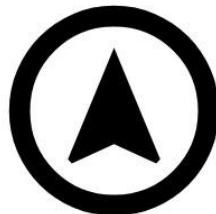


ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Σύστημα πλοήγησης σε εσωτερικούς χώρους για
άτομα με προβλήματα όρασης»



Indoor Nav

Της φοιτήτριας:
Μαρίας-Θεοδοσίας
Κωνσταντινίδου
Αρ. Μητρώου: 2021075

Επιβλέπων
Ονοματεπώνυμο: Γεώργιος
Κοκκώνης
Βαθμίδα: Αναπληρωτής
Καθηγητής

4 Σεπτεμβρίου 2026

Τίτλος Δ.Ε: Σύστημα πλοήγησης σε εσωτερικούς χώρους για άτομα με προβλήματα όρασης

Κωδικός Δ.Ε. 26117

Ονοματεπώνυμο φοιτήτριας: Μαρία-Θεοδοσία Κωνσταντινίδου

Ονοματεπώνυμο εισηγητή: Γεώργιος Κοκκώνης

Ημερομηνία ανάληψης Δ.Ε. 17-02-2026

Ημερομηνία περάτωσης Δ.Ε. 04-09-2026

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Η παρούσα εργασία αποτελεί πνευματική ιδιοκτησία της φοιτήτριας Μαρίας-Θεοδοσίας Κωνσταντινίδου που την εκπόνησε/αν. Στο πλαίσιο της πολιτικής ανοικτής πρόσβασης, ο συγγραφέας/δημιουργός εκχωρεί στο Διεθνές Πανεπιστήμιο της Ελλάδος άδεια χρήσης του δικαιώματος αναπαραγωγής, δανεισμού, παρουσίασης στο κοινό και ψηφιακής διάχυσης της εργασίας διεθνώς, σε ηλεκτρονική μορφή και σε οποιοδήποτε μέσο, για διδακτικούς και ερευνητικούς σκοπούς, άνευ ανταλλάγματος. Η ανοικτή πρόσβαση στο πλήρες κείμενο της εργασίας, δεν σημαίνει καθ' οιονδήποτε τρόπο παραχώρηση δικαιωμάτων διανοητικής ιδιοκτησίας του συγγραφέα/δημιουργού, ούτε επιτρέπει την αναπαραγωγή, αναδημοσίευση, αντιγραφή, πώληση, εμπορική χρήση, διανομή, έκδοση, μεταφόρτωση (downloading), ανάρτηση (uploading), μετάφραση, τροποποίηση με οποιονδήποτε τρόπο, τμηματικά ή περιληπτικά της εργασίας, χωρίς τη ρητή προηγούμενη έγγραφη συναίνεση του συγγραφέα/δημιουργού.

Η έγκριση της διπλωματικής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραίτητως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

«Στην οικογένεια μου»

Πρόλογος

Η παρούσα διπλωματική εργασία διερευνά τη χρήση σύγχρονων τεχνολογιών για την αντιμετώπιση της ασφαλούς και ανεξάρτητης πλοήγησης για άτομα με προβλήματα όρασης. Η δυνατότητα αυτόνομης μετακίνησης, χωρίς δηλαδή την εξάρτηση από βλέποντα συνοδό, αποτελεί σημαντικό κομμάτι στην ανεξαρτησία και συνεπώς, στην ποιότητα ζωής των ατόμων αυτών. Ειδικότερα, σε άγνωστα ή σύνθετα εσωτερικά περιβάλλοντα, όπως εμπορικά κέντρα ή πανεπιστημιακά ιδρύματα, ο εντοπισμός συγκεκριμένων προορισμών και η ασφαλής μετάβαση προς αυτούς αποτελούν σημαντικές προκλήσεις. Βασικός στόχος της εργασίας είναι η ανάπτυξη μίας λύσης που επιτρέπει τον ακριβή προσανατολισμό και πλοήγηση των χρηστών σε εσωτερικούς χώρους, χωρίς να απαιτείται εγκατάσταση ειδικού εξοπλισμού. Παράλληλα, μέσω της εργασίας διερευνώνται οι δυνατότητες που προσφέρει η Επαυξημένη Πραγματικότητα στον σχεδιασμό εφαρμογών, οι οποίες θέτουν ως επίκεντρο τις ανάγκες προσβασιμότητας των χρηστών.

Περίληψη

Στόχος της παρούσας διπλωματικής εργασίας είναι ο σχεδιασμός και η υλοποίηση ενός καινοτόμου συστήματος πλοήγησης εσωτερικών χώρων για άτομα με προβλήματα όρασης, με τη χρήση τεχνολογιών Επαυξημένης Πραγματικότητας. Το σύστημα αναπτύχθηκε ως μία native Android εφαρμογή, αξιοποιώντας την πλατφόρμα ARCore της Google. Μέσω αυτής, επιτυγχάνεται η αναγνώριση και η συνεχής παρακολούθηση του περιβάλλοντος, επιτρέποντας τον ακριβή εντοπισμό της τρέχουσας θέσης του χρήστη. Για τις ανάγκες της πλοήγησης, ο φυσικός χώρος αναπαρίσταται με τη μορφή γραφήματος. Στο γράφημα αυτό ορίζονται δύο τύποι κόμβων: τα Cloud Anchors, τα οποία τοποθετούνται σε καίρια σημεία ενδιαφέροντος (π.χ. διαθέσιμοι προορισμοί όπως αίθουσες και είσοδοι), και τα Waypoints, τα οποία κατανέμονται κατά μήκος των διαδρόμων για τη βέλτιστη χάραξη της πορείας. Ο υπολογισμός της συντομότερης διαδρομής στον χώρο πραγματοποιείται με τη χρήση του αλγορίθμου αναζήτησης A*. Προκειμένου να διασφαλιστεί η μέγιστη δυνατή προσβασιμότητα, η εφαρμογή ενσωματώνει πολυτροπική αλληλεπίδραση. Πιο συγκεκριμένα, συνδυάζει ηχητική καθοδήγηση σε πραγματικό χρόνο, μέσω τεχνολογίας σύνθεσης φωνής (Text-to-Speech), και απτική ανάδραση (haptic feedback), μέσω διακριτών μοτίβων δόνησης. Τέλος, το σύστημα αξιολογήθηκε από 20 χρήστες με τη χρήση του ερωτηματολογίου UEQ-S, με σκοπό την αποτίμηση της ευχρηστίας και της συνολικής εμπειρίας χρήσης που προσφέρει η εφαρμογή.

«Indoor navigation system for people with visual impairments»

«Maria-Theodosia Konstantinidou»

Abstract

The aim of this thesis is the design and implementation of an innovative indoor navigation system for people with visual impairments, using Augmented Reality. The system is a native Android application, utilizing Google's ARCore platform. Through this platform, the environment can be recognized and continuously tracked, enabling accurate determination of the user's current position. For navigation purposes, the physical environment is modeled topologically in the form of a graph. Two types of nodes are defined within this graph: Cloud Anchors, which are placed at key points of interest (e.g., available destinations such as rooms and entrances), and Waypoints, which are distributed along corridors to enable optimal route planning. The shortest path within the environment is calculated using the A* search algorithm. To ensure the highest possible level of accessibility, the application incorporates multimodal interaction. More specifically, it combines real-time audio guidance through Text-to-Speech technology with haptic feedback provided through distinct vibration patterns. Finally, the system was evaluated by 20 users using the UEQ-S questionnaire, with the aim of assessing the usability and overall user experience provided by the application.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου, κ. Κοκκώνη για την καθοδήγηση και τις συμβουλές του κατά τη διάρκεια εκπόνησης της παρούσας διπλωματικής εργασίας. Επίσης, θα ήθελα να εκφράσω την ευγνωμοσύνη μου στην οικογένεια και τους φίλους μου για τη διαρκή στήριξη και ενθάρρυνση που μου πρόσφεραν καθ' όλη τη διάρκεια των σπουδών μου.

Περιεχόμενα

Πρόλογος.....	iv
Περίληψη.....	v
Abstract	vi
Ευχαριστίες	vii
Περιεχόμενα	viii
Κατάλογος Σχημάτων	xii
Κατάλογος Πινάκων.....	xiv
Συντομογραφίες.....	xv
Κεφάλαιο 1ο: Εισαγωγή	1
1.1 Εισαγωγή και κίνητρο	1
1.2 Στόχοι εργασίας.....	2
1.3 Δομή εργασίας.....	2
Κεφάλαιο 2ο: Βιβλιογραφική Ανασκόπηση	3
2.1 Εισαγωγή.....	3
2.2 Βασικές έννοιες εσωτερικού εντοπισμού και πλοήγησης	3
2.2.1 Βασικές έννοιες	3
2.2.2 Συστήματα εντοπισμού θέσης σε εσωτερικούς χώρους	4
2.3 Συστήματα που βασίζονται σε υποδομές	4
2.3.1 Wi-Fi.....	4
2.3.2 BLE.....	5
2.3.3 Άλλες τεχνολογίες (RFID, UWB)	6
2.3.4 Σύστημα AR με οπτικούς δείκτες.....	7
2.4 Συστήματα χωρίς προσθήκη υποδομών	8
2.4.1 Μηχανική Όραση (Computer Vision)	8
2.4.2 Οπτική-αδρανειακή Οδομετρία (Visual – Inertial Odometry)	8
2.4.3 Pedestrian Dead Reckoning.....	9
2.5 Συστήματα πλοήγησης σε εσωτερικούς χώρους για άτομα με οπτική αναπηρία	9
2.6 Συγκριτική αξιολόγηση και συμπεράσματα.....	10
Κεφάλαιο 3ο: Θεωρητικό και τεχνολογικό υπόβαθρο.....	11
3.1 Εισαγωγή.....	11
3.2 Θεωρητικό υπόβαθρο	11
3.2.1 Επαυξημένη πραγματικότητα.....	11

3.2.2	Αναπαράσταση του χώρου ως γράφος	12
3.2.3	Αλγόριθμος A*	13
3.3	Τεχνολογίες ανάπτυξης εφαρμογής.....	14
3.3.1	Android και Kotlin	14
3.3.2	Jetpack Compose	14
3.3.3	ARCore.....	14
3.3.4	SceneView	17
3.3.5	Cloud Firestore	18
3.4	Τεχνολογίες προσβασιμότητας.....	18
3.4.1	Σύνθεση ομιλίας	18
3.4.2	Αναγνώριση ομιλίας.....	19
3.4.3	Απτική ανάδραση	20
Κεφάλαιο 4ο:	Ανάλυση απαιτήσεων και σχεδιασμός διεπαφής.....	21
4.1	Λειτουργικές και μη λειτουργικές απαιτήσεις	21
4.1.1	Λειτουργικές απαιτήσεις	21
4.1.2	Μη λειτουργικές απαιτήσεις.....	21
4.1.3	Απαιτήσεις προσβασιμότητας	21
4.2	Σενάρια Χρήσης	22
4.2.1	Σενάριο Χρήσης Α: Χαρτογράφηση κτηρίου.....	22
4.2.2	Σενάριο Χρήσης Β: Πλοήγηση σε κτήριο	22
4.3	Σχεδιασμός διεπαφής.....	23
4.4	Στοιχεία προσβασιμότητας.....	24
4.4.1	Σχεδιασμός φωνητικής αλληλεπίδρασης.....	24
4.4.2	Σχεδιασμός μοτίβων δόνησης.....	25
Κεφάλαιο 5ο:	Υλοποίηση συστήματος.....	27
5.1	Εισαγωγή.....	27
5.2	Μοντέλο δεδομένων.....	27
5.2.1	Κλάση FloorPlan	27
5.2.2	Κλάση PathStep.....	28
5.2.3	Κλάση PathConnectionType	29
5.2.4	Κλάση PathConnection	30
5.2.5	Αποθήκευση και ανάκτηση δεδομένων μέσω Cloud Firestore	31
5.3	Υλοποίηση χαρτογράφησης	34
5.3.1	Αρχικοποίηση συνεδρίας AR	34
5.3.2	Ανίχνευση και σταθεροποίηση οριζόντιας επιφάνειας.....	35

5.3.3	Εκτέλεση hit-test – δημιουργία προσωρινής τοπικής άγκυρας	36
5.3.4	Σάρωση του χώρου γύρω από την άγκυρα	37
5.3.5	ARCore API και εξουσιοδότηση.....	38
5.3.6	Φιλοξενία Cloud Anchor και δημιουργία κόμβου.....	38
5.3.7	Προσθήκη ενδιάμεσων σημείων πλοήγησης (Waypoints)	39
5.3.8	Δημιουργία διακλαδώσεων.....	40
5.3.9	Αλλαγή ορόφου	41
5.3.10	Έλεγχος εγκυρότητας γράφου.....	42
5.3.11	Αποθήκευση χαρτογραφημένου κτηρίου (FloorPlan).....	43
5.4	Υλοποίηση Πλοήγησης	43
5.4.1	Προσβάσιμη αλληλεπίδραση και διαχείριση διαλόγου	44
5.4.2	Ανάκτηση χάρτη και κατασκευή γράφου	52
5.4.3	Αρχικοποίηση συνεδρίας και επίλυση Cloud Anchors	54
5.4.4	Μετασχηματισμός συντεταγμένων και εντοπισμός χρήστη	56
5.4.5	Υπολογισμός βέλτιστης διαδρομής	57
5.4.6	Παρακολούθηση πορείας και υπολογισμός οδηγιών.....	58
5.4.7	Επανάληψη οδηγιών.....	61
5.4.8	Ανίχνευση απόκλισης από την πορεία και αποκατάσταση	61
5.4.9	Ολοκλήρωση πλοήγησης.....	63
5.5	Παρουσίαση διεπαφής και βασικών ροών αλληλεπίδρασης.....	63
5.5.1	Κοινή αρχική οθόνη (LandingScreen).....	63
5.5.2	Διεπαφή διαχειριστή – Ροή χαρτογράφησης	64
5.5.3	Διεπαφή τελικού χρήστη - Ροή πλοήγησης	67
Κεφάλαιο 6ο:	Αξιολόγηση εφαρμογής.....	74
6.1	Στόχοι αξιολόγησης.....	74
6.2	Μεθοδολογία	74
6.3	Αποτελέσματα	75
6.3.1	Δημογραφικά χαρακτηριστικά	76
6.3.2	Τεχνολογική εξοικείωση και προσβασιμότητα	77
6.3.3	Αποτελέσματα ερωτηματολογίου UEQ-S	79
6.3.4	Συνοπτική στατιστική ανάλυση UEQ-S.....	84
6.3.5	Σύγκριση με το UEQ Benchmark.....	85
Κεφάλαιο 7ο:	Συμπεράσματα και μελλοντικές επεκτάσεις	86
7.1	Συμπεράσματα.....	86
7.2	Περιορισμοί συστήματος	86

7.3	Προτάσεις Βελτίωσης - Μελλοντικές επεκτάσεις.....	87
ΒΙΒΛΙΟΓΡΑΦΙΑ.....		88

Κατάλογος Σχημάτων

Σχήμα 2.1: Παράδειγμα Wi-Fi Fingerprinting [14].....	5
Σχήμα 3.1: Διαφορετικοί τύποι AR [31]	12
Σχήμα 3.2: Λειτουργία επιταχυνσιόμετρου [37]	15
Σχήμα 3.3: Λειτουργία γυροσκοπίου [37].....	15
Σχήμα 3.4: Παράδειγμα hit-test [36]	16
Σχήμα 5.1: Ορισμός της κλάσης FloorPlan.....	28
Σχήμα 5.2: Ορισμός της κλάσης PathStep	29
Σχήμα 5.3: Ορισμός της κλάσης PathConnectionType.....	30
Σχήμα 5.4: Ορισμός της κλάσης PathConnection	31
Σχήμα 5.5: Μέθοδος saveFloorPlan()	31
Σχήμα 5.6: Μέθοδος loadFloorPlan().....	32
Σχήμα 5.7: Μέθοδος loadAllFloorPlans()	32
Σχήμα 5.8: Μέθοδος updateFloorPlan()	33
Σχήμα 5.9: Μέθοδος deleteFloorPlan()	33
Σχήμα 5.10: Αρχικοποίηση συνεδρίας μέσω ARSceneView	35
Σχήμα 5.11: Συνάρτηση isPlaneMature().....	36
Σχήμα 5.12: Απόσπασμα της onSingleTapUp(), όπου ορίζεται η εκτέλεση του hit-test	36
Σχήμα 5.13: Δημιουργία άγκυρας μέσω της createAnchor()	37
Σχήμα 5.14: Συνάρτηση refinePendingAnchorPose().....	38
Σχήμα 5.15: Ορισμός διάρκειας ζωής ενός Cloud Anchor	38
Σχήμα 5.16: Έλεγχοι αποδοχής και ορισμός υπάρχοντος κόμβου ως σημείο έναρξης της νέας διακλάδωσης	41
Σχήμα 5.17: Δημιουργία αντικειμένου pendingStairMapping	42
Σχήμα 5.18: Ενδεικτικό απόσπασμα της συνάρτησης graphValidationError(), για τους ελέγχους εγκυρότητας του γράφου.....	43
Σχήμα 5.19: Αποθήκευση του χαρτογραφημένου κτηρίου στη συλλογή floorPlans, μέσω της συνάρτησης saveFloorPlan()	43
Σχήμα 5.20: Αρχικοποίηση TextToSpeech()	45
Σχήμα 5.21: Παρουσίαση recognizerIntent.....	45
Σχήμα 5.22: Συνάρτηση onResults()	46
Σχήμα 5.23: Συνάρτηση destroy()	47
Σχήμα 5.24: Ορισμός της enum κλάσης VoicePhase.....	47
Σχήμα 5.25: Απόσπασμα συνάρτησης speakThenListen()	48
Σχήμα 5.26: Συνάρτηση speakNavigationInstruction().....	49
Σχήμα 5.27: Συνάρτηση vibrateForFloorTransition()	50
Σχήμα 5.28: Πραγματοποίηση δόνησης στη συσκευή	50
Σχήμα 5.29: Συνάρτηση startBuildingLoop()	51
Σχήμα 5.30: Συνάρτηση promptForBuilding().....	51
Σχήμα 5.31: Συνάρτηση loadFloorPlan()	52
Σχήμα 5.32: Παρουσίαση κλάσεων GraphNode, GraphEdge, IndoorGraph	53
Σχήμα 5.33: Συνάρτηση computeMappedPositions()	53
Σχήμα 5.34: Συνάρτηση buildNavigationGraph().....	54
Σχήμα 5.35: Οι λίστες που περιγράφηκαν παραπάνω	55
Σχήμα 5.36: Συναρτήσεις mappedAnchorPose() και cameraPoseInMap().....	56

Σχήμα 5.37: Συνάρτηση squaredDistance()	57
Σχήμα 5.38: Συνάρτηση reconstructPath()	58
Σχήμα 5.39: Απόσπασμα επιστροφής της συνάρτησης determineTurnAction()	59
Σχήμα 5.40: Συνάρτηση findNextDecisionIndex()	59
Σχήμα 5.41: Απόσπασμα περιπτώσεων οδηγιών που παράγονται από την generateInstruction()	60
Σχήμα 5.42: Απόσπασμα ορισμού cueKey	60
Σχήμα 5.43: Απόσπασμα δυνατότητας επανάληψης οδηγίας μέσω χειρονομίας	61
Σχήμα 5.44: Συνάρτηση distancePointToSegmentXZ()	62
Σχήμα 5.45: Παρουσίαση αρχικής οθόνης (LandingScreen) στο light theme (αριστερά) και στο dark theme (δεξιά)	64
Σχήμα 5.46: Οθόνες AdminLoginScreen, AdminDashboardScreen, AddBuildingScreen	65
Σχήμα 5.47: Ανίχνευση επιπέδου και παρουσίαση λειτουργιών στην οθόνη χαρτογράφησης κτηρίου (ARMappingScreen)	66
Σχήμα 5.48: Τοποθέτηση Cloud Anchors και Waypoints στο χώρο	67
Σχήμα 5.49: Οθόνη επιλογής κτηρίου (αριστερά) και προορισμού (δεξιά) για πλοήγηση (DestinationPickerScreen)	68
Σχήμα 5.50: Προσπάθεια εύρεσης της τρέχουσας θέσης του χρήστη, επιλύοντας τα Cloud Anchors	69
Σχήμα 5.51: Κατάσταση πλοήγησης α) κανονική πλοήγηση, όπου οι οδηγίες φαίνονται στην οθόνη	70
Σχήμα 5.52: Προειδοποίηση επερχόμενης σκάλας (αριστερά), ειδοποίηση για μετάβαση μέσω σκάλας άμεσα (δεξιά)	71
Σχήμα 5.53: Κατάσταση πλοήγησης β) Άφιξη στον προορισμό	72
Σχήμα 5.54: : Καταστάσεις πλοήγησης γ) Ενημέρωση για απόκλιση από την πορεία (αριστερά) και δ) Απώλεια παρακολούθησης (δεξιά)	73
Σχήμα 6.1: Διάγραμμα αποτελεσμάτων στην ερώτηση "Σε ποια ηλικιακή ομάδα ανήκετε;"	76
Σχήμα 6.2: Διάγραμμα απαντήσεων ερώτησης προσδιορισμού του φύλου των συμμετεχόντων	76
Σχήμα 6.3: Διάγραμμα απαντήσεων σχετικά με το επίπεδο σπουδών των συμμετεχόντων	77
Σχήμα 6.4: Διάγραμμα απαντήσεων στην ερώτηση: "Έχετε χρησιμοποιήσει στο παρελθόν παρόμοιες εφαρμογές σε κινητό ή tablet; (εφαρμογές εσωτερικής πλοήγησης);"	78
Σχήμα 6.5: Διάγραμμα απαντήσεων στην ερώτηση: "Πόσο κατανοητές βρήκατε τις φωνητικές οδηγίες πλοήγησης της εφαρμογής (Text-to-Speech) κατά τη διάρκεια της πορείας σας;"	78
Σχήμα 6.6: Διάγραμμα απαντήσεων στην ερώτηση "Πόσο εύκολος σας φάνηκε ο φωνητικός χειρισμός της εφαρμογής (επιλογή κτηρίου/προορισμού μέσω ομιλίας) σε σύγκριση με τα οπτικά στοιχεία της οθόνης σε σύγκριση με τα οπτικά στοιχεία της οθόνης;"	79
Σχήμα 6.7: Αποτελέσματα αξιολόγησης εφαρμογής ως προς το ζεύγος χαρακτηρισμών "Βαρετή - Συναρπαστική"	80
Σχήμα 6.8: Αποτελέσματα αξιολόγησης εφαρμογής ως προς το ζεύγος χαρακτηρισμών "Αδιάφορη - Ενδιαφέρουσα"	80
Σχήμα 6.9: : Αποτελέσματα αξιολόγησης εφαρμογής ως προς το ζεύγος χαρακτηρισμών "Συμβατική - Εφευρετική"	81
Σχήμα 6.10: Αποτελέσματα αξιολόγησης εφαρμογής ως προς το ζεύγος χαρακτηρισμών "Παρεμποδιστική - Υποστηρικτική"	81
Σχήμα 6.11: Αποτελέσματα αξιολόγησης εφαρμογής ως προς το ζεύγος χαρακτηρισμών "Περίπλοκη - Εύκολη"	82
Σχήμα 6.12: Αποτελέσματα αξιολόγησης εφαρμογής ως προς το ζεύγος χαρακτηρισμών "Συνηθισμένη - Πρωτοπόρα"	82

Σχήμα 6.13: Αποτελέσματα αξιολόγησης εφαρμογής ως προς το ζεύγος χαρακτηρισμών "Ανεπαρκής - Επαρκής"	83
Σχήμα 6.14: Αποτελέσματα αξιολόγησης εφαρμογής ως προς το ζεύγος χαρακτηρισμών "Σαφής - Μπερδεμένη"	83
Σχήμα 6.15: Διάγραμμα σύγκρισης αποτελεσμάτων UEQ με το UEQ Benchmark	85

Κατάλογος Πινάκων

Πίνακας 4.1: Περιγραφή των μοτίβων δόνησης που χρησιμοποιούνται στην εφαρμογή.	25
Πίνακας 6.1: Παρουσίαση μέσης τιμής και τυπικής απόκλισης για κάθε ζεύγος χαρακτηρισμών.....	84

Συντομογραφίες

Δ.Ε. Διπλωματική Εργασία

ΔΙΠΙΑΕ Διεθνές Πανεπιστήμιο της Ελλάδος

GNSS Global Navigation Satellite System (Παγκόσμιο Δορυφορικό Σύστημα Πλοήγησης)

GPS Global Positioning System (Παγκόσμιο Σύστημα Στιγματοθέτησης)

BLE Bluetooth Low Energy

AR Augmented Reality (Επαυξημένη Πραγματικότητα)

ΠΟΥ Παγκόσμιος Οργανισμός Υγείας

UWB Ultra-Wideband

IMU Inertial Measurement Unit (Αδρανειακή Μονάδα Μέτρησης)

IPS Indoor Positioning System (Σύστημα Εντοπισμού Θέσης σε Εσωτερικούς Χώρους)

AP Access Point (Σημείο Πρόσβασης)

RSSI Received Signal Strength Indicator

k-NN k-Nearest Neighbors

MQTT Message Queuing Telemetry Transport

RFID Radio Frequency Identification

ToA Time of Arrival

TDoA Time Difference of Arrival

VIO Visual-Inertial Odometry

PDR Pedestrian Dead Reckoning

VR Virtual Reality (Εικονική Πραγματικότητα)

SLAM Simultaneous Localization and Mapping

API Application Programming Interface

UI User Interface (Διεπαφή Χρήστη)

UEQ User Experience Questionnaire

UEQ-S User Experience Questionnaire – Short Version

UX User Experience (Εμπειρία Χρήστη)

TTS Text-to-Speech

Κεφάλαιο 1ο: Εισαγωγή

1.1 Εισαγωγή και κίνητρο

Η πλοήγηση και η ικανότητα προσανατολισμού στο περιβάλλον αποτελούν θεμελιώδεις ανθρώπινες ανάγκες. Η εξέλιξη της τεχνολογίας οδήγησε στην αντικατάσταση των παραδοσιακών μεθόδων πλοήγησης από σύγχρονα συστήματα, με χαρακτηριστικό παράδειγμα το Παγκόσμιο Δορυφορικό Σύστημα Πλοήγησης (Global Navigation Satellite System - GNSS), στο οποίο εντάσσεται το Παγκόσμιο Σύστημα Στιγματοθέτησης (Global Positioning System - GPS) [1]. Το GPS επιτρέπει τον ακριβή προσδιορισμό της θέσης του χρήστη και βρίσκει εφαρμογή σε ευρύ φάσμα συστημάτων πλοήγησης σε εξωτερικούς χώρους. Ωστόσο, όσον αφορά τη χρήση του σε εσωτερικούς χώρους, το GPS δε λειτουργεί αποδοτικά, καθώς υποφέρει από εξασθένηση του σήματος και συνεπώς η ακρίβεια εντοπισμού είναι χαμηλή. [2]

Η αυξανόμενη πολυπλοκότητα των σύγχρονων κτηρίων έχει δημιουργήσει την ανάγκη για αξιόπιστη πλοήγηση και στο εσωτερικό τους. Νοσοκομεία, αεροδρόμια, εμπορικά κέντρα και πανεπιστημιούπολεις αποτελούν σύνθετους χώρους, με πολλαπλούς ορόφους και διαδρόμους, οι οποίοι δυσχεραίνουν τον προσανατολισμό των επισκεπτών [3]. Τα συστήματα GNSS δεν αποτελούν αξιόπιστη λύση για την πλοήγηση στο εσωτερικό κτηρίων, καθώς η παρουσία τοίχων και άλλων δομικών στοιχείων εξασθενεί τη λήψη των δορυφορικών σημάτων. Ο περιορισμός αυτός οδήγησε στην ανάπτυξη συστημάτων εσωτερικής πλοήγησης που αξιοποιούν εναλλακτικές τεχνολογίες, όπως Wi-Fi, Bluetooth Low Energy (BLE), επαυξημένη πραγματικότητα (Augmented Reality - AR).

Η ανάγκη για πλοήγηση σε εσωτερικούς χώρους είναι ιδιαίτερα σημαντική για άτομα με προβλήματα όρασης. Στη διεθνή βιβλιογραφία, η κατηγοριοποίηση αυτών διακρίνεται σε γενικές γραμμές σε δύο κατηγορίες: α) άτομα με πλήρη απώλεια όρασης και β) άτομα με μερική απώλεια όρασης [4]. Τα άτομα με πλήρη απώλεια όρασης αδυνατούν να αντιληφθούν οποιαδήποτε μορφή φωτός και οπτικών ερεθισμάτων και η αλληλεπίδρασή τους με το περιβάλλον βασίζεται σε μη οπτικές μεθόδους, όπως το λευκό μπαστούνι. Αντίθετα, τα άτομα με μερική απώλεια όρασης διατηρούν ένα μέρος της όρασης τους και πολλές φορές μπορούν να διακρίνουν σχήματα, χρώματα ή βάθος πεδίου σε ένα βαθμό.

Σύμφωνα με τον Παγκόσμιο Οργανισμό Υγείας (ΠΟΥ) τουλάχιστον 2,2 δισεκατομμύρια άνθρωποι παρουσιάζουν κάποια μορφή οπτικής αναπηρίας. Ανάμεσα τους, εκατομμύρια από αυτούς ζουν με σοβαρή ή πλήρη οπτική απώλεια. Η μερική ή πλήρης απώλεια όρασης επιφέρει σημαντικές προκλήσεις στην ανεξαρτησία κατά την καθημερινότητα τους, όπως στην εκπαίδευση, εργασία και ανεξάρτητη μετακίνηση τους. Αυτό συμβαίνει καθώς η όραση αποτελεί το βασικό μέσο αναγνώρισης σημείων αναφοράς, εμποδίων και αλλαγών στην κατεύθυνση [5].

Τα παραδοσιακά βοηθήματα δεν ικανοποιούν πλήρως τις ανάγκες πλοήγησης τους σε άγνωστα περιβάλλοντα. Βοηθήματα όπως το λευκό μπαστούνι ή οι σκύλοι οδηγοί, μολονότι συμβάλλουν στην ανίχνευση εμποδίων σε επίπεδο εδάφους, αδυνατούν να ανιχνεύσουν εμπόδια σε επίπεδο σώματος ή κεφαλιού [2]. Επιπλέον, δεν προσφέρουν πληροφορίες για την θέση του ατόμου στον χώρο, ενώ το άτομο δεν έχει επαρκείς οπτικές ενδείξεις για την πλοήγηση του. Έτσι, ο προσανατολισμός και ο εντοπισμός διαθέσιμης και κατάλληλης διαδρομής προς τον προορισμό, καθίσταται ιδιαίτερα δύσκολος και τις περισσότερες φορές η πλοήγηση του βασίζεται σε βλέποντες συνοδούς [2], [6].

Οι παραπάνω περιορισμοί των υφιστάμενων λύσεων αναδεικνύουν την ανάγκη ανάπτυξης προσβάσιμων συστημάτων εσωτερικής πλοήγησης. Τα συστήματα αυτά οφείλουν να παρέχουν στα

άτομα με προβλήματα όρασης επαρκή χωρική πληροφορία, σχετικά με τη θέση και κατεύθυνση τους και ακριβείς οδηγίες για την ασφαλή πλοήγηση τους στον επιλεγμένο προορισμό. Η μετάδοση της πληροφορίας είναι σημαντικό να πραγματοποιείται μέσω προσβάσιμων τρόπων αλληλεπίδρασης, όπως η φωνητική καθοδήγηση και η απτική ανάδραση [7].

1.2 Στόχοι εργασίας

Σκοπός της παρούσας εργασίας είναι η σχεδίαση, ανάπτυξη και αξιολόγηση μίας καινοτόμας εφαρμογής εσωτερικής πλοήγησης κτηρίων για άτομα με οπτική αναπηρία. Ειδικότερα, το σύστημα υλοποιείται ως native Android εφαρμογή, αξιοποιώντας τις δυνατότητες της επαυξημένης πραγματικότητας μέσω του περιβάλλοντος ARCore της Google. Βασικό στόχο αποτελεί η ανεξάρτητη πλοήγηση των χρηστών εντός κτηρίων, χωρίς την ανάγκη εγκατάστασης εξοπλισμού στον χώρο. Πιο συγκεκριμένα, οι στόχοι της εργασίας είναι οι ακόλουθοι:

- Χαρτογράφηση εσωτερικών χώρων μέσω Cloud Anchors για την αναπαράσταση των διαθέσιμων προορισμών και Waypoints για ενδιάμεσα σημεία της διαδρομής.
- Αναπαράσταση του χώρου ως γράφου, όπου τα σημεία αποτελούν κόμβους και οι μεταξύ τους συνδέσεις ακμές.
- Αποθήκευση κτηρίων, προορισμών και διαδρομών.
- Εντοπισμός του χρήστη στο χαρτογραφημένο κτήριο.
- Υπολογισμός διαδρομής προς τον επιλεγμένο προορισμό.
- Υποστήριξη φωνητικής αλληλεπίδρασης για τον χειρισμό των βασικών λειτουργιών της εφαρμογής.
- Παροχή φωνητικών οδηγιών πλοήγησης και συμπληρωματική ενημέρωση μέσω απτικής ανάδρασης.
- Σχεδίαση προσβάσιμης διεπαφής για άτομα με προβλήματα όρασης, η οποία υποστηρίζει τον φωνητικό χειρισμό των βασικών λειτουργιών της εφαρμογής.

1.3 Δομή εργασίας

Η παρούσα εργασία οργανώνεται σε επτά κεφάλαια. Στο πρώτο κεφάλαιο παρουσιάζονται το κίνητρο, και οι στόχοι της εργασίας. Στο δεύτερο κεφάλαιο παρουσιάζεται βιβλιογραφική ανασκόπηση υφιστάμενων λύσεων στο πρόβλημα της εσωτερικής πλοήγησης. Αρχικά, στην ενότητα 2.1 παρουσιάζονται βασικές έννοιες σχετικά με τον εσωτερικό εντοπισμό και την πλοήγηση. Στις ενότητες 2.2 και 2.3 αξιολογούνται τόσο συστήματα που βασίζονται στην εγκατάσταση επιπλέον υποδομών, όπως λύσεις με χρήση Wi-Fi, BLE, όσο και λύσεις χωρίς την ανάγκη υποδομών, όπως συστήματα που αξιοποιούν την επαυξημένη πραγματικότητα. Στο τρίτο κεφάλαιο παρουσιάζεται το θεωρητικό και τεχνολογικό υπόβαθρο της εργασίας. Ειδικότερα, αναλύονται έννοιες, όπως η επαυξημένη πραγματικότητα, το ARCore, τα Cloud Anchors, η αναπαράσταση χώρων μέσω γράφου, ο αλγόριθμος A* για την επιλογή της βέλτιστης διαδρομής καθώς και τεχνολογίες προσβασιμότητας, συγκεκριμένα φωνητικής και απτικής ανάδρασης. Στο τέταρτο κεφάλαιο, αναλύονται οι λειτουργικές και μη λειτουργικές απαιτήσεις του συστήματος, παρουσιάζεται ο σχεδιασμός της διεπαφής και της προσβασιμότητας, όπως και τα σενάρια χρήσης της εφαρμογής. Στο πέμπτο κεφάλαιο περιγράφεται η τεχνική υλοποίηση της εφαρμογής. Αναλύονται τα μοντέλα δεδομένων, οι λειτουργίες χαρτογράφησης και πλοήγησης, η αποθήκευση δεδομένων και οι μηχανισμοί προσβάσιμης αλληλεπίδρασης. Επιπλέον, παρουσιάζεται η διεπαφή της εφαρμογής και η ροή μεταξύ των οθονών. Στο έκτο κεφάλαιο, μετά από αξιολόγηση του συστήματος, παρουσιάζονται τα αποτελέσματα των δοκιμών, μέσω του ερωτηματολογίου UEQ-S. Τέλος, στο έβδομο κεφάλαιο συνοψίζονται τα βασικά συμπεράσματα της εργασίας, καταγράφονται οι περιορισμοί της εφαρμογής και παρέχονται προτάσεις για πιθανές μελλοντικές βελτιώσεις και επεκτάσεις.

Κεφάλαιο 2ο: Βιβλιογραφική Ανασκόπηση

2.1 Εισαγωγή

Όπως αναφέρθηκε στο εισαγωγικό κεφάλαιο, οι περιορισμοί των συστημάτων GNSS και η αδυναμία αξιόπιστης λειτουργίας τους σε εσωτερικά περιβάλλοντα, οδήγησαν στην ανάπτυξη συστημάτων εσωτερικού εντοπισμού και πλοήγησης. Τα συστήματα αυτά, αξιοποιούν εναλλακτικές τεχνολογίες, όπως Wi-Fi, Bluetooth Low Energy (BLE), Ultra-Wideband (UWB), δεδομένα από την αδρανειακή μονάδα μέτρησης (IMU) και τεχνικές μηχανικής όρασης και επαυξημένης πραγματικότητας (AR). Το κεφάλαιο αυτό, εξετάζει τις προσεγγίσεις τέτοιων συστημάτων, με στόχο την κατανόηση των υπάρχουσών προσεγγίσεων, των πλεονεκτημάτων και των περιορισμών τους.

Συγκεκριμένα, στο 2.2 παρουσιάζονται βασικές έννοιες που αφορούν τον εσωτερικό εντοπισμό και πλοήγηση, όπως και την κατηγοριοποίηση των συστημάτων αυτών. Στα 2.3 και 2.4 παρουσιάζονται συγκεκριμένες προσεγγίσεις με αντιπροσωπευτικά συστήματα, διακρίνοντας τα με βάση το αν απαιτούν εγκατάσταση υποδομής ή όχι. Η Ενότητα 2.5 εστιάζει σε λύσεις που απευθύνονται σε άτομα με οπτική αναπηρία, καθώς αποτελούν την κύρια ομάδα χρηστών της παρούσας εργασίας. Τέλος, στο 2.6 παρουσιάζεται συγκριτική αξιολόγηση των συστημάτων που παρουσιάστηκαν, συνοψίζοντας τα βασικά πλεονεκτήματα και περιορισμούς τους.

