ΑΛΕΞΑΝΔΡΕΙΟ Τ.Ε.Ι. ΘΕΣΣΑΛΟΝΙΚΗΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Επαύξηση του περιεχομένου ενός βιβλίου και ενός χάρτη με 3D μνημεία με χρήση Apple SDK ARkit

του φοιτητή

Διαμαντίδη Βασίλειου

Αρ. Μητρώου: 134005

Επιβλέπων καθηγητής

Κεραμόπουλος Ευκλείδης

Θεσσαλονίκη 2019

ΠΡΟΛΟΓΟΣ

Η Επαυξημένη πραγματικότητα Augmented Reality (AR) περιγράφει τις εμπειρίες των χρηστών στις οποίες προστίθενται στοιχεία 2D ή 3D στην ζωντανή προβολή από τη φωτογραφική μηχανή μιας συσκευής με τρόπο που τα στοιχεία αυτά φαίνεται να “κατοικούν” στον πραγματικό κόσμο. Τα πιο συνηθισμένα είδη εμπειρίας AR εμφανίζουν μια προβολή από μια κάμερα, που προστίθεται άλλο οπτικό περιεχόμενο, δίνοντας στον χρήστη έναν νέο τρόπο να βλέπει και να αλληλοεπιδρά με τον κόσμο γύρω του.

ΠΕΡΙΛΗΨΗ

Η πτυχιακή εργασία έχει σαν σκοπό την ολοκληρωμένη τεκμηρίωση των στοιχείων που χρησιμοποιήθηκαν για την δημιουργία της εφαρμογής του Ολυμπιακού μουσείου, καθώς και η ανάλυση των κλάσεων του κώδικα που χρησιμοποιήθηκαν και γράφτηκαν. Με αυτό το κείμενο και την τεκμηρίωση μπορεί ο αναγνώστης να έχει μία πλήρη εικόνα για τον προγραμματισμό κινητών που έχουν το λειτουργικό σύστημα iOS και μπορεί να το χρησιμοποιήσει ως ένα εγχειρίδιο για τον προγραμματισμό των δικών του επαυξημένης πραγματικότητας εφαρμογών και όχι μόνο.

ABSTRACT

The application is innovative and advanced. Created for the Olympic Museum in order to make the museum more public-friendly and evolved. The application is for mobile devices with iOS 11.3 operating system and contains three different examples of augmented reality use. The movement of athletes in images as well as the mapping of an ancient map and the addition of an olive wreath on the head of the user.

ΠΕΡΙΕΧΟΜΕΝΑ

Ευρετήριο σχημάτων, εικόνων και αποσπασμάτων κώδικα

ΕΙΣΑΓΩΓΗ

Σκοπός της εργασίας είναι η δημιουργία μίας εφαρμογής με λειτουργείες επαυξημένης πραγματικότητας καθώς και η λεπτομερής περιγραφή του τρόπου και των εργαλειών τα οποία χρειάστηκαν για τη δημιουργεία της. Η εφαρμογή δημιουργήθηκε σε συνεργασία με το Ολυμπιακό μουσείο Θεσσαλονίκης. Η εργασία αποτελείται από τέσσερα κεφάλαια. Το πρώτο κεφάλαιο εξηγεί τη λειτουργεία του iOS λειτουργικού συστήματος, τον προγραμματισμό των εφαρμογών που τρέχουν σε αυτό , τα βασικότερα πλαίσια που χρησιμοποιούνται καθώς και τις σημαντικότερες κλάσεις και αντικείμενα κλάσεων, που δε λείπουν από καμία εφαρμογή. Το δεύτερο κεφάλαιο εξειδικεύεται στο πλαίσιο ArKit, και SceneKit που είναι τα δύο πλαίσια που χρησιμοποιούνται για να γίνονται διεπαφές χρήστη επαυξημένης πραγματικότητας. Το τρίτο κεφάλαιο ασχολείται με όλα τα εργαλεία που χρησιμοποιήθηκαν για την παράδοση του τελικού αποτελέσματος της εφαρμογής. Το τέταρτο κεφάλαιο αναλύει την εφαρμογή, τον τρόπο λειτουργίας της καθώς και τον τρόπο υλοποίησης της.

ΚΕΦΑΛΑΙΟ 1

iOS development

ΕΙΣΑΓΩΓΗ

Σε αυτό το κεφάλαιο περιγράφονται όλα τα βασικά χαρακτηριστικά που χρειάζεται να ξέρει κάποιος προγραμματιστής για την δημιουργία εφαρμογών για κινητά με λειτουργικό σύστημα iOS. Αρχικά περιγράφεται το ίδιο το λειτουργικό σύστημα και η γλώσσα στην οποία γράφονται οι εφαρμογές και έπειτα το εργαλείο που χρησιμοποιείται για την γραφή του κώδικα. Τέλος αναλύεται το βασικότερο framework όλων των εφαρμογών το UIKit και των σημαντικότερων κλάσεων του.

iOS

Το iOS είναι ένα λειτουργικό σύστημα που δημιουργήθηκε από την Apple αποκλειστικά για το υλικό που παράγει η ίδια η εταιρεία. Είναι το λειτουργικό σύστημα που χρησιμοποιείται σε πολλές από τις κινητές συσκευές της, συμπεριλαμβανομένων των iPhone, iPad και iPod Touch. Είναι το δεύτερο πιο δημοφιλές κινητό λειτουργικό σύστημα παγκοσμίως μετά το Android.

Αρχικά παρουσιάστηκε το 2007, για το iPhone. Το iOS έχει επεκταθεί για να υποστηρίξει και άλλες συσκευές της Apple: όπως το iPod Touch (Σεπτέμβριος 2007) και το iPad (Ιανουάριος 2010). Από τον Μάρτιο του 2018 το App Store της Apple περιέχει περισσότερα από 2,1 εκατομμύρια iOS εφαρμογές, 1 εκατομμύριο από τα οποία είναι για iPads.

Η διεπαφή χρήστη iOS βασίζεται σε άμεσο χειρισμό, χρησιμοποιώντας χειρονομίες πολλαπλής αφής. Τα στοιχεία ελέγχου της διασύνδεσης αποτελούνται από διακόπτες και κουμπιά. Η αλληλεπίδραση με το λειτουργικό σύστημα περιλαμβάνει χειρονομίες όπως μετακινούμενο άγγιγμα, ελαφρύ χτύπημα, μετακινούμενο άγγιγμα με δύο δάχτυλα συνθέσεις αυτών, τα οποία διαθέτουν ειδικούς ορισμούς στα πλαίσια του λειτουργικού συστήματος iOS και των διεπαφών πολλαπλής αφής. Τα εσωτερικά επιταχυνσιόμετρα χρησιμοποιούνται επίσης σε μερικές εφαρμογές.

Σημαντικές εκδόσεις του iOS κυκλοφορούν ετησίως. Η έκδοση iOS 12, η οποία είναι η νεότερη έκδοση, είναι η δωδέκατη του λειτουργικού συστήματος iOS της Apple, και είναι διάδοχος του iOS 11. Ανακοινώθηκε κατά τη διάρκεια του συνεδρίου προγραμματιστών, WWDC18, της Apple, στις 4 Ιουνίου 2018 και κυκλοφόρησε στο ευρύ κοινό στις 17 Σεπτεμβρίου 2018. Αισθητικά μοιάζει με το iOS 11, ωστόσο περιλαμβάνει αρκετές ενημερώσεις όσον αφορά τις επιδόσεις και τη διάρκεια ζωής της μπαταρίας καθώς και νέες λειτουργίες σε αρκετές εφαρμογές. Βασική προσθήκη στο iOS 12 είναι η δεύτερη έκδοση του πλαισίου ARKit.

Swift

Η Swift είναι μια γλώσσα προγραμματισμού γενικής χρήσης που κατασκευάζεται χρησιμοποιώντας μια σύγχρονη προσέγγιση για την ασφάλεια, την απόδοση και τα σχέδια σχεδιασμού λογισμικού.

Ο στόχος της γλώσσας Swift είναι να δημιουργήσει την καλύτερη διαθέσιμη γλώσσα για χρήσεις που κυμαίνονται από τον προγραμματισμό συστημάτων έως εφαρμογές για κινητά και υπολογιστές, κλιμακούμενες μέχρι υπηρεσίες cloud. Το πιο σημαντικό είναι ότι η Swift έχει σχεδιαστεί για να διευκολύνει το γράψιμο και τη συντήρηση των προγραμμάτων για τον προγραμματιστή.

Η Swift είναι:

- Ασφαλής. Τα λάθη του προγραμματιστή πρέπει να πιάνονται πριν από την παραγωγή του λογισμικού. Η Swift έχει μηχανισμούς οι οποίοι προλαμβάνουν τα λάθη αυτά και είναι προαιρετικοί. Η επιλογή για ασφάλεια σημαίνει μερικές φορές ότι η Swift είναι αυστηρή γλώσσα, αλλά η σαφήνεια της έχει σαν αποτέλεσμα εξοικονομεί χρόνο μακροπρόθεσμα.
- Γρήγορη. Η Swift προορίζεται να αντικαταστήσει τις γλώσσες που βασίζονται σε C (C, C ++ και Objective-C). Ως εκ τούτου, η Swift είναι συγκρίσιμη με αυτές τις γλώσσες στην απόδοση για τις περισσότερες εργασίες. Η απόδοση είναι επίσης προβλέψιμη και συνεπής, όχι μόνο γρήγορη σε κάποιες περιπτώσεις.
- Εκφραστική. Η Swift επωφελείται από δεκαετίες προόδου στην επιστήμη των υπολογιστών για να προσφέρει σύνταξη που είναι ευχάριστη στη χρήση, με σύγχρονα χαρακτηριστικά που περιμένουν οι προγραμματιστές. Η γλώσσα συνεχώς εξελίσσεται για να γίνεται ακόμη καλύτερη.

Τα εργαλεία αποτελούν κρίσιμο μέρος της Swift. Εντάσσονται καλά στο σύνολο εργαλείων ενός προγραμματιστή, δημιουργούν κώδικα γρήγορα, παρουσιάζουν διαγνώσεις για τον κώδικα και δίνουν τη δυνατότητα δια δραστικών εμπειριών ανάπτυξης. Τα εργαλεία μπορούν να κάνουν τον προγραμματισμό πιο ισχυρό. Ένα παράδειγμα εργαλείου είναι το Swift-based playgrounds που υπάρχει στο Xcode.[4]

Χαρακτηριστικά / Features

Η Swift περιλαμβάνει λειτουργίες που διευκολύνουν την ανάγνωση και γραφή του κώδικα, δίνοντας στον προγραμματιστή τον απαραίτητο έλεγχο σε μια πραγματική γλώσσα προγραμματισμού συστημάτων. Η Swift υποστηρίζει υποτιθέμενους

τύπους για να καταστήσει τον κώδικα καθαρότερο και λιγότερο επιρρεπείς σε λάθη, και οι ενότητες εξαλείφουν τις επικεφαλίδες και παρέχουν χώρους ονομάτων. Η μνήμη διαχειρίζεται αυτόματα και δεν χρειάζεται καν η πληκτρολόγηση κάποιου συμβόλου στο τέλος κάθε γραμμής κώδικα για την ένδειξη ολοκλήρωσης γραμμής. Η Swift δανείζεται επίσης από άλλες γλώσσες, για παράδειγμα οι ονομαζόμενες παράμετροι που προωθούνται από την γλώσσα Objective-C εκφράζονται σε μια καθαρή σύνταξη που κάνει τα API στη Swift εύκολο να διαβαστούν και να διατηρηθούν.

Τα χαρακτηριστικά της Swift έχουν σχεδιαστεί για να συνεργάζονται και να δημιουργήσουν μια γλώσσα που είναι ισχυρή, αλλά διασκεδαστική στη χρήση. Ορισμένες πρόσθετες λειτουργίες του Swift περιλαμβάνουν:

- Closures ενοποιημένα με δείκτες λειτουργίας [6]
- Tuples και πολλαπλές επιστρεφόμενες τιμές [7]
- Generics [8]
- Γρήγορη και συνοπτική επανάληψη σε ένα range ή ένα collection
- Structs που υποστηρίζουν μεθόδους, επεκτάσεις και πρωτόκολλα [9]
- Functional programming patterns, π.χ. map και filter [10, 11]
- Ισχυρό ενσωματωμένο χειρισμό σφαλμάτων [11]
- Προχωρημένη ροή ελέγχου με την εκτέλεση do, guard, defer, και repeat λέξεων-κλειδιών [12]

Ασφάλεια

Η Swift σχεδιάστηκε εξ αρχής για να είναι ασφαλέστερη από τις γλώσσες που βασίζονται σε C και εξαλείφει ολόκληρες κατηγορίες μη ασφαλούς κώδικα. Οι μεταβλητές αρχικοποιούνται πάντα πριν από τη χρήση, οι πίνακες και οι ακέραιοι αριθμοί ελέγχονται για υπερχείλιση και η μνήμη διαχειρίζεται αυτόματα. Η σύνταξη ρυθμίζεται με ευκολία - για παράδειγμα, οι απλές λέξεις-κλειδιά τριών χαρακτήρων ορίζουν μια μεταβλητή (var) ή μια σταθερά (let).

Ένα άλλο χαρακτηριστικό ασφαλείας είναι ότι από προεπιλογή τα αντικείμενα Swift δεν μπορούν ποτέ να είναι μηδενικά και η προσπάθεια να δημιουργηθεί ή να χρησιμοποιηθεί ένα μηδενικό αντικείμενο θα έχει ως αποτέλεσμα ένα compile-time error. Αυτό καθιστά τον κώδικα γραφής πολύ καθαρότερο και ασφαλέστερο και αποτρέπει μια κοινή αιτία των συνθηκών εκτέλεσης. Ωστόσο, υπάρχουν περιπτώσεις όπου μία τιμή είναι κατάλληλη για αυτές τις καταστάσεις η Swift διαθέτει ένα καινοτόμο χαρακτηριστικό γνωστό ως optional. Ένα optional μπορεί να περιέχει τιμή αλλά μπορεί και όχι, η σύνταξη της Swift αναγκάζει τη διαχείριση αυτού με ασφάλεια χρησιμοποιώντας κάποιες τεχνικές.

Η πρώτη τεχνική είναι χρησιμοποιώντας ένα «?» για να υποδείξει στον compiler την κατανόηση της συμπεριφοράς και τον χειρισμό με ασφάλεια. Όταν θα φτάσει η στιγμή της εκτέλεσης του προγράμματος και θα χρειαστεί να χρησιμοποιηθούν τα

δεδομένα της μεταβλητής η οποία έχει ερωτηματικό, αν η μεταβλητή δεν έχει τιμή τότε η εφαρμογή θα συνεχίσει χωρίς την εκτέλεση της γραμμής αυτής.

Στην δεύτερη τεχνική χρησιμοποιείται ένα «!» που ορίζει στον compiler πως αν χρειαστεί η ανάγνωση της τιμής αυτής της μεταβλητής αλλά έχει μηδενική τιμή τότε υποχρεωτικά πρέπει να τερματιστεί η εφαρμογή. Η τεχνική αυτή χρησιμοποιείται σε συγκεκριμένες περιπτώσεις και είναι αρκετά χρήσιμη για της εύρεση λαθών κατά την ανάπτυξη κώδικα, πριν την δημοσίευση της τελικής εφαρμογής.

Η τρίτη τεχνική είναι η ανάθεση προκαθορισμένης τιμής και χρησιμοποιείται με τα σύμβολα «??». Μετά τα σύμβολα ακολουθεί η προκαθορισμένη τιμή. Όταν κατά την εκτέλεση του προγράμματος χρειαστεί η τιμή της μεταβλητής αυτής, τότε αν δεν υπάρχει θα χρησιμοποιηθεί η προκαθορισμένη τιμή.

Υποστήριξη πλατφόρμας

Μια από τις πιο συναρπαστικές πτυχές της ανάπτυξης με τη γλώσσα Swift είναι ότι γίνεται να μεταφερθεί σε ένα ευρύ φάσμα από πλατφόρμες, συσκευές και περιπτώσεις χρήσεως.

Στόχος είναι να παρέχεται συμβατότητα πηγής για τη Swift σε όλες τις πλατφόρμες, παρόλο που οι πραγματικοί μηχανισμοί εφαρμογής ενδέχεται να διαφέρουν από τη μία πλατφόρμα στην άλλη. Το κύριο παράδειγμα είναι ότι οι πλατφόρμες της Apple περιλαμβάνουν Objective-C runtime, το οποίο απαιτείται για την πρόσβαση σε πλαίσια της πλατφόρμας της Apple όπως UIKit και AppKit. Στις άλλες πλατφόρμες, όπως το Linux, δεν υπάρχει Objective-C runtime, επειδή δεν είναι απαραίτητο.

Το έργο Swift Core Libraries στοχεύει στην επέκταση των δυνατοτήτων cross-platform της Swift παρέχοντας φορητές υλοποιήσεις βασικών πλαισίων της Apple (όπως το Foundation) χωρίς εξαρτήσεις για το χρόνο εκτέλεσης της Objective-C. Αν και οι βασικές βιβλιοθήκες βρίσκονται σε πρώιμο στάδιο ανάπτυξης, παρέχουν τελικά βελτιωμένη συμβατότητα πηγής για τον κώδικα Swift σε όλες τις πλατφόρμες.

Πλατφόρμες της Apple

Η Swift ανοιχτού κώδικα μπορεί να χρησιμοποιηθεί σε Mac για να στοχεύσει όλες τις πλατφόρμες της Apple: iOS, macOS, watchOS και tvOS. Επιπλέον, οι δυαδικές κατασκευές Swift ανοιχτού κώδικα ενσωματώνονται με τα εργαλεία ανάπτυξης προγραμματιστών Xcode, συμπεριλαμβανομένης της πλήρους υποστήριξης για το σύστημα δημιουργίας Xcode, την ολοκλήρωση του κώδικα στον επεξεργαστή και την ολοκληρωμένη αποσφαλμάτωση, επιτρέποντας σε οποιονδήποτε να πειραματιστεί με τις τελευταίες εξελίξεις της Swift σε ένα γνωστό Cocoa και Cocoa Touch περιβάλλον ανάπτυξης. [21]

XCode

Το Xcode είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) για macOS που περιέχει μια σειρά από εργαλεία ανάπτυξης λογισμικού που δημιουργήθηκαν από την Apple για macOS, iOS, watchOS και tvOS. Πρώτα κυκλοφόρησε το 2003. Η πιο πρόσφατη σταθερή έκδοση είναι η έκδοση 12.2 και διατίθεται δωρεάν μέσω του Mac App Store για χρήστες MacOS High Sierra και MacOS Mojave. Οι εγγεγραμμένοι προγραμματιστές μπορούν να πραγματοποιήσουν λήψη ενημερωμένων εκδόσεων και προγενέστερων εκδόσεων μέσω της ιστοσελίδας της Apple Developer. [13]

Το Xcode υποστηρίζει τον πηγαίο κώδικα για τις γλώσσες προγραμματισμού C, C ++, Objective-C, Objective-C ++, Java, AppleScript, Python, Ruby, ResEdit και Swift με διάφορα μοντέλα προγραμματισμού, Cocoa, Carbon και Java. Με επιπρόσθετα εργαλεία υπάρχει υποστήριξη για το GNU Pascal, Free Pascal, Ada, C #, Perl, D και Fortran.

Το Xcode μπορεί να δημιουργήσει fat binary αρχεία που περιέχουν κώδικα για πολλαπλές αρχιτεκτονικές με το εκτελέσιμο αρχείο Mach-O. Αυτά ονομάζονται καθολικά δυαδικά αρχεία, τα οποία επιτρέπουν το λογισμικό να λειτουργεί τόσο σε πλατφόρμες PowerPC όσο και Intel (x86) και μπορεί να περιλαμβάνει κώδικα 32 bit και 64 bit και για τις δύο αρχιτεκτονικές. Χρησιμοποιώντας το iOS SDK, το Xcode μπορεί επίσης να χρησιμοποιηθεί για να μεταγλωττίσει και να εντοπίσει σφάλματα σε εφαρμογές για iOS που εκτελούνται σε επεξεργαστές αρχιτεκτονικής ARM.

Το Xcode περιλαμβάνει το εργαλείο GUI Instruments, το οποίο τρέχει πάνω από ένα δυναμικό πλαίσιο ανίχνευσης, το DTrace, το οποίο δημιουργήθηκε από την Sun Microsystems και κυκλοφόρησε ως μέρος του OpenSolaris.

Η κύρια εφαρμογή της σουίτας είναι το ολοκληρωμένο περιβάλλον ανάπτυξης (IDE), που ονομάζεται επίσης Xcode. Το Xcode περιλαμβάνει τα περισσότερα έγγραφα τεκμηρίωσης της Apple και ενσωματωμένο Interface Builder, μια εφαρμογή που χρησιμοποιείται για την κατασκευή γραφικών διεπαφών χρήστη.

Έως το Xcode 4.1, η σουίτα Xcode περιλαμβάνει μια τροποποιημένη έκδοση της συλλογής μεταγλωττιστή GNU. Στο Xcode 3.1 μέχρι το Xcode 4.6.3 περιλάμβανε τον μεταγλωττιστή LLVM-GCC, με front ends από τη συλλογή μεταγλωττιστή GNU και μια γεννήτρια κώδικα βασισμένη στο LLVM. Στο Xcode 3.2 και αργότερα, περιλάμβανε τον μεταγλωττιστή Clang C / C ++ / Objective-C, με πρόσφατα γραμμένα front ends και μια γεννήτρια κώδικα βασισμένη στο LLVM και τον στατικό αναλυτή Clang. Από το Xcode 4.2, ο μεταγλωττιστής Clang έγινε ο προεπιλεγμένος μεταγλωττιστής. Από το Xcode 5.0, ο Clang ήταν ο μόνος μεταγλωττιστής που παρείχε.

Παλαιότερα, το Xcode υποστήριζε τη διανομή μιας διαδικασίας δημιουργίας προϊόντων σε πολλά συστήματα. Μία σχετική τεχνολογία ονομαζόταν Shared Workgroup Build, η οποία χρησιμοποίησε το πρωτόκολλο Bonjour για να ανακαλύψει αυτόματα τα συστήματα που παρέχουν υπηρεσίες μεταγλωττιστή και μια τροποποιημένη έκδοση του προϊόντος distcc του ελεύθερου λογισμικού για να διευκολύνει τη διανομή του φόρτου εργασίας. Οι παλαιότερες εκδόσεις του Xcode παρείχαν ένα σύστημα με το όνομα Dedicated Network Builds. Αυτές οι δυνατότητες δεν υπάρχουν στις υποστηριζόμενες εκδόσεις του Xcode.

Στις 4 Ιουνίου 2018, στο συνέδριο της Apple Worldwide Developers, ανακοινώθηκε η έκδοση 10 Xcode. μια έκδοση beta κυκλοφόρησε την ίδια ημέρα. Το Xcode 10 παρουσίασε υποστήριξη για το Dark Mode που ανακοινώθηκε για το MacOS Mojave, τις πλατφόρμες συνεργασίας Bitbucket και GitLab (εκτός από το GitHub), μοντέλα μηχανικής εκπαίδευσης από περιβάλλοντα εκμάθησης που ονομάζονται “Playgrounds” και τα νέα χαρακτηριστικά της Swift 4.2 και του Metal 2.1 [4]. Το Xcode 10 έχασε επίσης υποστήριξη για την κατασκευή εφαρμογών macOS 32-bit και δεν υποστηρίζει πλέον την ενσωμάτωση του Subversion. Το Xcode 10 κυκλοφόρησε δημοσίως στις 17 Σεπτεμβρίου 2018. [2]

UIKit

Το πλαίσιο UIKit παρέχει την απαιτούμενη υποδομή για εφαρμογές iOS ή tvOS. Παρέχει την αρχιτεκτονική παραθύρων και προβολών για την υλοποίηση της διασύνδεσης, του χειρισμού της υποδομής και των τύπων εισόδου πολλών επαφών και άλλων τύπων στην εφαρμογή. Άλλα χαρακτηριστικά που προσφέρονται περιλαμβάνουν υποστήριξη κινούμενων εικόνων, υποστήριξη εγγράφων, υποστήριξη σχεδίασης και εκτύπωσης, πληροφορίες σχετικά με την τρέχουσα συσκευή, διαχείριση κειμένου και εμφάνιση, υποστήριξη αναζήτησης, υποστήριξη προσβασιμότητας, υποστήριξη επέκτασης εφαρμογών και διαχείριση πόρων.

Storyboard

Ένα Storyboard είναι μια οπτική αναπαράσταση του περιβάλλοντος χρήστη μιας εφαρμογής iOS, που δείχνει οθόνες περιεχομένου και τις συνδέσεις μεταξύ αυτών των οθονών. Ένα storyboard αποτελείται από μια σειρά σκηνών, καθένα από τα οποία αντιπροσωπεύει έναν ελεγκτή προβολής και τα view του. Οι ελεγκτές προβολής συνδέονται με αντικείμενα segue, τα οποία αντιπροσωπεύουν μια μετάβαση μεταξύ δύο ελεγκτών προβολής.

