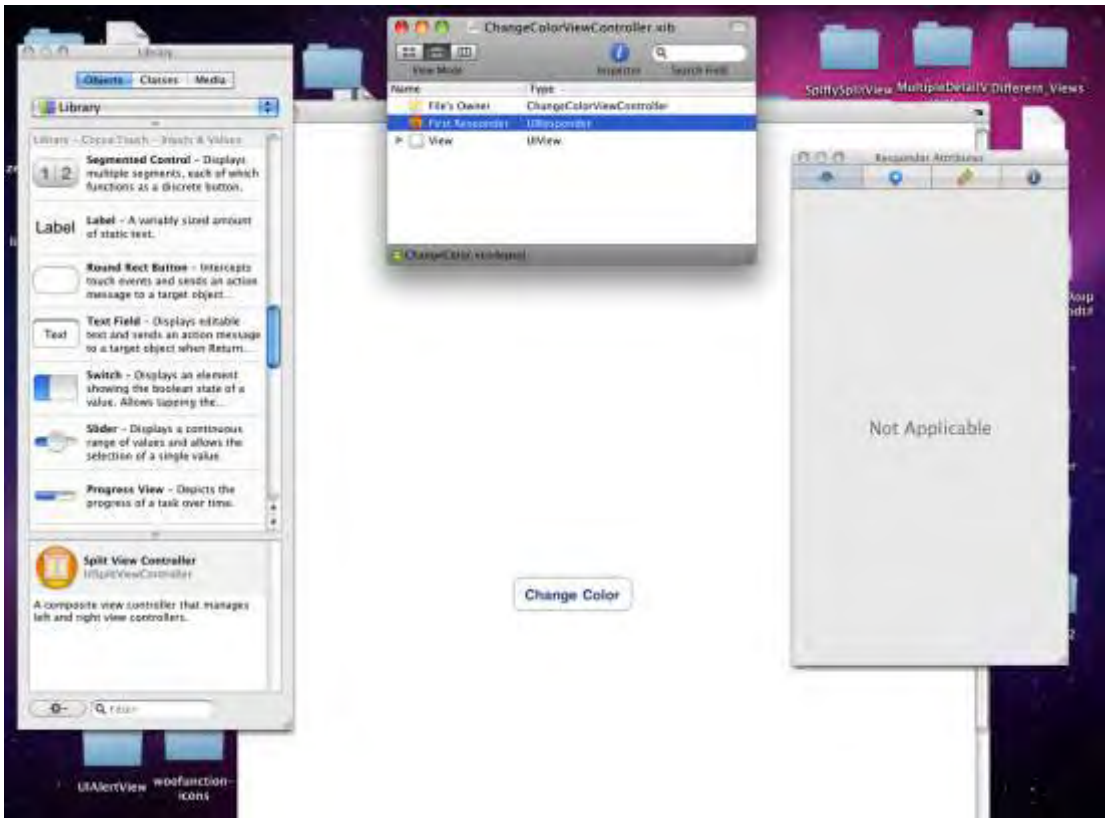
Η Library είναι μια "xcode βιβλιοθήκη" όπου μπορεί ο προγραμματιστής να να βρεί όλα τα απαραίτητα στοιχεία που χρειάζεται για να δημιουργήσει ένα πλήρες πρόγραμμα. Μπορεί να βρεί *Controllers, Data Views, Inputs, Values, Windows, Views* και *Bars.* Επίσης, υπάρχουν και όλες οι Κλάσεις που μπορεί να χρειαστούν στον προγραμματιστή και οι οποίες ταξινομούνται αλφαβητικά. Τέλος, υπάρχουν και πληροφορίες για τις Κλάσεις όπως "ποιά είναι η κληρονομική τους κατάσταση", ή "πώς ορίζεται μια κλάση στο πρόγραμμα"

Το Κύριο Μέρος του Interface Builder είναι το τμήμα που περιέχει όλα τα στοιχεία του Interface της εφαρμογής, από ένα απλό κουμπί μέχρι ένα ολόκληρο Window. Έτσι, μέσω αυτού μπορούμε να διακρίνουμε όλα τα στοιχεία του προγράμματος, καθώς και τις ενέργειες που έχουν.

Ο Inspector είναι χωρισμένος σε 4 τμήματα, τα *Attributes*, τα *Connection*, το *Size*, και το *Identity*. Αυτά μας παρέχουν τις πληροφορίες, που χρειάζεται ο προγραμματιστής, για τα στοιχεία που βρίσκοντα στο Κύριο Μέρος του Interface Builder καθώς και πως συνδεόνται μεταξύ τους.

Για καλύτερη ανάλυση του Interface Builder θα δημιουργήσουμε βήμα-βήμα μια απλή εφαρμογή που θα δείχνουμε τι χρησιμοποιούμε και για ποιό λόγο. Η απλή εφαρμογή που θέλουμε να κάνουμε θα έχει ως σκοπό να πατάμε το κουμπί και να αλλάζει αυτόματα το χρώμα του View.

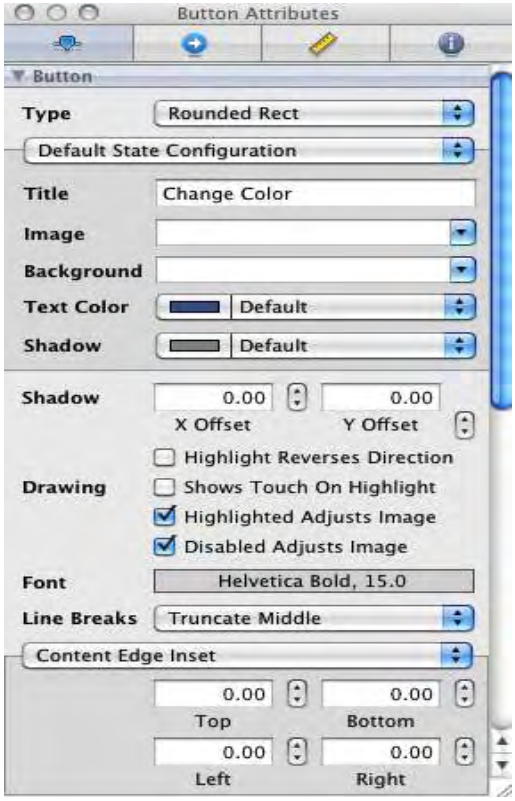
Αφού έχουμε δημιουργήσει ένα νέο project στο Xcode (με ονομασία ChangeColor) κάνουμε διπλό κλικ πάνω στο .xib και εμφανίζεται στην οθόνη μας το περιεχόμενο της εικόνας 3.3 και μπορούμε να δούμε ότι στην περιοχή *Navigation Area* και στο φάκελο Classes έχουμε 4 αρχεία το ChangeColorAppDelegate.h, το ChangeColorAppDelegate.m, το ChangeColorViewController.h και το ChangeColorViewController.m, ενώ στο φάκελο Resources μπορούμε να δούμε το GangeColorViewController.xib.



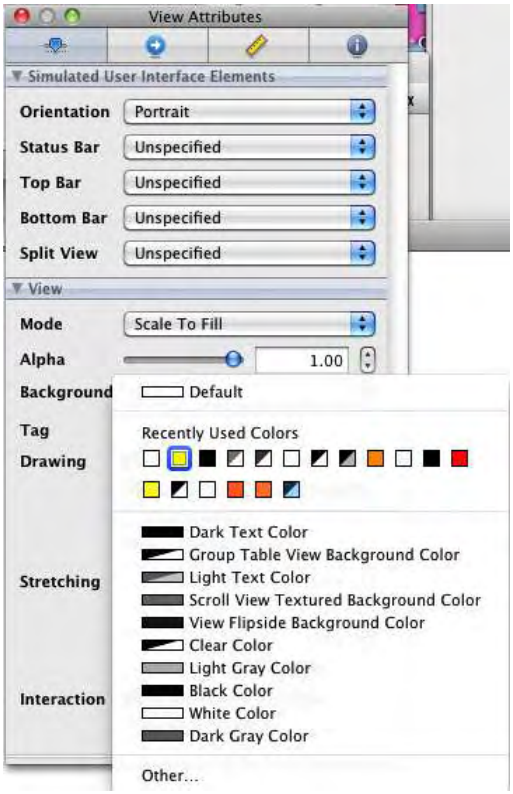
Εικόνα 3.3

Έπειτα επιλέγουμε από το Κύριο Μενού το File's Owner και πηγαίνουμε στις επιλογές του Inspector και διαλέγουμε την επιλογή identity η οποία βρίσκεται τελείως δεξιά, όπου και αλλάζουμε το περιεχόμενο του ορίσματος Class και το αντικαθιστούμε με το ChangeColorViewController και με αυτό τον τρόπο θα έχουμε σαν "υπεύθυνο" του File's Owner το ChangeColorViewController.

Μετά εισάγουμε τις κατάλληλες εντολές στο *Editor Area* του Xcode προκειμένου να έχουμε κάποια "ενέργεια" στο κουμπί του προγράμματος που θα δημιουργήσουμε. Αφού εισάγαμε τις εντολές πάμε στο Xcode και σέρνουμε ένα κουμπί από τη ***Library*** στο ***View*** μας. Πατώντας πάνω στο κουμπί πάμε στη συνέχεια στο Inspector και διαλέγουμε την επιλογή ***Attributes*** και δίνουμε ονομασία στο κουμπί αλλάζοντας το περιεχόμενο που βρίσκεται δεξιά του ***Title*** (όπως φαίνεται στην εικόνα 3.4). Επίσης, επιλέγοντας το ***View*** από το Κύριο Μενού πάμε πάλι στο ***Inspector*** και στην επιλογή ***Attributes*** και αλλάζουμε την επιλογή ***Background*** στο αρχικό χρώμα που επιθυμούμε να έχει το View (όπως φαίνεται στην εικόνα 3.5).



Εικόνα3.4



Εικόνα 3.5

Τέλος θα δούμε τις λίγες γραμμές εντολών που δώσαμε καθώς και τις συνδέσεις που κάνουμε με τα κουμπιά και το τελικό αποτέλεσμα. Στο φάκελο Classes και συγκεκριμένα στο αρχείο ChangeColorViewController.h βάζουμε τις εξής εντολές:

```
#import <UIKit/UIKit.h>
@interface ChangeColorViewController : UIViewController {

}
-(IBAction)buttonPressed:(UIButton *)sender;
@end
```

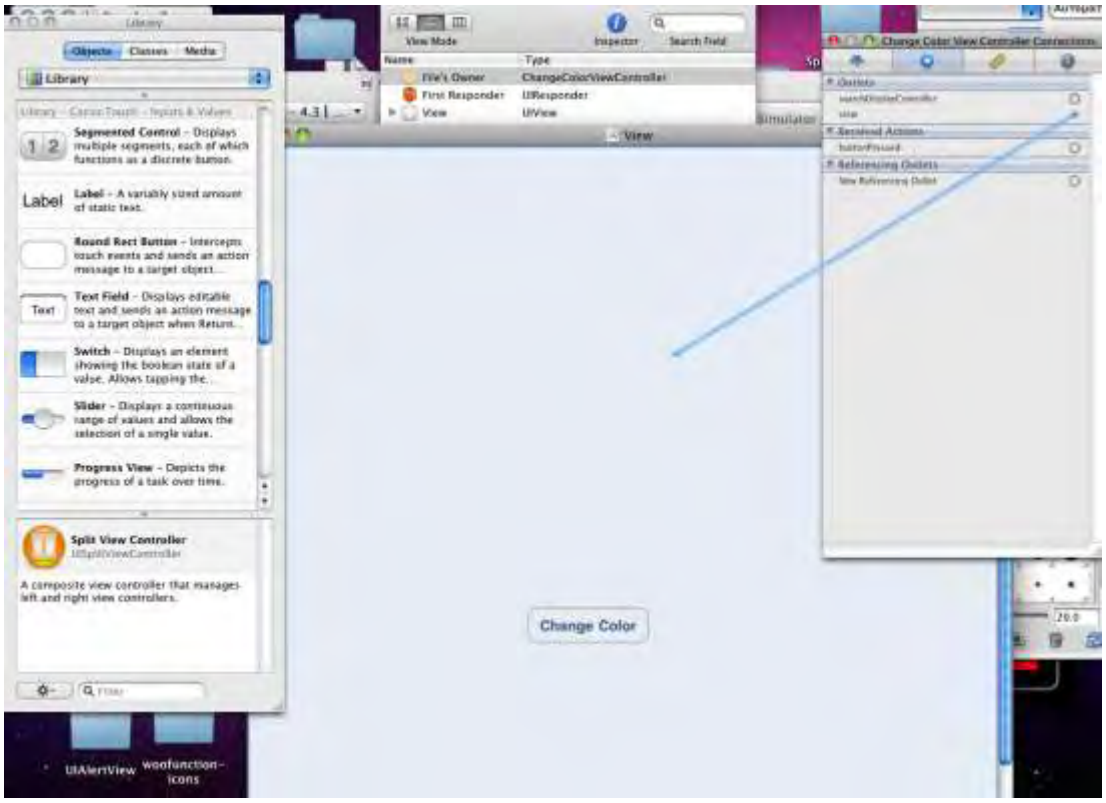
ενώ στο αρχείο ChangeColorViewController.m θα προσδιορίσουμε τη θα κάνει η ενέργεια που ορίσαμε προηγουμένως με τις παρακάτω εντολές:

```
-(IBAction)buttonPressed:(UIButton *)sender{
      self.view.backgroundColor = [UIColor redColor];
}
```

και όπως βλέπουμε η ενέργεια θα αλλάζει το χρώμα του View σε κόκκινο πατώντας το κουμπί.

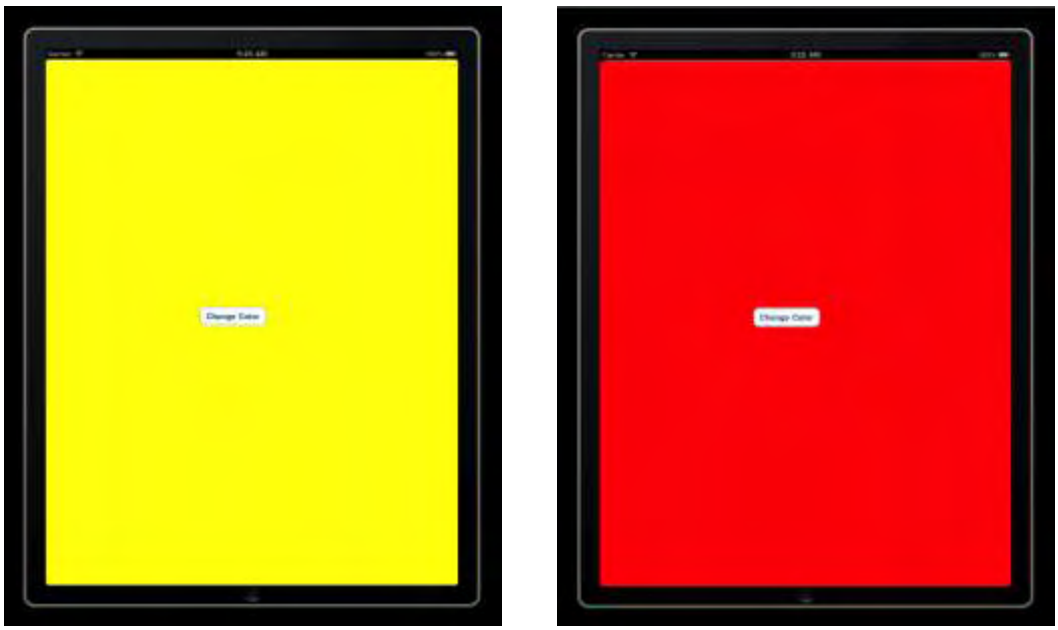
Επίσης, πρέπει να δημιουργήσουμε και τα κατάλληλα ***connections*** για το View και για το button. Έτσι επιλέγοντας το ***File's Owner*** και πηγαίνοντας στην επιλογή ***Connections*** βλέπουμε στα ***Outlets*** το ***view*** το οποίο και επιλέγουμε και

σέρνουμε προς το View του Interface Builder όπως φαίνεται στην εικόνα 3.6. Την ίδια ακριβώς διαδικασία κάνουμε και για να συνδέσουμε το κουμπί με το ***buttonPressed*** που βρίσκεται στο ***Received Actions***.



Εικόνα 3.6

Με αυτό τον τρόπο βλέπουμε στην εικόνα 3.7 την αρχική κατάσταση του View καθώς και την αλλαγή που παράγει το πάτημα του κουμπιού.

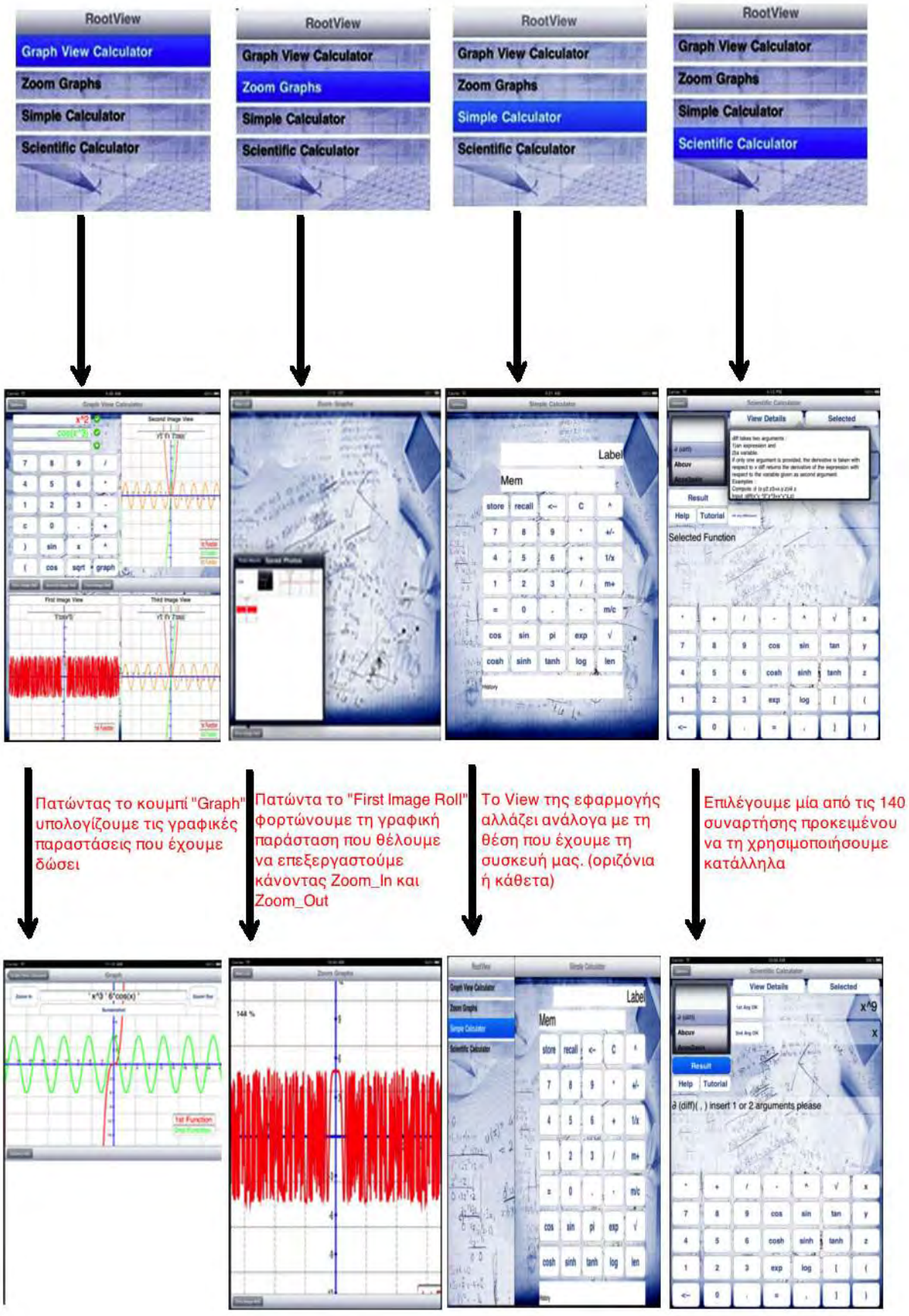


Εικόνα 3.7

Κάτι τελευταίο που θα πρέπει να αναφέρουμε είναι πως υπάρχει και η επιλογή ***Size*** στο ***Inspector*** όπου μπορούμε να τροποποιήσουμε το μέγεθος από οποιοδήποτε αντικείμενο επιθυμούμε.