2.2 Βασικές έννοιες εσωτερικού εντοπισμού και πλοήγησης

2.2.1 Βασικές έννοιες

Τα συστήματα εσωτερικού εντοπισμού και πλοήγησης συνδέονται με τρεις βασικές έννοιες: α) προσδιορισμός θέσης (positioning), β) εντοπισμός (localization), γ) πλοήγηση (navigation) [8]. Στη βιβλιογραφία, πολλές φορές οι όροι positioning και localization χρησιμοποιούνται ως συνώνυμοι, ωστόσο αξίζει να παρουσιασθεί η διάκριση τους.

Σύμφωνα με τους ορισμούς των [8],[9] ο προσδιορισμός θέσης απαντά στην ερώτηση “Σε ποιες συντεταγμένες βρίσκομαι;” και αφορά τη διαδικασία υπολογισμού των απόλυτων ή σχετικών συντεταγμένων προς ένα αντικείμενο ή άνθρωπο μέσα σε ένα καθορισμένο σύστημα αναφοράς, όπως οι συντεταγμένες (x,y,z) στο Καρτεσιανό Σύστημα.

Ο εντοπισμός απαντά στην ερώτηση “Πού βρίσκομαι σε σχέση με το περιβάλλον γύρω μου;” και αφορά την κατανόηση της θέσης του χρήστη εντός ενός περιβάλλοντος. Δίνει έμφαση στη σύνδεση της θέσης του χρήστη με τον χώρο στον οποίο βρίσκεται, όπως για παράδειγμα ότι ο χρήστης βρίσκεται στον πρώτο όροφο, κοντά στην αίθουσα B1.

Τέλος, η πλοήγηση ορίζεται ως ο σχεδιασμός της διαδρομής και η καθοδήγηση του χρήστη προς τον προορισμό του. Απαντά στην ερώτηση “Πώς θα κατευθυνθώ από την τρέχουσα τοποθεσία μου στον προορισμό που επιθυμώ;”. Ανάλογα με το περιβάλλον λειτουργίας τους, τα συστήματα εντοπισμού θέσης και πλοήγησης κατηγοριοποιούνται σε συστήματα εξωτερικού και εσωτερικού χώρου. [8]

Ο εντοπισμός της θέσης του χρήστη στον χώρο, αποτελεί το πρώτο βήμα για ένα ολοκληρωμένο σύστημα πλοήγησης. Στη συνέχεια, πρέπει να πραγματοποιηθεί η αναπαράσταση του εσωτερικού χώρου, ο υπολογισμός της κατάλληλης διαδρομής και η καθοδήγηση του χρήστη [8].

2.2.2 Συστήματα εντοπισμού θέσης σε εσωτερικούς χώρους

Ένα σύστημα εντοπισμού θέσης σε εσωτερικούς χώρους (Indoor Positioning System - IPS) ορίζεται ως ένα σύστημα το οποίο συμβάλλει στον προσδιορισμό της θέσης αντικειμένων ή ανθρώπων στο εσωτερικό κτηρίων [10]. Στα περισσότερα IPS υπάρχουν δύο βασικά είδη κόμβων: οι σταθεροί κόμβοι αναφοράς (Reference Nodes – RNs) ή διαφορετικά άγκυρες, με γνωστές θέσεις και οι κινητοί κόμβοι (Mobile Nodes – MNs), των οποίων η θέση πρέπει να προσδιορισθεί.

Ανάλογα με τον τρόπο συλλογής και επεξεργασίας των δεδομένων, τα IPS κατηγοριοποιούνται σε δύο βασικές τοπολογίες: self-positioning, remote-positioning. Σε ένα self-positioning σύστημα η θέση του κόμβου, δηλαδή της συσκευής, υπολογίζεται από τον ίδιο τον κόμβο, καθώς λαμβάνει δεδομένα από τους κόμβους αναφοράς. Παρ'όλο που διασφαλίζεται ιδιωτικότητα, ο κόμβος επιβαρύνεται υπολογιστικά και η κατανάλωση ενέργειας του είναι υψηλή. Αντίθετα, σε ένα remote-positioning σύστημα, όπου η συσκευή λειτουργεί ως πομπός, ο υπολογισμός θέσης πραγματοποιείται από ένα εξωτερικό σύστημα, όπως ένας διακομιστής (server), μεταφέροντας το υπολογιστικό κόστος στην υποδομή αυτή [10].

Τέλος, όσον αφορά την ανάγκη εγκατάστασης υποδομής (infrastructure), τα IPS χωρίζονται σε συστήματα που βασίζονται σε υποδομή (infrastructure-based) και σε συστήματα χωρίς υποδομή (infrastructure-less). [11]. Στην πρώτη κατηγορία ανήκουν συστήματα που χρησιμοποιούν, για παράδειγμα BLE beacons, RFID tags, οπτικούς δείκτες. Στην δεύτερη κατηγορία ανήκουν προσεγγίσεις που αξιοποιούν για παράδειγμα την κάμερα της συσκευής, χωρίς να απαιτούν την εγκατάσταση ειδικού εξοπλισμού στον χώρο.

2.3 Συστήματα που βασίζονται σε υποδομές

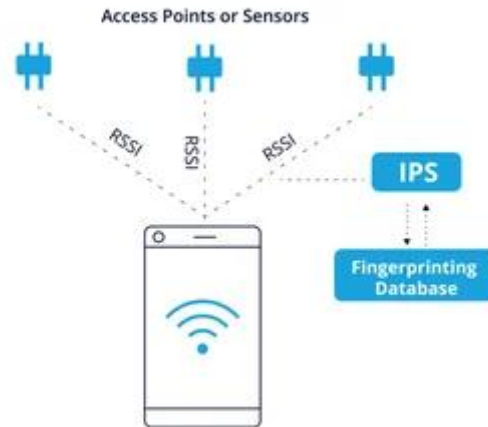
2.3.1 Wi-Fi

Το Wi-Fi αποτελεί μία ευρέως χρησιμοποιημένη τεχνολογία [9]. Βασίζεται στο πρότυπο IEEE 802.11 και λειτουργεί στις συχνότητες 2.4 και 5 GHz. Για την υλοποίηση ενός συστήματος εσωτερικού εντοπισμού με Wi-Fi, αξιοποιούνται τα σημεία πρόσβασης (Access Points – AP), τα οποία είναι ήδη διαθέσιμα στα περισσότερα σύγχρονα κτήρια. Με αυτό τον τρόπο, πολλές φορές περιορίζεται η ανάγκη εγκατάστασης πρόσθετου εξοπλισμού, χωρίς να εξαλείφεται η εξάρτηση του συστήματος από αυτόν.

Μία από τις συχνότερες μετρήσεις που χρησιμοποιούνται για τον εντοπισμό θέσης μέσω Wi-Fi είναι το Received Signal Strength Indicator (RSSI), το οποίο εκφράζει την ισχύ του λαμβανόμενου σήματος. Οι τιμές RSSI χρησιμοποιούνται ως ένδειξη εγγύτητας της συσκευής προς τα σημεία πρόσβασης. Όσο μεγαλύτερη είναι η ένδειξη RSSI, τόσο πιο κοντά βρίσκεται η συσκευή στο σημείο πρόσβασης [12], [13]

Όσον αφορά τις μεθόδους εντοπισμού θέσης, μία από τις συχνότερες που χρησιμοποιείται μέσω του Wi-Fi είναι το fingerprinting [10]. Το fingerprinting διαθέτει δύο διακριτές φάσεις. Στην offline φάση συλλέγονται μετρήσεις σημάτων από πολλαπλά σημεία αναφοράς, δηλαδή με γνωστή θέση. Το σύνολο κάθε σημείου σχηματίζει ένα fingerprint. Τα fingerprints αποθηκεύονται σε βάση δεδομένων, δημιουργώντας τον χάρτη χώρου. Κατά την online φάση, συλλέγονται νέες τιμές από τη θέση της συσκευής. Αυτές, συγκρίνονται με τις αποθηκευμένες από την offline φάση τιμές, με χρήση αλγορίθμων όπως ο αλγόριθμος k-Nearest Neighbors (k-NN), καθώς και τεχνικές μηχανικής ή βαθιάς μάθησης. Με την σύγκριση, προσδιορίζεται η πιθανότερη θέση της συσκευής [10]. Σε συστήματα βασισμένα στο Wi-Fi, τα fingerprints συνήθως δημιουργούνται από μετρήσεις του RSSI.

Το fingerprinting μπορεί να προσφέρει αξιόπιστα και ακριβή αποτελέσματα. Ωστόσο, αποτελεί μία χρονοβόρα μέθοδο, η οποία απαιτεί πολλή επεξεργαστική ισχύ και παρουσιάζει ευαισθησία σε αλλαγές του περιβάλλοντος, όπως μεταβολές στη διαρρύθμιση του κτηρίου, όπου η διαδικασία της χαρτογράφησης θα πρέπει να επαναληφθεί. [9], [10].



Σχήμα 2.1: Παράδειγμα Wi-Fi Fingerprinting [14]

Στα βασικά πλεονεκτήματα του Wi-Fi, περιλαμβάνονται η ευρεία κάλυψη, η ευρεία υποστήριξή του από κινητές συσκευές και η δυνατότητα αξιοποίησης της προ υπάρχουσας υποδομής. Επομένως, το κόστος είναι χαμηλό, όταν υπάρχει ήδη κατάλληλος αριθμός AP. Ωστόσο, η ακρίβεια των σχετικών συστημάτων θεωρείται χαμηλή έως μέτρια, καθώς η ισχύς του σήματος επηρεάζεται σημαντικά από εμπόδια, όπως τοίχοι, άνθρωποι. Επιπλέον, η ενεργειακή κατανάλωση του συστήματος είναι υψηλή.

Χαρακτηριστικό παράδειγμα ολοκληρωμένου συστήματος που βασίζεται στο Wi-Fi, αποτελεί το σύστημα το [15]. Το συγκεκριμένο σύστημα βασίζεται στην ένδειξη ισχύος λαμβανόμενου σήματος (RSSI) του υπάρχοντος δικτύου Wi-Fi. Η εφαρμογή που προτείνεται, σαρώνει τα διαθέσιμα δίκτυα και στέλνει τις τιμές RSSI και τα αναγνωριστικά των APs στο Raspberry Pi, μέσω του πρωτοκόλλου MQTT. Αφού ανακτηθεί ο χάρτης αποτυπωμάτων, εφαρμόζεται ο αλγόριθμος k-NN και συγκρίνει τις τρέχουσες μετρήσεις RSSI με τον αποθηκευμένο χάρτη αποτυπωμάτων. Τέλος, για τον υπολογισμό διαδρομής, εφαρμόζεται ο αλγόριθμος Dijkstra. Η ακρίβεια που καταγράφηκε ήταν 78.57%, με σφάλματα απόστασης μικρότερα των 3 μέτρων. Μετά από δοκιμές με διαφορετικό αριθμό δρομολογητών και διαφορετικές παραμέτρους δειγματοληψίας, προέκυψε πως η απόδοση του συστήματος εξαρτάται από τον αριθμό και τη διάταξη των AP.

2.3.2 BLE

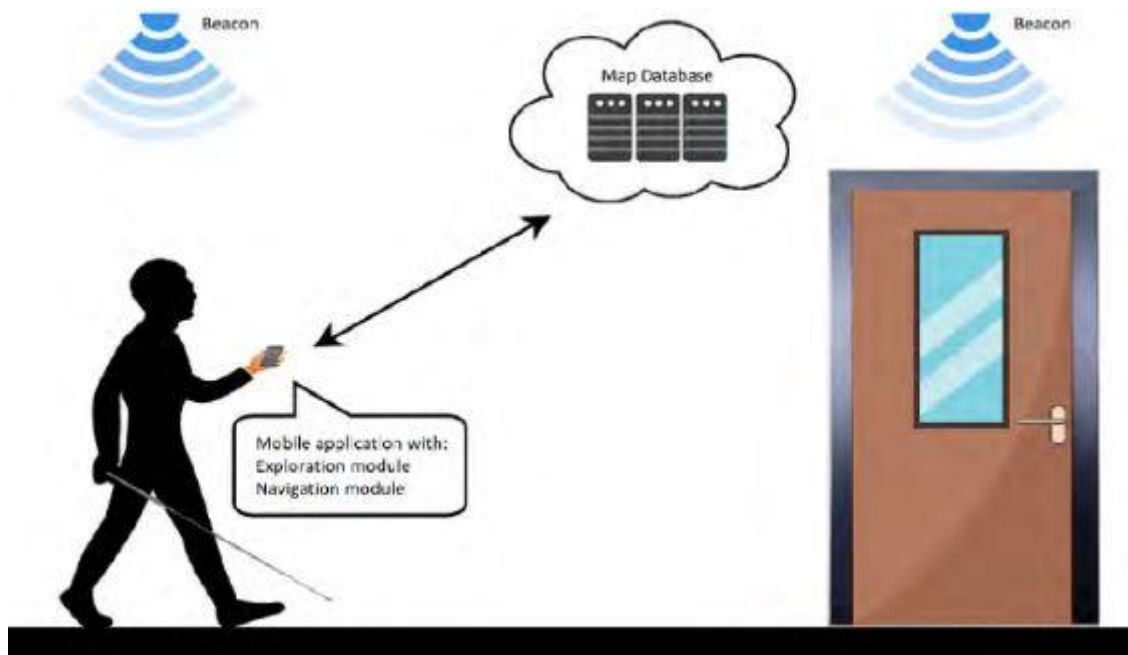
Το BLE αποτελεί ασύρματη τεχνολογία επικοινωνίας μικρής εμβέλειας, σχεδιασμένη με γνώμονα την χαμηλή κατανάλωση ενέργειας. Λειτουργεί στα 2.4 Ghz. [9] και υποστηρίζεται από τις περισσότερες κινητές συσκευές σήμερα. Σε συστήματα εσωτερικού εντοπισμού θέσης, χρησιμοποιούνται συσκευές, γνωστές ως BLE beacons, οι οποίες εγκαθίστανται σε γνωστές θέσεις του κτηρίου. Αυτά, μεταδίδουν περιοδικά μικρά πακέτα δεδομένων που περιλαμβάνουν ένα μοναδικό αναγνωριστικό. Αφού η κινητή συσκευή λάβει τα πακέτα, καταγράφει την τιμή RSSI.

Η θέση του χρήστη μπορεί να υπολογιστεί με μεθόδους όπως το fingerprinting και το proximity detection.[10], [16]. Στη μέθοδο ανίχνευσης εγγύτητας (proximity detection) με BLE, η θέση του

χρήστη προσεγγίζεται από τη θέση του beacon από το οποίο λαμβάνεται η ισχυρότερη τιμή RSSI, χωρίς να απαιτείται ακριβής υπολογισμός συντεταγμένων [8].

Το BLE αποτελεί κατάλληλη επιλογή για εντοπισμό θέσης και πλοήγηση σε εσωτερικούς χώρους, λόγω του χαμηλού κόστους, της χαμηλής κατανάλωσης ενέργειας και της ευρείας υποστήριξης του Bluetooth από κινητές συσκευές. Επιπλέον, το μέγεθος των beacons είναι μικρό, οπότε μπορεί εύκολα να τοποθετηθεί σε διαφορετικά σημεία εντός του κτηρίου. Ωστόσο, το κόστος αγοράς, εγκατάστασης και συντήρησης των beacons, αλλά και η τάση των μετρήσεων RSSI να επηρεάζεται από αλλαγές στο περιβάλλον, όπως ανθρώπινη κίνηση αποτελούν σημαντικά μειονεκτήματα για τον ακριβή προσδιορισμό θέσης. [17].

Στην εργασία [17] παρουσιάζεται ένα σύστημα προσβάσιμης πλοήγησης σε εσωτερικούς χώρους, το οποίο αξιοποιεί την τεχνολογία Bluetooth Low Energy (BLE). Για τη λειτουργία του, εγκαθίστανται beacons στον χώρο, καθένα από τα οποία μπορεί να συνδέεται με έως και τέσσερα σημεία ενδιαφέροντος. Αξιοποιώντας τις μετρήσεις της ισχύος του λαμβανόμενου σήματος (RSSI), το σύστημα εκτιμά την εγγύτητα του χρήστη σε ένα beacon και μέσω αυτής της πληροφορίας καθοδηγεί τον χρήστη στον προορισμό του. Το συγκεκριμένο σύστημα έχει τα ίδια γενικά μειονεκτήματα του BLE (κόστος εγκατάστασης και συντήρησης, ευαισθησία του RSSI σε παρεμβολές), γεγονός που επηρεάζει την επεκτασιμότητα του σε μεγαλύτερα και πολυπλοκότερα κτήρια. Παράδειγμα του τρόπου λειτουργίας του συστήματος παρουσιάζεται στο Σχήμα 2.2.



Σχήμα 2.2: Αρχιτεκτονική συστήματος με BLE beacons [17]

2.3.3 Άλλες τεχνολογίες (RFID, UWB)

Άλλες τεχνολογίες που χρησιμοποιούνται συχνά σε συστήματα εσωτερικού εντοπισμού θέσης είναι οι τεχνολογίες Radio Frequency Identification (RFID) και Ultra-Wideband (UWB).

Ένα σύστημα RFID περιλαμβάνει ετικέτες (tags), αναγνώστες (readers) και σύστημα επεξεργασίας δεδομένων. Σε μία τυπική εφαρμογή RFID, οι ετικέτες τοποθετούνται σε γνωστά σημεία του κτηρίου και ο φορητός αναγνώστης ανιχνεύοντας τις ετικέτες, εκτιμά την θέση του [12]. Όταν ανιχνευθεί μία ετικέτα, η εφαρμογή αντιστοιχίζει το αναγνωριστικό της με την αποθηκευμένη θέση της. Διαφορετικά, μπορεί οι αναγνώστες να είναι εγκατεστημένοι στον χώρο και να ανιχνεύουν μία ετικέτα που φέρει η συσκευή του χρήστη.

Στα βασικά πλεονεκτήματα της τεχνολογίας αυτής περιλαμβάνεται το χαμηλό κόστος ανά ετικέτα (tag), η ευκολία εγκατάστασης και η χαμηλή κατανάλωση ενέργειας. Ωστόσο, η εμβέλεια τους είναι συνήθως περιορισμένη και η ακρίβεια εξαρτάται από τον τύπο RFID, τη θέση του εξοπλισμού και τη μέθοδο επεξεργασίας μετρήσεων [18].

Το UWB λειτουργεί στο εύρος συχνοτήτων 3.1-10.6 Ghz. Σε ένα σύστημα εντοπισμού θέσης, εγκαθίστανται σταθεροί δείκτες ή άγκυρες σε γνωστές θέσεις, ενώ ο χρήστης ή το αντικείμενο που παρακολουθείται, φέρει μία ετικέτα UWB. Η θέση υπολογίζεται κυρίως μέσω χρονικών μετρήσεων της διάδοσης του σήματος. Τέτοιες τεχνικές είναι, για παράδειγμα, ο χρόνος άφιξης (Time of Arrival – ToA), ο οποίος εκφράζει τη χρονική στιγμή κατά την οποία το σήμα φτάνει από τον πομπό στον δέκτη και η διαφορά χρόνου άφιξης (Time Difference of Arrival – TDoA), κατά την οποία συγκρίνονται οι χρονικές στιγμές που το σήμα φτάνει σε διαφορετικές συγχρονισμένες άγκυρες, χωρίς να απαιτείται η ακριβής χρονική στιγμή εκπομπής του από την ετικέτα [10], [12].

Ένα σύστημα με UWB χαρακτηρίζεται από χαμηλή κατανάλωση ενέργειας, υψηλό ρυθμό δεδομένων και υψηλή ακρίβεια. Σημαντικό μειονέκτημα ωστόσο είναι το κόστος του εξοπλισμού, καθώς απαιτείται η εγκατάσταση εξειδικευμένων αγκυρών και η πολυπλοκότητα εγκατάστασης, υλοποίησης του συστήματος, καθώς απαιτείται αξιόπιστη λήψη σημάτων και ακριβής συγχρονισμός μεταξύ των συσκευών [10].

2.3.4 Σύστημα AR με οπτικούς δείκτες.

Μία εναλλακτική προσέγγιση συστήματος εντοπισμού και πλοήγησης σε εσωτερικούς χώρους βασίζεται στην τοποθέτηση δεικτών (markers), όπως κωδικών QR, σε προκαθορισμένα σημεία του κτηρίου. Κάθε δείκτης αντιστοιχεί σε γνωστή θέση και αφού σαρωθεί μέσω της κάμερας ενός κινητού τηλεφώνου, επιτρέπει τον προσδιορισμό της θέσης του χρήστη [7].

Ένα παράδειγμα τέτοιου συστήματος παρουσιάζεται στην εργασία [19]. Στο σύστημα αυτό, τοποθετούνται οριζόντιες λωρίδες QR κωδικών κατά μήκος των διαδρόμων του κτηρίου. Με τη σάρωση ενός QR, η εφαρμογή προσδιορίζει την τρέχουσα θέση του χρήστη. Αφού ο χρήστης επιλέξει τον επιθυμητό προορισμό, το σύστημα υπολογίζει τη συντομότερη διαδρομή, με βάση τον αλγόριθμο A*. Οι οδηγίες παρέχονται σε ηχητική μορφή. Αντίστοιχη υλοποίηση αποτελεί και το SINS_AR [20], στο οποίο τοποθετούνται κωδικοί QR σε σημεία ενδιαφέροντος. Ο χώρος αρχικά αναπαρίσταται μέσω πλέγματος κελιών, στο οποίο διακρίνονται οι προσπελάσιμες και μη περιοχές. Από τη δισδιάστατη κάτοψη του κτηρίου δημιουργείται επίσης ένα τρισδιάστατο μοντέλο του. Το σύστημα εφαρμόζει τον αλγόριθμο Theta* για τον υπολογισμό της συντομότερης διαδρομής. Η διαδρομή προβάλλεται στην οθόνη του κινητού με χρήση επαυξημένης πραγματικότητας.

Παρά το χαμηλό κόστος εγκατάστασης κωδικών QR, τα συστήματα που κάνουν χρήση οπτικών δεικτών, εμφανίζουν σημαντικούς περιορισμούς. Αρχικά, η λειτουργία τους περιορίζεται αποκλειστικά σε κτήρια όπου έχουν τοποθετηθεί κατάλληλοι κωδικοί QR. Επιπλέον, η αναγνώριση των δεικτών επηρεάζεται από πληθώρα παραγόντων, όπως ο σωστός φωτισμός, η απόσταση και φθορές στους δείκτες, μειώνοντας, έτσι, την αξιοπιστία προσδιορισμού θέσης.

2.4 Συστήματα χωρίς προσθήκη υποδομών

Σε αντίθεση με συστήματα που βασίζονται σε εγκατάσταση υποδομών, όπως AP ή BLE beacons, τα συστήματα χωρίς πρόσθεση φυσική υποδομή, αξιοποιούν κυρίως την κάμερα και τους ενσωματωμένους αισθητήρες του κινητού τηλεφώνου. Αυτή η προσέγγιση μειώνει το κόστος αρχικής εγκατάστασης και επιτρέπει την επέκταση του συστήματος σε μεγαλύτερα ή πολυπλοκότερα κτήρια, χωρίς να απαιτείται φυσική παρέμβαση στα εκάστοτε κτήρια. Σε τέτοια συστήματα, ο χώρος αναπαρίσταται μέσω ψηφιακής κάτοψης από την χαρτογράφηση του χώρου. Κύριες μεθόδους αποτελούν η μηχανική όραση, ή οπτική-αδρανειακή οδομετρία και το Pedestrian Dead Reckoning [18].

2.4.1 Μηχανική Όραση (Computer Vision)

Τα συστήματα που βασίζονται στην Μηχανική Όραση (Computer Vision) επεξεργάζονται τις εικόνες που λαμβάνουν από την κάμερα, ώστε να εξάγουν οπτικά χαρακτηριστικά από αυτές [8]. Έπειτα, συγκρίνοντας τα με μία βάση δεδομένων αναφοράς ή με τρισδιάστατη αναπαράσταση του χώρου, προσδιορίζει την θέση του χρήστη.

Χαρακτηριστικό παράδειγμα αποτελεί το CamNav ένα camera-based σύστημα, το οποίο αναλύοντας τις εικόνες από την κάμερα με τον τρόπο που περιγράφηκε παραπάνω, εκτιμά τη θέση που βρίσκεται ο χρήστης [21], [22]. Συγκεκριμένα, δημιουργείται μία βάση εικόνων από γνωστά σημεία στο κτήριο. Έπειτα, η εικόνα που καταγράφηκε από τον χρήστη συγκρίνεται με τη βάση, ώστε να αναγνωρισθεί η θέση του. Σύμφωνα με τα αποτελέσματα αξιολόγησης του, πετυχαίνει ακρίβεια έως 97.52%.

Η προσέγγιση αυτή δεν απαιτεί την τοποθέτηση υποδομής στο χώρο, ενώ προσφέρει ακριβή αναγνώριση του. Ωστόσο, παρά την υψηλή ακρίβεια της μηχανικής όρασης, σε τέτοια συστήματα, η συνεχής επεξεργασία εικόνων αυξάνει υπολογιστικές και ενεργειακές απαιτήσεις. Επιπλέον, η απόδοση τους επηρεάζεται από μεταβολές στο περιβάλλον.

2.4.2 Οπτική-αδρανειακή Οδομετρία (Visual – Inertial Odometry)

Η οπτική-αδρανειακή οδομετρία (Visual Inertial Odometry – VIO) συνδυάζει οπτικά δεδομένα από την κάμερα με πληροφορίες του αδρανειακού συστήματος της κινητής συσκευής, όπως επιτάχυνση και γωνιακή ταχύτητα [8]. Το σύστημα με αυτό τον τρόπο εκτιμά διαρκώς τη θέση και τον προσανατολισμό της συσκευής. Η τεχνολογία αυτή θα αναλυθεί περισσότερο στην Ενότητα 3.2

Μέχρι το 2020, τα κυρίαρχα συστήματα ήταν αυτά βασισμένα σε τεχνολογίες που ήδη αναφέρθηκαν, όπως Wi-Fi, BLE και RFID. Ωστόσο, τα τελευταία χρόνια, τα συστήματα εσωτερικής πλοήγησης που χρησιμοποιούν AR σημειώνουν σημαντική ανάπτυξη. Συγκεκριμένα, σύμφωνα με [23], η υιοθέτηση AR πλοήγησης αγγίζει το 45%, με τις markerless AR επιλογές, μέσω πλατφορμών όπως το ARCore, ARKit να φτάνουν το 37%. Η ανάπτυξη πλατφορμών AR, όπως το ARCore και ARKit επιτρέπει τη δημιουργία markerless εφαρμογών AR.

Συγκεκριμένα, το IndoorNavX [23] είναι ένα markerless σύστημα εσωτερικής πλοήγησης, το οποίο αναπτύχθηκε με ARCore και Sceneform. Το σύστημα ανιχνεύει επιφάνειες και τοποθετώντας ψηφιακές άγκυρες στον χώρο, μπορεί να αναπαριστά μία διαδρομή. Τα δεδομένα του χάρτη και των άγκυρών αποθηκεύονται τοπικά, ενώ η καθοδήγηση παρέχεται με τρισδιάστατα βέλη που προβάλλονται στην εικόνα της κάμερας.

Τέτοια συστήματα, έχουν υψηλές υπολογιστικές και ενεργειακές ανάγκες, καθώς το σύστημα είναι υπεύθυνο για την διαρκή σάρωση και σύγκριση του περιβάλλοντος με αποθηκευμένα δεδομένα. Επιπλέον, η αξιοπιστία της μπορεί να μειωθεί σε δυναμικά μεταβαλλόμενα περιβάλλοντα, λόγω

έλλειψης φωτισμού και περιορισμένων οπτικών χαρακτηριστικών. Η γρήγορη καταγραφή επίσης μπορεί να οδηγήσει στην απώλεια παρακολούθησης ή στο drift.

2.4.3 Pedestrian Dead Reckoning

Μία εναλλακτική προσέγγιση χωρίς την ανάγκη κάμερας είναι το PDR. Σε ένα σύστημα PDR, η κίνηση του χρήστη παρακολουθείται διαρκώς μέσω των αδρανειακών αισθητήρων της κινητής συσκευής, όπως το επιταχυνσιόμετρο και το γυροσκόπιο. [8] Η μέθοδος αυτή υπολογίζει τη θέση του χρήστη μέσω της ανίχνευσης των βημάτων του, την εκτίμηση μήκους του βήματος και τον προσδιορισμό της κατεύθυνσης του χρήστη.

Ένα σύστημα με την προσέγγιση του PDR παρουσιάζεται στο [24]. Στο συγκεκριμένο σύστημα, τα δεδομένα των αισθητήρων της κινητής συσκευής χρησιμοποιούνται για την ανίχνευση των βημάτων και την εκτίμηση της θέσης της. Παράλληλα, ένα μοντέλο βαθιάς μάθησης, επεξεργάζεται τις μετρήσεις του βαρομέτρου για την αναγνώριση μεταβάσεων σε διαφορετικούς ορόφους.

Ένα σημαντικό μειονέκτημα στη χρήση PDR αποτελεί το sensor drift. Οι μικρές ανακρίβειες των αισθητήρων με την πάροδο του χρόνου προκαλούν σημαντικές αποκλίσεις της πραγματικής θέσης του χρήστη και για αυτό συνήθως συνδυάζεται με άλλες τεχνολογίες εντοπισμού θέσης σε υβριδικά συστήματα.

2.5 Συστήματα πλοήγησης σε εσωτερικούς χώρους για άτομα με οπτική αναπηρία

Πέρα από τα γενικά συστήματα εσωτερικού εντοπισμού, στη βιβλιογραφία έχουν προταθεί συγκεκριμένες λύσεις για άτομα με οπτική αναπηρία. Η διαφοροποίηση τους βρίσκεται συνήθως στον τρόπο μετάδοσης της πληροφορίας. Οι πληροφορίες σε αυτά δεν μεταφέρονται κυρίως μέσω οπτικών στοιχείων, αλλά μέσω φωνητικών οδηγιών και εντολών.

Στο [25], χρησιμοποιούνται τα δεδομένα από τους αδρανειακούς και μαγνητικούς αισθητήρες του κινητού τηλεφώνου για τον εντοπισμό και παρακολούθηση της κίνησης και θέσης του χρήστη. Δημιουργήθηκαν δύο εφαρμογές: η εφαρμογή Wayfinding καθοδηγεί τον χρήστη στον προορισμό του με βάση την κάτοψη ενός κτηρίου, ενώ η εφαρμογή Backtracking καταγράφει την πορεία του και επιτρέπει την επιστροφή του ακολουθώντας την αντίθετη διαδρομή. Οι εφαρμογές αξιολογήθηκαν από άτομα με οπτική αναπηρία. Όσον αφορά την προσβασιμότητα, η αλληλεπίδραση πραγματοποιείται μέσω έξυπνου ρολογιού, επιτρέποντας τη χρήση της εφαρμογής χωρίς τη συνεχή παρακολούθηση του χώρου μέσω ,για παράδειγμα, κάμερας του κινητού τηλεφώνου. Επιπλέον, οι οδηγίες παρέχονται φωνητικά. Ωστόσο, η ακρίβεια επηρεάζεται από σφάλματα των αδρανειακών αισθητήρων.

Δύο συστήματα που παρουσιάστηκαν στις προηγούμενες υπό-ενότητες έχουν σχεδιαστεί για άτομα με οπτική αναπηρία. Αυτά τα συστήματα είναι το CamNav [21] και το [19]. Το CamNav παρέχει τακτικά φωνητικές πληροφορίες σχετικά με την τρέχουσα θέση του χρήστη, την κατεύθυνση και πιθανά εμπόδια. Παράλληλα, ο χρήστης έχει τη δυνατότητα μέσω φωνητικών εντολών να επιλέξει προορισμό και να ξεκινήσει ή σταματήσει την πλοήγηση. Αντίστοιχα, το [19], παρέχει σε ηχητική μορφή τις πληροφορίες, με την εφαρμογή να παρέχει επιπλέον ειδικές χειρονομίες για τον χειρισμό της. Ωστόσο, η ανάγκη εντοπισμού των κωδικών QR και σωστής ευθυγράμμισης της κάμερας προς αυτούς αποτελεί σημαντική πρόκληση για έναν χρήστη με οπτική αναπηρία. Μία ακόμη προσβάσιμη προσέγγιση παρουσιάστηκε στην εργασία [17]. Κατά τη λειτουργία εξερεύνησης, ανακοινώνονται κοντινά σημεία ενδιαφέροντος, ενώ κατά τη λειτουργία πλοήγησης παρέχονται φωνητικές οδηγίες προς τον προορισμό. Παρέχεται επίσης απτική ανάδραση, με μία δόνηση κατά την προσέγγιση ενός BLE beacon και δύο δονήσεις κατά την προσέγγιση του χρήστη προς τον προορισμό.

Συνεπώς, η προσβασιμότητα ενός συστήματος εξαρτάται από τον τρόπο αλληλεπίδρασης με αυτό. Η φωνητική καθοδήγηση, οι προσβάσιμες χειρονομίες και η απτική ανάδραση μπορούν να αξιοποιηθούν για την προσβάσιμη και ασφαλή μετακίνηση ατόμων με οπτική αναπηρία.

2.6 Συγκριτική αξιολόγηση και συμπεράσματα

Από τη βιβλιογραφική ανασκόπηση, προκύπτει πως κάθε τεχνολογία διαφοροποιείται όσον αφορά την ακρίβεια, την ενεργειακή κατανάλωση και το κόστος εγκατάστασης. Τα συστήματα που βασίζονται σε υποδομή, εξαρτώνται από τον εξοπλισμό εγκατεστημένο στο εσωτερικό ενός κτηρίου.

Συγκεκριμένα, το Wi-Fi μπορεί να αξιοποιήσει το υφιστάμενο δίκτυο, ενώ για τη χρήση των BLE, RFID, UWB απαιτείται η τοποθέτηση και συντήρηση beacons, tags, readers ή αγκυρών. Το κόστος είναι ανάλογο της έκτασης και πολυπλοκότητας του χώρου, περιορίζοντας την επεκτασιμότητα τέτοιων συστημάτων. Τέτοιες προσεγγίσεις είναι καταλληλότερες για νεόκτιστα κτήρια ή κτήρια που ήδη διαθέτουν τον απαιτούμενο εξοπλισμό [26].

Αντίθετα, προσεγγίσεις που χρησιμοποιούν την κάμερα και τους αισθητήρες της κινητής συσκευής για τον εντοπισμό του χρήστη, δεν απαιτούν την εγκατάσταση εξοπλισμού στον εσωτερικό χώρο. Ωστόσο, έχουν υψηλές υπολογιστικές και ενεργειακές ανάγκες, ενώ η απόδοση τους εξαρτάται άμεσα από τις συνθήκες φωτισμού και την έλλειψη οπτικών διακριτών οπτικών χαρακτηριστικών.

Για τις εφαρμογές πλοήγηση ατόμων με οπτική αναπηρία, πολύ σημαντικό ρόλο παίζει ο ακριβής εντοπισμός του χρήστη στον χώρο, καθώς ο χρήστης πρέπει να γνωρίζει το ακριβές σημείο που βρίσκεται στον χώρο, ώστε η πλοήγηση του να ξεκινήσει με ασφάλεια και ακρίβεια. Επιπλέον, όσον αφορά την πλοήγηση, από τη βιβλιογραφία συμπεραίνεται πως η δυνατότητα χειρισμού του συστήματος πρέπει να πραγματοποιείται μέσω μη οπτικών χαρακτηριστικών, όπως χειρονομίες, φωνητική αλληλεπίδραση και απτική ανάδραση [27].

Κεφάλαιο 3ο: Θεωρητικό και τεχνολογικό υπόβαθρο

3.1 Εισαγωγή

Στο παρόν κεφάλαιο παρουσιάζεται το θεωρητικό και τεχνολογικό υπόβαθρο που απαιτείται για την ανάπτυξη της παρούσας εργασίας. Αρχικά, περιγράφονται η έννοια της επαυξημένης πραγματικότητας (AR) και τα βασικά χαρακτηριστικά του ARCore. Στη συνέχεια, παρουσιάζεται η αναπαράσταση του χαρτογραφημένου κτηρίου ως γράφος και ο αλγόριθμος A^* , που χρησιμοποιείται για τον υπολογισμό της διαδρομής προς τον επιλεγμένο προορισμό. Τέλος, αναφέρονται συνοπτικά οι τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής, καθώς και οι τεχνολογίες φωνητικής και απτικής ανάδρασης, που υποστηρίζουν την προσβασιμότητα του συστήματος.

3.2 Θεωρητικό υπόβαθρο

3.2.1 Επαυξημένη πραγματικότητα

Η επαυξημένη πραγματικότητα (Augmented Reality - AR) αποτελεί τεχνολογία που επιτρέπει τον συνδυασμό του πραγματικού του και ψηφιακού κόσμου. Σε αντίθεση με την εικονική πραγματικότητα (Virtual Reality – VR), στην οποία ο χρήστης βρίσκεται πλήρως σε παραγμένο από τον υπολογιστή περιβάλλον, η επαυξημένη πραγματικότητα στοχεύει στην παρουσίαση ψηφιακής πληροφορίας απευθείας στον πραγματικό κόσμο. Σύμφωνα με τον ορισμό του Azuma στο [28], ένα σύστημα επαυξημένης πραγματικότητας χαρακτηρίζεται από τρία βασικά στοιχεία: τον συνδυασμό πραγματικού και ψηφιακού, τη λειτουργία του διαδραστικά σε πραγματικό χρόνο και την ενσωμάτωση των εικονικών αντικειμένων σε τρισδιάστατο (3D) χώρο.

Για τη δημιουργία μίας εφαρμογής επαυξημένης πραγματικότητας, πρέπει η εικονική πληροφορία να παρουσιάζεται σε χωρική και χρονική συσχέτιση με τον πραγματικό κόσμο. Τα εικονικά στοιχεία πρέπει να βρίσκονται στην κατάλληλη θέση, με τον κατάλληλο προσανατολισμό και να προσαρμόζονται στη μεταβολή θέσης του χρήστη. Στόχος είναι ο χρήστης να τα αντιλαμβάνεται ως μέρος του φυσικού περιβάλλοντος, ενισχύοντας συνεπώς την αντίληψη του για αυτό.

