Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Πτυχιακή εργασία

Θέμα:

"Ανάπτυξη επιτραπέζιου παιγνίου σε περιβάλλον iOS"
"Development of Board Game in iOS Platform".

Φοιτητής: Παντελής Γιαζιτσής

ΓΕΝΙΚΑ

Η εφαρμογή αυτή δημιουργήθηκε στα πλαίσια της πτυχιακής μου εργασίας στο τμήμα Μηχανικών Ηλεκτρονικών Υπολογιστών Τηλεπικοινωνιών και Δικτύων του Πανεπιστημίου Θεσσαλίας. Πρόκειται για την μεταφορά ενός επιτραπέζιου παιχνιδιού σε συσκευή iPad.

Η ανάπτυξη της εφαρμογής έγινε σε λειτουργικό MAC OS X (10.9, Mavericks). Το IDE που χρησιμοποιήθηκε είναι το XCODE 5.0.

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω τον επιβλέπων καθηγητή μου κ. Αλκιβιάδη Ακρίτα για την βοήθειά του και για το ερέθισμα που μου έδωσε ώστε να ασχοληθώ με τον τομέα του mobile development.
Επίσης τον designer Χρίστο Χριστοδουλόπουλο για την σχεδίαση των γραφικών της εφαρμογής και την πολύτιμη βοήθεια του.

Τέλος την οικογένεια μου που με στήριξε καθ’ όλη την διάρκεια των σπουδών μου.

Παντελής Γιαζιτσής

Περιεχόμενα

1. Εισαγωγή και ιστορικά στοιχεία.

Τα κινητά τηλέφωνα έχουν γίνει αναπόσπαστο κομμάτι της καθημερινής μας ζωής. Ο καθένας πλέον, από ένα μικρό παιδί μέχρι έναν ηλικιωμένο, έχει το δικό του κινητό. Ωστόσο τα κινητά τηλέφωνα που βλέπουμε τώρα δεν έμοιαζαν τόσο πολύ με τα παλαιότερα μοντέλα.

Η αλματώδης εξέλιξη της τεχνολογίας έφερε αυτήν την τεράστια αλλαγή μετατρέποντας τα τεράστια κινητά της δεκαετίας του ’90 σε αυτά που χρησιμοποιούμαι τώρα.

1.1 Smartphones και tablets

Το smartphone είναι πλέον ένας ισχυρός κινητός υπολογιστής. Σε συνδυασμό με τα tablets έχουν δημιουργήσει μια καινούρια καταναλωτική συνήθεια και προσδοκίες που αφορούν την εκμετάλευση πληροφοριών και την αγορά αγαθών και υπηρεσιών. Τεράστιο ρόλο σε αυτήν την εξέλιξη έπαιξε η αναβάθμιση των υπηρεσιών διαδικτύου καθώς και η ανάπτυξη των social networks και ιδιαίτερα του facebook.

Η πραγματική “επανάσταση” των smartphones ξεκίνησε με την παρουσίαση του πρώτου iPhone από την apple το 2007. Το iPhone επαναπροσδιόρισε την έννοια της κινητης συσκευής σε μια συσκευή με γρήγορη πρόσβαση στο internet και με πρωτοποριακή οθόνη αφής. Επίσης δημιούργησε και προώθησε την ιδέα ενός strore με mobile εφαρμογές.

Το 2008 η google παρουσίασε ένα εναλλακτικό λειτουργικό σύστημα για mobile συσκευές, το Android. Πρόκειται για ένα δωρεάν λογισμικό ανοιχτού κώδικα που

δίνει την δυνατότητα σε κάθε κατασκευαστή να το τροποποιήσει ώστε να γίνει κατάλληλο για την συσκευή του. Γενικά το android έχει γνωρίσει μεγάλη επιτυχία τα τελευταία χρόνια καθώς είναι εγκαταστημένο σε πάνω απο το 60% των smartphones παγκοσμίως.

Το iPhone οδήγησε την apple στο να σχεδιάσει και να παρουσιάσει το 2010 το tablet που άλλαξε τα δεδομένα στον χώρο, το ipad. Οι περισσοτεροι θεωρούν ότι η ιστορία του tablet ξεκινάει το 2010. Ωστόσο αυτό απέχει πολύ από την πραγματικότητα. Η πρώτη προσπάθεια είχε γίνει την δεκαετία του ’ 50 με την χρησιμοποίηση αναγνώρισης γραφής από το χέρι αντί για το πληκρολόγιο. Το 1999 η Microsoft δημιούργησε ένα tablet με στόχο να ανταγωνιστεί τα laptops. Αυτή η προσπαθεια όμωσς απέτυχε για τον βασικό λόγο του ότι δεν κατάφεραν να το κάνουν αρκετά μικρό. Η Apple αντιλήφθηκε ότι έπρεπε να προσπαθήσει να κάνει ενα μεγαλύτερο smartphone από το να προσπαθεί να κάνει έναν υπολογιστή μικρότερο. Το αποτέλεσμα ήταν το iPad και το επιτυχημένο μέλλον του.

1.2 Mobile apps

Η επικοινωνία στην καθημερινότητα μας έχει φτάσει σε τέτοιο σημείο που πολλοί άνθρωποι δεν μπορούν ούτε λεπτό χωρίς το κινητό τους. Το smartphone ειναι πλέον μια συσκευή μέσω της οποία μπορείς να μελετήσεις, να ενημερωθείς και να διασκεδάσεις.

Οι εφαρμόγες στα τέλη του 20ού αιώνα ήταν κάποια απλά παιχνίδια, επξεργαστές ήχων κλήσης, αριθμπομηχανές, ημερολόγια κλπ. Οι κατασκευαστές ήθελαν να κάνουν τις εφαρμογές τους πιο ελκυστικές. Ωστόσο οι προγραμματιστές ήθελαν όλο και περισσότερη ελευθερία για να προγραμματίσουν αυτό που είχαν στο μυαλό τους, πράγμα που δεν είχαν από τους κατασκευαστές. Αυτό οδήγησε στην δημιουργία εφαρμογών μόνο απο την ομάδα προγραμματιστών του κάθε καταστεκευαστή. Ενα κλασικό παράδειγμα είναι η Nokia που ενσωμάτωσε το παιχνίδι snake (φιδάκι) στα κινητά της αρκετα χρόνια πριν.

Οι πελάτες με την σειρά τους απαιτούσαν όλο και περισσότερες δυνατότητες και περισσότερα παιχνίδια από τις συσκευές τους. Κάπως έτσι οι κατασκευαστές συνειδητοποίησαν την ανάγκη να τροποποιήσουν την “πολιτική προστασίας” τους και να δώσουν χώρο για δημιουργία και σε προγραμματιστες εκτός του κύκλου της εταιρίας τους. Πλέον οι προγραμματιστές mobile εφαρμογών μπορούν να προγραμματίζουν σε διαφορετικά

περιβάλλοντα, με διαφορετικές γλώσσες και διαφορετικά εργαλεία.

1.3 iOS

Το iOS είναι το λειτουργικό σύστημα που τρέχει σε όλα τα iPhone. Από την μέρα της παρουσίασης του μέχρι σήμερα έχει περάσει από αρκετές φασεις και εκδόσεις. Το πιο συναρπαστίκο πράγμα που μπορούσε να κάνει ο χρήστης με το πρώτο iOS ήταν να παρακολουθεί βίντεο στο youtube και να παίζει web games. Η σημερινη πραγματικότητα απέχει πολύ από αυτό.

iOS 1.0: Στην παρουσίαση του iPhone στης 29 Ιουνίου του 2007 ο Steve Jobs παρουσίασε το λειτουργικό σαν παραλλαγή του OS X. Η Apple ονόμασε το λειτουγικό αρχικά iPhone OS (άλλαξε μετά από κάποιυς μήνες σε iOS. Αν και το πρώτο iOS ήταν πραγματικά πρωτοποριάκο είναι πιο πολύ γνωστό για τα χαρακτηριστικά που δεν παρείχε όπως το App store, το iTunes κλπ. Αρχικά δεν υποστηρίχθηκαν οι εφαρμογές τρίτων. Ο Steve Jobs ενθάρρυνε τους προγραμματιστές να αναπτύξουν WEB εφαρμογές που θα συμπεριφέρονται σαν “native”. Ωστόσο η Apple άλλαξε σύντομα την γνώμη της παρέχοντας το πρώτο ios SDK τον Μάρτιο του 2008.

iOS 2.0: Το σημαντικότερο νέο για το iOS 2.0 ήταν το App store. Πλέον οι προγραμματιστές μπορούν να ανεβάσουν την δικιά τους εφαρμογή. Μετά από λίγο καιρό έρχεται η περίοδος που άκουγε κάποιος πολύ συχνά την φράση "Υπάρχει μια εφαρμογή για αυτό…"!