4.Ανάλυση του κώδικα

4.1 Ξενάγηση στο interface της εφαρμογής

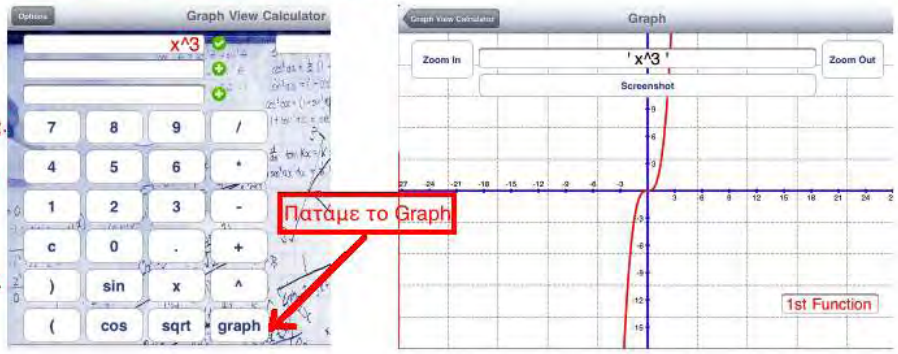


4.2Ανάλυση του κώδικα της εφαρμογής

4.2.1 FirstViewController

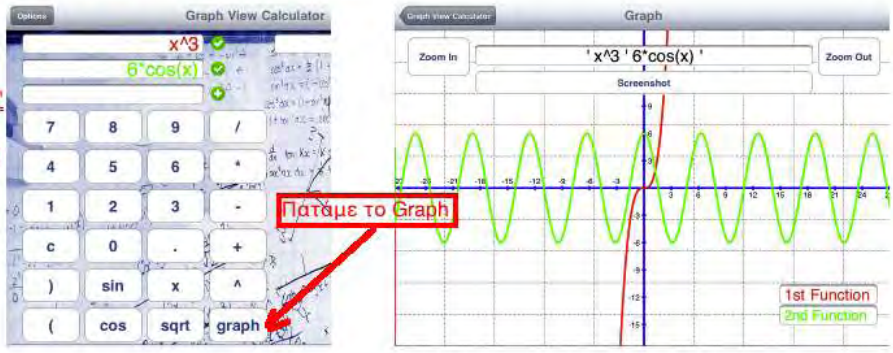
Γραφική παράσταση μιας συνάρτησης

Βήματα
1) Πληκτρολογούμε τη συνάρτηση με το πληκτρολόγιο της εφαρμογής.
2) Κατοχυρώνουμε τη γραφική συνάρτηση με το "+" (εισαγωγή των επόμενων συναρτήσεων πατώντας "+").
3) Πατάμε το κουμπί Graph για την εμφάνιση της/των συνάρτησης/εων.

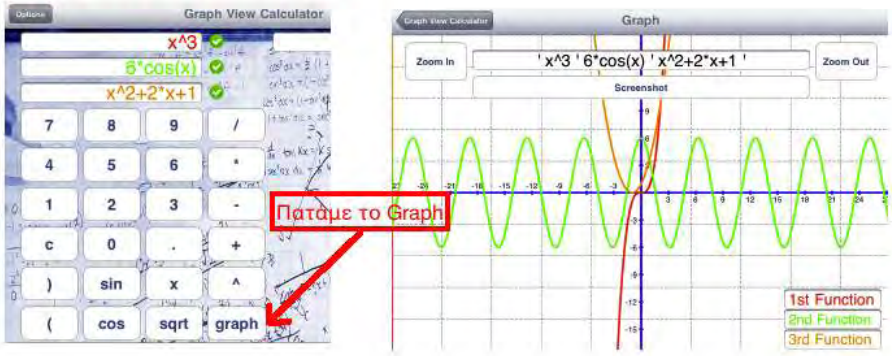


Γραφική παράσταση 2 συναρτήσεων

Ανάλυση κουμπιών του πεδίου "Graph"
1) Zoom-In και Zoom_Out για να μεγενθύνουμε και να σμικρύνουμε τη γραφική παράσταση.
2) Screenshot για αποθήκευση της γραφικής παράστασης.



Γραφική παράσταση 3 συναρτήσεων



FirstDetailViewController.h

```
#import <UIKit/UIKit.h>
#import "MultipleDetailViewsWithNavigatorAppDelegate.h"
//Κάνουμε #import τα παρακάτω 6 αρχεία (3 NewViewController και 3 GraphView) γιατί χρειάζονται για το σχεδιασμό των
```

```
//γραφικών παραστάσεων (ανάλογα όταν επιθυμούμε να
κάνουμε τον υπολογισμό των γραφικών παραστάσεων 1 ή 2
ή 3 συναρτήσεων)
#import "NewViewController.h"
#import "NewViewController2.h"
#import "NewViewController3.h"

#import "GraphView.h"
#import "GraphView2.h"
#import "GraphView3.h"
//Κάνουμε #import το CalculatorBrain αρχεί ο διότι με τη βοήθεια
του υπολογίζουμε τις τεταγμένες των σημείων της
//κάθε γραφικής παράστασης
#import "CalculatorBrain.h"

#import <MobileCoreServices/MobileCoreServices.h>

@interface FirstDetailViewController : UIViewController <UIImagePickerControllerDelegate,
UINavigationControllerDelegate, UIPopoverControllerDelegate, UISplitViewControllerDelegate> {

    UIPopoverController *popoverController; //popover για την επιλογή των
διάφορων επιλογών μας (στους Calculators)
     MultipleDetailViewsWithNavigatorAppDelegate *appDelegate;

     UITextField *display; //display του calculator graph για τις γραφικές
παραστάσεις που δίνει ο χρήστης
     UITextField *display2;
     UITextField *display3;
     CalculatorBrain *brain; //το χρησιμοποιούμε για να
υπολογίσουμε τις γραφικές παραστάσεις μας
     NSMutableArray *expression; //χρησιμοποιούνται προκειμένου να
μπορέσουμε να πάρουμε το string που
     //δίνει ο χρήστης μέσω των display και να το
μετατρέψουμε σε NSMutableArray για ευκολότερη χρήση
     NSMutableArray *expression2;
     NSMutableArray *expression3;
     BOOL decimalEntered; //για να έχουμε μία μόνο φορά
υποδιαστολή στους αριθμούς μας

     UIToolbar *toolbar; //το χρησιμοποιούμε για να
εμφανίζονται τα κουμπιά First Image Roll κλπ.
    UIPopoverController *popoverControllerAlbum; //popover που μέσω των
κουμπιών στο toolbar εμφανίζεται
     // προκειμένου να φανούν οι επιλογές για γραφικές
παραστάσεις που έχουμε που έχουμε
    UIImageView *imageViewFirst; //image για την 1η γραφική παράσταση
     UIImageView *imageViewSecond; //image για την 2η γραφική παράσταση
     UIImageView *imageViewThird; //image για την 3η γραφική παράσταση
    //BOOL newMedia;

     UITextField *firstTextField; //εκεί δίνουμε τη συνάρτηση για
την 1η γραφική παράσταση προς απεικόνιση
     UITextField *secondTextField; //εκεί δίνουμε τη συνάρτηση για
```

```
τη 2η γραφική παράσταση προς απεικόνιση
    UITextField *thirdTextField; //εκεί δίνουμε τη συνάρτηση για
την 3η γραφική παράσταση προς απεικόνιση

    UIButton *firstGraphOkButton; //είναι το κουμπί που
χρησιμοποιούμε για να κατοχυρώσουμε την 1η συνάρτηση
πως απεικόνιση
    UIButton *secondGraphOkButton; //είναι το κουμπί που
χρησιμοποιούμε για να κατοχυρώσουμε την 2η συνάρτηση
πως απεικόνιση
    UIButton *thirdGraphOkButton; //είναι το κουμπί που
χρησιμοποιούμε για να κατοχυρώσουμε την 3η συνάρτηση
πως απεικόνιση
}
@property (nonatomic, assign) MultipleDetailViewsWithNavigatorAppDelegate *appDelegate;

@property (retain,nonatomic) NSMutableArray *expression;
@property (retain,nonatomic) NSMutableArray *expression2;
@property (retain,nonatomic) NSMutableArray *expression3;
-(IBAction)digitPressed:(UIButton *)sender; //ενέργεια για τα κουμπιά των
ψηφίων, το "x" και την "."
-(IBAction)operationPressed:(UIButton *)sender; // ενέργεια για τα κουμπιά
όλων των πράξεων όπως επί σης και των παρενθέσεων
-(IBAction)graphPressed:(UIButton *)sender; //ενέργεια για το κουμπί graph,
με το οποίο εμφανίζεται (υπολογίζεται-σχεδιάζεται)
η/οι γραφική/ές παραστάσεις

@property (nonatomic, retain) IBOutlet UITextField *display;
@property (nonatomic, retain) IBOutlet UITextField *display2;
@property (nonatomic, retain) IBOutlet UITextField *display3;
@property (nonatomic, retain) IBOutlet UIImageView *imageViewFirst;
@property (nonatomic, retain) IBOutlet UIImageView *imageViewSecond;
@property (nonatomic, retain) IBOutlet UIImageView *imageViewThird;
@property (nonatomic, retain) UIPopoverController *popoverControllerAlbum;
@property (nonatomic, retain) IBOutlet UIToolbar *toolbar;
- (IBAction)useCameraRollFirst: (id)sender; //κουμπιά για να
εμφανίζονται οι γραφικές παραστάσεις που έχουμε
αποθηκεύσει και να επιλέγουμε ποιά θέλουμε να
εμφανίσουμε
- (IBAction)useCameraRollSecond: (id)sender;
- (IBAction)useCameraRollThird: (id)sender;

@property (nonatomic, retain) IBOutlet UITextField *firstTextField;
@property (nonatomic, retain) IBOutlet UITextField *secondTextField;
@property (nonatomic, retain) IBOutlet UITextField *thirdTextField;
- (IBAction)textFieldReturn:(id)sender; //ενέργεια η οποία δίνεται σε
κάθε TextField προκειμένου να απομακρύνεται το
πληκτρολόγιο του IPad όταν πατάμε το κουμπί return
- (IBAction)backgroundTouched:(id)sender; //ενέργεια η οποία δίνεται στο
background της εφαρμογής προκειμένου να απομακρύνεται το
πληκτρολόγιο του IPad οταν πατάμε στο background

-(IBAction)firstGraphOk:(id)sender; //ενέργεια για το κουμπί "+" που
```

```
βρίσκεται δεξιά από το πρώτο TextField για να
κατοχυρώσουμε την 1η συνάρτηση και εν συνεχεία
γίνεται "τικ"
-(IBAction)secondGraphOk:(id)sender; //ενέργεια για το κουμπί "+" που
βρίσκεται δεξιά από το δεύτερο TextField για να
κατοχυρώσουμε την 2η συνάρτηση και εν συνεχεία
γίνεται "τικ"
-(IBAction)thirdGraphOk:(id)sender; //ενέργεια για το κουμπί "+" που
βρίσκεται δεξιά από το τρίτο TextField για να
κατοχυρώσουμε την 3η συνάρτηση και εν συνεχεία
γίνεται "τικ"

@property (nonatomic, retain) IBOutlet UIButton *firstGraphOkButton;
@property (nonatomic, retain) IBOutlet UIButton *secondGraphOkButton;
@property (nonatomic, retain) IBOutlet UIButton *thirdGraphOkButton;

@end
```