Το Xcode παρέχει ένα πρόγραμμα επεξεργασίας για storyboards, όπου γίνεται ο σχεδιασμός του περιβάλλοντος της εφαρμογής προσθέτοντας αντικείμενα του

πλαισίου UIKit χωρίς τη χρήση κώδικα. Επιπλέον, ένα storyboard δίνει τη δυνατότητα να συνδυάζεται ένα view με τον ελεγκτή προβολής και διαχειρίζεται τη μεταφορά δεδομένων μεταξύ ελεγκτών προβολής. Η χρήση storyboard είναι ο συνιστάμενος τρόπος για τον σχεδιασμό του περιβάλλοντος χρήστη της εφαρμογής, επειδή επιτρέπει να απεικονίζεται η εμφάνιση και η ροή του περιβάλλοντος χρήστη σε έναν καμβά.

Στο iPhone, κάθε σκηνή αντιστοιχεί σε περιεχόμενο πλήρους οθόνης. στο iPad, μπορούν να εμφανιστούν ταυτόχρονα πολλαπλές σκηνές στην οθόνη - για παράδειγμα, χρησιμοποιώντας ελεγκτές αναδυόμενων προβολών. Κάθε σκηνή έχει μία βάση, η οποία εμφανίζει εικονίδια που αντιπροσωπεύουν τα αντικείμενα ανώτατου επιπέδου της σκηνής. Η βάση χρησιμοποιείται κυρίως για τη δημιουργία συνδέσεων δράσης και εξόδου μεταξύ του ελεγκτή προβολή και των view που έχει [5].

View Controller

Οι ελεγκτές προβολής “View Controllers” αποτελούν τη βάση της εσωτερικής δομής της εφαρμογής. Κάθε εφαρμογή έχει τουλάχιστον έναν ελεγκτή προβολής και οι περισσότερες εφαρμογές έχουν πολλούς. Κάθε ελεγκτής προβολής διαχειρίζεται ένα τμήμα της διεπαφής χρήστη της εφαρμογής καθώς και τις αλληλεπιδράσεις μεταξύ αυτής της διασύνδεσης και των δεδομένων. Οι ελεγκτές προβολής διευκολύνουν επίσης τις μεταβάσεις μεταξύ διαφορετικών τμημάτων του περιβάλλοντος χρήστη.

Επειδή διαδραματίζουν έναν τόσο σημαντικό ρόλο στην εφαρμογή, οι ελεγκτές προβολής βρίσκονται στο επίκεντρο. Η κλάση UIViewController καθορίζει τις μεθόδους και τις ιδιότητες για τη διαχείριση των μεθόδων, το χειρισμό συμβάντων, τη μετάβαση από έναν ελεγκτή προβολής σε άλλο και το συντονισμό με άλλα μέρη της εφαρμογής.

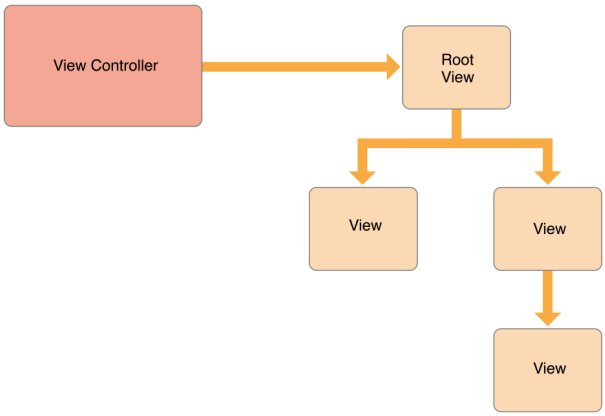
Υπάρχουν δύο τύποι ελεγκτών προβολής:

- Οι Content View Controller διαχειρίζονται ένα διακριτό κομμάτι του περιεχομένου της εφαρμογής και αποτελούν τον κύριο τύπο ελεγκτή προβολής που δημιουργείται.
- Οι Container View Controller συλλέγουν πληροφορίες από άλλους ελεγκτές προβολής (γνωστοί ως child view controllers) και παρουσιάζονται με τρόπο που διευκολύνει την πλοήγηση ή παρουσιάζει διαφορετικά το περιεχόμενο αυτών των ελεγκτών προβολής.

Οι περισσότερες εφαρμογές είναι ένα μείγμα και των δύο τύπων ελεγκτών προβολής.

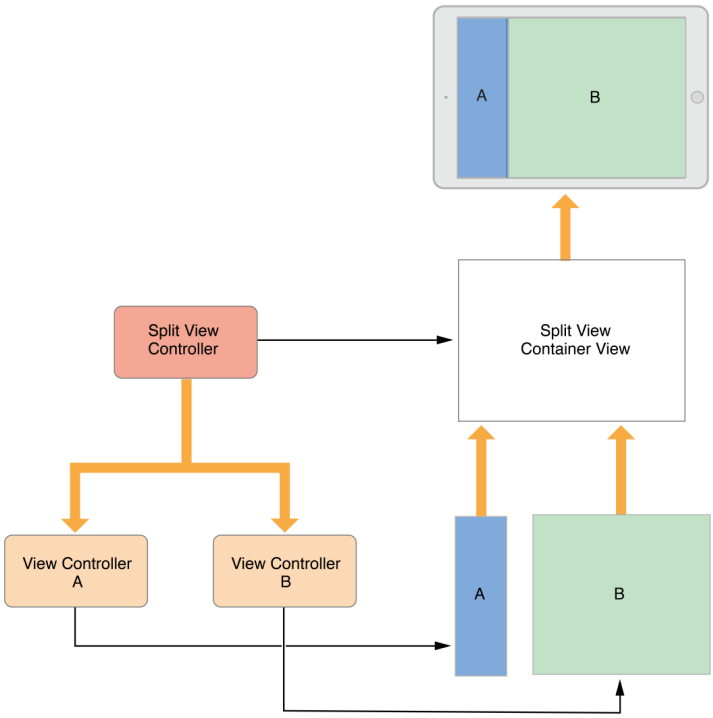
Διαχείριση προβολής

Ο σημαντικότερος ρόλος ενός ελεγκτή προβολής είναι η διαχείριση μιας ιεραρχίας των view. Κάθε ελεγκτής προβολής έχει ένα μοναδικό root view που περικλείει όλο το περιεχόμενο του ελεγκτή. Στη ριζική αυτή προβολή, προστίθενται οι προβολές που χρειάζονται για την εμφάνιση περιεχομένου. Το **σχήμα 1** απεικονίζει την ενσωματωμένη σχέση μεταξύ του ελεγκτή προβολής και των view του. Ο ελεγκτής προβολής έχει πάντοτε αναφορά στην ριζική του προβολή και κάθε προβολή έχει ισχυρές αναφορές στις υπό προβολές του.



Σχήμα 1: "Σχέση μεταξύ ενός ελεγκτή προβολής και των view του"

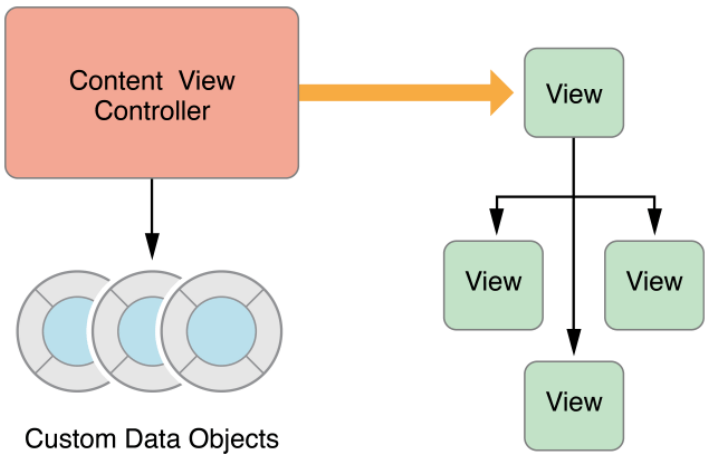
Ένας content view controller διαχειρίζεται όλα τα view του από μόνος του. Ένας container view controller διαχειρίζεται τα δικά του view και root view από έναν ή περισσότερους child view controllers. Το container δεν διαχειρίζεται το περιεχόμενο των παιδιών του. Διαχειρίζεται μόνο τη ριζική όψη, το μέγεθος και την τοποθέτηση σύμφωνα με το σχεδιασμό του container. Το **σχήμα 2** απεικονίζει τη σχέση μεταξύ ενός split view controller και των παιδιών του. Ο split view controller διαχειρίζεται το συνολικό μέγεθος και τη θέση των παιδικών view του, αλλά οι child view controllers διαχειρίζονται το πραγματικό περιεχόμενο αυτών των προβολών.



Σχήμα 2: "Οι ελεγκτές προβολής μπορούν να διαχειριστούν περιεχόμενο από άλλους ελεγκτές προβολής"

Διαχείριση δεδομένων

Ένας ελεγκτής προβολής ενεργεί ως ενδιάμεσος μεταξύ των view που διαχειρίζεται και των δεδομένων της εφαρμογής. Οι μέθοδοι και οι ιδιότητες της κλάσης UIViewController επιτρέπουν την διαχείριση της οπτικής παρουσίασης της εφαρμογής. Όταν γίνεται υποκλάση της κλάσης UIViewController, προστίθενται οι μεταβλητές που χρειάζονται για τη διαχείριση των δεδομένων της υποκλάσης. Η προσθήκη προσαρμοσμένων μεταβλητών δημιουργεί μια σχέση όπως η εικόνα στο **σχήμα 3**, όπου ο ελεγκτής προβολής έχει αναφορές στα δεδομένα και στις προβολές που χρησιμοποιούνται για την παρουσίαση αυτών των δεδομένων.



Σχήμα 3: "Ένας ελεγκτή προβολής διαμεσολαβεί μεταξύ Custom Data Object και View"

Πρέπει πάντα να διατηρείται ένας καθαρός διαχωρισμός ευθυνών στους ελεγκτές προβολής και τα αντικείμενα δεδομένων. Το μεγαλύτερο μέρος της λογικής για την εξασφάλιση της ακεραιότητας των δομών δεδομένων ανήκει στα ίδια τα αντικείμενα δεδομένων. Ο ελεγκτής προβολής μπορεί να επικυρώσει την είσοδο που προέρχεται από views και στη συνέχεια να πακετάρει εκείνη την είσοδο με τη μορφή που απαιτούν τα αντικείμενα δεδομένων, αλλά πρέπει να ελαχιστοποιείται ο ρόλος του στη διαχείριση των πραγματικών δεδομένων.

Ένα αντικείμενο UIDocument είναι ένας τρόπος για τη διαχείριση των δεδομένων ξεχωριστά από τους ελεγκτές προβολής. Ένα αντικείμενο εγγράφου είναι ένα αντικείμενο ελέγχου που γνωρίζει πώς να διαβάζει και να γράφει δεδομένα σε μόνιμη αποθήκευση. Όταν γίνει υποκλάση αυτού, προστίθεται οποιαδήποτε λογική και μέθοδος που χρειάζεται για να εξαχθούν αυτά τα δεδομένα και να μεταβιβαστούν σε έναν ελεγκτή προβολής ή σε άλλα μέρη της εφαρμογής. Ο ελεγκτής προβολής μπορεί να αποθηκεύσει ένα αντίγραφο των δεδομένων που λαμβάνει για να διευκολύνει την ενημέρωση των προβολών, αλλά το έγγραφο εξακολουθεί να κατέχει τα πραγματικά δεδομένα.

UITabBarController

Η διεπαφή Tab bar εμφανίζει καρτέλες στο κάτω μέρος του παραθύρου για επιλογή μεταξύ των διαφορετικών τρόπων λειτουργίας και εμφάνιση των προβολών για τη συγκεκριμένη λειτουργία. Αυτή η κατηγορία χρησιμοποιείται γενικά ως-είναι, αλλά μπορεί επίσης να είναι υποκλάση.

Κάθε καρτέλα διεπαφής UITabBarController συνδέεται με έναν προσαρμοσμένο ελεγκτή προβολής. Όταν ο χρήστης επιλέξει μια συγκεκριμένη καρτέλα, ο tab bar controller εμφανίζει το root view του αντίστοιχου ελεγκτή προβολής, αντικαθιστώντας οποιεσδήποτε προηγούμενες προβολές. Επειδή η επιλογή μιας καρτέλας αντικαθιστά τα περιεχόμενα της διεπαφής, ο τύπος διεπαφής που διαχειρίζεται δεν πρέπει να είναι παρόμοια με κανέναν τρόπο. Στην πραγματικότητα, οι διασυνδέσεις στη γραμμή καρτών χρησιμοποιούνται συνήθως είτε για την παρουσίαση διαφορετικών τύπων πληροφοριών είτε για την παρουσίαση των ίδιων πληροφοριών χρησιμοποιώντας ένα εντελώς διαφορετικό στυλ διεπαφής.

Αλληλεπιδράσεις χρηστών

Οι ελεγκτές προβολής είναι αντικείμενα απόκρισης και είναι σε θέση να χειριστούν συμβάντα που κατεβαίνουν στην αλυσίδα απόκρισης. Παρόλο που είναι σε θέση να το πράξουν, οι ελεγκτές προβολής σπάνια χειρίζονται άμεσα συμβάντα αφής. Αντίθετα, τα views συνήθως χειρίζονται τα δικά τους συμβάντα αφής και αναφέρουν τα αποτελέσματα σε μια μέθοδο ενός συνδεδεμένου αντικειμένου ή ενός αντικειμένου στόχου, που είναι συνήθως ο ελεγκτής προβολής. Επομένως, τα περισσότερα συμβάντα σε έναν ελεγκτή προβολής αντιμετωπίζονται χρησιμοποιώντας μεθόδους ή μεθόδους ανάθεσης.

Διαχείριση πόρων

Ένας ελεγκτής προβολής αναλαμβάνει κάθε ευθύνη για τα view και τα αντικείμενα που δημιουργεί. Η κλάση UIViewController χειρίζεται αυτόματα τις περισσότερες πτυχές της διαχείρισης προβολής. Για παράδειγμα, το UIKit απελευθερώνει αυτόματα πόρους που σχετίζονται με την προβολή και δεν χρειάζονται πλέον. Στις υποκλάσεις UIViewController, είναι υπεύθυνος ο developer για τη διαχείριση των αντικειμένων που δημιουργούνται.

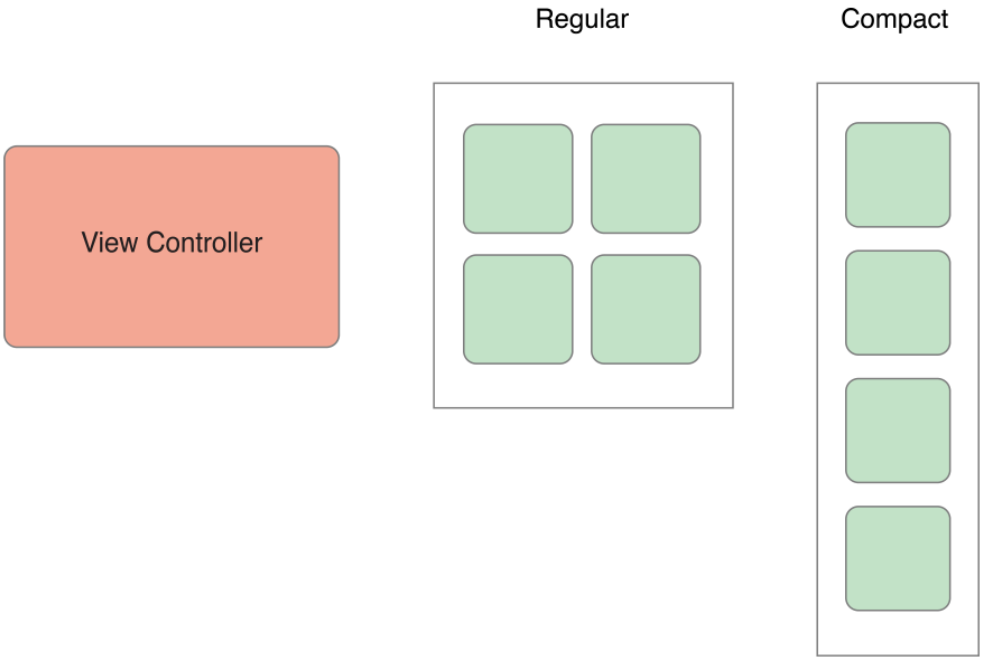
Όταν η διαθέσιμη ελεύθερη μνήμη είναι χαμηλή, το UIKit ζητά από τις εφαρμογές να απελευθερώσουν πόρους που δεν χρειάζονται πλέον. Ένας τρόπος με τον οποίο γίνεται αυτό είναι να κληθεί η μέθοδος didReceiveMemoryWarning των ελεγκτών προβολής. Αυτή η μέθοδος χρησιμοποιείται για την κατάργηση αναφορών σε αντικείμενα που δεν χρειάζονται πλέον ή μπορούν να αναδημιουργηθούν εύκολα αργότερα. Για παράδειγμα, γίνεται να χρησιμοποιηθεί αυτή η μέθοδος για την κατάργηση προσωρινών δεδομένων. Είναι σημαντικό να απελευθερωθεί όσο το δυνατόν περισσότερη μνήμη όταν συμβαίνει μια κατάσταση χαμηλής μνήμης. Οι εφαρμογές που καταναλώνουν υπερβολική μνήμη μπορεί να τερματιστούν απότομα από το σύστημα για να ανακτήσουν τη μνήμη.

Προσαρμοστικότητα

Οι ελεγκτές προβολής είναι υπεύθυνοι για την παρουσίαση των view τους και για την προσαρμογή αυτής της παρουσίασης ώστε να ταιριάζουν με το υποκείμενο περιβάλλον. Κάθε εφαρμογή iOS θα πρέπει να μπορεί να λειτουργεί σε iPad και σε πολλά διαφορετικά μεγέθη iPhone. Αντί να παρέχουν διαφορετικούς ελεγκτές προβολής και να προβάλλουν ιεραρχίες για κάθε συσκευή, είναι πιο εύκολο να χρησιμοποιείται ένας ελεγκτής προβολής που προσαρμόζει τα view του στις μεταβαλλόμενες απαιτήσεις χώρου.

Στο iOS, οι ελεγκτές προβολής πρέπει να χειρίζονται coarse-grained αλλαγές και fine-grained αλλαγές. Coarse-grained αλλαγές συμβαίνουν όταν αλλάζουν τα χαρακτηριστικά ενός ελεγκτή προβολής. Τα χαρακτηριστικά περιγράφουν το συνολικό περιβάλλον, όπως η κλίμακα προβολής. Δύο από τα πιο σημαντικά χαρακτηριστικά είναι οι οριζόντιες και κατακόρυφες κλάσεις μεγέθους του ελεγκτή

προβολής, οι οποίες δείχνουν πόση απόσταση έχει ο ελεγκτής προβολής στη δεδομένη διάσταση. Μπορούν να χρησιμοποιηθούν αλλαγές κατηγορίας μεγέθους για τον τρόπο με τον οποίο σχεδιάζονται τα Views, όπως φαίνεται στο **σχήμα 4** Όταν η οριζόντια τάξη μεγέθους είναι κανονική, ο ελεγκτής προβολής εκμεταλλεύεται τον επιπλέον οριζόντιο χώρο για να οργανώσει το περιεχόμενό του. Όταν η οριζόντια κλάση μεγέθους είναι συμπαγής, ο ελεγκτής προβολής οργανώνει το περιεχόμενό του κάθετα. [15]



Σχήμα 4: "Αλλαγές μεγέθους στα View"

Προσαρμογή προβολών σε αλλαγές τάξης μεγέθους

Μέσα σε μια δεδομένη τάξη μεγέθους, είναι πιθανό να εμφανιστούν ανά πάσα στιγμή περισσότερες αλλαγές μεγέθους. Όταν ο χρήστης περιστρέφει ένα iPhone από portrait σε lanscape, η κατηγορία μεγέθους μπορεί να μην αλλάζει, αλλά οι διαστάσεις της οθόνης συνήθως αλλάζουν. Όταν χρησιμοποιείται το Auto Layout, το UIKit προσαρμόζει αυτόματα το μέγεθος και τη θέση των προβολών ώστε να ταιριάζει με τις νέες διαστάσεις. Οι ελεγκτές προβολής μπορούν να κάνουν πρόσθετες ρυθμίσεις ανάλογα με τις ανάγκες.

Auto Layout

Το Auto Layout υπολογίζει δυναμικά το μέγεθος και τη θέση όλων των προβολών στην ιεραρχία του view, βάσει των περιορισμών που έχουν τεθεί. Για παράδειγμα, γίνεται να περιοριστεί ένα κουμπί έτσι ώστε να είναι οριζόντια κεντραρισμένο με μια προβολή εικόνας και έτσι ώστε το πάνω άκρο του κουμπιού να παραμένει πάντα 8 pixel κάτω από το κάτω μέρος της εικόνας. Εάν το μέγεθος ή η θέση της προβολής εικόνων αλλάξει, η θέση του κουμπιού προσαρμόζεται αυτόματα για να ταιριάζει.

Αυτή η προσέγγιση σχεδιασμού βάσει περιορισμών επιτρέπει την δημιουργία διεπαφών χρήστη που ανταποκρίνονται δυναμικά σε εσωτερικές και εξωτερικές αλλαγές.

Foundation

Το πλαίσιο Foundation παρέχει ένα βασικό επίπεδο λειτουργικότητας για εφαρμογές και πλαίσια, όπως αποθήκευση, επεξεργασία κειμένου, υπολογισμοί ημερομηνίας και ώρας, ταξινόμηση, φιλτράρισμα και δικτύωση. Οι κλάσεις, τα πρωτόκολλα και οι τύποι δεδομένων που ορίζονται από το Foundation χρησιμοποιούνται σε όλα τα SDK των macOS, iOS, watchOS και tvOS.

UIView

Τα αντικείμενα της κλάσεις UIView είναι οι θεμελιώδεις δομικές μονάδες της διεπαφής χρήστη μίας εφαρμογής και η κλάση UIView καθορίζει τις συμπεριφορές που είναι κοινές σε όλα τα αντικείμενα UIView καθώς και των υποκλάσεων της. Ένα αντικείμενο UIView καθιστά το περιεχόμενο εντός των ορίων του ορθογωνίου και χειρίζεται οποιεσδήποτε αλληλεπιδράσεις με αυτό το περιεχόμενο. Η κλάση UIView είναι μια συγκεκριμένη κλάση που μπορεί να εμφανίζεται και να χρησιμοποιείται για να εμφανιστεί ένα σταθερό χρώμα φόντου. Μπορεί επίσης να υποκατασταθεί για να σχεδιαστεί πιο περίπλοκο περιεχόμενο. Για την εμφάνιση ετικετών, εικόνων, κουμπιών και άλλων στοιχείων διεπαφής που βρίσκονται συνήθως σε εφαρμογές, χρησιμοποιούνται οι υποκλάσεις της κλάσης UIView που παρέχονται από το πλαίσιο UIKit.

Επειδή τα αντικείμενα UIView είναι ο κύριος τρόπος αλληλεπίδρασης της εφαρμογής με τον χρήστη, έχουν ορισμένες ευθύνες όπως:

- Σχεδιασμός και Animation
    - Τα αντικείμενα κλάσης UIView σχεδιάζουν περιεχόμενο στην ορθογώνια περιοχή τους χρησιμοποιώντας το UIKit ή Core Graphics.
    - Ορισμένες ιδιότητες μπορούν να γίνονται animated σε νέες τιμές.
- Διαχείριση διάταξης και των subviews
    - Οι προβολές ενδέχεται να περιέχουν και περισσότερες υπό προβολές.
    - Οι προβολές μπορούν να προσαρμόσουν το μέγεθος και τη θέση των υπό προβολών τους.
    - Χρησιμοποιείται το Auto Layout για να οριστούν οι κανόνες για την αλλαγή μεγέθους και την επανατοποθέτηση των προβολών σε σχέση με αλλαγές στην ιεραρχία των προβολών.
- Διαχείριση συμβάντων
    - Μία προβολή είναι μια υποκλάση της κλάσης UIResponder και μπορεί να ανταποκριθεί σε αγγίγματα στην οθόνη και άλλα είδη συμβάντων.

- Οι προβολές μπορούν να εγκαταστήσουν αναγνωριστές χειρονομιών για να χειριστούν τις κοινές χειρονομίες.

Οι προβολές μπορούν να ενσωματωθούν μέσα σε άλλες προβολές για να δημιουργηθούν ιεραρχίες προβολών, οι οποίες προσφέρουν έναν βολικό τρόπο για την οργάνωση σχετικού περιεχομένου. Η ανίχνευση μίας προβολής δημιουργεί μια σχέση γονέα-παιδιού μεταξύ της εμφάνισης παιδιού που είναι ενωμένη (subview) και του γονέα (superview). Μια γονική προβολή μπορεί να περιέχει οποιονδήποτε αριθμό υπό προβολών αλλά κάθε υπο προβολή έχει μόνο ένα γονέα. Από προεπιλογή, όταν η ορατή περιοχή μίας υποπροβολής εκτείνεται εκτός των ορίων του, δεν υπάρχει αποκοπή του περιεχομένου της υπο προβολής. Υπάρχει η δυνατότητα να χρησιμοποιηθεί η ιδιότητα clipsToBounds για να αλλάξει η συμπεριφορά αυτή.