iOS 3.0: Παρουσιάστηκε τον Ιούνιο του 2009. Συμπεριελάμβανε την δυνατότητα του copy and paste καθώς επίση και την μαγνητική πυξίδα στους χάρτες.

iOS 3.2: Δεν πρόκρειται για ένα απλό update. Το ios 3.2 έφερε το λειτουργικό στο ipad με την δυνατότητα να αλλάζει η απεικόνιση της εικόνας από portrait σε landscape.

iOS 4.0: Ένα από τα μεγαλύτερα χαρακτηρηστικά του ios 4.0 ήταν η δυνατότητα συνομιλίας με video μέσω της εφαρμογής Face Time. Επίσης η Apple παρουσίασε και την εφαρμογή iBooks καθώς και την λειτουργία του multitasking που δίνει την δυνατότητα στον χρήστη να μεταφέρεται ανάμεσα στις ανοιχτες εφαρμογές του.

iOS 5.0: Το 2011 η Apple χρησιμοποίησε μερικά σημαντικά "όπλα" για να προωθήσει το iOS στον πόλεμο των λειτουργικών συστημάτων smartphone. Η προσθήκη πάνω από 200 νέων χαρακτηριστικών έκανε το iOS 5 την μεγαλύτερη αναβάθμιση μέχρι εκείνη την στιγμή.

iOS 6.0: Το iOS 6.0 χαρακτήριστηκε με μερικές από τις μεγαλύτερες αλλαγές που έχει εφαρμόσει η Apple στο iPhone.Η Apple αποφάσισε να αντικαταστήσει τους χάρτες της Google με δικούς της που παρέχουν την δυνατότητα της τρισδιάστατης απεικόνισης. Η αρχή ωστόσο των χαρτών της Apple δεν ήταν τόσο καλή. Το App Store επίσης ξανασχεδιάστηκε εξ' ολοκλήρου. Επίσης και το Facebook ενσωματώθηκε στις επαφές του χρήστη.

iOS 7.0: Στο iOS 7.0 έχουμε την μεγαλύτερη επανασχεδίαση που έγινε ποτέ απο τότε που πρωτοκυκλοφόρησε το λετουργικο.

2. iOS SDK

2.1 Η Objective-C

Η objective c είναι μια αντικειμενοστρεφής γλώσσα προγραμματισμού που προσθέτει το Smalltalk-style messaging* στην γλώσσα προγραμματισμού C . Είναι η βασική γλώσσα προγραμματισμού που χρησιμοποιείται από τη Apple για το OS X και τα iOS λειτουργικά συστήματα και τα αντίστοιχά τους APIs και frameworks.

Αρχικά αναπτύχθηκε στις αρχές της δεκαετίας του 1980, επιλέχτηκε ως βασική γλώσσα για να χρησιμοποιηθεί από την NeXt για το λειτουργικό σύστημα NeXTSTEP, από το οποίο προέρχονται το OS X και το iOS.

Ο κώδικας της objective-c έχει συνήθως σαν επέκταση των αρχείων την κατάληξη .m, ενώ τα header files έχουν την κατάληξη .h.

Η objective-C είναι ένα υπερσύνολο της C. Είναι δυνατό να μεταγλωττίσεί κάποιος οποιοδήποτε πρόγραμμα της C με έναν μεταγλωττιστή της objective-C. Επίσης να συμπεριληφθεί κώδικας της C μέσα σε μια κλάσση της objective-C.

Το μοντέλο της objective-C είναι βασισμένο στο πέρασμα μηνυμάτων σε αντικείμενα. Στην objective-C δεν καλείται απλά μια μέθοδος αλλά στέλνει ένα μήνυμα.

Ο στόχος ενός μηνύματος επιλύεται στο χρόνο εκτέλεσης με το ίδιο το λαμβάνον αντικείμενο που ερμηνεύει το μήνυμα.

Μια συνέπεια αυτού είναι ότι το message-passing σύστημα δεν έχει κανέναν έλεγχο τύπων. Το αντικείμενο στο οποίο κατευθύνεται το μήνυμα δεν είναι εγγυημένο ότι θα αποκριθεί σε αυτό, και εάν όχι, δημιουργείται απλά μια εξαίρεση.

Η αποστολή των μηνυμάτων στο αντικείμενο έχει την

* Ο τρόπος κλήσης μεθόδων στέλνοντας μηνύματα στα αντικέιμενα

μορφή του ακόλουθου κώδικα:
[obj method:argument];

Interfaces and implementations

Η objective-C απαιτεί τον ορισμό του interface μιας κλάσης και του implementation σε ξεχωριστά blocks. Συνήθως οι προγραμματιστές προτιμούν την τοποθέτιση του interface σε ένα αρχείο επικεφαλίδων (header file) και το implementation σε ένα αρχείο κώδικα (.m). Τα αρχεία επικεφαλίδων είναι παρόμοια με τα αρχεία επικεφαλίδων της C ενώ τα αρχεία implementation (.m) είναι παρόμοια με τα αρχεία κώδικα της C.

Interface

Το interface μιας κλάσης καθορίζεται σε ένα αρχείο επικεφαλίδων όπως παρακάτω:

```
@interface classname : superclassname {
 // instance variables
}
+ classMethod1;
+ (return_type)classMethod2;
+
(return_type)classMethod3:(param1_type)param1_varName;

-
(return_type)instanceMethod1With1Parameter:(param1_type
)param1_varName;
-
(return_type)instanceMethod2With2Parameters:(param1_typ
e)param1_varName
param2_callName:(param2_type)param2_varName;
@end
```

Παραπάνω, τα συν (+) δείχνουν σε μεθόδους κλάσσης, ή σε μεθόδους που μπορούν να κληθούν από την ίδια την κλάση (όχι στα πλαίσια ενός στιγμιοτύπου) και τα μείον (-) δείχνουν σε μεθόδους στιγμιοτύπων και μπορούν να κληθούν απο στιγμιότυπο της κλάσσης. Οι μέθοδοι κλάσσης δεν έχουν επίσης καμία πρόσβαση στις μεταβλητές στιγμιοτύπου (instance variables).

Τα ορίσματα μιας μεθόδου αρχίζουν με μια άνω και κάτω τελεία που ακολουθείται από τον τύπο του ορίσματος σε παρένθεση και το όνομα του.

Για παράδειγμα:

```
- (void)setRangeStart:(int)start end:(int)end;
- (void)importDocumentWithName:(NSString *)name
withSpecifiedPreferences:
(Preferences *)prefs beforePage:(int)insertPage;
```

Implementation

Στο interface δηλώνεται μόνο η διεπαφή των μεθόδων. Ο κώδικας γράφεται στο αρχείο implementation (.m).

```
@implementation classname
+ (return_type)classMethod
{
 // implementation
}
- (return_type)instanceMethod
{
 // implementation
}
@end
```

Instantiation (δημιουργία στιγμιοτύπου κλάσσης)

Γίνεται σε δύο φασεις. Αρχικά δευσμεύεται η μνήμη για το στιγμιότυπο και μετά η διαμόρφωση του. Ένα αντικείμενο δεν είναι πλήρως - λειτουργικό αν δεν ολοκληρωθούν και τα δύο βήματα. Αυτά τα βήματα πρέπει να ολοκληρωθούν με μια γραμμή κώδικα έτσι ώστε να μην υπάρχει ποτέ ένα δεσμευμένο αντικείμενο που δεν είναι διαμορφωμένο.

Για παράδειγμα:

```
MyObject *o = [[MyObject alloc] init];
```

Το μήνυμα init εκτελεί την διαμόρφωση του στιγμιοτύπου μετά τη δημιουργία. Η μέθοδος init αναπτύσεται συχνά ως εξής:

```
- (id)init {
    self = [super init];
    if (self) {
        // perform initialization of object here
    }
    return self;
}
```

Στο παραπάνω παράδειγμα, παρατηρούμε οτι επιστρέφεται το αντικείμενο id. Αυτός ο τύπος αντικειμένου αντιπροσωπεύει «το δείκτη σε οποιοδήποτε αντικείμενο» στην objective-C.

Αυτό το πρότυπο constructor χρησιμοποιείται για να εξασφαλίστει ότι το αντικείμενο αρχικοποιείται κατάλληλα από την superclass του προτού εκτελέστει η μέθοδος init. Εκτελούνται οι ακόλουθες ενέργειες:

1. **self = [super init]** στέλνει το init μήνυμα στην

superclass του στιγμιοτύπου και αναθέτει το αποτέλεσμα στον δείκτη self.