FirstDetailViewController.m

```
@interface FirstDetailViewController ()
@property (nonatomic, retain) UIPopoverController *popoverController;
@end

@implementation FirstDetailViewController
//παρακάρωκάνουμε synthesize όσα έχουμε δηλώσει στο .h
αρχείο
@synthesize popoverController;
@synthesize appDelegate;

@synthesize display, display2, display3;
@synthesize expression, expression2, expression3;

@synthesize imageViewFirst, imageViewSecond, imageViewThird, popoverControllerAlbum, toolbar;
@synthesize firstTextField, secondTextField, thirdTextField;
@synthesize firstGraphOkButton, secondGraphOkButton, thirdGraphOkButton;

//boolean μεταβλητές οι οποίες χρησιμοποιούνται για την
εμφάνιση των γραφικών παραστάσεων που έχουμε
αποθηκεύσει
BOOL firstImage, secondImage, thirdImage;
//boolean μεταβλητές οι οποίες χρησιμοποιούνται για να
ξέρουμε πόσες γραφικές παραστάσεις θα σχεδιάσουμε
(έχουμε το δικαίωμα να δώσουμε από 1 μέχρι 3 γραφικές
συναρτήσεις για να σχεδιάσουμε)
BOOL firstGraph, secondGraph, thirdGraph;

//η παρακάτω χρησιμοποιείται αυτόματα κατα την
αρχικοποίηση του FirstViewController
-(id) init {
```

```
    if (self=[super init]) {
      self.appDelegate = (MultipleDetailViewsWithNavigatorAppDelegate *)[[UIApplication
sharedApplication] delegate];
    }
    return self;
}

// Η παρακάτω συνάρτηση χρησιμοποιείται για να
ορίσουμε το SplitViewController που χρησιμοποιούμε για να
διαλέξουμε τις επιλογές των διάφορων Calculators που
έχουμε
// Ορίζουμε το popover που χρησιμοποιούμε, τον τίτλο που
θα έχει καθώς και που θα βρίσκεται
- (void)splitViewController: (UISplitViewController*)svc
     willHideViewController:(UIViewController *)aViewController
          withBarButtonItem:(UIBarButtonItem*)barButtonItem
       forPopoverController: (UIPopoverController*)pc {

    barButtonItem.title = @"Options";
   [[self navigationItem] setLeftBarButtonItem:barButtonItem];
    [self setPopoverController:pc];
    self.appDelegate.rootPopoverButtonItem = barButtonItem;

}
// Η παρακάτω συνάρτηση χρησιμοποιείται για να
ορίσουμε το menu στα αριστερά της οθόνης, όταν έχουμε το
iPad σε οριζόντια θέση
- (void)splitViewController: (UISplitViewController*)svc
     willShowViewController:(UIViewController *)aViewController
  invalidatingBarButtonItem:(UIBarButtonItem *)barButtonItem {

    [[self navigationItem] setLeftBarButtonItem:nil];
    //Ορίζουμε το popover να μην εμφανιστεί
    [self setPopoverController:nil];
    //Ορίζουμε το root (που βρίσκεται αριστερά) με τι θα
ισούται και τι θα εμφανίζει
    self.appDelegate.rootPopoverButtonItem = barButtonItem;
}

//Παίρνει το display.text και από string που είναι το κάνει
NSMutableArray που είναι το expression
-(void)creatFirstExpression{
    NSString *firstHelp1;
    NSString *firstHelp2;
    NSString *firstHelp3;
    NSString *firstHelp4;
    int point1=0;
    int point2=0;
    BOOL thereIsANumber=NO;
    BOOL thereIsATrigo=NO;

    if (!expression) {
      expression = [[NSMutableArray alloc] init];
```

```
    }
    //Παίρνει το display.text για να το επεξεργαστεί
    NSString *first= display.text;
    int i;

    for (i=0; i<[first length]; i++) {

     //Με τις παρακάτω 2 εντολές διαλέγουμε "ένα-ένα"
στοιχείο για να το επεξεργαστούμε και να μπορέσουμε
να κάνουμε τις σωστές καταχωρήσεις στο NSMutableArray
     //Πχ αμαέχουμε cos(235*x) να κατυχωρήσουμε ολόκληρο
αριθμό (πχ 235) και όχι σπαστό(δηλαδή 23 και 5 χώρια) ή να
κατηχωρήσουμε ολόκληρο το cos και διαφορετικές
     //κατυχωρήσεις το * και το x.
     //Δηλαδή στο παράδειγμας θα είχαμε [0]:cos [1]:( [2]:235
[3]:* [4]:x [5]:)
     firstHelp1 = [first substringFromIndex:i];
     firstHelp2 = [firstHelp1 substringToIndex:1];

     //Όταν συναντήσουμε ένα από τα παρακάτω τότε
κοιτάμε αν πιο πρίν είχαμε αριθμό ή αν είχαμε καμιά
συνάρτηση
     if([firstHelp2 isEqual:@"/"]||
        [firstHelp2 isEqual:@"*"]||
        [firstHelp2 isEqual:@"-"]||
        [firstHelp2 isEqual:@"+"]||
        [firstHelp2 isEqual:@"{"]||
        [firstHelp2 isEqual:@"}"]||
        [firstHelp2 isEqual:@"("]||
        [firstHelp2 isEqual:@")"]||
        [firstHelp2 isEqual:@"^"]||
        [firstHelp2 isEqual:@","]||
        [firstHelp2 isEqual:@"x"]||
        [firstHelp2 isEqual:@"y"]||
        [firstHelp2 isEqual:@"z"]){

          //Αν υπήρχε πιο πριν αριθμός τον συλλέγουμε
και τον αποθηκεύουμε
          if (thereIsANumber==YES) {

               firstHelp3 = [first substringToIndex:i];
               firstHelp4 = [firstHelp3 substringFromIndex:point1];

               //Για τους αριθμούς πρέπει να τους
μετατρέψουμε απο String σε number για να τα
χρησιμοποιήσουμε εύκολα μετά στους υπολογισμούς
               NSNumberFormatter * f = [[NSNumberFormatter alloc] init];
               [f setNumberStyle:NSNumberFormatterDecimalStyle];
               NSNumber *number=[f numberFromString:firstHelp4];

               [expression addObject:number];

               thereIsANumber=NO;
```

```
                point2=0;
                [f release];

            }
            //Αν υπήρχε πιο πριν συνάρτηση την αποθηκεύουμε
            else if(therelsATrigo==YES){

                firstHelp3 = [first substringToIndex:i];
                firstHelp4 = [firstHelp3 substringFromIndex:point1];

                [expression addObject:firstHelp4];

                therelsATrigo=NO;
                point2=0;

            }


            [expression addObject:firstHelp2];

        }
        //Αν τελικά βρούμε αριθμό συνεχίζουμε το ψάξιμο και απλά δηλώνουμε πως έχουμε εντοπίσει την αρχή του αριθμού
        else if([firstHelp2 isEqual:@"1"]||
                [firstHelp2 isEqual:@"2"]||
                [firstHelp2 isEqual:@"3"]||
                [firstHelp2 isEqual:@"4"]||
                [firstHelp2 isEqual:@"5"]||
                [firstHelp2 isEqual:@"6"]||
                [firstHelp2 isEqual:@"7"]||
                [firstHelp2 isEqual:@"8"]||
                [firstHelp2 isEqual:@"9"]||
                [firstHelp2 isEqual:@"0"]||
                [firstHelp2 isEqual:@"."]){
            if (therelsANumber==NO) {
                therelsANumber=YES;
                point1 = i;
                point2++;
            }
            else {
                point2++;
            }
            //Είναι για την περίπτωση του τελευταίου αριθμού στη συνάρτηση που δεχόμαστε
            if((therelsANumber==YES)&&(i==[first length]-1)){

                firstHelp3 = [first substringToIndex:i+1];
                firstHelp4 = [firstHelp3 substringFromIndex:point1];

                NSNumberFormatter * f = [[NSNumberFormatter alloc] init];
                [f setNumberStyle:NSNumberFormatterDecimalStyle];
```

```
                NSNumber *number=[f numberFromString:firstHelp4];

                [expression addObject:number];

                thereIsANumber=NO;
                point2=0;
                [f release];
            }

        }
        else{

            if (thereIsATrigo==NO) {

                thereIsATrigo=YES;
                point1=i;
                point2++;
            }
            else {
                point2++;
            }


        }

    }

    //Απλά είναι για να ελέχγουμε αν είναι όντως αυτό που δώσαμε
    //και το εμφανίζουμε στο Terminal
    for (id obj in expression) {
        if ([obj isKindOfClass:[NSNumber class]]) {
            NSLog(@"ena");
            NSLog(@" %@", obj);
        }
        else if([obj isKindOfClass:[NSString class]]){
            NSLog(@"dyo");
            NSLog(@" %@", obj);
        }
    }


}

//Παίρνει το display2.text και από string που είναι το κάνει NSMutableArray που είναι το expression2
//Η συνάρτηση creatSecondExpression θα έχει ίδιο περιεχόμενο με τη creatFirstExpression απλά αφορά
//τη 2η γραφική παράσταση
-(void)creatSecondExpression{
    .......
}
```

```
//Παίρνει το display3.text και από string που είναι το κάνει
NSMutableArray που είναι το expression3
//Η συνάρτηση creatSecondExpression θα έχει ίδιο περιεχόμενο με
τη creatFirstExpression απλά αφορά
//την 3η γραφική παράσταση
-(void)creatThirdExpression{
    ........
}

//Δημιουργεί ένα GraphViewController και το χρησιμοποιεί για
το NavigationController
//Το expression που δημιουργήθηκε απο την προηγούμενη
συνάρτηση το περνάει στο savedExpression του GraphViewController για
να το επεξεργαστεί και να υπολογίσει/σχεδιάσει τη
συνάρτηση
//Ενεργοποιείται κάθε φορά που πατάμε το κουμπί Graph
- (void)showCoordinates {

    if ((firstGraph==1)&&(secondGraph==1)&&(thirdGraph==1)) {

     NSLog(@"First Expression");

     NewViewController *gvc = [[NewViewController alloc] init];

     gvc.title = @" ' ";
     gvc.title = [gvc.title stringByAppendingString:display.text];
     gvc.title = [gvc.title stringByAppendingString:@" ' "];

     gvc.title = [gvc.title stringByAppendingString:display2.text];
     gvc.title = [gvc.title stringByAppendingString:@" ' "];

     gvc.title = [gvc.title stringByAppendingString:display3.text];
     gvc.title = [gvc.title stringByAppendingString:@" ' "];

     gvc.savedExpression = expression;
     gvc.savedExpression2 = expression2;
     gvc.savedExpression3 = expression3;

     [self.navigationController pushViewController:gvc animated:YES];
     [gvc release];

    }
    else if ((firstGraph==1)&&(secondGraph==1)) {
     NSLog(@"Second Expression");

     NewViewController2 *gvc = [[NewViewController2 alloc] init];

     gvc.title = @" ' ";
     gvc.title = [gvc.title stringByAppendingString:display.text];
     gvc.title = [gvc.title stringByAppendingString:@" ' "];

     gvc.title = [gvc.title stringByAppendingString:display2.text];
     gvc.title = [gvc.title stringByAppendingString:@" ' "];
```

```
        gvc.savedExpression = expression;
        gvc.savedExpression2 = expression2;

        [self.navigationController pushViewController:gvc animated:YES];
        [gvc release];

    }
    else if (firstGraph==1){

        NewViewController3 *gvc = [[NewViewController3 alloc] init];

        gvc.title = @" ′ ";
        gvc.title = [gvc.title stringByAppendingString:display.text];
        gvc.title = [gvc.title stringByAppendingString:@" ′ "];

        gvc.savedExpression = expression;

        [self.navigationController pushViewController:gvc animated:YES];
        [gvc release];

    }
}

-(IBAction)digitPressed:(UIButton *)sender{
    NSString *digit = sender.titleLabel.text;

    //Παρακάτω φαίνεται πως σε κάθε συνάρτηση δίνουμε και διαφορετικό χρώμα στα γράμματα που γράφουμε όπως επίσης και στο γραμμή της αντίστοιχης γραφικής παράστασης
    display.textColor = [UIColor redColor];
    display2.textColor = [UIColor greenColor];
    display3.textColor = [UIColor orangeColor];


    if ( !(decimalEntered == 1) || !([@"." isEqualToString:digit])) {
     //Για να ξεχωρίζουμε σε ποιά σειρά θα πρέπει να γράψει τα διαφορετικα textField (display, display2, display3)
     if(firstGraph == 0){
            display.text = [display.text stringByAppendingString:digit];
     }
     else if (secondGraph  == 0) {
            display2.text = [display2.text stringByAppendingString:digit];
     }
     else if (thirdGraph  == 0) {
            display3.text = [display3.text stringByAppendingString:digit];
     }
    }


    if ([@"." isEqualToString:digit]) decimalEntered = YES;
```

```
        return;
}

-(IBAction)operationPressed:(UIButton *)sender{
        decimalEntered=0;
        display.textColor = [UIColor redColor];
        display2.textColor = [UIColor greenColor];
        display3.textColor = [UIColor orangeColor];

        NSString *digit = sender.titleLabel.text;

        //Απομακρύνουμε ό,τι έχει γραφτεί σε όλα τα tiextField
        if ([digit isEqualToString:@"c"]) {
          display.text = @"";
          [expression removeAllObjects];

          display2.text = @"";
          [expression2 removeAllObjects];

          display3.text = @"";
          [expression3 removeAllObjects];

          firstGraph=NO;
          secondGraph=NO;
          thirdGraph=NO;

          //Αλλάζουμε την εικόνα "τικ" σε "+" στο κουμπί που καταχωρούμε τις συναρτήσεις εφόσον έχουμε πατήσει το κουμπί  "c"
          UIImage *btnImage1 = [UIImage imageNamed:@"001_01.png"];
          [firstGraphOkButton setImage:btnImage1 forState:UIControlStateNormal];
          [secondGraphOkButton setImage:btnImage1 forState:UIControlStateNormal];
          [thirdGraphOkButton setImage:btnImage1 forState:UIControlStateNormal];
        }
        else {
          if(firstGraph == 0){
                display.text = [display.text stringByAppendingString:digit];
          }
          else if (secondGraph  == 0) {
                display2.text = [display2.text stringByAppendingString:digit];
          }
          else if (thirdGraph  == 0) {
                display3.text = [display3.text stringByAppendingString:digit];
          }

        }

        return;
}

//Καλεί  την showCordinates που έχουμε δημιουργήσει παραπάνω εφόσον υπάρχει κάτι στο textField
-(IBAction)graphPressed:(UIButton *)sender{
```

```
        display.textColor = [UIColor redColor];
        display2.textColor = [UIColor greenColor];
        display3.textColor = [UIColor orangeColor];

        //Είναι για να προειδοποιεί τον χρήστη σε
περίπτωση που πατήσει το "+" και δεν έχει δώσει καμία
συνάρτηση στο TextField
        if (firstGraph) {
         if ([display.text length] == 0) {
                UIAlertView *error = [[UIAlertView alloc] initWithTitle:@"Error"

        message:@"You are not introduced GRAPH! ¥n Please try again!"

delegate:self

cancelButtonTitle:@"Okay"

otherButtonTitles:nil];
                [error show];
                [error release];
         }
         else {
                [self creatFirstExpression];
         }
        }
        if (secondGraph){
         if ([display2.text length] == 0) {
                UIAlertView *error = [[UIAlertView alloc] initWithTitle:@"Error"

        message:@"You are not introduced GRAPH! ¥n Please try again!"

delegate:self

cancelButtonTitle:@"Okay"

otherButtonTitles:nil];
                [error show];
                [error release];
         }
         else {
                [self creatSecondExpression];
         }
        }
        if (thirdGraph){
         if ([display3.text length] == 0) {
                UIAlertView *error = [[UIAlertView alloc] initWithTitle:@"Error"

        message:@"You are not introduced GRAPH! ¥n Please try again!"

delegate:self

cancelButtonTitle:@"Okay"
```

```
otherButtonTitles:nil];
                [error show];
                [error release];
        }
        else {
                [self creatThirdExpression];
        }
      }

      if (([display.text length] > 0)||(([display.text length] > 0)&&([display2.text length] >
0))||(([display.text length] > 0)&&([display2.text length] > 0)&&([display3.text length] > 0)))
{

        [self showCoordinates];

      }

      return;
}

- (IBAction) useCameraRollFirst: (id)sender{

      firstImage=TRUE;
      //Αν πατήσουμε πάνω στο κουμπί να απομακρύνει το
popover
    if ([self.popoverControllerAlbum isPopoverVisible]) {
        [self.popoverControllerAlbum dismissPopoverAnimated:YES];
        [popoverControllerAlbum release];
    }
      //Διαφορετικά αν πατήσουμε πάνω στο κουμπί να μας
εμφανίσει το popover
      else {
        if ([UIImagePickerController isSourceTypeAvailable:
             UIImagePickerControllerSourceTypeSavedPhotosAlbum])
        {
             //Παίρνουμε τη φωτογραφία που διαλέξαμε και
την επιστρέφουμε, δηλαδή την εμφανίζουμε
            UIImagePickerController *imagePicker =[[UIImagePickerController alloc] init];
            imagePicker.delegate = self;
            imagePicker.sourceType =UIImagePickerControllerSourceTypePhotoLibrary;
            //imagePicker.mediaTypes = [NSArray arrayWithObjects:(NSString *) kUTTypeImage,nil];
            imagePicker.allowsEditing = NO;

            self.popoverControllerAlbum = [[UIPopoverController
alloc]initWithContentViewController:imagePicker];
            popoverControllerAlbum.delegate = self;

            [self.popoverControllerAlbum presentPopoverFromBarButtonItem:sender
permittedArrowDirections:UIPopoverArrowDirectionUp animated:YES];

            [imagePicker release];
        }
```

```
    }
}

Σημείωση: Υπάρχουν και οι συναρτήσεις useCameraRollSecond και
useCameraRollThird οι οποίες δεν αναφέρονται γιατί είναι
όμοιες με την useCameraRollFirst, αλλά αφορούν τις ενέργειες
για τα άλλα δύο κουμπιά.

-(IBAction)backgroundTouched:(id)sender{
    //Όπως βλέπουμε περιλαμβάνει όλα όσα προκαλούν την
εμφάνιση του πληκτρολογίου
    //και αν πατήσουμε στο Background εξαφανίζεται το
πληκτρολόγιο
    [display resignFirstResponder];
    [firstTextField resignFirstResponder];
    [display2 resignFirstResponder];
    [secondTextField resignFirstResponder];
    [display3 resignFirstResponder];
    [thirdTextField resignFirstResponder];
}

-(IBAction)firstGraphOk:(id)sender{
    //Με την παρακάτω δήλωση δείχνουμε ότι έχουμε
κατοχυρώσει την πρώτη γραφική παράσταση
    firstGraph = YES;

    NSLog(@"first %d", firstGraph);
    NSLog(@"second %d", secondGraph);
    NSLog(@"third %d", thirdGraph);

    //Αλλάζουμε την εικόνα του κουμπιού "+" και γίνεται
"κλικ"
    UIImage *btnImage1 = [UIImage imageNamed:@"001_06.png"];
   [sender setImage:btnImage1 forState:UIControlStateNormal];
}

Σημείωστη: Υπάρχουν και η secondGraphOk και η thirdGraphOk οι
οποίες δεν αναφέρονται γιατί είναι παρόμοιες με την
firstGraphOk.

//Η συνάρτηση που ακολουθεί χρησιμοποιείται για να
φορτώσει την εικόνα που επιλέγουμε απο τα popovers
-(void)imagePickerController:(UIImagePickerController *)picker
didFinishPickingMediaWithInfo:(NSDictionary *)info
{
    [self.popoverControllerAlbum dismissPopoverAnimated:true];
    [popoverControllerAlbum release];

    NSString *mediaType = [info objectForKey:UIImagePickerControllerMediaType];
    [self dismissModalViewControllerAnimated:YES];
    if ([mediaType isEqualToString:(NSString *)kUTTypeImage]) {
        UIImage *image = [info objectForKey:UIImagePickerControllerOriginalImage];
       if (firstImage) {
```

```
            imageViewFirst.image = image;
            firstImage=FALSE;
        }
        else if(secondImage){
            imageViewSecond.image = image;
            secondImage=FALSE;
        }
        else if(thirdImage){
            imageViewThird.image = image;
            thirdImage=FALSE;
        }
    }
}
//Μας ειδοποιεί σε περίπτωση εσφαλμένης φόρτωσης οποιαδήποτε εικόνας
-(void)image:(UIImage *)image
finishedSavingWithError:(NSError *)error
 contextInfo:(void *)contextInfo
{
    if (error) {
        UIAlertView *alert = [[UIAlertView alloc]
                              initWithTitle: @"Save failed"
                              message: @"Failed to save image"¥
                              delegate: nil
                              cancelButtonTitle:@"OK"
                              otherButtonTitles:nil];
        [alert show];
        [alert release];
    }
}

//Είναι για τις περιπτώσεις όπου γυρίζουμε το iPad και διαλέγουμε τι θα συμβεί με το view
//Στην παρακάτω περίπτωση ελέγχουμε αν θα εμφανιστεί ή όχι το menu στα αριστερά όταν γυρίζουμε οριζόντια τη συσκευή μας.
- (BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)interfaceOrientation {

    if (interfaceOrientation == UIInterfaceOrientationLandscapeLeft || interfaceOrientation ==
UIInterfaceOrientationLandscapeRight) {
        [[self navigationItem] setLeftBarButtonItem:nil];
    }
    else {
        [[self navigationItem] setLeftBarButtonItem:self.appDelegate.rootPopoverButtonItem];
    }
    return YES;

}

//Καλείται όταν ξεκινάει να φαίνεται το View
//και όπως φαίνεται αρχικοποιεί έναν αντίγραφο τη CalculatroBrain που χρησιμοποιούμε για να κάνουμε τους υπολογισμούς των συντεταγμένων των
```

```
//γραφικών παραστάσεων, δίνει ονόματα στα κουμπιά του
toolBar αλλά και δίνει συγκεκριμένη ενέργεια στο καθένα
- (void)viewDidLoad {

    [super viewDidLoad];
     brain = [[CalculatorBrain alloc] init];

     UIBarButtonItem *cameraRollFirst = [[UIBarButtonItem alloc]
                                                    initWithTitle:@"First Image Roll"
                                                    style:UIBarButtonItemStyleBordered
                                                    target:self

action:@selector(useCameraRollFirst:)];
    UIBarButtonItem *cameraRollSecond = [[UIBarButtonItem alloc]
                                                        initWithTitle:@"Second Image
Roll"

style:UIBarButtonItemStyleBordered
                                                        target:self

action:@selector(useCameraRollSecond:)];
     UIBarButtonItem *cameraRollThird = [[UIBarButtonItem alloc]
                                                        initWithTitle:@"Third Image
Roll"

style:UIBarButtonItemStyleBordered
                                                        target:self

action:@selector(useCameraRollThird:)];
    NSArray *items = [NSArray arrayWithObjects: cameraRollFirst, cameraRollSecond,
cameraRollThird, nil];
    [toolbar setItems:items animated:NO];
    [cameraRollFirst release];
    [cameraRollSecond release];
     [cameraRollThird release];

     return;
}

//Καλείται αποδεσμεύεται το View και θέλουμε να
αποδεσμεύσουμε κάποια στοιχεία για να μην υπάρχουν
στη μνήμη
- (void)viewDidUnload {

     self.popoverController = nil;
     self.display =nil;
     self.display2 =nil;
     self.display3 =nil;

     self.imageViewFirst = nil;
    self.imageViewSecond= nil;
     self.imageViewThird = nil;
     self.popoverControllerAlbum = nil;
```

```
    self.toolbar = nil;

      self.expression = nil;
      self.expression2 = nil;
      self.expression3 = nil;

      self.firstTextField = nil;
      self.secondTextField = nil;
      self.thirdTextField = nil;

      self.firstGraphOkButton = nil;
      self.secondGraphOkButton = nil;
      self.thirdGraphOkButton = nil;

      self.popoverControllerAlbum = nil;
      self.appDelegate = nil;

      return;
}
- (void)dealloc {
    [popoverController release];
      [display release];
      [display2 release];
      [display3 release];

      [toolbar release];
    [popoverControllerAlbum release];
    [imageViewFirst release];
      [imageViewSecond release];
      [imageViewThird release];

      [brain release];

      [firstTextField release];
      [secondTextField release];
      [thirdTextField release];

      [expression release];
      [expression2 release];
      [expression3 release];

      [popoverControllerAlbum release];
      [appDelegate release];

      [firstGraphOkButton release];
      [secondGraphOkButton release];
      [thirdGraphOkButton release];

    [super dealloc];
}

@end
```

Σημείωση: Τα παραπάνω αρχεία FirstDetailViewController.h και FirstDetailViewController.m περιέχουν ολόκληρο τον κώδικά τους προκείμενου να δείξουμε πως ορίζεται και η παραμικρή λεπτομέρεια. Στα αρχεία που ακολουθούν δε θα εστιάζουμε σε λεπτομέρειες, αντίθετα θα δείχνουμε τα νέα στοιχεία που εισάγουμε.