Για την υποστήριξη των παραπάνω, αλλά και για την ύπαρξη ενός ολοκληρωμένου συστήματος AR απαιτούνται τουλάχιστον τρεις βασικές λειτουργίες: η παρακολούθηση (tracking), η χωρική ευθυγράμμιση (registration) και κάποιος τρόπος παρουσίασης της πληροφορίας (visualization) [29].

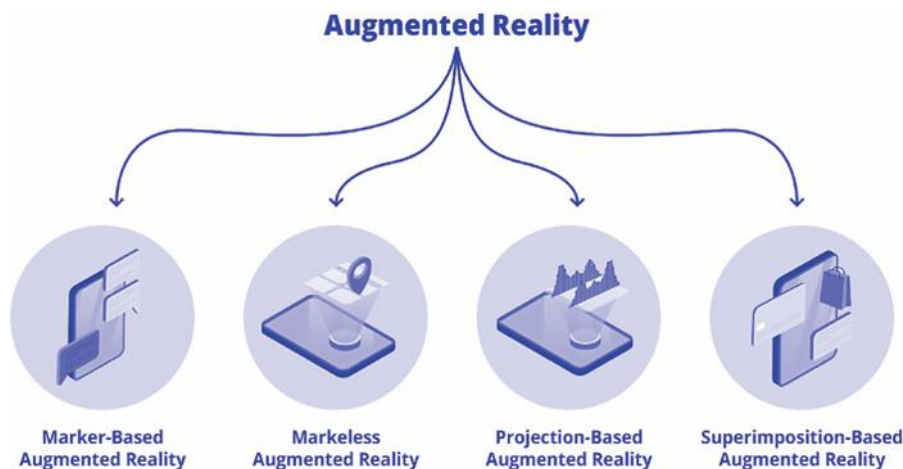
Η παρακολούθηση αποτελεί διαδικασία κατά την οποία εκτιμάται συνεχώς η θέση και ο προσανατολισμός της συσκευής του χρήστη στον τρισδιάστατο χώρο. Στις κινητές εφαρμογές AR πραγματοποιείται μέσω του συνδυασμού των εικόνων της κάμερας, από τις οποίες εντοπίζονται χαρακτηριστικά σημεία του περιβάλλοντος, και των μετρήσεων αισθητήρων, όπως το γυροσκόπιο και το επιταχυνσιόμετρο [30].

Παράλληλα, η παρακολούθηση υποστηρίζεται και μέσω του ταυτόχρονου εντοπισμού και χαρτογράφησης (Simultaneous Localization and Mapping – SLAM). Κατά τη διαδικασία αυτή, η συσκευή προσδιορίζει διαρκώς τη θέση και τον προσανατολισμό της (Pose), δημιουργώντας τον χάρτη ενός αρχικά άγνωστου περιβάλλοντος. Στην παρούσα εφαρμογή, η λειτουργία SLAM παρέχεται από το ARCore, το οποίο συνδυάζοντας τις εικόνες της κάμερας με δεδομένα από τους αισθητήρες της συσκευής, υπολογίζει συνέχεια το Pose της κάμερας.

Με τον όρο χωρική ευθυγράμμιση αναφερόμαστε στην ευθυγράμμιση των συντεταγμένων ανάμεσα σε εικονικά και πραγματικά αντικείμενα. Το σύστημα AR πρέπει να τοποθετεί το εικονικό αντικείμενο στη σωστή θέση και με τον κατάλληλο προσανατολισμό, σε σχέση με το περιβάλλον. Έτσι, το εικονικό αντικείμενο μπορεί να παραμείνει συνδεδεμένο με ένα συγκεκριμένο σημείο του χώρου, ακόμη κι αν η συσκευή κινείται. Τέλος, η παρουσίαση της πληροφορίας αφορά τον τρόπο με τον οποίο ο πραγματικός και εικονικός χώρος γίνονται αντιληπτοί από τον χρήστη. Στις κινητές εφαρμογές, η εικόνα της κάμερας προβάλλεται στην οθόνη και ενισχύεται με εικονικά στοιχεία.

Σύμφωνα με [31] υπάρχουν τέσσερα είδη AR, τα οποία παρουσιάζονται στο Σχήμα 3.1

- AR βασισμένη σε δείκτες (Marker-based AR): Το σύστημα AR αναζητά προκαθορισμένους οπτικούς δείκτες στο περιβάλλον, όπως κωδικούς QR. Μόλις μέσω της κάμερας εντοπισθεί κάποιος δείκτης, προβάλλει σε αυτό το ψηφιακό περιεχόμενο. Το αρνητικό με αυτή την προσέγγιση είναι πως ο δείκτης θα πρέπει να είναι ορατός και ορθά τοποθετημένος στον χώρο.
- AR χωρίς δείκτες (Markerless AR): το σύστημα δεν εξαρτάται από οπτικούς δείκτες, αλλά μέσω των αισθητήρων του κινητού και αλγορίθμων για την αναγνώριση οπτικών χαρακτηριστικών στο περιβάλλον, όπως επιφάνειες, μπορεί να τοποθετήσει ψηφιακό περιεχόμενο σε αυτό. Μία τέτοια τεχνολογία αποτελεί το SLAM.
- AR βασισμένη στην προβολή (Projection based AR): το περιεχόμενο αντί να βρίσκεται σε μία οθόνη, προβάλλεται απευθείας πάνω σε πραγματικές επιφάνειες, όπως σε ένα τοίχο.
- AR βασισμένο στην υπέρθεση (Superimposition based AR): το ψηφιακό περιεχόμενο αντικαθιστά ή ενισχύει μέρη του πραγματικού κόσμου. Για παράδειγμα, μία τέτοια εφαρμογή θα μπορούσε να αντικαταστήσει οπτικά ένα αντικείμενο με ένα διαφορετικό.



Σχήμα 3.1: Διαφορετικοί τύποι AR [31]

3.2.2 Αναπαράσταση του χώρου ως γράφος

Πέρα από τον εντοπισμό θέσης, σημαντικό σημείο για ένα σύστημα εσωτερικής πλοήγησης αποτελεί και ο τρόπος αναπαράστασης του χώρου. Τα μοντέλα εσωτερικής πλοήγησης διακρίνονται σε μετρικά (metric-based), τοπολογικά (topology-based) και δικτυακά (network) [32]. Τα μετρικά αναπαρίστανται με ακριβείς παγκόσμιες συντεταγμένες, ενώ τα τοπολογικά μέσω ενός γράφου, χρησιμοποιούν δηλαδή κόμβους και ακμές. Τέλος, τα δικτυακά μοντέλα συνδυάζουν τις παραπάνω προσεγγίσεις, καθώς οι κόμβοι περιέχουν και πληροφορίες θέσης, για τον υπολογισμό της απόστασης μεταξύ αντικειμένων. Η παρούσα εργασία χρησιμοποιεί το τοπολογικό μοντέλο για την αναπαράσταση του χώρου.

Σύμφωνα με τη θεωρία των δομών δεδομένων [33], ένας γράφος αποδίδεται μαθηματικά ως $G = (V, E)$. Αποτελείται από σύνολο κορυφών ή κόμβων V και σύνολο ακμών E , οι οποίες εκφράζουν τις σχέσεις μεταξύ των κόμβων. Ένας γράφος χαρακτηρίζεται ως μη κατευθυνόμενος, όταν οι ακμές του στερούνται διάταξης, δηλαδή είναι αμφίδρομες. Επιπλέον, ένας γράφος ονομάζεται ζυγισμένος όταν σε κάθε ακμή του αντιστοιχεί ένα κόστος, το οποίο μπορεί να εκφράζει απόσταση, χρόνο, κόστος μετάβασης μεταξύ κόμβων. Ως μονοπάτι από την κορυφή V_p στην κορυφή V_q του γράφου G ορίζεται μία ακολουθία κόμβων, όπου η μετάβαση από τον έναν στον άλλο γίνεται μόνο μέσω μίας καταγεγραμμένης ακμής.

Στην παρούσα εργασία, η αναπαράσταση του εσωτερικού χώρου κάθε χαρτογραφημένου κτηρίου πραγματοποιείται μέσω ενός μη κατευθυνόμενου, ζυγισμένου γράφου. Κατά τη διαδικασία της χαρτογράφησης, τα σημεία ενδιαφέροντος, όπως η είσοδος, οι αίθουσες, καθώς και τα ενδιάμεσα σημεία που καταγράφονται κατά μήκος κάθε διαδρομής, αναπαρίστανται ως κόμβοι ενός γράφου. Οι επιτρεπόμενες μεταβάσεις μεταξύ διαδοχικών σημείων αποδίδονται ως ακμές, με το βάρος κάθε ακμής να χαρακτηρίζει την απόσταση μεταξύ των δύο κόμβων που αυτή συνδέει. Μετά την ολοκλήρωση της χαρτογράφησης, ο γράφος χρησιμοποιείται για την αναζήτηση μίας διαδρομής από την τρέχουσα θέση του χρήστη προς τον επιλεγμένο προορισμό και την καθοδήγηση του προς αυτόν.

3.2.3 Αλγόριθμος A^*

Ο αλγόριθμος A^* αποτελεί αλγόριθμο ευρετικής αναζήτησης για τον υπολογισμό διαδρομών ελάχιστου κόστους σε γράφους. Η αξιολόγηση κάθε κόμβου n πραγματοποιείται μέσω της συνάρτησης $F(n) = g(n) + h(n)$, όπου $g(n)$ είναι το πραγματικό κόστος της διαδρομής από την αρχική κατάσταση μέχρι τον κόμβο, ενώ $h(n)$ είναι η εκτίμηση του ελάχιστου υπολειπόμενου κόστους από τον συγκεκριμένο κόμβο μέχρι την κατάσταση στόχου. Σε κάθε επανάληψη επιλέγεται για εξέταση ο κόμβος με την μικρότερη τιμή $f(n)$, εξετάζονται οι γειτονικοί του κόμβοι και ενημερώνεται το κόστος τους, εφόσον εντοπιστεί καλύτερη διαδρομή [34].

Η απόδοση του εξαρτάται από την ποιότητα της ευρετικής συνάρτησης. Όσο πιο κοντά βρίσκεται η τιμή $h(n)$ στο πραγματικό υπόλοιπο κόστος, τόσο περισσότερο περιορίζεται ο χώρος αναζήτησης. Αν το $h(n)=0$ για όλους τους κόμβους, τότε ο A^* λειτουργεί όπως ο αλγόριθμος Dijkstra.

Στην παρούσα εφαρμογή, ο αλγόριθμος A^* χρησιμοποιείται για τον υπολογισμό της διαδρομής από την θέση του χρήστη προς τον επιλεγμένο προορισμό. Αφού το χαρτογραφημένο κτήριο ανακτηθεί, τα καταγεγραμμένα σημεία και οι μεταξύ τους συνδέσεις σχηματίζουν έναν ζυγισμένο γράφο. Ως αρχικός κόμβος επιλέγεται αυτός που αντιστοιχεί στην πλησιέστερη καταγεγραμμένη θέση του χρήστη, ενώ ως κόμβος-στόχος ορίζεται ο επιλεγμένος προορισμός. Το πραγματικό κόστος ($g(n)$) υπολογίζεται ως το άθροισμα των αποστάσεων των ακμών που έχουν διασχιστεί από τον αρχικό κόμβο έως τον κόμβο (n). Η ευρετική συνάρτηση ($h(n)$) υπολογίζεται με βάση την ευκλείδεια απόσταση του κόμβου (n) από τον προορισμό στο οριζόντιο επίπεδο

$$h(n) = \sqrt{\{(x_n - x_t)^2 + (z_n - z_t)^2\}} \quad (3.1)$$

όπου (x_n, z_n) είναι οι οριζόντιες συντεταγμένες του εξεταζόμενου κόμβου (node) και $((x_t, z_t))$ οι αντίστοιχες συντεταγμένες του προορισμού (target). Το αποτέλεσμα του αλγορίθμου είναι μια

διατεταγμένη ακολουθία κόμβων από την αφετηρία έως τον προορισμό. Η ακολουθία αυτή αποτελεί τη διαδρομή πλοήγησης και χρησιμοποιείται στη συνέχεια από την εφαρμογή για την παραγωγή των κατάλληλων φωνητικών και απτικών οδηγιών προς τον χρήστη

3.3 Τεχνολογίες ανάπτυξης εφαρμογής

3.3.1 Android και Kotlin

Το Android είναι ένα από τα πιο ευρέως διαδεδομένα λειτουργικά συστήματα ανοικτού κώδικα και είναι βασισμένο στον πυρήνα Linux. Έχει σχεδιασθεί κυρίως για συσκευές που έχουν οθόνη αφής, όπως κινητά τηλέφωνα, tablets, έξυπνα ρολόγια. [35]. Η αρχιτεκτονική του Android περιλαμβάνει διαφορετικά επίπεδα, τα οποία επιτρέπουν στις εφαρμογές να χρησιμοποιούν τις υπηρεσίες του συστήματος, χωρίς να απαιτείται η αλληλεπίδραση τους με το υλικό της συσκευής. Σε υψηλό επίπεδο, μέσω των διαθέσιμων API, οι εφαρμογές μπορούν να αξιοποιήσουν λειτουργίες της συσκευής, όπως η κάμερα και οι αισθητήρες, ενώ σε χαμηλότερο επίπεδο βρίσκονται οι βιβλιοθήκες που παρέχει το σύστημα.

Η Kotlin αποτελεί μία ανοιχτού κώδικα, στατικά τυποποιημένη γλώσσα προγραμματισμού, η οποία υποστηρίζει αντικειμενοστραφή και συναρτησιακό προγραμματισμό. Χαρακτηρίζεται από συνοπτική σύνταξη, διαλειτουργικότητα με τη Java και μηχανισμούς ασφάλειας έναντι μηδενικών αναφορών (null safety), περιορίζοντας έτσι τα σφάλματα λόγω πρόσβασης σε μηδενικές αναφορές. Επιπλέον, υποστηρίζει τα coroutines, τα οποία επιτρέπουν την εκτέλεση ασύγχρονων εργασιών, χωρίς τη δέσμευση του κύριου νήματος της εφαρμογής, για την εκτέλεση λειτουργιών, όπως η ανάκτηση δεδομένων από απομακρυσμένες υπηρεσίες, διατηρώντας ομαλή τη λειτουργία της διεπαφής χρήστη. Η Kotlin αποτελεί την προτεινόμενη γλώσσα της Google για τη σύγχρονη ανάπτυξη εφαρμογών Android [35].

3.3.2 Jetpack Compose

Το Jetpack Compose αποτελεί σύγχρονο εργαλείο για την ανάπτυξη γραφικών διεπαφών χρήστη (UI) native εφαρμογών Android. Κατά την παραδοσιακή ανάπτυξη διεπαφών, ο προγραμματιστής έπρεπε να αλλάζει τα επιμέρους στοιχεία του UI ως απόκριση στις αλλαγές της εφαρμογής. Αντίθετα, η Jetpack Compose βασίζεται στο δηλωτικό μοντέλο προγραμματισμού, όπου η διεπαφή περιγράφεται με βάση την τρέχουσα κατάσταση της και αποτελεί χρήσιμο εργαλείο σε εφαρμογές με δυναμικό περιεχόμενο. [35].

Τα βασικά στοιχεία του Compose είναι οι composable συναρτήσεις, οι οποίες χρησιμοποιούνται για την περιγραφή των στοιχείων της διεπαφής. Έτσι η διεπαφή οργανώνεται σε επαναχρησιμοποιήσιμα και ανεξάρτητα στοιχεία. Επιπλέον, κεντρική έννοια αποτελεί η κατάσταση (state), η οποία περιλαμβάνει πληροφορίες που μπορεί να μεταβληθούν κατά την εκτέλεση της εφαρμογής και επηρεάζουν τον τρόπο με τον οποίο εμφανίζεται η διεπαφή. Κάθε φορά που μία τιμή κατάστασης αλλάζει, το Compose πραγματοποιεί ένα re-composition των composable συναρτήσεων που εξαρτώνται από αυτή την τιμή.

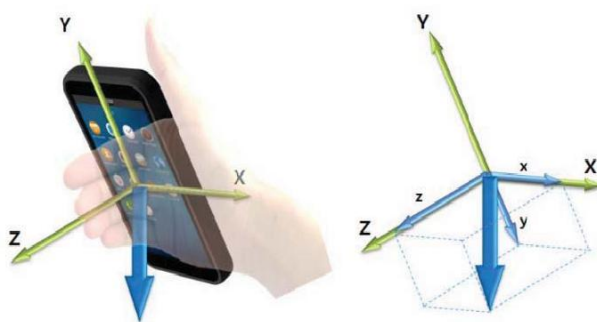
3.3.3 ARCore

Το ARCore αποτελεί πλατφόρμα της Google για την ανάπτυξη εφαρμογών επαυξημένης πραγματικότητας σε κινητές συσκευές [30], [36]. Παρέχει API, τα οποία επιτρέπουν στη συσκευή την κατανόηση και αλληλεπίδραση με το γύρω περιβάλλον. Αξιοποιώντας την κάμερα και τους αισθητήρες

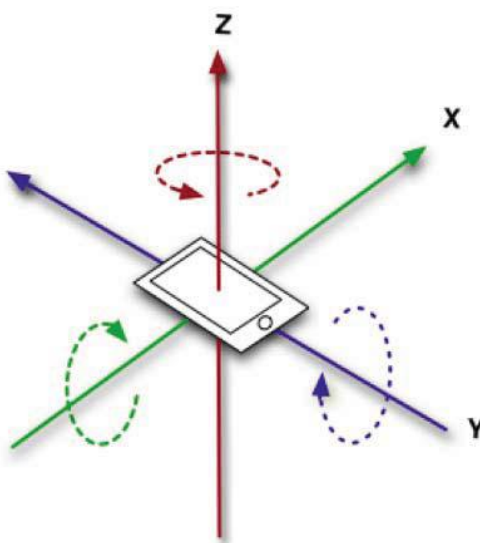
της κινητής συσκευής, μπορεί να αναγνωρίσει χαρακτηριστικά στο περιβάλλον, επιτρέποντας την τοποθέτηση ψηφιακών στοιχείων σε αυτό. Οι λειτουργίες του ARCore εκτελούνται μέσα σε μία συνεδρία (Session) ARCore. Η συνεδρία αποτελεί το βασικό σημείο πρόσβασης στις λειτουργίες του ARCore και είναι υπεύθυνη για τη διαχείριση της κατάστασης και του κύκλου ζωής του συστήματος AR.

Στο πλαίσιο, λοιπόν, μίας συνεδρίας, η λειτουργία του ARCore βασίζεται σε τρεις βασικές δυνατότητες: α) παρακολούθηση κίνησης (motion tracking), β) κατανόηση του περιβάλλοντος (environmental understanding) και γ) εκτίμηση του φωτισμού (light estimation) [36].

Αρχικά, για την παρακολούθηση κίνησης, το ARCore χρησιμοποιεί τη μέθοδο SLAM, που περιγράφηκε στην Ενότητα 3.2.1. Συγκεκριμένα, εντοπίζει διακριτά χαρακτηριστικά σημεία στις εικόνες της κάμερας και τα συνδυάζει με δεδομένα από την μονάδα αδρανειακών μετρήσεων της συσκευής (Inertial Measurement Unit - IMU). Η IMU περιλαμβάνει το επιταχυνσιόμετρο, το οποίο μετρά τη γραμμική επιτάχυνση στους τρεις άξονες (x,y,z), και το γυροσκόπιο, το οποίο καταγράφει τη γωνιακή ταχύτητα γύρω από τους ίδιους άξονες. Έτσι, επιτρέπει στο ARCore τη διαρκή εκτίμηση του Pose της κάμερας. Παραδείγματα της λειτουργίας των δύο αισθητήρων παρουσιάζονται στα Σχήματα 3.2 και 3.3



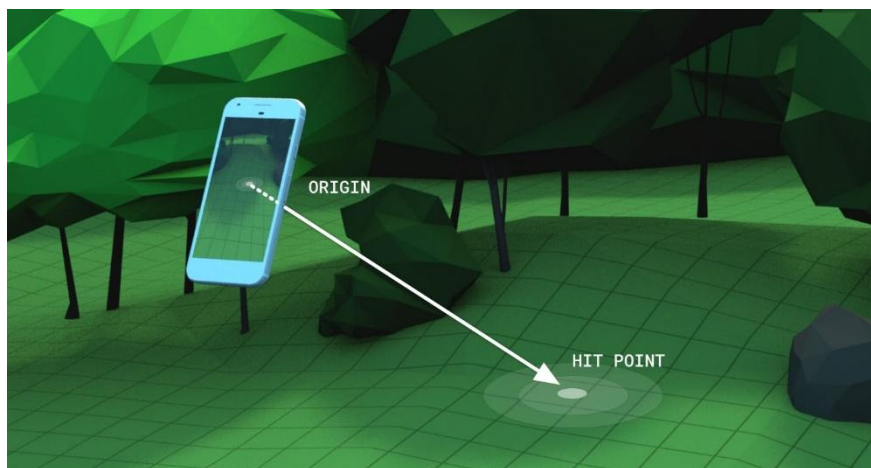
Σχήμα 3.2: Λειτουργία επιταχυνσιόμετρου [37]



Σχήμα 3.3: Λειτουργία γυροσκοπίου [37]

Εκτός της παρακολούθησης κίνησης, σημαντικό ρόλο παίζει και η δυνατότητα κατανόησης του περιβάλλοντος. Το ARCore αναζητά διακριτά σημεία, όπως γωνίες επίπλων ή περιοχές με μεταβολές στην υφή. Όταν εντοπίζεται συστάδα τέτοιων σημείων σε ίδιο ύψος και προσανατολισμό, που φαίνεται δηλαδή να ανήκουν σε κοινή οριζόντια ή κατακόρυφη επιφάνεια, το σύστημα μπορεί να θεωρήσει την επιφάνεια αυτή ως επίπεδο (Plane). Το επίπεδο αυτό μπορεί να είναι είτε οριζόντιο, όπως ένα δάπεδο, είτε κάθετο, όπως ένας τοίχος.

Η επιλογή θέσης πάνω στο επίπεδο αυτό απαιτεί την εκτέλεση ενός hit-test. Το hit-test αποτελεί διαδικασία κατά την οποία μια εικονική ακτίνα εκπέμπεται από την κάμερα της συσκευής προς τον τρισδιάστατο χώρο, με σκοπό τον εντοπισμό του σημείου στο οποίο τέμνει μια επιφάνεια που έχει αναγνωριστεί από το σύστημα επαυξημένης πραγματικότητας [35]. Το αποτέλεσμα περιλαμβάνει, μεταξύ άλλων, την απόσταση από την αρχή της ακτίνας και το Pose του σημείου τομής, το οποίο μπορεί να χρησιμοποιηθεί για την τοποθέτηση ενός εικονικού αντικειμένου στο πραγματικό περιβάλλον.



Σχήμα 3.4: Παράδειγμα hit-test [36]

Παράλληλα, μέσω της εκτίμησης φωτισμού, το σύστημα εκτιμά τον περιβάλλοντα φωτισμό της σκηνής, παρέχοντας πληροφορίες για αυτόν, όπως η ένταση. Αυτές οι πληροφορίες μπορούν να χρησιμοποιηθούν κατά την απόδοση ψηφιακών αντικείμενων, εξασφαλίζοντας την ρεαλιστική προσαρμογή αυτών στον πραγματικό χώρο.

Για την σύνδεση εικονικών αντικείμενων σε συγκεκριμένες θέσεις του πραγματικού χώρου, χρησιμοποιούνται οι άγκυρες (Anchors). Κάθε άγκυρα συνδέεται με ένα συγκεκριμένο Pose και λειτουργεί ως τοπικό σημείο αναφοράς για τα αντικείμενα που είναι εξαρτημένα από αυτό. Όσο το ARCore ενημερώνει το μοντέλο του περιβάλλοντος, προσαρμόζει και το Pose της άγκυρας, διασφαλίζοντας την τοποθέτηση εικονικών αντικείμενων. Ωστόσο, η λειτουργία των απλών αγκυρών περιορίζεται στην εκάστοτε τρέχουσα συνεδρία (Session) AR.

Για την αντιμετώπιση του περιορισμού αυτού, παρέχεται ένα ειδικό είδος άγκυρας, τα Cloud Anchors, τα οποία επιτρέπουν τη διατήρηση και ανάκτηση μίας άγκυρας σε διαφορετικές συνεδρίες ή συσκευές [35]. Κατά τη φιλοξενία ενός Cloud Anchor, το ARCore λαμβάνει το Pose της συσκευής καταγράφει οπτικά χαρακτηριστικά γύρω από την άγκυρα και επιστρέφει ένα μοναδικό Cloud Anchor ID. Κατά την ανάκτηση, συγκρίνει τα χαρακτηριστικά του περιβάλλοντος με τα αποθηκευμένα και ανακατασκευάζει τη θέση και τον προσανατολισμό της άγκυρας.

Στην παρούσα εφαρμογή, το ARCore χρησιμοποιείται τόσο κατά τη χαρτογράφηση, όσο και κατά την πλοήγηση. Κατά τη χαρτογράφηση, ανιχνεύονται οριζόντια επίπεδα, τα οποία αντιστοιχούν στο δάπεδο και πραγματοποιούνται hit-tests για την τοποθέτηση σημείων στο περιβάλλον. Τα σημεία ενδιαφέροντος, όπως η κύρια είσοδος και οι αίθουσες φιλοξενούνται ως Cloud Anchors. Κατά την πλοήγηση, η εφαρμογή προσπαθεί να επιλύσει τα αποθηκευμένα Cloud Anchors, ώστε να πραγματοποιηθεί αντιστοίχιση της τρέχουσας συνεδρίας AR με το χαρτογραφημένο κτήριο.

3.3.4 SceneView

Το SceneView αποτελεί βιβλιοθήκη ανοιχτού κώδικα για την ανάπτυξη τρισδιάστατων (3D) εφαρμογών και εφαρμογών επαυξημένης πραγματικότητας (AR). Αποτελεί τον διάδοχο του παρωχημένου Sceneform της Google, του οποίου η επίσημη ανάπτυξη έχει διακοπεί. Το Sceneform είχε δημιουργηθεί με σκοπό την απλούστευση δημιουργίας εφαρμογών AR, παρέχοντας ένα υψηλότερου επιπέδου API πάνω από τις λειτουργίες του ARCore, χωρίς να απαιτούνται γνώσεις OpenGL [38]. Το SceneView διατηρεί αυτή τη φιλοσοφία παρέχοντας περιβάλλον υψηλού επιπέδου για τη δημιουργία και διαχείριση τρισδιάστατων και AR σκηνών.

Στην πλατφόρμα Android, το SceneView βασίζεται στον μηχανισμό απόδοσης (rendering engine) Google Filament, και το ίδιο το SceneView παρέχει ένα υψηλότερου επιπέδου API για τη διαχείριση της σκηνής και των αντικειμένων της. Ως βασικό framework διεπαφής χρησιμοποιείται το Jetpack Compose, ώστε η τρισδιάστατη σκηνή να περιγράφεται με δηλωτικό (declarative) τρόπο. Έτσι, το περιεχόμενο της σκηνής μπορεί να καθορίζεται με βάση την κατάσταση της εφαρμογής και να ενημερώνονται δυναμικά όταν αυτή μεταβάλλεται.

Το SceneView παρέχει έτοιμους μηχανισμούς για τη διαχείριση τρισδιάστατων αντικειμένων, τη φόρτωση μοντέλων και την οργάνωση του εικονικού περιεχομένου στη σκηνή. Το εικονικό περιεχόμενο οργανώνεται μέσα σε μία σκηνή (Scene), ενώ τα επιμέρους στοιχεία της σκηνής αναπαρίστανται μέσω κόμβων (Nodes). Ο κόμβος αποτελεί το βασικό στοιχείο χωρικής τοποθέτησης και διαχείρισης, καθώς διαθέτει ιδιότητες θέσης, προσανατολισμού και κλίμακας στον τρισδιάστατο χώρο.

Η δομή αυτή είναι σημαντική σε εφαρμογές επαυξημένης πραγματικότητας, όπου το εικονικό περιεχόμενο πρέπει να τοποθετείται σε συγκεκριμένες θέσεις και να παραμένει χωρικά συσχετισμένο με το πραγματικό περιβάλλον. Ανάλογα με τον σκοπό χρήσης, παρέχονται ειδικοί τύποι κόμβων, οι οποίοι αντιστοιχίζουν το εικονικό περιεχόμενο με τα δεδομένα του πραγματικού περιβάλλοντος. Μεταξύ αυτών, υπάρχει ο βασικός Node, ο AnchorNode, που διασφαλίζει την αγκύρωση ενός κόμβου σε πραγματική θέση στον χώρο που παρέχεται από το ARCore και ο CloudAnchorNode, που είναι ειδικά σχεδιασμένος για τη διαχείριση και επίλυση Cloud Anchors.

Ειδικότερα, το ARCore συνεργάζεται με το SceneView. Το ARCore παρέχει πληροφορίες για το πραγματικό περιβάλλον, όπως η ανίχνευση επιπέδων (planes) και δεδομένα παρακολούθησης, ενώ το SceneView χρησιμοποιεί αυτές τις πληροφορίες για να αποδώσει τα αντίστοιχα εικονικά αντικείμενα.

Παράλληλα, το SceneView αναλαμβάνει μεγάλο μέρος της διαχείρισης του τρισδιάστατου περιεχομένου και της διαδικασίας απόδοσης. Μέσω του μηχανισμού απόδοσης Google Filament, τρισδιάστατα μοντέλα και άλλα γραφικά στοιχεία μπορούν να αποδοθούν, χωρίς να απαιτείται άμεση χρήση και γνώση λειτουργιών της μηχανής γραφικών σε χαμηλό επίπεδο.

Ένα επιπλέον χαρακτηριστικό του SceneView είναι η υποστήριξη αλληλεπίδρασης με τα τρισδιάστατα αντικείμενα. Η βιβλιοθήκη περιλαμβάνει μηχανισμούς για χειρονομίες, ανίχνευση συγκρούσεων και

raycasting. Οι δυνατότητες αυτές επεκτείνουν τη λειτουργικότητα της βιβλιοθήκης, πέρα από την απλή εμφάνιση τρισδιάστατων αντικειμένων. Επιπλέον, υποστηρίζει πολλαπλές πλατφόρμες. Εκτός από το Android, το SceneView επεκτείνεται σε περιβάλλοντα όπως iOS, macOS, Web και άλλες. Ανάλογα με την πλατφόρμα χρησιμοποιούνται διαφορετικοί μηχανισμοί απόδοσης και frameworks.

Στην παρούσα εργασία, το SceneView χρησιμοποιείται για την αρχικοποίηση και διαμόρφωση της συνεδρίας AR, λήψη ενημερώσεων κάθε καρέ (frame) και απόδοση του εικονικού περιεχομένου που αναπαριστά τις άγκυρες και τα ενδιάμεσα σημεία πλοήγησης (Waypoints).

3.3.5 Cloud Firestore

Το Cloud Firestore αποτελεί μία NoSQL βάση δεδομένων, που φιλοξενείται στο υπολογιστικό νέφος (cloud) και βασίζεται στην υποδομή του Google Cloud. [39]. Σε αντίθεση με τις σχεσιακές βάσεις δεδομένων, στις οποίες τα δεδομένα οργανώνονται σε πίνακες και γραμμές, τα δεδομένα αποθηκεύονται σε έγγραφα (documents), τα οποία αποτελούνται από ζεύγη πεδίων και τιμών και προσδιορίζονται από ένα μοναδικό αναγνωριστικό και οργανώνονται σε συλλογές (collections).

Σημαντικό χαρακτηριστικό είναι πως δεν απαιτείται προκαθορισμένο σχήμα (schema), όπως στις σχεσιακές βάσεις δεδομένων. Τα πεδία ενός εγγράφου μπορεί να διαφέρουν από εκείνα ενός άλλου εγγράφου της ίδιας συλλογής. Έτσι, προσφέρεται ευελιξία κατά την ανάπτυξη μίας εφαρμογής, καθώς η δομή των δεδομένων προσαρμόζεται στις ανάγκες του εκάστοτε συστήματος. Επιπλέον, τα δεδομένα μπορούν να ανακτώνται από το cloud όταν χρειάζεται, ενώ παράλληλα η εφαρμογή να ενημερώνεται όταν αλλάζουν τα αντίστοιχα δεδομένα στην βάση. Έτσι, επιτυγχάνεται ο συγχρονισμός των δεδομένων μεταξύ εφαρμογής και απομακρυσμένης βάσης.

Πρόσβαση στα δεδομένα παρέχεται μέσω APIs και SDKs που παρέχονται από το Firebase για υποστηριζόμενες πλατφόρμες. Για το λειτουργικό σύστημα Android, η εφαρμογή έχει δυνατότητες ανάγνωσης, δημιουργίας, ενημέρωσης και διαγραφής εγγράφων. Η επικοινωνία με το Firebase γίνεται μέσω σύνδεσης διαδικτύου. Η απομακρυσμένη αποθήκευση δεδομένων επιτρέπει την κοινή χρήση ίδιου συνόλου δεδομένων μεταξύ διαφορετικών συσκευών.

Στην παρούσα εργασία, το Cloud Firestore χρησιμοποιείται για την απομακρυσμένη αποθήκευση των χαρτογραφημένων κτηρίων, η οποία είναι απαραίτητη για τη λειτουργία της εφαρμογής. Η επιλογή του Cloud Firestore εγγυάται την άμεση ενσωμάτωση του σε εφαρμογές Android και στην ευελιξία του εγγραφοκεντρικού μοντέλου.

3.4 Τεχνολογίες προσβασιμότητας

Με γνώμονα την ανεξάρτητη χρήση της εφαρμογής από άτομα με οπτική αναπηρία, αξιοποιούνται οι εξής τεχνολογίες: σύνθεση ομιλίας, αναγνώριση ομιλίας και απτική ανάδραση του Android. Οι τεχνολογίες αυτές εξαλείφουν την ανάγκη οπτικής επαφής με την οθόνη και συμβάλλουν στη δημιουργία ενός πολυτροπικού συστήματος, το οποίο προσαρμόζεται στις ανάγκες των χρηστών [2], [4].

3.4.1 Σύνθεση ομιλίας

Για την παροχή φωνητικών οδηγιών, χρησιμοποιείται η κλάση TextToSpeech του Android, η οποία μετατρέπει το γραπτό κείμενο σε συνθετική ομιλία και επιτρέπει την άμεση αναπαραγωγή της από τη συσκευή [40]. Η κλάση αποτελεί μέρος του πακέτου android.speech.tts και επιτρέπει στις εφαρμογές

να στέλνουν κείμενο σε μηχανισμό σύνθεσης ομιλίας, χωρίς να απαιτείται η αποθήκευση προκαθορισμένων ηχητικών αρχείων.

Η λειτουργία του TextToSpeech απαιτεί αρχικά την αρχικοποίηση του μηχανισμού, μέσω του `onInitListener`. Μετά την ολοκλήρωση χρήσης πρέπει να απελευθερωθούν οι πόροι που χρησιμοποίησε ο μηχανισμός, μέσω της `shutdown()`. Ο χρήστης έχει την δυνατότητα καθορισμού γλώσσας σύνθεσης, με την συνάρτηση `setLanguage()`. Αυτή, δέχεται ένα αντικείμενο `Locale` και ο μηχανισμός αναλαμβάνει να χρησιμοποιήσει την καταλληλότερη φωνή για την γλώσσα που ορίστηκε. Άλλες δυνατότητες που παρέχονται περιλαμβάνουν την αλλαγή ταχύτητας της ομιλίας, μέσω της `setSpeechRate()`, την ρύθμιση του ύψους της φωνής, μέσω της `setPitch()` και την επιλογή συγκεκριμένης φωνής, μέσω της `setVoice()`. Έτσι η φωνητική έξοδος μπορεί να προσαρμοστεί σε μεγάλο βαθμό στις απαιτήσεις της κάθε εφαρμογής.

Για τον χειρισμό των φωνητικών μηνυμάτων, παρέχονται ειδικές συναρτήσεις. Η αναπαραγωγή ενός φωνητικού μηνύματος πραγματοποιείται μέσω της μεθόδου `speak()`, η οποία λειτουργεί ασύγχρονα. Αντίθετα, η μέθοδος `stop()` διακόπτει την τρέχουσα εκφώνηση μηνύματος και απορρίπτει όσα βρίσκονται στην ουρά. Επιπλέον, είναι δυνατή η ενημέρωση της εφαρμογής για το στάδιο μίας εκφώνησης, μέσω του `UtteranceProgressListener`. Έτσι, ανιχνεύονται γεγονότα που είτε ξεκίνησαν να εκφωνούνται, είτε ολοκληρώθηκαν είτε η εκφώνηση τους απέτυχε.

Στη συγκεκριμένη εφαρμογή το TextToSpeech αποτελεί το βασικό τρόπο φωνητικής εξόδου και χρησιμοποιείται για την παροχή προειδοποιητικών ή κατευθυντήριων οδηγιών, όπως και για την παροχή ενημερωτικών μηνυμάτων προς τον χρήστη. Οι οδηγίες παράγονται δυναμικά σε συσχέτιση με την κατάσταση πλοήγησης και περιλαμβάνουν πληροφορίες σχετικά με την κατεύθυνση, προσανατολισμό της κίνησης, την απόσταση από το επόμενο σημείο, καθώς και την άφιξη στον προορισμό. Αποτελεί βασικό μηχανισμό αλληλεπίδρασης, καθώς επιτρέπει στον χρήστη την λήψη βασικών πληροφοριών, χωρίς εξάρτηση από οπτικά στοιχεία.

3.4.2 Αναγνώριση ομιλίας

Για την αναγνώριση ομιλίας του χρήστη και την μετατροπή του σε κείμενο επεξεργάσιμο από την εφαρμογή, χρησιμοποιείται το `SpeechRecognizer` API. [41]. Η κλάση `SpeechRecognizer` παρέχει πρόσβαση στην υπηρεσία αναγνώρισης ομιλίας του συστήματος και επιτρέπει στην εφαρμογή να ξεκινήσει και να διαχειριστεί μία συνεδρία αναγνώρισης.