2. **if (self)** ελέγχει αν το αντικείμενο δημιουργήθηκε σωστά
3. **return self** επιστρέφει την τιμή του self στον caller.

Protocols

Το πρωτόκολλο είναι μια λίστα μεθόδων που μπορεί μια κλάση να επιλέξει αν αναπτύξει ή όχι.Τα πρωτόκολλα περιλαμβάνουν συχνά τις προαιρετικές μεθόδους, οι οποίες, εάν εφαρμόζονται, μπορούν να αλλάξουν τη συμπεριφορά μιας κλάσης. Συνήθως αναπτύσονται από άλλες κλάσεις.

Για παράδειγμα:

```
@protocol NSLocking
- (void)lock;
- (void)unlock;
@end
```

Δηλώνεται στο interface της κλάσης ότι το πρωτόκολλο θα εφαρμοστεί και θα αναπτυχθούν οι μεθόδοι του.

```
@interface NSLock : NSObject <NSLocking>
//...
@end
```

2.2 ARC memory management

Στην objective-C το ARC (**Automatic Reference Counting**) είναι ο τρόπος διαχείρισης της μνήμης μέσω του οποίου παρακολουθείται το πλήθος των αναφορών σε κάθε αντικείμενο που δημιούργησε ο compiler.

Παλαιότερα στην objective-C ο προγραμματιστής θα έπρεπε να δεσμεύει και να απελευθερώνει αντικείμενα μόνος του.

Στο ARC ο compiler το κάνει αυτό

Η Apple ενσωμάτωσε το ARC το 2011 για την ανάπτυξη εφαρμογών στο λειτουργικό σύστημα Lion στο Mac OS X και στο iOS 5.

2.3 Xcode

Το Xcode είναι ένα ενσωματωμένο περιβάλλον ανάπτυξης (IDE) που περιέχει μια σειρά εργαλείων ανάπτυξης λογισμικού που προσφέρονται από την Apple για την ανάπτυξη του λογισμικού για το OS X και το iOS. Πρώτα εμφανίστηκε το 2003, η πιό πρόσφατη έκδοση είναι η έκδοση 5.0 και είναι διαθέσιμη μέσω του Mac App Store δωρεάν.

Το Xcode περιλαμβάνει επίσης το μεγαλύτερο μέρος του **Apple's developer documentation**, και τον ενσωματωμένο interface builder, μια εφαρμογή που δημιουργεί το γραφικό περιβάλλον του χρήστη.

2.4 Ιεραρχία κλάσσεων και μερικές απο αυτές

Το σημαντικότερο framework για την ανάπτυξη των εφαρμογών είναι το Cocoa touch. Το **Cocoa Touch** είναι framework για τη δημιουργία της διεπαφής του χρήστη για τα προγράμματα που τρέχουν σε λετουργικό iOS (για iPhone, iPod Touch, και iPad).

Το Cocoa Touch παρέχει με την σειρά του κάποια πολύ σημαντικά frameworks για την ανάπτυξη εφαρμογών σε iOS. Κάποια από αυτά είναι το Foundation Kit Framework και το UIKit Framework.

- UIkit

Το UIKit παρέχει τις κλάσεις που απαιτούνται για να κατασκευάστεί και να διαχειριστεί η διεπαφή χρήστη μιας εφαρμογής για iOS. Παρέχει μια σειρά από αντικείμενα που είναι σχεδιασμένα για την διαχείριση της διεπαφής χρήστη σε οθόνες αφής.

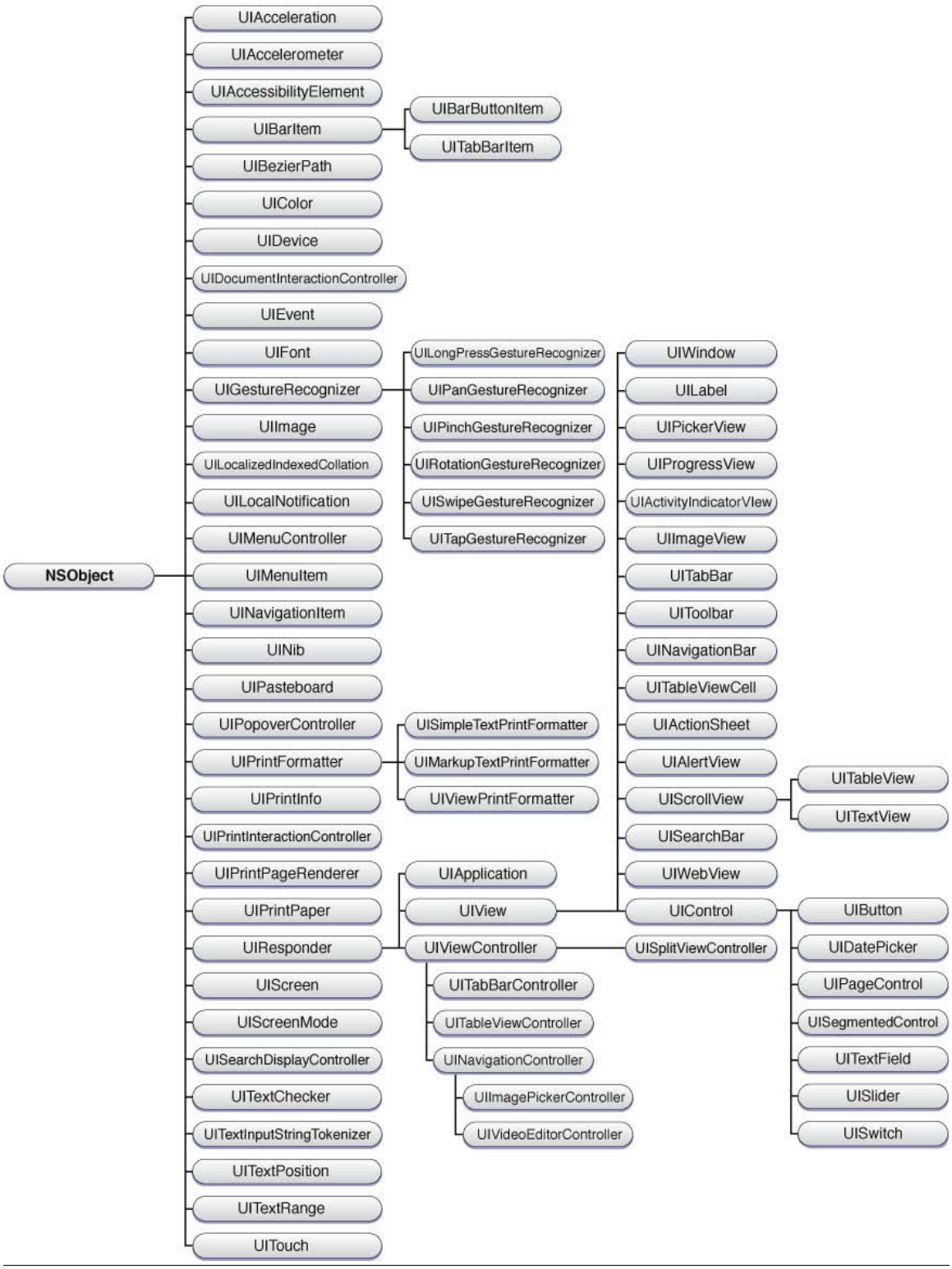


Figure 1: Ιεραρχία κλάσεων του UIkit

Κάποιες απο αυτές τις κλάσεις είναι:

• **NSObject** είναι η “root class” από τις περισσότερες ιεραρχίες κλάσεων της Objective-C. Μέσω του NSObject, τα αντικείμενα κληρονομούν την ικανότητα να συμπεριφέρονται σαν αντικείμενα της Objective-C.
• Η κλάση **UIViewController** παρέχει το βασικό μοντέλο διαχείρισης εικόνας για όλες της iOS εφαρμογές. Ένας ViewController διαχειρίζεται ένα σύνολο views που αποτελούν την διεπαφή της εφαρμογής με τον χρήστη.
• Η κλάση **UIView** καθορίζει μια ορθογώνια περιοχή στην οθόνη και παρέχει την δυνατότητα της διαχείρισης του περιεχομένου της συγκεκριμένης περιοχής.
• Το **UIWindow** είναι ένα αντικείμενο γνωστό ως παράθυρο που διαχειρίζεται και συντονίζει τις οθόνες που παρουσιάζει η εφαρμογή. Αν η εφαρμογη δεν παρουσιάζει το περιεχόμενο της σε μια εξωτερική οθόνη, έχει μόνο ένα παράθυρο.
• Ένα **UITouch** αντικείμενο διαχειρίζεται το γεγονός της μετακίνησης ενός δάχτυλου στην οθόνη.
• Ένα **UIScreen** αντικείμενο περιέχει το ορθογώνιο ολόκληρης της οθόνης της συσκευής.
• Μία **UIDevice** κλάση παρέχει στιγμιότυπο που αντιπροσωπεύει την παρούσα συσκευή. Από αυτό μπορούμε να πάρουμε πληροφορίες για τη συσκευή όπως το όνομα της, το μοντέλο, την έκδοση λειτουργικού κ.α.