4.2.1.1 GraphView

GraphView.h

```
@class GraphView;

//Αποτελείέναprotocol το οποίο περιέχει συναρτήσεις
και χρησιμοποιείται από το GraphView
@protocol GraphViewDelegate
//Οι παρακάτω yDataToGraph βρίσκονται στους GraphViewController
//και με τη σειρά τους καλούν την evaluateExpression της
CalculatorBrain
//η οποία υπολογίζει και επιστρέφει το "y" γιαένα
σύνολο από "x"
-(float)yDataToFirstGraph:(GraphView *)requestor withThisX:(float)xValue;
-(float)yDataToSecondGraph:(GraphView *)requestor withThisX:(float)xValue;
-(float)yDataToThirdGraph:(GraphView *)requestor withThisX:(float)xValue;
@end


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

}

@property(assign) id<GraphViewDelegate> delegate;

@property float newScale;

@end
```

GraphView.m

```
//Με την drawRect προσπαθούμε να σχεδιάσουμε τη γραφική
μας παράσταση
- (void)drawRect:(CGRect)rect
{

      NSLog(@"%f",newScale);

      //ΔηλώνουμεένασημείοCGPoint midPoint;
      //Με τις παρακάτω 2 εντολές βρίσκουμε το σημείο (0,0)
του καρτεσιανού επιπέδου που θα σχεδιάσουμε τις
```

```
γραφικές μας παραστάσεις
    //Όπως βλέπουμε για οποιοδήποτε (πλάτους και
μήκους) GraphView εμείς προσαρμόζουμε το σημείο ανάλογα
το width και το height
    midPoint.x = self.bounds.origin.x + self.bounds.size.width/2;
    midPoint.y = self.bounds.origin.y + self.bounds.size.height/2;

    NSLog(@"origin x = %G",self.bounds.origin.x); //x=0
    NSLog(@"origin y = %G",self.bounds.origin.y); //y=0
    NSLog(@"height = %G",self.bounds.size.height); // height= όσος είναι ο χώρος
που πιάνει το GraphView
    NSLog(@"WIDTH = %G",self.bounds.size.width); //width= όσος είναι ο χώρος
που πιάνει το GraphView

    CGPoint axisOrigin;
    axisOrigin.x = midPoint.x;
    axisOrigin.y = midPoint.y;


    //Το CGRect είναι ένα ζεύγος συντεταγμένων, το ύψος
και το πλάτος ενός σχήματος
    CGRect bounds = self.bounds;
    NSLog(@"rect bounds: %@", NSStringFromCGRect(bounds));
    NSLog(@"rect self.bounds: %@", NSStringFromCGRect(self.bounds));
    //Με την παρακάτω εντολή δηλώνουμε μια γραμμή με
την οποία θα σχεδιάσουμε τη γραφικά μας παράσταση
    CGContextRef context = UIGraphicsGetCurrentContext();

    //Δηλώνουμε χρώμα μπλέ για το καρτεσιανό επίπεδο
    [[UIColor blueColor] setStroke];

    //Καλούμε την παρακάτω συνάρτηση που βρίσκεται
στο AxesDrawer
    //+ (void)drawAxesInRect:(CGRect)bounds originAtPoint:(CGPoint)axisOrigin
scale:(CGFloat)pointsPerUnit
    //η οποία δημιουργεί τους άξονες του καρτεσιανού
επιπέδου
    [AxesDrawer drawAxesInRect:bounds //draw axis of graph
                originAtPoint:axisOrigin
                                scale:newScale];

    int x;
    int xRange;


    //Για το πλατος της γραμμής της γραφικής μας
παράστασης
    CGContextSetLineWidth(context, 3.0);

    //Η γραμμή ορίζεται να ξεκινήσει σε αυτό το σημείο
    CGContextMoveToPoint(context, 0, 181.0);

    //Ορίζουμε τη xRangeίση με το πλάτος του GraphView διότι
```

```
όπως εί παμε παραπά νω υπολογί ζουμε το y για έ να σύνολο
από x
    //έ τσι ορί ζουμε το x από το "-width/2" μέ χρι το "width/2"
    xRange = self.bounds.size.width;

    //Δηλώνουμε με κόκκινο τη γραμμή για την 1η γραφική
μας παρά σταση
    [[UIColor redColor] setStroke];
    //Έχουμε μια λούπα που εί ναι ί σες οι φορέ ς της
επανά ληψης με το πλά τος του GraphView
    //Σημειωτέ ον ότι πρέ πει να κά νουμε και χρή ση του
Scale που χρησιμοποιεί ται και στην κατασκευή του
καρτεσιανού επιπέ δου και σε αυτή του zoom που θα δούμε
αργότερα
    //Έπί σης, πρέ πει να επισημαί νουμε ότι το GraphView
αριθμεί τε με pixels πά νω στην οθόνη ,δλδ πά νω στο GraphView. Η
αρί θμηση τους όμως δε γί νεται σύμφωνα με το
καρτεσιανό επί πεδο που
    //γνωρί ζουμε (δηλαδή πά νω το κέ ντρο της οθόνης να
εί ναι το σημεί ο (0,0)), αλλά αντί θετα το σημεί ο (0,0)
αντιστοιχεί στο σημεί ο που βρί σκεται στη γωνί α πά νω-
αριστερά , και έ χουμε μόνο
    //θετική αρί θμηση. Τα σημεί α έ χουν συντεταγμέ νες
(x,y), όπου το x βρί σκεται οριζόντια, ενώ το y βρί σκεται
κά θετα.
    //Όπως αναφέ ραμε και πρί ν πως υπολογί ζουμε τα y για
έ να σύνολο x και επειδή η αρί θμηση (των pixels) έ χει μόνο
θετικούς αριθμούς ξεκινά με από το x=0 μέ χρι την το width
του GraphView
    //Εν συνεχεί α βρί σκουμε το x που αντιστοιχεί στο
καρτεσιανό επί πεδο (xVar) και μετά το xPlot προσαρμόζεται
σύμφωνα με το Scale για να υπολογί σουμε την τιμή yValye
βά ση του xPlot.
    //Ακόμη, υπολογί ζουμε το yPlot σύμφωνα με το yValue και
το Scale. Ουσιαστικά το xVar και το yPlot εί ναι οι
συντεταγμέ νες των pixels, αν η απεικόνιση του GraphView ή ταν
όπως έ να καρτεσιανό επί πεδο
    //Τέ λος, ενώνουμε τα σημεί α που βρή καμε για να
δημιουργηθεί σταδιακά η γραφική μας παρά σταση
    for (x = 0 ; x <= xRange; x++)
    {
     float xVar = x - xRange/2;
     float xPlot = xVar/newScale;
     float yValue = [self.delegate yDataToFirstGraph:self withThisX:xPlot];
     float yPlot = yValue * newScale;
     //NSLog(@"x pixels = %g x units = %g y units = %g y pixels = %g",xVar,xPlot, yValue,
yPlot);
     CGContextAddLineToPoint(context, x, (self.bounds.size.height/2 - yPlot));
    }
    CGContextStrokePath(context);
```

Σημεί ωση: Υπά ρχει κώδικας και για τις υπόλοιπες 2
συναρτή σεις, αλλά επειδή εί ναι όμοιος δεν

```
επαναλαμβάνεται.

.........
}
```

4.2.1.2 AxesDrawer

AxesDrawer.h

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

//Έτσι σχεδιάζουμε τους άξονες του καρτεσιανού
επιπέδου
//Για την κατασκευή του καρτεσιανού επιπέδου
χρειαστή καμε τη σημαντική βοήθεια του αρχείου AxesDrawer
,όπου το σκεπτικό μας βασίστηκε σε αυτό
@interface AxesDrawer : NSObject

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

@end
```

AxesDrawer.m

```
#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

//Ορίζουμε το Font της γραμματοσειράς που θα
χρησιμοποιηθεί για την αρίθμηση των αξόνων
#define HASH_MARK_FONT_SIZE 12.0

#define HORIZONTAL_TEXT_MARGIN 6
#define VERTICAL_TEXT_MARGIN 3

//Χρησιμοποιείται από την drawHashMarksInRect
+ (void)drawString:(NSString *)text atPoint:(CGPoint)location withAnchor:(int)anchor
{
      if ([text length])
      {
       //Καθορίζουμε το font της γραμματοσειράς
       UIFont *font = [UIFont systemFontOfSize:HASH_MARK_FONT_SIZE];

       //Τοποθετούμε πάνω στους άξονες τους αριθμούς
       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;

        //Ανάλογα εκεί που βάζουμε τους αριθμούς τους ανεβάζουμε/κατεβάζουμε ή τους πάμε πιο αριστερά/δεξιά
        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

//Χρησιμοποιείται από την drawAxesInRect προκειμένου να δημιουργήσει και να απεικονίσει την αρίθμηση των αξόνων
+ (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;
      NSString *neg=@"-";

      //Μέσα στην παρακάτω λούπα καλείται η drawString
      for (int offset = unitsPerHashmark; !started || stillGoing; offset += unitsPerHashmark)
      {
        NSString *positiveLabel = nil;
        NSString *negativeLabel = nil;
        BOOL drew = NO;
        CGFloat scaledOffset = floor(offset * pointsPerUnit);
        CGPoint hashMarkPoint;

        //Για να εμφανίζονται οι αριθμοί στο θετικό τμήμα του άξονα των x
        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;
        }

        //Για να εμφανίζονται οι αριθμοί στο αρνητικό τμήμα του άξονα των x
        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)];
            if (!negativeLabel) negativeLabel = [NSString stringWithString:neg];
negativeLabel=[negativeLabel stringByAppendingFormat:@"%d",offset];
            [self drawString:negativeLabel atPoint:hashMarkPoint withAnchor:ANCHOR_BOTTOM];
            drew = YES;
      }

      //για να εμφανίζονται οι αριθμοί στο θετικό τμήμα του άξονα των y
      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;
      }

      //Για να εμφανίζονται οι αριθμοί στο αρνητικό
τμήμα του άξονα των y
      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)];
                  negativeLabel = [NSString stringWithString:neg];
                  negativeLabel=[negativeLabel stringByAppendingFormat:@"%d",offset];
            }
            [self drawString:negativeLabel atPoint:hashMarkPoint withAnchor:ANCHOR_RIGHT];
            drew = YES;
      }

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

    CGContextStrokePath(context);
}

//Χρησιμοποιείται από το GraphView
//Δημιουργείτους άξονες του καρτεσιανού επιπέδου
+ (void)drawAxesInRect:(CGRect)bounds originAtPoint:(CGPoint)axisOrigin
```

```
scale:(CGFloat)pointsPerUnit
{
      CGContextRef context = UIGraphicsGetCurrentContext();

      UIGraphicsPushContext(context);

      int i;
      float distanceWidth, distanceHeight;

      //Παρακάτω δημιουργούμε τις διακεκομμένες
γραμμές που έχουμε στο καρτεσιανό επίπεδο και οι
οποίες είναι γκρι
      [[UIColor grayColor] setStroke];
      CGContextSetLineWidth(context, 1.0);

      CGFloat dash1[] = {5.0, 2.0};
      CGContextSetLineDash(context, 0.0, dash1, 2);

      //Η παρακάτω λούπα είναι για να δημιουργήσουμε τις
κάθετες γκρι γραμμές
      distanceWidth = bounds.size.width/10;
      for (i=0; i<10; i++) {

        CGContextBeginPath(context);
        CGContextMoveToPoint(context, axisOrigin.x-i*distanceWidth, bounds.origin.y);
        CGContextAddLineToPoint(context, axisOrigin.x-i*distanceWidth,
bounds.origin.y+bounds.size.height);
        CGContextStrokePath(context);
        if ((axisOrigin.x-i*distanceWidth)<0) {
              //NSLog(@"arnitika");
        }
        else {
              CGContextBeginPath(context);
              CGContextMoveToPoint(context, axisOrigin.x+i*distanceWidth, bounds.origin.y);
              CGContextAddLineToPoint(context, axisOrigin.x+i*distanceWidth,
bounds.origin.y+bounds.size.height);
              CGContextStrokePath(context);

        }

      }

      //Η παρακάτω λούπα είναι για να δημιουργήσουμε τις
οριζόντιες διακεκομμένες γκρί γραμμές
      distanceHeight = bounds.size.height/10;
      for (i=0; i<10; i++) {

        CGContextBeginPath(context);
        CGContextMoveToPoint(context, bounds.origin.x, axisOrigin.y-i*distanceHeight);
        CGContextAddLineToPoint(context, bounds.origin.x+bounds.size.width, axisOrigin.y-
i*distanceHeight);

        CGContextStrokePath(context);
```

```
        if ((axisOrigin.y-i*distanceHeight)<0) {
                //NSLog(@"arnitika");

        }
        else {
                CGContextBeginPath(context);
                CGContextMoveToPoint(context, bounds.origin.x, axisOrigin.y+i*distanceHeight);
                CGContextAddLineToPoint(context, bounds.origin.x+bounds.size.width,
axisOrigin.y+i*distanceHeight);

                CGContextStrokePath(context);

        }


      }

      [[UIColor blueColor] setStroke];
      CGContextSetLineWidth(context, 3.0);

      //Το παρακάτω το κάνουμε για να μην έχουμε
διακεκομμένες γραμμές για να κάνουμε τους κύριους
άξονες του καρτεσιανού επιπέδου
      dash1[0] = 0.0;
      dash1[1] = 0.0;
      CGContextSetLineDash(context, 0.0, dash1, 0);

      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);

      //Καλούμε την παρακάτω συνάρτηση
      //+ (void)drawHashMarksInRect:(CGRect)bounds originAtPoint:(CGPoint)axisOrigin
scale:(CGFloat)pointsPerUnit
      //η οποία παράγει την αρίθμηση των αξόνων
      [self drawHashMarksInRect:bounds originAtPoint:axisOrigin scale:pointsPerUnit];

      UIGraphicsPopContext();
}

@end
```

4.2.1.3 CalculatorBrain

CalculatorBrain.h

```
@interface CalculatorBrain : NSObject
{
      //NSMutableArray *internalExpression;
```

```
}
//@property(nonatomic,retain) NSMutableArray *internalExpression;

//-(id)init;

+(float)evaluateExpression:(NSMutableArray *)anExpression
       usingVariableValue:(float)variables;
@end

CalculatorBrain.mm

//Υπολογίζουμε την παράσταση που του έχουμε δώσει
(συναρτήση του x) και επιστρέφει το αποτέλεσμα
+(float)evaluateExpression:(NSMutableArray *)anExpression
       usingVariableValue:(float)xValue {

     //Αφού έχουμε "τεμαχίσει" το NSString, που έχουμε πάρει
από τα Text, καλούμε την παρούσα συνάρτηση
     NSMutableArray *expr = [[NSMutableArray alloc] initWithArray:anExpression copyItems:YES];

     //NSLog(@"%@", anExpression);

     float operand1,operand2;
     int pos1;
     int pos2;
     int pos3;
     NSNumber *numberOperand;

     //Παρακάτω ελέγχουμε τα στοιχεία που βρίσκονται
μέσα στο NSMutableArray και ανάλογα τα αφαιρούμε (και τις
παρενθέσεις επίσης) και καλούμε αναδρομικά τη
συνάρτηση μέχρι να φτάσουμε
     //σε ένα τελικό αποτέλεσμα
     //Παρακάτω ελέγχουμε άμα συναντήσουμε ένα
συνημίτονο, το αφαιρούμε και εν συνεχεία αφαιρούμε
και τις παρενθέσεις και καλούμε αναδρομικά τη
συνάρτηση με όρισμα το περιεχόμενο του συνημιτόνου
     if ([expr indexOfObject:@"cos"]!=NSIntegerMax) {

       int countOfLeftParenthesis=0;
       int countOfRightParenthesis=0;
       int lastParenthesis=0;

       NSMutableArray *expres = [[NSMutableArray alloc] initWithArray:anExpression
copyItems:YES];

       while ([expres indexOfObject:@"("]!=NSIntegerMax) {
              [expres removeObjectAtIndex:[expres indexOfObject:@"("]];
              countOfLeftParenthesis++;
       }

       while ([expres indexOfObject:@")"]!=NSIntegerMax) {
```

```
            [expres removeObjectAtIndex:[expres indexOfObject:@")"]];
            countOfRightParenthesis++;
        }

        lastParenthesis = [expres count]-1+countOfLeftParenthesis+countOfRightParenthesis;


        pos1=[expr indexOfObject:@"cos"];
        pos2=[expr indexOfObject:@"("];
        pos3=lastParenthesis;


        [expr removeObjectAtIndex:pos3];
        [expr removeObjectAtIndex:pos1];
        [expr removeObjectAtIndex:pos1];

        NSMutableArray *exprHelp = [[NSMutableArray alloc] init];
        int i, k;
        float res;

        k=pos3-pos2-1;

        for (i=0; i<k; i++) {
            [exprHelp addObject:[expr objectAtIndex:pos1]];
            [expr removeObjectAtIndex:pos1];
        }


        res = [self evaluateExpression:exprHelp usingVariableValue:xValue];
        res = cos(res);
        numberOperand = [NSNumber numberWithDouble:res];
        [expr insertObject:numberOperand atIndex:pos1];


        operand1=res;

        [expres release];
        [exprHelp release];
    }
    Σημείωση: Υπάρχει κώδικας και για το cos και το sqrt,
αλλά επειδή είναι όμοιος με το παραπάνω δεν
επαναλαμβάνεται.

.....