Η γεωμετρία κάθε προβολής ορίζεται από τις ιδιότητες frame και bounds. Η ιδιότητα frame ορίζει την προέλευση και τις διαστάσεις της προβολής στο σύστημα συντεταγμένων της προβολής γονέα του. Η ιδιότητα bounds ορίζει τις εσωτερικές διαστάσεις του view. Η ιδιότητα center παρέχει έναν βολικό τρόπο για να επανατοποθετείται μια προβολή χωρίς να αλλάζει απευθείας η ιδιότητα frame ή bounds.

Animations

Τα Animation παρέχουν οπτικές μεταβάσεις μεταξύ διαφορετικών καταστάσεων της διεπαφής χρήστη. Στο iOS, οι κινούμενες εικόνες χρησιμοποιούνται εκτεταμένα για να επανατοποθετήσουν τις προβολές, να αλλάξουν το μέγεθός τους, να τις αφαιρέσουν από ιεραρχίες προβολής και να τις αποκρύψουν. Μπορούν να χρησιμοποιηθούν κινούμενα σχέδια για την εμφάνιση σχολίων προς το χρήστη ή για την εφαρμογή οπτικών εφέ.

Στο iOS, η δημιουργία animation δεν απαιτεί γραφή κώδικα σχεδίασης. Όλες οι τεχνικές κινούμενης εικόνας που χρησιμοποιούν την ενσωματωμένη υποστήριξη που παρέχεται από το Core Animation. Το μόνο που χρειάζεται είναι να ενεργοποιείται το animation και το Core Animation διαχειρίζεται την απόδοση μεμονωμένων frame. Αυτό κάνει πολύ εύκολη τη δημιουργία έξυπνων κινούμενων εικόνων με λίγες μόνο γραμμές κώδικα.

Και το UIKit και το Core Animation παρέχουν υποστήριξη για animation, αλλά το επίπεδο υποστήριξης που παρέχεται από κάθε τεχνολογία ποικίλλει. Στο UIKit, τα animation πραγματοποιούνται χρησιμοποιώντας αντικείμενα UIView. Τα αντικείμενα UIView υποστηρίζουν ένα βασικό σύνολο κινούμενων σχεδίων που καλύπτουν πολλές κοινές εργασίες. Για παράδειγμα, μπορούν να γίνουν αλλαγές στις ιδιότητες των προβολών ή να χρησιμοποιηθούν animation για την αντικατάσταση ενός συνόλου από προβολές με ένα άλλο.

ΕΠΙΛΟΓΟΣ

Αυτό το κεφάλαιο προσφέρει τις απαραίτητες γνώσεις που χρειάζεται κάποιος developer για να καταλάβει πως γίνεται ο προγραμματισμός της εφαρμογής που θα εξηγηθεί παρακάτω καθώς και κάθε βασική γνώση που χρειάζεται κάποιος για να να κατανοήσει τον τρόπο με τον οποίο προγραμματίζονται οι εφαρμογές για συσκευές με λειτουργικό σύστημα iOS.

ΚΕΦΑΛΑΙΟ 2

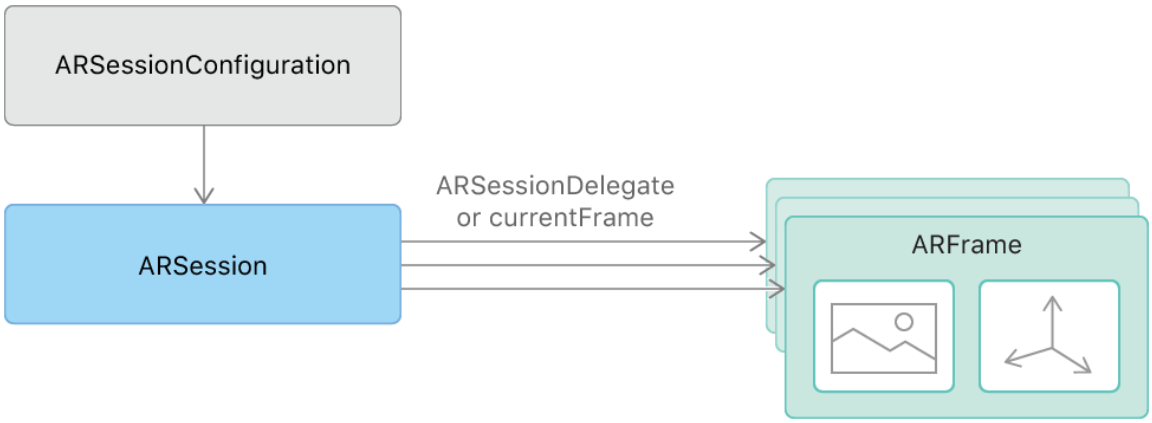
Augmented Reality with Apple

ΕΙΣΑΓΩΓΗ

Για την υποστήριξη επαυξημένης πραγματικότητας η Apple δημιούργησε το πλαίσιο ArKit. Το ArKit συνδυάζει την παρακολούθηση κίνησης της συσκευής, τη λήψη της σκηνής της κάμερας, την προηγμένη επεξεργασία σκηνών και την ευκολία προβολής για να απλοποιήσει το έργο της δημιουργίας μιας εμπειρίας AR. Είναι εφικτό να χρησιμοποιηθούν αυτές οι τεχνολογίες για τη δημιουργία πολλών εμπειριών AR χρησιμοποιώντας είτε την οπίσθια κάμερα είτε την μπροστινή κάμερα μιας συσκευής iOS.

ARKit με Metal

Το ARKit περιλαμβάνει κλάσεις για εύκολη προβολή των εμπειριών AR με το SceneKit ή το SpriteKit. Ωστόσο, αν δημιουργηθεί από τον προγραμματιστή μία άλλη μηχανή απόδοσης (ή ενσωματωθεί με έναν μηχανισμό τρίτου κατασκευαστή), το ARKit παρέχει επίσης όλη την υποστήριξη που απαιτείται για την εμφάνιση μιας εμπειρίας AR με μια προσαρμοσμένη προβολή.



Σχήμα 5: "ARSessionConfiguration cycle"

Σε οποιαδήποτε εμπειρία AR, το πρώτο βήμα είναι η διαμόρφωση ενός αντικειμένου ARSession για τη διαχείριση της λήψης κάμερας και της επεξεργασίας κίνησης. Μια συνεδρία προσδιορίζει και διατηρεί μια αντιστοιχία

μεταξύ του πραγματικού χώρου που κατοικεί η συσκευή και ενός εικονικού χώρου στον οποίο διαμορφώνεται περιεχόμενο AR. Για την εμφάνιση μίας εμπειρίας AR σε μια προσαρμοσμένη προβολή, χρειάζεται:

1. Η ανάκτηση video frames και πληροφοριών παρακολούθησης από την περίοδο λειτουργίας.
2. Η αναπαραγωγή αυτών των frame εικόνων ως φόντο για την προβολή.
3. Η χρησιμοποίηση των πληροφοριών παρακολούθησης για την τοποθέτηση και τον σχεδιασμό περιεχομένου AR πάνω από την εικόνα της κάμερας.

Ανάκτηση των video frames και τα δεδομένα παρακολούθησης από το Session

Αρχικά πρέπει να δημιουργηθεί και να διατηρηθεί η αναφορά ενός αντικειμένου ARSession και να εκτελεστεί με ένα session configuration κατάλληλο για το είδος της εμπειρίας AR που πρόκειται να υποστηριχτεί. Το session καταγράφει βίντεο από την κάμερα, παρακολουθεί τη θέση και τον προσανατολισμό της συσκευής σε μοντέλο 3D χώρου και παρέχει αντικείμενα ARFrame. Κάθε τέτοιο αντικείμενο περιέχει τόσο εικόνα ατομικής εικόνας frame βίντεο όσο και πληροφορίες εντοπισμού θέσης από τη στιγμή που καταγράφηκε το frame.

Υπάρχουν δύο τρόποι για την απόκτηση πρόσβασης σε αντικείμενα ARFrame που παράγονται από μια συνεδρία AR, ανάλογα με το αν η εφαρμογή ευνοεί ένα μοτίβο έλξης ή ένα push design.

Για τον έλεγχο του frame timing (pull design pattern), χρησιμοποιείται η ιδιότητα currentFrame της περιόδου σύνδεσης για τη λήψη της τρέχουσας εικόνας καρέ και πληροφορίες παρακολούθησης κάθε φορά που επανασχεδιάζονται τα περιεχόμενα της οθόνης. Το πλαίσιο ARKit χρησιμοποιεί αυτήν την προσέγγιση.

Εναλλακτικά, εάν ο σχεδιασμός της εφαρμογής ευνοεί ένα μοτίβο ώθησης, εφαρμόζεται η μέθοδος της συνόδου (_: didUpdate :) και η συνεδρία θα την καλέσει μία φορά για κάθε καρέ βίντεο που συλλαμβάνει (στα 60 καρέ ανά δευτερόλεπτο από προεπιλογή).

Μόλις αποκτηθεί ένα καρέ, χρειάζεται να σχεδιαστεί η εικόνα της κάμερας, να ενημερωθεί και να προβληθεί οποιοδήποτε περιεχόμενο επικάλυψης που περιλαμβάνεται στη AR εμπειρία.

Σχεδίαση της εικόνας της κάμερας

Κάθε ιδιότητα capturedImage του αντικειμένου ARFrame περιέχει ένα buffer pixel που έχει ληφθεί από τη φωτογραφική μηχανή της συσκευής. Για τον σχεδιασμό αυτής της εικόνας ως φόντο για την προσαρμοσμένη προβολή, χρειάζεται η δημιουργία υφών από το περιεχόμενο της εικόνας και η υποβολή εντολών εμφάνισης GPU, που χρησιμοποιεί αυτές τις υφές.

Τα pixel buffer’s κωδικοποιούνται σε μια διπολική μορφή δεδομένων YCbCr (επίσης αποκαλούμενη YUV). Για την προβολή της εικόνας χρειάζεται η μετατροπή αυτών των δεδομένων pixel σε μια σχετική μορφή RGB. Για render με το πλαίσιο Metal, εκτελείται αυτή η μετατροπή πιο αποτελεσματικά στον κώδικα shader της GPU. Χρησιμοποιούνται τα API CVMetalTextureCache για τη δημιουργία δύο υφών από το buffer pixel, ένα το καθένα για τα επίπεδα luma (Y) και chroma (CbCr) του buffer.

Στη συνέχεια, κωδικοποιούνται οι εντολές εμφάνισης που σχεδιάζουν αυτές τις δύο υφές χρησιμοποιώντας μία λειτουργία θραύσματος που εκτελεί μετατροπή YCbCr σε RGB με ένα μετασχηματισμό χρώματος.

Παρακολούθηση και αναπαραγωγή περιεχομένου επικάλυψης

Οι εμπειρίες AR συνήθως επικεντρώνονται στην απόδοση περιεχομένου 3D επικάλυψης έτσι ώστε το περιεχόμενο να φαίνεται ότι είναι μέρος του πραγματικού κόσμου που εμφανίζεται στην εικόνα της κάμερας. Για να επιτευχθεί αυτή η ψευδαίσθηση, χρησιμοποιείται η κλάση ARAnchor για τη μοντελοποίηση της θέσης και του προσανατολισμού ενός 3D περιεχομένου σε σχέση με τον πραγματικό χώρο.

Προβολή με Ρεαλιστικό φωτισμό

Όταν ρυθμίζονται τα shaders για σχεδίαση περιεχομένου 3D στη σκηνή, γίνεται να χρησιμοποιηθούν εκτιμώμενες πληροφορίες φωτισμού σε κάθε αντικείμενο ARFrame για να παράγεται πιο ρεαλιστική σκίαση.

AR WorldTracking Configuration

Το ARWorldTrackingConfiguration παρέχει την εμπειρία στην οποία εμφανίζεται μια προβολή από μια κάμερα, που προστίθεται άλλο οπτικό περιεχόμενο. Οι ARKit χάρτες και κομμάτια του πραγματικού χώρου που φιλοξενεί ο χρήστης και το ταιριάζει με έναν συντονισμένο χώρο για να τοποθετηθεί εικονικό περιεχόμενο. Η παγκόσμια παρακολούθηση προσφέρει επίσης δυνατότητες για να αποκατασταθούν οι εμπειρίες AR, όπως η αναγνώριση αντικειμένων και εικόνων στο περιβάλλον του χρήστη και η ανταπόκριση σε πραγματικές συνθήκες φωτισμού.

Όλες οι διαμορφώσεις AR δημιουργούν μια αντιστοιχία μεταξύ του πραγματικού κόσμου που κατοικεί η συσκευή και ενός εικονικού χώρου συντονισμού 3D όπου μπορεί να διαμορφωθεί περιεχόμενο. Όταν η εφαρμογή εμφανίζει αυτό το περιεχόμενο μαζί με μια ζωντανή εικόνα κάμερας, ο χρήστης βιώνει την ψευδαίσθηση ότι το εικονικό περιεχόμενο είναι μέρος του πραγματικού κόσμου.

Η δημιουργία και η διατήρηση αυτής της αντιστοιχίας μεταξύ χώρων απαιτεί την παρακολούθηση της κίνησης της συσκευής. Η κλάση ARWorldTrackingConfiguration παρακολουθεί την κίνηση της συσκευής με έξι

βαθμούς ελευθερίας: συγκεκριμένα, τρεις άξονες (κίνηση σε x, y και z) και οι τρεις άξονες περιστροφής (δηλαδή περιστροφή ως προς x, y, z).

Αυτό το είδος παρακολούθησης μπορεί να δημιουργήσει εμπειρίες AR: Ένα εικονικό αντικείμενο μπορεί να φαίνεται ότι παραμένει στο ίδιο σημείο σε σχέση με τον πραγματικό κόσμο, ακόμα και όταν ο χρήστης κλίνει τη συσκευή για να κοιτάξει πάνω ή κάτω από το αντικείμενο ή κινεί τη συσκευή γύρω για να δει τις πλευρές και την πλάτη του αντικειμένου.

Οι World-tracking sessions παρέχουν επίσης διάφορους τρόπους με τους οποίους η εφαρμογή αναγνωρίζει ή αλληλοεπιδρά με στοιχεία της πραγματικής σκηνής που είναι ορατά στη camera:

Είναι εφικτό να:

- χρησιμοποιηθεί η λειτουργία planeDetection για να βρεθούν οριζόντιες ή κάθετες επιφάνειες πραγματικού κόσμου, προσθέτοντάς τις στην περίοδο λειτουργίας ως αντικείμενα ARPlaneAnchor.
- χρησιμοποιηθεί η λειτουργία detectImages για να αναγνωριστούν και να παρακολουθηθεί η κίνηση γνωστών 2D εικόνων, προσθέτοντάς τα στη σκηνή ως αντικείμενα ARImageAnchor.
- χρησιμοποιηθεί το detectObjects για να αναγνωρίσετε γνωστά αντικείμενα 3D, προσθέτοντάς τα στη σκηνή ως αντικείμενα ARObjectAnchor.
- χρησιμοποιηθούν hit-testing methods σε ARFrame, ARSCNView ή ARSKView για να βρεθούν οι 3D θέσεις των χαρακτηριστικών πραγματικού κόσμου που ανταποκρίνονται σε ένα σημείο 2D στην προβολή της κάμερας.

AR Face Tracking Configuration

Μια AR Face Tracking Configuration ανιχνεύει το πρόσωπο του χρήστη σε σχέση με την κάμερα που βλέπει προς τα εμπρός. Όταν εκτελείται αυτή η ρύθμιση, μια συνεδρία AR εντοπίζει το πρόσωπο του χρήστη (εάν είναι ορατό στην εικόνα της μπροστινής κάμερας) και προσθέτει στη λίστα των anchors του ένα αντικείμενο ARFaceAnchor που αντιπροσωπεύει το πρόσωπο. Κάθε anchor προσώπου παρέχει πληροφορίες σχετικά με τη θέση και τον προσανατολισμό του προσώπου, την τοπολογία του και τις λειτουργίες που περιγράφουν τις εκφράσεις του προσώπου. Η παρακολούθηση προσώπου είναι διαθέσιμη μόνο σε συσκευές iOS με front camera TrueDepth.

Η κλάση ARFaceTrackingConfiguration δεν παρέχει μεθόδους ή ιδιότητες, αλλά υποστηρίζει όλες τις ιδιότητες που κληρονομούνται από την ARConfiguration της υπερκλάσης. Επιπλέον, όταν ενεργοποιείται η ρύθμιση isLightEstimationEnabled, η διαμόρφωση παρακολούθησης προσώπου χρησιμοποιεί το ανιχνευμένο

πρόσωπο ως αισθητήρα φωτός και παρέχει μια εκτίμηση του κατευθυντικού ή περιβαλλοντικού φωτισμού (αντικείμενο ARDirectionalLightEstimate).

AR Orientation Tracking Configuration

Όλες οι διαμορφώσεις AR δημιουργούν μια αντιστοιχία μεταξύ του πραγματικού κόσμου που κατοικεί η συσκευή και ενός εικονικού χώρου συντονισμού 3D όπου μπορεί να διαμορφωθεί περιεχόμενο. Όταν η εφαρμογή εμφανίζει αυτό το περιεχόμενο μαζί με μια ζωντανή εικόνα κάμερας, ο χρήστης βιώνει την ψευδαίσθηση ότι το εικονικό περιεχόμενο είναι μέρος του πραγματικού κόσμου.

Η δημιουργία και η διατήρηση αυτής της αντιστοιχίας μεταξύ χώρων απαιτεί την παρακολούθηση της κίνησης της συσκευής. Η κλάση AROrientationTrackingConfiguration παρακολουθεί την κίνηση της συσκευής με τρεις βαθμούς ελευθερίας : συγκεκριμένα οι τρεις άξονες περιστροφής (δηλαδή περιστροφή ως προς x, y, z).

Αυτό το βασικό επίπεδο παρακολούθησης κινήσεων μπορεί να δημιουργήσει περιορισμένες εμπειρίες AR: Ένα εικονικό αντικείμενο μπορεί να φαίνεται ότι είναι μέρος του πραγματικού κόσμου, ακόμα και όταν ο χρήστης περιστρέφει τη συσκευή για να κοιτάξει πάνω, κάτω ή δίπλα από αυτό το αντικείμενο. Ωστόσο, αυτή η διαμόρφωση δεν μπορεί να παρακολουθήσει την κίνηση της συσκευής: η μη εναλλακτική αλλαγή της θέσης της συσκευής σπάει την ψευδαίσθηση AR, προκαλώντας την εμφάνιση εικονικού περιεχομένου σε παρασυρόμενα επίπεδα σε σχέση με τον πραγματικό κόσμο. Για παράδειγμα, ο χρήστης δεν μπορεί να περπατήσει για να δει τις πλευρές και το πίσω μέρος ενός εικονικού αντικειμένου. Επιπλέον δεν υποστηρίζει plane detection ή hit testing. Χρησιμοποιείται ως εναλλακτική λύση σε καταστάσεις όπου η ARWorldTrackingConfiguration είναι προσωρινά μη διαθέσιμη.

ARImageTrackingConfiguration

Η ARImageTrackingConfiguration είναι διαμόρφωση που χρησιμοποιεί την πίσω κάμερα για την ανίχνευση και την παρακολούθηση γνωστών εικόνων.

Με το ARImageTrackingConfiguration, το ARKit δημιουργεί έναν τρισδιάστατο χώρο όχι παρακολουθώντας την κίνηση της συσκευής σε σχέση με τον κόσμο, αλλά μόνο εντοπίζοντας και παρακολουθώντας την κίνηση των γνωστών 2D εικόνων εν όψει της κάμερας. Η ARWorldTrackingConfiguration μπορεί επίσης να ανιχνεύσει εικόνες, αλλά κάθε διαμόρφωση έχει τα δικά της πλεονεκτήματα:

Η παγκόσμια παρακολούθηση έχει υψηλότερο κόστος από την παρακολούθηση μόνο για την εικόνα, επομένως η συνεδρία μπορεί να παρακολουθεί αξιόπιστα περισσότερες εικόνες ταυτόχρονα με το ARImageTrackingConfiguration.

Η παρακολούθηση μόνο εικόνας επιτρέπει την πρόσδεση εικονικού περιεχομένου σε γνωστές εικόνες μόνο όταν οι εικόνες αυτές έχουν θέα στην κάμερα. Η παγκόσμια παρακολούθηση με ανίχνευση εικόνων επιτρέπει να χρησιμοποιούνται γνωστές εικόνες για την προσθήκη εικονικού περιεχομένου στον κόσμο 3D και συνεχίζει να παρακολουθεί τη θέση αυτού του περιεχομένου στον παγκόσμιο χώρο ακόμα και μετά την απεικόνιση της εικόνας.

Όταν μια διαμόρφωση παρακολούθησης εικόνας ανιχνεύει γνωστές εικόνες, παρακολουθεί την κίνηση τους σε τρεις άξονες (κίνηση σε x, y και z) και τρεις άξονες περιστροφής (δηλαδή περιστροφή ως προς x, y, z).

Για να χρησιμοποιηθεί το ARImageTrackingConfiguration, ορίζονται αντικείμενα ARReferenceImage (είτε κατά τη διάρκεια εκτέλεσης, είτε με τη συσχέτιση τους στον κατάλογο asset resources του Xcode) και ορίζονται σαν τιμή στην ιδιότητα trackingImages της διαμόρφωσης.

ARObjectScanningConfiguration

Η AR Object Scanning Configuration είναι μια διαμόρφωση που χρησιμοποιεί την πίσω κάμερα για τη συλλογή χωρικών δεδομένων υψηλής πιστότητας για χρήση στην ανίχνευση ή σάρωση τρισδιάστατων αντικειμένων.

Για την ανίχνευση ενός πραγματικού 3D αντικειμένου σε μια εμπειρία AR, το ARWorldTrackingConfiguration χρειάζεται μια υψηλής πιστότητας τρισδιάστατη σάρωση του αντικειμένου - ένα ARReferenceObject. Η εκτέλεση μιας περιόδου σύνδεσης με το ARObjectScanningConfiguration επιτρέπει τη συλλογή δεδομένων υψηλής πιστότητας που απαιτείται για τη σάρωση αντικειμένων. Αφού σαρωθεί ένα αντικείμενο σε μια περίοδο λειτουργίας με αυτήν τη διαμόρφωση, καλείται η μέθοδος createReferenceObject για να γίνει η εξαγωγή μίας περιοχής των δεδομένων εσωτερικής χωρικής χαρτογράφησης της περιόδου σύνδεσης για χρήση ως αντικείμενο αναφοράς.

Το ARObjectScanningConfiguration προορίζεται για χρήση μόνο σε αναπτυξιακά σενάρια. Η χωρική χαρτογράφηση υψηλής πιστότητας έχει υψηλό κόστος απόδοσης και ενέργειας και απενεργοποιεί τις λειτουργίες ARKit που δεν είναι απαραίτητες για τη σάρωση αντικειμένων αναφοράς. Για την παραγωγή εμπειριών AR για τους τελικούς χρήστες, χρησιμοποιείται το ARWorldTrackingConfiguration.

Εκτός από τη δυνατότητα ανίχνευσης αντικειμένων, το AR Object Scanning Configuration είναι παρόμοιο με το ARWorldTrackingConfiguration: παρακολουθείται τόσο η θέση της συσκευής με τον προσανατολισμό όσο και ο

προσανατολισμός με έξι βαθμούς ελευθερίας (6DOF) και υποστηρίζει plane detection και hit testing. Για την ενεργοποίηση της σάρωσης αντικειμένων υψηλότερης πιστότητας, οι συνεδρίες ανίχνευσης αντικειμένων παραλείπουν άλλες λειτουργίες της world-tracking session.

ARAnchor

Μια πραγματική θέση με προσανατολισμό που μπορεί να χρησιμοποιηθεί για την τοποθέτηση αντικειμένων σε μια σκηνή AR.

Για την παρακολούθηση στατικών θέσεων με προσανατολισμούς πραγματικών ή εικονικών αντικειμένων σε σχέση με την κάμερα, πρέπει να δημιουργηθούν αντικείμενα anchor “άγκυρες” και να χρησιμοποιηθεί η μέθοδος **add(anchor:)** για να προστεθούν στην συνεδρία AR.

Η προσθήκη μίας άγκυρας στη συνεδρία βοηθά το ARKit να βελτιστοποιήσει την ακρίβεια παρακολούθησης θέσεων στην περιοχή γύρω από την άγκυρα, έτσι ώστε τα εικονικά αντικείμενα να φαίνεται ότι παραμένουν στη θέση τους σε σχέση με τον πραγματικό κόσμο.

Ορισμένες λειτουργίες του ARKit προσθέτουν αυτόματα πρόσθετες άγκυρες σε μια συνεδρία. Οι συνεδρίες παγκόσμιας παρακολούθησης μπορούν να προσθέσουν αντικείμενα των κλάσεων ARPlaneAnchor, ARObjectAnchor και ARImageAnchor αν έχουν ενεργοποιηθεί οι αντίστοιχες λειτουργίες, οι συνεδρίες Face-tracking προσθέτουν αντικείμενα ARFaceAnchor.