Η διαδικασία αναγνώρισης ξεκινά μέσω της μεθόδου `startListening()`, όπου περιέχεται ένα αντικείμενο `Intent` με παραμέτρους της αναγνώρισης. Για αυτό, χρησιμοποιείται η κλάση `RecognizerIntent`, η οποία παρέχει σταθερές και παραμέτρους για τη διαμόρφωση της διαδικασίας. Για παράδειγμα, η επιλογή `ACTION_RECOGNIZE_SPEECH` στέλνει στο σύστημα αίτημα αναγνώρισης ομιλίας και επιστρέφει το αποτέλεσμα στην εφαρμογή.

Η επικοινωνία της εφαρμογής με τη διαδικασία αναγνώρισης ομιλίας πραγματοποιείται μέσω του `RecognizerIntent`. Το `RecognizerIntent` παρέχει σταθερές και παραμέτρους για τη διαμόρφωση του `Intent` που χρησιμοποιείται στη διαδικασία αναγνώρισης ομιλίας. Γεγονότα που μπορούν να ορισθούν στην εφαρμογή είναι η έναρξη ακρόασης, η λήψη αποτελεσμάτων, η ολοκλήρωση ακρόασης. Συγκεκριμένα, τα αποτελέσματα αναγνώρισης επιστρέφονται σε μορφή λίστας πιθανών φράσεων, από την οποία η εφαρμογή μπορεί να επιλέξει το καταλληλότερο για επιπλέον επεξεργασία. Παράλληλα, μερικά αποτελέσματα κατά την ομιλία μπορούν να επιστραφούν, μέσω της παραμέτρου `EXTRA_PARTIAL_RESULTS`.

Η διαδικασία αναγνώρισης ομιλίας μπορεί να τελειώσει με διαφορετικούς τρόπους. Η `stopListening()` σταματά την ακρόαση και επιτρέπει την ολοκλήρωση της διαδικασίας, ενώ η `cancel()` ακυρώνει τη διαδικασία. Τέλος, με την ολοκλήρωση χρήσης της αναγνώρισης ομιλίας, συνεπώς και του αντικειμένου `SpeechRecognizer`, πρέπει να καλείται η `destroy()`, η οποία απελευθερώνει τους , σχετιζόμενους με την υπηρεσία αναγνώρισης ομιλίας, πόρους.

3.4.3 Απτική ανάδραση

Η απτική ανάδραση αποτελεί μηχανισμό επικοινωνίας ανάμεσα στον χρήστη και την εφαρμογή, με τη χρήση της δόνησης της κινητής συσκευής. Στο λειτουργικό σύστημα Android, η λειτουργία αυτή παρέχεται μέσω των API δόνησης, συγκεκριμένα, μέσω των κλάσεων `Vibrator`, `VibratorManager`, `VibrationEffect`. Οι συγκεκριμένες κλάσεις επιτρέπουν τη δημιουργία διαφορετικών μοτίβων δόνησης, με διακύμανση στην διάρκεια και ένταση τους [42].

Για τη διαχείριση της δόνησης είναι υπεύθυνη η κλάση `Vibrator`, η οποία παρέχει πρόσβαση στον μηχανισμό δόνησης της συσκευής και καλεί ένα `VibrationEffect`. Η `VibratorManager` παρέχει πρόσβαση στους μηχανισμούς δόνησης που διαθέτει η συσκευή και επιτρέπει τη διαχείρισή τους. Έτσι, μπορεί να ανακτηθεί ο προεπιλεγμένος `Vibrator` και να ληφθούν πληροφορίες για τους μηχανισμούς δόνησης. Επιπλέον, αν η συσκευή διαθέτει περισσότερους από έναν μηχανισμό δόνησης είναι διαθέσιμος ο συγχρονισμένος συνδυασμός εφέ.

Η `VibrationEffect` χρησιμοποιείται για την περιγραφή του εφέ που πρόκειται να εκτελεστεί από έναν `Vibrator`. Το Android υποστηρίζει διαφορετικές μορφές εφέ, όπως απλή δόνηση συγκεκριμένης διάρκειας, έως πιο σύνθετες κυματομορφές. Για απλή δόνηση, καθορίζεται η διάρκεια και το πλάτος (amplitude) της, το οποίο εκφράζει την ένταση με την οποία ενεργοποιείται ο μηχανισμός δόνησης. Ο έλεγχος του πλάτους δεν υποστηρίζεται από όλες τις συσκευές. Φυσικά, παρέχονται και προκαθορισμένα εφέ, μέσω της `VibrationEffect`, όπως το `EFFECT_CLICK`, `EFFECT_DOUBLE_CLICK`. Το Android επιτρέπει επιπλέον και τη δημιουργία κυματομορφών (waveforms). Οι κυματομορφές επιτρέπουν τη μεταβολή της δόνησης κατά τη διάρκεια του χρόνου.

Κεφάλαιο 4ο: Ανάλυση απαιτήσεων και σχεδιασμός διεπαφής

4.1 Λειτουργικές και μη λειτουργικές απαιτήσεις

Στο στάδιο της ανάλυσης απαιτήσεων, προσδιορίζονται οι βασικές λειτουργίες που πρέπει να διαθέτει η εφαρμογή, καθώς και οι προδιαγραφές που πρέπει να πληροί, ώστε να επιτυγχάνεται ο βασικός της στόχος, η ασφαλής και αυτόνομη πλοήγηση ατόμων με οπτική αναπηρία σε εσωτερικούς χώρους. Οι απαιτήσεις αυτές διακρίνονται σε λειτουργικές και μη λειτουργικές.

4.1.1 Λειτουργικές απαιτήσεις

- Πρόσβαση στις δύο βασικές λειτουργίες της εφαρμογής: χαρτογράφηση από διαχειριστή, πλοήγηση από τελικό χρήστη.
- Αυθεντικοποίηση διαχειριστή και δυνατότητα προβολής, προσθήκης, μετονομασίας ή διαγραφής χαρτογραφημένων κτηρίων.
- Χαρτογράφηση εσωτερικών χώρων με την τοποθέτηση Cloud Anchors στα σημεία ενδιαφέροντος (όπως κεντρική είσοδος, αίθουσες) και Waypoints σε ενδιάμεσα σημεία της έγκυρης διαδρομής.
- Έλεγχος εγκυρότητας, αποθήκευση και ανάκτηση των χαρτογραφημένων κτηρίων.
- Επιλογή κτηρίου και προορισμού, εντοπισμός θέσης του χρήστη μέσω επίλυσης των αποθηκευμένων Cloud Anchors από το ARCore.
- Υπολογισμός της διαδρομής προς τον προορισμό και παροχή φωνητικών οδηγιών και συμπληρωματικών μοτίβων δόνησης, με δυνατότητα αναγνώρισης απόκλισης από την σωστή πορεία.

4.1.2 Μη λειτουργικές απαιτήσεις

- Χαμηλή καθυστέρηση: Η εφαρμογή οφείλει να εξασφαλίζει την έγκαιρη παροχή φωνητικών και απτικών οδηγιών, ώστε να αποφευχθεί πιθανή σύγχυση και αποπροσανατολισμός του τελικού χρήστη. [43]
- Ασφάλεια και αξιοπιστία καθοδήγησης: Η εφαρμογή πρέπει να αποφεύγει παραπλανητικές και ασαφείς οδηγίες και να ενημερώνει τον χρήστη όταν δεν είναι δυνατή η αξιόπιστη καθοδήγηση του. [4]
- Ακρίβεια εντοπισμού θέσης και καθοδήγησης: Η εφαρμογή πρέπει να εκτιμά με ακρίβεια την θέση του χρήστη, ώστε να υπολογίζεται σωστά η διαδρομή προς τον προορισμό του.
- Συμβατότητα: Η εφαρμογή πρέπει να λειτουργεί σε κινητές συσκευές Android που υποστηρίζουν το ARCore.
- Προσβασιμότητα και ευχρηστία: Η χρήση των βασικών λειτουργιών της εφαρμογής δεν πρέπει να εξαρτάται από οπτικά στοιχεία. Η εφαρμογή πρέπει να παρέχει σύντομες φωνητικές οδηγίες και συμπληρωματικά μοτίβα δόνησης για την καθοδήγηση στον προορισμό του τελικού χρήστη.

4.1.3 Απαιτήσεις προσβασιμότητας

- Οι βασικές λειτουργίες της εφαρμογής δεν πρέπει να εξαρτώνται από την οπτικά στοιχεία στη διεπαφή, αλλά πρέπει να μπορούν να εκτελούνται μέσω φωνητικών οδηγιών, και αναγνώρισης ομιλίας.
- Η φωνητική καθοδήγηση πρέπει να είναι σύντομη, σαφής και έγκαιρη, ώστε να περιορισθεί το γνωστικό φορτίο του τελικού χρήστη.
- Σημαντικά γεγονότα, όπως η στροφή κατά την πλοήγηση ή η άφιξη στον προορισμό, θα πρέπει να τονίζεται συμπληρωματικά μέσω μοτίβων δόνησης
- Η εφαρμογή πρέπει να ενημερώνει τον τελικό χρήστη για απώλεια παρακολούθησης του ARCore μέσω της κάμερας, και συνεπώς αδυναμία αξιόπιστης καθοδήγησης.

- Ο χρήστης θα πρέπει να έχει τη δυνατότητα επανάληψης της τελευταίας οδηγίας και απενεργοποίησης της απτικής ανάδρασης, εφόσον το επιθυμεί.

4.2 Σενάρια Χρήσης

4.2.1 Σενάριο Χρήσης Α: Χαρτογράφηση κτηρίου

Στο σενάριο χρήσης χαρτογράφησης κτηρίου, χρήστης είναι ο διαχειριστής, με στόχο την χαρτογράφηση του εσωτερικού χώρου ενός κτηρίου. Προϋποθέσεις για την εκτέλεση του σεναρίου αποτελούν η ύπαρξη έγκυρων στοιχείων διαχειριστή, η παραχώρηση άδειας πρόσβασης στην κάμερα, η ύπαρξη σύνδεσης στο διαδίκτυο και η παρουσία του χρήστη στον χώρο που επιθυμεί να χαρτογραφήσει.

Η διαδικασία της χαρτογράφησης ξεκινά με την επιλογή σύνδεσης ως διαχειριστής και την εισαγωγή των απαραίτητων στοιχείων ταυτοποίησης. Με την επιτυχή σύνδεση του χρήστη ως διαχειριστή, εμφανίζεται ο πίνακας διαχείρισης των κτηρίων. Εκεί, είναι δυνατή η προβολή των ήδη χαρτογραφημένων κτηρίων, όπου ο διαχειριστής έχει τη δυνατότητα διαγραφής ή μετονομασίας κάποιου χαρτογραφημένου κτηρίου, καθώς και η έναρξη χαρτογράφησης νέου. Στην τελευταία περίπτωση, καταχωρεί την ονομασία του κτηρίου και ξεκινά η διαδικασία καταγραφής του χώρου.

Κατά τη χαρτογράφηση, ο διαχειριστής κινείται στο εσωτερικό του κτηρίου και μέσω της κάμερας, το ARCore αναλαμβάνει την ανίχνευση του περιβάλλοντος. Αρχικά, τοποθετεί ένα Cloud Anchor στην κύρια είσοδο του κτηρίου. Στη συνέχεια, τοποθετούνται επιπλέον Cloud Anchors σε σημεία ενδιαφέροντος, σε διαθέσιμους προορισμούς δηλαδή, όπως αίθουσες και Waypoints σε ενδιάμεσα σημεία στη διαδρομή και σε αλλαγές κατεύθυνσης. Στις διασταυρώσεις διαδρόμων, ο διαχειριστής μπορεί, επιλέγοντας ένα ήδη καταγεγραμμένο σημείο, να ξεκινήσει από αυτό την χαρτογράφηση μίας νέας διακλάδωσης. Σε κτήρια με περισσότερα από έναν ορόφους, καταγράφονται επίσης οι μεταβάσεις μέσω σκάλας, με την τοποθέτηση Cloud Anchors στην αρχή και το τέλος της σκάλας.

Αφού ολοκληρωθεί η καταγραφή, ο διαχειριστής επιλέγει το κουμπί ολοκλήρωσης της χαρτογράφησης. Το σύστημα αφού εκτελέσει κάποιους ελέγχους, όπως ότι έχουν καταγραφεί τα απαραίτητα σημεία και ότι σχηματίζεται ένα ολοκληρωμένο μονοπάτι μεταξύ αυτών, αποθηκεύει τον χάρτη, ο οποίος προστίθεται στη λίστα των χαρτογραφημένων κτηρίων και τίθεται διαθέσιμος για πλοήγηση. Εάν κάποιος έλεγχος αποτύχει, ο διαχειριστής ενημερώνεται και καλείται να διορθώσει τη χαρτογράφηση.

4.2.2 Σενάριο Χρήσης Β: Πλοήγηση σε κτήριο

Στο σενάριο χρήσης πλοήγησης, χρήστης είναι το άτομο με οπτική αναπηρία, με στόχο την αυτόνομη καθοδήγησή του από την τρέχουσα θέση του στον επιλεγμένο προορισμό στο εσωτερικό ενός κτηρίου. Η αλληλεπίδραση με εφαρμογή πραγματοποιείται κυρίως μέσω φωνητικών οδηγιών και αναγνώριση ομιλίας. Προϋποθέσεις για την εκτέλεση του σεναρίου είναι η ύπαρξη αποθηκευμένου χαρτογραφημένου κτηρίου, η χρήση συσκευής που υποστηρίζει ARCore, η παραχώρηση πρόσβασης στην κάμερα και στο μικρόφωνο, η σύνδεση της συσκευής στο διαδίκτυο και η φυσική παρουσία του χρήστη στο χώρο, στον οποίο επιθυμεί να καθοδηγηθεί.

Με την εκκίνηση της εφαρμογής, ο χρήστης ενημερώνεται πως για να ξεκινήσει τη διαδικασία πλοήγησης, πρέπει να πατήσει στο κέντρο της οθόνης. Με την επιλογή της λειτουργίας πλοήγησης, η εφαρμογή ζητά από τον χρήστη να επιλέξει το κτήριο στο οποίο επιθυμεί να καθοδηγηθεί. Ο χρήστης αναφέρει προφορικά την ονομασία του κτηρίου και το σύστημα αναγνωρίζει και επιβεβαιώνει την απάντηση. Αντίστοιχα, η εφαρμογή ζητά από τον χρήστη να εκφωνήσει τον προορισμό που επιθυμεί να

μεταβεί. Η απάντηση του αντιστοιχίζεται με τους διαθέσιμους προορισμούς του επιλεγμένου κτηρίου. Αν η εφαρμογή επιβεβαιώσει την τοποθεσία, ξεκινά η διαδικασία της πλοήγησης, διαφορετικά ζητά επανάληψη της απάντησης. Τα διαθέσιμα κτήρια και οι προορισμοί μπορούν να επιλεγούν και μέσω των αντίστοιχων στοιχείων στην οθόνη.

Κατά την έναρξη της διαδικασίας πλοήγησης, η εφαρμογή ζητά από τον χρήστη να κρατά τη συσκευή του στραμμένη προς τα εμπρός και ελαφρώς προς τα κάτω και να κινείται αργά στον χώρο, ώστε αρχικά να προσδιορισθεί η θέση του στο κτήριο. Αφού προσδιορισθεί η θέση του, η πλοήγηση είναι έτοιμη και κατά την κίνηση του χρήστη παρέχονται φωνητικές οδηγίες σχετικά με την κατεύθυνση που πρέπει να ακολουθήσει και την απόσταση που θα διανύσει. Όπως θα αναλυθεί στο 4.3.2, σημαντικά γεγονότα, όπως αλλαγή κατεύθυνσης, προειδοποιήσεις, άφιξη στον προορισμό ενισχύονται μέσω απτικής ανάδρασης. Ο χρήστης έχει επιπλέον τη δυνατότητα επανάληψης της τελευταίας οδηγίας με διπλό πάτημα στο πάνω μέρος της οθόνης.

Εάν χαθεί προσωρινά η παρακολούθηση της θέσης του χρήστη, η εφαρμογή διακόπτει την παροχή οδηγιών και ενημερώνει τον χρήστη φωνητικά και μέσω μοτίβου δόνησης, προτρέποντας το να διακόψει την κίνηση του και να σαρώσει ξανά το περιβάλλον. Όταν αποκατασταθεί ο εντοπισμός θέσης, η καθοδήγηση συνεχίζεται. Αντίστοιχα, σε περίπτωση που ο χρήστης απομακρυνθεί από την διαδρομή, ενημερώνεται με τον ίδιο τρόπο και υπολογίζεται εκ νέου η διαδρομή από τον κοντινότερο κόμβο, εφόσον αυτό είναι εφικτό. Τέλος, κατά την άφιξη στον επιλεγμένο προορισμό, η εφαρμογή παρέχει φωνητική ανακοίνωση και απτική ειδοποίηση, ολοκληρώνοντας τη διαδικασία της πλοήγησης.

4.3 Σχεδιασμός διεπαφής

Ο σχεδιασμός της γραφικής διεπαφής πρέπει να περιλαμβάνει χρήση χρωμάτων υψηλής αντίθεσης, ευδιάκριτα μεγέθη κειμένου και στοιχείων, επισήμανση των λειτουργιών με ακρίβεια, και η υποστήριξη τεχνολογιών προσβασιμότητας, όπως οι αναγνώστες οθόνης (πχ Talkback). Οι αρχές αυτές είναι σημαντικές για χρήστες με μερική όραση, καθώς μπορούν να χρησιμοποιούν τα οπτικά στοιχεία συμπληρωματικά.

Για την βελτίωση της αναγνωσιμότητας, υποστηρίζεται φωτεινό και σκοτεινό θέμα, ανάλογα με τη ρύθμιση της συσκευής. Και στα δύο θέματα χρησιμοποιούνται σταθερές χρωματικές παλέτες, όπου το χρώμα του κειμένου διακρίνεται πλήρως από αυτό του φόντου. Στο φωτεινό θέμα χρησιμοποιείται κυρίως μαύρο κείμενο σε λευκό ή ανοιχτό γκρι φόντο, ενώ στο σκοτεινό θέμα, λευκό κείμενο σε σκούρο γκρι φόντο. Οι βασικοί συνδυασμοί χρώματος κειμένου και φόντου ελέγχθηκαν σύμφωνα με τις οδηγίες προσβασιμότητας του Android [44], και ικανοποιούν τον λόγο αντίθεσης 4:5:1 για κείμενο μικρού μεγέθους και 3:1 για μεγαλύτερου μεγέθους.

Όσον αφορά την τυπογραφία της εφαρμογής, για τον τίτλο της εφαρμογής χρησιμοποιείται 40sp, ενώ στις υπόλοιπες οθόνες χρησιμοποιούνται μεγέθη από 18sp έως 28sp για βασικές πληροφορίες, τίτλους, κουμπιά. Μικρότερα μεγέθη προορίζονται για δευτερεύουσες πληροφορίες.

Σχετικά με το μέγεθος των βασικών διαδραστικών στοιχείων, η κύρια επιλογή έναρξης πλοήγησης καταλαμβάνει μεγάλο μέρος της αρχικής οθόνης, ώστε να επιλεγεί με ευκολία. Τα περισσότερα κουμπιά έχουν μέγεθος 48 ή 56dp, ακολουθώντας τη σύσταση του Android για ελάχιστη περιοχή αφής 48x48dp.

Ένα επιπλέον χαρακτηριστικό προσβάσιμων εφαρμογών Android, σύμφωνα με τις συστάσεις του [44] είναι η προσθήκη περιγραφών περιεχόμενου στα βασικά διαδραστικά στοιχεία της εφαρμογής, ώστε οι υπηρεσίες προσβασιμότητας, όπως το Talkback να μπορούν να αναγνωρίζουν και να ανακοινώνουν τη λειτουργία κάθε στοιχείου στον χρήστη. Στην παρούσα εφαρμογή, περιγραφές μέσω της ιδιότητας

contentDescription έχουν αποδοθεί στα κουμπιά επιστροφής, ανοίγματος ρυθμίσεων, εμφάνισης οδηγιών χαρτογράφησης. Κατά την επεξεργασία και διαγραφή, η περιγραφή περιλαμβάνει και την ονομασία του κτηρίου. Τέλος, το στοιχείο ενεργοποίησης ή απενεργοποίησης της απτικής ανάδρασης χαρακτηρίζεται ως διακόπτης, ώστε το Talkback να έχει τη δυνατότητα να ανακοινώσει τη λειτουργία και την τρέχουσα κατάσταση του.

4.4 Στοιχεία προσβασιμότητας

Σε ανώνυμη διαδικτυακή έρευνα που πραγματοποιήθηκε το 2021, ερευνήθηκαν οι ανάγκες, οι επιθυμίες και τα προβλήματα χρήσης τεχνολογικών λύσεων εσωτερικής και εξωτερικής πλοήγησης για άτομα με προβλήματα όρασης. [45]. Οι απαντήσεις ήταν από 78 άτομα με οπτική αναπηρία, οι 25 εκ των οποίων δηλώνουν ειδικοί στη

χρήση ηλεκτρονικών βοηθημάτων μετακίνησης (Electronic Travel Aid – ETA) για άτομα με οπτική αναπηρία, με 10 χρόνια εμπειρία. Παρατηρήθηκε, πως περισσότερο από το 40% είναι δυσαρεστημένο από τις διαθέσιμες επιλογές εσωτερικής πλοήγησης. Επιπλέον, στην έρευνα [27], 25 ειδικοί στη χρήση ETA με 10+ χρόνια εμπειρίας κλήθηκαν να αναφέρουν τα σημαντικότερα προβλήματα που αντιμετωπίζουν κατά την πλοήγηση τους σε εσωτερικούς χώρους. Ως σημαντικότερη δυσκολία κρίθηκε η εύρεση μίας αίθουσας από τον αριθμό της. Μεταξύ των επόμενων προβλημάτων, αναφέρθηκε η εύρεση ανελκυστήρων, η εύρεση της εισόδου και η αναγνώριση αντικειμένων και πιθανών εμποδίων. Τα αποτελέσματα αυτά αναδεικνύουν την ανάγκη παροχής σαφών πληροφοριών και οδηγιών προς τον προορισμό, τις μεταβάσεις μεταξύ ορόφων και τα δυναμικά στοιχεία του περιβάλλοντος.

Σύμφωνα με την ανασκόπηση των Agora et.al [4], κατά τον σχεδιασμό διεπαφών για άτομα με οπτική αναπηρία, σημαντική κρίνεται η παροχή πολυτροπικής ανάδρασης. Ειδικότερα, η συνθετική ομιλία, η οποία αποτελεί το συνηθέστερο τρόπο ανάδρασης σε τέτοιου είδους εφαρμογές [7], μπορεί να συνδυαστεί με απτική ανάδραση (όπως η δόνηση της κινητής συσκευής). Στην παρούσα εφαρμογή, η συνθετική ομιλία αποτελεί το βασικό μέσο μετάδοσης πληροφοριών στον χρήστη, τόσο για την ενημέρωση των λειτουργιών της εφαρμογής, όσο και για την παροχή οδηγιών κατά την πλοήγηση του στον χώρο. Παράλληλα, η δόνηση προστίθεται ως συμπληρωματικό μέσο για την προειδοποίηση σημαντικών γεγονότων, όπως η αλλαγή κατεύθυνσης κατά την πλοήγηση του χρήστη ή άφιξη στον επιλεγμένο προορισμό.

4.4.1 Σχεδιασμός φωνητικής αλληλεπίδρασης

Η φωνητική αλληλεπίδραση βασίζεται στην μνήμη του χρήστη, καθώς η μετάδοση πληροφορίας μέσω ομιλίας είναι παροδική. Σύμφωνα με τους Murad et al. [43], οι φωνητικές οδηγίες πρέπει να είναι σύντομες και ακριβείς, περιλαμβάνοντας μόνο τις απαραίτητες για την εκτέλεση της εκάστοτε ενέργειας πληροφορίες. Η παροχή υπερβολικής ακουστικής πληροφορίας είναι πιθανό να προκαλέσει γνωστική επιβάρυνση, σύγχυση ή ακόμη και απόσπαση της προσοχής του χρήστη από τους ήχους στο περιβάλλον, το οποίο μπορεί να οδηγήσει στην μη ασφαλή μετακίνηση του [4].

Μία ακόμη σημαντική αρχή κατά τη σχεδίαση φωνητικής αλληλεπίδρασης είναι η παροχή ακριβής και έγκαιρης ανάδρασης. Ο χρήστης πρέπει να ενημερώνεται για το αν η απάντηση του αναγνωρίστηκε, ποιο γεγονός εκτελείται και αν υπήρξε κάποιο σφάλμα κατά την εκτέλεση. Σε περίπτωση αποτυχίας αναγνώρισης, η εφαρμογή πρέπει να τον καθοδηγεί για επανάληψη ή διόρθωση της ενέργειας του. Επίσης, πολύ σημαντική κρίνεται και η συνέπεια κατά τη διατύπωση φωνητικών μηνυμάτων. Παρόμοια γεγονότα πρέπει να περιγράφονται με παρόμοια ορολογία και δομή προτάσεων. Η χρήση απλού

λεξιλογίου συμβάλλει στην ευκολότερη κατανόηση των οδηγιών. Τέλος, τονίζεται η ανάγκη για ανθρωποκεντρική σχεδίαση και η διατήρηση της αίσθησης ελέγχου του χρήστη, καθώς μεταξύ χρηστών οι ανάγκες, δυνατότητες και απαιτήσεις διαφέρουν σε μεγάλο βαθμό [4].

Με βάση της παραπάνω αρχές, στην παρούσα εφαρμογή, αρχικά η επιλογή κτηρίου και προορισμού γίνεται σε διαδοχικά στάδια, ενώ οι οδηγίες πλοήγησης περιορίζονται σε πληροφορίες απαραίτητες για την μετακίνηση του χρήστη. Αν κάποια απάντηση του χρήστη δεν αναγνωρισθεί, ο χρήστης ενημερώνεται και καλείται να επαναλάβει. Επιπροσθέτως, η εφαρμογή παρέχει τη δυνατότητα επανάληψης της τελευταίας φωνητικής οδηγίας, καθώς και μέσω μενού ρυθμίσεων την ενεργοποίηση ή απενεργοποίηση της συμπληρωματικής απτικής ανάδρασης, διατηρώντας έτσι έναν βαθμό ελέγχου κατά την αλληλεπίδραση του με την εφαρμογή.

4.4.2 Σχεδιασμός μοτίβων δόνησης

Σύμφωνα με τον ορισμό του [46], ένα μοτίβο δόνησης χαρακτηρίζεται από την επαναλαμβανόμενη ενεργοποίηση και παύση του μηχανισμού δόνησης. Διαφοροποιείται ως προς επιμέρους χαρακτηριστικά, όπως η διάρκεια, ο ρυθμός και τα χρονικά διαστήματα ανάμεσα στους παλμούς. Ο συνδυασμός αυτών των χαρακτηριστικών ευνοεί τη δημιουργία διακριτών μοτίβων δόνησης, τα οποία αντιστοιχούν σε διαφορετική κατηγορία γεγονότος.

Για τον σχεδιασμό των μοτίβων απτικής ανάδρασης μέσω δονήσεων, αξιοποιήθηκαν οι αρχές που παρουσιάζονται στο [46]. Αρχικά, παρατηρήθηκε πως η χρήση διαφορετικού μοτίβου δόνησης για κάθε μεμονωμένο γεγονός επιβαρύνει το γνωσιακό φορτίο του χρήστη, επιδρώντας αρνητικά στην κατανόηση και αναγνώριση των ίδιων των μοτίβων. Για τον λόγο αυτό, τα γεγονότα της εφαρμογής κατηγοριοποιήθηκαν σε βασικές ομάδες, καθεμία από τις οποίες αντιστοιχεί σε κοινό μοτίβο δόνησης. Παράλληλα, κάθε μοτίβο δόνησης διαφέρει από τα υπόλοιπα ως προς τον ρυθμό ή τη διάρκεια των παλμών, με σκοπό την αποφυγή σύγχυσης κατά το περπάτημα και χρήση της εφαρμογής. Τα μοτίβα αυτά παρουσιάζονται στον Πίνακα 4.1.

Κατηγορία γεγονότος	Μοτίβο δόνησης
Επιβεβαίωση χειρονομίας	Ένας σύντομος παλμός
Αλλαγή κατεύθυνσης	Δύο σύντομοι παλμοί
Αλλαγή ορόφου	Τρεις παλμοί μεσαίας διάρκειας
Προειδοποίηση	Τέσσερις σύντομοι παλμοί
Άφιξη στον προορισμό	Μακρύς – σύντομος – μακρύς παλμός

Πίνακας 4.1: Περιγραφή των μοτίβων δόνησης που χρησιμοποιούνται στην εφαρμογή.

Η σχέση μεταξύ της διάρκειας του μοτίβου και της κρισιμότητας του γεγονότος αποτέλεσε βασικό πυλώνα στον σχεδιασμό των μοτίβων. Γεγονότα που απαιτούν άμεση προσοχή ή δράση από τον χρήστη, όπως η στροφή κατά την καθοδήγηση προς έναν προορισμό, χρειάζονται σύντομα και γρήγορα μοτίβα. Αντίθετα, μη επείγοντα, επιβεβαιωτικά γεγονότα, αποδίδονται με μοτίβα μεγαλύτερης διάρκειας.

Για την επιβεβαίωση χειρονομίας, επιλέχθηκε ένας σύντομος παλμός, που δίνει την αίσθηση του πατήματος ενός κουμπιού.

Για την αλλαγή κατεύθυνσης, επιλέχθηκε μοτίβο δύο σύντομων παλμών. Το ίδιο μοτίβο χρησιμοποιείται τόσο για την αριστερή, όσο και για τη δεξιά στροφή, καθώς η ακριβής κατεύθυνση προσδιορίζεται μέσω των φωνητικών οδηγιών. Λειτουργεί ως επισήμανση ότι απαιτείται αλλαγή πορείας. Έτσι, εξαλείφεται η ανάγκη δημιουργίας διαφορετικών μοτίβων για κάθε κατεύθυνση. Για τη μετάβαση μεταξύ ορόφων επιλέχθηκαν τρεις παλμοί μεσαίας διάρκειας, ώστε το μοτίβο να διακρίνεται ξεκάθαρα από την απλή αλλαγή κατεύθυνσης. Η δόνηση ενημερώνει για προσέγγιση σε σημείο μετάβασης μέσω σκάλας.

Για τις προειδοποιήσεις επιλέχθηκε μοτίβο τεσσάρων σύντομων παλμών. Το σύντομο μοτίβο έχει σκοπό την άμεση προσοχή του χρήστη σε καταστάσεις, όπου επηρεάζεται η αξιοπιστία της καθοδήγησης και η ασφάλεια του χρήστη. Παραδείγματα τέτοιων καταστάσεων αποτελούν η απόκλιση από την υπολογισμένη διαδρομή, η απώλεια παρακολούθησης από το ARCore.

Τέλος, η άφιξη στον προορισμό αποδίδεται μέσω ενός ενιαίου, μεγαλύτερης διάρκειας παλμού. Καθώς το συγκεκριμένο γεγονός δεν αποτελεί επείγον γεγονός, αλλά δηλώνει επιβεβαίωση επιτυχούς ολοκλήρωσης της διαδρομής, επιλέχθηκε μεγαλύτερης διάρκειας μοτίβο.

Για τη σωστή και αποτελεσματική χρήση της απτικής ανάδρασης είναι απαραίτητο οι χρήστες της να εξοικειωθούν με τα μοτίβα δόνησης. Καθώς η παρούσα εφαρμογή δεν περιλαμβάνει ξεχωριστή λειτουργία εκμάθησης των μοτίβων, τα μοτίβα θα πρέπει να παρουσιάζονται στους συμμετέχοντες πριν τη χρήση της εφαρμογής. Επιπλέον, η προσθήκη της λειτουργίας εκμάθησης μοτίβων θα μπορούσε να αποτελέσει μελλοντική επέκταση της εφαρμογής.

Κεφάλαιο 5ο: Υλοποίηση συστήματος

5.1 Εισαγωγή

Το παρόν κεφάλαιο περιγράφει αναλυτικά την τεχνική υλοποίηση του συστήματος χαρτογράφησης και πλοήγησης σε εσωτερικούς χώρους για άτομα με οπτική αναπηρία. Αρχικά, παρουσιάζεται το μοντέλο δεδομένων της εφαρμογής, καθώς και ο τρόπος αποθήκευσης και ανάκτησης δεδομένων. Στη συνέχεια αναλύεται η τεχνική υλοποίηση των δύο δυνατοτήτων της εφαρμογής, δηλαδή της χαρτογράφησης και της πλοήγησης και οι μηχανισμοί φωνητικής και απτικής ανάδρασης. Τέλος, παρουσιάζεται η ροή των δύο λειτουργιών, μέσω των βασικών οθονών που προσφέρονται. Η εφαρμογή αναπτύχθηκε για το λειτουργικό σύστημα Android, με χρήση της γλώσσας προγραμματισμού Kotlin.

5.2 Μοντέλο δεδομένων

Κάθε χαρτογραφημένο κτήριο αναπαρίσταται ως γράφος, ο οποίος δημιουργείται και αποθηκεύεται κατά την χαρτογράφηση. Κατά την πλοήγηση, τα δεδομένα του γράφου ανακτώνται και χρησιμοποιούνται για την ανακατασκευή του γράφου και τον υπολογισμό της διαδρομής προς τον επιλεγμένο προορισμό. Οι κόμβοι του γράφου αντιστοιχούν στους άγκυρες (Cloud Anchors) και στα ενδιάμεσα σημεία της διαδρομής (Waypoints), ενώ, οι ακμές αναπαριστούν τις επιτρεπόμενες και διαθέσιμες συνδέσεις μεταξύ τους. Από αυτούς, όσοι έχουν χαρακτηριστεί ως προορισμοί εμφανίζονται στις διαθέσιμες επιλογές του τελικού χρήστη. Παρακάτω παρουσιάζονται οι κλάσεις που υλοποιούν αυτή την αναπαράσταση.

5.2.1 Κλάση FloorPlan

Η κλάση FloorPlan, η οποία παρουσιάζεται στο Σχήμα 5.1, αντιπροσωπεύει το τελικό χαρτογραφημένο κτήριο, το οποίο χαρακτηρίζεται από το μοναδικό αναγνωριστικό του (floorPlanId) και το όνομα του (buildingName).

Το πεδίο originNodeId περιέχει το αναγνωριστικό της πρώτης άγκυρας που φιλοξενείται επιτυχώς κατά την χαρτογράφηση. Η θέση και ο προσανατολισμός όλων των υπόλοιπων αγκυρών που τοποθετούνται, εκφράζονται σε σχέση με το σύστημα αναφοράς της πρώτης. Έτσι Το σύστημα δεν βασίζεται στο προσωρινό σύστημα συντεταγμένων της εκάστοτε συνεδρίας AR.

Οι λίστες nodes και connections συμβάλλουν στην αναπαράσταση του χώρου ως γράφο. Αρχικά, η λίστα nodes περιλαμβάνει αντικείμενα τύπου PathStep, τα οποία εκφράζουν είτε Cloud Anchors είτε Waypoints. Αντίστοιχα, η λίστα connections περιλαμβάνει αντικείμενα τύπου PathConnection, που αναπαριστούν τις ακμές του γράφου και καθορίζουν τις συνδέσεις μετακίνησης μεταξύ ζευγών κόμβων.

Με την ολοκλήρωση της χαρτογράφησης, το αντικείμενο FloorPlan αποθηκεύεται στο Cloud Firestore. Κατά την έναρξη της πλοήγησης ανακτάται από τη βάση και χρησιμοποιείται για την ανακατασκευή του γράφου και τον υπολογισμό διαδρομής προς τον επιλεγμένο προορισμό.

```

FloorPlan.kt

data class FloorPlan(
    val floorPlanId: String = "",
    val buildingName: String = "",
    val originNodeId: String = "",
    val nodes: List<PathStep> = emptyList(),
    val connections: List<PathConnection> = emptyList()
)

```

Σχήμα 5.1: Ορισμός της κλάσης FloorPlan

5.2.2 Κλάση PathStep

Η κλάση Pathstep, η οποία παρουσιάζεται στην Εικόνα 5.2, αναπαριστά έναν κόμβο του γράφου. Κάθε κόμβος διαθέτει αρχικά ένα μοναδικό αναγνωριστικό (id). Για τους κόμβους που αντιστοιχούν σε άγκυρες χρησιμοποιείται το Cloud Anchor ID που επιστρέφεται με την επιτυχή φιλοξενία τους, ενώ για τα ενδιάμεσα Waypoints, δημιουργείται τοπικά αναγνωριστικό με πρόθεμα “wp_”. Το πεδίο isAnchor διακρίνει τις δύο κατηγορίες των κόμβων.

Επιπλέον, το πεδίο label περιγράφει το όνομα του σημείου, όπως «Είσοδος», «Αίθουσα B1», το πεδίο createdAt δηλώνει την χρονική στιγμή δημιουργίας του και το πεδίο floor προσδιορίζει τον όροφο στον οποίο βρίσκεται ο κόμβος.

Πληροφορίες για την χωρική θέση του κόμβου αποθηκεύονται μέσω των πεδίων x,y,z, που αναπαριστούν τις συντεταγμένες θέσης του σημείου στο κοινό τοπικό σύστημα συντεταγμένων του χάρτη, που ορίζεται από την πρώτη άγκυρα. Αντίστοιχα, μέσω των πεδίων qx,qy,qz,qw περιγράφονται οι συνιστώσες του τετραδονίου που περιγράφει τον προσανατολισμό του κόμβου.