    //Η παρακάτω λούπα είναι για τις πράξεις της
διαίρεσης, του πολλαπλασιασμού, ύψωση σε δύναμε, της
αφαίρεσης και της πρόσθεσης
    //Προσέχουμε να έχουμε τη σωστή προτεραιότητα
πράξεων
    while (1) {
     //NSLog(@"mpla a");
```

```
        pos1=[expr indexOfObject:@"*"];
        pos2=[expr indexOfObject:@"/"];
        pos3=[expr indexOfObject:@"^"];

        // *
        if(pos1!=NSIntegerMax){

                if ((pos1<pos2)&&(pos1<pos3)) {

                        if ([[expr objectAtIndex:pos1-1] isEqual:@"x"]) {
                                operand1=xValue;

                        }
                        else {
                                operand1=[[expr objectAtIndex:pos1-1] floatValue];
                        }
                        if ([[expr objectAtIndex:pos1+1] isEqual:@"x"]) {
                                operand2=xValue;
                        }
                        else {
                                operand2=[[expr objectAtIndex:pos1+1] floatValue];
                        }
                        [expr removeObjectAtIndex:pos1-1];
                        [expr removeObjectAtIndex:pos1-1];
                        [expr removeObjectAtIndex:pos1-1];

                        operand1=operand1*operand2;
                        numberOperand = [NSNumber numberWithDouble:operand1];

                        [expr insertObject:numberOperand atIndex:pos1-1];
                }

        }
```

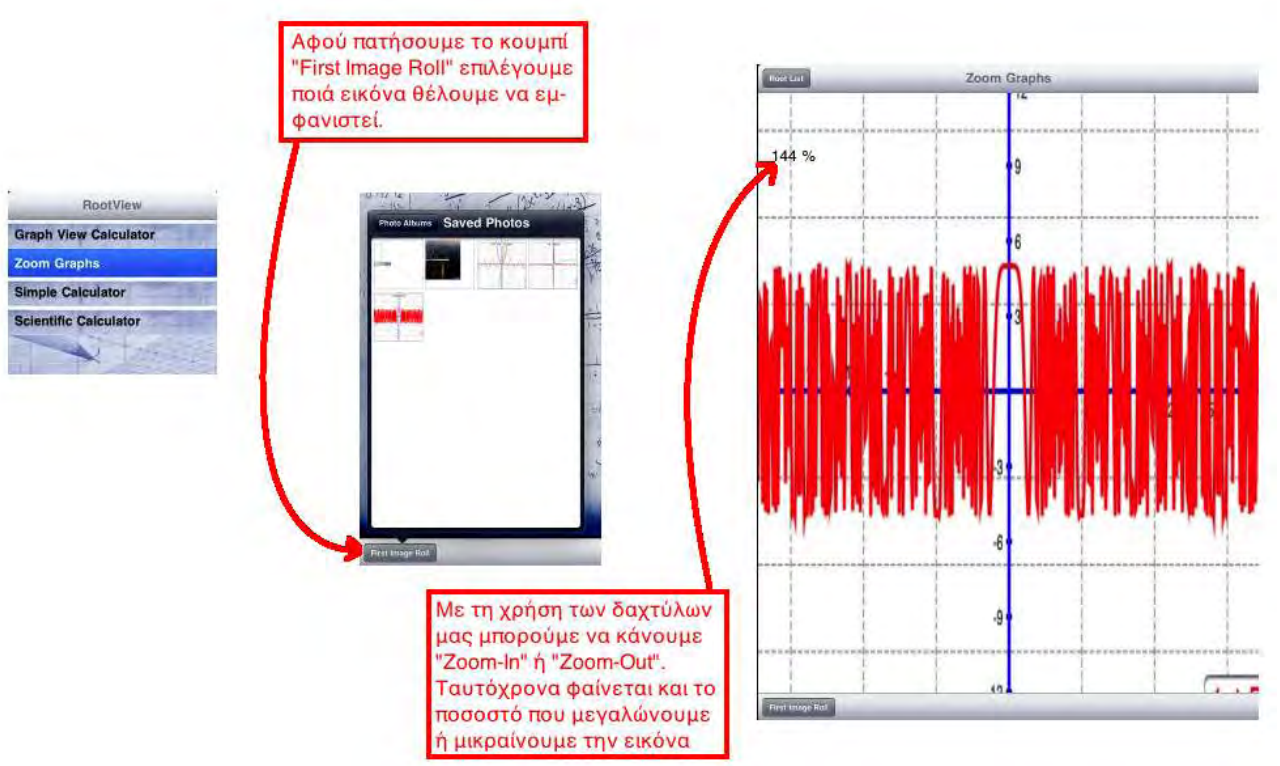
Σημείωση: Υπάρχει ακριβώς ο ίδιος κώδικας με αυτού του πολλαπλασιασμού και για τη διαίρεση και τη δύναμη, αλλά δεν επαναλαμβάνεται γιατί είναι ο ίδιος

```
        if ((pos1==NSIntegerMax)&&(pos2==NSIntegerMax)&&(pos3==NSIntegerMax)) {
                break;
        }
      }

      while (1) {
        pos1=[expr indexOfObject:@"+"];
        pos2=[expr indexOfObject:@"-"];

        if(pos1!=NSIntegerMax){
                if (pos1<pos2) {

                        if ([[expr objectAtIndex:pos1-1] isEqual:@"x"]) {
                                operand1=xValue;
                        }
```

```
                else {
                        operand1=[[expr objectAtIndex:pos1-1] doubleValue];
                }
                if ([[expr objectAtIndex:pos1+1] isEqual:@"x"]) {
                        operand2=xValue;
                }
                else {
                        operand2=[[expr objectAtIndex:pos1+1] doubleValue];
                }

                [expr removeObjectAtIndex:pos1-1];
                [expr removeObjectAtIndex:pos1-1];
                [expr removeObjectAtIndex:pos1-1];

                operand1=operand1+operand2;
                numberOperand = [NSNumber numberWithDouble:operand1];

                [expr insertObject:numberOperand atIndex:pos1-1];
        }

  }
  if(pos2!=NSIntegerMax){
        if (pos2<pos1) {

                if ([[expr objectAtIndex:pos2-1] isEqual:@"x"]) {
                        operand1=xValue;
                }
                else {
                        operand1=[[expr objectAtIndex:pos2-1] doubleValue];
                }
                if ([[expr objectAtIndex:pos2+1] isEqual:@"x"]) {
                        operand2=xValue;
                }
                else {
                        operand2=[[expr objectAtIndex:pos2+1] doubleValue];
                }
                [expr removeObjectAtIndex:pos2-1];
                [expr removeObjectAtIndex:pos2-1];
                [expr removeObjectAtIndex:pos2-1];

                operand1=operand1-operand2;
                numberOperand = [NSNumber numberWithDouble:operand1];

                [expr insertObject:numberOperand atIndex:pos2-1];
        }

  }

  if ((pos1==NSIntegerMax)&&(pos2==NSIntegerMax)) {
        break;
  }
}
```

```
        if ([[expr objectAtIndex:0] isEqual:@"x"]) {
          operand1=xValue;
          [expr removeObjectAtIndex:0];
        }

        [expr removeAllObjects];
        [expr release];

        return operand1;


}
```

4.2.2 SecondViewController



SecondDetailViewController.h

```
//Αυτό το τμήμα της εφαρμογής είναι για να μας δείξει τη δυνατότητα που έχει ο χρήστης να χειρίζεται εικόνες μέσω διάφορων ενεργειών που παρέχει το ipad
@interface SecondDetailViewController : UIViewController<UIScrollViewDelegate,
UIImagePickerControllerDelegate,
UINavigationControllerDelegate, UIPopoverControllerDelegate, UISplitViewControllerDelegate> {
        UIPopoverController *popoverController;
        MultipleDetailViewsWithNavigatorAppDelegate *appDelegate;

        //UIScrollView *firstScrollView;
        IBOutlet UIImageView *firstImageView;
        UILabel *zoomValue;
```

```
    UIToolbar *toolbar;
    UIPopoverController *popoverControllerAlbum;
}
@property (nonatomic, retain) UIPopoverController *popoverController;
@property (nonatomic, assign) MultipleDetailViewsWithNavigatorAppDelegate *appDelegate;

//@property (nonatomic, retain) IBOutlet UIScrollView *firstScrollView;
@property (nonatomic, retain) IBOutlet UIImageView *firstImageView;
@property (nonatomic, retain) IBOutlet UILabel *zoomValue;

@property (nonatomic, retain) UIPopoverController *popoverControllerAlbum;
@property (nonatomic, retain) IBOutlet UIToolbar *toolbar;

- (IBAction)useCameraRollFirst: (id)sender;

@end
```

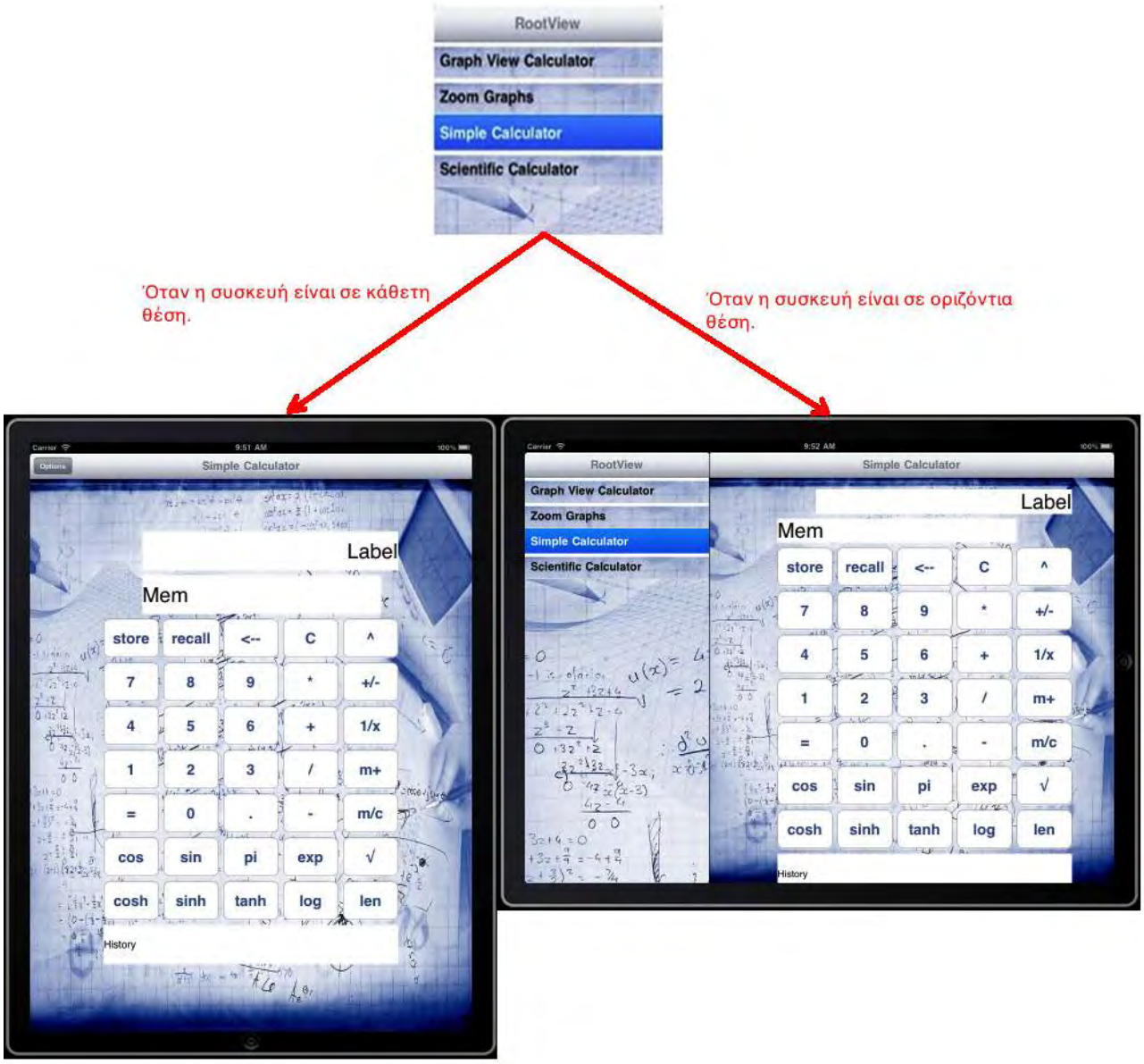
SecondDetailViewController.m

```
- (void)viewDidLoad {

    //Οι 3 επόμενες γραμμές είναι υπεύθυνες για την περίπτωση του διπλού "κλικ" πάνω στην οθόνη
    //Η ενέργεια που κάνουν καθορίζεται από τη συνάρτηση handleTapGesture που θα δείξουμε παρακάτω
    UITapGestureRecognizer *tapGesture = [[UITapGestureRecognizer alloc] initWithTarget:self
action:@selector(handleTapGesture:)];
    tapGesture.numberOfTapsRequired = 2;
    [self.firstImageView addGestureRecognizer:tapGesture];
    [tapGesture release];

    //Οι 2 επόμενες γραμμές ευθύνονται για την περίπτωση που κάνουμε τη χρήση 2 δακτύλων ταυτόχρονα για zoom-in και zoom-out
    //Η ενέργεια που κάνουν καθορίζεται από τη συνάρτηση handlePinchGesture όπου θα δείξουμε παρακάτω
    UIPinchGestureRecognizer *pinchGesture = [[UIPinchGestureRecognizer alloc]
initWithTarget:self action:@selector(handlePinchGesture:)];
    [firstImageView addGestureRecognizer:pinchGesture];
    [pinchGesture release];

    [super viewDidLoad];
    self.title=@"Zoom Graphs";

    UIBarButtonItem *cameraRollFirst = [[UIBarButtonItem alloc]
                                        initWithTitle:@"First Image
Roll"

    style:UIBarButtonItemStyleBordered
                                        target:self

    action:@selector(useCameraRollFirst:)];
```

```
    NSArray *items = [NSArray arrayWithObjects: cameraRollFirst, nil];
    [toolbar setItems:items animated:NO];

    [cameraRollFirst release];

    return;
}
//---handle tap gesture---
-(IBAction) handleTapGesture:(UIGestureRecognizer *) sender {
    if (sender.view.contentMode == UIViewContentModeScaleAspectFit)
      sender.view.contentMode = UIViewContentModeCenter;
    else
      sender.view.contentMode = UIViewContentModeScaleAspectFit;
}

//---handle pinch gesture---
-(IBAction) handlePinchGesture:(UIGestureRecognizer *) sender {

    //Παίρνει τιμή ανάλογα με ότι κάνουμε στην οθόνη
με τα δάχτυλα μας
    CGFloat factor = [(UIPinchGestureRecognizer *) sender scale];
    int intFactor;

    NSLog(@"%f", factor);
    factor = factor *100;
    intFactor = factor;
    NSString *strFactor = [NSString stringWithFormat:@"%d", intFactor];
    zoomValue.text = strFactor;
    zoomValue.text = [zoomValue.text stringByAppendingString:@" %"];

    factor = factor/100;

    //Σε περίπτωση που το factor>1 (δηλαδή το intFactor που
εμφανίζεται στην οθόνη) τότε κάνουμε zoom-in
    if (factor > 1) {
      firstImageView.transform = CGAffineTransformMakeScale(lastScaleFactor + (factor-1),

lastScaleFactor + (factor-1));
    }
    //Σε περίπτωση που το factor<=1 τότε κάνουμε zoom-out
    else {
      firstImageView.transform = CGAffineTransformMakeScale(lastScaleFactor * factor,

lastScaleFactor * factor);
    }

    if (sender.state == UIGestureRecognizerStateEnded){
      if (factor > 1) {
            lastScaleFactor += (factor-1);
      } else {
            lastScaleFactor *= factor;
      }
    }
```

}

4.2.3 ThirdViewController



ThirdDetailViewController.h

```
//Το τρίτο τμήμα της εφαρμογής είναι ένα απλό κουμπιουτεράκι όπου δείχνουμε σταδιακά από τα μέρη που αποτελείται και πώς εισάγουμε τους αριθμούς και τις
//πράξεις προκειμένου να γίνουν οι κατάλληλοι υπολογισμοί
//Επίσης, χρησιμοποιούμε και το ακόλουθο μέρος του προγράμματος για να δείξουμε πως λειτουργεί το αξελερόμετρο (accelometer) για να έχουμε σωστή αλλαγή των views
//ανάλογα με τη θέση που βρίσκεται η συσκευη
```

```
(οριζόντια-κάθετη)
@interface ThirdDetailViewController : UIViewController<UIPopoverControllerDelegate,
UISplitViewControllerDelegate, UIAccelerometerDelegate> {
      UIPopoverController *popoverController;
      MultipleDetailViewsWithNavigatorAppDelegate *appDelegate;

      //Όπως αναφέραμε παραπάνω έχουμε 2 διαφορετικά views για τις διαφορετικές θέσεις που μπορεί να βρίσκεται η συσκευή μας.
      UIView *view1;
      UIView *view2;

    UIToolbar *toolbar;

      //Εδώ εμφανίζονται τα νούμερα που εισάγουμε καθώς και το αποτέλεσμα κάθε πράξεις
      UILabel *display1;
      //Εδώ εμφανίζεται ό,τι θέλουμε να αντιγραψουμε απο το display πατώντας το κουμπι "store"
      UILabel *memDisplay1;
      //Αυτό βρίσκεται στο κάτω μέρος της οθόνης μας όπου εμφανίζεται ολόκληρη η πράξη μαζί με τα operands και το αποτέλεσμα
      UILabel *helpDisplay1;

      //Τα παρακάτω είναι ότι και τα πιο πάνω απλά τα χρησιμοποιούμε για το δεύτερο view
      UILabel *display2;
      UILabel *memDisplay2;
      UILabel *helpDisplay2;

      //Με τη βοήθεια του calculatrBrain2 που δημιουργήσαμε μπορούμε και κάνουμε τις πράξεις
      CalculatorBrain2 *brain;
      BOOL userInMiddleOfNumberEntry;
      BOOL decimalEntered;
      BOOL variableEntered;
}
@property (nonatomic, retain) UIPopoverController *popoverController;
@property (nonatomic, assign) MultipleDetailViewsWithNavigatorAppDelegate *appDelegate;

@property (nonatomic, retain) IBOutlet UIToolbar *toolbar;

@property (nonatomic, retain) IBOutlet UIView *view1;
@property (nonatomic, retain) IBOutlet UIView *view2;
@property (retain,nonatomic) IBOutlet UILabel *display1;
@property (retain,nonatomic) IBOutlet UILabel *memDisplay1;
@property (retain,nonatomic) IBOutlet UILabel *helpDisplay1;
@property (retain,nonatomic) IBOutlet UILabel *display2;
@property (retain,nonatomic) IBOutlet UILabel *memDisplay2;
@property (retain,nonatomic) IBOutlet UILabel *helpDisplay2;

//Για τους αριθμούς και την "."
```

```
-(IBAction)digitPressed:(UIButton *)sender;
//Για τις πράξεις
-(IBAction)operationPressed:(UIButton *)sender;
//Για το κουμπί "<--"
-(IBAction)digitDeleted:(UIButton *)sender;

@end
```

ThirdDetailViewController.m

```
-(IBAction)digitPressed:(UIButton *)sender{
      NSString *digit = sender.titleLabel.text;
      if ( !(decimalEntered == 1) || !([@"." isEqualToString:digit])) {
       if (userInMiddleOfNumberEntry) {
              display1.text = [display1.text stringByAppendingString:digit];
              display2.text = [display2.text stringByAppendingString:digit];
       }
       else {
              display1.text = digit;
              display2.text = digit;
              userInMiddleOfNumberEntry = YES;
       }
      }
      if ([@"." isEqualToString:digit]) decimalEntered = YES;
      return;
}

-(IBAction)digitDeleted:(UIButton *)sender{

      NSString *number= display1.text;
      //NSString *number= display2.text;
      NSString *last;

      if ( [number length] > 1 ){
       last = [number substringFromIndex:[number length] - 1];
       number = [number substringToIndex:[number length] - 1];

       if ([last isEqual:@"."]) {
              decimalEntered=0;
       }
      }
      else {
       number= @"";
      }

      display1.text=number;
      display2.text=number;
}

-(IBAction)operationPressed:(UIButton *)sender{

      if (userInMiddleOfNumberEntry) {
       //Παίρνει το περιεχόμενο του DISPLAY και το
```

```
μετατρέπει από string σε Double
        [brain setOperand:[[display1 text] doubleValue]];
        //Ξαναετοιμάζεται για να πάρει τον επόμενο αριθμό
        userInMiddleOfNumberEntry = NO;
        decimalEntered = NO;
      }
      //μετατρέπει την πράξει σε string
      NSString *operation = sender.titleLabel.text;

      double result = [brain doCalculation:operation];

      if([operation isEqual:@"store"]){
        memDisplay1.text = [NSString stringWithFormat:@"%g", [brain tempStore]];
        memDisplay2.text = [NSString stringWithFormat:@"%g", [brain tempStore]];
      }
      else if([operation isEqual:@"m+"]){
        memDisplay1.text = [NSString stringWithFormat:@"%g", [brain tempStore]];
        memDisplay2.text = [NSString stringWithFormat:@"%g", [brain tempStore]];
      }
      else if([operation isEqual:@"m/c"]) {
        memDisplay1.text = [NSString stringWithFormat:@"0"];
        memDisplay2.text = [NSString stringWithFormat:@"0"];
      }
      else if([operation isEqual:@"C"]) {
        display1.text = [NSString stringWithFormat:@"0"];
        display2.text = [NSString stringWithFormat:@"0"];
      }
      else if([operation isEqual:@"="]){
        [brain addResultToExpression:result];
        helpDisplay1.text= [CalculatorBrain2 descriptionOfExpression:brain.internalExpression];
        helpDisplay2.text= [CalculatorBrain2 descriptionOfExpression:brain.internalExpression];

        display1.text = [NSString stringWithFormat:@"%g", result];
        display2.text = [NSString stringWithFormat:@"%g", result];
        [brain cleanExpression];
      }
      else{
        display1.text = [NSString stringWithFormat:@"%g", result];
        display2.text = [NSString stringWithFormat:@"%g", result];
      }

      return;
}

//Κάνουμε τη χρήση του accelerometer προκειμένου να έχουμε
το επιθυμητό View
-(void)accelerometer:(UIAccelerometer *) acc
        didAccelerate:(UIAcceleration *) acceleration {

      if (acceleration.x == 1) {
        NSLog(@"Toympa....");
        self.view = self.view2;
      }
```