3d Scenekit

Το SceneKit συνδυάζει μια μηχανή απόδοσης υψηλής απόδοσης με ένα περιγραφικό API για εισαγωγή, χειρισμό και προβολή τρισδιάστατων στοιχείων. Σε αντίθεση με τα API χαμηλότερου επιπέδου όπως το Metal και το OpenGL που απαιτούν την εφαρμογή των αλγορίθμων απόδοσης που εμφανίζουν μια σκηνή, το SceneKit απαιτεί μόνο περιγραφές του περιεχομένου της σκηνής και των ενεργειών ή των κινούμενων εικόνων που πρόκειται να εκτελέσει.

SCNView

Στο iOS και στο tvOS, το SCNView είναι μια υποκατηγορία του UIView. Ως τμήμα της ιεραρχίας προβολής του κάθε λειτουργικού συστήματος, ένα αντικείμενο SCNView παρέχει μια θέση για το περιεχόμενο SceneKit στο περιβάλλον εργασίας χρήστη της εφαρμογής. Μπορεί να δημιουργηθεί μία προβολή SceneKit χρησιμοποιώντας τη μέθοδο **init (frame: options :)** ή προσθέτοντάς την σε ένα αρχείο nib ή storyboard. Για την παροχή περιεχομένου για μια προβολή SceneKit, αντιστοιχείται ένα αντικείμενο SCNScene στην ιδιότητα της σκηνής.

ARSCNView

Μια προβολή για την αναπαράσταση εμπειριών AR που επαυξάνουν το περιεχόμενο που προβάλλεται από την κάμερα με περιεχόμενο 3D SceneKit.

Η κλάση ARSCNView παρέχει τον ευκολότερο τρόπο δημιουργίας εμπειριών επαυξημένης πραγματικότητας που συνδυάζουν εικονικό περιεχόμενο 3D με προβολή κάμερας συσκευής του πραγματικού κόσμου.

Η προβολή πραγματοποιεί αυτόματα τη ζωντανή ροή βίντεο από την κάμερα της συσκευής ως παρασκήνιο της σκηνής. Το παγκόσμιο σύστημα συντεταγμένων της σκηνής SceneKit της οθόνης ανταποκρίνεται άμεσα στο σύστημα συντεταγμένων του κόσμου AR που δημιουργήθηκε από τη διαμόρφωση της σύνδεσης. Η προβολή μετακινεί αυτόματα τη κάμερα της σκηνής του SceneKit ώστε να ταιριάζει με την πραγματική κίνηση της συσκευής.

Επειδή το ARKit ταιριάζει αυτόματα το χώρο SceneKit με τον πραγματικό κόσμο, τοποθετώντας ένα εικονικό αντικείμενο έτσι ώστε να φαίνεται ότι διατηρεί μια θέση σε πραγματικό κόσμο απαιτεί μόνο τη σωστή ρύθμιση της θέσης SceneKit του αντικειμένου.

Δεν χρειάζεται απαραιτήτως να χρησιμοποιηθεί η κλάση ARAnchor για την παρακολούθηση των θέσεων των αντικειμένων που προστίθενται στη σκηνή, αλλά με την εφαρμογή των μεθόδων του πρωτοκόλλου ARSCNViewDelegate, γίνεται να προστίθεται περιεχόμενο SceneKit σε οποιαδήποτε anchor που εντοπίζονται αυτόματα από το ARKit.

Παροχή τρισδιάστατου εικονικού περιεχομένου με το SceneKit

Επειδή το ARKit αντιστοιχεί αυτόματα το χώρο SceneKit στον πραγματικό κόσμο, τοποθετώντας ένα εικονικό αντικείμενο έτσι ώστε να φαίνεται ότι διατηρεί μια θέση σε πραγματικό κόσμο απαιτεί να ρυθμιστεί σωστά η θέση του αντικειμένου στο SceneKit. Για παράδειγμα, σε μια προεπιλεγμένη διαμόρφωση, ο παρακάτω κώδικας τοποθετεί έναν κύβο 10 εκατοστών 20 εκατοστά μπροστά από την αρχική θέση της κάμερας:

```
let cubeNode = SCNNode(geometry: SCNBox(width: 0.1, height: 0.1,
length: 0.1, chamferRadius: 0))
cubeNode.position = SCNVector3(0, 0, -0.2) // SceneKit AR
coordinates are in meters
sceneView.scene.rootNode.addChildNode(cubeNode)
```

Κώδικας 1: "Προσθήκη αντικειμένου κλάσης SNCNode σε σκηνή"

Ο παραπάνω κώδικας τοποθετεί ένα αντικείμενο απευθείας στη σκηνή SceneKit της οθόνης. Το αντικείμενο εμφανίζεται αυτόματα για να παρακολουθήσει μια πραγματική θέση, επειδή το ARKit αντιστοιχεί τον χώρο του SceneKit στον πραγματικό χώρο.

Εναλλακτικά, γίνεται να χρησιμοποιηθούν αντικείμενα της κλάσης ARAnchor για την παρακολούθηση θέσεων πραγματικού κόσμου, είτε δημιουργώντας anchor προγραμματιστικά και προσθέτοντάς τους στη σύνοδο, είτε παρατηρώντας anchors που έχουν δημιουργηθεί αυτόματα από το ARKit. Για παράδειγμα, όταν είναι ενεργοποιημένο το plane detection, το ARKit προσθέτει και ενημερώνει anchors για κάθε επίπεδο ανίχνευσης. Για την προσθήκη οπτικού περιεχομένου για αυτές τις άγκυρες, εφαρμόζονται οι μέθοδοι του ARSCNViewDelegate όπως είναι οι εξής:

```
func renderer(_ renderer: SCNSceneRenderer, didAdd node: SCNNode,
for anchor: ARAnchor) {
```

- **Αυτή η απεικόνιση καλύπτει μόνο ανιχνευόμενα επίπεδα**

```
  guard let planeAnchor = anchor as? ARPlaneAnchor else { return }
```

- **Δημιουργείται ένα επίπεδο SceneKit για την απεικόνιση του κόμβου χρησιμοποιώντας τη θέση και την έκταση του**.

```
let plane = SCNPlane(width: CGFloat(planeAnchor.extent.x), height:
CGFloat(planeAnchor.extent.z))
    let planeNode = SCNNode(geometry: plane)
    planeNode.position = SCNVector3Make(planeAnchor.center.x, 0,
planeAnchor.center.z)
```

- **Τα SCNPlanes είναι κατακόρυφα προσανατολισμένα στον τοπικό συντεταγμένο χώρο τους**
- **Περιστροφή του αντικειμένου για να ταιριάζει με τον οριζόντιο προσανατολισμό του ARPlaneAnchor**

```
  planeNode.transform = SCNMatrix4MakeRotation(-Float.pi / 2, 1,
0, 0)
```

- **Στο ARKit ανήκει ο κόμβος που αντιστοιχεί στην άγκυρα, οπότε το επίπεδο πρέπει να γίνει παιδί του κόμβου**

```
    node.addChildNode(planeNode)
}
```

Κώδικας 2: "Παράδειγμα προσθήκης οπτικού περιεχομένου σε άγκυρα (anchor)"

SCNMaterial

Τα αντικείμενα της κλάσης SCNMaterial είναι ένα σύνολο χαρακτηριστικών που ορίζουν την εμφάνιση μιας γεωμετρικής επιφάνειας όταν παρέχεται.

Όταν δημιουργείται ένα υλικό (material), ορίζετε μια συλλογή οπτικών χαρακτηριστικών και των επιλογών τους, τα οποία είναι επαναχρησιμοποιούμενα για πολλαπλές γεωμετρίες σε μια σκηνή.

Ένα υλικό έχει πολλές οπτικές ιδιότητες, κάθε μία από τις οποίες ορίζει διαφορετικό μέρος της διαδικασίας φωτισμού και σκίασης του SceneKit. Κάθε οπτική ιδιότητα είναι μια εμφάνιση της κλάσης SCNMaterialProperty που παρέχει ένα συμπαγές χρώμα, υφή ή άλλο περιεχόμενο 2D για εκείνη την πτυχή του rendering του SceneKit. Η ιδιότητα φωτισμού Model του υλικού καθορίζει τη φόρμουλα που χρησιμοποιεί το SceneKit για να συνδυάσει τις οπτικές ιδιότητες με τα φώτα στη σκηνή, για να παράγει το τελικό χρώμα για κάθε pixel στην απόδοση της σκηνής.

Είναι εφικτό να επισυνάπτονται ένα ή περισσότερα υλικά σε μια αναφορά της κλάσης SCNGeometry χρησιμοποιώντας την πρώτη ιδιότητα υλικού ή υλικών. Πολλαπλές γεωμετρίες μπορούν να αναφέρουν το ίδιο υλικό. Σε αυτήν την περίπτωση, η αλλαγή των χαρακτηριστικών του υλικού αλλάζει την εμφάνιση κάθε γεωμετρίας που χρησιμοποιεί.

2d SpriteKit

Το πλαίσιο SpriteKit χρησιμοποιείται για την προσθήκη υψηλής απόδοσης περιεχομένου 2D με ομαλές κινούμενες εικόνες, ή για τον προγραμματισμό 2D παιχνιδιών.

Το SpriteKit είναι ένα πλαίσιο γενικού σκοπού για την σχεδίαση σχημάτων, σωματιδίων, κειμένου, εικόνων και βίντεο σε δύο διαστάσεις. Χρησιμοποιεί το πλαίσιο Metal για να επιτύχει απόδοση υψηλής απόδοσης, προσφέροντας παράλληλα μια απλή διεπαφή προγραμματισμού για να διευκολύνει τη δημιουργία παιχνιδιών και άλλων εφαρμογών με υψηλά γραφικά. Χρησιμοποιώντας ένα πλούσιο σύνολο animation και συμπεριφορών φυσικής, γίνεται γρήγορα η προσθήκη “ζωής” στα οπτικά στοιχεία και η μετάβαση μεταξύ των οθονών.

Το SpriteKit υποστηρίζεται από iOS, macOS, tvOS και watchOS και ενσωματώνεται καλά σε πλαίσια όπως το GameplayKit και το SceneKit.

SKView

Ένα αντικείμενο SKView παρουσιάζεται, καλώντας τη μέθοδο **presentScene(_:)** της προβολής. Όταν εμφανίζεται μια σκηνή από την προβολή, εναλλάσσεται μεταξύ της εκτέλεσης της προσομοίωσής της (η οποία κινεί το περιεχόμενο) και την δημιουργία του περιεχομένου για προβολή.

SKScene

Ένα αντικείμενο SKScene αντιπροσωπεύει μια σκηνή περιεχομένου στο SpriteKit. Μια σκηνή είναι ο ριζικός κόμβος σε ένα δέντρο κόμβων SpriteKit (SKNode). Αυτοί

οι κόμβοι παρέχουν περιεχόμενο που γίνεται κινούμενο και προετοιμάζεται για εμφάνιση από τη σκηνή. Για την εμφάνιση μίας σκηνής, χρησιμοποιείται ένα αντικείμενο της κλάσης SKView, SKRenderer ή WKInterfaceSKScene. Η κλάση SKScene είναι μια υποκλάση της SKEffectNode και επιτρέπει την εφαρμογή ορισμένων εφέ σε ολόκληρη τη σκηνή.

ARSKView

Ένα αντικείμενο της κλάσης ARSKView χρησιμοποιείται για την προβολή εμπειριών AR που επαυξάνουν την προβολή της κάμερας με περιεχόμενο 2D SpriteKit.

Η κλάση ARSKView χρησιμοποιείται για τη δημιουργία εμπειριών επαυξημένης πραγματικότητας που τοποθετούν στοιχεία 2D σε τρισδιάστατο χώρο μέσα σε μια προβολή κάμερας. Όταν εκτελείτε το αντικείμενο ARSession της προβολής η προβολή πραγματοποιεί αυτόματα τη ζωντανή ροή βίντεο από την κάμερα της συσκευής ως φόντο σκηνής.

Όταν εφαρμόζονται οι μέθοδοι του πρωτοκόλλου ARSKViewDelegate για τη συσχέτιση του περιεχομένου του SpriteKit με τις θέσεις του πραγματικού κόσμου, η προβολή αυτόματα κλιμακώνει και περιστρέφει αυτούς τους κόμβους SpriteKit έτσι ώστε να φαίνονται ότι παρακολουθούν τον πραγματικό κόσμο που βλέπει η κάμερα.

Παροχή 2D εικονικού περιεχομένου με το SpriteKit

Το SpriteKit χρησιμοποιείται για 2D οπτικό περιεχόμενο, αλλά η επαυξημένη πραγματικότητα περιλαμβάνει πραγματικούς 3D χώρους. Η κλάση ARSKView χρησιμοποιείται για τη δημιουργία εμπειρίας AR με την παροχή 2D sprites (αντικείμενα SKNode) που αντιστοιχούν σε πραγματικές 3D θέσεις (αντικείμενα ARAnchor). Όταν ο χρήστης μετακινήσει τη συσκευή, η προβολή περιστρέφεται αυτόματα και κλιμακώνεται στους κόμβους SpriteKit που αντιστοιχούν στα αντικείμενα της κλάσης ARAnchor, έτσι ώστε να φαίνονται να παρακολουθούν τον πραγματικό κόσμο που βλέπει η κάμερα.

Για παράδειγμα, γίνεται να τοποθετηθούν εικόνες 2D που φαίνεται να επιπλέουν σε χώρο 3D:

- **Δημιουργία ενός μετασχηματισμού με απόσταση 0,2 μ. μπροστά από την κάμερα.**

```
var translation = matrix_identity_float4x4
translation.columns.3.z = -0.2
let transform =
simd_mul(view.session.currentFrame.camera.transform, translation)
```

- **Προσθήκη ενός ARAnchor στη συνεδρία.**

```
let anchor = ARAnchor(transform: transform)
view.session.add(anchor: anchor)
```

- **Δημιουργία ενός κόμβου που αντιπροσωπεύει ένα anchor**

```
func view(_ view: ARSKView, nodeFor anchor: ARAnchor) -> SKNode? {
    return SKLabelNode(text: "Hello world")
}
```

Κώδικας 3: "Προσθήκη δισδιάστατων εικόνων σε τρισδιάστατο χώρο"

Η μέθοδος **view(_:nodeFor:)** παραπάνω επιστρέφει ένα αντικείμενο SKLabelNode, το οποίο εμφανίζει μια ετικέτα κειμένου. Όπως και οι περισσότεροι κόμβοι SpriteKit, αυτή η κλάση δημιουργεί μια 2D οπτική αναπαράσταση, οπότε η κλάση ARSKView παρουσιάζει τον κόμβο σε στυλ πινακίδας: Το αντικείμενο ζυγίζει και περιστρέφεται (γύρω από τον άξονα z του) έτσι ώστε να φαίνεται ότι ακολουθεί την τρισδιάστατη θέση της άγκυράς του, αλλά πάντα στραμμένη προς την κάμερα.

ΕΠΙΛΟΓΟΣ

Για την υποστήριξη επαυξημένης πραγματικότητας η apple παρέχει ένα σύνολο από βιβλιοθήκες στους προγραμματιστές για να διαλέξουν ανάλογα με τις απαιτήσεις τους και τους προσφέρει τη δυνατότητα να συνδυάζουν τις βιβλιοθήκες αυτές και με δικές τους. Η διαδικασία δημιουργίας επαυξημένων δια δραστικών σκηνών είναι αρκετά απλή δε χρειάζεται χρήση πολύπλοκων μαθηματικών από τον προγραμματιστή για την συσχέτιση πραγματικού βίντεο με επαυξημένης προβολής. Για εφαρμογές κινητών με λειτουργικό σύστημα iOS ο βέλτιστος τρόπος δια δραστικών εφαρμογών είναι με το ArKit.

ΚΕΦΑΛΑΙΟ 3

Εργαλεία βελτιστοποίησης εμπειρίας

ΕΙΣΑΓΩΓΗ

Για την δημιουργία της τελικής εφαρμογής χρησιμοποιήθηκαν εργαλεία, κάποια από τα οποία ήταν απαραίτητα για την ολοκλήρωση της εφαρμογής και κάποια από τα οποία προσθέτουν κάποια χαρακτηριστικά για καλύτερη εμπειρία χρήστη. Τα εργαλεία αυτά είναι το Firebase, τα Cocoapods, η χρήση εντοπισμού γλώσσας χρήστη, ο AVAudioPlayer, τα Core Data, τα User Defaults, το MessageUI και το Adobe Photoshop.

Firebase

Η Firebase είναι μια πλατφόρμα ανάπτυξης εφαρμογών για κινητά και web που αναπτύχθηκε από την Firebase, Inc. το 2011, και στη συνέχεια εξαγοράστηκε από την Google το 2014. Από τον Οκτώβριο του 2018, η πλατφόρμα Firebase διαθέτει 18 προϊόντα που χρησιμοποιούνται από περισσότερες από 1,5 εκατομμύρια εφαρμογές. [22]

Παρέχει στους προγραμματιστές ένα API που επιτρέπει την ενσωμάτωση της online δυνατότητας συνομιλίας στις ιστοσελίδες τους. Μετά την έκδοση της υπηρεσίας συνομιλίας, οι Tamplin και Lee διαπίστωσαν ότι χρησιμοποιείται για να διαβιβάζει δεδομένα εφαρμογών που δεν ήταν μηνύματα συνομιλίας. Οι προγραμματιστές χρησιμοποίησαν το Envolve για να συγχρονίσουν δεδομένα εφαρμογών, όπως η κατάσταση του παιχνιδιού, σε πραγματικό χρόνο στους χρήστες τους. Οι Tamplin και Lee αποφάσισαν να χωρίσουν το σύστημα συνομιλίας και την αρχιτεκτονική πραγματικού χρόνου που την τροφοδοτούσαν. Ίδρυσαν τη Firebase ως ξεχωριστή εταιρεία τον Απρίλιο του 2012.

Firebase Υπηρεσίες

- **Analytics**
  Το Firebase Analytics είναι μια δωρεάν λύση μέτρησης εφαρμογών που παρέχει πληροφορίες για τη χρήση της εφαρμογής και την αφοσίωση του χρήστη.

- **Firebase Cloud Messaging**

Παλαιότερα γνωστό ως Google Cloud Messaging (GCM), το Firebase Cloud Messaging (FCM) είναι μια λύση μεταξύ των πλατφόρμων για μηνύματα και ειδοποιήσεις για εφαρμογές Android, iOS και web, οι οποίες από το 2016 μπορούν να χρησιμοποιηθούν χωρίς κόστος

- **Firebase Auth**
  Το Firebase Auth είναι μια υπηρεσία που μπορεί να πιστοποιήσει τους χρήστες χρησιμοποιώντας μόνο τον κωδικό πελάτη. Υποστηρίζει παροχή κοινωνικής σύνδεσης Facebook, GitHub, Twitter και Google (και Παιχνίδια Google Play). Επιπλέον, περιλαμβάνει ένα σύστημα διαχείρισης χρηστών, στο οποίο οι προγραμματιστές μπορούν να ενεργοποιήσουν τον έλεγχο ταυτότητας χρήστη με σύνδεση μέσω ηλεκτρονικού ταχυδρομείου και κωδικού πρόσβασης που είναι αποθηκευμένη με το Firebase.

- **Realtime Database**
  Το Firebase παρέχει μια βάση δεδομένων σε πραγματικό χρόνο ως υπηρεσία. Η υπηρεσία παρέχει στους προγραμματιστές εφαρμογών ένα API που επιτρέπει στα δεδομένα εφαρμογών να συγχρονίζονται μεταξύ των πελατών και να αποθηκεύονται στο cloud του Firebase. Η εταιρεία παρέχει βιβλιοθήκες πελατών που επιτρέπουν την ενσωμάτωση με εφαρμογές Android, iOS, JavaScript, Java, Objective-C, Swift και Node.js. Η βάση δεδομένων είναι επίσης προσβάσιμη μέσω ενός API REST και συνδέσεων για διάφορα πλαίσια JavaScript, όπως AngularJS, React, Ember.js και Backbone.js. Το API REST χρησιμοποιεί το πρωτόκολλο Server-Sent Events, το οποίο είναι ένα API για τη δημιουργία συνδέσεων HTTP και τη λήψη push notifications από ένα διακομιστή. Οι προγραμματιστές που χρησιμοποιούν τη βάση δεδομένων σε πραγματικό χρόνο μπορούν να εξασφαλίσουν τα δεδομένα τους, χρησιμοποιώντας τους κανόνες ασφαλείας που εφαρμόζει η εταιρεία από πλευράς διακομιστή.

- **Firebase Storage**
  Το Firebase Storage παρέχει ασφαλή updload και download αρχείων για εφαρμογές Firebase, ανεξάρτητα από την ποιότητα του δικτύου. Ο προγραμματιστής μπορεί να το χρησιμοποιήσει για την αποθήκευση εικόνων, ήχου, βίντεο ή άλλου περιεχομένου που δημιουργεί ο χρήστης. Η αποθήκευση Firebase υποστηρίζεται από το Google Cloud Storage.

- **Firebase Hosting**
  Το Firebase Hosting είναι μια στατική και δυναμική υπηρεσία φιλοξενίας ιστοσελίδων που ξεκίνησε στις 13 Μαΐου 2014. Υποστηρίζει φιλοξενία στατικών αρχείων όπως CSS, HTML, JavaScript και άλλα αρχεία, καθώς και υποστήριξη μέσω των λειτουργιών Cloud. Η υπηρεσία παραδίδει αρχεία μέσω δικτύου παροχής περιεχομένου (CDN) μέσω της κρυπτογράφησης HTTP Secure (HTTPS) και της κρυπτογράφησης Secure Sockets Layer (SSL). Η εταιρεία Firebase συνεργάζεται με το Fast, ένα CDN, για να παρέχει το υποστηρικτικό CDN Firebase Hosting. Η εταιρεία δηλώνει ότι η Firebase Hosting εξελίχθηκε

από αιτήματα πελατών. Οι προγραμματιστές χρησιμοποιούσαν τη βάση δεδομένων Firebase για τη βάση δεδομένων σε πραγματικό χρόνο, αλλά χρειάζονταν ένα μέρος για να φιλοξενήσουν το περιεχόμενό τους.

- **ML Kit**
  Το ML Kit είναι ένα mobile σύστημα machine learning για προγραμματιστές που ξεκίνησε στις 8 Μαΐου 2018 σε beta κατά τη διάρκεια του Google I / O 2018. Τα εργαλεία AP API του ML Kit διαθέτουν μια ποικιλία χαρακτηριστικών όπως αναγνώριση κειμένου, ανίχνευση προσώπων, σάρωση γραμμικών κωδικών, ετικετών εικόνων και αναγνώριση ορόσημων. Αυτήν τη στιγμή διατίθεται για προγραμματιστές iOS ή Android. Επίσης, μπορούν να εισαχθούν μοντέλα TensorFlow Lite, αν τα API δεν είναι αρκετά. Τα API μπορούν να χρησιμοποιηθούν σε συσκευή ή σε cloud.

- **Crashlytics**
  Τα Crashlytics δημιουργούν λεπτομερείς αναφορές για τα σφάλματα στην εφαρμογή. Τα σφάλματα ομαδοποιούνται σε ομάδες παρόμοιων ιχνών στοίβας και ταξινομούνται βάσει της σοβαρότητας των επιπτώσεων στους χρήστες εφαρμογών. Εκτός από τις αυτόματες αναφορές, ο προγραμματιστής μπορεί να καταγράφει προσαρμοσμένα συμβάντα για να βοηθήσει να καταγράψει τα βήματα που οδηγούν σε αναγκαστικό τερματισμό της εφαρμογής.

- **Performance**
  Το Firebase Performance παρέχει πληροφορίες σχετικά με την απόδοση μιας εφαρμογής και τις λανθάνουσες περιόδους λειτουργίας των χρηστών της εφαρμογής.

- **Firebase Test Lab για Android και iOS**
  Το Firebase Test Lab για Android και iOS παρέχει υποδομή βασισμένη σε cloud για δοκιμή εφαρμογών Android και iOS. Με μία ενέργεια, οι προγραμματιστές μπορούν να ξεκινήσουν τη δοκιμή των εφαρμογών τους σε μια μεγάλη ποικιλία συσκευών. Τα αποτελέσματα των δοκιμών - συμπεριλαμβανομένων των αρχείων καταγραφής, των βίντεο και των στιγμιότυπων οθόνης - διατίθενται στο έργο στην κονσόλα Firebase. Ακόμα κι αν ένας προγραμματιστής δεν έχει γράψει κανένα κώδικα δοκιμής για την εφαρμογή του, το Lab Test μπορεί να τρέξει αυτόματα την εφαρμογή, αναζητώντας crashes. Το Test Lab για iOS βρίσκεται αυτή τη στιγμή σε beta.

- **Firebase Ειδοποιήσεις**
  Η ειδοποίηση Firebase είναι μια υπηρεσία που επιτρέπει ειδοποιήσεις στοχευμένων χρηστών για προγραμματιστές εφαρμογών για κινητά χωρίς κόστος.

Cocoapods

Το CocoaPods είναι ένας διαχειριστής εξάρτησης επιπέδου εφαρμογών για τις γλώσσες Objective-C, Swift και οποιεσδήποτε άλλες γλώσσες που εκτελούνται στο χρόνο εκτέλεσης της Objective-C, όπως το RubyMotion, το οποίο παρέχει μια τυπική μορφή διαχείρισης εξωτερικών βιβλιοθηκών. Έχει πάνω από 56.000 βιβλιοθήκες και χρησιμοποιείται σε πάνω από 3 εκατομμύρια εφαρμογές [16]. Αναπτύχθηκε από τους Eloy Durán και Fabio Pelosin, οι οποίοι συνεχίζουν να διαχειρίζονται το έργο με τη βοήθεια και τη συμβολή πολλών άλλων. Ξεκίνησαν την ανάπτυξη τον Αύγουστο του 2011 και πραγματοποίησαν την πρώτη δημόσια κυκλοφορία την 1η Σεπτεμβρίου 2011.