• Foundation Kit Framework

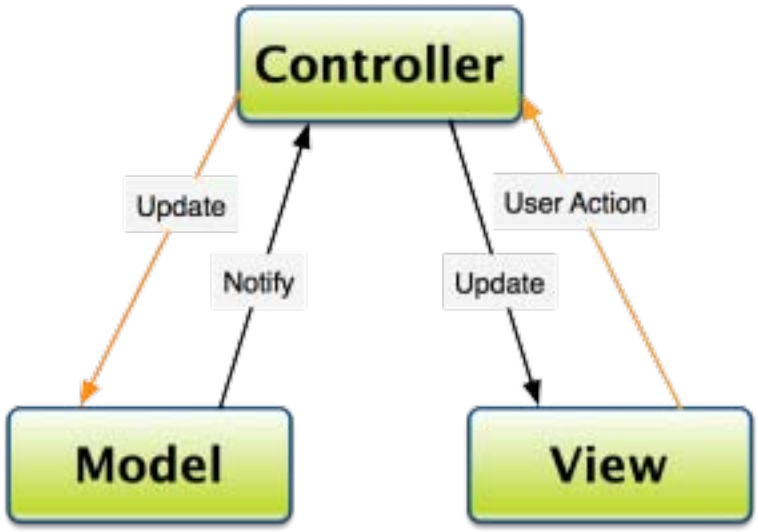
Παρέχει τις βασικές κλάσεις όπως οι κλάσεις δομών δεδομένων.

• Η **NSMutableArray** κλάσση δηλώνει την προγραμματική διεπαφή σε αντικείμενο που διαχειρίζεται μια σειρά αντικειμένων.
• Η **NSDictionary** κλάσση δηλώνει την προγραμματική διεπαφή σε αντικείμενο που διαχειρίζεται μια σειρά κλειδιών και τιμών.
• Το **NSNotification** αντικείμενο τοποθετεί πληροφορίες που μπορούν να γίνουν “broadcast” σε άλλα αντικείμενα από το NSNotificationCenter.

2.5 Μοντέλο σχεδίασης MVC στο iOS

Οι iOS εφαρμογές ακολουθούν το Model View Controller πρότυπο σχεδίασης.

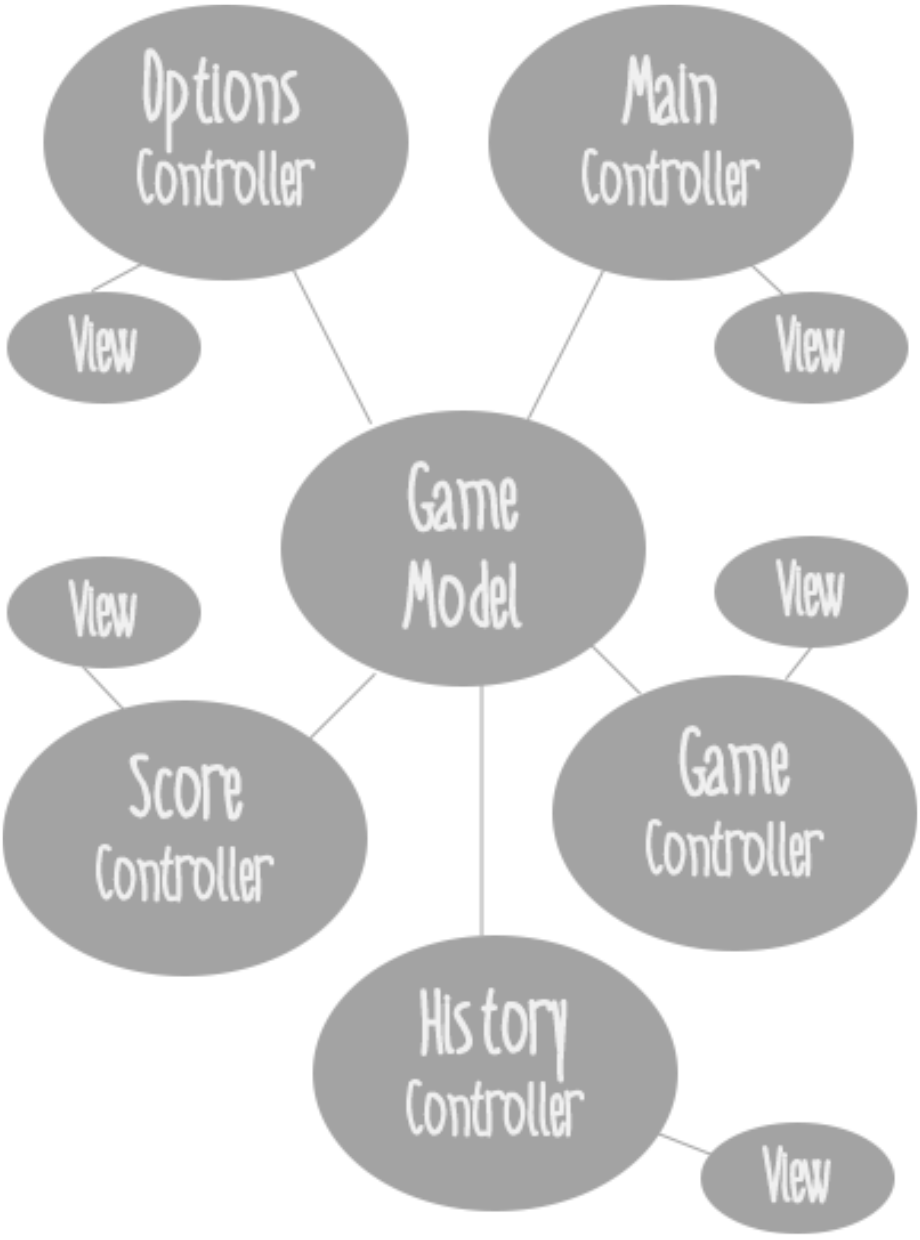
• Model: Είναι υπεύθυνο για την λογική της εφαρμογής
• View: Είναι υπεύθυνο για αυτό που βλέπει ο χρήστης
• Controller: Λειτουργεί ως μεσολαβητής μεταξύ του model και του view. Δεν πρέπει να υπάρξει οποιαδήποτε άμεση συνομιλία μεταξύ model και view. Ο controller ενημερώνει το view για αλλαγές βάσει του model. Εάν ο χρήστης αλληλεπιδράσει με το view ο controller είναι αυτός που



Μοντέλο MVC

ενημερώνει το model.

Ένα model μπορεί να αλληλεπιδρά με παραπάνω από έναν controller. Κά τέτοιο συμβαίνει και σε αυτήν την εφαρμογή.