```
    else if (acceleration.x ==-1) {
      NSLog(@"Toympa....");
      self.view = self.view2;
    }

}
```

4.2.3.1 CalculatorBrain2

CaclulatorBrain2.h

```
@interface CalculatorBrain2 : NSObject
{
    //Για τις συνθήκες σφάλματος
    int errorFlag;
    //Τα operand
    double newNumber;
    double firstNumber;
    //Για να κρατάει στη μνήμη αυτό που υπάρχει στο
display
    double tempStore;

    //Η πράξη
    NSString *waitingOperation;

    //Είναι ένας πίνακας για να κρατάει ό,τι έχουμε
δώσει και να το εμφανίζει τέρμα κάτω
    NSMutableArray *internalExpression;
}

@property(nonatomic) double tempStore;
@property(nonatomic) double newNumber;
@property(nonatomic) double firstNumber;

@property(nonatomic, retain) NSMutableArray *internalExpression;

//Για την αρχικοποίηση
-(id)init;
//Για να δέχεται τον αριθμό που πληκτρολογεί ο
χρήστης και να τον αποθηκεύει σαν operand και επίσης τον
αποθηκεύει στο internalExpression
-(void)setOperand:(double)aDouble;
//Είναι το κύριο μέρος αυτού του αρχείο, όπου κάνει
τους υπολογισμούς και επίσης το αποθηκεύει στο
internalExpression
-(double)doCalculation:(NSString *)operation;
//Αποθηκεύει το αποτέλεσμα στο internalExpression
-(void) addResultToExpression: (double)result;
//Ενεργοποιείται όταν πατάμε το κουμπί "C" και
"καθαρίζει" το firstNumber, το newNumber, το tempStore και το
waitingOperation
//Ουσιαστικά όμως η "δουλειά" του είναι να καθαρίζει
το (NSMutableArray) internalExpression και ναπομακρύνει όλα όσα
περιέχει
```

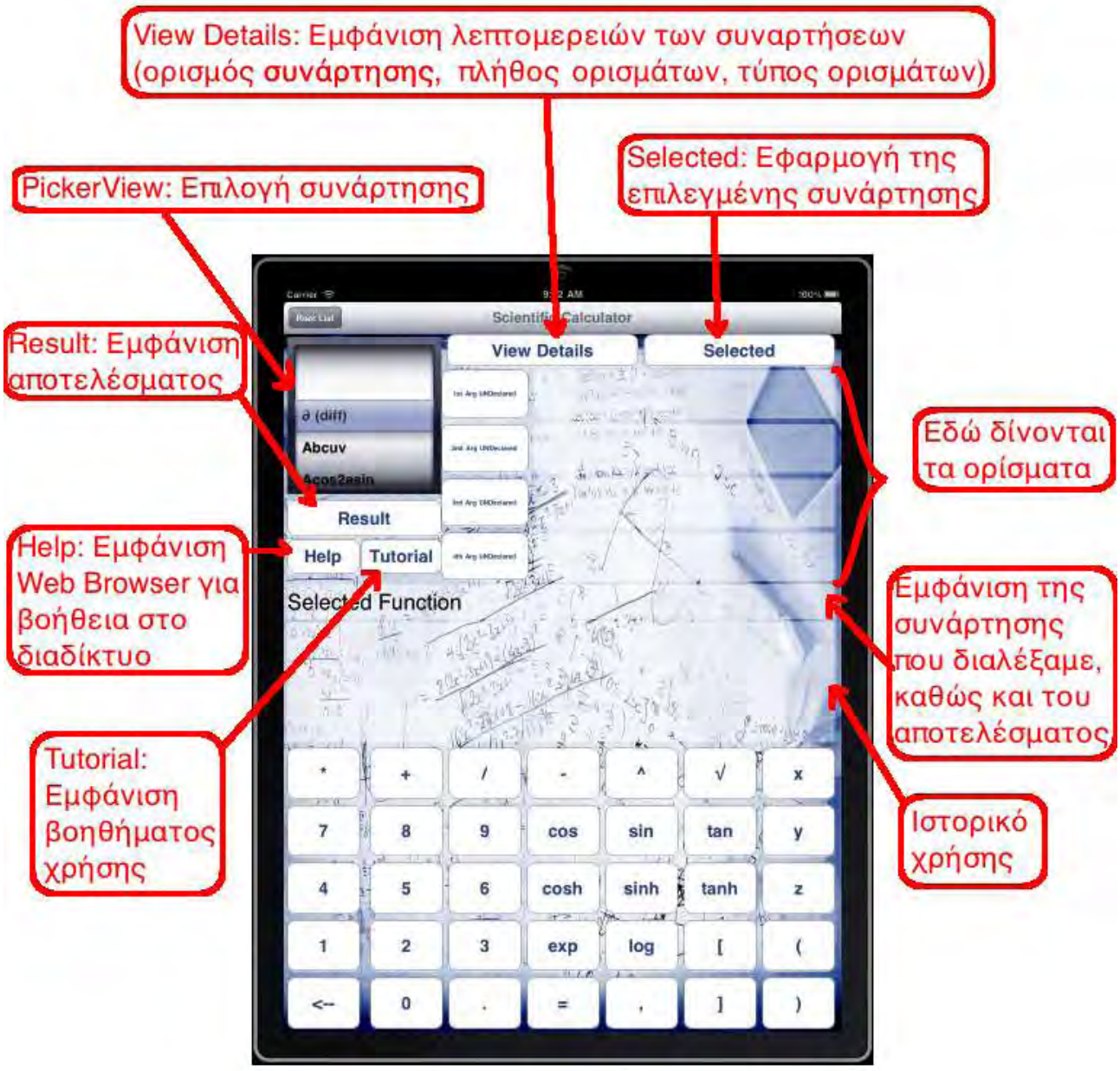
```
-(void) cleanExpression;

+(NSString *) descriptionOfExpression: (id)anExpression;

@end
```

CalculatorBrain2.m

```
//Χρησιμοποιείται από 3 συναρτήσεις για την
αποθήκευση α)της πράξης, β)τον στοιχείων της πράξης
και γ)του αποτελέσματος στο internalExpression
//Οι συναρτήσεις είναι η doCalculation, η setOperand και η
addResultToExpression
-(void) addToExpression:(id) keyLabel{

      if (!internalExpression){
        internalExpression=[[NSMutableArray alloc] init];
      }
      [internalExpression addObject:keyLabel];

      return;
}
//Καλείται από το doCalculation για να κάνεις τις πράξεις
-(void)doWaitingOperation
{
      if ([@"+" isEqual:waitingOperation]) newNumber = firstNumber + newNumber;
      else if ([@"*" isEqual:waitingOperation]) newNumber = firstNumber * newNumber;
      else if ([@"-" isEqual:waitingOperation]) newNumber = firstNumber - newNumber;
      else if ([@"/" isEqual:waitingOperation]) {
        if (newNumber){
              newNumber = firstNumber / newNumber;
        }
        else {
              errorFlag = 1;
        }
      }
      else if ([@"^"isEqual:waitingOperation]) newNumber = pow(firstNumber, newNumber);
      return;
}

//Η καρδιά του CalculatorBrain2
-(double)doCalculation:(NSString *)operation
{
      [self addToExpression:operation];
      if (     [@"+/-"      isEqual:operation]) newNumber = - newNumber;
      else if ([@"1/x"      isEqual:operation] && (newNumber > 0)) newNumber = 1/newNumber;
      else if ([@"√"        isEqual:operation] && (newNumber >= 0)) newNumber = sqrt(newNumber);
      else if ([@"pi"       isEqual:operation]) newNumber = M_PI;
      else if ([@"sin"      isEqual:operation]) newNumber = sin(newNumber);
      else if ([@"cos"      isEqual:operation]) newNumber = cos(newNumber);
      else if ([@"store"  isEqual:operation]) tempStore = newNumber;
      else if ([@"recall" isEqual:operation]) newNumber = tempStore;
      else if ([@"m+"     isEqual:operation]) tempStore = tempStore + newNumber;
```

```
    else if ([@"m/c"    isEqual:operation]) tempStore = 0.0;
    else if ([@"log"      isEqual:operation]) newNumber = log10(newNumber);
    else if ([@"ln"       isEqual:operation]) newNumber = log(newNumber);
    else if ([@"exp"      isEqual:operation]) newNumber = exp(newNumber);
    else if ([@"tanh"     isEqual:operation]) newNumber = tanh(newNumber);
    else if ([@"cosh"     isEqual:operation]) newNumber = cosh(newNumber);
    else if ([@"sinh"     isEqual:operation]) newNumber = sinh(newNumber);
    else if ([@"c"        isEqual:operation])
    {
      firstNumber = 0.0;
      newNumber = 0.0;
      tempStore = 0.0;
      waitingOperation = nil;
      operation = nil;
    }
    else {
      [self doWaitingOperation];
      waitingOperation = operation;
      firstNumber = newNumber;
    }
    return newNumber;
}

//Είναι μια συνάρτηση η οποία δε χρησιμοποιείται
+(NSString *)descriptionOfExpression:(id)anExpression
{
    NSMutableString *expDisplayString = [[NSMutableString alloc]init];
    for (id obj in anExpression) {
      if ([obj isKindOfClass:[NSNumber class]]) {
            [expDisplayString appendString:[obj stringValue]];
            [expDisplayString appendString:@" "];
      }
      else if ([obj isKindOfClass:[NSString class]]) {
            [expDisplayString appendString:obj];
            [expDisplayString appendString:@" "];
      }
    }
    [expDisplayString autorelease];
    return expDisplayString;
}
```

4.2.4 FourthViewController



FourthDetailViewController.h

```
#import <UIKit/UIKit.h>
#import "MultipleDetailViewsWithNavigatorAppDelegate.h"
//Κάνουμε #import το παρακάτω αρχείο προκειμένου να εμφανίζεται στο popover το view του MyPopOverView
#import "MyPopOverView.h"
//Ομοίως και το παρακάτω για τον απλό webBrowser που έχουμε σχεδιάσει, και εμφανίζεται μέσω navigationController
#import "WebViewController.h"
//Το ίδιο συμβαίνει και με το παρακάτω προκειμένου να έχουμε ένα Tutorial με εικόνες που κάνουν Scroll μέσω navigationController
#import "ScrollingViewController.h"
```

```
@interface FourthDetailViewController : UIViewController<
UIPopoverControllerDelegate,
UISplitViewControllerDelegate,
UIPickerViewDelegate,
UIPickerViewDataSource> {
    UIPopoverController *popoverController;
    MultipleDetailViewsWithNavigatorAppDelegate *appDelegate;
    //Το PickerView χρησιμοποιείται για να εμφανίζονται οι εντολές που έχουμε διαθέσιμες και τις έχουμε αποθηκεύσει στο pickerData.
    UIPickerView *singlePicker;
    NSArray *pickerData;
    //Κουμπί για να εμφανίζονται οι πληροφορίες για τις συναρτησεις μέσω του popover myPopOver
    UIButton *details;
    //Κουμπί προκειμένου να επιλέξουμε τη συνάρτηση που βρίσκεται στο PickerView
    UIButton *selec;
    //Τα παρακάτω κουμπιά είναι για να γίνει επιτυχής κατυχώρηση των arguments
    UIButton *butArg1;
    UIButton *butArg2;
    UIButton *butArg3;
    UIButton *butArg4;
    UIButton *butArg1No;
    UIButton *butArg2No;
    UIButton *butArg3No;
    UIButton *butArg4No;
    //UILabel *firstArg;
    //Τα παρακάτω 4 TextView είναι για να εμφανίζονται τα (μεχρι) 4 ορίσματα που μπορούμε να δώσουμε στις συναρτήσεις
    UITextView *firstArg;
    //UILabel *secondArg;
    UITextView *secondArg;
    //UILabel *thirdArg;
    UITextView *thirdArg;
    //UILabel *fourthArg;
    UITextView *fourthArg;
    //Το παρακάτω TextView είναι για να κρατάμε ιστορικό με τις συναρτήσεις που έχουμε χρησιμοποιήσει καθώς και για τα αποτελέσματα αυτών
    UITextView *history;
    //Με τις παρακάτω boolean μεταβλητές διακρίνουμε αν έχουμε δώσει το κάθενα από τα 4 argument
    BOOL arg1;
    BOOL arg2;
    BOOL arg3;
    BOOL arg4;
    BOOL decimalEntered;
    //τα παρακάτω 4 NSMutableArray τα χρησιμοποιούμε για να αποθηκεύουμε τα περιεχόμενα των 4 παραπανω TextView
```

```
    NSMutableArray *expression1;
    NSMutableArray *expression2;
    NSMutableArray *expression3;
    NSMutableArray *expression4;

    //Εκεί εμφανίζεται η συνάρτηση που επιλέξαμε για
να χρησιμοποιήσουμε
    UILabel *selectedFunction;

    MyPopOverView *myPopOver;

    IBOutlet UIButton *btn;

}
@property (nonatomic, retain) UIPopoverController *popoverController;
@property (nonatomic, assign) MultipleDetailViewsWithNavigatorAppDelegate *appDelegate;

@property (nonatomic, retain) IBOutlet UIPickerView *singlePicker;
@property (nonatomic, retain) NSArray *pickerData;
@property (nonatomic, retain) IBOutlet UIButton *details;
@property (nonatomic, retain) IBOutlet UIButton *selec;
@property (nonatomic, retain) IBOutlet UIButton *butArg1;
@property (nonatomic, retain) IBOutlet UIButton *butArg2;
@property (nonatomic, retain) IBOutlet UIButton *butArg3;
@property (nonatomic, retain) IBOutlet UIButton *butArg4;
@property (nonatomic, retain) IBOutlet UIButton *butArg1No;
@property (nonatomic, retain) IBOutlet UIButton *butArg2No;
@property (nonatomic, retain) IBOutlet UIButton *butArg3No;
@property (nonatomic, retain) IBOutlet UIButton *butArg4No;
//@property (nonatomic, retain) IBOutlet UILabel *firstArg;
@property (nonatomic, retain) IBOutlet UITextView *firstArg;
//@property (nonatomic, retain) IBOutlet UILabel *secondArg;
@property (nonatomic, retain) IBOutlet UITextView *secondArg;
//@property (nonatomic, retain) IBOutlet UILabel *thirdArg;
@property (nonatomic, retain) IBOutlet UITextView *thirdArg;
//@property (nonatomic, retain) IBOutlet UILabel *fourthArg;
@property (nonatomic, retain) IBOutlet UITextView *fourthArg;
@property (nonatomic, retain) IBOutlet UITextView *history;
@property (nonatomic, retain) NSMutableArray *expression1;
@property (nonatomic, retain) NSMutableArray *expression2;
@property (nonatomic, retain) NSMutableArray *expression3;
@property (nonatomic, retain) NSMutableArray *expression4;

//Δίνει ενέργεια στο κουμπί "selec"
-(IBAction)buttonSelected;
//Δίνει ενέργεια στα κουμπιά butArg1, butArg2, butArg3, butArg4,
butArg1No, butArg2No, butArg3No, butArg4No
-(IBAction)buttonArg:(UIButton *)sender;
//Δίνει ενέργεια στο κουμπί με ονομασία "Result" για να
παράξει και να εμφανίσει το αποτέλεσμα
-(IBAction)buttonAction;

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

```
-(IBAction)operationPressed:(UIButton *)sender;
-(IBAction)digitDeleted:(UIButton *)sender;

@property (nonatomic, retain) IBOutlet UILabel *selectedFunction;

@property (nonatomic, retain) MyPopOverView *myPopOver;
@property (nonatomic, retain) UIButton *btn;
//Δίνει ενέργεια στο κουμπί που εμφανίζει τις
πληροφορίες για τις συναρτήσεις
-(IBAction) btnShow:(id) sender;
//δίνει ενέργεια στο κουμπί με ονομασία "Help" και μας
παραπέμπει σε ένα webBrowser
-(IBAction) btnWeb:(id) sender;
//δίνει ενέργεια στο κουμπί με ονομασία "Tutorial" που μας
παραπέμπει σε ένα άλλο View με ScrollView.
//Έτσι μας δίνεται η ευκολία μέσω κάποιο εικόνων να
μάθουμε πως να χειριστούμε το εργαλείο αυτό.
-(IBAction) tutorial:(id) sender;
- (IBAction)backgroundTouched:(id)sender;
- (IBAction)textFieldReturn:(id)sender;

@end
```