Το CocoaPods επικεντρώνεται στην κατανομή των βιβλιοθηκών τρίτων με βάση την πηγή και στην αυτόματη ενσωμάτωση σε έργα Xcode. Το CocoaPods είναι χτισμένο με Ruby και εγκαθίσταται με τη προεπιλεγμένη Ruby διαθέσιμη στο macOS. Συνιστάται να χρησιμοποιείται η προεπιλεγμένη ruby.

Το CocoaPods τρέχει από τη γραμμή εντολών και είναι επίσης ενσωματωμένο στο ολοκληρωμένο περιβάλλον ανάπτυξης του AppCode του JetBrains. Εγκαθιστά εξαρτήσεις για μια εφαρμογή με εξειδίκευση εξαρτήσεων αντί για χειροκίνητη αντιγραφή αρχείων προέλευσης. Εκτός από την εγκατάσταση πολλών διαφορετικών βιβλιοθηκών, παρέχονται επίσης τα master repositories για πολλές βιβλιοθήκες Open Source και διατηρούνται αποθηκευμένα στο git και φιλοξενούνται στο GitHub. Το σύστημα ανάλυσης εξάρτησης CocoaPods τροφοδοτείται από το Molinillo, το οποίο χρησιμοποιείται επίσης από άλλα μεγάλα έργα όπως τα Bundler, RubyGems και Berkshelf.

Internationalization and Localization

Ο εντοπισμός της περιοχής είναι η διαδικασία μετάφρασης της εφαρμογής σε πολλές γλώσσες. Αλλά προτού μπορέσει να εντοπιστεί η περιοχή της εφαρμογής πρέπει πρώτα να υποστηρίζει διεθνοποίηση. Η διεθνοποίηση είναι η διαδικασία της προσαρμογής της εφαρμογής σε διαφορετικές γλώσσες, περιοχές και πολιτισμούς. Επειδή μια ενιαία γλώσσα μπορεί να χρησιμοποιηθεί σε πολλά μέρη του κόσμου, η εφαρμογή πρέπει να προσαρμοστεί στις περιφερειακές και πολιτιστικές συμβάσεις του τόπου κατοικίας ενός ατόμου. Μια διεθνοποιημένη

εφαρμογή εμφανίζεται σαν να είναι μια εγγενής εφαρμογή σε όλες τις γλώσσες και περιοχές που υποστηρίζει.

Το App Store είναι διαθέσιμο σε περισσότερες από 150 διαφορετικές χώρες και η διεθνοποίηση της εφαρμογής είναι το πρώτο βήμα για την επίτευξη αυτής της παγκόσμιας αγοράς. Στο App Store Connect, καθορίζεται αν η εφαρμογή είναι διαθέσιμη σε όλες τις περιοχές ή συγκεκριμένες περιοχές. Οι χρήστες σε άλλες χώρες μπορούν να χρησιμοποιήσουν την εφαρμογή σε οποιαδήποτε από τις διαθέσιμες της εφαρμογής.

Το Xcode παρέχει ένα εργαλείο για την δημιουργία διαφορετικών λεκτικών για κάθε γλώσσα που θέλει ο προγραμματιστής να υποστηρίζει η εφαρμογή του. Πρώτα διεθνοποιείται η διεπαφή χρήστη και ο κώδικας κατά την ανάπτυξη. Αυτό έχει σαν αποτέλεσμα την δημιουργία αρχείων τόσα όσες και οι γλώσσες που υποστηρίζει η εφαρμογή. Σε κάθε αρχείο κάθε γραμμή υποστηρίζει την μετάφραση ενός μοναδικού κλειδιού, για παράδειγμα **«“key_helloWorld” = “Καλημέρα κόσμε”»** . Κάθε κλειδί που δημιουργείται πρέπει να υπάρχει σε όλα τα αρχεία γλωσσών. Ανάλογα με την γλώσσα που παίρνει η εφαρμογή κατά την εκτέλεση διαβάζεται και η αντίστοιχη μετάφραση για να παρουσιαστεί στον χρήστη. Στον κώδικα αυτό γράφεται σαν **« “Lebel.text = Localized(“key_helloWorld ”)»**. Με αυτόν τον τρόπο η εφαρμογή μεταφράζεται σε διαφορετικές γλώσσες.

AVAudioPlayer

Τα αντικείμενα της κλάσης AVAudioPlayer χρησιμοποιούνται για αναπαραγωγή ήχου εκτός εάν αναπαράγεται ήχος που έχει ληφθεί από ροή δικτύου ή απαιτεί πολύ χαμηλή λανθάνουσα κατάσταση εισόδου / εξόδου.

Χρησιμοποιώντας ένα αντικείμενο κλάσης AVAudioPlayer γίνεται:

- Αναπαραγωγή ήχου οποιασδήποτε διάρκειας
- Αναπαραγωγή ήχων από αρχεία ή προσωρινά αποθηκευμένα αρχεία
- Αναπαραγωγή ήχου σε λούπα
- Αναπαραγωγή πολλαπλών ήχων ταυτόχρονα
- Έλεγχος σχετικά με το επίπεδο αναπαραγωγής, στερεοφωνική τοποθέτηση και ρυθμό αναπαραγωγής για κάθε ήχο που αναπαράγεται
- Μετακίνηση σε ένα συγκεκριμένο σημείο σε ένα αρχείο ήχου, το οποίο υποστηρίζει τέτοιες λειτουργίες εφαρμογής όπως γρήγορη μετακίνηση προς τα εμπρός και προς τα πίσω
- Απόκτηση δεδομένων που μπορούν να χρησιμοποιηθούν για μετρήσεις σε επίπεδο αναπαραγωγής

Η κλάση AVAudioPlayer επιτρέπει την αναπαραγωγή ήχου σε οποιαδήποτε μορφή διαθέσιμη σε iOS και macOS. Επίσης δίνεται και η δυνατότητα της

διαχείρισης των διακοπών (όπως μια εισερχόμενη κλήση σε iOS κινητό τηλέφωνο) και η ενημέρωση του περιβάλλον εργασίας χρήστη όταν τελειώσει ο ήχος.

Αυτή η κλάση χρησιμοποιεί τη λειτουργία δηλωμένων ιδιοτήτων της Objective-C για τη διαχείριση πληροφοριών σχετικά με έναν ήχο, όπως το σημείο αναπαραγωγής στο χρονοδιάγραμμα του ήχου, καθώς και για την πρόσβαση σε επιλογές αναπαραγωγής, όπως η ένταση και η επαναφορά. [17]

Core Data

Τα Core Data είναι ένα πλαίσιο που χρησιμοποιείται για τη διαχείριση των αντικειμένων του στρώματος μοντέλου της εφαρμογής. Παρέχει γενικευμένες και αυτοματοποιημένες λύσεις σε κοινές εργασίες που σχετίζονται με τον κύκλο ζωής αντικειμένων και τη διαχείριση γραφικών αντικειμένων. [18]

Τα Core Data συνήθως μειώνουν κατά 50 έως 70% του ποσού του κώδικα που γράφεται για την υποστήριξη του επιπέδου μοντέλου. Αυτό οφείλεται κυρίως στις ακόλουθες ενσωματωμένες λειτουργίες που δεν χρειάζεται να εφαρμόζονται, να δοκιμάζονται ή να βελτιστοποιούνται:

- Αλλαγή της παρακολούθησης και της ενσωματωμένης διαχείρισης της αναίρεσης.
- Διατήρηση της διάδοσης των αλλαγών, συμπεριλαμβανομένης της διατήρησης της συνέπειας των σχέσεων μεταξύ αντικειμένων.
- Λανθασμένη φόρτωση αντικειμένων, μερική υλοποίηση συμβολαίων μελλοντικής εκπλήρωσης (σφάλμα) και ανταλλαγή δεδομένων αντιγραφής-εγγραφής για μείωση των γενικών εξόδων.
- Αυτόματη επικύρωση των τιμών ιδιοκτησίας. Τα διαχειριζόμενα αντικείμενα επεκτείνουν τις τυπικές μεθόδους επικύρωσης κωδικοποίησης-κλειδιού για να εξασφαλίσουν ότι οι μεμονωμένες τιμές βρίσκονται μέσα σε αποδεκτές περιοχές, έτσι ώστε οι συνδυασμοί τιμών να έχουν νόημα.
- Εργαλεία μετεγκατάστασης σχήματος που απλοποιούν τις αλλαγές σχήματος και επιτρέπουν την εκτέλεση αποτελεσματικής μετανάστευσης σχήματος επιτόπου.
- Προαιρετική ενσωμάτωση με το επίπεδο ελεγκτή της εφαρμογής για υποστήριξη του συγχρονισμού διεπαφής χρήστη.
- Ομαδοποίηση, φιλτράρισμα και οργάνωση δεδομένων στη μνήμη και στο περιβάλλον χρήστη.
- Αυτόματη υποστήριξη για την αποθήκευση αντικειμένων σε αποθήκες εξωτερικών δεδομένων.

- Σύνθετη συλλογή ερωτημάτων. Εναλλακτικός τρόπος γραφής SQL, υποστήριξη δημιουργίας πολύπλοκων ερωτημάτων, συσχετίζοντας ένα αντικείμενο της κλάσης NSPredicate με ένα αίτημα λήψης.
- Παρακολούθηση της έκδοσης και ασφάλεια για την υποστήριξη της αυτόματης επίλυσης συγκρούσεων πολλών εγγράφων.
- Αποτελεσματική ενσωμάτωση με τις αλυσίδες εργαλείων macOS και iOS.

User Defaults

Μια διεπαφή στη βάση δεδομένων προεπιλογών του χρήστη, όπου διατηρούνται σταθερά ζεύγη κλειδιών-τιμών στις εκκινήσεις της εφαρμογής. Η κλάση UserDefaults παρέχει μια προγραμματική διασύνδεση για την αλληλεπίδραση με το σύστημα προεπιλογών. Το σύστημα προεπιλογών επιτρέπει σε μια εφαρμογή να προσαρμόζει τη συμπεριφορά της ώστε να ταιριάζει με τις προτιμήσεις ενός χρήστη. Για παράδειγμα, επιτρέπει στους χρήστες να καθορίζουν τις προτιμώμενες μονάδες μέτρησης ή την ταχύτητα αναπαραγωγής πολυμέσων. Οι εφαρμογές αποθηκεύουν αυτές τις προτιμήσεις, αναθέτοντας τιμές σε ένα σύνολο παραμέτρων στη βάση δεδομένων προεπιλογών ενός χρήστη. Οι παράμετροι αναφέρονται ως προεπιλογές επειδή χρησιμοποιούνται συνήθως για τον προσδιορισμό της προεπιλεγμένης κατάστασης της εφαρμογής κατά την εκκίνηση ή του τρόπου με τον οποίο ενεργεί από προεπιλογή.

Κατά το χρόνο εκτέλεσης, χρησιμοποιούνται αντικείμενα της κλάσης UserDefaults για να διαβάζονται οι προεπιλογές που χρησιμοποιεί η εφαρμογή από τη βάση δεδομένων προεπιλογών του χρήστη. Η κλάση UserDefaults αποθηκεύει τις πληροφορίες για να μην χρειάζεται να ανοίξει τη βάση δεδομένων προεπιλογών του χρήστη κάθε φορά που χρειάζεται να διαβαστεί μια προεπιλεγμένη τιμή. Όταν ορίζεται μια προεπιλεγμένη τιμή, αλλάζει συγχρονισμένα μέσα στη τρέχων διαδικασία και ασύγχρονα σε άλλες διαδικασίες. [19]

MessageUI

Το MessageUI είναι ένα framework που υποστηρίζει τη σύνταξη μηνυμάτων ηλεκτρονικού ταχυδρομείου και κειμένου, ώστε οι χρήστες να μπορούν να επεξεργάζονται και να στέλνουν μηνύματα χωρίς να βγαίνουν από την εφαρμογή.

Το πλαίσιο MessageUI παρέχει εξειδικευμένους ελεγκτές προβολής για την παρουσίαση τυποποιημένων διεπαφών σύνθεσης για μηνύματα κειμένου μηνυμάτων ηλεκτρονικού ταχυδρομείου και μηνυμάτων SMS (Short Message Service). Αυτές οι διεπαφές χρησιμοποιούνται για την προσθήκη δυνατοτήτων χωρίς την απαίτηση της εγκατάλειψης της εφαρμογής από τον χρήστη.

Για την εμφάνιση μίας διεπαφής σύνθεσης, παρουσιάζεται ο αντίστοιχος ελεγκτής προβολής από την εφαρμογή. Μόλις παρουσιαστεί, ο χρήστης έχει τη δυνατότητα να προσαρμόσει τα περιεχόμενα πριν από την αποστολή ή ακύρωση του μηνύματος. Το προσαρμοσμένο αντικείμενο στη συνέχεια χειρίζεται την αφαίρεση του ελεγκτή προβολής από την διεπαφή χρήστη, με βάση τη δράση του χρήστη. Η αποστολή του μηνύματος γίνεται μέσω του ηλεκτρονικού ταχυδρομείου του χρήστη που έχει ήδη προσθέσει στη συσκευή του. [20]

Adobe Photoshop

Το Adobe Photoshop είναι ένας επεξεργαστής γραφικών raster που αναπτύχθηκε και δημοσιεύθηκε από την Adobe Inc. για macOS και Windows.

Το πρόγραμμα δημιουργήθηκε το 1988 από τους Thomas και John Knoll. Από τότε, έχει γίνει το πιο γνωστό και πιο πολυχρησιμοποιημένο βιομηχανικό πρότυπο στην επεξεργασία raster γραφικών. Μπορεί να επεξεργαστεί και να συνθέσει εικόνες raster σε πολλαπλά στρώματα και να υποστηρίξει μάσκες, σύνθετα alpha και διάφορα έγχρωμα μοντέλα, όπως RGB, CMYK, CIELAB, spot color και duotone. Το Photoshop χρησιμοποιεί τις δικές του μορφές αρχείων PSD και PSB για να υποστηρίξει αυτές τις λειτουργίες.

Εκτός από τα ράστερ γραφικά, έχει περιορισμένες δυνατότητες επεξεργασίας ή εκτύπωσης κειμένου, διανυσματικών γραφικών (ειδικά μέσω διαδρομής αποκοπής), 3D γραφικών και βίντεο. Το σύνολο χαρακτηριστικών του προγράμματος μπορεί να επεκταθεί με plug-ins, προγράμματα που αναπτύσσονται και διανέμονται ανεξάρτητα από το ίδιο και μπορούν να λειτουργήσουν μέσα σε αυτό και να προσφέρουν νέες ή βελτιωμένες λειτουργίες.

Το σχήμα ονομασίας του Photoshop βασίστηκε αρχικά σε αριθμούς έκδοσης. Ωστόσο, τον Οκτώβριο του 2002 κάθε νέα έκδοση του Photoshop χαρακτηρίστηκε με "CS" συν ένα αριθμό. Τον Ιούνιο του 2013, με την εισαγωγή του Creative Cloud, το πρόγραμμα αδειοδότησης του Photoshop άλλαξε σε αυτό του λογισμικού ως μοντέλο μίσθωσης υπηρεσίας και τα επιθέματα "CS" αντικαταστάθηκαν με το "CC". Ιστορικά, το Photoshop συνδυάστηκε με πρόσθετο λογισμικό όπως το Adobe ImageReady, το Adobe Fireworks, το Adobe Bridge, το Adobe Device Central και το Adobe Camera RAW.

ΕΠΙΛΟΓΟΣ

Η τελική εφαρμογή που δημιουργήθηκε έπρεπε να είναι όσο το δυνατόν πιο επίκαιρη στη σύγχρονη εποχή και για την επίτευξη αυτού δεν αρκούσε μόνο η χρήση των framework της apple έπρεπε να ενσωματωθούν κι άλλα εργαλεία. Όλα ήταν εξίσου σημαντικά.

ΚΕΦΑΛΑΙΟ 4

Εφαρμογή Ολυμπιακού μουσείο

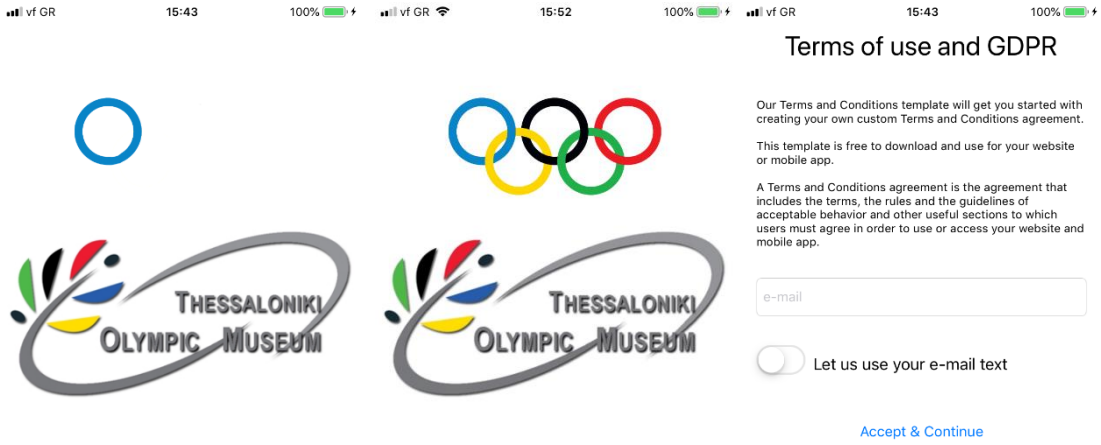
ΕΙΣΑΓΩΓΗ

Η εφαρμογή για το Ολυμπιακό μουσείο είναι μία κανονική εφαρμογή μουσείου με κομμάτια Augmented Reality. Η εφαρμογή αποτελείται κυρίως από τρία μέρη Augmented reality, την επαύξηση του χάρτη, την υπηρεσία παρουσίασης των αθλημάτων καθώς και την προσθήκη του κότινου στον ολυμπιονίκη. Επίσης, έχει ένα παιχνίδι ερωτήσεων και μία αρχική οθόνη με πληροφορίες για το μουσείο. Η εφαρμογή είναι γραμμένη με το MVC (Model View Controller) Pattern που προτείνει η apple [14]. Ο λόγος που επιλέχτηκε αυτό το pattern είναι γιατί είναι μία αρκετά καλά χωρισμένη σε κομμάτια εφαρμογή και δεν έχει σημεία που να γίνεται πολύ περίπλοκη για τη χρήση πιο σύνθετων pattern όπως το MVP [23], MVVM [24] ή VIPER [25] που είναι τα πιο συνηθισμένα design patterns που χρησιμοποιούνται σε πιο σύνθετες αναπτύξεις λογισμικού.

Εικόνα 6: “Εικόνα επιφάνειας κινητού με εγκατεστημένη την εφαρμογή ”

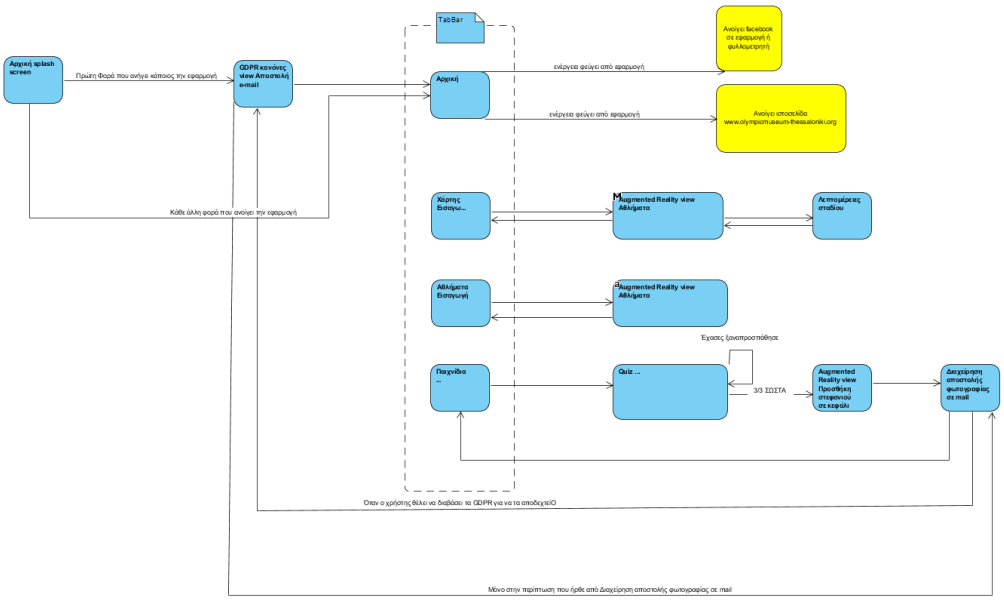
Δομή Εφαρμογής

Στην εφαρμογή αρχικά παρουσιάζεται μία αρχική διεπαφή στην οποία φορτώνει το logo της εφαρμογής και στο παρασκήνιο φορτώνουν τα δεδομένα τα οποία θα χρησιμοποιηθούν. Αν ο χρήστης μπαίνει στην εφαρμογή για πρώτη φορά τότε πηγαίνει στην διεπαφή των GDPR και της παραχώρησης mail, στη συνέχεια ανεξάρτητα με το αν έδωσε mail στην εφαρμογή ο χρήστης πηγαίνει στο κύριο κομμάτι της εφαρμογής. Για την επιλογή του logo (ελληνικό ή αγγλικό) και της γλώσσας των λεκτικών χρησιμοποιείται το εργαλείο Internationalization και localization της εφαρμογής που αντιστοιχεί τις εικόνες και τα λεκτικά ανάλογα με την επιλεγμένη γλώσσα του κινητού του χρήστη. Η εφαρμογή υποστηρίζει ελληνικά και αγγλικά και για κάθε άλλη περίπτωση ορίζεται σαν γλώσσα της εφαρμογής η αγγλική.



Εικόνα 7: "Αρχική εικόνα εφαρμογής και διεπιφάνεια GDPR όρων"

Στην εφαρμογή χρησιμοποιείται ένας UITabBarController για να παρουσιάζονται οι διαφορετικές διεπαφές της εφαρμογής. Η εφαρμογή αποτελείται από τέσσερεις βασικούς διαφορετικούς ελεγκτές προβολής οι οποίοι και φορτώνονται στο Tab bar. Η πρώτη είναι η αρχική οθόνη που παρουσιάζει τα στοιχεία της εφαρμογής, η δεύτερη είναι ο χάρτης με την επαυξημένη πραγματικότητα, η τρίτη είναι η επαύξηση των εικόνων των αθλητών και η τέταρτη είναι το παιχνίδι quiz που αν ο χρήστης απαντήσει σωστά σε 3 ερωτήσεις πηγαίνει στην διεπαφή που του εμφανίζει στο κεφάλι ένα κλαδί ελιάς και στέφεται ολυμπιονίκης.



Σχήμα 8: "Διάγραμμα ροής της εφαρμογής "

Η εφαρμογή έχει τρεις βοηθητικές Singleton κλάσεις των οποίων οι αναφορές μπορούν να κληθούν από οποιοδήποτε σημείο μέσα στην εφαρμογή και να έχουν το ίδιο αντικείμενο. Οι κλάσεις αυτές είναι η AnalyticsManager, DataManager, StyleManager.

AnalyticsManager

Η κλάση AnalyticsManager είναι αρμόδια για οποιαδήποτε προσθήκη αναλυτικών στο Firebase. Από οποιαδήποτε άλλη κλάση καλείτε η μέθοδος **addFirstOneAnalyticTest(name: String)**.

```
func addFirstOneAnalyticTest(name: String) {
        Analytics.logEvent(AnalyticsEventSelectContent,
parameters: [
            AnalyticsParameterItemID: "id-
\(AlalyticsIdentifiers.FirstOne.rawValue) \(name)"
            ])
    }
```

Κώδικας 4: "Προσθήκη δεδομένων στην κονσόλα του firebase"

AnalyticsIdentifiers είναι ένας enumerator με δύο τιμές και χρησιμοποιείται για να κατηγοριοποιεί τα δεδομένα που στέλνονται στην κονσόλα. Για παράδειγμα, όταν ο χρήστης πατάει σε ένα στάδιο στο χάρτη στέλνεται από το AnalyticsIdentifiers ο identifier του χάρτη και στη συνέχεια το όνομα του σταδίου.

Στην ουσία ο AnalyticsManager είναι ένας ενδιάμεσος της κλάσης Analytics της βιβλιοθήκης Firebase και του υπόλοιπου project.