Το σχέδιο της εφαρμογής

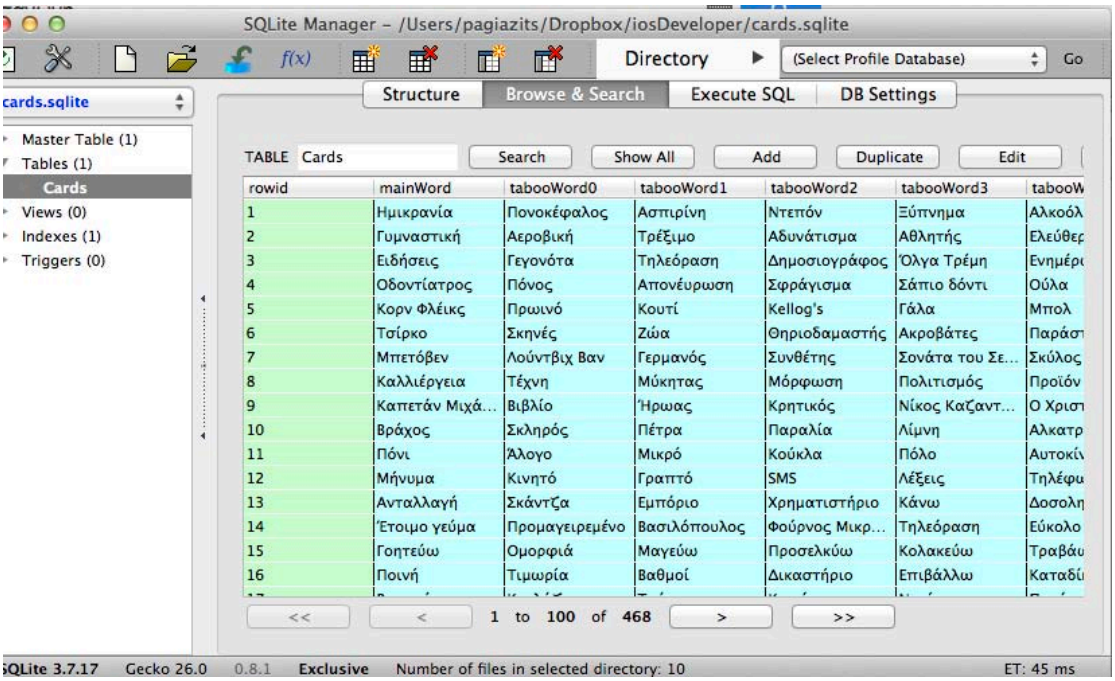
3. Υλοποίηση

Η εφαρμογή είναι η μεταφορά ενός επιτραπέζιου παιχνιδιού στο iPad. Ένας αριθμός παικτών από 4 και πάνω κάθονται αντικρυστά σε κύκλο χωρισμένοι σε 2 ομάδες. Οι παίκτες παίρνουν σειρά και κάθε φορά που είναι η σειρά κάποιου προσπαθεί να περιγράψει όσες περισσότερες λέξεις μπορεί στους συμπαίκτες του μέχρι να τελειώσει ο χρόνος του. Όμως κάθε κάρτα έχει κάποιες απαγορευμένες λέξεις οι οποίες δεν πρέπει να λεχθούν. Αν ο παίκτης που περιγράφει πει κάποια από αυτές τις λέξεις τότε η αντίπαλη ομάδα πατάει το κουμπί λάθους και ο παίκτης πρέπει να συνεχίσει στην επόμενη λέξη.

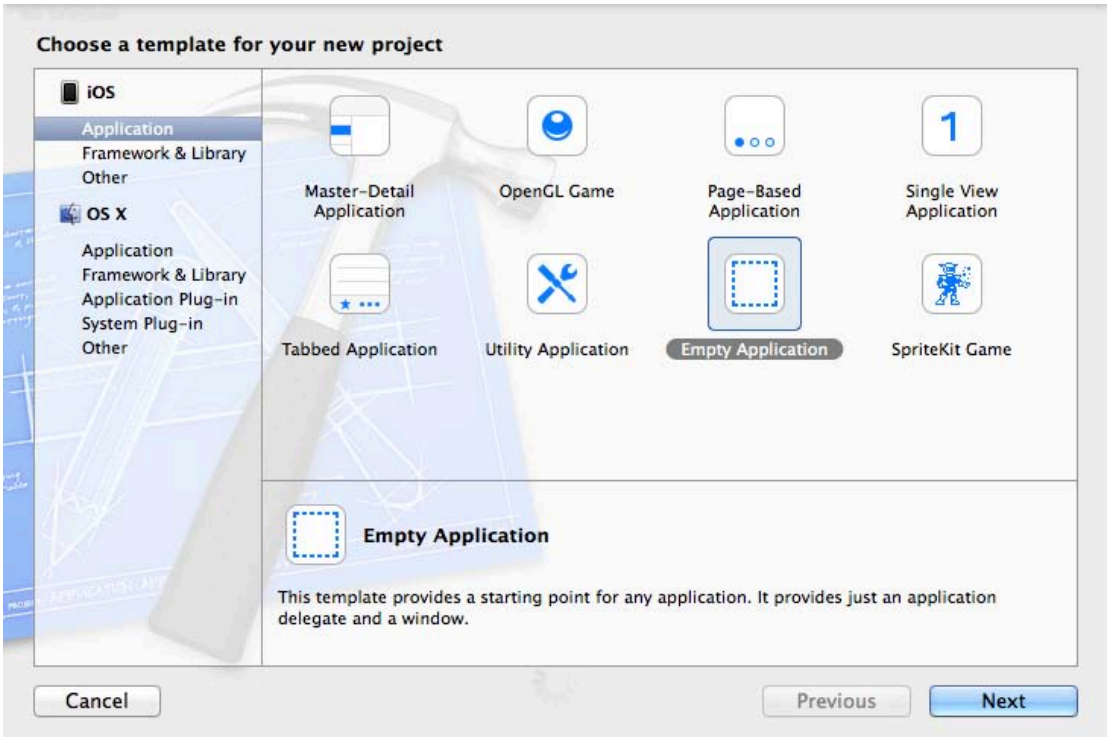
3.1 Δημιουργία βάσης δεδομένων

Η βάση από την οποία παίρνει η εφαρμογή τις κάρτες δημιουργήθηκε με ένα add-on του browser firefox το SQLite manager. Πρόκειται για μία SQL βάση στην οποία πέρασα τις κάρτες. Κάθε στοιχείο (κάρτα) της βάσης αποτελείται από 6 πεδία τύπου string. Την ζητούμενη λέξη και τις 5 «απαγορευμένες». Σίγουρα ήταν μια ανιαρή διαδιακασία!

Το αρχείο που δημιουργήθηκε είναι το cards.sqlite που απλά τοποθετήθηκε στο main bundle της εφαρμογής.



Δημιουργία του project



Αρχικά δημιούργησα ένα project στο xCode επιλέγοντας το template όπως βλέπετε στην παραπάνω εικόνα.

Στη συνέχεια δίνουμε στο project όνομα και λοιπές πληροφορίες και το δημιουργούμε. Το Xcode δημιουργεί κάποια αρχεία μόνο του που είναι απαραίτητα για οιποιοδήποτε project. Ένα από αυτα είναι η κλάσση appDelegate.h.Όπως φαίνεται στον παρακάτω κώδικα στην συνάρτηση που καλεί το λειτουργικο όταν ξεκινάει η εφαρμογή δημιουργείται ο πρώτος viewController.

```
  - (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
    {
        self.window = [[UIWindow alloc] initWithFrame:[[UIScreen
    mainScreen] bounds]];
        // Override point for customization after application
    launch.

        [self initializeRuntimeCardsArray];

        self.viewController = [[iPantViewController alloc]
    initWithNibName:@"iPantViewController" bundle:nil];
        self.window.rootViewController = self.viewController;
        self.soundOn = YES;
        [self.window makeKeyAndVisible];
        return YES;
    }
```

Επίσης σε αυτήν την μέθοδο φορτώνω και όλη την βάση των καρτών καλόντας την [self initializeRuntimeCardsArray]. Με αυτήν την μέθοδο δημιουργείται ένας πίνακας αντικειμένων, όπου κάθε αντικείμενο είναι μία καρτα, ορατός από όλες τις κλάσεις της εφαρμογής.

Στο Xcode υπάρχουν κάποια σημεία στα οποία δηλωνονται κάποια απαραίτητα στοιχίεα για να μπορέσει ο compiler να παράξει το εκτελειμο αρχείο της εφαρομογής. Τέτοια είναι το πεδίο που δηλώνονται τα frameworks, τα certificates της εφαρμογής, τα app icons κλπ.

3.2 Βασικές κλάσσεις

• Η εφαρμογή διαθέτει μια κλάσση την GameModel που διαχειρίζεται όλα τα στοιχεία της εφαρμογής οπως σκορ ομάδων, σειρα, αριθμό γύρων κλπ. Χρησιμοποιείται ένα μοναδικό στιγμιότυπο της που είναι οράτο από καθε κλάσση της εφαρμογής. Αυτό το στιγμιότυπο χρησιμοποιείται από τις άλλες κλάσεις με την κλήση:

```
[GameModel sharedInstance]
```

Καποιες από τις βασικές του μεθόδους είναι:

```
-(NSString *)nameOfTeam1{

    _nameOfTeam1 = [self getUDObjectForKey:@"nameOfTeam1"];
    return _nameOfTeam1;
}


-(NSInteger)RoundDuration{

    _RoundDuration = [[self getUDObjectForKey:@"roundDuration"]
integerValue];

    return _RoundDuration;
}
```