FourthDetailViewController.mm

```
Καλούμε τις βιβλιοθήκες που θα χρειαστούν για τους
υπολογισμούς
#import "FourthDetailViewController.h"
#include "gmp.h"
#include "gmpxx.h"
#include <giac/giac.h>
#include <giac/misc.h>
#include <giac/csturm.h>
#include <string>
#include <iostream>

using namespace std;
using namespace giac;


-(IBAction) btnWeb:(id) sender{
    //Όπως βλέπουμε δημιουργούμε ένα "αντίγραφο" του
WebViewController και εν συνεχεία το κάνουμε "push" με
navigationController για να εμφανιστεί σε ένα διαφορετικό View
    WebViewController *wvc = [[WebViewController alloc] init];

    [self.navigationController pushViewController:wvc animated:YES];
    [wvc release];
}

-(IBAction) btnShow:(id) sender {

    //Αν δεν έχουμε έτοιμο popover τότε το δημιουργούμε
```

```
και το οποίο θα περιλαμβάνει αντίγραφο του MyPopOverView
    if (self.popoverController == nil) {
        myPopOver = [[MyPopOverView alloc] initWithNibName:@"MyPopOverView" bundle: nil];
       UIPopoverController *popover = [[UIPopoverController alloc]
initWithContentViewController:myPopOver];

       self.popoverController = popover;
        [popover release];
    }
     //Στις παρακάτω 2 εντολές δίχνουμε που θέλουμε να
βρίσκεται το popover και τι μέγεθος θέλουμε να έχει
     //πχ το CGRectMake(0,0,950,30); το 1ο και 2ο όρισμα είναι το
σημείο εκκίνησης, το 3ο είναι το πλάτος και το 4ο
είναι το ύψος
     // (0,0)              (950,0)
     //      x - - - - - - - - x
     //      |                 |
     //      |                 |
     //      |                 |
     //      x - - - - - - - - x
     //      (0,30)                    (950,30)
     //άρα η κάτω δεξιά γωνία θα είναι το σημείο (εδώ θα
είναι το σημείο (950,30) ) που θα εμφανίζεται η "μύτη" από
το popover
     //Επίσης η δεύτερη εντολή μας δείχνει το μέγεθος
του παραθύρου που θα εμφανιστεί
     //Προσοχή το παράθυρο δεν μπορεί να έχει μεγάλες
διαστάσεις
    CGRect popoverRect = CGRectMake(0, 0, 950, 30);
     [self.popoverController setPopoverContentSize:CGSizeMake(500.0f, 200.0f)];

    [self.popoverController
      presentPopoverFromRect:popoverRect
      inView:self.view
      permittedArrowDirections:UIPopoverArrowDirectionAny
      animated:YES];

     //Παρακάτω μέσω της σειράς του pickerView
καταλαβαίνουμε ποιο έχουμε διαλέξει και το
αποθηκεύουμε σε ένα NSString
     NSInteger row = [singlePicker selectedRowInComponent:0];
     NSString *selected = [pickerData objectAtIndex:row];

     //Όταν επιλέξουμε μία από τις συναρτήσεις τότε
έχουμε τη δυνατότητα μέσω image ή μέσω TextView να
εμφανιστούν οι αντίστοιχες πληροφορίες
     //Όταν επιλέγουμε να εμφανιστούν οι πληροφορίες
μέσω image, εξαφανίζουμε το TextView ή όταν επιλέγουμε να
εμφανιστούν με TextView, εξαφανίζουμε το image
     //Για τις παρακάτω συναρτήσεις οι πληροφορίες
εμφανίζονται μέσω TextView
     if([selected isEqual:@"Limit"]){
      myPopOver.imageDetails.hidden = YES;
```

```
        myPopOver.example.text = [NSString stringWithString:@"limit compute the limit of an
expression in a finite or infinite point. It is also possible with an optional argument to
compute unidirectional limit (1 for right limit and -1 for left limit .limit takes three or four
arguments : ¥n1)an expression, ¥n2)the name of a variable (for example x), ¥n3)the limit point
(for example a) and ¥n4)an optional argument, by default 0, to indicate if the limit is
unidirectional. n)This argument is equal to -1 for a left limit (x<a) or is equal to 1 for a
right limit (x>a) or is equal to 0 for a limite. limit returns the limit of the expression when
the variable (for example x) approachs the limit point (for example a). ¥nInput :
limit(1/x,x,0,-1)¥nOutput : -(infinity)¥nInput : limit(1/x,x,0,1)¥nOutput : +(infinity)¥nInput :
limit(1/x,x,0,0)¥nOutput : infinity"];
    }
//Για τις παρακάτω συναρτήσεις διαλέγουμε να
εμφανιστούν οι πληροφορίες με τη βοήθεια image
    else if([selected isEqual:@"RSolve"]){
      myPopOver.imageDetails.hidden = YES;
      myPopOver.example.text = [NSString stringWithString:@"Eikona"];

      myPopOver.imageDetails.hidden = NO;
      [myPopOver.imageDetails setImage:[UIImage imageNamed:@"reverse_rsolve"]];
      //παρακάτω φαίνεται πως δίνουμε διαφορετικές
τιμές για το πώς και πού θα εμφανίζεται το popover
      CGRect popoverRect = CGRectMake(0, 0, 600, 30);
      [self.popoverController setPopoverContentSize:CGSizeMake(600.0f, 800.0f)];

      [self.popoverController
       presentPopoverFromRect:popoverRect
       inView:self.view
       permittedArrowDirections:UIPopoverArrowDirectionAny
       animated:YES];
    }


Σημείωση: Ακολουθούν και οι υπόλοιπες συναρτήσεις
.......
}

-(IBAction) tutorial:(id) sender{

    //Δημιουργούμε ένα αντίγραφο του ScorllingViewController και
το κάνουμε push
    //Παρακάτω θα δούμε τι περιέχει το ScrollingViewController
    ScrollingViewController *wvc = [[ScrollingViewController alloc] init];

    [self.navigationController pushViewController:wvc animated:YES];
    [wvc release];

}

-(IBAction)buttonAction{

    //Ανάλογα με αυτό που διαλέξαμε μας επιστρέφει ένα
NSInteger με το οποίο βρίσκουμε απο το pickerData το NSString που
αντιστοιχεί στη συνάρτηση που διαλέξαμε
```

```
	NSInteger row = [singlePicker selectedRowInComponent:0];
	NSString *selected = [pickerData objectAtIndex:row];
	NSString *selected1 = [pickerData objectAtIndex:row];

	NSString *input;
	string input1;

	//Ανάλογα με τη συνάρτηση που έχουμε ορίζουμε το πλήθος ορισμάτων που πρέπει να μας επιστρέψει ο χρήστης
	//Αν δε μας επιστρέψει ο χρήστης το σωστό πλήθος ορισμάτων τότε ειδοποιείται με ένα warning
	//ΠΡΟΣΟΧΗ: τα ορίσματα για να κατοχηρωθούν πρέπει να πατήσουμε τα κουμπιά που βρίσκονται αριστερά απο το TextView
	if([selected isEqual:@"Limit"]){

	 //Έτσι ελέγχουμε αν το πλήθος των ορισμάτων που έχουν δωθεί είναι σωστό, διαφορετικά μας εμφανίζει μια προειδοποίηση
	 if(arg3==NO){
		UIAlertView *error =
		[[UIAlertView alloc] initWithTitle:@"Error"
		                                   message:@"You don´t put enouph arguments! ¥n Please check it!"
		                                  delegate:self cancelButtonTitle:@"Okay"
		                         otherButtonTitles:nil];
		[error show];
		[error release];
	 }
	 else{

		if (arg4==YES) {
			selectedFunction.text=[NSString stringWithString:selected];
			selectedFunction.text=[selectedFunction.text stringByAppendingString:@"("];
			selectedFunction.text=[selectedFunction.text stringByAppendingString:firstArg.text];
			selectedFunction.text=[selectedFunction.text stringByAppendingString:@","];
			selectedFunction.text=[selectedFunction.text stringByAppendingString:secondArg.text];
			selectedFunction.text=[selectedFunction.text stringByAppendingString:@","];
			selectedFunction.text=[selectedFunction.text stringByAppendingString:thirdArg.text];
			selectedFunction.text=[selectedFunction.text stringByAppendingString:@","];
			selectedFunction.text=[selectedFunction.text stringByAppendingString:fourthArg.text];
			selectedFunction.text=[selectedFunction.text stringByAppendingString:@")"];
			selectedFunction.text=[selectedFunction.text stringByAppendingString:@"-->>"];


			input=[NSString stringWithString:firstArg.text];
```

```
                input=[input stringByAppendingString:@","];
                input=[input stringByAppendingString:secondArg.text];
                input=[input stringByAppendingString:@","];
                input=[input stringByAppendingString:thirdArg.text];
                input=[input stringByAppendingString:@","];
                input=[input stringByAppendingString:fourthArg.text];

                input1 = [input UTF8String];
            }
            else {
                selectedFunction.text=[NSString stringWithString:selected];
                selectedFunction.text=[selectedFunction.text stringByAppendingString:@"("];
                selectedFunction.text=[selectedFunction.text
stringByAppendingString:firstArg.text];
                selectedFunction.text=[selectedFunction.text stringByAppendingString:@","];
                selectedFunction.text=[selectedFunction.text
stringByAppendingString:secondArg.text];
                selectedFunction.text=[selectedFunction.text stringByAppendingString:@","];
                selectedFunction.text=[selectedFunction.text
stringByAppendingString:thirdArg.text];
                selectedFunction.text=[selectedFunction.text stringByAppendingString:@")"];
                selectedFunction.text=[selectedFunction.text stringByAppendingString:@"--
>>"];

                input=[NSString stringWithString:firstArg.text];
                input=[input stringByAppendingString:@","];
                input=[input stringByAppendingString:secondArg.text];
                input=[input stringByAppendingString:@","];
                input=[input stringByAppendingString:thirdArg.text];

                //Δημιουργούμε το input (που είναι σε μορφή
NSString) το οποίο και μετατρέπουμε με τον ακόλουθο τρόπο
προκειμένου (σε string) προκειμένου
                //να είναι σωστό το όρισμα που θα δώσουμε
στη συνάρτηση της C++ που στην προκείμενη περίπτωση
είναι η _limit()
                input1 = [input UTF8String];
            }

            identificateur x("x");
            gen e16(input1,0);
            gen e = _limit(e16, 0);

            string str1 = print(e,0);
            const char *str = str1.c_str();

            NSString *s = [[NSString alloc] initWithUTF8String:str];

            selectedFunction.text = [selectedFunction.text stringByAppendingString:s];

            selected1 = [selected1 stringByAppendingString:@"("];
            selected1 = [selected1 stringByAppendingString:input];
            selected1 = [selected1 stringByAppendingString:@")"];
```

```
            history.text=[history.text stringByAppendingString:selected1];
            history.text=[history.text stringByAppendingString:@"¥n"];

            history.text=[history.text stringByAppendingString:s];
            history.text=[history.text stringByAppendingString:@"¥n"];

            [s release];
      }
    }
Σημείωση: Ακολουθούν και οι υπόλοιπες συναρτήσεις
.........
}

-(IBAction)buttonSelected{

    arg1=NO;
    arg2=NO;
    arg3=NO;
    arg4=NO;

    firstArg.text=@"";
    secondArg.text=@"";
    thirdArg.text=@"";
    fourthArg.text=@"";

    butArg1.hidden = NO;
    butArg2.hidden = NO;
    butArg3.hidden = NO;
    butArg4.hidden = NO;
    butArg1No.hidden = YES;
    butArg2No.hidden = YES;
    butArg3No.hidden = YES;
    butArg4No.hidden = YES;

    NSInteger row = [singlePicker selectedRowInComponent:0];
    NSString *selected = [pickerData objectAtIndex:row];

    //Με την παρακάτω διαδικασία επιλέγουμε πόσα textView
θα εμφανίζονται για να δώσουμε τα ορίσματα τα οποία
χρειάζεται κάθε συνάρτηση,
    //καθώς επίσης και τα κουμπία τα οποία θα
εμφανίζονται για να κατοχυρώσουμε τα ορίσματα
    if([selected isEqual:@"Limit"]){
      selectedFunction.text=[NSString stringWithString:@"Limit( , ) insert 3 or 4 arguments
please"];
      butArg1.hidden=NO;
      butArg2.hidden=NO;
      butArg3.hidden=NO;
      butArg4.hidden=NO;
      firstArg.hidden=NO;
      secondArg.hidden=NO;
      thirdArg.hidden=NO;
```

```
        fourthArg.hidden=NO;
      }
Σημείωση: Ακολουθούν και οι υπόλοιπες συναρτήσεις.
.........
}

-(IBAction)buttonArg:(UIButton *)sender{

      NSString *arg=sender.titleLabel.text;

      //Ανάλογα ποιό κουμπί έχουμε πατήσει
κατοχηρώνουμε το αντίστοιχο όρισμα και μεταφερόμαστε
στο επόμενο text προς συμπλήρωση.
      if ([@"1st Arg UNDeclared" isEqualToString:arg]) {
        arg1=YES;
        butArg1.titleLabel.text = [NSString stringWithFormat:@"1st Argument OK"];
        butArg1.hidden = YES;
        butArg1No.hidden = NO;
      }
      else if ([@"2nd Arg UNDeclared" isEqualToString:arg]) {
        arg2=YES;
        butArg2.titleLabel.text = [NSString stringWithFormat:@"2nd Argument OK"];
        butArg2.hidden = YES;
        butArg2No.hidden = NO;
      }
      else if ([@"3rd Arg UNDeclared" isEqualToString:arg]) {
        arg3=YES;
        butArg3.titleLabel.text = [NSString stringWithFormat:@"3rd Argument OK"];
        butArg3.hidden = YES;
        butArg3No.hidden = NO;
      }
      else if ([@"4th Arg UNDeclared" isEqualToString:arg]) {
        arg4=YES;
        butArg4.titleLabel.text = [NSString stringWithFormat:@"4th Argument OK"];
        butArg4.hidden = YES;
        butArg4No.hidden = NO;
      }

}
- (void)viewDidLoad {
    [super viewDidLoad];
      self.title=@"Scientific Calculator";

      butArg1.hidden = NO;
      butArg2.hidden = NO;
      butArg3.hidden = NO;
      butArg4.hidden = NO;
      butArg1No.hidden = YES;
      butArg2No.hidden = YES;
      butArg3No.hidden = YES;
      butArg4No.hidden = YES;

      //Προσδιορίζουμε το Font της συμβολοσειράς που
```

```
γράφουμε στα 4 TextView
      [firstArg setFont:[UIFont systemFontOfSize:40]];
      [secondArg setFont:[UIFont systemFontOfSize:40]];
      [thirdArg setFont:[UIFont systemFontOfSize:40]];
      [fourthArg setFont:[UIFont systemFontOfSize:40]];

      //Όπως έχουμε αναφέρει προηγουμένως οι επιλογές
που εμφανίζονται στο pickerView βρίσκονται στο pickerdata
      //To pickerdata εμπεριέχει ό,τι περιέχει και το array
      NSArray *array = [[NSArray alloc] initWithObjects:
                              @"∂ (diff)",
                              @"Abcuv",
                              @"Acos2asin",
                              @"ALU",
                              @"At",
                              @"Bernulli",
Σημείωση: Ακολουθούν και οι υπόλοιπες συναρτήσεις .
.............

self.pickerData = array;
      [array release];
}
```

4.2.4.1 MyPopOverView

MyPopOverView.mm

```
@implementation MyPopOverView

@synthesize example;

@synthesize imageDetails;


- (BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)interfaceOrientation {

    return YES;
}


- (void)didReceiveMemoryWarning {

      [super didReceiveMemoryWarning];

}


- (void)viewDidUnload {
    [super viewDidUnload];

      self.example = nil;
      self.imageDetails = nil;

}
```

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

      [example release];
      [imageDetails release];
}


@end

WebViewController.h
@interface WebViewController : UIViewController {

      IBOutlet UIWebView *webView;
      IBOutlet UITextField *webAdress;
      //Εί ν α ι  α υ τ ό  π ο υ  δ ε ί  χ ν ε ι  ο τ ι  φ ο ρ τ ώ ν ε ι  η  σ ε λ ί δ α
      IBOutlet UIActivityIndicatorView *active;

}
//Δ ί ν ε ι  ε ν έ ρ γ ε ι α  σ τ ο  UITextField π ρ ο κ ε ι μ έ ν ο υ  ν α
α π ο μ α κ ρ ύ ν ε τ α ι  τ ο  π λ η κ τ ρ ο λ ό γ ι ο  τ η ς  σ υ σ κ ε υ ή ς
-(IBAction)closeText;

@end


//Α ξ ι ο σ η μ ε ί ω τ ο  ε ί ν α ι  ν α  α ν α φ έ ρ ο υ μ ε  π ω ς  έ ν α  UIWebView έ χ ε ι
ή δ η  ο ρ ι σ μ έ ν ε ς  τ ι ς  ε ξ ή ς  ε ν έ ρ γ ε ι ε ς : goBack, goForward, reload κ α ι
stopLoading κ α ι  έ τ σ ι  ν α  τ ι ς  σ υ ν δ έ σ ο υ μ ε  μ ε  ό π ο ι α  κ ο υ μ π ι ά
θ έ λ ο υ μ ε

WebViewController.m

- (void)viewDidLoad {
    [super viewDidLoad];

      //Β ά ζ ο υ μ ε  σ τ ο  TextField τ η ν  ι σ τ ο σ ε λ ί δ α  π ο υ  θ α
φ ο ρ τ ώ σ ο υ μ ε
      webAdress.text = [webAdress.text stringByAppendingFormat:@"http://www.google.com"];
      //Φ ο ρ τ ώ ν ο υ μ ε  τ η  σ ε λ ί δ α  π ο υ  ε π ι θ υ μ ο ύ μ ε  γ ι α  α ρ χ ι κ ή
      [webView loadRequest:[NSURLRequest requestWithURL:[NSURL
URLWithString:@"http://www.google.com"]]];
      [NSTimer scheduledTimerWithTimeInterval:1.0 target:self selector:@selector(checkLoad)
userInfo:nil repeats:YES];
      [NSTimer scheduledTimerWithTimeInterval:1.0 target:self selector:@selector(checkNotLoad)
userInfo:nil repeats:YES];
}

//Ό σ η  ώ ρ α  φ ο ρ τ ώ ν ε τ α ι  η  σ ε λ ί δ α  ε ν ε ρ γ ο π ο ι ε ί τ α ι  τ ο  Activity
Indicator (active)
- (void)checkLoad {
```

```
      if (webView.loading) {
        [active startAnimating];
      }
}
//Όταν σταματή σει το "φόρτωμα" της σελίδας σταματάει το Activity Indicator
- (void)checkNotLoad {
      if (!(webView.loading)) {
        [active stopAnimating];
      }
}
```

4.2.4.2 WebViewController

```
WebViewController.h

@interface WebViewController : UIViewController {

      IBOutlet UIWebView *webView;
      IBOutlet UITextField *webAdress;
      //Είναι αυτό που δείχνει οτι φορτώνει η σελίδα
      IBOutlet UIActivityIndicatorView *active;

}
//Δίνει ενέργεια στο UITextField προκειμένου να απομακρύνεται το πληκτρολόγιο της συσκευής
-(IBAction)closeText;

@end


//Αξιοσημείωτο είναι να αναφέρουμε πως έναUIWebView έχει ήδη ορισμένες τις εξής ενέργειες: goBack, goForward, reload και stopLoading και έτσι να τις συνδέσουμε με όποια κουμπιά θέλουμε

WebViewController.m

@implementation WebViewController

-(IBAction)closeText {

      [webView loadRequest:[NSURLRequest requestWithURL:[NSURL URLWithString:[webAdress text]]]];

}

- (void)viewDidLoad {
    [super viewDidLoad];

      //Βάζουμε στο TextField την ιστοσελίδα που θα φορτώσουμε
      webAdress.text = [webAdress.text stringByAppendingFormat:@"http://www.google.com"];
```

```
      //Φορτώνουμε τη σελίδα που επιθυμούμε για αρχική
      [webView loadRequest:[NSURLRequest requestWithURL:[NSURL
URLWithString:@"http://www.google.com"]]];
      [NSTimer scheduledTimerWithTimeInterval:1.0 target:self selector:@selector(checkLoad)
userInfo:nil repeats:YES];
      [NSTimer scheduledTimerWithTimeInterval:1.0 target:self selector:@selector(checkNotLoad)
userInfo:nil repeats:YES];
}

//Όση ώρα φορτώνεται η σελίδα ενεργοποιείται το Activity
Indicator (active)
- (void)checkLoad {
      if (webView.loading) {
        [active startAnimating];
      }
}
//Όταν σταματή σει το "φόρτωμα" της σελίδας σταματά ει
το Activity Indicator
- (void)checkNotLoad {
      if (!(webView.loading)) {
        [active stopAnimating];
      }
}

@end
```

4.2.4.3 ScrollingView

```
ScrollingViewController.h

@interface ScrollingViewController : UIViewController
<UIScrollViewDelegate>
{
      IBOutlet UIScrollView* scrollView;
      IBOutlet UIPageControl* pageControl;

    BOOL pageControlIsChangingPage;
}
@property (nonatomic, retain) UIView *scrollView;
@property (nonatomic, retain) UIPageControl* pageControl;

//Ενέργεια με την οποία εναλλάσουμε τις εικόνες
- (IBAction)changePage:(id)sender;