DataManager

Ο DataManager αναλαμβάνει να διαχειρίζεται όλα τα δεδομένα που έρχονται από τις διαδικτυακές κλήσεις της εφαρμογής καθώς και να τα αποθηκεύει στην μνήμη της εφαρμογής. Λόγω του ότι η εφαρμογή στο back – end δεν έχει κάποιο μηχανισμό για versioning η εφαρμογή κατεβάζει τα δεδομένα που χρειάζεται από τη βάση σε κάθε άνοιγμα της εφαρμογής. Τα δεδομένα που κυρίως χρειάζεται να αποθηκευτούν είναι οι ερωτήσεις με τις απαντήσεις τους. Όλες οι ερωτήσεις και απαντήσεις αποθηκεύονται στα Core Data της εφαρμογής. Μετά ο χρήστης μπορεί να παίξει όσες φορές θέλει το παιχνίδι χωρίς να χρειάζεται να κατεβάζει ξανά τις ερωτήσεις ούτε καν να έχει σύνδεση στο δίκτυο. Όταν ο χρήστης ανοίξει την εφαρμογή χωρίς σύνδεση στο διαδίκτυο και έχει κατεβάσει τα δεδομένα που χρειάζεται η εφαρμογή θα προσπαθήσει να κατεβάσει τα δεδομένα ξανά αλλά εφόσον δεν έχει σύνδεση στο διαδίκτυο θα χρησιμοποιήσει κανονικά την εφαρμογή με τα παλιά δεδομένα. Επίσης από την μεριά του μουσείου μπορούν να αλλάξουν τις ερωτήσεις, τις απαντήσεις καθώς και να προσθέσουν – αφαιρέσουν ερωτήσεις χωρίς να χρειάζεται η αναβάθμιση της εφαρμογής.

Επίσης αρμοδιότητα του DataManager είναι να προσθέτει στη βάση του firebase το e-mail του χρήστη αν ο χρήστης το αποδεχθεί καθώς και να ελέγχει για το αν ο χρήστης έχει αποδεχθεί ή απορρίψει την προσθήκη του mail του στη βάση της εφαρμογής. Ο τρόπος με τον οποίο επιτυγχάνεται αυτό είναι με τη χρήση των User Defaults. Και των μεθόδων:

```
  func isFirstTimeOpenApp() -> Bool {
        return (UserDefaults.standard.string(forKey:
UserDefaultsIdentifier.firstTimeLogin.rawValue) == nil)
    }
    func setFirstTimeOpenAppToFalse() {
        UserDefaults.standard.set("false", forKey:
UserDefaultsIdentifier.firstTimeLogin.rawValue)
    }
    func didDeviceGaveMail() -> Bool {
        return (UserDefaults.standard.string(forKey:
UserDefaultsIdentifier.deviceGaveMail.rawValue) == nil)
    }
    func setDeviceGaveMailToFalse(mail: String) {
        UserDefaults.standard.set(mail, forKey:
UserDefaultsIdentifier.deviceGaveMail.rawValue)
    }
    func getDeviceGaveMail() -> String {
```

```
        return UserDefaults.standard.string(forKey:
UserDefaultsIdentifier.deviceGaveMail.rawValue) ?? "" //optional
unwrapping με προκαθορισμένη τιμή
  }
```

Κώδικας 5: "Παραδείγματα μεθόδων που χρησιμοποιούν τα User Defaults"

Ο UseDefaultsIdentifier είναι ένας enumerator που έχει σαν τιμή για κάθε περίπτωση το μοναδικό κλειδί για την αποθήκευση του ονόματος του κάθε UserDefault αντικειμένου που χρειάζεται να αποθηκευτεί ή να διαβαστεί.

```
enum UserDefaultsIdentifier: String {
    case firstTimeLogin = "com.OMAR.FirstTimeLogin"
    case deviceGaveMail = "com.OMAR.DeviceGaveMail"
}
```

Κώδικας 6: "Παράδειγμα enumerator UserDefaultsIdentifier"

Κάθε μία από τις παραπάνω μεθόδους έχει σαν σκοπό να θέσει ή να διαβάσει την τιμή ενός UserDefault και κάθε μία είναι προσπελάσιμη μέσω του DataManager από κάθε κλάση του project.

Για την ανάγνωση των ερωτήσεων από κάθε κλάση χρησιμοποιείται η μέθοδος

```
func getQuestionsForTest(_ numberOfQuestion: Int? = 3, completion:
@escaping (_ managedQuestions: [ManagedQuestion], _ error: Error?)
-> ()){
        var arrayOfQuestions:[ManagedQuestion] = []
        let mainContext =
persistentContainer?.managedObjectContext
        mainContext?.perform {
            arrayOfQuestions = mainContext?.fetchObjects(of:
ManagedQuestion.self, withEntityName: "ManagedQuestion") ?? []
             completion(arrayOfQuestions.shuffled(), nil) //Κλήση
του completion handler
        }
    }
```

Κώδικας 7: "Ανάγνωση ερωτήσεων από την εσωτερική βάση "

η οποία έχει έναν completion handler ο οποίος την κάνει να λειτουργεί ασύγχρονα. Από οποιαδήποτε κλάση καλείται η μέθοδος αυτή με τον τρόπο

```
DataManager.shared.getQuestionsForTest(maxNumberOfQuestions,
completion: { managedQuestions, error in
…
})
```

Κώδικας 8: "Παράδειγμα κλήσης της μεθόδου getQuestionsForTest"

Όταν στην κλάση που κάλεσε τη μέθοδο κληθεί ο completion handler περνιούνται παραμετρικά τόσες τυχαίες ερωτήσεις όσες η τιμή της μεταβλητής **maxNumberOfQuestions**, από όλες τις ερωτήσεις που πάρθηκαν από τον DataManager και ήταν αποθηκευμένες στα Core Data και σε τυχαία σειρά.

Για την αποθήκευση των ερωτήσεων και απαντήσεων στη βάση χρησιμοποιείται η μέθοδος

```
func fetchQuestionsFromInternet(landuagePassed: String,
completion: @escaping (_ error: Error?) -> ()) {
        var arrayOfFetchedQuestions:[Question] = []
        let ref: DatabaseReference =
Database.database().reference()

ref.child("questions").child(landuagePassed).observe(.value) {
(snapshot) in
            if let value = snapshot.value as? NSDictionary {
                for v in value {
                    let questionFetched = Question(swiftyJson:
v.value)

arrayOfFetchedQuestions.append(questionFetched)
                }
                self.saveQuestionsToDataBase(questions:
arrayOfFetchedQuestions)
                completion(nil)
            }else{
                completion(nil)
            }
        }
    }
```

Κώδικας 9: "Λήψη ερωτήσεων από τη βάση questions του firebase"

Η μέθοδος **fetchQuestionsFromInternet** έχει σαν σκοπό την επικοινωνία με τη βάση η οποία βρίσκεται στο firebase και την αποθήκευση των ερωτήσεων στα core data της εφαρμογής. Αρχικά δημιουργείται μία αναφορά στη NoSQL βάση δεδομένων του firebase και στη συνέχεια από την ριζική αναφορά στο παιδί με μοναδικό κλειδί το αναγνωριστικό "questions" το οποίο χρειάζεται για να πάρει τις ερωτήσεις. Η απάντηση της βάσης είναι ένα json στιγμιότυπο το οποίο αντιστοιχεί σε ένα σύνολο ερωτήσεων και απαντήσεων με βάση την επιλεγμένη γλώσσα. Τέλος καλείται η μέθοδος saveQuestionsToDataBase η οποία αποθηκεύει το σύνολο των αντικειμένων της κλάσης Question στα Core Data της εφαρμογής.

```
private func saveQuestionsToDataBase(questions: [Question]) {
        persistentContainer?.performBackgroundTask({ context in
            context.deleteObjects(withEntityName:
"ManagedQuestion")
```

```
            if let entity =
NSEntityDescription.entity(forEntityName: "ManagedQuestion", in:
context) {
                questions.enumerated().forEach({ _ =
ManagedQuestion.create(with: $0.element, entity: entity,
insertInto: context) })
            }
            context.saveContext()
        })

    }
```

Κώδικας 10: "Μέθοδος προσθήκης ερωτήσεως στην εσωτερική βάση (Core data) της εφαρμογής"

Η μέθοδος saveQuestionsToDataBase είναι μια "private" μέθοδος και μπορεί να κληθεί μόνο μέσα από μεθόδους της κλάσης DataManager. Σκοπός της είναι κάθε φορά που καλείται να διαγράψει όλα τα αντικείμενα που έχουν αποθηκευτεί στα Core Data της κλάσης Question της εφαρμογής και να αποθηκεύσει έναν πίνακα από τα καινούργια αντικείμενα της κλάσης Question.

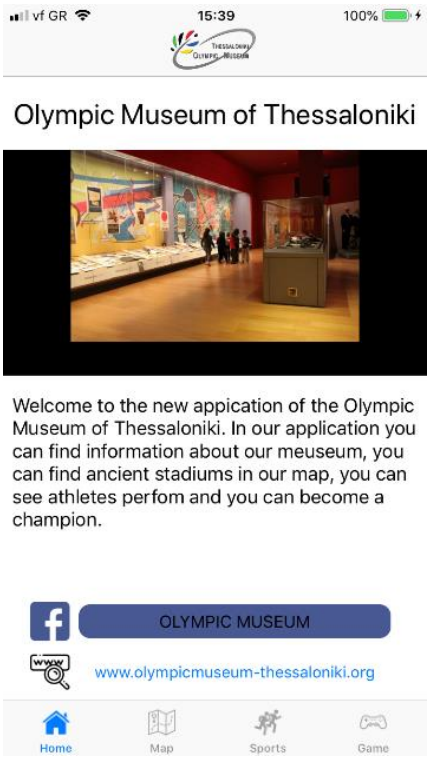
Τέλος η κλάση DataManager διαθέτει ακόμα μία function την

```
func saveMailToDataBase(mail: String) {
        let ref: DatabaseReference =
Database.database().reference()
        let emailsTable = ref.child("emails")
        emailsTable.childByAutoId().setValue(mail)
        setDeviceGaveMailToFalse(mail: mail)
    }
```

Κώδικας 11: "Μέθοδος προσθήκης mail στη βάση emails του firebase "

Η μέθοδος **saveMailToDataBase** αποθηκεύει ένα mail του χρήστη σε ένα άλλο παιδί της ριζικής αναφοράς της βάσης του firebase, το παιδί με τον μοναδικό κλειδί "emails" ως αναγνωριστικό.

Αρχική



Εικόνα 9 Διεπιφάνεια πληροφοριών εφαρμογής μουσείου

Στην αρχική σελίδα ο χρήστης βλέπει τον τίτλο του Ολυμπιακού μουσείου πάνω κεντραρισμένο και ακριβώς από κάτω υπάρχει ένα Image Slide show που εναλλάσσει φωτογραφίες του μουσείου. Ο χρήστης μπορεί με το άγγιγμα του δαχτύλου του να σταματήσει την αυτόματη αλλαγή των εικόνων καθώς και να πράξει ο ίδιος μία αλλαγή προς τα πίσω ή προς τα μπροστά. Ακριβώς από κάτω βρίσκεται ένα μήνυμα περιγραφής του μουσείου και της εφαρμογής και από κάτω βρίσκεται ένα κουμπί για να πάει τον χρήστη στο Facebook του μουσείου καθώς και ένα κουμπί για τα πάει τον χρήστη στην ιστοσελίδα του μουσείου. Είναι μία πολύ απλή διεπαφή χρήστη με τη μόνη περίπλοκη διεπαφή αυτή του imageSlideShow.

Image Slide Show

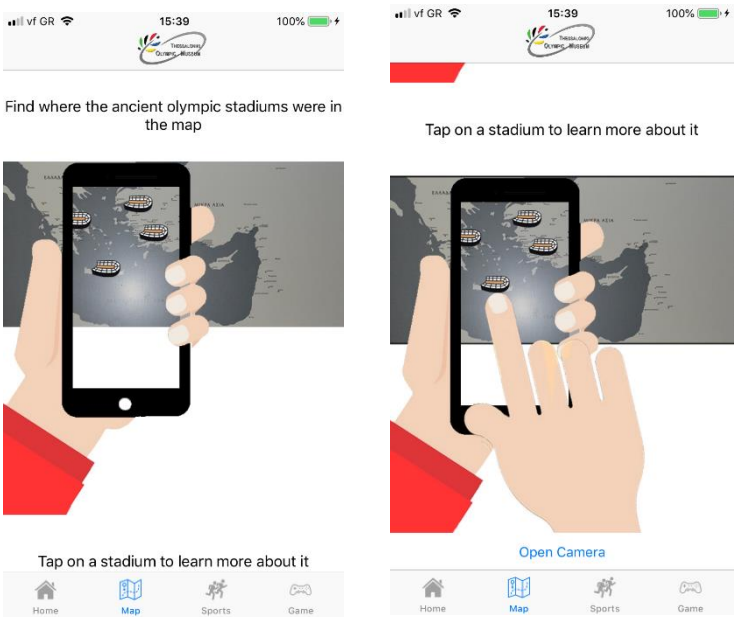
Το Image Slide Show είναι μία βιβλιοθήκη που έχει προστεθεί στο project με τη χρήση Cocoapods. Βασική του αρμοδιότητα είναι σε έναν ελεγκτή προβολής να εναλλάσσει εικόνες που του έχουν προστεθεί. Η βιβλιοθήκη αυτή υποστηρίζει κι άλλες λειτουργείες όπως να γίνει προβληθεί εικόνα σε πλήρη οθόνη και να έχει δείκτη σελίδας. Η διαμόρφωση του αντικειμένου ImageSlideShow γίνεται ως εξής:

```
@IBOutlet weak var imageSlideShow: ImageSlideshow!
let images:[UIImage] = [UIImage(named:
"insideMeuseum1")!,UIImage(named:
"insideMeuseum2")!,UIImage(named: "insideMeuseum3")!] // (!) optional
unwrapping με τη χρήση <!>
...
imageSlideShow.setImageInputs([ImageSource(image:
images[0]),ImageSource(image: images[1]),ImageSource(image:
images[2])]) imageSlideShow.backgroundColor = UIColor.black
imageSlideShow.slideshowInterval = 3.0
imageSlideShow.pageControlPosition =
PageControlPosition.underScrollView
imageSlideShow.contentScaleMode =
UIView.ContentMode.scaleAspectFit
...
```

Κώδικας 12: "Διαμόρφωση αντικειμένου κλάσης ImageSlideShow"

Στον κώδικα ορίζονται οι εικόνες που θα έχει το αντικείμενο σαν πηγή, δηλαδή οι εικόνες που έχουν ήδη οριστεί στην δήλωση μεταβλητών. Ορίζεται το μαύρο σαν χρώμα για το κομμάτι που περισσεύει στις εικόνες που δεν έχουν ratio 16:9. Ορίζεται ο αυτόματος χρόνος εναλλαγής των εικόνων να είναι τα 3 δευτερόλεπτα και τέλος ορίζεται ο τρόπος που θα μπαίνουν οι εικόνες μέσα στο Container view να είναι **UIView.ContentMode.scaleAspectFit**. Αυτό σημαίνει ότι αν είναι μεγαλύτερες σε pixel από το διαθέσιμο μέγεθος που παρέχει η υπέρ προβολή (superview) θα πρέπει να μικρύνουν για να χωρέσουν κρατώντας τις αναλογίες τους. Αν ήταν **UIView.ContentMode.scaleToFit** τότε δεν θα φαινόντουσαν ολόκληρες και ανάλογα την υλοποίηση του container ή θα μεγάλωνε η υπέρ προβολή ή δε θα φαινόταν ολόκληρη η εικόνα και θα φαινόντουσαν μόνο εκείνα τα κομμάτια που χωρούσαν ανάλογα με την υλοποίηση της υπέρ προβολής.

Augmented Map



Εικόνα 10: "Εισαγωγική διεαπαφή χρήστη πληροφοριών χάρτη"

Στην επιλογή του χάρτη αρχικά φορτώνεται ένας UIViewController ο MapIntroViewContorller που περιέχει ένα scrollview στο οποίο αναγράφονται οδηγίες για το πως ο χρήστης θα χρησιμοποιήσει την εφαρμογή για να δει τις τοποθεσίες των σταδίων που βρίσκονται στον χάρτη και στο τέλος υπάρχει το κουμπί που τον πηγαίνει στον UIViewController ARMapView που διαχειρίζεται την Augmented λειτουργία της εφαρμογής. Ο ARMapView κατά την επιλογή του μουσείου πηγαίνει τον χρήστη στον επόμενο ελεγκτή προβολής που είναι ο CityView που φαίνονται οι εικόνες του σταδίου καθώς και πληροφορίες για αυτό. Από εκεί ο χρήστης μπορεί να επιστρέψει στον ARMapView.

ARMapView

Εικόνα 11: "Επαύξηση χάρτη"

Ο ARMapView ελεγκτής προβολής υλοποιεί το ARSCNViewDelegate το οποίο είναι το πρωτόκολλο που χρησιμοποιείται να παράγονται οι πληροφορίες της σκηνής. Η οθόνη αποτελείται από ένα κουμπί για να πάει ο χρήστης πίσω στις πληροφορίες και από την sceneView όπου φαίνεται το περιεχόμενο της κάμερας καθώς και τα στοιχεία που προστίθενται σε αυτό. Όταν οριστεί το πρωτόκολλο της sceneView ναι είναι ίσο με τον ARMapView που το υλοποιεί αυτόματα ενεργοποιούνται οι δυνατότητες των function:

```
func session(_ session: ARSession, didFailWithError error: Error)
func sessionWasInterrupted(_ session: ARSession)
func sessionInterruptionEnded(_ session: ARSession)
func renderer(_ renderer: SCNSceneRenderer, nodeFor anchor:
ARAnchor) -> SCNNode?
func renderer(_ renderer: SCNSceneRenderer, didUpdate node:
SCNNode, for anchor: ARAnchor)
func renderer(_ renderer: SCNSceneRenderer, didAdd node: SCNNode,
for anchor: ARAnchor)
```

Κώδικας 13: "Μέθοδοι που χρησιμοποιούνται από ελεγκτή προβολής"

Όταν εμφανιστεί η προβολή στον χρήστη δηλαδή όταν κληθεί η μέθοδος **viewDidAppear(_ animated: Bool)** καλείται η μέθοδος **self.setUpSceneView().** Η μέθοδος αυτή αναλαμβάνει να ορίσει το κατάλληλο configuration για την σκηνή.

```
func setUpSceneView() {
        let configuration = ARWorldTrackingConfiguration()
        configuration.planeDetection = [.vertical, .horizontal]
```

```
        configuration.detectionImages =
ARReferenceImage.referenceImages(inGroupNamed: "map", bundle: nil)
        sceneView.session.run(configuration)
        self.addTapGestureToSceneView()
    }
```

Κώδικας 14: "Ορισμός διαμόρφωσης"

Αρχικά ορίζεται ότι θα χρησιμοποιηθεί το ARWorldTrackingConfiguration και ότι θα μπορεί να εντοπίσει επιφάνειες και οριζόντιες και κάθετες. Ο λόγος που επιλέγεται να μπορεί να εντοπίσει και οριζόντιες επιφάνειες ενώ γνωρίζουμε ότι ο χάρτης θα βρίσκεται σε κάθετη επιφάνεια είναι για να μη χρειάζεται τροποποίηση η εφαρμογή σε περίπτωση που μετακινηθεί ή υπάρξει και δεύτερος χάρτης. Ορίζεται η μεταβλητή detectionImages να είναι ίση με το σύνολο των εικόνων που περιέχονται στο group με όνομα "map". Στην συγκεκριμένη περίπτωση είναι μόνο μία δηλαδή ο χάρτης. Τέλος τρέχει η συνεδρία της σκηνής το configuration για να ενεργοποιηθούν όλοι οι μηχανισμοί ανίχνευσης εικόνων και επαύξησης περιεχομένου.

Επίσης προστίθεται ένας UITapGestureRecogniser στην σκηνή με σκοπό ανιχνεύονται τα αγγίγματα του χρήστη στην οθόνη και να ελέγχεται το κατά πόσο ο χρήστης πάτησε πάνω σε ένα από τα στάδια που έχουν προστεθεί στον χάρτη οθόνη. Η μέθοδος που αναλαμβάνει τον έλεγχο αυτό είναι η **addNodeToScene(withGestureRecognizer recognizer: UIGestureRecognizer)**

Αναγνώριση εικόνας

```
func renderer(_ renderer: SCNSceneRenderer, didAdd node: SCNNode,
for anchor: ARAnchor) {
if(mapIsEmpty){
        guard let imageAnchor = anchor as? ARImageAnchor else {
return }
        if let imageName = imageAnchor.referenceImage.name{
            if imageName == "ancientGreece"{
                let allStadiums = [StadumInfo.stadioNemeas,
StadumInfo.stadioDelfon, StadumInfo.stadioOlympias,.stadioRodou]
                for stadium in allStadiums{
                    let xZero =
imageAnchor.referenceImage.physicalSize.width / -2
                    let xPosition:Float = Float(xZero +
(stadium.info.xPositionCM * 0.01))
                    let zZero =
imageAnchor.referenceImage.physicalSize.height / -2
let zPosition:Float = Float(zZero + (stadium.info.yPositionCM *
0.01))
                    self.addSphereToMap(imageAnchor:
imageAnchor,node: node, name: stadium.info.name, x: xPosition, z:
zPosition,image: stadium.info.image)                }
                self.mapIsEmpty = false
                }
```

```
                    }
            }
}
```

Κώδικας 15: "Αναγνώριση εικόνας"

Η αναγνώριση εικόνας γίνεται αυτόματα από το ArKit μέσω της **renderer** delegate μεθόδου. Όταν αναγνωριστεί μία εικόνα, προστίθεται πάνω της ένα αντικείμενο τύπου ARImageAnchor που είναι υποκλάση της ARAnchor. Αφού γίνουν οι ελέγχοι για το ότι όντως είναι οι συγκεκριμένη εικόνα και ότι τα στάδια δεν έχουν ξαναμπεί στην προβολή, τα στάδια προστίθενται μέσω της **addSphereToMap** στην προβολή και ο χρήστης θα μπορεί να πατήσει για να βλέπει πληροφορίες.

Ο τύπος StadumInfo είναι ένας enumerator που έχει τόσες περιπτώσεις όσες και τα στάδια τα οποία θα προβληθούν και ανάλογα με την περίπτωση επιστρέφει τις κατάλληλες πληροφορίες σε μορφή Tuple, δηλαδή όνομα σταδίου, περιγραφή σταδίου, τη θέση της μικρής εικόνας του σταδίου στον χάρτη ως προς τον άξονα X και Y, την μικρή εικόνα που θα προστεθεί στον χάρτη, καθώς και εικόνες για να δει ο χρήστης στις λεπτομέρειες του σταδίου. Με αυτόν τον τρόπο οι αλλαγές είναι πολύ απλές και δεν χρειάζεται πολύπλοκη τροποποίηση της εφαρμογής.

func addSphereToMap

```
func addSphereToMap(imageAnchor: ARImageAnchor,node: SCNNode,
name:String, x: Float, z:Float, image:UIImage){
        let sphere = SCNNode()
        let plane = SCNPlane(width: 0.02, height: 0.02)
        plane.firstMaterial!.diffuse.contents = image
        plane.firstMaterial!.lightingModel = .constant
        sphere.geometry = plane
        sphere.eulerAngles.x = -.pi / 2
        sphere.name = name
        sphere.position = SCNVector3(x: x, y: 0, z: z)
        node.name = "parentNode"
        node.addChildNode(sphere)
        self.sceneView.scene.rootNode.addChildNode(node)
    }
```

Κώδικας 16: "Μέθοδος προσθήκης εικόνας σε σκηνή"

Στη συγκεκριμένη μέθοδο δημιουργείται μία εικόνα δηλαδή ένα SCNPlane που έχει σαν τιμή στην ιδιότητα της μεταβλητής material τύπου SCNMaterial, την μικρή εικόνα που θα φαίνεται στο επαυξημένο περιεχόμενο. Η θέσεις ορίζονται σε μορφή x, y, z ως προς την εικόνα του χάρτη. Η θέση χ ορίζει την απόσταση από αριστερά προς τα δεξιά σε εκατοστά, η θέση y ορίζει την απόσταση από το σημείο που βρίσκεται η εικόνα και πίσω δηλαδή στον 3d κόσμο, αν πάρει αρνητική τιμή φαίνεται να είναι πιο κοντά δηλαδή πιο μεγάλο, αν πάρει αρνητική τιμή φαίνεται

πιο πίσω από την εικόνα και τέλος η θέση z ορίζει την απόσταση που ξεκινάει η εικόνα από πάνω προς τα κάτω.