Επιστρέφουν διάφορες τιμές από ρυθμίσεις του παιχνιδιού.

```
-(void)roundEnded{



    if ([[self getUDObjectForKey:@"currnetRound"] integerValue]
== [[self getUDObjectForKey:@"numberOfRounds"] integerValue] &&
[[self getUDObjectForKey:@"turn"] integerValue] == 1) { //end of
game


        [self setUDobject:[NSNumber numberWithInteger:0]
forKey:@"gameIsOpenned"];
        return;
```

```
    }


    if ([[self getUDObjectForKey:@"turn"] integerValue] == 1) {


        NSInteger newRound = [[self
getUDObjectForKey:@"currnetRound"] integerValue] + 1;
        [self setUDobject:[NSNumber numberWithInteger:newRound]
forKey:@"currnetRound"];

    }
    self.turn = !self.turn;
    NSInteger newTurn = [[self getUDObjectForKey:@"turn"]
integerValue];
    if (newTurn == 0) {
        newTurn = 1;
    }
    else{
        newTurn = 0;
    }

    [self setUDobject:[NSNumber numberWithInteger:newTurn]
forKey:@"turn"];



}
```

Καλείται κάθε φορά που τελειώνει ένας γύρος και ανανεώνει τις απαιτούμενες ρυθμίσεις

```
-(void)setGameSettingsWithTeamNames:(NSArray *)teamNames
roundDuration:(NSInteger)duration
numberOfRounds:(NSInteger)numOfRounds andMembersOfTeam1:(NSArray
*)members1 andMembersOfTeam2:(NSArray *)members2{


    [self setUDobject:[teamNames objectAtIndex:0]
forKey:@"nameOfTeam1"];
    [self setUDobject:[teamNames objectAtIndex:1]
forKey:@"nameOfTeam2"];
    [self setUDobject:[NSNumber numberWithInteger:duration]
forKey:@"roundDuration"];
    [self setUDobject:[NSNumber numberWithInteger:numOfRounds]
forKey:@"numberOfRounds"];
    [self setUDobject:[NSNumber numberWithBool:YES]
forKey:@"settingsOk"];
```

```
    [self setUDobject:members1 forKey:@"membersOfTeam1"];
    [self setUDobject:members2 forKey:@"membersOfTeam2"];


}
```

Αποθηκεύει τις ρυθμίσεις που επέλεξε ο χρήστης

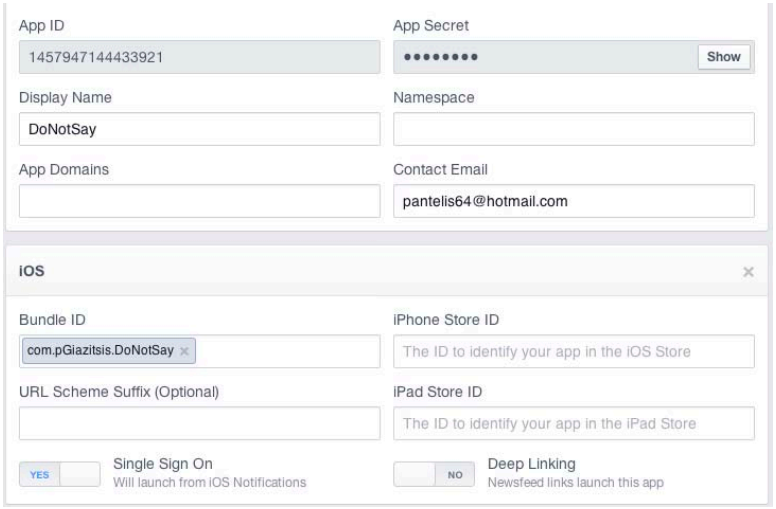
• Επίσης υπάρχει και η κλάση Sounds που διαχειρίζεται την αναπαραγωγή των ήχων. Όπως και η GameModel χρησιμοποιείται από την εφαρμογή με ένα μοναδικό στιγμιότυπο της που είναι ορατό από καθε κλάση της. Η βασική της μέθοδος είναι:

```
-(void)playSound:(NSString *)soundFile{

    if (_SoundOff) {
        return;
    }


    NSString *pathsoundFile = [[NSBundle mainBundle]
pathForResource:soundFile ofType:@"mp3"];
      _sound= [[AVAudioPlayer alloc] initWithContentsOfURL:[NSURL
fileURLWithPath:pathsoundFile] error:NULL];


    if ([[NSFileManager defaultManager]
fileExistsAtPath:pathsoundFile]) {
        NSLog(@"Exists");
    }
    _sound.volume = 1;
    [_sound play];

}
```

Αφού ελέγξει αν ο χρήστης έχει ενεργοποιημένους τους ήχους αναπαράγει τον ζητούμενο ήχο.

3.3 Facebook SDK

Για να χρησιμοποιήσει μία εφαρμογή του framework του facebook πρέπει ο προγραμματιστής να δηλωθεί στο site του facebook ως developer και επίσης να δηλωθεί η εφαρμογή που θα χρησιμοποιήσει αυτό το framework. Αυτό επιτυγχάνεται αναγράφοντας τον, μοναδικό για κάθε εφαρμογή bundle identifier. Στη συνέχεια δηλώνεται και στο info-plist (αρχείο ρυθμίσεων) της εφαρμογής.



Δήλωση στο facebook

3.4 Οθόνες της εφαρμογής

Κάθε οθόνη της εφαρμογής είναι ένας ViewContoller. Συνήθως ένας viewController δημιουργείται «πακέτο» με ένα αρχείο xib που είναι αρχείο για το interface του χρήστη.

- 1η οθόνη (mainViewController)

Έδω ο χρήστης έχει κάποιες βασικές επιλογές.

1) Να ξεκινήσει το παιχνίδι
Αν υπάρχει παιχνίδι σε εξέλιξη η εφαρμογή θα ρωτήσει αν θέλει ο χρήστης να επιστρέψει σε αυτό ή αν θέλει να ξεκινήσει ενα νέο.

2) Οδηγίες παιχνιδιού
Εμφανίζει ένα κείμενο με οδηγίες προς τον χρήστη.

3) Ιστορικό
Εμφανίζει το αρχείο με τις παρτίδες που έχουν τελειωσει με ταξινόμηση ημερομηνίας.

4) Ήχος on/off
Ενεργοποίηση απενεργοποίηση των ήχων

5) Πληροφορίες για την εφαρμογή.

MainViewController

Αρχικά παρατηρούμε ενα γραφικό καρουζέλ στο κέντρο της οθόνης. Το συγκεκριμένο κομματι είναι ένας κωδικας που έχει φτιαχτεί απο προγραμματιστες και διατιθεται στο github (βλ. Βιβλιογραφία). Αυτό το προσάρμοσα στις ανάγκες της εφαρμογής.

Η σημαντικότερη συνάρτηση αυτής της οθόνης είναι:

```
- (void)carousel:(iCarousel *)carousel
didSelectItemAtIndex:(NSInteger)index
{

    if (index == 0) {
```

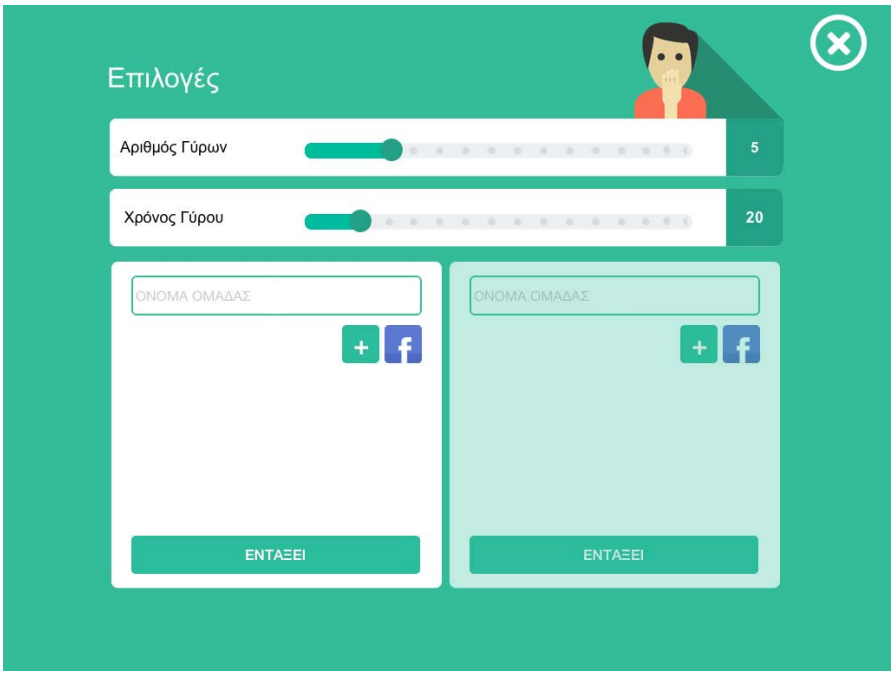
```
        if ([[GameModel sharedInstance] isGameOpenned]) {
//game is open

            UIAlertView *alert = [[UIAlertView alloc]
initWithTitle:@"Ανοιχτό παιχνίδι" message:@"θέλετε να
συνεχίσεται" delegate:self cancelButtonTitle:@"OXI"
otherButtonTitles:@"NAI", nil];
            [alert show];
        }
        else{
            [[Sounds sharedInstance] playSound:@"whistle"];

            [[GameModel sharedInstance] clearGame];
            OptionsViewController *optionController =
[[OptionsViewController alloc] init];
            [self presentViewController:optionController
animated:YES completion:nil];

        }

    }
    if (index == 1) {
        [[Sounds sharedInstance] playSound:@"klik"];

        HistoryViewController *optionController =
[[HistoryViewController alloc] init];
        [self presentViewController:optionController animated:YES
completion:nil];

    }
    if (index == 2) {

        [[Sounds sharedInstance] playSound:@"klik"];

        alert = [[CustomAlert alloc]
initWithFrame:self.view.frame];
        [alert setInstryctionsText];

        [alert showAlertfromView:self.view];

    }
}
```