Τέλος, το πεδίο isDestination προσδιορίζει αν ένα σημείο εμφανίζεται ως διαθέσιμος προορισμός στον τελικό χρήστη. Ως προορισμοί ορίζονται τα σημεία τα οποία αντιστοιχούν σε ένα πραγματικό προορισμό στο κτήριο, όπως οι αίθουσες. Τα ενδιάμεσα σημεία πλοήγησης Waypoints, τα οποία καταγράφονται για τη χάραξη της επιτρεπόμενης πορείας, όπως και σημεία που τοποθετούνται για την αρχή και το τέλος μίας σκάλας δε θα πρέπει να εμφανίζονται ως διαθέσιμοι προορισμοί.

```

PathStep.kt
data class PathStep(
    // Cloud Anchor ID for anchors; generated "wp_" ID for waypoints.
    val id: String = "",
    val label: String = "",
    val createdAt: Timestamp = Timestamp.now(),
    val floor: Int = 0,

    // Pose in the first-anchor map coordinate frame.
    val x: Float = 0f,
    val y: Float = 0f,
    val z: Float = 0f,
    val qx: Float = 0f,
    val qy: Float = 0f,
    val qz: Float = 0f,
    val qw: Float = 1f,

    @get:PropertyName(value = "isAnchor")
    @set:PropertyName(value = "isAnchor")
    var isAnchor: Boolean = false,

    // True only for anchors that are available destinations
    @get:PropertyName(value = "isDestination")
    @set:PropertyName(value = "isDestination")
    var isDestination: Boolean = false
)

```

Σχήμα 5.2: Ορισμός της κλάσης PathStep

5.2.3 Κλάση PathConnectionType

Η enum κλάση PathConnectionType, που παρουσιάζεται στην Εικόνα 5.3, αφορά το είδος μετάβασης μεταξύ δύο κόμβων. Οι τιμές που μπορεί να λάβει είναι οι ακόλουθες:

- WALK, για απλή μετακίνηση μεταξύ κόμβων του ίδιου ορόφου
- STAIRS_UP, για μετάβαση προς ανώτερο όροφο
- STAIRS_DOWN, για μετάβαση προς χαμηλότερο όροφο
-

Για συνδέσεις ανάμεσα σε διαφορετικούς ορόφους, το είδος μετάβασης καθορίζεται από την κατεύθυνση της κίνησης. Έτσι, η μέθοδος reversed() αντιστρέφει τις τιμές STAIRS_UP και STAIRS_DOWN. Η τιμή WALK παραμένει αμετάβλητη, αφού η αμφίδρομη διάσχιση της διαδρομής δεν επηρεάζεται από την κατεύθυνση.

Για παράδειγμα, ανάμεσα στους κόμβους A και B υπάρχουν δύο πιθανές κινήσεις. Είτε από το A (όροφος 0) προς το B (όροφος 1), όπου θα χαρακτηριστεί ως STAIRS_UP, είτε από το B προς το A, δηλαδή η ίδια διαδρομή αλλά προς την αντίθετη κατεύθυνση, μέσω της συνάρτησης reversed(), θα ερμηνευθεί ως STAIRS_DOWN.

```

PathConnectionType.kt
enum class PathConnectionType {
    17 Usages
    WALK,
    12 Usages
    STAIRS_UP,
    10 Usages
    STAIRS_DOWN;

    /** Returns the equivalent movement type when traversing an edge backwards. */
    1 Usage
    fun reversed(): PathConnectionType = when (this) {
        WALK -> WALK
        STAIRS_UP -> STAIRS_DOWN
        STAIRS_DOWN -> STAIRS_UP
    }
}

```

Σχήμα 5.3: Ορισμός της κλάσης PathConnectionType

5.2.4 Κλάση PathConnection

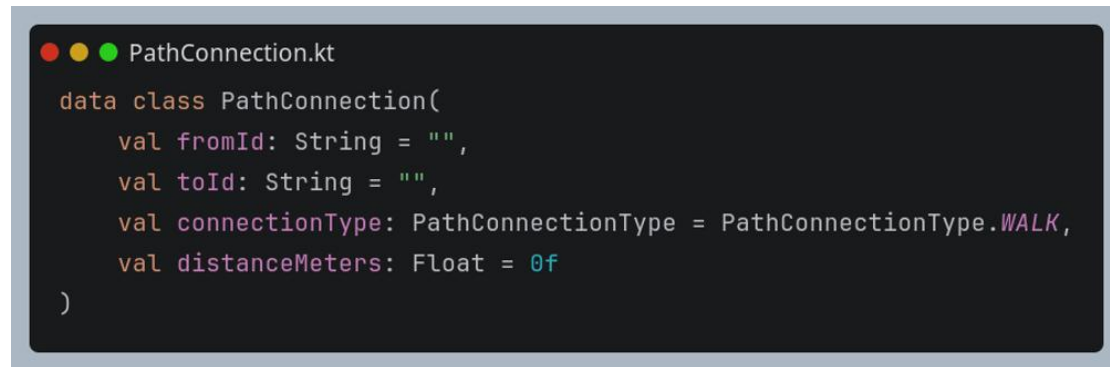
Η κλάση PathConnection, που φαίνεται στην Εικόνα 5.3 χρησιμοποιείται για την αναπαράσταση μίας ακμής του γράφου του χαρτογραφημένου κτηρίου, περιγράφοντας μία επιτρεπόμενη σύνδεση μεταξύ δύο κόμβων, δηλαδή μεταξύ δύο αντικειμένων PathStep.

Τα άκρα της σύνδεσης ορίζονται από τα πεδία fromId και toId, τα οποία περιέχουν τα μοναδικά αναγνωριστικά του αρχικού και τελικού κόμβου. Παρόλο που τα ονόματα των πεδίων δηλώνουν φορά, ο γράφος είναι μη κατευθυνόμενος, δηλαδή οι σχέσεις ανάμεσα στους κόμβους είναι αμφίδρομες. Συνοπτικά, σε μία σύνδεση με κόμβους A και B, είναι εφικτή η μετάβαση τόσο από τον κόμβο A προς τον B, όσο από τον B προς τον A. Η επιλογή αυτή επιτρέπει την πλοήγηση ανεξάρτητα κατεύθυνσης, χωρίς να χρειάζεται χαρτογράφηση και για την αντίστροφη πορεία. Κάθε σύνδεση αποθηκεύεται μία φορά και προστίθεται στη λίστα γειτνίασης

Το πεδίο connectionType ορίζει τον τρόπο μετάβασης μεταξύ των κόμβων και παίρνει μία από τις τιμές του pathConnectionType: WALK, η οποία δηλώνει κίνηση στο ίδιο επίπεδο και STAIRS_UP, STAIRS_DOWN. Οι οποίες δηλώνουν μετάβαση μέσω σκάλας προς ανώτερο ή χαμηλότερο όροφο. Κατά την αντίστροφη διάσχιση της διαδρομής ο τύπος προσαρμόζεται μέσω της συνάρτησης reversed(). Συγκεκριμένα, η τιμή WALK παραμένει ίδια, ενώ οι τιμές STAIRS_UP, STAIRS_DOWN εναλλάσσονται.

Τέλος, το πεδίο distanceMeters αποθηκεύει την απόσταση μεταξύ των δύο κόμβων σε μέτρα και αποτελεί το βάρος κάθε ακμής κατά τον υπολογισμό της διαδρομής. Η τιμή του υπολογίζεται σε προσέγγιση σύμφωνα με την ευκλείδεια απόσταση από τον τύπο:

$$= \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \quad (4.1)$$



```

PathConnection.kt

data class PathConnection(
    val fromId: String = "",
    val toId: String = "",
    val connectionType: PathConnectionType = PathConnectionType.WALK,
    val distanceMeters: Float = 0f
)

```

Σχήμα 5.4: Ορισμός της κλάσης PathConnection

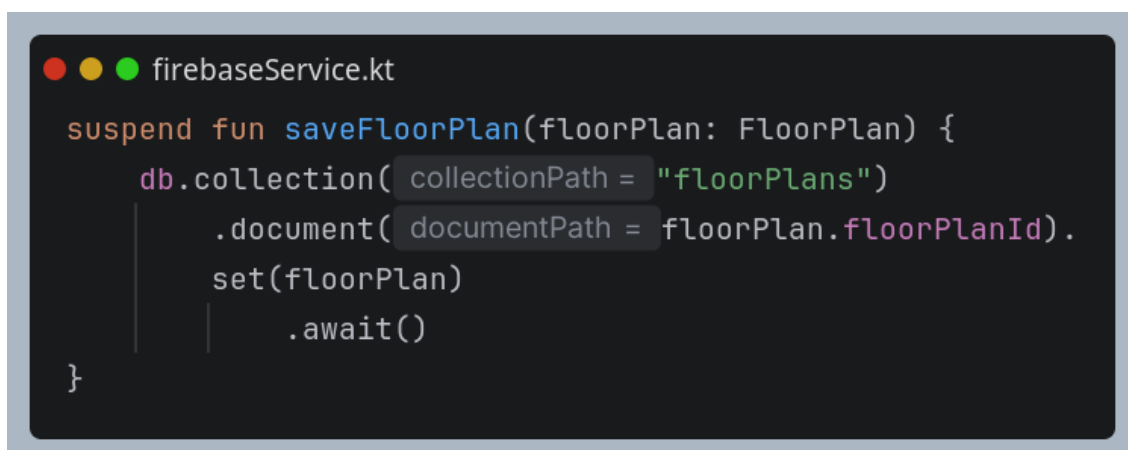
5.2.5 Αποθήκευση και ανάκτηση δεδομένων μέσω Cloud Firestore

Η αποθήκευση και ανάκτηση των δεδομένων των χαρτογραφημένων κτηρίων υλοποιείται μέσω της κλάσης FirestoreService, η οποία αποτελεί το κεντρικό σημείο πρόσβασης της εφαρμογής στο Cloud Firestore. Οι λειτουργίες της εφαρμογής χρησιμοποιούν μεθόδους της συγκεκριμένης κλάσης. Τα χαρτογραφημένα κτήρια αποθηκεύονται στη συλλογή floorPlans και κάθε έγγραφο της συλλογής αντιστοιχεί σε ένα αντικείμενο FloorPlan, ενώ ως αναγνωριστικό του εγγράφου χρησιμοποιείται το floorPlanId του αντίστοιχου κτηρίου.

Όλες οι μέθοδοι της συγκεκριμένης κλάσης χρησιμοποιούν το await(), επιτρέποντας την αναμονή ολοκλήρωσης των αντίστοιχων λειτουργιών χωρίς να δεσμεύεται το κύριο νήμα της εφαρμογής.

5.2.5.1 Μέθοδος saveFloorPlan()

Η μέθοδος saveFloorPlan(), η οποία παρουσιάζεται στο Σχήμα 5-5, χρησιμοποιεί την πράξη set(), η οποία δημιουργεί το αντίστοιχο έγγραφο εάν δεν υπάρχει ή αντικαθιστά το περιεχόμενό του εάν υπάρχει ήδη. Η συγκεκριμένη μέθοδος χρησιμοποιείται κατά την ολοκλήρωση της διαδικασίας χαρτογράφησης, όταν έχουν δημιουργηθεί οι κόμβοι και οι συνδέσεις του κτηρίου. Με αυτόν τον τρόπο, ολόκληρη η δομή του FloorPlan, συμπεριλαμβανομένων των κόμβων και των ακμών του γράφου, αποθηκεύεται σε ένα ενιαίο έγγραφο.



```

firebaseService.kt

suspend fun saveFloorPlan(floorPlan: FloorPlan) {
    db.collection(collectionPath = "floorPlans")
        .document(documentPath = floorPlan.floorPlanId)
        .set(floorPlan)
        .await()
}

```

Σχήμα 5.5: Μέθοδος saveFloorPlan()

5.2.5.2 Μέθοδος loadFloorPlan()

Η loadFloorPlan() είναι υπεύθυνη για την ανάκτηση ενός συγκεκριμένου χαρτογραφημένου κτηρίου με βάση το floorPlanId. Η μέθοδος χρησιμοποιείται κατά την έναρξη της διαδικασίας πλοήγησης, ώστε να ανακτηθούν οι απαραίτητες πληροφορίες για το κτήριο, τους κόμβους και τις συνδέσεις του γράφου.

```
● ● ● firebaseService.kt

suspend fun loadFloorPlan(floorPlanId: String): FloorPlan? {
    val document = db.collection( collectionPath = "floorPlans")
        .document( documentPath = floorPlanId)
        .get()
        .await()

    return document.toObject( valueType = FloorPlan::class.java)
        ?.copy(floorPlanId = document.id)
}
```

Σχήμα 5.6: Μέθοδος loadFloorPlan()

5.2.5.3 Μέθοδος loadAllFloorPlans()

Αντίστοιχα, η loadAllFloorPlans() ανακτά όλα τα διαθέσιμα χαρτογραφημένα κτήρια από τη συλλογή floorPlans. Η μέθοδος χρησιμοποιείται τόσο από τη διεπαφή διαχείρισης, μέσω της AdminDashboardScreen, όσο και από τη διεπαφή επιλογής κτηρίου, μέσω της DestinationPickerScreen.

```
● ● ● firebaseService.kt

suspend fun loadAllFloorPlans(): List<FloorPlan> {
    val snapshot = db.collection( collectionPath = "floorPlans")
        .get()
        .await()

    return snapshot.documents.mapNotNull { document ->
        document.toObject( valueType = FloorPlan::class.java)
            ?.copy(floorPlanId = document.id)
    }
}
```

Σχήμα 5.7: Μέθοδος loadAllFloorPlans()

5.2.5.4 Μέθοδος updateFloorPlan()

Η `updateFloorPlan()` χρησιμοποιεί την πράξη `update()`, η οποία επιτρέπει την τροποποίηση συγκεκριμένων πεδίων ενός ήδη υπάρχοντος εγγράφου χωρίς να αντικαθίσταται το σύνολο των δεδομένων του. Στην παρούσα υλοποίηση τροποποιείται το πεδίο `buildingName`, ώστε ο διαχειριστής να μπορεί να μετονομάσει ένα κτήριο από την `AdminDashboardScreen`, χωρίς να επηρεάζονται τα υπόλοιπα δεδομένα του `FloorPlan`, όπως οι κόμβοι και οι συνδέσεις του γράφου

```

firebaseService.kt

suspend fun updateFloorPlan(buildingName: String, floorPlanId: String) {
    db.collection(collectionPath = "floorPlans")
        .document(documentPath = floorPlanId)
        .update(field = "buildingName", value = buildingName)
        .await()
}

```

Σχήμα 5.8: Μέθοδος `updateFloorPlan()`

5.2.5.5 Μέθοδος deleteFloorPlan()

Τέλος, η `deleteFloorPlan()` διαγράφει το αντίστοιχο έγγραφο από τη συλλογή `floorPlans` με βάση το `floorPlanId`. Η λειτουργία αυτή χρησιμοποιείται για τη διαγραφή ενός ήδη αποθηκευμένου χάρτη από το σύστημα διαχείρισης.

```

firebaseService.kt

suspend fun deleteFloorPlan(floorPlanId: String) {
    db.collection(collectionPath = "floorPlans")
        .document(documentPath = floorPlanId)
        .delete()
        .await()
}

```

Σχήμα 5.9: Μέθοδος `deleteFloorPlan()`

5.3 Υλοποίηση χαρτογράφησης

Η χαρτογράφηση πραγματοποιείται από εξουσιοδοτημένο διαχειριστή και έχει ως στόχο τη δημιουργία ψηφιακής αναπαράστασης του εσωτερικού χώρου ενός κτηρίου. Όπως αναλύθηκε στην Ενότητα 5.2 κατά τη διάρκεια της χαρτογράφησης καταγράφονται τα σημεία ενδιαφέροντος, δηλαδή οι διαθέσιμοι προορισμοί και τα ενδιάμεσα Waypoints, τα βοηθητικά σημεία για αλλαγή ορόφου και οι μεταξύ τους συνδέσεις.

Τα δεδομένα που δημιουργούνται κατά τη χαρτογράφηση οργανώνονται σύμφωνα με το μοντέλο δεδομένων που παρουσιάστηκε στην Ενότητα 5.2. Οι κόμβοι αποθηκεύονται ως αντικείμενα PathStep, ενώ οι συνδέσεις μεταξύ τους ως αντικείμενα PathConnection. Με την ολοκλήρωση της διαδικασίας, το σύνολο αυτών των δεδομένων συγκεντρώνεται σε ένα αντικείμενο FloorPlan και αποθηκεύεται στο Cloud Firestore μέσω της FirestoreService.

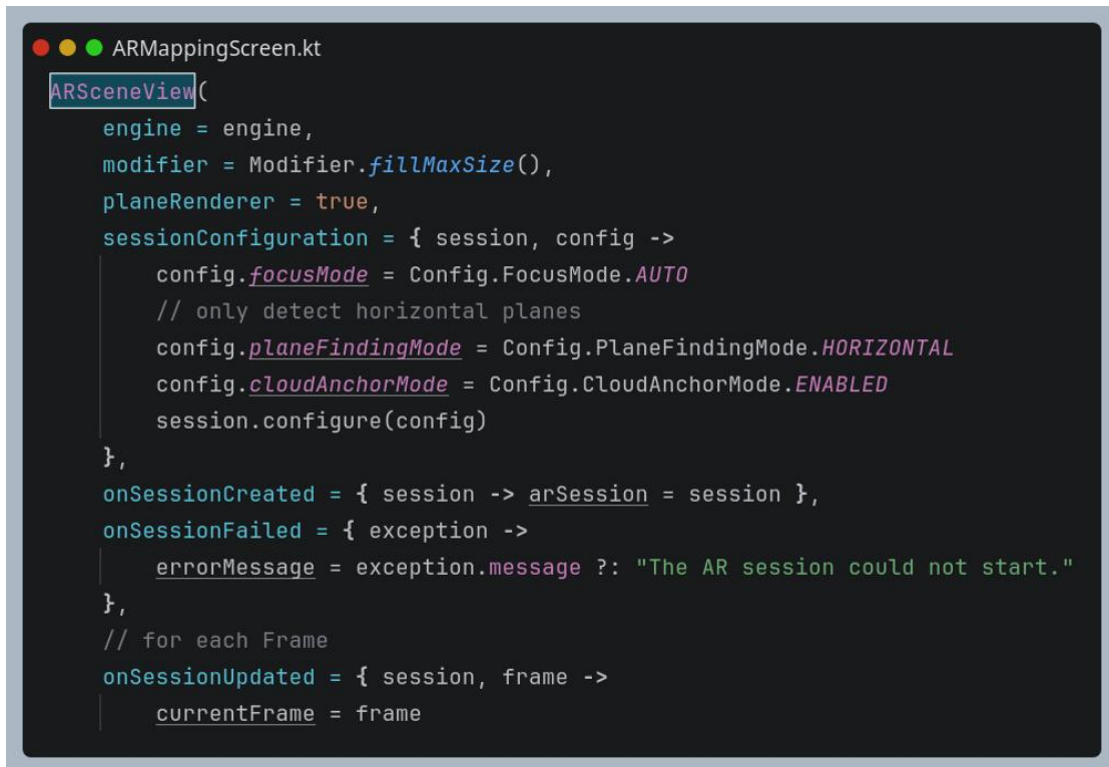
Στις ακόλουθες υπό-ενότητες περιγράφονται αναλυτικά τα επιμέρους στάδια της διαδικασίας χαρτογράφησης, από την αρχικοποίηση της συνεδρίας AR και τον εντοπισμό του χώρου έως την τοποθέτηση των αγκυρών και των Waypoints, τη δημιουργία των συνδέσεων και την τελική αποθήκευση του χάρτη.

5.3.1 Αρχικοποίηση συνεδρίας AR

Η διαδικασία της χαρτογράφησης ξεκινά με την αρχικοποίηση μίας συνεδρίας AR. Αρχικά, μία συνεδρία είναι υπεύθυνη για τη διαχείριση της κατάστασης και του κύκλου ζωής του συστήματος. Μέσω της κλάσης Session, η οποία αποτελεί το βασικό σημείο πρόσβασης στο ARCore API, ελέγχεται ο κύκλος ζωής της, δηλαδή η έναρξη, η παύση και ο τερματισμός της λειτουργίας του ARCore. Παράλληλα μέσω της συνεδρίας λαμβάνονται καρέ (Frames), τα οποία παρέχουν πρόσβαση στην εικόνα της κάμερας και την εκτιμώμενη τρέχουσα θέση και προσανατολισμό (Pose) της συσκευής στο τοπικό σύστημα συντεταγμένων της συγκεκριμένης συνεδρίας.

Για την προβολή της σκηνής και την έναρξη της συνεδρίας, χρησιμοποιείται το στοιχείο ARSceneview της βιβλιοθήκης SceneView., μέσα στο οποίο παραμετροποιείται η συνεδρία και παρουσιάζεται στο Σχήμα 5-5. Ειδικότερα, ενεργοποιείται η αυτόματη εστίαση κάμερας, η ανίχνευση επιφανειών περιορίζεται μόνο σε οριζόντιες, επιτρέπεται η χρήση Cloud Anchors. Επιπλέον, με τον ορισμό του planeRenderer σε true, ενεργοποιείται η οπτική απεικόνιση των ανιχνευμένων επιπέδων για ακριβέστερη τοποθέτηση των σημείων από τον διαχειριστή. Η παραμετροποίηση αυτή, παρουσιάζεται στο ακόλουθο απόσπασμα κώδικα.

Με τη δημιουργία της συνεδρίας, το onSessionCreated αποθηκεύει μία αναφορά της στην μεταβλητή arSession. Το onSessionUpdated ενημερώνει την μεταβλητή currentFrame με το τελευταίο διαθέσιμο καρέ. Σε περίπτωση αποτυχίας αρχικοποίησης, αποθηκεύεται το κατάλληλο μήνυμα στην μεταβλητή errorMessage.



```

ARMappingScreen.kt
ARSceneView(
    engine = engine,
    modifier = Modifier.fillMaxSize(),
    planeRenderer = true,
    sessionConfiguration = { session, config ->
        config.focusMode = Config.FocusMode.AUTO
        // only detect horizontal planes
        config.planeFindingMode = Config.PlaneFindingMode.HORIZONTAL
        config.cloudAnchorMode = Config.CloudAnchorMode.ENABLED
        session.configure(config)
    },
    onSessionCreated = { session -> arSession = session },
    onSessionFailed = { exception ->
        errorMessage = exception.message ?: "The AR session could not start."
    },
    // for each Frame
    onSessionUpdated = { session, frame ->
        currentFrame = frame
    }
)

```

Σχήμα 5.10: Αρχικοποίηση συνεδρίας μέσω ARSceneView

5.3.2 Ανίχνευση και σταθεροποίηση οριζόντιας επιφάνειας

Καθώς ο διαχειριστής κινεί τη συσκευή στον χώρο, το ARCore εκτιμά συνεχώς το Pose της κάμερας, και αναζητά συστάδες χαρακτηριστικών σημείων, ώστε να τις αντιστοιχίσει σε κοινή επίπεδη επιφάνεια. Οι ανιχνευμένες επιφάνειες αναπαρίσταται από αντικείμενα κλάσης Plane και ενημερώνονται διαρκώς, όσο το σύστημα λαμβάνει περισσότερες πληροφορίες για το περιβάλλον γύρω του. Στην εφαρμογή, η ανίχνευση περιορίζεται σε οριζόντια επίπεδα.

Για την τοποθέτηση αγκυρών χρησιμοποιούνται επίπεδα που βρίσκονται σε κατάσταση TRACKING. Αυτό σημαίνει πως παρακολουθούνται ενεργά από το ARCore μέσω της ιδιότητας trackingState και το Pose τους είναι το τρέχον [47].

Για την αποφυγή τοποθέτησης αγκυρών σε επίπεδα που μόλις ανιχνεύθηκαν, η εφαρμογή καταγραφή στη δομή planeFirstSeenAt τη στιγμή που κάθε επίπεδο παρακολουθήθηκε για πρώτη φορά. Η συνάρτηση isPlaneMature(), που παρουσιάζεται στην Εικόνα 5.6, ελέγχει αν έχουν περάσει περισσότερα από δύο δευτερόλεπτα από την καταγραφή, αν το μέγεθος του επιπέδου είναι επαρκές και αν δεν έχει συγχωνευθεί με άλλο. Αυτοί οι έλεγχοι αποτελούν κριτήρια αποδοχής του επιπέδου πριν την τοποθέτηση μίας άγκυρας.

```

ARMMappingScreen.kt
// checks if a Plane has stabilized enough
fun isPlaneMature(plane: Plane): Boolean {
    val firstSeen = planeFirstSeenAt[plane] ?: return false
    val ageMs = System.currentTimeMillis() - firstSeen
    val area = plane.extentX * plane.extentZ
    return ageMs > 2000 && area > 0.25f && plane.subsumedBy == null
}

```

Σχήμα 5.11: Συνάρτηση isPlaneMature().

5.3.3 Εκτέλεση hit-test – δημιουργία προσωρινής τοπικής άγκυρας

Όταν ο διαχειριστής αγγίζει ένα σημείο της οθόνης, η εφαρμογή εκτελεί hit test, μέσω της `frame.hitTest(e)`, ώστε να πραγματοποιηθεί η αντιστοίχιση του δυσδιάστατου σημείου της οθόνης που πατήθηκε με την πραγματική τρισδιάστατη θέση στο περιβάλλον. Επιστρέφεται για κάθε σημείο, ένα αποτέλεσμα hit (`HitResult`), το οποίο περιέχει την απόσταση του από την αρχή της ακτίνας, το Pose της τομής και το αντικείμενο `Trackable` που συναντήθηκε

Από τα αποτελέσματα, αναζητείται ένα αντικείμενο `Plane` που βρίσκεται σε κατάσταση ενεργής παρακολούθησης (`TRACKING`) και έχει τύπο `HORIZONTAL_UPWARD_FACING`, ώστε να αποκλειστούν οι κατακόρυφες επιφάνειες (`VERTICAL`) και οι οριζόντιες επιφάνειες που είναι στραμμένες προς τα κάτω, όπως οι οροφές (`HORIZONTAL_DOWNWARD_FACING`) [47]. Στη συνέχεια, εφαρμόζεται η συνάρτηση `isPlaneMature()`, με τα κριτήρια που παρουσιάστηκαν στην Ενότητα 5.3.2. Η εκτέλεση του hit-test και οι έλεγχοι επιλογής επιπέδου παρουσιάζονται στην Εικόνα 5.7

```

ARMMappingScreen.kt
// hit test
val hitResults = frame.hitTest( motionEvent = e)

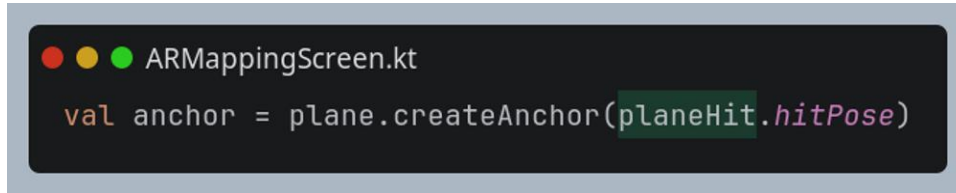
val planeHit = hitResults.firstOrNull { hit ->
    val plane = hit.trackable as? Plane ?: return@firstOrNull false

    plane.type == Plane.Type.HORIZONTAL_UPWARD_FACING &&
    plane.trackingState == TrackingState.TRACKING &&
    plane.isPoseInPolygon(hit.hitPose)
}

```

Σχήμα 5.12: Απόσπασμα της `onSingleTapUp()`, όπου ορίζεται η εκτέλεση του hit-test

Τέλος, μέσω της συνάρτησης `createAnchor()`, όπως φαίνεται στην Εικόνα 5.8 δημιουργείται μία τοπική άγκυρα, η οποία ισχύει μόνο για την τρέχουσα συνεδρία AR και είναι συνδεδεμένη με το συγκεκριμένο Plane. Καθώς το ARCore λαμβάνει περισσότερες παρατηρήσεις και βελτιώνει την εκτίμησή του για τη γεωμετρία του χώρου, το Pose της άγκυρας μπορεί να προσαρμόζεται ελαφρά, ώστε να διατηρείται η χωρική της σχέση με το επίπεδο. Στο στάδιο αυτό δεν έχει φιλοξενηθεί ακόμη η άγκυρα στο Cloud.



Σχήμα 5.13: Δημιουργία άγκυρας μέσω της `createAnchor()`

5.3.4 Σάρωση του χώρου γύρω από την άγκυρα

Αφού ο διαχειριστής εισάγει τα στοιχεία της άγκυρας (όνομα, όροφος) ξεκινά η διαδικασία σάρωσης του χώρου γύρω από αυτή και έπειτα η φιλοξενία της ως Cloud Anchor. Αρχικά, η εφαρμογή καθοδηγεί τον χρήστη να σαρώσει την περιοχή γύρω από την τοποθεσία που θέλει να βάλει μία άγκυρα, διατηρώντας την κάμερα του κινητού στραμμένη προς την περιοχή της άγκυρας. Γύρω από την άγκυρα δημιουργείται ένας εικονικός κύκλος ακτίνας 25 εκατοστών, ο οποίος χωρίζεται σε δέκα τμήματα. Κάθε τμήμα αναπαρίσταται από ένα αντικείμενο `ScanSegment`, ενώ η θέση τους ορίζεται στο οριζόντιο επίπεδο με κέντρο τη θέση της άγκυρας. Στόχος είναι η παρατήρηση της άγκυρας από διαφορετικές κατευθύνσεις και συνεπώς, η συλλογή περισσότερων οπτικών χαρακτηριστικών πριν τη φιλοξενία της.

Εφόσον η κάμερα είναι στραμμένη προς την άγκυρα, υπολογίζεται η γωνιακή θέση της. Η γωνία μετατρέπεται σε δείκτη από το 0 έως το 9, για τον προσδιορισμό του αντίστοιχου τμήματος του κύκλου. Για κάθε νέο τμήμα, πραγματοποιείται ένα `hit-test`. Από το αποτέλεσμα του, καταγράφεται το ύψος και αποθηκεύεται στη λίστα `pendingAnchorYSamples`, ώστε να συγκεντρωθούν μετρήσεις του επιπέδου από διαφορετικές οπτικές γωνίες.

Μετά την ολοκλήρωση και των δέκα τμημάτων, καλείται η συνάρτηση `refinePendingAnchorPose()`, η οποία χρησιμοποιεί τις μετρήσεις ύψους που συγκεντρώθηκαν για να υπολογίσει μία πιο σταθερή εκτίμηση της κατακόρυφης θέσης της άγκυρας. Αν υπάρχουν τουλάχιστον πέντε μετρήσεις, επιλέγεται η μεσαία τιμή της ταξινομημένης συλλογής. Το διορθωμένο Pose μεταβιβάζεται στη `hostPendingAnchor()`, η οποία ξεκινά τη διαδικασία φιλοξενίας. Αν δεν έχουν συγκεντρωθεί αρκετές μετρήσεις, η συνάρτηση επιστρέφει `null` και για την φιλοξενία διατηρείται το αρχικό Pose της τοπικής άγκυρας.

```

ARMappingScreen.kt
fun refinePendingAnchorPose(): Pose? {
    val anchor = pendingHostAnchor ?: return null
    if (pendingAnchorYSamples.size < 5) return null
    val medianY = pendingAnchorYSamples.sorted()[pendingAnchorYSamples.size / 2]
    return Pose(
        translation = floatArrayOf(anchor.pose.tx(), medianY, anchor.pose.tz()),
        rotation = anchor.pose.rotationQuaternion
    )
}

```

Σχήμα 5.14: Συνάρτηση refinePendingAnchorPose()

5.3.5 ARCore API και εξουσιοδότηση

Για τη χρήση Cloud Anchors απαιτείται η ενεργοποίηση του ARCore API σε έργο Google Cloud. Υπάρχουν δύο τρόποι εξουσιοδότησης. Ο πρώτος είναι μέσω API key, όπου η διάρκεια ζωής των Cloud Anchors στον διακομιστή περιορίζεται στην μία ημέρα. Αντίθετα, χρησιμοποιώντας την keyless μέθοδο μέσω OAuth 2.0, όπου η εφαρμογή αναγνωρίζεται από το package name και το SHA-1, η διάρκεια ζωής των Cloud Anchors έχουν διάρκεια έως 365 μέρες. Στην παρούσα εργασία χρησιμοποιείται η δεύτερη μέθοδος, με τη διάρκεια ζωής να ορίζεται στο μέγιστο, δηλαδή 365 ημέρες, ώστε οι άγκυρες να μπορούν να επιλύονται σε μελλοντικές συνεδρίες [36].

```

ARMappingScreen.kt
nodeToHost.host(
    session = session,
    ttlDays = 365
) { cloudAnchorId, sta

```

Σχήμα 5.15: Ορισμός διάρκειας ζωής ενός Cloud Anchor

5.3.6 Φιλοξενία Cloud Anchor και δημιουργία κόμβου

Αφού ολοκληρωθεί η σάρωση του χώρου γύρω από την άγκυρα, ξεκινά η διαδικασία φιλοξενίας της τοπικής άγκυρας στην υπηρεσία Cloud Anchors του ARCore API. Για αυτό καλείται η συνάρτηση host() του αντικειμένου CloudAnchorNode της βιβλιοθήκης SceneView.

Η κατάσταση TASK_IN_PROGRESS δηλώνει πως η φιλοξενία βρίσκεται σε εξέλιξη. Μόλις επιστραφεί η κατάσταση SUCCESS μαζί με έγκυρο μοναδικό αναγνωριστικό Cloud Anchor Id, η άγκυρα θεωρείται πως φιλοξενήθηκε επιτυχώς. Σε περίπτωση αποτυχίας ο κόμβος δεν προστίθεται στον γράφο και ο διαχειριστής ενημερώνεται.

Πριν τη δημιουργία του κόμβου, το Pose της τοπικής άγκυρας μετατρέπεται από το σύστημα συντεταγμένων της τρέχουσας συνεδρίας AR στο κοινό σύστημα συντεταγμένων του χάρτη. Καθώς η πρώτη άγκυρα ορίζει την αρχή αυτού του συστήματος, το Pose αυτής αποθηκεύεται ως Pose.IDENTITY, δηλαδή με μηδανική μετατόπιση και μοναδιαία περιστροφή [47].

Για τις επόμενες προς φιλοξενία άγκυρες, εφαρμόζεται η συνάρτηση `inverse()`, η οποία υπολογίζει τον αντίστροφο μετασχηματισμό του Pose της πρώτης άγκυρας και η `compose()`, η οποία εφαρμόζει τον μετασχηματισμό αυτό στο Pose της νέας άγκυρας. Έτσι, το αποτέλεσμα είναι η θέση και ο προσανατολισμός της νέας άγκυρας σε σχέση με την πρώτη και όχι ως προς το σύστημα συντεταγμένων της τρέχουσας συνεδρίας AR. Η μετατροπή κρίνεται απαραίτητη, καθώς κάθε νέα συνεδρία δημιουργεί διαφορετικό τοπικό σύστημα συντεταγμένων.

Με βάση το Pose δημιουργείται αντικείμενο `PathStep`. Σε αυτό, αποθηκεύονται μεταξύ άλλων στοιχείων οι συντεταγμένες θέσεις (x, y, z), καθώς και τα τέσσερα στοιχεία του τετραδονίου περιστροφής qx, qy, qz, qw . Στη συνέχεια ο κόμβος προστίθεται στη λίστα `mappedNodes`.

Καθώς η πρώτη άγκυρα αποτελεί τον αρχικό κόμβο του γράφου, δεν συνδέεται με κάποιο προηγούμενο κόμβο. Αντίθετα, κάθε επόμενη άγκυρα συνδέεται με τον τρέχοντα ενεργό κόμβο, το αναγνωριστικό του οποίου διατηρείται στη μεταβλητή `activeNodeId`. Ο ενεργός κόμβος χαρακτηρίζει το τελευταίο σημείο που προστέθηκε στην διαδρομή. Ωστόσο, σε περίπτωση διακλάδωσης, μπορεί να είναι προηγούμενος κόμβος, ο οποίος επιλέχθηκε ως νέο σημείο εκκίνησης από τον χειριστή, όπως θα αναλυθεί στην συνέχεια.

Η σύνδεση μεταξύ δύο κόμβων αναπαρίσταται από ένα αντικείμενο `PathConnection`, όπου αποθηκεύονται τα αναγνωριστικά των δύο κόμβων, ο τύπος της σύνδεσης και η απόσταση ανάμεσα τους, η οποία υπολογίζεται μέσω της ευκλείδειας απόστασης για τρία σημεία και αποτελεί το βάρος της ακμής, όπως παρουσιάστηκε στην ενότητα περιγραφής της κλάσης `PathConnection`.

5.3.7 Προσθήκη ενδιάμεσων σημείων πλοήγησης (Waypoints)

Πέρα από τα Cloud Anchors, ο χειριστής μπορεί να προσθέσει Waypoints, ενδιάμεσα δηλαδή σημεία πλοήγησης. Αυτά αποτυπώνουν την πορεία μίας επιτρεπτής διαδρομής και αποτελούν κόμβο του γράφου. Στις οδηγίες της εφαρμογής, προτείνεται να τοποθετούνται ανά δύο με τρία μέτρα, αλλά και σε σημεία όπου αλλάζει η κατεύθυνση της διαδρομής, όπως σε στροφές ή διακλαδώσεις.

Για την προσθήκη ενός Waypoint, πρέπει να έχει φιλοξενηθεί ήδη κάποια άγκυρα, ώστε να είναι διαθέσιμα το σύστημα αναφοράς του χάρτη και ενεργός κόμβος με τον οποίο θα συνδεθεί το νέο σημείο. Σε αντίθεση με την τοποθέτηση άγκυρας, η προσθήκη Waypoint δεν απαιτεί το άγγιγμα ενός σημείου της οθόνης. Πατώντας το κουμπί «Drop Waypoint» εκτελείται η συνάρτηση `dropWaypoint()`, η οποία εκτελεί hit-test με ακτίνα που ξεκινά από τη θέση της κάμερας της συσκευής και κατευθύνεται κατακόρυφα προς τα κάτω. Από τα αποτελέσματα επιλέγεται επίπεδο τύπου HORIZONTAL_UPWARD_FACING, το οποίο βρίσκεται σε κατάσταση TRACKING.