Αναγνώριση τοποθεσίας touch σε επαυξημένο περιβάλλον

```
@objc func addNodeToScene(withGestureRecognizer recognizer:
UIGestureRecognizer) {
            let tapLocation = recognizer.location(in:
self.sceneView)
            let hitTestOptions: [SCNHitTestOption : Any] =
[.boundingBoxOnly: false, .searchMode:
SCNHitTestSearchMode.all.rawValue]
            let hitTestResults = sceneView.hitTest(tapLocation,
options: hitTestOptions)
            if let node = hitTestResults.last?.node {
                if(node.name ?? "" != ""){
                    self.detectedShere(name: node.name ?? "")
                }else{
                    if let node = hitTestResults.first?.node {

                        if(node.name ?? "" != ""){
                            self.detectedShere(name: node.name ??
"")
                        }
                    }
                }
            }

    }
```

Κώδικας 17: "Μέθοδος αναγνώρισης αγγιγμάτων σε κόμβους της σκηνής"

Η μέθοδος **addNodeToScene** ελέγχει εάν έγινε άγγιγμα σε έναν Node και αυτός ο node έχει όνομα να καλέσει τη μέθοδο **detectedShere**. Για να γίνει αυτό βρίσκεται για το τρέχον frame η δισδιάστατη τοποθεσία του αγγίγματος πάνω στην οθόνη. Έπειτα καλείται η μέθοδος του αντικειμένου sceneView τύπου ARSCNView **hitTest** η οποία επιστρέφει ένα σύνολο από nodes οι οποίοι αγγίχτηκαν, δηλαδή αν ένας node ήταν πίσω από έναν άλλο θα επιστρέψει και τους δύο αλλιώς αν σε εκείνο το σημείο βρισκόταν μόνο ένας node η μέθοδος θα επιστρέψει έναν. Επειδή το plane που δείχνει το χώρο που καταλαμβάνει η εικόνα του χάρτη μπορεί να είναι μπροστά από την εικόνα που προστέθηκε του χάρτη ή από κάτω ελέγχονται ο πρώτος node του αποτελέσματος και ο τελευταίος. Όποιου το όνομα είναι διάφορο του τίποτα αυτός είναι και ο κόμβος που περιέχει την πληροφορία για το στάδιο.

```
func detectedShere(name:String) -> String{

        var detectedStadim:StadumInfo?
        switch name {
        case "Στάδιο Δελφών":
            detectedStadim = StadumInfo.stadioDelfon
            break
        case "Στάδιο Ολυμπίας":
            detectedStadim = StadumInfo.stadioOlympias
            break
        case "Στάδιο Νεμέας":
            detectedStadim = StadumInfo.stadioNemeas
            break
....
        case "Στάδιο Ρόδου":
            detectedStadim = StadumInfo.stadioRodou
            break
        default:
            return "Default detectedSphere name: \(name)"
        }

        if let stadiumDetected = detectedStadim{
            cityViewController?.images =
stadiumDetected.info.moreImages
            cityViewController?.cityTitle =
stadiumDetected.info.name
            cityViewController?.desc =
stadiumDetected.info.description
            self.present(cityViewController!, animated: true,
completion: nil)
            return ""
        }else{
            return "None"
        }
    }
```

Κώδικας 18: "Μέθοδος προσθήκη σταδίου σε χάρτη"

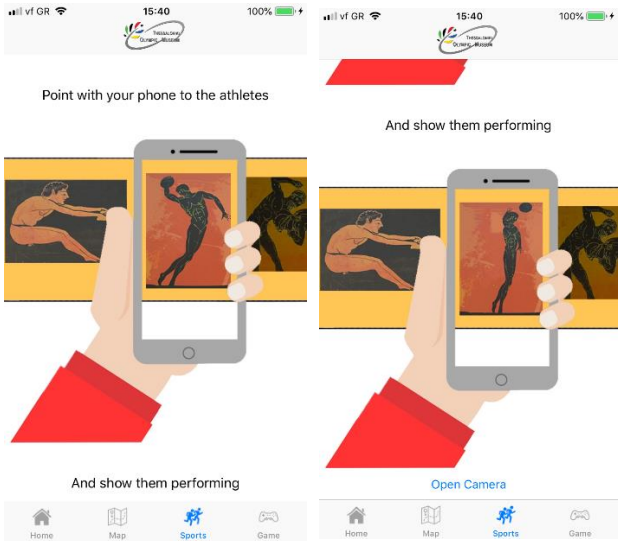
Η μέθοδος detectedShere ελέγχει το όνομα του node που βρέθηκε από την addNodeToScene και αν το όνομα αυτό είναι ίδιο με το όνομα ενός σταδίου τότε ο χρήστης μεταφέρεται στην επόμενη διεπιφάνεια δηλαδή την διεπιφάνεια που βλέπει τα στάδια.

CityView

Εικόνα 12: "Διεπαφή πληροφοριών σταδίου"

Ο χρήστης στον συγκεκριμένο ελεγκτή προβολής βλέπει σε ένα slideshow τις εικόνες για το στάδιο, τον τίτλο του σταδίου, την περιγραφή του σταδίου καθώς έχει τη δυνατότητα να πάει πίσω στην προβολή του χάρτη με ένα κουμπί.

Augmented Sports



Εικόνα 13: " Εισαγωγική διεπαφή χρήστη αγωνισμάτων "

Η τρίτη επιλογή του χρήστη από το tab menu είναι η επαύξηση περιεχομένου των εικόνων που αναπαριστούν τα αρχαία ολυμπιακά αγωνίσματα. Οι εικόνες είναι 9 και τα αγωνίσματα είναι ακόντιο, άλμα, δρόμος, ιπποδρομία, παγκράτιο, πάλη, πένταθλο, πυγμή, σφαίρα. Ο χρήστης όπως και στην επαύξηση του χάρτη αρχικά βλέπει μία εισαγωγική οθόνη τον ελεγκτή προβολής “SportsIntroViewController” με πληροφορίες για το πως θα χρησιμοποιήσει την εφαρμογή για να δει την επαύξηση του περιεχομένου και ένα κουμπί που θα τον πάει στον ελεγκτή προβολής “ImageRecognitionViewController” που αναλαμβάνει την επαύξηση περιεχομένου με την προσθήκη των κινούμενων εικόνων πάνω στις εικόνες που δίνει την εντύπωση της πως οι αθλητές ζωντανεύουν.

Image Recognition View Controller

Όπως και στον χάρτη η διεπαφή χρήστη αποτελείται από ένα view στο οποίο φαίνεται το περιεχόμενο της κάμερας και ότι άλλο προστεθεί στη συνέχεια, καθώς και ένα κουμπί από το οποίο ο χρήστης μπορεί να ανακατευθυνθεί πίσω. Πάλι χρησιμοποιούνται οι μέθοδοι του πρωτοκόλλου **ARSCNViewDelegate** καθώς και για την αναγνώριση της εικόνας η μέθοδος **renderer(_ renderer: SCNSceneRenderer, didAdd node: SCNNode, for anchor: ARAnchor)**.

Η διαφορετικότητα αυτής της περίπτωσης χρήσης με την περίπτωση του χάρτη είναι ότι δεν χρειάζεται να κρατάει ο χρήστης πληροφορίες για το περιεχόμενο που προστίθεται και ότι το περιεχόμενο προστίθεται ακριβώς στο σημείο και μέγεθος

της πραγματικής εικόνας που υπάρχει στο χώρο. Κάθε κινούμενη εικόνα που προστίθεται είναι μία .gif εικόνα και από τη στιγμή που μπει δεν αφαιρείται μέχρι ο χρήστης να βγει από την διεπαφή.

```
func renderer(_ renderer: SCNSceneRenderer, didAdd node: SCNNode,
for anchor: ARAnchor) {
        guard let imageAnchor = anchor as? ARImageAnchor else {
return }
        let referenceImage = imageAnchor.referenceImage

        DispatchQueue.main.async {
            switch referenceImage.name{
            case "diskos":
                let plane = SCNPlane(width:
referenceImage.physicalSize.width, height:
referenceImage.physicalSize.height)

                let bundleURL = Bundle.main.url(forResource:
"sfairaSmall", withExtension: "gif")
                let animation : CAKeyframeAnimation =
self.createGIFAnimation(url: bundleURL!)!
                let layer = CALayer()
                layer.bounds = CGRect(x: 0, y: 0, width: 586,
height: 800)

                layer.add(animation, forKey: "contents")
                let tempView = UIView.init(frame: CGRect(x: 0, y:
0, width: 586, height: 800))
                tempView.layer.bounds = CGRect(x: -293, y: -400,
width: tempView.frame.size.width, height:
tempView.frame.size.height)
                tempView.layer.addSublayer(layer)

                let newMaterial = SCNMaterial()
                newMaterial.isDoubleSided = true
                newMaterial.diffuse.contents = tempView.layer
                plane.materials = [newMaterial]
                let planeNode = SCNNode(geometry: plane)
                planeNode.eulerAngles.x = -.pi / 2
                node.addChildNode(planeNode)
                break

       .....
.....

                let plane = SCNPlane(width:
referenceImage.physicalSize.width, height:
referenceImage.physicalSize.height)

                let bundleURL = Bundle.main.url(forResource:
"pygmi", withExtension: "gif")
                let animation : CAKeyframeAnimation =
self.createGIFAnimation(url: bundleURL!)!
```

```
                let layer = CALayer()
                layer.bounds = CGRect(x: 0, y: 0, width: 600,
height: 397)

                layer.add(animation, forKey: "contents")
                let tempView = UIView.init(frame: CGRect(x: 0, y:
0, width: 600, height: 397))
                tempView.layer.bounds = CGRect(x: -300, y: -198,
width: tempView.frame.size.width, height:
tempView.frame.size.height)
                tempView.layer.addSublayer(layer)

                let newMaterial = SCNMaterial()
                newMaterial.isDoubleSided = true
                newMaterial.diffuse.contents = tempView.layer
                plane.materials = [newMaterial]
                let planeNode = SCNNode(geometry: plane)
                planeNode.eulerAngles.x = -.pi / 2
                node.addChildNode(planeNode)
                break
            default:
    break
  }
        }
```

Κώδικας 19: "Έλεγχος και προσθήκη animation gif σε πραγματική εικόνα"

Στην συγκεκριμένη μέθοδο ελέγχεται το όνομα της εικόνας του Resource group που έχει περαστεί σαν παράμετρος στην μεταβλητή detectionImages του configuration της session. Ανάλογα με το όνομα της εικόνας προστίθεται και η αντίστοιχη εικόνα τύπου GIF ακριβώς από πάνω. Για να γίνει αυτό πρώτα δημιουργείται ένα αντικείμενο της κλάσης SCNPlane με το φυσικό μέγεθος της εικόνας που βρέθηκε. Δημιουργείται αναφορά του animation της εικόνας μέσω της μεθόδου **createGIFAnimation** και ένα αντικείμενο της κλάσης CALayer στο οποίο προστίθεται το animation. Έπειτα δημιουργείται ένα αντικείμενο κλάσης UIView στο οποίο εισάγεται το layer που είχε δημιουργηθεί. Στη συνέχεια δημιουργείται ένα αντικείμενο τύπου SNCMaterial στο οποίο θέτεται στην μεταβλητή content το layer του UIView. Στη συνέχεια ορίζεται το αντικείμενο SCNMaterial σαν material του plane που είχε δημιουργηθεί. Δημιουργείται ένα καινούργιο αντικείμενο SCNNode με παράμετρο στον δομιτή το plane. Τέλος στον node που βρέθηκε από την μέθοδο προστίθεται σαν παιδί το αντικείμενο της κλάσης SCNNode που δημιουργήθηκε και ο χρήστης βλέπει το animation μέσω του κινητού του. Η διαδικασία επαναλαμβάνεται για κάθε εικόνα που αναγνωρίζεται και δεν έχει προστεθεί κινούμενη εικόνα για αυτή. Κάθε εικόνα τύπου GIF έχει δημιουργηθεί με τη χρήση του f Photoshop.

Εικόνα 14: "Παράδειγμα GIF εικόνας του αθλήματος πυγμής"

Μέθοδος δημιουργίας gif animation createGIFAnimation

```
func createGIFAnimation(url:URL) -> CAKeyframeAnimation? {

        guard let src = CGImageSourceCreateWithURL(url as CFURL,
nil) else { return nil }
        let frameCount = CGImageSourceGetCount(src)

        var time : Float = 0


        var framesArray = [AnyObject]()
        var tempTimesArray = [NSNumber]()

        for i in 0..<frameCount {

            // Default διάρκεια frames
            var frameDuration : Float = 0.1;

            let cfFrameProperties =
CGImageSourceCopyPropertiesAtIndex(src, i, nil)
            guard let framePrpoerties = cfFrameProperties as?
[String:AnyObject] else {return nil}
```

```
            guard let gifProperties =
framePrpoerties[kCGImagePropertyGIFDictionary as String] as?
[String:AnyObject]
                else { return nil }
```

- **Χρησιμοποιείται kCGImagePropertyGIFUnclampedDelayTime ή kCGImagePropertyGIFDelayTime**

```
            if let delayTimeUnclampedProp =
gifProperties[kCGImagePropertyGIFUnclampedDelayTime as String] as?
NSNumber {
                frameDuration = delayTimeUnclampedProp.floatValue
            } else {
                if let delayTimeProp =
gifProperties[kCGImagePropertyGIFDelayTime as String] as? NSNumber
{
                    frameDuration = delayTimeProp.floatValue
                }
            }

            if frameDuration < 0.011 {
                frameDuration = 0.100;
            }
```

- **Προσθήκη frames στον πίνακα των frames**

```
      if let frame = CGImageSourceCreateImageAtIndex(src, i, nil)
{
                tempTimesArray.append(NSNumber(value:
frameDuration))
                framesArray.append(frame)
            }

            time = time + frameDuration
        }

        var timesArray = [NSNumber]()
        var base : Float = 0
        for duration in tempTimesArray {
            timesArray.append(NSNumber(value: base))
            base += ( duration.floatValue / time )
        }
```

- **Από τη βιβλιογραφία της κλάσης 'CAKeyframeAnimation':**
- **Η πρώτη τιμή του πίνακα πρέπει να είναι 0.0 και η τελευταία πρέπει να είναι 1.0.**

```
        timesArray.append(NSNumber(value: 1.0))
```

- **Δημιουργία animation**

```
    let animation = CAKeyframeAnimation(keyPath: "contents")

        animation.beginTime = 0.0
        animation.duration = CFTimeInterval(time)
        animation.repeatCount = Float.greatestFiniteMagnitude;
        animation.isRemovedOnCompletion = false
        animation.fillMode = CAMediaTimingFillMode.forwards
        animation.values = framesArray
        animation.keyTimes = timesArray
```

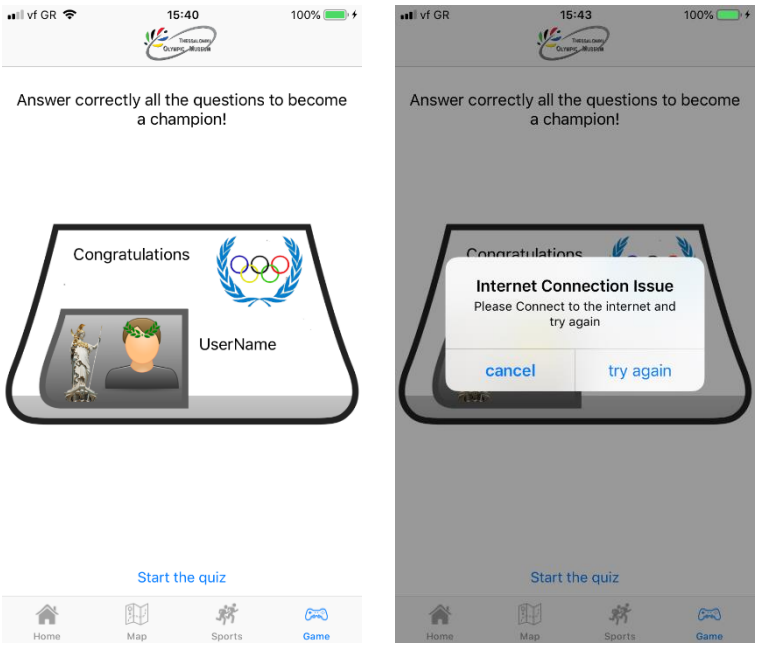
```
        animation.calculationMode =
CAAnimationCalculationMode.discrete

        return animation;
    }
```

Κώδικας 20: "Μέθοδος δημιουργίας animation"

Η function **createGIFAnimation** βρίσκεται στο αρχείο **UIIViewController+Extensions.swift** το οποίο περιλαμβάνει ένα extension της κλάσης UIViewController και κάθε μέθοδος που βρίσκεται σε αυτό το extension μπορεί να κληθεί από κάθε αντικείμενο της κλάσης UIViewController και κάθε αντικείμενο υποκλάσης UIViewController με **self.createGIFAnimation(url:)**.

Στη μέθοδο αυτή δημιουργείται το αντικείμενο τύπου CAKeyframeAnimation το οποίο μπορεί να προβληθεί μέσα από ένα αντικείμενο τύπου CALayer και να δημιουργηθεί η εμπειρία του χρήστη ότι βλέπει ένα επαυξημένο περιεχόμενο με animation και ότι ζωντανεύουν οι εικόνες.

Quiz game



Εικόνα 15: "Εισαγωγική διεπαφή χρήστη παιχνιδιού ερωτήσεων"

Η τέταρτη διεπαφή χρήστη της εφαρμογής είναι ένα παιχνίδι στο οποίο ο χρήστης έχει να απαντήσει τρεις ερωτήσεις και αν απαντήσει σωστά και στις τρείς τότε μπορεί να βγάλει φωτογραφία τον εαυτό του, η οποία θα επαυξηθεί με περιεχόμενο για να μπει ένα στεφάνι στο κεφάλι του και το άγαλμα της θεάς Νίκης. Οι ερωτήσεις που προβάλονται στο χρήστη επιλέγονται τυχαία ανάμεσα σε όσες διαθέσιμες ερωτήσεις υπάρχουν και οι διαθέσιμες απαντήσεις των ερωτήσεων δεν εμφανίζονται πάντα με την ίδια σειρά.

Επαύξηση περιεχομένου γίνεται με δύο τρόπους. Ο πρώτος είναι με τη χρήση του ARFaceTrackingConfiguration όπου αυτόματα το ArKit καταλαβαίνει το κεφάλι του χρήστη από την μπροστινή κάμερα και ενσωματώνει στο κεφάλι του το στεφάνι ελιάς. Ο δεύτερος είναι επιτρέποντας στον χρήστη να μετακινήσει με άγγιγμα το στεφάνι ελιάς και να το τοποθετήσει στο κεφάλι του εκεί που δείχνει η κάμερα.

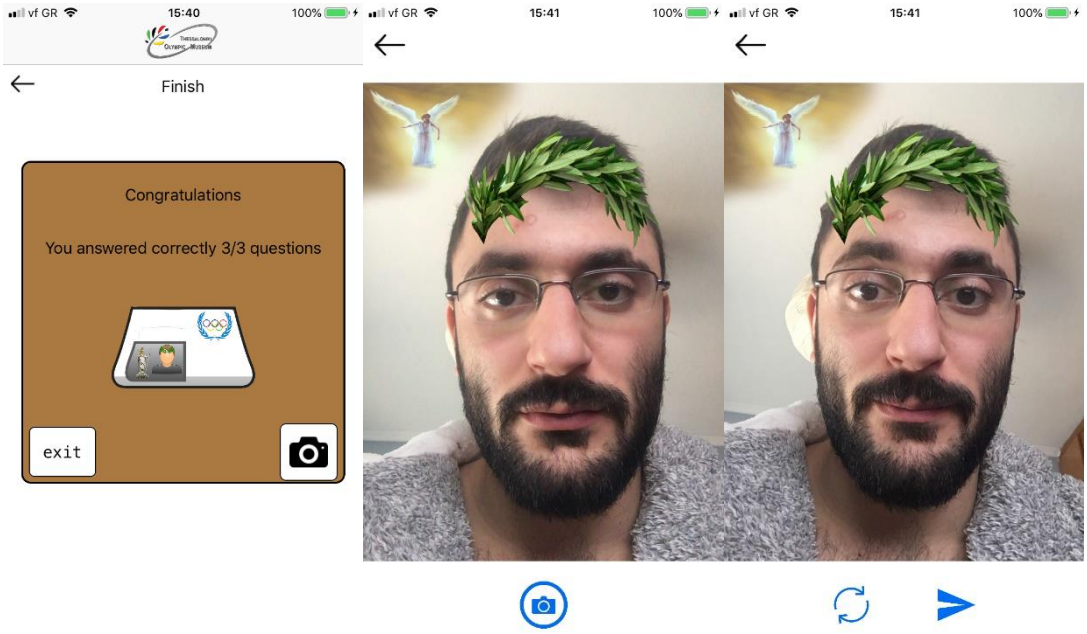
Ο λόγος για τον οποίο έπρεπε να υπάρχουν δύο προσεγγίσεις για αυτήν την υλοποίηση είναι γιατί το ARFaceTrackingConfiguration λειτουργεί μόνο για συσκευές iOS που έχουν μπροστινή truedepth camera. Οι συσκευές αυτές είναι λίγες και θα περιοριζόντουσαν αρκετοί χρήστες για αυτό έπρεπε να υπάρχει και μία εναλλακτική.

Τρόπος λειτουργίας του παιχνιδιού

Εικόνα 16: "Διεπαφές χρήστη παιχνιδιού ερωτήσεων"

Η διεπιφάνεια του παιχνιδιού αποτελείται από ένα μεγάλο αντικείμενο της κλάσης UILabel πάνω στο οποίο αναγράφεται η ερώτηση που πρέπει να απαντήσει ο χρήστης και από κάτω μέσα σε ένα αντικείμενο κλάσης UIStackView υπάρχουν τέσσερα αντικείμενα κλάσης UIButton στα οποία αναγράφονται οι τέσσερεις διαφορετικές απαντήσεις τις οποίες έχει να επιλέξει ο χρήστης. Όταν απαντήσει ο χρήστης σωστά ακούγεται ο ήχος “correct.mp3” που βρίσκεται στο σύνολο των αρχείων της εφαρμογής, με τη χρήση της κλάσης SoundHandler και το κουμπί γίνεται πράσινο. Τα κουμπιά απενεργοποιούνται για 2 δευτερόλεπτα έτσι ώστε να μην μπορεί ο χρήστης να τα πατήσει και μετά από 2 δευτερόλεπτα φορτώνεται η καινούργια ερώτηση στη διεπιφάνεια χρήστη. Όταν ο χρήστης απαντήσει λάθος σε μία ερώτηση ακούγεται ο ήχος “error.wav” η απάντηση που έδωσε γίνεται κόκκινη ενώ η σωστή απάντηση γίνεται πράσινη και ομοίως απενεργοποιούνται τα κουμπιά και φορτώνεται η καινούργια ερώτηση. Για την αναπαραγωγή κάθε ήχου χρησιμοποιείται το ο AVAudioPlayer. Όταν δεν απαντήσει σωστά σε όλες τις ερωτήσεις τότε του δίνεται η δυνατότητα να ξαναπροσπαθήσει. Όταν ο χρήστης απαντήσει σε όλες τις ερωτήσεις σωστά τότε του δίνεται η δυνατότητα να προχωρήσει στην επαύξηση του περιεχομένου της μπροστινής του κάμερας και να μπει το στεφάνι στο κεφάλι του για να στέφει ολυμπιονίκης. Ανάλογα με το αν η

συσκευή το υποστηρίζει ARFaceTracking θα προηγηθεί σε επαύξηση περιεχομένου με ARKit ή χωρίς ARkit. Και στις δύο περίπτωσης ο χρήστης θα πλοηγηθεί στον ελεγκτή προβολής “SelfieChampionScreenViewController” και μέσα στη μέθοδο **viewDidAppear** θα ελέγχει μέσα από την static μέθοδο **ARFaceTrackingConfiguration.isSupported** αν η συσκευή υποστηρίζει ARFaceTracking.



Εικόνα 17: "Διεπαφές χρήστη σε περίπτωση επιτυχίας"

Επαύξηση περιεχομένου χωρίς ARKit

Στην συγκεκριμένη περίπτωση ο χρήστης βλέπει τον εαυτό του μέσα από την κάμερα καθώς και το στεφάνι το οποίο μπορεί να το τοποθετήσει όπου αυτός κρίνει. Από κάτω φαίνεται το κουμπί της κάμερα το οποίο αναλαμβάνει να τραβήξει την φωτογραφία. Όταν ο χρήστης τραβήξει την φωτογραφία του δίνεται η δυνατότητα αποστολής της φωτογραφίας μέσω mail ή να ξανατραβήξει αν δεν του άρεσε.

Για να μπορεί ο χρήστης να βλέπει τον εαυτό του από την μπροστινή κάμερα χρησιμοποιούνται τα αντικείμενα των κλάσεων από το πλαίσιο “AVFoundation” και οι delegate μέθοδοι του πρωτοκόλλου “AVCaptureVideoDataOutputSampleBufferDelegate”.

Αρχικά δημιουργείται ένα αντικείμενο της κλάσης AVCaptureSession του οποίου η αρμοδιότητα είναι να χειρίζεται το stream της κάμερας. Στη συνέχεια ένα αντικείμενο ελέγχου της μπροστινής κάμερα τύπου AVCaptureDevice, ένα αντικείμενο της κλάσης AVCapturePhotoOutput το οποίο θα χειριστεί την προσωρινή αποθήκευση της εικόνας και τέλος ένα αντικείμενο της κλάσης AVCaptureVideoPreviewLayer το οποίο χειρίζεται την προβολή του video μέσα από ένα αντικείμενο κλάσης UIView.