Αυτή καλείται όταν ο χρήστης επιλέξει μία από τις 3 καρτέλες που βρίσκονται στο καρουζέλ. Στην πρώτη επιλογή ξεκιναέι το παιχνίδι (index == 0). Στην δεύτερη προβάλεται η οθόνη με το αρχείο (index == 1). Στην τρίτη εμφανίζεται ένα παράθυρο τύπου alert με ένα κείμενο με οδηγίες για το πως

παίζεται το παιχνίδι.

- 2η οθόνη (OptionsViewController)

Όταν ο χρήστης ξεκινήσει ένα νέο παιχνίδι τότε εμφανίζεται αυτή η οθόνη. Εδώ μπορεί ο χρήστης να ρυθμίσει της επιλογές του. Τον χρόνο του κάθε γύρου, τον αριθμό των γύρων, και τα μέλη της κάθε ομαδας.



OptionsViewController

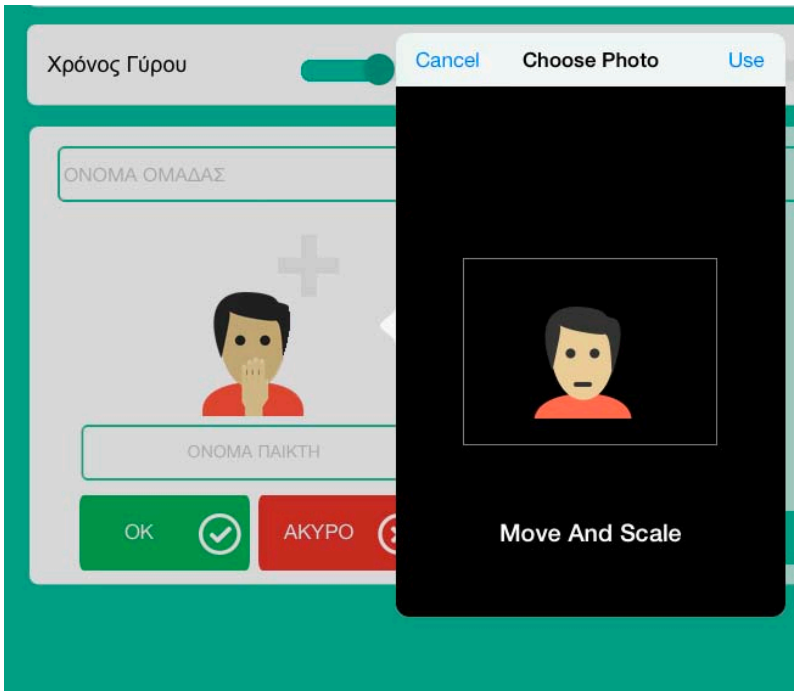
Τα μέλη της κάθε ομάδας δηλώνονται με το όνομα και την φωτογραφία τους. Μπορεί ο χρήστης να επιλέξει φωτογραφία από τις αποθηκευμένες του κινητού αλλα και από το facebook.

<u>Δημιουργία ομάδας:</u>

Εδώ δινεται το όνομα της ομάδας και τα μέλη της.

Ένα σημαντικό κομματι αυτής της οθόνης είναι το κομματι που προστιθεται ενα καινούριο μέλος στην ομάδα. Η μία επιλογή είναι από τους φίλους του χρήστη στο facebook. Η άλλη είναι να γράψει το όνομα και να διαλέξει μια φωτογραφία απο τις αποθηκευμένες στην συσκευή. Για την

επιλογή του facebook χρησιμοποιήθηκε το framework που παρέχει το facebook για το ios. Για την άλλη επιλογή χρησιμοποίησα και προσάρμοσα ένα κομματι κώδικα από το github για να μπορώ να κάνω την απαραίτητη επεξεργασία των εικόνων (βλ. Βιβλιογραφία).



Δημιουργία ομάδας

Όταν επιλεγεί ο δεύτερος τρόπος (πατόντας το +) τότε εμφανίζεται μια νέα μικρή καρτέλα στην θέση της ομάδας για να δωθούν τα στοιχεία του μέλους.

Όταν πλέον δημιουργηθούν και οι δύο ομάδες ξεκινάει το παιχνίδι. Καλείται η αντιίστοιχη μέθοδος της GameModel και ρυθμίζονται οι παράμετροι του παιχνιδιού.

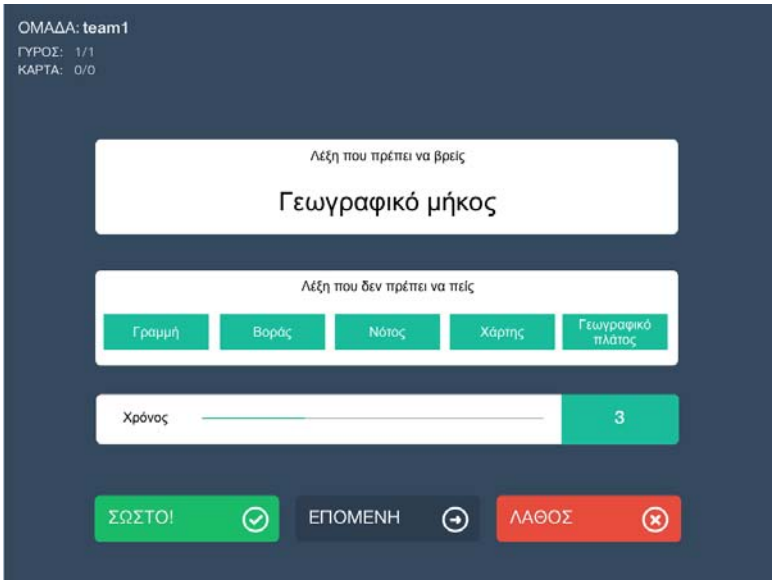
- 3η οθόνη (GameViewController)

Αυτή η οθόνη είναι και η οθονη του παιχνιδου. Υπάρχει μία μπάρα για την καταμέτηρηση του χρόνου. Αυτή γεμίζει περιοδικά με την βοήθεια ενός αντικειμένου, που μας παρέχει το iOS SDK, του NSTimer. Προκειται για ένα αντικείμενο που κάλει μια συνάρτηση που του ορίζουμε περιοδικά. Στον δικό μου κώδικα δημιουργείται όταν ξεκινήσει ο γύρος

```
timer = [NSTimer scheduledTimerWithTimeInterval: 1.0
                                         target: self
                                       selector:
@selector(updateProgress)
                                       userInfo: nil
                                        repeats: YES];
```

και καλεί την updateProgress κάθε ένα δευτερόλεπτο.

```
-(void)updateProgress{

    self.timeProgress.progress += step;

    secCounter++;
    self.timeLabel.text = [NSString
stringWithFormat:@"%d",secCounter];
    if (secCounter == roundTime) {

        [timer invalidate];

        [self endRound];
    }

}
```