Οι συντεταγμένες x και z λαμβάνονται από το Pose της κάμερας, ενώ η κατακόρυφη συντεταγμένη y προκύπτει από το αποτέλεσμα του hit-test. Επομένως, όταν εντοπίζεται κατάλληλο επίπεδο, το Waypoint αντιστοιχεί στην κατακόρυφη προβολή της κάμερας σε αυτό. Αν δε βρεθεί κατάλληλο αποτέλεσμα, χρησιμοποιείται ως εκτίμηση ύψους η κατακόρυφη συντεταγμένη y του ενεργού κόμβου. Το απόσπασμα της Εικόνας 5.12 αποτελεί τμήμα της συνάρτησης `dropWaypoint()` και παρουσιάζει τον υπολογισμό θέσης του νέου σημείου.

Για το Waypoint δημιουργείται αντικείμενο τύπου PathStep. Πριν την προσθήκη του στον γράφο, υπολογίζεται η ευκλείδεια απόσταση του από τον ενεργό κόμβο. Αν η απόσταση δεν υπερβαίνει τα 20 εκατοστά, η προσθήκη απορρίπτεται, ώστε να αποφεύγεται η δημιουργία πολλαπλών σχεδόν ίδιων σημείων.

Με την αποδοχή του Waypoint, δημιουργείται ένα αντικείμενο PathConnection, που συνδέει τον νέο με τον ενεργό κόμβο. Η σύνδεση χαρακτηρίζεται ως WALK. Έπειτα, ο κόμβος προστίθεται στην λίστα mappedNodes, η σύνδεση μεταξύ τους στη λίστα connections. Το activeNodeId ενημερώνεται με το αναγνωριστικό του Waypoint, το οποίο καθίσταται ο νέος ενεργός κόμβος.

5.3.8 Δημιουργία διακλαδώσεων

Σε πραγματικές συνθήκες, οι διαθέσιμες διαδρομές δεν σχηματίζουν πάντοτε γραμμική ακολουθία. Υπάρχουν διασταυρώσεις σε διαδρόμους, κόμβοι που μπορεί να συνδέονται με περισσότερες από μία διαδρομές. Για τον λόγο αυτό, υποστηρίζεται η δυνατότητα χαρτογράφησης σε διακλαδώσεις, επιτρέποντας στον διαχειριστή να επιλέξει έναν ήδη χαρτογραφημένο κόμβο ως σημείο αναφοράς της διακλάδωσης.

Η μεταβλητή activeNodeId αποθηκεύει το αναγνωριστικό του ενεργού κόμβου, του σημείου, δηλαδή, από το οποίο θα συνεχισθεί η χαρτογράφηση. Κανονικά, κάθε νέος συνδέεται με τον ενεργό κόμβο και στη συνέχεια γίνεται και ο ίδιος ενεργός.

Για τη δημιουργία διακλάδωσης, ο διαχειριστής επιλέγει τη λειτουργία έναρξης νέου κλάδου, δηλαδή του κουμπιού «Start branch from existing point» όπου εμφανίζεται κατάλογος με τους κόμβους που έχουν ήδη χαρτογραφηθεί. Ο επιλεγμένος κόμβος αναπαρίσταται από την μεταβλητή candidateNode. Αρχικά, ανακτάται το τρέχον καρέ και το Pose του κόμβου. Στη συνέχεια υπολογίζονται οι διαφορές dx , dz μεταξύ των συντεταγμένων της κάμερας και του επιλεγμένου κόμβου, στο σύστημα συντεταγμένων της συνεδρίας AR. Από αυτές προκύπτει η horizontalDistance. Εφόσον η απόσταση της κάμερας είναι έως 1.5 μέτρο από το σημείο, η επιλογή γίνεται αποδεκτή.

Αφού ο διαχειριστής επιλέξει τον κόμβο στον οποίο επιθυμεί να δημιουργήσει διακλάδωση, το αναγνωριστικό του κόμβου ανατίθεται στην μεταβλητή activeNodeId και ο τρέχων όροφος υιοθετείται από αυτόν του προηγούμενου σημείου. Συνεπώς, κάθε επόμενος κόμβος συνδέεται πλέον με τον επιλεγμένο κόμβο.

Για παράδειγμα, ο διαχειριστής μπορεί να χαρτογραφήσει ένα διάδρομο μέχρι το τέλος του και έπειτα να επιστρέψει σε ένα Waypoint, το οποίο βρίσκεται σε διασταύρωση. Αφού το επιλέξει ως ενεργό κόμβο, συνεχίζει την χαρτογράφηση του συγκεκριμένου διαδρόμου.

```

ARMappingScreen.kt
val frame = currentFrame
val selectedPose = nodePoses[candidateNode.id]
if (
    frame == null ||
    frame.camera.trackingState != TrackingState.TRACKING ||
    selectedPose == null
) {
    coroutineScope.launch {
        snackbarHostState.showSnackbar(
            message = "Tracking is not ready. Scan the area and try again."
        )
    }
    return@TextButton
}
val cameraPose = frame.camera.pose
val dx = cameraPose.tx() - selectedPose.tx()
val dz = cameraPose.tz() - selectedPose.tz()
val horizontalDistance = sqrt(x = dx * dx + dz * dz)
if (horizontalDistance > 1.5f) {
    coroutineScope.launch {
        snackbarHostState.showSnackbar(
            message = "Move within 1.5 metres of '${candidateNode.label}' " +
                "before starting this branch."
        )
    }
    return@TextButton
}
activeNodeId = candidateNode.id
currentMappingFloor = candidateNode.floor
currentAnchorFloor = candidateNode.floor.toString()
showBranchDialog = false

```

Σχήμα 5.16: Έλεγχοι αποδοχής και ορισμός υπάρχοντος κόμβου ως σημείο έναρξης της νέας διακλάδωσης

5.3.9 Αλλαγή ορόφου

Η αλλαγή ορόφου κατοχυρώνεται μέσω δύο αγκυρών: μία στην αρχή και μία στο τέλος της σκάλας. Μεταξύ τους, δημιουργείται ακμή τύπου STAIRS_UP ή STAIRS_DOWN, ανάλογα με την κατεύθυνση της μετάβασης.

Για την έναρξη της διαδικασίας, αρχικά, ο ενεργός κόμβος πρέπει να είναι ένα Cloud Anchor, το οποίο έχει τοποθετηθεί στην αρχή της σκάλας, μέσω του κουμπιού «Stairs». Οι πληροφορίες σχετικά με την κατεύθυνση και τον όροφο προορισμού αποθηκεύονται στο αντικείμενο pendingStairMapping, μαζί με το αναγνωριστικό της άγκυρας εισόδου.

Όταν ο διαχειριστής φτάσει στον νέο όροφο, τοποθετεί μία ακόμη άγκυρα στο τέλος της σκάλας. Με την επιτυχή φιλοξενία της, δημιουργείται ένα PathConnection από την άγκυρα αρχής προς την άγκυρα τέλους, όπου ο τύπος σύνδεσης περιέχει την κατεύθυνση που καθορίστηκε στο pendingStairMapping.

Τέλος, η άγκυρα τέλους αποτελεί πλέον τον ενεργό κόμβο και αφού ενημερωθεί ο τρέχων όροφος, η χαρτογράφηση συνεχίζεται κανονικά.

```
ARMappingScreen.kt

pendingStairMapping = PendingStairMapping(
    direction = selectedStairDirection,
    entranceAnchorId = entranceAnchorId,
    entranceWasDestination =
        entranceAnchor.isDestination,
    startFloor = startFloor,
    destinationFloor = targetFloor,
    number = stairNumber
)

mappedNodes = mappedNodes.map { candidateNode ->
    if (candidateNode.id == entranceAnchorId) {
        candidateNode.copy(isDestination = false)
    } else {
        candidateNode
    }
}
```

Σχήμα 5.17: Δημιουργία αντικειμένου pendingStairMapping

5.3.10 Έλεγχος εγκυρότητας γράφου

Με την ολοκλήρωση της χαρτογράφησης και πριν την αποθήκευση του χάρτη, πραγματοποιείται έλεγχος εγκυρότητας του, μέσω της συνάρτησης `graphValidationError()`. Αυτή είναι υπεύθυνη για τον έλεγχο της σωστής δομής του γράφου.

Αρχικά, ελέγχεται η ύπαρξη έγκυρης αρχικής άγκυρας, καθώς και η καταχώρηση τουλάχιστον δύο αγκυρών στον χάρτη. Επιπλέον, επειδή μόνο οι άγκυρες επιτρέπεται να αναπαριστούν προορισμούς, επιβεβαιώνεται πως δεν έχει δηλωθεί ως προορισμός κάποιο Waypoint. Όσον αφορά τη δομή του γράφου, ελέγχεται η σωστή καταχώρηση των κόμβων και των ακμών και η δυνατότητα πρόσβασης σε όλους τους κόμβους από την πρώτη άγκυρα. Στο παρακάτω απόσπασμα κώδικα, παραθέτονται ενδεικτικά ορισμένοι από τους ελέγχους που πραγματοποιούνται.

```

ARMMappingScreen.kt
if (mappedNodes.count { it.isAnchor } < 2) {
    return "Place at least two Cloud Anchors: the entrance and one additional anchor."
}
if (mappedNodes.any { it.isDestination && !it.isAnchor }) {
    return "Only Cloud Anchors can be marked as navigation destinations."
}

```

Σχήμα 5.18: Ενδεικτικό απόσπασμα της συνάρτησης graphValidationError(), για τους ελέγχους εγκυρότητας του γράφου

5.3.11 Αποθήκευση χαρτογραφημένου κτηρίου (FloorPlan)

Αν δεν εντοπισθούν σφάλματα στην γράφο προς αποθήκευση, δημιουργείται μοναδικό αναγνωριστικό για τον χάρτη και δημιουργείται το αντικείμενο FloorPlan, που αντιστοιχεί στο τρέχον χαρτογραφημένο κτήριο. Εκεί, αποθηκεύονται το όνομα του κτηρίου, το αναγνωριστικό της πρώτης άγκυρας, όπως και οι κόμβοι και οι ακμές του γράφου, όπως φαίνεται στην Εικόνα 5.16

Έπειτα, το αντικείμενο FloorPlan αποθηκεύεται σε έγγραφο της συλλογής floorPlans στο Cloud Firestore, μέσω της συνάρτησης saveFloorPlan(), μέσω του αναγνωριστικού του εγγράφου floorPlanId. Η συνάρτηση παρουσιάζεται στην Εικόνα 5.17. Κατά την αποθήκευση, η μεταβλητή isSavingFloorPlan έχει τιμή true, ώστε να απενεργοποιείται το κουμπί αποθήκευσης χάρτη.

Με την επιτυχή αποθήκευση του κτηρίου, εμφανίζεται μήνυμα επιβεβαίωσης και καλείται η συνάρτηση onMappingFinished(), ολοκληρώνοντας τη διαδικασία της χαρτογράφησης. Σε περίπτωση αποτυχίας αποθήκευσης, πέρα από το μήνυμα σφάλματος, εξασφαλίζεται η παραμονή των στοιχείων του γράφου στην τρέχουσα κατάσταση οθόνης, επιτρέποντας στον χειριστή να ξανά προσπαθήσει, χωρίς να χρειαστεί εκ νέου χαρτογράφηση

```

firebaseService.kt
suspend fun saveFloorPlan(floorPlan: FloorPlan) {
    db.collection( collectionPath = "floorPlans")
        .document( documentPath = floorPlan.floorPlanId).
        set(floorPlan)
        .await()
}

```

Σχήμα 5.19: Αποθήκευση του χαρτογραφημένου κτηρίου στη συλλογή floorPlans, μέσω της συνάρτησης saveFloorPlan()

5.4 Υλοποίηση Πλοήγησης

Αφού ο χρήστης επιλέξει ένα κτήριο και τον συγκεκριμένο προορισμό που επιθυμεί, ξεκινά η διαδικασία πλοήγησής του προς αυτόν. Η οθόνη ARNavigationScreen λαμβάνει ως παραμέτρους το μοναδικό αναγνωριστικό του χάρτη του επιλεγμένου κτηρίου (floorPlanId) και του επιλεγμένου προορισμού (destinationAnchorId). Σύμφωνα με αυτά, ανακτάται από το Cloud Firestore το αντικείμενο FloorPlan, το οποίο αναπαριστά τον χάρτη του συγκεκριμένου κτηρίου.

Μετά τη φόρτωση του χάρτη, αρχικοποιείται μία συνεδρία ARCore και ξεκινά προσπάθεια επίλυσης (resolve) των Cloud Anchors που τοποθετήθηκαν ως εικονικά αντικείμενα στους διαθέσιμους προορισμούς κατά τη διαδικασία χαρτογράφησης. Όταν επιλυθεί τουλάχιστον ένα, επιλέγεται ως ενεργό σημείο αναφοράς. Με βάση την γνωστή θέση του Cloud Anchor στον χάρτη και τη θέση του στην τρέχουσα συνεδρία ARCore, το Pose της κάμερας μετατρέπεται από το σύστημα συντεταγμένων της τρέχουσας συνεδρίας στο κοινό σύστημα συντεταγμένων του χάρτη και υπολογίζεται η θέση και ο προσανατολισμός της συσκευής, και κατ' επέκταση του χρήστη, μέσα στο κτήριο.

Στη συνέχεια, επιλέγεται ο πλησιέστερος κατάλληλος κόμβος του γράφου και εφαρμόζεται ο αλγόριθμος A* για τον υπολογισμό της συντομότερης διαδρομής προς τον επιλεγμένο προορισμό. Κατά την κίνηση του χρήστη στον χώρο, ενημερώνεται διαρκώς η θέση του και ελέγχεται η απόσταση του από το επόμενο σημείο της διαδρομής. Με βάση τις αποστάσεις και τον προσανατολισμό του χρήστη, παράγονται σύντομες, κατάλληλες φωνητικές οδηγίες. Μοτίβα δόνησης χρησιμοποιούνται για επιπλέον ενημέρωση του χρήστη σχετικά με κρίσιμα γεγονότα, όπως στροφές, σκάλες, απόκλιση από την πορεία και άφιξη στον προορισμό. Με την άφιξη του χρήστη στον προορισμό, ανακοινώνεται η άφιξη του και ολοκληρώνεται η διαδικασία της πλοήγησης.

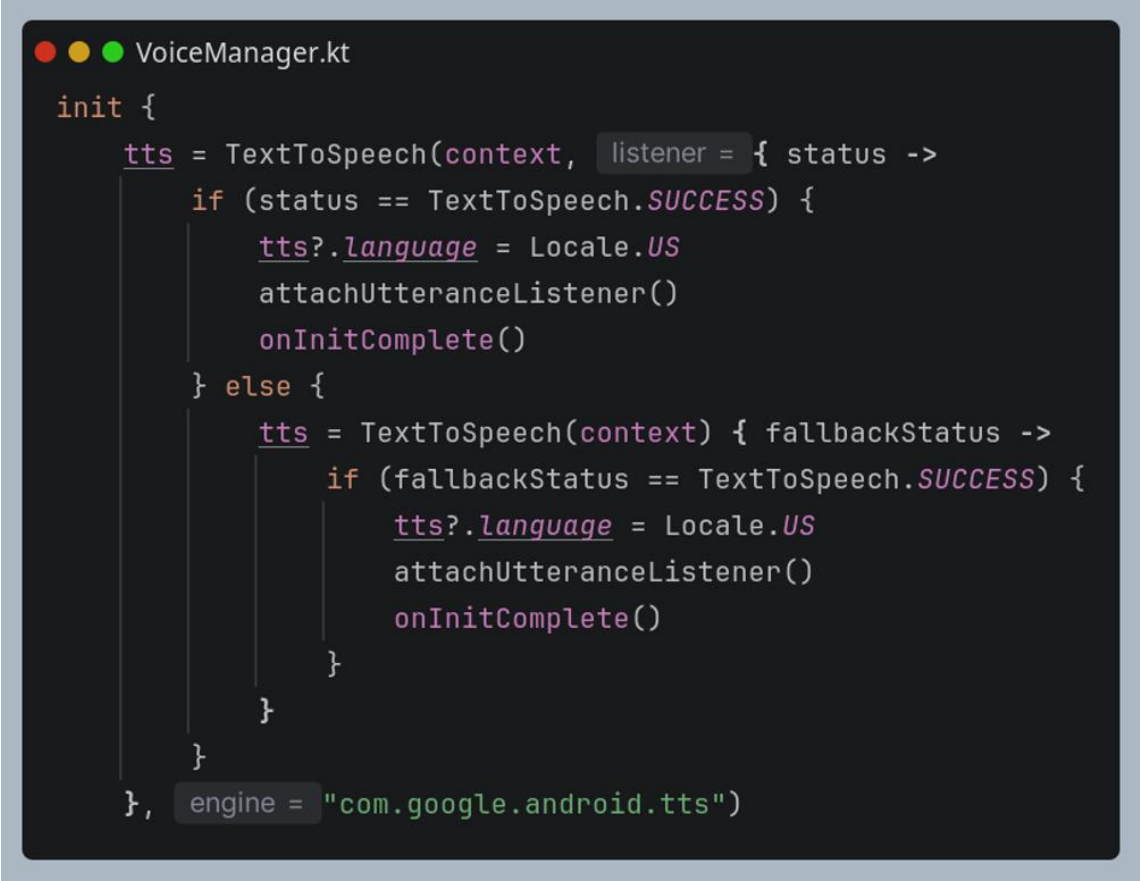
5.4.1 Προσβάσιμη αλληλεπίδραση και διαχείριση διαλόγου

Η φωνητική αλληλεπίδραση αποτελεί το βασικό μέσο αλληλεπίδρασης μεταξύ του χρήστη και της εφαρμογής. Περιλαμβάνει λειτουργίες όπως η αναγνώριση προφορικών απαντήσεων του χρήστη, εκφώνηση ερωτήσεων, ενημερωτικών μηνυμάτων και είναι αναλαμβάνει την πλήρη καθοδήγηση του χρήστη στον χώρο, εξαλείφοντας την εξάρτηση του χρήστη από οπτικά στοιχεία. Η υλοποίηση της βασίζεται στις κλάσεις VoiceManager και VoiceViewModel.

5.4.1.1 Κλάση VoiceManager

Η VoiceManager περιέχει τις λειτουργίες σύνθεσης και αναγνώρισης ομιλίας. Αρχικά, δημιουργείται ένα αντικείμενο TextToSpeech, το οποίο χρησιμοποιείται για τη μετατροπή των μηνυμάτων της εφαρμογής σε ομιλία και ένα αντικείμενο SpeechRecognizer, το οποίο συμβάλλει στην καταγραφή και μετατροπή ομιλίας σε κείμενο. VoiceViewModel. Στην VoiceManager ορίζονται τέσσερα callbacks. Συγκεκριμένα, το onInitComplete είναι υπεύθυνο για την ενημέρωση επιτυχής αρχικοποίησης της σύνθεσης ομιλίας, το onSpeechResult διαθέτει το αναγνωρισμένο κείμενο, το onListeningStateChange ενημερώνει για την κατάσταση ακρόασης και το onSpeechError διαθέτει τα σφάλματα αναγνώρισης κατά την ακρόαση.

Στο Σχήμα 5-15 παρουσιάζεται η αρχικοποίηση του TextToSpeech, όπου επιχειρείται χρήση της μηχανής σύνθεσης ομιλίας της Google, μέσω του com.google.android.tts. Σε περίπτωση αποτυχίας, επιχειρείται η χρήση της προεπιλεγμένη μηχανή της συσκευής. Μετά την αρχικοποίηση, ορίζονται ως γλώσσα τα Αγγλικά, μέσω του Locale.US και καλείται η attachUtteranceListener(), που συνδέει τον μηχανισμό παρακολούθησης των εκφωνήσεων και ενεργοποιείται το callback onInitComplete().



```

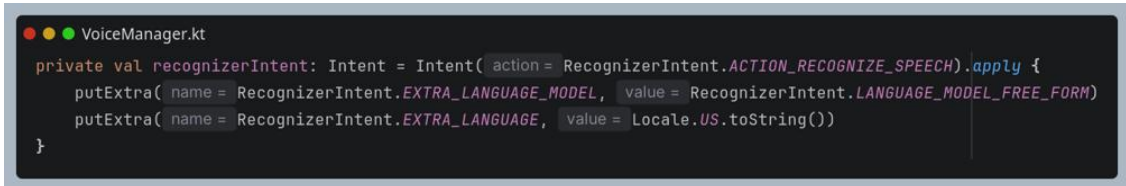
VoiceManager.kt

init {
    tts = TextToSpeech(context, listener = { status ->
        if (status == TextToSpeech.SUCCESS) {
            tts?.language = Locale.US
            attachUtteranceListener()
            onInitComplete()
        } else {
            tts = TextToSpeech(context) { fallbackStatus ->
                if (fallbackStatus == TextToSpeech.SUCCESS) {
                    tts?.language = Locale.US
                    attachUtteranceListener()
                    onInitComplete()
                }
            }
        }
    })
    engine = "com.google.android.tts"
}

```

Σχήμα 5.20: Αρχικοποίηση TextToSpeech()

Για την αναγνώριση ομιλίας αρχικά ελέγχεται η ύπαρξη της υπηρεσίας στη συσκευή, μέσω του `SpeechRecognizer.isRecognitionAvailable()`. Αν υπάρχει, δημιουργείται το αντικείμενο `SpeechRecognizer`, που συνδέεται με ένα `RecognitionListener`. Οι παράμετροι ορίζονται στο αντικείμενο `recognizerIntent`, όπου όπως φαίνεται στο απόσπασμα κώδικα του σχήματος 5-16, ως γλωσσικό μοντέλο επιλέγεται το `LANGUAGE_MODEL_FREE_FORM` για την απάντηση του χρήστη σε φυσική γλώσσα και ως γλώσσα αναγνώρισης τα αγγλικά Ηνωμένων Πολιτειών.



```

VoiceManager.kt

private val recognizerIntent: Intent = Intent(action = RecognizerIntent.ACTION_RECOGNIZE_SPEECH).apply {
    putExtra(name = RecognizerIntent.EXTRA_LANGUAGE_MODEL, value = RecognizerIntent.LANGUAGE_MODEL_FREE_FORM)
    putExtra(name = RecognizerIntent.EXTRA_LANGUAGE, value = Locale.US.toString())
}

```

Σχήμα 5.21: Παρουσίαση recognizerIntent

Όταν ο μηχανισμός `RecognitionListener` είναι έτοιμος να δεχθεί ομιλία, η συνάρτηση `onReadyForSpeech()` καλεί το `onListeningStateChanged(true)`, όπως φαίνεται στο Σχήμα 5-17. Τα αποτελέσματα της ακρόασης είναι διαθέσιμα στη μέθοδο `onResults()`. Από την λίστα των αποτελεσμάτων, επιλέγεται το πρώτο αποτέλεσμα και μεταφέρει στην `VoiceViewModel`, μέσω του `onSpeechResult`.



```

VoiceManager.kt
override fun onResults(results: Bundle?) {
    onListeningStateChanged(false)
    val matches = results?.getStringArrayList(key = SpeechRecognizer.RESULTS_RECOGNITION)
    if (!matches.isNullOrEmpty()) {
        onSpeechResult(matches[0])
    } else {
        onSpeechError(VoiceError.NO_MATCH)
    }
}

```

Σχήμα 5.22: Συνάρτηση onResults()

Η εκφώνηση των μηνυμάτων από την εφαρμογή πραγματοποιείται μέσω της συνάρτησης `speak()`, η οποία δέχεται το κείμενο προς εκφώνηση και την παράμετρο `interrupt`. Όταν αυτή είναι αληθής, χρησιμοποιείται το `QUEUE_FLUSH`, ώστε να αντικατασταθούν τα παλιά μηνύματα από το καινούριο. Με την επιλογή `QUEUE_ADD` προστίθεται στην ουρά. Η συνάρτηση επίσης δέχεται το προαιρετικό callback `onDone`, μέσω του οποίου μπορεί να εκτελεστεί μία ενέργεια, για παράδειγμα η σύνδεση μίας ερώτησης από την εφαρμογή με την έναρξη αναγνώρισης της απάντησης. Κάθε εκφώνηση λαμβάνει μοναδικό αναγνωριστικό. Απόσπασμα της συνάρτησης `speak()` παραθέται κάτω.

Μόλις ολοκληρωθεί μία εκφώνηση, η μέθοδος `onDone()` του `Listener` καλεί την `notifyUtteranceFinished()`. Η συνάρτηση αφού εντοπίσει το callback που αντιστοιχεί στο αναγνωριστικό της εκφώνησης, το αφαιρεί από τη δομή `utteranceCallbacks` και το εκτελεί στο κύριο νήμα, ώστε η ολοκλήρωση μίας ερώτησης να συνδεθεί με την αρχή αναγνώρισης της απάντησης.

Η έναρξη ακρόασης πραγματοποιείται μέσω της `startListening()`. Μέσω των παραμέτρων του `recognizerIntent` ενεργοποιείται η συνάρτηση. Αν η εφαρμογή δεν έχει άδεια πρόσβασης στο μικρόφωνο, επιστρέφεται σφάλμα `NO_PERMISSION`, ενώ αν ο μηχανισμός χρησιμοποιείται ήδη επιστρέφεται `RECOGNIZER_BUSY`. Η `cancelListening()` ακυρώνει την τρέχουσα διαδικασία αναγνώρισης ομιλίας και χρησιμοποιείται όταν ο χρήστης εγκαταλείπει τη διαδικασία φωνητικής επιλογής.

Διακοπή όλων των φωνητικών λειτουργιών πραγματοποιείται μέσω της συνάρτησης `stopAll()` καθαρίζονται τα callbacks, τερματίζεται η εκφώνηση και ακυρώνεται η αναγνώριση, διατηρώντας τα αντικείμενα διαθέσιμα για μελλοντική χρήση. Ωστόσο, η τελική αποδέσμευση των μηχανισμών ομιλίας πραγματοποιείται μέσω της `destroy()`, η οποία καλεί την `stopAll()` και επιπλέον τερματίζει το `TextToSpeech` και το `SpeechRecognizer`.

```

VoiceManager.kt
fun destroy() {
    stopAll()
    tts?.shutdown()
    speechRecognizer?.destroy()
    tts = null
    speechRecognizer = null
    mainHandler.removeCallbacksAndMessages( token = null)
}

```

Σχήμα 5.23: Συνάρτηση destroy()

5.4.1.2 Κλάση VoiceViewModel

Η κλάση VoiceViewModel διατηρεί τις αναγνωρισμένες απαντήσεις και τις επιλογές του χρήστη. Με βάση αυτά αποφασίζει σχετικά με το ποιο μήνυμα θα εκφωνηθεί, αν χρειάζεται καινούρια ακρόαση και πως θα δεχθεί επεξεργασία η επόμενη απάντηση.

Η πορεία του διαλόγου ορίζεται από την κλάση VoicePhase. Στο Σχήμα 5-19, φαίνονται οι τιμές που μπορεί να λάβει. Αναλυτικότερα, αυτές είναι αρχικά η κατάσταση IDLE δηλώνει πως την τρέχουσα στιγμή δεν βρίσκεται σε εξέλιξη διαδικασία φωνητικής επιλογής. Οι καταστάσεις PICKING_BUILDING και PICKING_ROOM δηλώνουν την αναμονή επιλογής κτηρίου και αντίστοιχα προορισμού. Τέλος, οι καταστάσεις CONFIRMING_BUILDING και CONFIRMING_ROOM δηλώνουν την επιβεβαίωση των επιλογών αυτών.

```

VoiceViewModel.kt
enum class VoicePhase {
    13 Usages
    IDLE,
    4 Usages
    PICKING_BUILDING,
    4 Usages
    CONFIRMING_BUILDING,
    8 Usages
    PICKING_ROOM,
    4 Usages
    CONFIRMING_ROOM
}

```

Σχήμα 5.24: Ορισμός της enum κλάσης VoicePhase

Για την αρχικοποίηση της VoiceViewModel εκτελείται συνάρτηση initializeVoice() δημιουργεί το αντικείμενο VoiceManager και εκφωνείται το αρχικό μήνυμα της εφαρμογής.

Έπειτα, για την εκπόνηση διαλόγων, η βασική συντονισμός είναι η `speakThenListen()`, η οποία εκφωνεί ένα μήνυμα και όταν απαιτείται απάντηση, ενεργοποιεί τον μηχανισμό ακρόασης. Μέσω του callback `onDone` καλείται η `startListening()`, αφού η ερώτηση ολοκληρωθεί. Αν η φάση του διαλόγου δεν παραμένει η ίδια, δηλαδή ο χρήστης έχει ακυρώσει τη διαδικασία ή έχει φύγει από την οθόνη, η κατάσταση μετατρέπεται σε `IDLE` και δεν ξεκινάει η ακρόαση

```

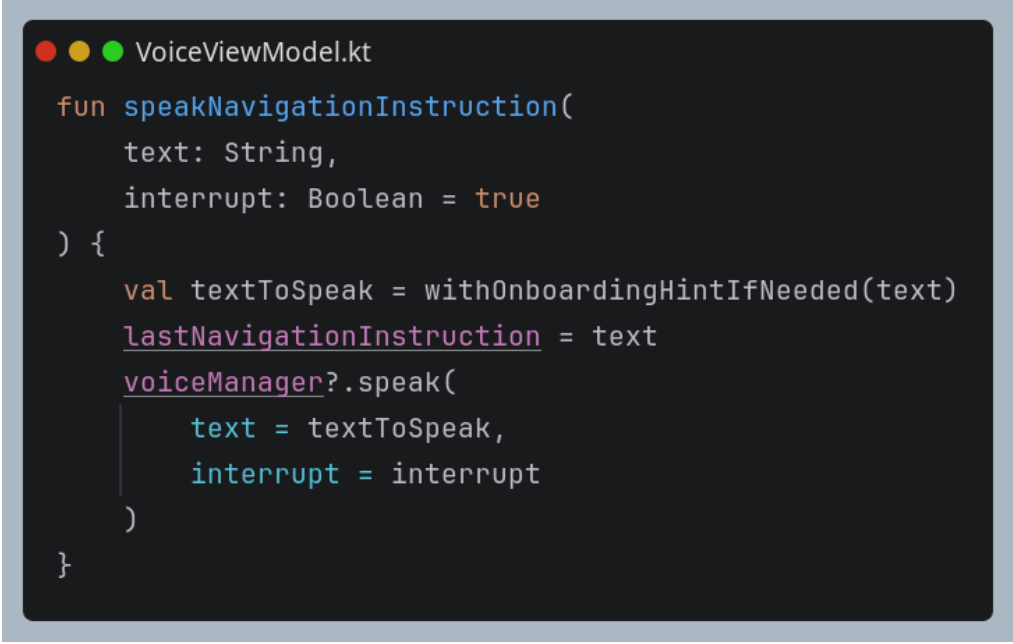
VoiceViewModel.kt

private fun speakThenListen(
    shouldListen: Boolean,
    text: String
) {
    val manager = voiceManager ?: return
    val expectedPhase = currentPhase
    manager.speak(
        text = text,
        interrupt = true,
        onDone = if (shouldListen) {
            {
                // The user may have left the destination screen while TTS
                // was speaking. Only start STT if the same phase is active.
                if (
                    currentPhase == expectedPhase &&
                    currentPhase != VoicePhase.IDLE
                ) {
                    manager.startListening()
                }
            }
        }
    )
}

```

Σχήμα 5.25: Απόσπασμα συνάρτησης `speakThenListen()`

Η συνάρτηση `speak()` χρησιμοποιείται για μηνύματα που δε χρειάζεται να αποθηκευτούν, όπως το μήνυμα που ενημερώνει τον χρήστη για την προσπάθεια εντοπισμού της τρέχουσας θέσης του. Αντίθετα, οι οδηγίες της εφαρμογής πρέπει να αποθηκεύονται, καθώς πρέπει η τελευταία κάθε φορά οδηγία να μπορεί να επαναληφθεί σε αίτημα του χρήστη. Για αυτό χρησιμοποιείται η συνάρτηση `speakNavigationInstruction()`. Αυτό επιτρέπει στην συνάρτηση `repeatLastNavigationInstruction()` να επαναλαμβάνει την τελευταία οδηγία.



```

VoiceViewModel.kt

fun speakNavigationInstruction(
    text: String,
    interrupt: Boolean = true
) {
    val textToSpeak = withOnboardingHintIfNeeded(text)
    lastNavigationInstruction = text
    voiceManager?.speak(
        text = textToSpeak,
        interrupt = interrupt
    )
}

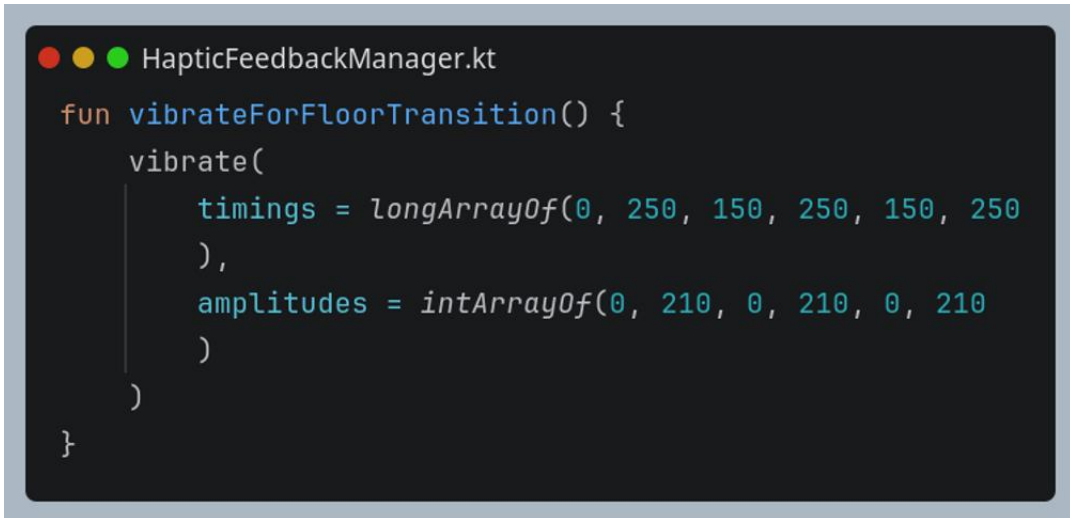
```

Σχήμα 5.26: Συνάρτηση speakNavigationInstruction()

Όταν μετά την ολοκλήρωση μιας εκφώνησης πρέπει να εκτελεστεί κάποια ενέργεια, χρησιμοποιούνται οι `speakAndThen()` και `speakNavigationInstructionAndThen()`. Η πρώτη αφορά γενικά μηνύματα, ενώ η δεύτερη αποθηκεύει παράλληλα το κείμενο ως την τελευταία οδηγία πλοήγησης. Όταν ολοκληρώνεται η φωνητική διαδικασία, η `stopAllVoiceOperations()` επαναφέρει την κατάσταση σε IDLE και καλεί την `stopAll()` της `VoiceManager`. Για την οριστική ολοκλήρωση της `VoiceViewModel`, καλείται η `destroy()`, ώστε να αποδεσμευτούν οι μηχανισμοί αναγνώρισης και σύνθεσης ομιλίας.