Μέσα στην κλάση **viewDidAppear** καλούνται με τη σειρά οι μέθοδοι

- **setupCaptureSession()**
- **setUpDevice()**
- **setUpInputOutput()**
- **setupPreviewLayer()**
- **startRunningCaptureSession()**

```
func setupCaptureSession(){
        captureSession.sessionPreset =
AVCaptureSession.Preset.photo
}
```

Κώδικας 21: "Μέθοδος ορισμού του τύπου της συνεδρίας"

Η μέθοδος ορίζει τον τύπο της session που έχει το αντικείμενο captureSession.

```
func setUpDevice(){
        let deviceDescoverySession =
AVCaptureDevice.DiscoverySession(deviceTypes:
[AVCaptureDevice.DeviceType.builtInWideAngleCamera], mediaType:
AVMediaType.video, position: AVCaptureDevice.Position.front)
        let devices = deviceDescoverySession.devices
        for device in devices {
            if device.position == AVCaptureDevice.Position.front {
                self.frontCamera =
deviceDescoverySession.devices.first
            }
        }
}
```

Κώδικας 22: "Μέθοδος που ορίζει από ποια κάμερα θα ανοίξει το stream"

Η μέθοδος setUpDevice αναγνωρίζει όλες τις κάμερες του κινητού και ορίζει σαν κάμερα την μπροστινή κάμερα.

```
func setUpInputOutput(){
        do {
```

```
            let captureDecviceInput = try
AVCaptureDeviceInput(device: self.frontCamera)
            captureSession.addInput(captureDecviceInput)
            photoOutput = AVCapturePhotoOutput()

photoOutput?.setPreparedPhotoSettingsArray([AVCapturePhotoSettings
(format: [AVVideoCodecKey: AVVideoCodecType.jpeg])],
completionHandler: nil)
            captureSession.addOutput(photoOutput!)
        } catch {
            print(error)
        }
    }
```

Κώδικας 23: "Αρχικοποίηση παραμέτρων ροής"

Η μέθοδος setUpInputOutput αρχικοποιεί τις παραμέτρους της ροής του βίντεο που θα προβάλλεται μέσα από ένα αντικείμενο κλάσης UIView.

```
func setupPreviewLayer(){
        cameraPreviewLayer = AVCaptureVideoPreviewLayer(session:
captureSession)
        cameraPreviewLayer?.videoGravity =
AVLayerVideoGravity.resizeAspect
        cameraPreviewLayer?.connection?.videoOrientation =
AVCaptureVideoOrientation.portrait
        cameraPreviewLayer?.frame = CGRect(x: 0, y: 0, width:
self.cameraImageView.frame.width, height:
self.cameraImageView.frame.height)

self.cameraImageView.layer.insertSublayer(cameraPreviewLayer!, at:
0)

    }
```

Κώδικας 24: "Ενσωμάτωση βίντεο σε αντικείμενο κλάσης UIView"

Η μέθοδος **setupPreviewLayer** ενσωματώνει το βίντεο που θα προβάλλεται στο αντικείμενο της κλάσης UIView που υπάρχει στην οθόνη του χρήστη.

```
func startRunningCaptureSession(){
captureSession.startRunning()
}
```

Κώδικας 25: "Αρχικοποίηση της ροής του βίντεο"

Τέλος η μέθοδος **startRunningCaptureSession** ξεκινάει τη ροή του βίντεο και ο χρήστης μπορεί να δει το περιεχόμενο που προβάλλεται από την μπροστινή του κάμερα.

```
@IBAction func didClickCaptureButton(_ sender: Any) {
        photoOutput?.capturePhoto(with: AVCapturePhotoSettings(),
delegate: self)
        self.captureButton.isHidden = true
        self.reloadImageButton.isHidden = false
        self.sendButton.isHidden = false
    }
```

Κώδικας 26: "Τέλος συνεδρίας"

Όταν ο χρήστης πατήσει το κουμπί της κάμερας καλείται η μέθοδος **didClickCaptureButton** η οποία σταματάει το session της ροής και αποθηκεύει στον χρήστη το τελευταίο frame που προβλήθηκε στο view που προβαλλόταν το περιεχόμενο, κρύβει το κουμπί της κάμερας και εμφανίζει τα κουμπιά για να μπορέσει ο χρήστης να ξανατραβήξει φωτογραφία ή να τη στείλει. Η μέθοδος capturePhoto του αντικειμένου photoOutput της κλάσης **AVCapturePhotoOutput** είναι μία delegate μέθοδος του πρωτοκόλλου **AVCapturePhotoCaptureDelegate** το οποίο υλοποιεί σε extension ο ελεγκτής προβολής **SelfieChampionScreenViewController**.

```
extension SelfieChampionScreenViewController:
AVCapturePhotoCaptureDelegate {
    func photoOutput(_ output: AVCapturePhotoOutput,
didFinishProcessingPhoto photo: AVCapturePhoto, error: Error?) {
        if let imageData = photo.fileDataRepresentation() {
            let capturedTempImage = UIImage(data: imageData)
            self.capturedImage = UIImage(cgImage:
capturedTempImage!.cgImage!, scale: capturedTempImage!.scale,
orientation: .leftMirrored)
            self.cameraImageView.image = self.capturedImage
            self.captureSession.stopRunning()
            self.captureSession = AVCaptureSession()
            self.frontCamera = nil
            self.photoOutput = nil
            self.cameraPreviewLayer?.removeFromSuperlayer()
            self.cameraPreviewLayer = nil
        }
    }
}
```

Κώδικας 27: "Ελευθέρωση μνήμης από μεταβλητές που δεν χρειάζονται"

Στην ουσία αυτό που γίνεται είναι μίας μορφής screenshot στο κομμάτι που προβαλλόταν το βίντεο και είχαν προστεθεί οι εικόνες του στεφανιού στο κεφάλι του χρήστη καθώς και του αγάλματος της θεάς νίκης. Στη συνέχεια η εικόνα που παράγεται φορτώνεται στο UIView που πρόβαλε τη ροή του βίντεο και αφαιρούνται από τη μνήμη της εφαρμογής τα αντικείμενα που είχαν χρησιμοποιηθεί για τον έλεγχο ροής του βίντεο.

Μετακίνηση στεφανιού με ανίχνευση αγγιγμάτων

Η μετακίνηση του στεφανιού με το άγγιγμα γίνεται μέσω των μεθόδων **touchesBegan, touchesMoved** και **touchesEnded** οι οποίες είναι μέθοδοι της κλάσης UIViewContorller.

```
override func touchesBegan(_ touches: Set<UITouch>, with event:
UIEvent?) {
if manualWay {
        if let touch = touches.first {
            if (touch.location(in: self.stefaniImageView).x > 0 &&
touch.location(in: self.stefaniImageView).y > 0 &&
touch.location(in: self.stefaniImageView).x <
self.stefaniImageView.frame.width && touch.location(in:
self.stefaniImageView).y < self.stefaniImageView.frame.height) {
                self.moveStefani = true
            }
        }
}
    }
```

Κώδικας 28: "Μέθοδος για αρχικοποίηση των αγγιγμάτων για μετακίνηση στεφανιού"

Στη μέθοδο touchesBegan ελέγχεται το κατά πόσο το άγγιγμα του χρήστη βρίσκεται πάνω στο στεφάνι. Αν βρίσκεται τότε η μεταβλητή **moveStefani,** η οποία είναι μία τύπου Boolean και ορίζει αν το στεφάνι πρέπει να μετακινηθεί στην touchesMoved, παίρνει την τιμή true.

```
    override func touchesMoved(_ touches: Set<UITouch>, with
event: UIEvent?) {
if manualWay {
        if let touch = touches.first {
            if moveStefani {
                let newPosition = touch.location(in: self.view)
                self.stefaniImageView.center = newPosition
                self.view.layoutIfNeeded()
            }
        }
}
    }
```

Κώδικας 29: " Μετακίνηση στεφανιού με βάση τα αγγίγματα "

Στην μέθοδο touchesMoved εφόσον η μεταβλητή moveStefani έχει την τιμή true ορίζεται το κέντρο της θέσης του στεφανιού να είναι στο κέντρο του αγγίγματος του χρήστη. Όσο ο χρήστης μετακινεί το δάχτυλο του η διαδικασία επαναλαμβάνεται και το στεφάνι μετακινείται.

```
override func touchesEnded(_ touches: Set<UITouch>, with event:
UIEvent?) {
```

```
if manualWay {
            self.moveStefani = false
}
 }
```

Κώδικας 30: "Ολοκλήρωση μετακίνησης στεφανιού"

Όταν ο χρήστης αφήσει το δάχτυλό του από την οθόνη του κινητού καλείται η μέθοδος **touchesEnded** η οποία απλά αλλάζει την τιμή της μεταβλητής **moveStefani** σε false. Το στεφάνι παραμένει στην τελευταία θέση που ήταν από την μέθοδο **touchesChanged**. Ο χρήστης έχει τη δυνατότητα να μην προσθέσει το στεφάνι στην φωτογραφία απλά αφαιρώντας το από το κομμάτι της οθόνης όπου προβάλλεται η ροή της κάμερας. Ο χρήστης δεν μπορεί να αφαιρέσει την εικόνα της θεάς Αθηνάς.

Επαύξηση περιεχομένου με ARKit

Κατά την δημιουργία των περιεχομένων του ελεγκτή προβολής **SelfieChampionScreenViewController** στο ίδιο σημείο που βρίσκεται το αντικείμενο της κλάσης UIView που περιείχε τη ροή του βίντεο για επαύξηση περιεχομένου χωρίς ARKit έχει προστεθεί ένα αντίστοιχο αντικείμενο τύπου ARSCNView το οποίο όμως έχει την ιδιότητα hidden ίση με true το οποίο σημαίνει ότι δεν φαίνεται. Όταν η συσκευή υποστηρίζει ARFaceTracking τότε η το αντικείμενο τύπου ARSCNView παίρνει την στην ιδιότητα hidden την τιμή false και εμφανίζεται και το αντικείμενο τύπου UIView παίρνει την ιδιότητα false. Η εικόνα της θεάς Αθηνάς φαίνεται κανονικά όπως πριν καθώς βρισκόταν πάνω και από τα δύο αντικείμενα και δημιουργείται το αντικείμενο της κλάσης ARFaceTrackingConfiguration το οποίο περνιέται παραμετρικά στη μέθοδο **run** της session του αντικειμένου της κλάσης sceneView.

```
let configuration = ARFaceTrackingConfiguration()
configuration.isLightEstimationEnabled = true
 sceneView.session.run(configuration, options: [.resetTracking,
.removeExistingAnchors])
self.setUpInputOutput()
```

Κώδικας 31: "Ορισμός διαμόρφωσης προσώπου"

Στη συνέχεια αναλαμβάνουν οι μέθοδοι του ARSCNViewDelegate να προσθέσουν περιεχόμενο στην σκηνή.

Για την ανίχνευση της τοποθεσίας του προσώπου χρησιμοποιούνται τα αντικείμενα της κλάσης ARFaceAnhor και ARFaceGeometry. Το ARFaceAnchor κληρονομεί από την ARAnchor. Δηλαδή είναι θέσεις στον πραγματικό κόσμο που παρακολουθούνται από το ARKit, οι οποίες δεν κινούνται όταν μετακινείτε η

συσκευή του χρήστη. Τα αντικείμενα της κλάσης ARFaceAnchor περιλαμβάνουν επιπλέον πληροφορίες σχετικά με ένα πρόσωπο, όπως τοπολογία και έκφραση. Το αντικείμενο της κλάσης ARFaceGeometry είναι μια τρισδιάστατη περιγραφή ενός προσώπου που περιλαμβάνει κορυφές και συντεταγμένες υφής.

Η κλάση ARSCNFaceGeometry χρησιμοποιεί τα δεδομένα από ένα αντικείμενο ARFaceGeometry για να δημιουργήσει ένα αντικείμενο κλάσης SCNGeometry, το οποίο μπορεί να χρησιμοποιηθεί για τη δημιουργία nodes SceneKit.

Τέλος καλείται μόνο η μέθοδος **setUpInputOutput()** για να γίνει το screenshot στο κομμάτι που εμφανίζεται το επαυξημένο περιεχόμενο.

```
func renderer(_ renderer: SCNSceneRenderer, nodeFor anchor:
ARAnchor) -> SCNNode? {
    guard let device = sceneView.device else {
      return nil
    }
    let faceGeometry = ARSCNFaceGeometry(device: device)
    let node = SCNNode(geometry: faceGeometry)
    node.geometry?.firstMaterial?.fillMode = .none
    return node
}
```

Κώδικας 32: "Εύρεση και τοποθέτηση κόμβου"

Η μέθοδος **renderer(_ renderer: SCNSceneRenderer, nodeFor anchor: ARAnchor) -> SCNNode?** χρησιμοποιείται για να βρεθεί και να τοποθετηθεί ένας κόμβος ο οποίος αντιπροσωπεύει τα χαρακτηριστικά του κεφαλιού του χρήστη που χρησιμοποιεί την εφαρμογή.

```
func renderer(
  _ renderer: SCNSceneRenderer,
  didUpdate node: SCNNode,
  for anchor: ARAnchor) {

  guard let faceAnchor = anchor as? ARFaceAnchor,
    let faceGeometry = node.geometry as? ARSCNFaceGeometry else {
      return
  }
If stefaniAdded = false {
       self.addStefaniToFace(anchor: anchor,node: node, name:
"stefani", image:UIImage(named: "stefani"))
       stefaniAdded = true
}
updateFeatures(for: node, using: faceAnchor)
}
```

Κώδικας 33: "Μέθοδος ανανέωσης της θέσης του στεφανιού"

Η μέθοδος func **renderer(_ renderer: SCNSceneRenderer, didUpdate node: SCNNode, for anchor: ARAnchor)** καλείται κάθε φορά που αλλάζει θέση ο κόμβος του κεφαλιού. Στην προκειμένη περίπτωση αν δεν έχει προστεθεί ο κόμβος που φαίνεται το στεφάνι προστίθεται και γίνεται ανανέωση της θέσης του στεφανιού. Η μέθοδος αυτή καλείται για κάθε μετακίνηση του κεφαλιού δηλαδή σχεδόν συνέχεια. Στη συνέχεια γίνεται η ανανέωση της θέσης των παιδιών της γεωμετρίας του κόμβου προσώπου.

```
func addStefaniToFace(anchor: ARAnchor,node: SCNNode, name:String,
image:UIImage){

        let sphere = SCNNode()
        let plane = SCNPlane(width: 0.02, height: 0.02)
        plane.firstMaterial!.diffuse.contents = image
        plane.firstMaterial!.lightingModel = .constant
        sphere.geometry = plane
        sphere.name = name
        node.name = name
        node.addChildNode(sphere)
        self.sceneView.scene.rootNode.addChildNode(node)

    }
```

Κώδικας 34: "Δημιουργεία αντικειμένου SCNNode για προσθήκη στεφανιού"

Στην μέθοδο func **addStefaniToFace(anchor: ARAnchor,node: SCNNode, name:String, image:UIImage)** δημιουργείται ένα αντικείμενο SCNPlane στο οποίο φαίνεται η εικόνα του στεφανιού το οποίο προστίθεται στο κεφάλι του χρήστη. Ο κόμβος αυτός προστίθεται στο αντικείμενο του κόμβου της γεωμετρίας του προσώπου.

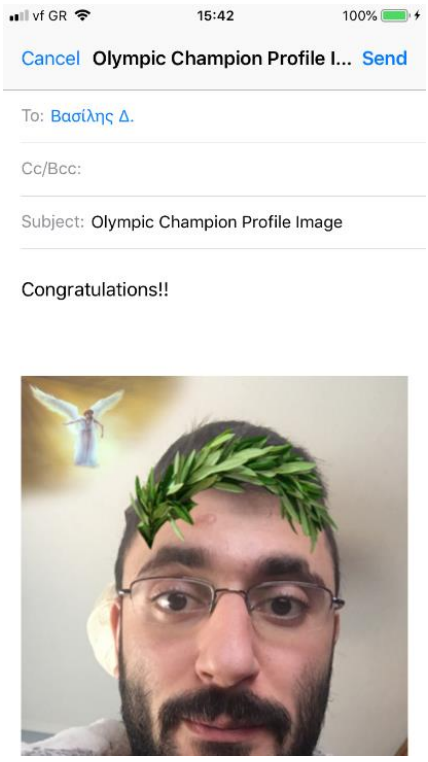
```
func updateStefaniPosition(for node: SCNNode, using anchor:
ARFaceAnchor) {
  let child = node.childNode(withName: "stefani", recursively:
false) as? SCNNode
  let vertices = [anchor.geometry.vertices[20]]
  child?.updatePosition(for: vertices)
}
```

Κώδικας 35: "Ανανέωση θέσης στεφανιού"

Η μέθοδος updateStefaniPosition(for node: SCNNode, using anchor: ARFaceAnchor) είναι η μέθοδος από την οποία τοποθετείται το στεφάνι στο σωστό μέρος του κεφαλιού. Η κλάση ARFaceGeometry έχει 1220 θέσεις και η θέση 20 αντιστοιχεί στη θέση του κέντρου πάνω του κεφαλιού.

Για τη λήψη της φωτογραφίας χρησιμοποιείται η ίδια τεχνική με την περίπτωση που ο χρήστης δεν χρησιμοποιούσε το ARKit και γίνεται και όταν ο χρήστης πατήσει το κουμπί της λήψης φωτογραφίας καλείται η μέθοδος photoOutput και με τον ίδιο τρόπο εμφανίζεται στον χρήστη η εικόνα που τράβηξε και σταματάει το session της sceneView.

Αποστολή φωτογραφίας σε email



Εικόνα 18: "Διεπαφή χρήστη για αποστολή email"

Η αποστολή της φωτογραφίας στο email του χρήστη γίνεται με τη χρήση του πρωτοκόλλου **MFMailComposeViewControllerDelegate** του πλαισίου MessageUI. Το πρωτόκολλο ορίζει τη μέθοδο που πρέπει να εφαρμόσει ο ελεγκτής προβολής **SelfieChampionScreenView** για τη διαχείριση της διεπαφής σύνθεσης αλληλογραφίας. Η μέθοδος του πρωτοκόλλου ειδοποιεί τον ελεγκτή προβολής όταν ο χρήστης έχει τελειώσει με τη διασύνδεση και είναι έτοιμος να τον απορρίψει. Η μέθοδος που έχει σαν ρόλο την παρουσίαση του παραθύρου για την σύνθεση του email είναι η **@IBAction func didClickSendButton(_ sender: Any).**

```
@IBAction func didClickSendButton(_ sender: Any) {
        if MFMailComposeViewController.canSendMail() {
            let mail = MFMailComposeViewController()
```

```
            mail.mailComposeDelegate = self

mail.setToRecipients([DataManager.shared.getDeviceGaveMail()])

            mail.setSubject("Olympic Champion Profile Image")
            mail.setMessageBody("<p>Congratulations!!</p>",
isHTML: true)

            let data = cameraContainerView!.pngData()
            mail.addAttachmentData(data!, mimeType: "image/jpg",
fileName: "image")

            present(mail, animated: true)
        }
    }
```

Κώδικας 36: "Μέθοδος αποστολής mail"

Η μέθοδος **didClickSendButton** καλείται από το άγγιγμα του χρήστη στο κουμπί της αποστολής. Στη μέθοδο αυτή ελέγχεται αν στη συσκευή επιτρέπεται η αποστολή email και αν ναι τότε παρουσιάζεται ο ελεγκτής προβολής **MFMailComposeViewController** με έτοιμη την αποστολή του email από το email του χρήστη που έχει θέσει στην συσκευή προς το email του χρήστη που έχει δώσει στην εφαρμογή. Ο χρήστης στον MFMailComposeViewController έχει την δυνατότητα της επεξεργασίας των στοιχείων της σύνθεσης του email.

Εικόνα 19: "Τελική απεσταλμένη εικόνα από email"

ΕΠΙΛΟΓΟΣ

Η εφαρμογή του Ολυμπιακού μουσείου είναι μία αρκετά σύγχρονη εφαρμογή ως προς τις δυνατότητες της. Υπάρχουν πολλές ακόμα δυνατότητες που μπορούν να γίνουν καθώς και πολλές διορθώσεις κυρίως σε γραφικά κομμάτια. Ο κώδικας της εφαρμογής δεν είναι περίπλοκος και είναι εύκολα επεκτάσιμος και τροποποιήσιμος. Επίσης τα δεδομένα της εφαρμογής είναι εύκολα τροποποιήσιμα χωρίς την χρήση προγραμματισμού, από απλές αλλαγές στην κονσόλα του Firebase και αυτό είναι ένα από τα μεγάλα προτερήματα της εφαρμογής καθώς γίνεται για το Ολυμπιακό μουσείο στο οποίο θα ήταν δύσκολο να χρειάζεται να γίνει update για μικρές αλλαγές όπως η προσθήκη ερωτήσεων.

ΣΥΜΠΕΡΑΣΜΑΤΑ

Όσον αφορά τον προγραμματισμό για κινητές συσκευές που έχουν το λειτουργικό σύστημα iOS, είναι αρκετά εύκολος τόσο λόγο της απλότητας που έχει το εργαλείο γραφής δηλαδή το xCode όσο και λόγο της γλώσσας Swift, η οποία λύνει τα χέρια του προγραμματιστή σε πολλές περιπτώσεις και κάνει τον προγραμματισμό όσο πιο απλό γίνεται. Επίσης υπάρχει αρκετή υποστήριξη από την Apple για τον συγκεκριμένο προγραμματισμό καθώς και μεγάλη κοινότητα στο διαδίκτυο όπου μπορεί να βοηθήσει σε προβλήματα.

Για την ολοκλήρωση της εφαρμογής από έναν προγραμματιστή χωρίς ομάδα το κακό είναι ότι πρέπει να είναι και developer και αναλυτής και σχεδιαστής κάτι το οποίο είναι αρκετά χρονοβόρο και αν δεν είναι καλός σε ένα από τα τρία φαίνεται στο αποτέλεσμα. Η χρήση αρκετών από τα εργαλεία ήταν μια καινούργια γνώση. Η λειτουργία επαυξημένης πραγματικότητας είναι καινοτόμα και θα χρησιμοποιηθεί αρκετά στο μέλλον. Η Apple το έδειξε αυτό καθώς στο ARKit 2 προσθεσε ακόμα περισσότερες δυνατότητες για τους προγραμματιστές. Από την πλευράς του σχεδιαστή η αρχική προσέγγιση ήταν να μπουν 3d αντικείμενα στο χώρο αλλά κατά την υλοποίηση του έργου διαπιστώθηκε πως λόγο των χαμηλών δυνατοτήτων της κάρτας γραφικών των συσκευών το αποτέλεσμα δεν θα ήταν αρεστό, με αποτέλεσμα να πρέπει να γίνουν 3d αντικείμενα που προβάλλουν 2d περιεχόμενο και το τελικό αποτέλεσμα των animation ήταν ικανοποιητικό.

Μελλοντικά στην εφαρμογή είναι εφικτό να προσθεθούν αρκετές λειτουργίες όπως όταν αναγνωρίζεται μία εικόνα ή ένα έκθεμα να ακούει ο χρήστης τις πληροφορίες που εναλλακτικά θα έπρεπε να διαβάσει ή να ακούσει από κάποιον ξεναγό, και να εμφανίζονται βέλη που δείχνουν προς το κομμάτι που επεξηγείται στον ακροατή. Επίσης η επαύξηση ενός χώρου με βίντεο είναι μία αρκετά απλή επέκταση της εφαρμογής η οποία θα μπορούσε να υλοποιηθεί.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] https://developer.apple.com/documentation/
[2] https://developer.apple.com/xcode/
[3] https://swift.org/documentation/
[4] https://developer.apple.com/swift-playgrounds/
[5] https://developer.apple.com/library/archive/documentation/General/Conceptual/Devpedia-CocoaApp/Storyboard.html
[6] https://docs.swift.org/swift-book/LanguageGuide/Closures.html
[7] https://docs.swift.org/swift-book/ReferenceManual/Types.html#grammar_tuple-type
[8] https://docs.swift.org/swift-book/LanguageGuide/Generics.html
[9] https://docs.swift.org/swift-book/LanguageGuide/ClassesAndStructures.html
[10] https://developer.apple.com/documentation/swift/array/3017522-map
[11] https://developer.apple.com/documentation/swift/string/2893557-filter
[12] https://docs.swift.org/swift-book/ReferenceManual/Statements.html
[13] https://developer.apple.com/download/
[14] https://developer.apple.com/library/archive/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html
[15] https://developer.apple.com/design/human-interface-guidelines/ios/visual-design/adaptivity-and-layout/
[16] https://cocoapods.org/
[17] https://developer.apple.com/documentation/avfoundation/avaudioplayer
[18] https://developer.apple.com/documentation/coredata
[19] https://developer.apple.com/documentation/foundation/userdefaults
[20] https://developer.apple.com/documentation/messageui
[21] https://developer.apple.com/library/archive/documentation/General/Conceptual/DevPedia-CocoaCore/Cocoa.html
[22] https://firebase.google.com/docs/
[23] http://www.gwtproject.org/articles/mvp-architecture.html
[24] https://code.msdn.microsoft.com/How-to-implement-MVVM-71a65441
[25] https://www.objc.io/issues/13-architecture/viper/

ΟΔΗΓΟΣ ΧΡΗΣΗΣ ΛΟΓΙΣΜΙΚΟΥ

Για την χρήση του λογισμικού απαιτείται κινητή συσκευή ή tablet με λειτουργικό σύστημα iOS 11.3 και πάνω. Δεν έχει βγει ακόμα επίσημη έκδοση στο Appstore της apple αλλά όταν βγει ο χρήστης θα μπορεί να κατεβάσει την εφαρμογή από το Appstore.