- (void)setupPage;

@end

ScrollingViewController.m

//Φορτώνουμε τις εικόνες και δίνουμε ορισμένες
ιδιότητες στο scroll
- (void)setupPage
```

```
{
      scrollView.delegate = self;

      [self.scrollView setBackgroundColor:[UIColor blackColor]];
      [scrollView setCanCancelContentTouches:NO];

      //Παρακάτω δίνουμε ορισμένες δυνατότητες στο
ScrollView
      scrollView.indicatorStyle = UIScrollViewIndicatorStyleWhite;
      scrollView.clipsToBounds = YES;
      scrollView.scrollEnabled = YES;
      scrollView.pagingEnabled = YES;

      //Παρακάτω ξεκινάμε να φορτώνουμε τις εικόνες που
έχουμε μέσα στο application
      NSUInteger nimages = 0;
      CGFloat cx = 0;
      for (; ; nimages++) {
        NSString *imageName = [NSString stringWithFormat:@"image%d.png", (nimages + 1)];
        UIImage *image = [UIImage imageNamed:imageName];
        if (image == nil) {
              break;
        }
        UIImageView *imageView = [[UIImageView alloc] initWithImage:image];

        CGRect rect = imageView.frame;
        rect.size.height = image.size.height;
        rect.size.width = image.size.width;
        rect.origin.x = ((scrollView.frame.size.width - image.size.width) / 2) + cx;
        rect.origin.y = ((scrollView.frame.size.height - image.size.height) / 2);

        imageView.frame = rect;

        [scrollView addSubview:imageView];
        [imageView release];

        cx += scrollView.frame.size.width;
      }

      self.pageControl.numberOfPages = nimages;
      [scrollView setContentSize:CGSizeMake(cx, [scrollView bounds].size.height)];
}

#pragma mark -
#pragma mark UIScrollViewDelegate stuff

//Κατα τη διάρκεια του scrolling τι συμβαίνει
- (void)scrollViewDidScroll:(UIScrollView *)_scrollView
{
    if (pageControlIsChangingPage) {
        return;
    }
```

```
    //Προκειμένου να γυρίσει η σελίδα όταν έχει
ξεπεραστεί το 50% αυτής
    CGFloat pageWidth = _scrollView.frame.size.width;
    int page = floor((_scrollView.contentOffset.x - pageWidth / 2) / pageWidth) + 1;
    pageControl.currentPage = page;
}
- (IBAction)changePage:(id)sender
{

    CGRect frame = scrollView.frame;
    frame.origin.x = frame.size.width * pageControl.currentPage;
    frame.origin.y = 0;

    [scrollView scrollRectToVisible:frame animated:YES];

     //Για να αλλάξει η σελίδα
     pageControlIsChangingPage = YES;
}
```

4.2.5 RootViewController

RootViewController.h

```
@class FirstDetailViewController;
@class SecondDetailViewController;
@class ThirdDetailViewController;
@class FourthDetailViewController;
@class MultipleDetailViewsWithNavigatorAppDelegate;

@interface RootViewController : UITableViewController {
    FirstDetailViewController *firstDetailViewController;
     SecondDetailViewController *secondDetailViewController;
     ThirdDetailViewController *thirdDetailViewController;
     FourthDetailViewController *fourthDetailViewController;

     MultipleDetailViewsWithNavigatorAppDelegate *appDelegate;
}

@property (nonatomic, retain) FirstDetailViewController *firstDetailViewController;
@property (nonatomic, retain) SecondDetailViewController *secondDetailViewController;
@property (nonatomic, retain) ThirdDetailViewController *thirdDetailViewController;
@property (nonatomic, retain) FourthDetailViewController *fourthDetailViewController;
@property (nonatomic, assign) MultipleDetailViewsWithNavigatorAppDelegate *appDelegate;


@end
```

RootViewController.m

```
- (NSInteger)tableView:(UITableView *)aTableView numberOfRowsInSection:(NSInteger)section {
    //Δηλώνουμε πόσα θα είναι τα στοιχεία του tableView για
να φαίνονται
    return 4;
```

```
}


- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath
*)indexPath {

    static NSString *CellIdentifier = @"CellIdentifier";

    UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:CellIdentifier];
    if (cell == nil) {
        cell = [[[UITableViewCell alloc] initWithStyle:UITableViewCellStyleDefault
reuseIdentifier:CellIdentifier] autorelease];
        cell.accessoryType = UITableViewCellAccessoryNone;
    }

    //Φορτώνουμε μια εικόνα για background
    self.view.backgroundColor = [UIColor colorWithPatternImage:[UIImage
imageNamed:@"math1.jpg"]];

    //Ορίζουμε τα ονόματα τως 4άρων επιλογών μας
    if (indexPath.row == 0) {
        cell.textLabel.text = @"Graph View Calculator";
    }
    else if(indexPath.row == 1){
        cell.textLabel.text = @"Zoom Graphs";
    }
    else if(indexPath.row == 2){
        cell.textLabel.text = @"Simple Calculator";
    }
    else if(indexPath.row == 3){
        cell.textLabel.text = @"Scientific Calculator";
    }
    return cell;
}

#pragma mark -
#pragma mark Table view delegate

- (void)tableView:(UITableView *)aTableView didSelectRowAtIndexPath:(NSIndexPath *)indexPath {

    NSUInteger row = indexPath.row;
    [self.appDelegate.splitViewController viewWillDisappear:YES];
    NSMutableArray *viewControllerArray=[[NSMutableArray alloc]
initWithArray:[[self.appDelegate.splitViewController.viewControllers objectAtIndex:1]
viewControllers]];
    [viewControllerArray removeLastObject];

    //Ανάλογα με τι έχουμε επιλέξει στις επιλογές
δημιουργούμε και καλούμε το ανάλογο ViewController
    if (row == 0) {
      self.firstDetailViewController=[[[FirstDetailViewController alloc] init]autorelease];
      [viewControllerArray addObject:self.firstDetailViewController];
      self.appDelegate.splitViewController.delegate = self.firstDetailViewController;
```

```
    }

    if (row == 1) {
        self.secondDetailViewController=[[[SecondDetailViewController alloc]init]autorelease];
        [viewControllerArray addObject:self.secondDetailViewController];
        self.appDelegate.splitViewController.delegate = self.secondDetailViewController;
    }

    if (row == 2) {
        self.thirdDetailViewController=[[[ThirdDetailViewController alloc]init]autorelease];
        [viewControllerArray addObject:self.thirdDetailViewController];
        self.appDelegate.splitViewController.delegate = self.thirdDetailViewController;
    }

    if (row == 3) {
        self.fourthDetailViewController=[[[FourthDetailViewController alloc]init]autorelease];
        [viewControllerArray addObject:self.fourthDetailViewController];
        self.appDelegate.splitViewController.delegate = self.fourthDetailViewController;
    }

    [[self.appDelegate.splitViewController.viewControllers objectAtIndex:1]
setViewControllers:viewControllerArray animated:NO];


    [self.appDelegate.splitViewController viewWillAppear:YES];
    [viewControllerArray release];

  }
```

4.2.5.1 NewViewController

NewViewController.h

```
//Έχουμε 3 διαφορετικά NewViewController τα οποία τα καλούμε
ανάλογα με τις συναρτήσεις που έχουμε δώσει για
απεικόνιση
//π.χ. αν θέλουμε να απεικονίσουμε 3 διαφορετικές
συναρτήσεις τότε καλούμε το NewViewController
//π.χ. αν θέλουμε να απεικονίσουμε 2 διαφορετικές
συναρτήσεις τότε καλούμε το NewViewController2
//π.χ. αν θέλουμε να απεικονίσουμε 1 διαφορετικές
συναρτήσεις τότε καλούμε το NewViewController3
@interface NewViewController : UIViewController
<GraphViewDelegate,UIScrollViewDelegate,UIImagePickerControllerDelegate,
UINavigationControllerDelegate, UIPopoverControllerDelegate>{

    //Είναι το GraphView όπου σχεδιάζουμε τις γραφικές
παραστάσεις
    IBOutlet GraphView *graphView;
    //Τα παρακάτω 3 είναι τα κουμπία του "Zoom-In", του "Zoom-
```

```
Out" και του "ScreenShot"
    IBOutlet UIButton *Button1;
    IBOutlet UIButton *Button2;
    IBOutlet UIButton *Button3;
    //Τα παρακάτω NSMutableArray τα χρειαζόμαστε για να
αποθηκεύσουμε τις συναρτήσεις που μας δόθηκαν από το
χρήστη
    NSMutableArray *savedExpression;
    NSMutableArray *savedExpression2;
    NSMutableArray *savedExpression3;

    //Το scale το χρησιμοποιούμε για να ρυμθίζουμε τη
γραφική παράσταση και την αρίθμηση των αξόνων όταν θα
κάνουμε zoom
    //Ουσιαστικά ξαναϋπολογίζουμε και σχεδιάζουμε τη
γραφικά παράσταση και τους άξονες κάθε φορα που
κάνουμε zoom-in ή zoom-out
    CGFloat  scale;

    //Είναι εκεί που φορτώνεται η εικόνα που
επιλέγουμε από το popover
    UIImageView *imageView;

    UIToolbar *toolbar;
   UIPopoverController *popoverController;

    //Χρειαζόμαστε έ να αντίγραφο του FirstDetailViewController
προκειμένου να μπορούμε να πάρουμε τα strings που
βρίσκονται μέσα στα display
    //και αντιστοιχούν στις συναρτήσεις προς
απεικόνιση
    FirstDetailViewController *firstController;
    UITextField *textField;

    UITextField *function1;
    UITextField *function2;
    UITextField *function3;
}

@property CGFloat scale;
@property(retain, nonatomic) GraphView *graphView;
@property(retain, nonatomic) NSArray *savedExpression;
@property(retain, nonatomic) NSArray *savedExpression2;
@property(retain, nonatomic) NSArray *savedExpression3;
//Δίνει ένέργεια στα κουμπιά "Zoom-In" και "Zoom-Out"
-(IBAction)zoomPressed:(UIButton *)sender;
//Δίνει ενέργεια στο κουμπί "Screenshot" για να αποθηκεύσει
τη γραφική παράσταση
-(IBAction)pushScreenshot;

@property (nonatomic, retain) IBOutlet UIImageView *imageView;

@property (nonatomic, retain) UIPopoverController *popoverController;
```

```
@property (nonatomic, retain) IBOutlet UIToolbar *toolbar;
- (IBAction)useCameraRoll: (id)sender;

@property (retain, nonatomic) FirstDetailViewController *firstController;
@property (retain, nonatomic) IBOutlet UITextField *textField;

@property (retain, nonatomic) IBOutlet UITextField *function1;
@property (retain, nonatomic) IBOutlet UITextField *function2;
@property (retain, nonatomic) IBOutlet UITextField *function3;

- (IBAction)textFieldReturn:(id)sender;
- (IBAction)backgroundTouched:(id)sender;

@end
```

NewViewController.m

```
- (void)viewDidLoad{

    //Το string με τις συναρτήσεις που γίνανε γραφικές παραστάσεις το παίρνουμε μέσω του "τίτλου" του παραθύρου
    //και εν συνεχεία αλλάζουμε τον "τίτλο" του παραθύρου σε "Graph"
    //Επίσης ορίζουμε χρώμα στη γραμματοσειρά
    textField.text = self.title;
    textField.textColor = [UIColor blackColor];

    graphView.newScale = 14.0;

    [super viewDidLoad];
    [self setTitle:@"Graph"];

    self.graphView.delegate = self;
    [self updateUI];

    UIBarButtonItem *cameraRollButton = [[UIBarButtonItem alloc]
                                         initWithTitle:@"Camera Roll"

style:UIBarButtonItemStyleBordered
                                         target:self

action:@selector(useCameraRoll:)];
    NSArray *items = [NSArray arrayWithObjects: cameraRollButton, nil];
    [toolbar setItems:items animated:NO];
    [cameraRollButton release];
    [super viewDidLoad];

     NSLog(@"textField NEW %@",savedExpression);

     function1.textColor = [UIColor redColor];
     NSString *result=@"1st Function";
     function1.text=result;
```

```
    function2.textColor = [UIColor greenColor];
    result=@"2nd Function";
    function2.text=result;

    function3.textColor = [UIColor orangeColor];
    result=@"3rd Function";
    function3.text=result;

}

-(IBAction)zoomPressed:(UIButton *)sender{

    //Ανάλογα με ποιό κουμπί έχει πατήσει ο χρήστης
(αυτό το καθορίζει ο sender) αποφασίζουμε αν θα κάνουμε
zoom-in ή  zoom-out
    NSString *zoom = sender.titleLabel.text;
    //Για να μη γίνεται επ΄αόριστα zoom-in αποφασίζουμε το
scale να επανέρχεται στα αρχικά επίπεδα
    if (scale <= 0)
    {
      scale = 14.0;
    }
    if ([@"Zoom In" isEqualToString:zoom])
    {
      self.scale = scale + 1.0;
    }
    else {
      self.scale =scale - 1.0;
    }

    NSLog(@"Zoom button pressed %@ = %f",zoom,scale);
    if (scale <= 0) {
      scale = 14.0;
    }
    graphView.newScale = scale;
    [self updateUI];
    return;
}

-(IBAction)pushScreenshot{
    //Για να μην εμφανίζονται τα buttons όταν κάνουμε
screenshot
    Button1.alpha = 0;
    Button2.alpha = 0;
    Button3.alpha = 0;

    //Παρακάτω φαίνεται πως αποθηκεύουμε τη
φωτογραφία στο album των φωτογραφιών
    UIGraphicsBeginImageContext(graphView.bounds.size);
    [self.view.layer renderInContext:UIGraphicsGetCurrentContext()];
    UIImage *viewImage = UIGraphicsGetImageFromCurrentImageContext();
    UIGraphicsEndPDFContext();
```

```
    UIImageWriteToSavedPhotosAlbum(viewImage, nil, nil, nil);

    Button1.alpha = 1;
    Button2.alpha = 1;
    Button3.alpha = 1;

}

-(float)yDataToFirstGraph:(GraphView *)requestor withThisX:(float)xValue;{
    //Καλεί την evaluateExpression που βρίσκεται στο CalculatorBrain
    float newY = [CalculatorBrain evaluateExpression:savedExpression
usingVariableValue:xValue];
    //NSLog(@"ena kai ena");

    return newY;
}

//Σε περίπτωση που δε γίνει σωστό "φόρτωμα" της
φωτογραφίας
-(void)image:(UIImage *)image
finishedSavingWithError:(NSError *)error
 contextInfo:(void *)contextInfo{
    if (error) {
        UIAlertView *alert = [[UIAlertView alloc]
                              initWithTitle: @"Save failed"
                              message: @"Failed to save image"
                              delegate: nil
                              cancelButtonTitle:@"OK"
                              otherButtonTitles:nil];
        [alert show];
        [alert release];
    }
}
```

5.Παρατηρήσεις

1) *Τσακμαλής Ανέστης, Ηλεκτρολόγος Μηχανικός (Εργαζόμενος Μηχανικός στον Ιδιωτικό τομέα)*

Το Calculator διαθέτει χαρακτηριστικά προσαρμοσμένα στη φιλοσοφία ενός tablet. Απευθύνεται στον απλό χρήστη και προσπαθεί να του προσφέρει απευθείας μαθηματικές εκφράσεις χωρίς να χρειάζεται να πληκτρολογήσει τους συμβολισμούς τους. Τα διάφορα εργαλεία του (Graph View, Simple Calc, Zoom Graph και Scientific Calc) είναι διαχωρισμένα ώστε να έχει ο χρήστης διαθέσιμες στην οθόνη του περισσότερες λειτουργίες. Τα tutorials είναι αρκετά απλοποιημένα και ευανάγνωστα και κάνουν το πρόγραμμα easy to use ακόμα και στο μην έχοντα γνώση προγραμματισμού. Οι εντυπώσεις μου είναι σε γενικές γραμμές θετικές και πιστεύω ότι θα μπορούσε να αντικαταστήσει πολλά από τα ήδη υπάρχοντα scientific calculators της αγοράς βοηθώντας μαθητές, φοιτητές, επιστήμονες και επαγγελματίες.

2)Τσελίγκα Ειρήνη, Απόφοιτος Φαρμακευτικής Σχολής

Το επιστημονικό κουμπιουτεράκι που χρησιμοποίσα είναι αρκετά οικείο και εύχρηστο για έναν

καινούργιο χρήστη χωρίς καθόλου πείρα σε tablets είτε σε smart phones. Τα Tutorials παρέχουν τις κατάλληλες πληροφορίες για τη σωστή χρήση της εφαρμογής καθώς και για τις δυνατότητές της. Ιδιαίτερη εντύπωση μου έκανε το πλήθος των συναρτήσεων που μπορεί να χρησιμοποιήσει οποιοσδήποτε χρήστης μέσω του Scientific Calc. Η γενική εικόνα είναι ιδιαίτερα καλή αν και ορισμένες επιπλέον προσθήκες θα έκαναν την εφαρμογή πιο ανταγωνιστική. Η γνώμη μου είναι πως οι δυνατότητες τους θα είναι σημαντικές για όσους εμπιστευτούν αυτό το επιστημονικό κουμπιουτεράκι.

3)Γιάννης Κουνάκης, Ηλεκτρολόγος Μηχανικός (Μεταπτυχιακός Φοιτητής)

Το Calculator είναι μία εφαρμογή για iPad που προσφέρει στο χρήστη όλα τα χαρακτηριστικά ενός scientific calculator. Η χρήση του είναι πολύ απλή και σε αυτό βοηθάνε τα tutorials που δείχνουν βήμα προς βήμα τί πρέπει να κάνει ο χρήστης. Έχει διάφορα sections Simple Calc, Graph View, Zoom Graph και Scientific Calc ώστε ο χρήστης να μπορεί να περιηγείται άνετα σε όλες τις διαθέσιμες λειτουργίες του και χωρίς να χρειάζεται να ψάχνει σε μενού ή περίεργους συνδυασμούς στο πληκτρολόγιο για να γράψει αυτό που θέλει. Αυτό είναι που μου έκανε μεγαλύτερη εντύπωση στην εφαρμογή αυτή, μιας και όλοι ξέρουμε πόσο δύσκολο είναι να βρεις και να γράψεις αυτό που θέλεις σε ένα scientific calculator. Είναι ένα πολύ καλό, ξεκάθαρο και απλό εργαλείο που θα βοηθήσει όποιον ασχολείται με υπολογισμό μαθηματικών εκφράσεων, απλών και εξεζητημένων. Θα εξοικονομήσει περισσότερο χρόνο για το χρήστη του και σίγουρα θα χρησιμοποιήσει το Scientific Calc για περίπλοκες μαθηματικές εκφράσεις.

6.Σύνδεσμοι

1.http://www.icodeblog.com/

2.http://www.reecefowell.com/

3.http://www.bogotobogo.com/

4.file:///usr/local/share/giac/doc/local/cascmd_en/index.html#htoc1

5.http://www.cimgf.com/

6.https://developer.apple.com/

7.http://www.iphonehellas.gr/

8.http://programming4.us/

9.http://mobiforge.com/Designing/article-listing

10.http://www.techotopia.com

11.http://www.xprogress.com/

12.http://stackoverflow.com/

13.http://www.stanford.edu/class/cs193p/cgi-bin/drupal/