5.4.1.3 Μοτίβα δόνησης

Η εφαρμογή μέσω δονήσεων, ενημερώνει συμπληρωματικά τον χρήστη για κρίσιμα γεγονότα. Η υλοποίηση βασίζεται στην κλάση `HapticManager` και την enum κλάση `HapticEvent`. Αρχικά, η `HapticEvent` έχει τις εξής τιμές: `GESTURE_ACK`, το οποίο επιβεβαιώνει την αναγνώριση μίας χειρονομίας, όπως το διπλό πάτημα, το `TURN` που προειδοποιεί για μία στροφή, το `FLOOR_TRANSITION` το οποίο προειδοποιεί για μετάβαση μέσω σκάλας, το `WARNING`, που χρησιμοποιείται σε καταστάσεις όπως η απόκλιση από τη διαδρομή και το `ARRIVAL`, το οποίο σηματοδοτεί την άφιξη στον τελικό προορισμό. Κάθε γεγονός αντιστοιχεί σε διαφορετικό μοτίβο δόνησης. Παράδειγμα δημιουργίας του μοτίβου για την αλλαγή ορόφου, παρουσιάζεται στο Σχήμα 5-22



```

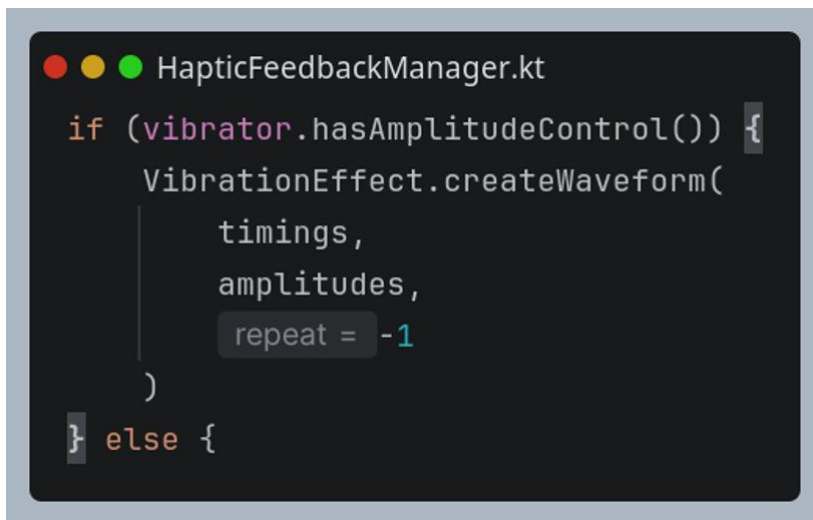
HapticFeedbackManager.kt

fun vibratesForFloorTransition() {
    vibrate(
        timings = longArrayOf(0, 250, 150, 250, 150, 250),
        amplitudes = intArrayOf(0, 210, 0, 210, 0, 210)
    )
}

```

Σχήμα 5.27: Συνάρτηση `vibratesForFloorTransition()`

Για την ενεργοποίηση του μοτίβου δόνησης στη συσκευή, χρησιμοποιείται η μέθοδος `createWaveform` της κλάσης `VibrationEffect`, όπου δίνονται ως παράμετροι το πλάτος και το μήκος του μοτίβου, με τον τρόπο που παρουσιάζεται στο Σχήμα 5-23.



```

HapticFeedbackManager.kt

if (vibrator.hasAmplitudeControl()) {
    VibrationEffect.createWaveform(
        timings,
        amplitudes,
        repeat = -1
    )
} else {

```

Σχήμα 5.28: Πραγματοποίηση δόνησης στη συσκευή

5.4.1.4 Φωνητική επιλογή κτηρίου και προορισμού

Η έναρξη φωνητικής επιλογής κτηρίου για ένα χρήστη, λειτουργεί μέσω της συνάρτησης `startBuildingSelectionLoop()`. Τα κύρια σημεία της συνάρτησης είναι πως η τρέχουσα κατάσταση ορίζεται σε `PICKING_BUILDING` και η εκφωνείται η ερώτηση για επιλογή κτηρίου, από την συνάρτηση `promptForBuilding`. Οι συναρτήσεις αυτές, όπως και το `prompt` φαίνονται στα Σχήματα 5-24 και 5-25.

```

VoiceViewModel.kt

fun startBuildingSelectionLoop() {
    voiceManager?.cancelListening()
    currentPhase = VoicePhase.PICKING_BUILDING
    phaseBeforePermissionRequest = null

    spokenTextResult = ""
    spokenRoomText = ""
    confirmedBuilding = null
    confirmedDestinationId = null
    clearPendingDestination()
    lastErrorMessage = null
    retryCount = 0

    promptForBuilding()
}

```

Σχήμα 5.29: Συνάρτηση startBuildingLoop()

```

VoiceViewModel.kt

private fun promptForBuilding() {
    speakThenListen(
        shouldListen = true,
        text = "Which building do you want to navigate to?"
    )
}

```

Σχήμα 5.30: Συνάρτηση promptForBuilding()

Για την επεξεργασία κάποιας αναγνωρισμένης απάντησης, εκτελείται η συνάρτηση `handleSpeechResult()`. Το αποτέλεσμα αυτής σχετίζεται με την τρέχουσα τιμή της `currentPhase`. Για παράδειγμα, για την κατάσταση `PICKING_BUILDING`, η εφαρμογή μεταβαίνει στην κατάσταση επιβεβαίωσης, όπως φαίνεται στην Εικόνα 5.X

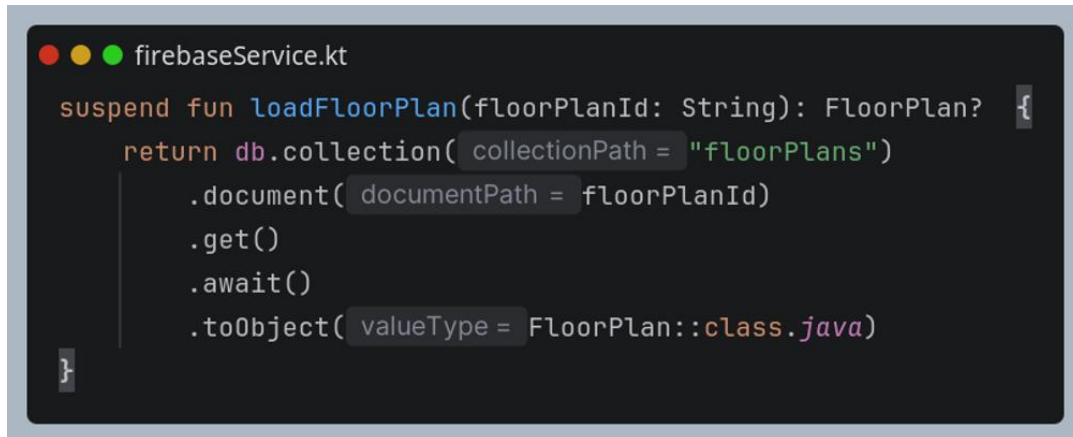
Εδώ η εξετάζεται αν ο χρήστης στην ερώτηση επιβεβαίωσης απάντησε καταφατικά ή αρνητικά. Αν η απάντηση είναι θετική, αποθηκεύεται το κτήριο και το `VoicePhase` γίνεται `PICKING_ROOM`, ώστε να επιλέξει το συγκεκριμένο προορισμό που επιθυμεί. Διαφορετικά, επαναλαμβάνεται η ερώτηση για την επιβεβαίωση του ονόματος του κτηρίου.

Στη συνέχεια, η `requestRoomConfirmation()` επιβεβαιώνει την ονομασία, τον όροφο και το `id` του προορισμού και αλλάζει την κατάσταση σε `CONFIRMING_ROOM`. Μόλις επιβεβαιωθεί ο προορισμός, ξεκινά η διαδικασία πλοήγησης. Με την επιβεβαίωση το αναγνωριστικό του προορισμού είναι διαθέσιμο μέσω του `confirmedDestinationId` και το όνομα του κτηρίου μέσω του `confirmedBuilding`. Έτσι, η οθόνη επιλογής προορισμού μπορεί να προσδιορίσει τον αντίστοιχο αποθηκευμένο χάρτη, ώστε να ξεκινήσει η διαδικασία πλοήγησης. Ακολουθεί η διαδικασία ανάκτησης του χάρτη, η μετατροπή σε γράφο.

5.4.2 Ανάκτηση χάρτη και κατασκευή γράφου

5.4.2.1 Ανάκτηση χάρτη

Μετά την επιλογή κτηρίου και προορισμού, η εφαρμογή ανακτά τον χάρτη του επιλεγμένου χαρτογραφημένου κτηρίου και κατασκευάζει τον γράφο που το αναπαριστά. Για την ανάκτηση του χάρτη, εκτελείται η συνάρτηση `loadFloorPlan()` (Σχήμα 5-26), η οποία δέχεται ως παράμετρο το μοναδικό αναγνωριστικό του συγκεκριμένου κτηρίου. Η συνάρτηση αυτή, αφού πραγματοποιήσει αναζήτηση του εγγράφου στη συλλογή `floorPlans`, μετατρέπει τα δεδομένα του σε αντικείμενο `FloorPlan`.

A screenshot of a code editor showing the implementation of the `loadFloorPlan` function in a file named `firebaseService.kt`. The function is a suspend function that takes a `floorPlanId` of type `String` and returns a `FloorPlan?`. It uses the Firestore database to retrieve a document from the `floorPlans` collection. The code is as follows:

```
suspend fun loadFloorPlan(floorPlanId: String): FloorPlan? {  
    return db.collection(collectionPath = "floorPlans")  
        .document(documentPath = floorPlanId)  
        .get()  
        .await()  
        .toObject(valueType = FloorPlan::class.java)  
}
```

Σχήμα 5.31: Συνάρτηση `loadFloorPlan()`

5.4.2.2 Έλεγχος εγκυρότητας

Πριν την κατασκευή του γράφου, ελέγχεται η εγκυρότητα των δεδομένων του γράφου μέσω της `validateGraph()`. Ανάμεσα στους ελέγχους που πραγματοποιούνται είναι πως υπάρχει η αρχική άγκυρα του χάρτη, οι αποθηκευμένες συντεταγμένες είναι έγκυρες, επιβεβαιώνεται πως για κάθε ακμή υπάρχουν δύο κόμβοι. Αυτοί οι έλεγχοι αποτρέπουν την έναρξη πλοήγησης με ελλιπή δεδομένα.

5.4.2.3 Αναπαράσταση γράφου

Για την αναπαράσταση του γράφου στο αρχείο `ARNavigationScreen`, χρησιμοποιούνται οι κλάσεις `GraphNode`, `GraphEdge` και `IndoorGraph`. Συγκεκριμένα, η κλάση `GraphNode` αναπαριστά έναν κόμβο του γράφου. Περιλαμβάνει τις ακόλουθες πληροφορίες για κάθε κόμβο: το αναγνωριστικό του, (`id`) το όνομα του (`label`), τις συντεταγμένες (`x,y,z`) του, τον όροφο (`floor`) στον οποίο τοποθετήθηκε, καθώς και ένδειξη για το αν αποτελεί `Cloud Anchor`, δηλαδή προορισμό ή `Waypoint`, δηλαδή ενδιάμεσο σημείο πλοήγησης (`isAnchor`). Η κλάση `GraphEdge` αναπαριστά μία ακμή του γράφου. Περιέχει τον γειτονικό κόμβο (`target`) τύπου `GraphNode` και το βάρος της σύνδεσης (`weight`). Τέλος, η κλάση `IndoorGraph` περιλαμβάνει την λίστα γειτνίασης του γράφου. Κάθε κλειδί του `adjencyMap` είναι αντικείμενο `GraphNode`, ενώ η αντίστοιχη τιμή είναι η λίστα των ακμών που συνδέουν τον κόμβο με τους γειτονικούς του.

```

ARNavigationScreen.kt

data class GraphNode(
    val id: String,
    val label: String,
    val x: Float,
    val y: Float,
    val z: Float,
    val floor: Int,
    val isAnchor: Boolean
)

data class GraphEdge(
    val target: GraphNode,
    val weight: Float
)

data class IndoorGraph(val adjacencyMap: Map<GraphNode, List<GraphEdge>>)

```

Σχήμα 5.32: Παρουσίαση κλάσεων GraphNode, GraphEdge, IndoorGraph

Η συνάρτηση `computeMappedPositions()`, χρησιμοποιώντας την συνάρτηση `associate` επιστρέφει ένα `Map`. Για κάθε αντικείμενο `PathStep`, δημιουργείται ένα ζευγάρι κλειδιού – τιμής, όπου το `key` είναι το αναγνωριστικό του σημείου και το `value` είναι ένα αντικείμενο `Position`, το οποίο περιέχει τις συντεταγμένες (x,y,z) του σημείου.

Για παράδειγμα, ένα key-value pair, θα μπορούσε να είναι: “main_anchor” → `Position (0f,0f,0f)`, το οποίο αφορά την κεντρική είσοδο, η οποία θεωρείται πως βρίσκεται στην αρχή του συστήματος συντεταγμένων.

```

ARNavigationScreen.kt

fun computeMappedPositions(floorPlan: FloorPlan): Map<String, Position> =
    floorPlan.nodes.associate { pathStep ->
        pathStep.id to Position(x = pathStep.x, y = pathStep.y, z = pathStep.z)
    }

```

Σχήμα 5.33: Συνάρτηση `computeMappedPositions()`

5.4.2.4 Κατασκευή γράφου

Η κατασκευή του γράφου πραγματοποιείται από την συνάρτηση `buildNavigationGraph()`, η οποία φαίνεται στο Σχήμα 5-29. Αρχικά, η συνάρτηση δέχεται ως είσοδο ένα αντικείμενο `FloorPlan` και αφού κατασκευάσει τον γράφο του κτηρίου, επιστρέφει ένα αντικείμενο `IndoorGraph`. Καλείται η `computeMappedPositions()` που περιγράφηκε παραπάνω. Μέσω της `map` κάθε `PathStep` μετατρέπεται σε αντικείμενο `GraphNode`, γίνεται δηλαδή κόμβος του γράφου. Για κάθε σημείο αναζητείται η αντίστοιχη θέση του. Έπειτα, αρχικοποιείται η λίστα γειτνίασης, δημιουργώντας μία κενή λίστα ακμών για κάθε κόμβο.

Η συνάρτηση αναζητά για κάθε αποθηκευμένη σύνδεση, τον αρχικό και τελικό κόμβο στο `FloorPlan` και το πεδίο `distanceMeters` χρησιμοποιείται για τον καθορισμό του βάρους κάθε ακμής του γράφου. Για κάθε σύνδεση δημιουργούνται δύο ακμές, για κάθε κατεύθυνση καθώς ο γράφος είναι αμφίδρομος. Όπως περιγράφηκε, στην αντίστροφη ακμή, αντιστρέφεται και ο τύπος σύνδεσης, όταν απαιτείται. Η

λίστα λίστα γειτνίασης χρησιμοποιείται για τη δημιουργία και επιστροφή του τελικού αντικειμένου IndoorGraph. Τέλος, η αναζήτηση του κόμβου προορισμού γίνεται με βάση το αναγνωριστικό προορισμού στη λίστα nodes,

```
ARNavigationScreen.kt

fun buildNavigationGraph(floorPlan: FloorPlan): IndoorGraph {
    val mappedPositions = computeMappedPositions(floorPlan)
    val nodes = floorPlan.nodes.map { pathStep ->
        val pos = mappedPositions[pathStep.id] ?: Position(x = 0f, y = 0f, z = 0f)
        GraphNode(
            id = pathStep.id,
            label = pathStep.label,
            x = pos.x,
            y = pos.y,
            z = pos.z,
            floor = pathStep.floor,
            isAnchor = pathStep.isAnchor
        )
    }
    val nodeById = nodes.associateBy { it.id }

    val adjMap = mutableMapOf<GraphNode, MutableList<GraphEdge>>()
    nodes.forEach { adjMap[it] = mutableListOf() }

    for (connection in floorPlan.connections) {
        val from = nodeById[connection.fromId] ?: continue
        val to = nodeById[connection.toId] ?: continue
        adjMap[from]?.add(
            GraphEdge(target = to, weight = connection.distanceMeters)
        )
        adjMap[to]?.add(
            GraphEdge(target = from, weight = connection.distanceMeters)
        )
    }

    return IndoorGraph(adjacencyMap = adjMap)
}
```

Σχήμα 5.34: Συνάρτηση buildNavigationGraph()

5.4.3 Αρχικοποίηση συνεδρίας και επίλυση Cloud Anchors

Αφού κατασκευαστεί ο γράφος, πρέπει να προσδιοριστεί η θέση του χρήστη στον χώρο. Γι' αυτό, αρχικοποιείται μία συνεδρία ARCore μέσω του ARSceneView, όπως περιγράφεται στην Ενότητα 5.3.1, η οποία χρησιμοποιείται για την παρακολούθηση μέσω κάμερας και για επίλυση των Cloud Anchors

που αποθηκεύτηκαν κατά τη χαρτογράφηση. Το μόνο που αλλάζει στην αρχικοποίηση είναι ο ορισμός του `planeRenderer` σε `false`. Παράλληλα, παραλείπεται ο ορισμός του `planeFindingMode`, καθώς η εμφάνιση της ανίχνευση επιφανειών δε χρειάζεται, καθώς ο χρήστης δε θα τοποθετήσει νέα ψηφιακά στοιχεία στον χώρο.

Για την επίλυση των Cloud Anchor, στέλνεται αίτημα επίλυσης στο ARCore Cloud Anchor API, το οποίο συγκρίνει τα οπτικά χαρακτηριστικά της τρέχουσας συνεδρίας μέσω κάμερας με αυτά που αποθηκεύθηκαν στο τρισδιάστατο χάρτη χαρακτηριστικών (3D feature map). Αν υπάρχει αντιστοίχιση, επιστρέφεται ένα τοπικό αντικείμενο Anchor, το οποίο εκφράζεται σε συντεταγμένες του συστήματος συντεταγμένων της τρέχουσας συνεδρίας AR.

Αρχικά, στη λίστα `allCloudAnchors` φιλτράρονται και επιλέγονται οι κόμβοι που έχουν `isAnchor = true`, δηλαδή αντιστοιχούν σε Cloud Anchors και διαθέτουν μη κενό αναγνωριστικό. Το αναγνωριστικό κάθε Cloud Anchor προστίθεται στη λίστα `pendingAnchorIds`, η οποία αναπαριστά τα αναγνωριστικά που εκκρεμούν προς επίλυση. Όσο ένα αντικείμενο Anchor βρίσκεται σε ενδιάμεση κατάσταση επίλυσης, καταχωρείται στη λίστα `resolvingAnchors`, ενώ μόλις επιλυθεί επιτυχώς, μεταφέρεται στη λίστα `resolvedAnchors`. Η αντιστοίχιση κάθε επιλυμένου Anchor με το αποθηκευμένο αναγνωριστικό του, διατηρείται στη δομή `resolvedAnchorToIdMap`, ώστε να είναι δυνατή η σύνδεση του με τα δεδομένα του FloorPlan.

ARNavigationScreen.kt

```
val allCloudAnchors = remember( key1 = floorPlan) {
    floorPlan.nodes.filter { it.isAnchor && it.id.isNotBlank() }
}

var resolvedAnchors by remember( key1 = floorPlan.floorPlanId) {
    mutableStateOf<List<Anchor>>( value = emptyList())
}

var resolvingAnchors by remember( key1 = floorPlan.floorPlanId) {
    mutableStateOf<List<Anchor>>( value = emptyList())
}

var resolvedAnchorToIdMap by remember( key1 = floorPlan.floorPlanId) {
    mutableStateOf<Map<Anchor, String>>( value = emptyMap())
}

val pendingAnchorIds = remember( key1 = floorPlan.floorPlanId) {
    mutableStateListOf<String>().apply {
        addAll( elements = allCloudAnchors.map { it.id })
    }
}

val anchorIdsInFlight = remember( key1 = floorPlan.floorPlanId) {
    mutableStateListOf<String>()
}
```

Σχήμα 5.35: Οι λίστες που περιγράφηκαν παραπάνω

Μέσω της `onSessionUpdated` ελέγχονται τα νέα αιτήματα επίλυσης. Ελέγχεται ότι η κάμερα βρίσκεται σε κατάσταση ενεργής καταγραφής (TRACKING) και πως υπάρχουν εκκρεμή αναγνωριστικά Cloud Anchor. Αν ισχύει, υπολογίζονται οι θέσεις επίλυσης, τα αντίστοιχα αναγνωριστικά αφαιρούνται από την λίστα εκκρεμών αιτημάτων και προστίθενται στην λίστα `anchorIdsInFlight`. Για κάθε ένα καλείται η συνάρτηση `resolveCloudAnchorAsync()`, η οποία επιστρέφει το τοπικό αντικείμενο Anchor και την κατάσταση επίλυσης.

Όταν η κατάσταση είναι SUCCESS, το αναγνωριστικό αφαιρείται από τη λίστα `anchorIdsInFlight` και το Anchor αφαιρείται από τη `resolvingAnchors`. Έπειτα, το νέο αντικείμενο προστίθεται στη λίστα `resolvedAnchors` και καταγράφεται η αντιστοίχισή του με το αναγνωριστικό του στην `resolvedAnchorToIdMap`.

5.4.4 Μετασχηματισμός συντεταγμένων και εντοπισμός χρήστη

Αφού επιλυθεί τουλάχιστον ένα Cloud Anchor, η εφαρμογή πρέπει να υπολογίσει τη θέση της κάμερας της συσκευής στο ίδιο σύστημα συντεταγμένων του αποθηκευμένου χάρτη του κτηρίου και όχι με βάση το σύστημα της τρέχουσας συνεδρίας ARCore. Αυτό θα γίνει μέσω της επιλυμένης άγκυρας, καθώς το Pose είναι γνωστό σε δύο διαφορετικά συστήματα αναφοράς: το ARCore επιστρέφει το Pose της στην τρέχουσα συνεδρία (`anchor.pose`), ενώ στα δεδομένα του χάρτη είναι διαθέσιμο το Pose σε σχέση με την πρώτη άγκυρα που τοποθετήθηκε κατά την χαρτογράφηση (Ενότητα 5.3.6).

Η `mappedAnchorPose()` ανακτά το αποθηκευμένο Pose της άγκυρας, όπως αυτό έχει καταγραφεί στο PathStep κατά την χαρτογράφηση, δηλαδή τη θέση και τον προσανατολισμό στο κοινό σύστημα συντεταγμένων του χάρτη. Η `cameraPoseInMap()` υπολογίζει το `anchorToCamera`, δηλαδή το Pose της κάμερας σε σχέση με την άγκυρα. Αυτό υπολογίζεται μέσω του `anchor.pose.inverse()`, δηλαδή μέσω του αντίστροφου μετασχηματισμού της άγκυρας σε σύνθεση (`compose()`) με το αποθηκευμένο Pose της άγκυρας στον χάρτη. Ουσιαστικά, το αποτέλεσμα είναι η θέση και ο προσανατολισμός της κάμερας εκφρασμένα στο ίδιο κοινό σύστημα συντεταγμένων με τον υπόλοιπο γράφο.

```
ARNavigationScreen.kt

fun mappedAnchorPose(anchorId: String): Pose? {
    val pathStep = allCloudAnchors.firstOrNull { it.id == anchorId } ?: return null
    val position = mappedPositions[anchorId] ?: return null
    return Pose(
        translation = floatArrayOf(position.x, position.y, position.z),
        rotation = floatArrayOf(pathStep.qx, pathStep.qy, pathStep.qz, pathStep.qw)
    )
}

fun cameraPoseInMap(anchor: Anchor, cameraPose: Pose): Pose? {
    val anchorId = resolvedAnchorToIdMap[anchor] ?: return null
    val mapAnchor = mappedAnchorPose(anchorId) ?: return null
    val anchorToCamera = anchor.pose.inverse().compose( rhs = cameraPose)
    return mapAnchor.compose( rhs = anchorToCamera)
}
```

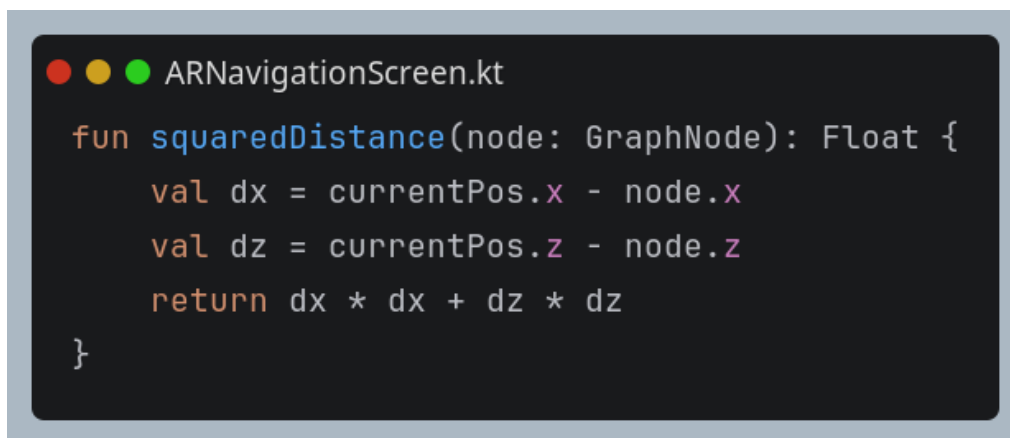
Σχήμα 5.36: Συναρτήσεις `mappedAnchorPose()` και `cameraPoseInMap()`

Η επιλογή της άγκυρας που χρησιμοποιείται ως το σημείο αναφοράς επαναπροσδιορίζεται όσο ο χρήστης κινείται στον χώρο. Αυτό συμβαίνει καθώς η ακρίβεια επίλυσης μίας άγκυρας μειώνεται όσο ο χρήστης απομακρύνεται από αυτή. Για τον προσδιορισμό της άγκυρας αναφοράς κάθε καρέ αξιολογείται ποια από τις ενεργά παρακολουθούμενες και επιλυμένες άγκυρες βρίσκονται εγγύτερα στην κάμερα. Μία νέα άγκυρα γίνεται αποδεκτή αν είναι πλησιέστερα στην κάμερα από την τρέχουσα ενεργή άγκυρα κατά τουλάχιστον 0,3 μέτρα, ενώ οι δύο λύσεις της κάμερας θα πρέπει να συμφωνούν μεταξύ τους με απόκλιση μικρότερη των 0.75 μέτρων. Αυτός ο έλεγχος πραγματοποιείται σαν μέτρο ασφαλείας, καθώς σε περίπτωση που δύο άγκυρες δίνουν σημαντικά διαφορετική εκτίμηση σχετικά με την θέση της καμερας, σημαίνει πως κάποια άγκυρα πιθανόν να μην έχει επιλυθεί σωστά.

5.4.5 Υπολογισμός βέλτιστης διαδρομής

Το επόμενο βήμα μετά τον επιτυχή εντοπισμό της τρέχουσας θέσης του χρήστη, είναι ο υπολογισμός της βέλτιστης διαδρομής προς τον επιλεγμένο προορισμό. Όπως περιγράφηκε στην υπό-ενότητα 5.4.2, κάθε κόμβος του γράφου αναπαρίσταται ως ένα αντικείμενο τύπου `GraphNode`, κάθε ακμή του γράφου αναπαρίσταται ως ένα αντικείμενο `GraphEdge`, ενώ η χάρτης γειτνίασης αναπαρίσταται από την κλάση `IndoorGraph`.

Αφού το χαρτογραφημένο κτήριο `FloorPlan` μετατραπεί σε αντικείμενο τύπου `IndoorGraph`, προσδιορίζεται ο πλησιέστερος κόμβος στην τρέχουσα θέση του χρήστη, ο οποίος χρησιμοποιείται ως αφετηρία. Μέσω της `squaredDistance` υπολογίζεται η διαφορά του τρέχον κόμβου με τους γύρω, ώστε να βρεθεί ο πλησιέστερος. Ο υπολογισμός πραγματοποιείται ως προς τους άξονες x, z , καθώς είναι δεδομένο πως ο πλησιέστερος κόμβος θα ανήκει στον ίδιο όροφο.



```

ARNavigationScreen.kt

fun squaredDistance(node: GraphNode): Float {
    val dx = currentPos.x - node.x
    val dz = currentPos.z - node.z
    return dx * dx + dz * dz
}

```

Σχήμα 5.37: Συνάρτηση `squaredDistance()`

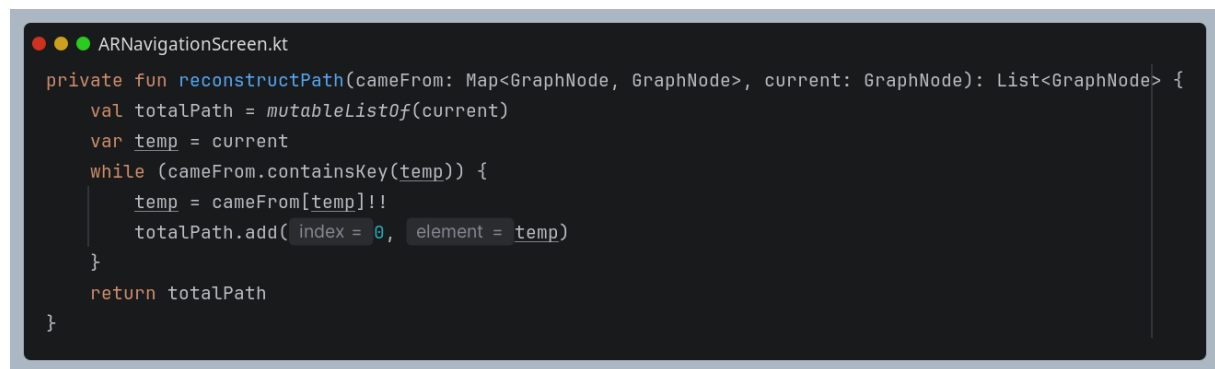
Όσον αφορά τον υπολογισμό της συντομότερης διαδρομής, χρησιμοποιείται ο αλγόριθμος A^* , ο οποίος εξετάζει κάθε κόμβο μέσω της συνάρτησης $f(n) = g(n) + h(n)$, όπου $g(n)$ το συνολικό κόστος διαδρομής από τον αρχικό κόμβο μέχρι τον τρέχον κόμβο και όπου $h(n)$, η ευρετική εκτίμηση της απόστασης του τρέχον κόμβου από τον προορισμό. Ο αλγόριθμος αναπαρίσταται από την συνάρτηση `findShortestPath()`.

Η συνάρτηση δέχεται ως είσοδο το αντικείμενο `IndoorGraph` και τα αναγνωριστικά των κόμβων αφετηρίας και προορισμού, ενώ επιστρέφει τη διαδρομή από την αφετηρία στον προορισμό, ως μία λίστα κόμβων. Αρχικά, δημιουργείται η ουρά προτεραιότητας `openSet`, όπου αποθηκεύονται οι κόμβοι που έχουν εντοπιστεί, αλλά δεν έχουν ακόμη εξεταστεί. Δημιουργούνται επίσης οι δομές `gScore`, η

οποία αναπαριστά το μικρότερο γνωστό κόστος μετάβασης από την αφετηρία προς κάθε κόμβο και η `cameFrom`, η οποία αντιστοιχεί κάθε κόμβο με τον προηγούμενό του στην καλύτερη γνωστή διαδρομή. Τέλος, η μεταβλητή `startH` αποθηκεύει την ευκλείδεια απόσταση από τον αρχικό κόμβο μέχρι τον προορισμό. Το πραγματικό κόστος του πρώτου κόμβου είναι 0, άρα $f(\text{start}) = 0 + h(\text{start}) = h(\text{start})$.

Ο αλγόριθμος αφαιρεί επαναληπτικά από την `openSet` τον κόμβο με τη μικρότερη $f(n)$ τιμή. Αν ο κόμβος που αφαιρείται είναι τελικός, η αναζήτηση ολοκληρώνεται, ενώ αν ο κόμβος δεν είναι τελικός, θεωρείται πως εξετάστηκε και ελέγχονται οι ακμές που τον συνδέουν με τους γειτονικούς του κόμβους, ενημερώνοντας το `gScore` όσων γειτόνων βρίσκουν καλύτερη διαδρομή μέσω του τρέχοντος κόμβου.

Αφού εξαχθεί ο κόμβος προορισμού από την ουρά, εκτελείται η `reconstructPath()` (Σχήμα 5-33), η οποία ξεκινώντας από τον τελικό κόμβο, ακολουθεί τις καταχωρίσεις της `cameFrom` προς τα πίσω μέχρι να φτάσει στον αρχικό κόμβο. Τέλος, η λίστα `totalPath` περιλαμβάνει αρχικά μόνο τον προορισμό, ενώ με κάθε επανάληψη προσδιορίζεται ο προηγούμενος κόμβος του προορισμού, μέσω της `cameFrom`. Ο κόμβος αυτός προστίθεται στην αρχή της λίστας, ώστε η διαδρομή να βρίσκεται στη σωστή σειρά.



```
private fun reconstructPath(cameFrom: Map<GraphNode, GraphNode>, current: GraphNode): List<GraphNode> {
    val totalPath = mutableListOf<GraphNode>(current)
    var temp = current
    while (cameFrom.containsKey(temp)) {
        temp = cameFrom[temp]!!
        totalPath.add(0, temp)
    }
    return totalPath
}
```

Σχήμα 5.38: Συνάρτηση `reconstructPath()`

5.4.6 Παρακολούθηση πορείας και υπολογισμός οδηγιών

Αφού υπολογιστεί η διαδρομή προς τον προορισμό, η εφαρμογή παρακολουθεί την θέση του χρήστη και την συγκρίνει με τη θέση του επόμενου σημείου της διαδρομής (`activeNavigationPath`), ώστε να αποφασίσει την κατάλληλη οδηγία να εκφωνήσει. Κάθε φορά, υπολογίζεται η απόσταση του χρήστη από τον επόμενο κόμβο της διαδρομής. Ο χρήστης θεωρείται πως έφτασε στον επόμενο κόμβο όταν ικανοποιούνται οι δύο παρακάτω συνθήκες: η απόσταση του είναι μικρότερη ή ίση των 0.60 μέτρων (`waypointArrivalThreshold`) και βρίσκεται στον ίδιο όροφο. Αφού οι συνθήκες ισχύουν για ένα χρονικό διάστημα που ορίζεται στην `requiredArrivalDurationMs`, επιβεβαιώνεται η άφιξη του στο επόμενο κόμβο.

Όσον αφορά τα Waypoints, αυτά πρέπει να τοποθετούνται ανά 2-3 μέτρα στον χώρο κατά την χαρτογράφηση. Αυτό σημαίνει πως πολλές φορές στην ίδια ευθεία υπάρχουν πολλά Waypoints. Για την αποφυγή επανάληψης οδηγιών ανά λίγα μέτρα, η εφαρμογή παράγει οδηγίες καθοδήγησης μόνο για το επόμενο σημείο απόφασης, δηλαδή αλλαγή κατεύθυνσης, ορόφου ή κατά την άφιξη στον προορισμό και όχι σε κάθε ενδιάμεσο σημείο.

Η `determineTurnAction()` αποφασίζει: αν κόμβος είναι ο τελευταίος της διαδρομής, ορίζει την ενέργεια σε `ARRIVE`, αν η σύνδεση προς τον επόμενο κόμβο είναι σκάλα ορίζει `STAIRS_UP/DOWN` αντίστοιχα, αν είναι ο πρώτος ορίζει αυτόματα `STRAIGHT`, ενώ σε κάθε άλλη περίπτωση, υπολογίζεται η διαφορά της γωνίας κατεύθυνσης πριν και μετά τον κόμβο. Έχουν ορισθεί οι 40 μοίρες ως κατώφλι,

καθώς είναι πιθανό να υπάρχουν μικρές αλλαγές στην κατεύθυνση κατά την πορεία του χρήστη στον χώρο.

```
ARNavigationScreen.kt
return when {
    angleDiff < -40.0 -> TurnAction(
        type = TurnType.LEFT,
        shouldPreannounce = true
    )
    angleDiff > 40.0 -> TurnAction(
        type = TurnType.RIGHT,
        shouldPreannounce = true
    )
    else -> TurnAction( type = TurnType.STRAIGHT, shouldPreannounce = false)
}
```

Σχήμα 5.39: Απόσπασμα επιστροφής της συνάρτησης determineTurnAction()

Μετά η findNextDecisionIndex() διατρέχει τους επόμενους κόμβους της διαδρομής και αναζητά αν το TurnAction είναι STRAIGHT. Αν δεν ισχύει αυτό, επιστρέφει τον δείκτη του επόμενου σημείου απόφασης.

```
ARNavigationScreen.kt
private fun findNextDecisionIndex(
    path: List<GraphNode>,
    fromIndex: Int,
    connectionTypes: Map<DirectedConnectionKey, PathConnectionType>
): Int {
    var index = fromIndex
    while (
        index < path.lastIndex &&
        determineTurnAction(path, index, connectionTypes).type == TurnType.STRAIGHT
    ) {
        index++
    }
    return index
}
```

Σχήμα 5.40: Συνάρτηση findNextDecisionIndex()

Η συνάρτηση `generateInstruction()` υπολογίζει την απόσταση μέχρι το επόμενο σημείο απόφασης και τη συνολική απόσταση μέχρι τον προορισμό. Διαμορφώνει το αντίστοιχο μήνυμα ανάλογα με το είδος της ενέργειας που επέστρεψε η `determineTurnAction()`.

```
ARNavigationScreen.kt
val mainInstruction = when (action.type) {
    TurnType.LEFT -> "Walk straight, then in $spokenDistance, turn left$destinationSuffix."
    TurnType.RIGHT -> "Walk straight, then in $spokenDistance, turn right$destinationSuffix."
    TurnType.STAIRS_UP -> "Stairs going up in $spokenDistance."
    TurnType.STAIRS_DOWN -> "Stairs going down in $spokenDistance."
    TurnType.ARRIVE -> "${targetNode.label} is $spokenDistance ahead."
    TurnType.STRAIGHT -> "Continue straight for $spokenDistance."
}
```

Σχήμα 5.41: Απόσπασμα περιπτώσεων οδηγιών που παράγονται από την `generateInstruction()`

Η απόσταση μέχρι το επόμενο σημείο απόφασης (`distanceToNext`) υπολογίζεται αθροίζοντας τα ενδιάμεσα σημεία διαδρομής (`Waypoints`)., Με τον ίδιο τρόπο υπολογίζεται και η εναπομείνουσα απόσταση μέχρι τον τελικό κόμβο (`totalDistanceRemaining`), αθροίζοντας τα υπόλοιπα σημεία της διαδρομής.

Η συνάρτηση `generateNowInstruction()`, παράγει την οδηγία για την στιγμή που ο χρήστης φτάνει πραγματικά στο σημείο απόφασης, για παράδειγμα μία οδηγία θα ήταν «Turn left now»

Επειδή η θέση του χρήστη ενημερώνεται πολλές φορές το δευτερόλεπτο, η ίδια οδηγία μπορεί να υπολογιστεί εκ νέου πολλές φορές πριν προλάβει ο χρήστης να προχωρήσει. Για να αποφευχθεί η εκφώνηση της ίδιας οδηγίας κάθε φορά, κάθε υποψήφια οδηγία συνοδεύεται από ένα αναγνωριστικό συμβάντος, το οποίο συγκρίνεται με το τελευταίο που είχε ήδη ανακοινωθεί. Η κλήση προς την `speakNavigationInstruction()` πραγματοποιείται μόνο όταν το `cueKey` του τρέχοντος συμβάντος διαφέρει από το `lastSpokenCueKey`.

```
ARNavigationScreen.kt
val cueKey = "segment:$currentPathIndex:${segmentInstruction.mainInstruction}"
if (lastSpokenCueKey != cueKey) {
    voiceViewModel.speakNavigationInstruction(
        text = segmentInstruction.mainInstruction
    )
    lastSpokenCueKey = cueKey
}
```

Σχήμα 5.42: Απόσπασμα ορισμού `cueKey`

5.4.7 Επανάληψη οδηγιών

Η επανάληψη μίας οδηγίας πραγματοποιείται μέσω της `repeatLastNavigationInstruction()`, η οποία ήδη έχει αναλυθεί στην Ενότητα 5.4.1.2 και ενεργοποιείται μέσω χειρονομίας διπλού πατήματος στο πάνω μέρος της οθόνης. Η χειρονομία συνοδεύεται από σύντομο μοτίβο δόνησης (`HapticEvent.GESTURE_ACK`), το οποίο επιβεβαιώνει στον χρήστη ότι η χειρονομία αναγνωρίστηκε. Μετά την επιτυχή επανάληψη μίας οδηγίας, ενημερώνεται το `spokenInstructionHoldUntil`, ώστε ο βρόχος παρακολούθησης να μην ξεκινήσει αμέσως να υπολογίζει και να προωθεί μία οδηγία πάνω από αυτή που επαναλήφθηκε.



```

ARNavigationScreen.kt

modifier = modifier.pointerInput(
    ...keys = voiceViewModel,
    hasFinishedNavigation,
    isCompletingStairTransition
) {
    detectTapGestures(
        onDoubleTap = {
            if (!hasFinishedNavigation) {
                voiceViewModel.performHapticFeedback(
                    event = HapticEvent.GESTURE_ACK
                )
                if (
                    !isCompletingStairTransition &&
                    voiceViewModel.repeatLastNavigationInstruction()
                ) {
                    spokenInstructionHoldUntil =
                        SystemClock.elapsedRealtime() + spokenInstructionHoldMs
                }
            }
        },
    )
}

```

Σχήμα 5.43: Απόσπασμα δυνατότητας επανάληψης οδηγίας μέσω χειρονομίας

5.4.8 Ανίχνευση απόκλισης από την πορεία και αποκατάσταση

Κατά τη διάρκεια της πορείας του χρήστη προς τον προορισμό χρησιμοποιώντας την προτεινόμενη διαδρομή, ελέγχεται από την εφαρμογή η απόκλιση του χρήστη από την πορεία. Η απόκλιση από την πορεία υπολογίζεται ως κάθετη απόσταση από το τμήμα που συνδέει τον τρέχοντα με τον επόμενο κόμβο. Η συνάρτηση `distancePointToSegmentXZ`, προβάλλει κάθετα τη θέση του χρήστη πάνω στο ευθύγραμμο τμήμα της διαδρομής και υπολογίζει την απόσταση από το σημείο προβολής.