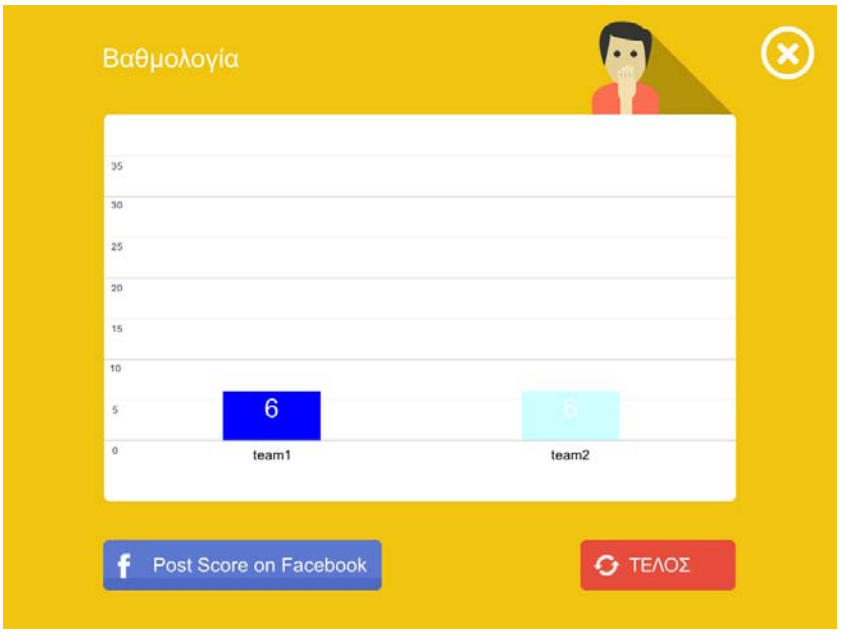


GameViewController

Μέσα σε αυτή την συναρτηση γίνεται και ο έλεγχος για το πότε πρέπει να τελεώσει ο γύρος και να σταματήσει να μετράει ο NSTimer. Όταν τελειώσει ο γύρος ενημερώνεται το GameModel για το σκορ και εμφανίζεται η οθόνη με τα σκορ της κάθε ομάδας.

- 4η οθόνη (ScoreViewController)

Σε αυτή την οθόνη εμφανίζεται σε γράφημα το σκορ του παιχνιδιού. Ο χρήστης πατόντας το πάνω δεξία κουμπί μπορεί να σταματήσει το παιχνίδι και αν θέλει να το συνεχίσει κάποια άλλη στιγμη. Επίσης με τον πλήκτο «ΞΑΝΑ» περνάμε στον επόμενο γύρο.



ScoreViewController

Στο τέλος του παιχνιδιού ενεργοποιείται το κουμπί «Post Score on Facebook» με το οποί ο χρήστης μπορεί να κοινοποιήσει το σκορ στον λογαριασμό του στο facebook.

Για την λειτουργία του Facebook χρησιμοποήθηκε το social framework του iOS. Ένα framework που δίνει την δυνατότητα στον προγραμματιστη να δημοσιοποιήσει εύκολα κάποιο σχόλιο στα social media twitter και facebook.

Κοινοποίηση στο Facebook

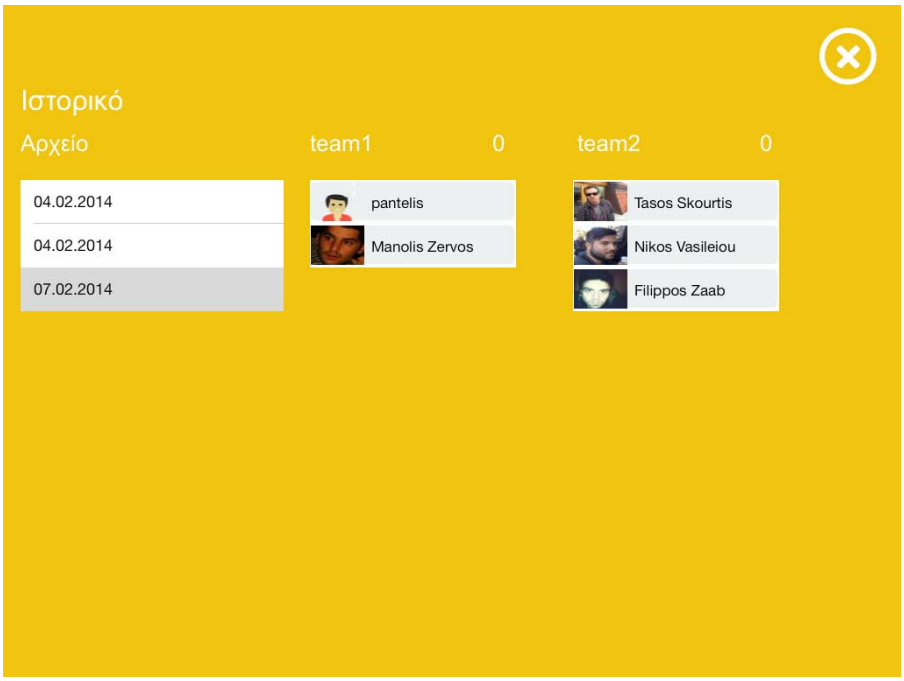
Επίσης στο τέλος του γύρου το κουμπί «ΞΑΝΑ» μετατρέπεται σε «ΤΕΛΟΣ» και όταν το πατήσει ο χρήστης μεταφέρεται στην αρχική οθόνη και μέσω του GameModel αποθηκεύεται το σκορ και οι ομάδες στο ιστορικό. Αυτό επιτυγχάνεται με την κλήση της συνάρτησης:

```
-(void)saveGame{

    NSMutableDictionary *dict = [[NSMutableDictionary alloc]
initWithObjects:[NSArray arrayWithObjects:[self
getUDObjectForKey:@"nameOfTeam1"],[self
getUDObjectForKey:@"nameOfTeam2"],[self
getUDObjectForKey:@"team1score"],[self
getUDObjectForKey:@"team2score"],[self
getUDObjectForKey:@"membersOfTeam1"],[self
getUDObjectForKey:@"membersOfTeam2"], nil] forKeys:[NSArray
arrayWithObjects:@"nameOfTeam1",@"nameOfTeam2",@"team1score",@"te
am2score",@"membersOfTeam1",@"membersOfTeam2", nil]];


    NSTimeInterval  today = [[NSDate date]
timeIntervalSince1970];
    NSString *intervalString = [NSString
stringWithFormat:@"%.0f", round(today) ];
    NSLog(@"%@",intervalString);


    NSString *plistName = intervalString;
    NSArray *paths =
NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
NSUserDomainMask, YES);
    NSString *documentsDirectory = [paths objectAtIndex:0];
    NSString *path = [documentsDirectory
stringByAppendingPathComponent:[NSString
stringWithFormat:@"%@.plist",plistName]];

    NSData *data = [NSKeyedArchiver
archivedDataWithRootObject:dict];
    if ([data writeToFile:path atomically:YES]) {
        NSLog(@"saved to path %@",path);
    }

    data = [NSData dataWithContentsOfFile:path];


}
```

Δημιουργεί ένα NSDictionary με τα στοιχεία του παιχνιδιού που θέλει να αποθηκεύσει και στην συνέχεια τα αποθηκεύει με την μορφή αρχείου plist με όνομα την ημερομηνία σε μορφή unix timestamp[†]. Τα αρχεία plist είναι αρχεία τύπου xml που χρησιμοποιούνται πολύς συχνά από εφαρμογές iOS για αποθήκευση και ανάκτηση ρυθμίσεων.

- 5η οθόνη (HistoryViewController)

Σε αυτή την οθόνη ο χρήστης μπορει να δει το σκορ από προηγούμενα παιχνίδια ταξινομημένα ανα ημερομηνία. Επιλέγει την ημερομηνία της επιλογής του από την πρώτη στήλη και στις άλλες δύο εμφανίζονται το σκορ και οι ομάδες.



HistoryViewController

Όαν δημιουργέιται για να προβληθεί ο συγκεκριμένος controller ανακτά από τον GameModel τα plist αρχεία και με την κατάλληλη επεξεργασία εξάγει τις απαιτούμενες πληροφορίες.

[†] Unix timestamp είναι ο αριθμός των δευτερολέπτων που έχουν περάσει απο την χρονιά 1970 (χρονιά δημιουργίας του λειτουργικού Unix)

4.Βιβλιογραφία

- http://www.mobilemarketer.com
- http://talklets.com
- http://www.uky.edu/~jclark/mas490apps/History%20of%20Mobile%20Apps.pdf
- http://www.cultofmac.com/191340/the-evolution-of-ios-from-iphone-os-to-ios-6-gallery/
- http://en.wikipedia.org/wiki/Objective-C
- https://developer.apple.com/library/ios/referencelibrary/GettingStarted/RoadMapiOS/FirstTutorial.html#//apple_ref/doc/uid/TP40011343-CH3-SW1
- http://www.hongkiat.com/blog/ios-history/
- https://itunes.apple.com
- https://developer.apple.com
- http://www.sitepoint.com/ios-application-development-mvc-programming-basics/
- http://cupsofcocoa.files.wordpress.com

Samples κώδικα

- https://github.com/nicklockwood/iCarousel
- https://github.com/gekitz/GKImagePicker