Η μεταβλητή `projection` εκφράζει το πόσο κοντά στην αρχή ή το τέλος είναι η προβολή, ώστε το σημείο προβολής να μη ξεπερνά τα άκρα του. Έτσι, ακόμη και αν ο χρήστης βρίσκεται πλάγια του τμήματος, η υπολογιζόμενη απόσταση αντιστοιχεί στην πραγματική εγγύτητα του στη διαδρομή.

Έλεγχος απόκλισης πραγματοποιείται μόνο για τμήματα τύπου WALK. Με την επιβεβαίωση την απόκλισης, ενεργοποιείται δόνηση προειδοποίησης (`HapticEvent.WARNING`) και αναζητείται ο

πλησιέστερος κόμβος του γράφου, με τον τρόπο που ήδη περιγράφηκε. Από τον κόμβο αυτό καλείται ξανά ο αλγόριθμος A* και υπολογίζεται εκ νέου η διαδρομή. Αν βρεθεί έγκυρη διαδρομή, αντικαθιστά την ενεργή διαδρομή πλοήγησης. Διαφορετικά ο χρήστης ενημερώνεται για την απόκλιση του από την πορεία και λαμβάνει οδηγίες για το πως να επιστρέψει στην χαρτογραφημένη διαδρομή, συγκεκριμένα προτείνεται να διακόψει την κίνηση του και να σαρώσει αργά τον περιβάλλοντα χώρο.

Η εφαρμογή επιπλέον παρακολουθεί και πότε η θέση του χρήστη σταματά να ενημερώνεται, όταν δηλαδή χαθεί το tracking. Αν έχουν περάσει πάνω από 3 δευτερόλεπτα από την τελευταία ενημέρωση της θέσης του χρήστη, τότε η θέση χαρακτηρίζεται μη ενημερωμένη και η καθοδήγηση τίθεται σε παύση. Το μήνυμα που θα εμφανιστεί στην οθόνη και θα ακουστεί ως προειδοποίηση μέσω της φωνητικής οδηγίας, ενημερώνει τον χρήστη πως έχει χαθεί η παρακολούθηση και τον προτρέπει να σκανάρει τον χώρο με την κάμερα, ώστε το ARCore να βρει ξανά σημεία αναφοράς.

```
ARNavigationScreen.kt

private fun distancePointToSegmentXZ(
    pointX: Float,
    pointZ: Float,
    startX: Float,
    startZ: Float,
    endX: Float,
    endZ: Float
): Float {
    val segmentX = endX - startX
    val segmentZ = endZ - startZ
    val lengthSquared = segmentX * segmentX + segmentZ * segmentZ
    if (lengthSquared <= 0.0001f) {
        val dx = pointX - startX
        val dz = pointZ - startZ
        return sqrt(x= dx * dx + dz * dz)
    }

    val projection = (
        (pointX - startX) * segmentX +
        (pointZ - startZ) * segmentZ
    ) / lengthSquared
    val t = projection.coerceIn(0f, 1f)
    val nearestX = startX + t * segmentX
    val nearestZ = startZ + t * segmentZ
    val dx = pointX - nearestX
    val dz = pointZ - nearestZ
    return sqrt(x= dx * dx + dz * dz)
}
```

Σχήμα 5.44: Συνάρτηση distancePointToSegmentXZ()

5.4.9 Ολοκλήρωση πλοήγησης

Η διαδρομή ολοκληρώνεται όταν ο χρήστης προσεγγίσει τον τελευταίο κόμβο της `activeNavigationPath`. Ξανά, υπολογίζεται η οριζόντια απόσταση του χρήστη από τον τελικό κόμβο προορισμού, με τον τρόπο που έχει ήδη περιγραφεί. Αν πρόκειται πράγματι για τον τελευταίο κόμβο της διαδρομής, η μεταβλητή `reachedAction` μέσω της `determineTurnAction()` επιστρέφει την τιμή `TurnType.ARRIVE` και δημιουργείται η τελική οδηγία.

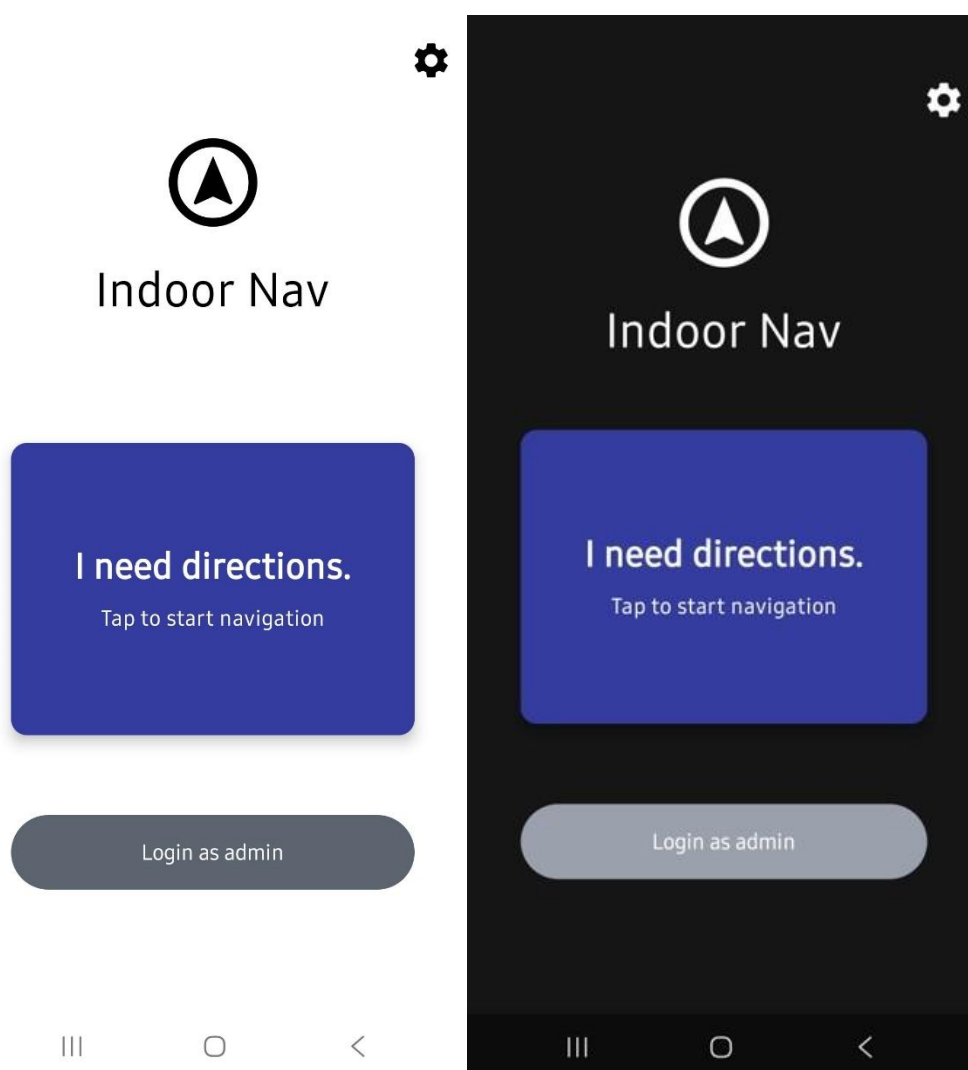
Η μεταβλητή `arrivalInstruction` αποκτά την οδηγία της συνάρτησης `generateNowInstruction()` για τον τελευταίο κόμβο, δηλαδή το τελικό μήνυμα «You have arrived at [X destination]». Το μήνυμα αποθηκεύεται στην `currentInstruction`, η μεταβλητή `hasFinishedNavigation` παίρνει την τιμή `true` και ενεργοποιείται το μοτίβο δόνησης `ARRIVAL` που ειδοποιεί τον χρήστη για άφιξη στον προορισμό. Τέλος, μέσω της συνάρτησης `speakAndThen()` εκφωνείται το τελικό μήνυμα και καλεί την `onNavigationFinished()`, η οποία επιστρέφει τον χρήστη στην αρχική οθόνη (`LandingScreen`). Με αυτό τον τρόπο, ολοκληρώνεται επιτυχώς η διαδικασία πλοήγησης.

5.5 Παρουσίαση διεπαφής και βασικών ροών αλληλεπίδρασης

Στην ενότητα αυτή παρουσιάζονται οι βασικές οθόνες της εφαρμογής και ο τρόπος σύνδεσης μεταξύ τους. Συγκεκριμένα, παρουσιάζονται η ροή του διαχειριστή, ο οποίος αναλαμβάνει την χαρτογράφηση και διαχείριση των κτηρίων και η ροή του χρήστη με οπτική αναπηρία, ο οποίος χρησιμοποιεί τη λειτουργία πλοήγησης της εφαρμογής.

5.5.1 Κοινή αρχική οθόνη (`LandingScreen`)

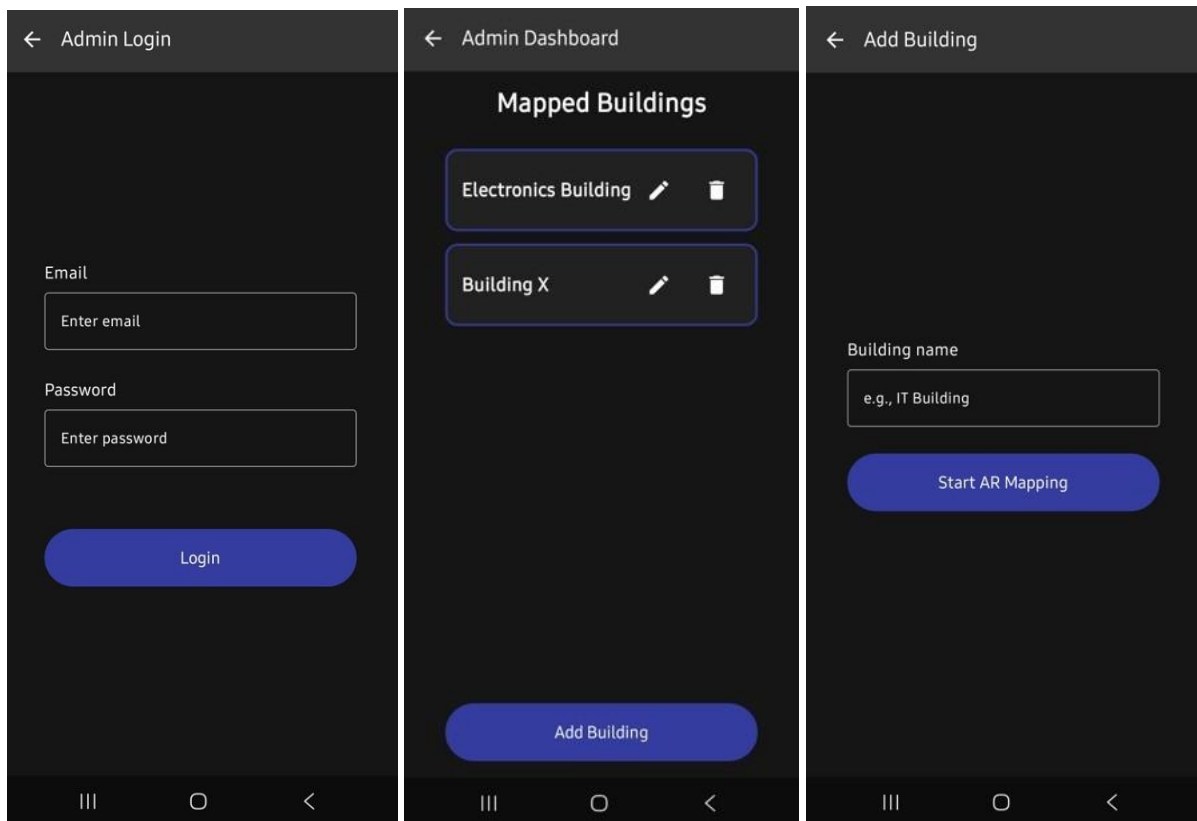
Με την εκκίνηση της εφαρμογής, εμφανίζεται η κοινή αρχική οθόνη. Σε αυτή, ο χρήστης μπορεί να διαλέξει ανάμεσα στις δύο λειτουργίες της εφαρμογής: χαρτογράφηση ή πλοήγηση. Για να μεταβεί στη λειτουργία της χαρτογράφησης, ο χρήστης επιλέγει το κουμπί «Login as admin» ώστε να συνδεθεί ως διαχειριστής και μεταφέρεται στην οθόνη σύνδεσης διαχειριστή (`AdminLoginScreen`). Αντίθετα, για να μεταβεί στη λειτουργία πλοήγησης, ο χρήστης επιλέγει το κουμπί στο κέντρο της οθόνης, όπως τον ενημερώνουν και οι αρχικές φωνητικές οδηγίες και μεταφέρεται στην οθόνη επιλογής κτηρίου (`DestinationPickerScreen`). Η `LandingScreen` λειτουργεί επομένως ως κοινό σημείο εισόδου στις δύο βασικές λειτουργίες του συστήματος, ενώ η περαιτέρω ροή διαφοροποιείται ανάλογα με την επιλογή του χρήστη. Η εφαρμογή υποστηρίζει επίσης ανοιχτόχρωμο και σκούρο θέμα, ακολουθώντας τη σχετική ρύθμιση της συσκευής. Για λόγους ομοιομορφίας, οι επόμενες οθόνες παρουσιάζονται στο σκούρο θέμα, ενώ η αρχική οθόνη παρουσιάζεται και στις δύο διαθέσιμες μορφές.



Σχήμα 5.45: Παρουσίαση αρχικής οθόνης (LandingScreen) στο light theme (αριστερά) και στο dark theme (δεξιά)

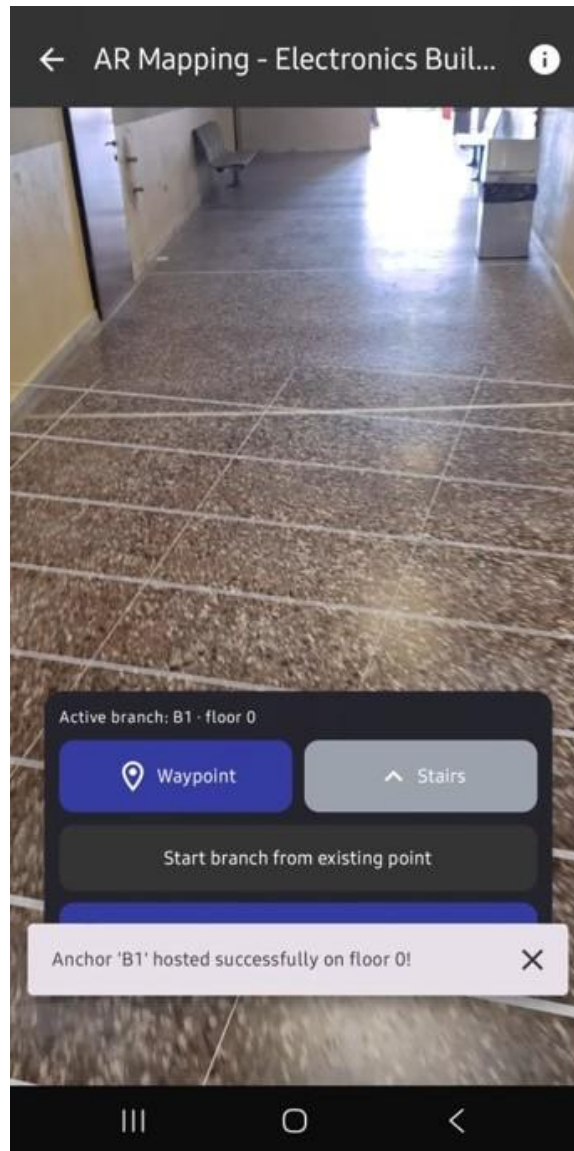
5.5.2 Διεπαφή διαχειριστή – Ροή χαρτογράφησης

Με την επιλογή της σύνδεσης ως διαχειριστής από την αρχική οθόνη, ο χρήστης εισάγει τα στοιχεία ταυτοποίησης του, ώστε να συνδεθεί στην εφαρμογή. Η οθόνη σύνδεσης του χρήστη ως διαχειριστή στην εφαρμογή, φαίνεται στο Σχήμα 5.46. Αφού συνδεθεί επιτυχώς, μεταφέρει στον πίνακα διαχείρισης (AdminDashboardScreen). Στον πίνακα διαχείρισης κτηρίων, εμφανίζεται το σύνολο των διαθέσιμων ήδη χαρτογραφημένων κτηρίων που έχουν αποθηκευτεί. Για κάθε κτήριο, ο διαχειριστής μπορεί είτε να μετονομάσει το κτήριο, είτε να το διαγράψει από την βάση. Για να προσθέσει νέο κτήριο, πατάει το κουμπί «Add Building» και μεταφέρεται στην οθόνη προσθήκης κτηρίου (AddBuildingScreen).



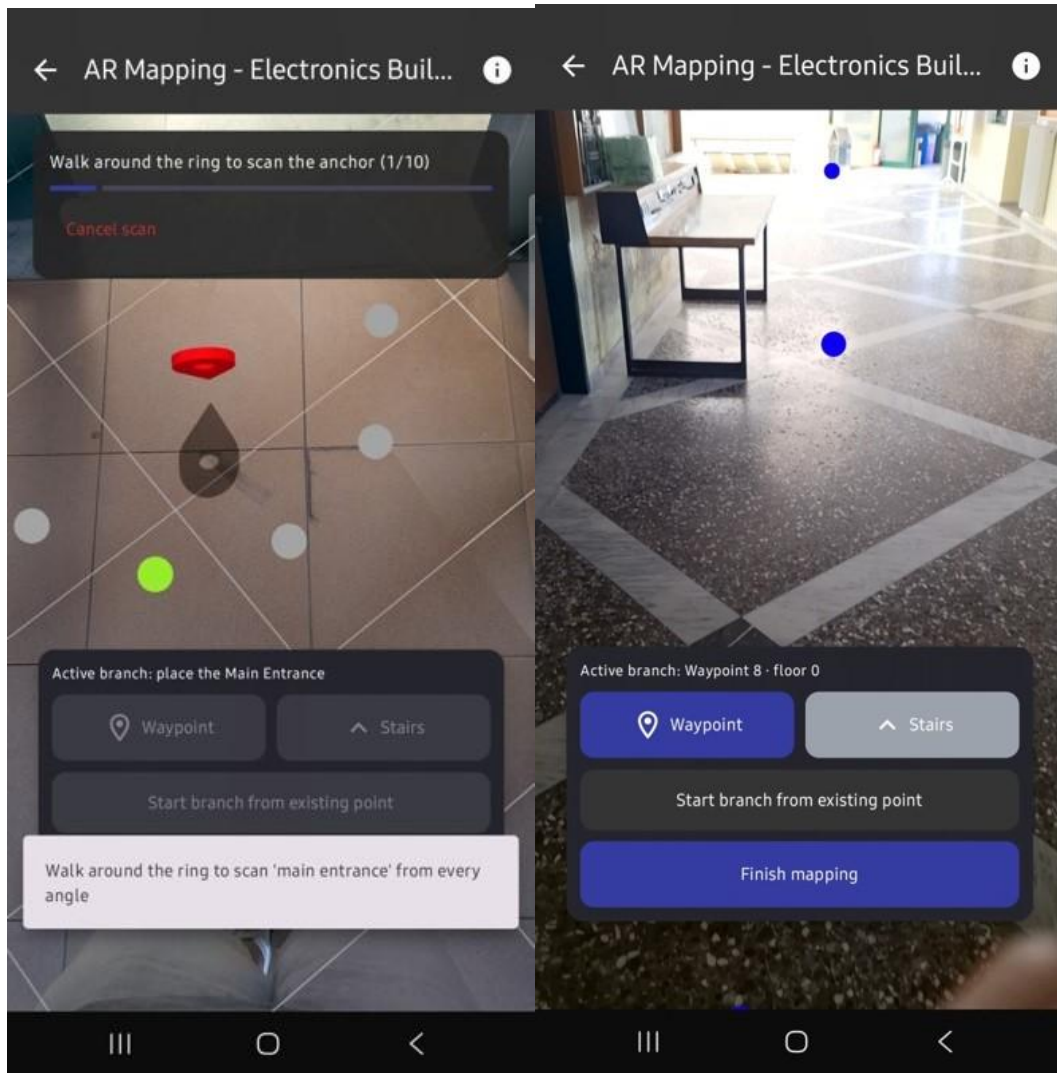
Σχήμα 5.46: Οθόνες AdminLoginScreen, AdminDashboardScreen, AddBuildingScreen

Στην οθόνη προσθήκης κτηρίου, ο διαχειριστής πληκτρολογεί το όνομα του κτηρίου που πρόκειται να χαρτογραφηθεί. Για την εισαγωγή του κτηρίου και ταυτόχρονα την έναρξη της χαρτογράφησης, ο διαχειριστής πατάει το κουμπί «Start AR Mapping», το οποίο τον μεταφέρει στην οθόνη χαρτογράφησης. Στην οθόνη χαρτογράφησης, αρχικά ανιχνεύονται τα επίπεδα μέσω της κάμερας. Ο διαχειριστής στη συνέχεια προσθέτει Cloud Anchors και Waypoints με τη μέθοδο που περιγράφηκε. Για την προσθήκη Waypoint στον χώρο, ο διαχειριστής επιλέγει το κουμπί «Waypoint» ενώ για προσθήκη Cloud Anchor πατά το σημείο στην οθόνη που αντιστοιχεί στο σημείο του χώρου που επιθυμεί. Για την προσθήκη σκάλας, πατά το κουμπί «Stairs» όπου του παρέχονται οδηγίες για τη διαδικασία αναφοράς σκάλας. Επιπλέον, για την προσθήκη διακλάδωσης, π.χ. σε διαδρόμους, πατάει το κουμπί «Start branch from existing point» όπου μπορεί να επιλέξει το σημείο από το οποίο θέλει να δημιουργηθεί η διακλάδωση. Παράδειγμα της οθόνης χαρτογράφησης, συγκεκριμένα των λειτουργιών που αναλύθηκαν παραπάνω, αλλά και της ανίχνευσης επιπέδου, παρουσιάζεται στο Σχήμα 5-47, μετά από δοκιμή στο κτήριο του τμήματος.



Σχήμα 5.47: Ανίχνευση επιπέδου και παρουσίαση λειτουργιών στην οθόνη χαρτογράφησης κτηρίου (ARMappingScreen)

Κατά την τοποθέτηση μιας Cloud Anchor, η εφαρμογή ακολουθεί τη διαδικασία χωρικής σάρωσης που απαιτείται για την επιτυχή φιλοξενία της άγκυρας στο Cloud. Ο διαχειριστής ακολουθεί τις οδηγίες που εμφανίζονται στην οθόνη και ολοκληρώνει τη διαδικασία σάρωσης του περιβάλλοντος. Με την επιτυχή ολοκλήρωση, η άγκυρα αποκτά το αντίστοιχο Cloud Anchor ID, το οποίο αποθηκεύεται ως αναγνωριστικό του κόμβου. Παράλληλα, τα Waypoints εμφανίζονται στην εικονική σκηνή και χρησιμοποιούνται για την πυκνότερη αναπαράσταση της επιτρεπόμενης πορείας. Με την τοποθέτηση διαδοχικών Waypoints και τη δημιουργία των αντίστοιχων συνδέσεων, η εφαρμογή σχηματίζει τη διαδρομή που θα χρησιμοποιηθεί αργότερα κατά την πλοήγηση. Παράδειγμα της σάρωσης περιβάλλοντος για την τοποθέτηση του Cloud Anchor, όπως και παράδειγμα τοποθέτησης και απεικόνισης των Waypoints παρουσιάζεται στο Σχήμα 5-50.

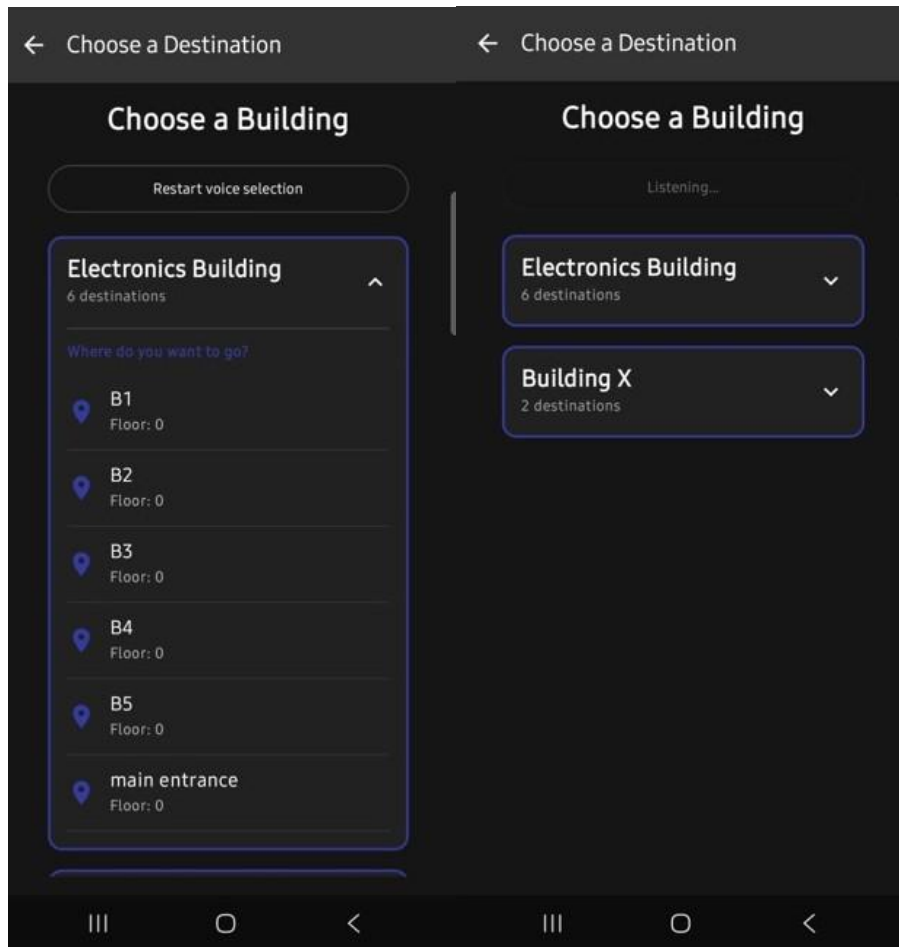


Σχήμα 5.48: Τοποθέτηση Cloud Anchors και Waypoints στο χώρο

5.5.3 Διεπαφή τελικού χρήστη - Ροή πλοήγησης

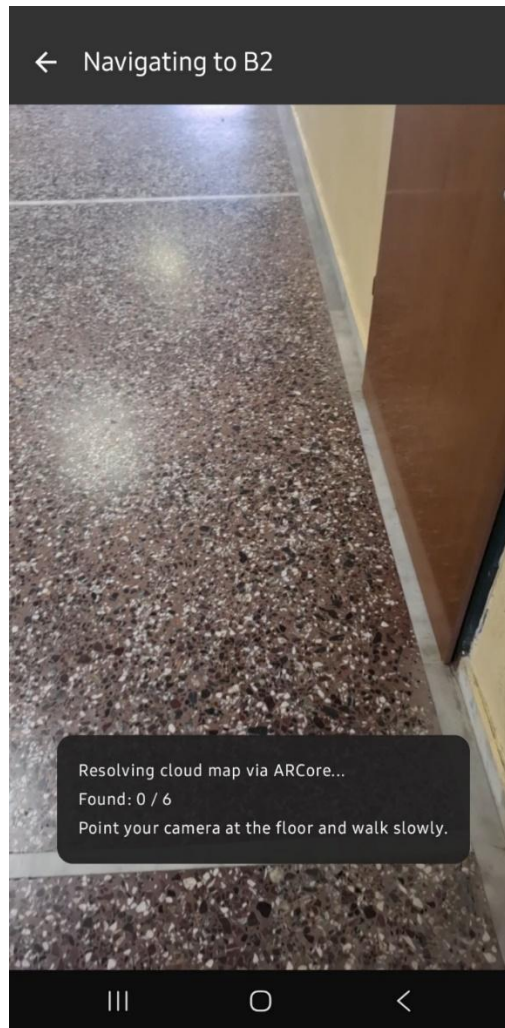
Με την επιλογή της λειτουργίας πλοήγησης από την αρχική οθόνη, ο τελικός χρήστης μεταφέρεται στην οθόνη επιλογής κτηρίου (DestinationPickerScreen). Αρχικά, εμφανίζονται τα διαθέσιμα χαρτογραφημένα κτήρια που έχουν αποθηκευτεί στο σύστημα. Ο χρήστης επιλέγει το κτήριο στο οποίο επιθυμεί να πλοηγηθεί και, στη συνέχεια, έναν από τους διαθέσιμους προορισμούς του συγκεκριμένου

κτηρίου. Η επιλογή του κτηρίου και του προορισμού μπορεί να πραγματοποιηθεί είτε μέσω της οθόνης είτε μέσω φωνητικών εντολών.



Σχήμα 5.49: Οθόνη επιλογής κτηρίου (αριστερά) και προορισμού (δεξιά) για πλοήγηση (DestinationPickerScreen)

Μετά την επιλογή του προορισμού, η εφαρμογή μεταβαίνει στην οθόνη πλοήγησης (ARNavigationScreen). Πριν ξεκινήσει η παροχή οδηγιών, πραγματοποιείται η διαδικασία εντοπισμού της τρέχουσας θέσης του χρήστη. Για τον σκοπό αυτό, η εφαρμογή επιχειρεί να επιλύσει τα αποθηκευμένα Cloud Anchors και να προσδιορίσει τη θέση και τον προσανατολισμό της συσκευής στο κοινό σύστημα συντεταγμένων του χαρτογραφημένου κτηρίου. Κατά τη διάρκεια της διαδικασίας αυτής, ο χρήστης ενημερώνεται μέσω της διεπαφής και των φωνητικών οδηγιών ότι πραγματοποιείται προσπάθεια εντοπισμού της θέσης του. Η διαδικασία συνεχίζεται μέχρι να επιτευχθεί επαρκής χωρικός εντοπισμός ή να προκύψει κατάσταση που δεν επιτρέπει την έναρξη της πλοήγησης.



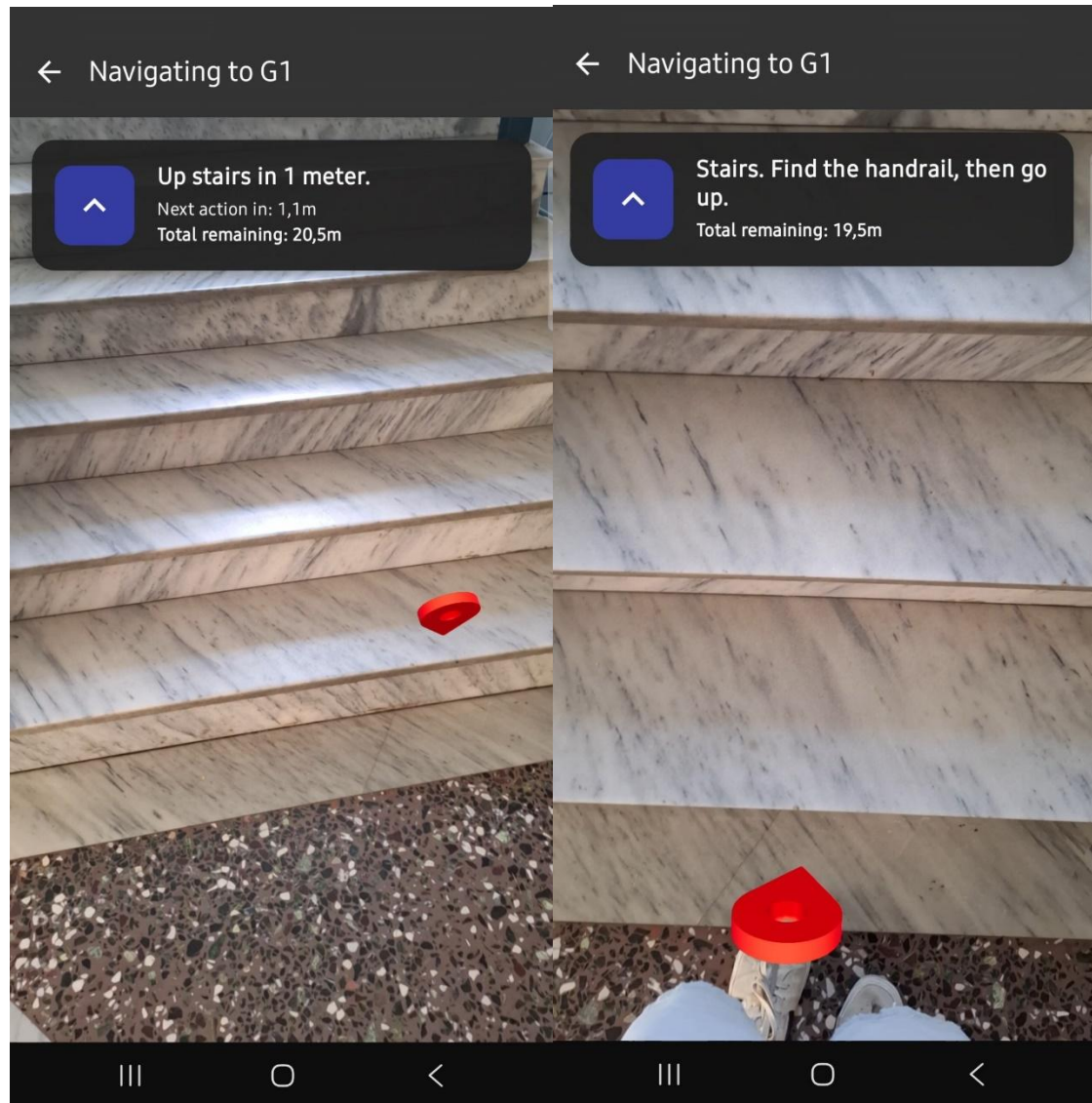
Σχήμα 5.50: Προσπάθεια εύρεσης της τρέχουσας θέσης του χρήστη, επιλύοντας τα Cloud Anchors

Υπάρχουν τέσσερις πιθανές καταστάσεις πλοήγησης. Μετά τον επιτυχή εντοπισμό της θέσης, η εφαρμογή εισέρχεται στην κανονική κατάσταση πλοήγησης. Σε αυτή την κατάσταση, η τρέχουσα θέση του χρήστη παρακολουθείται συνεχώς και συγκρίνεται με την υπολογισμένη διαδρομή προς τον επιλεγμένο προορισμό. Ανάλογα με την εξέλιξη της κίνησης, δημιουργούνται οι αντίστοιχες οδηγίες και παρέχονται στον χρήστη τόσο φωνητικά όσο και μέσω της διεπαφής. Η κατάσταση κανονικής πλοήγησης παρουσιάζεται στο Σχήμα 5-53.



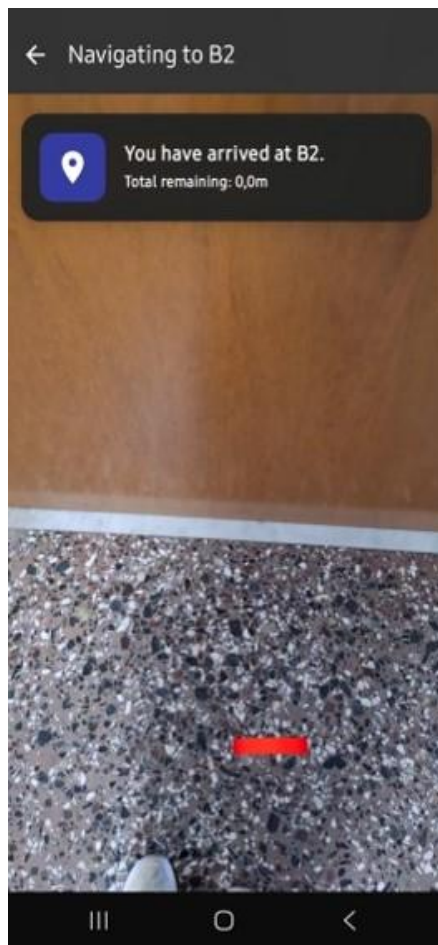
Σχήμα 5.51: Κατάσταση πλοήγησης α) κανονική πλοήγηση, όπου οι οδηγίες φαίνονται στην οθόνη

Μία ειδική περίπτωση της κανονικής πλοήγησης προκύπτει όταν η υπολογισμένη διαδρομή περιλαμβάνει μετάβαση μεταξύ διαφορετικών ορόφων μέσω σκάλας. Στην περίπτωση αυτή, η εφαρμογή αναγνωρίζει τη σύνδεση ως μετάβαση τύπου STAIRS_UP ή STAIRS_DOWN και προσαρμόζει αντίστοιχα τις οδηγίες που παρέχονται στον χρήστη. Πριν από την προσέγγιση της σκάλας, παρέχεται προειδοποίηση ώστε ο χρήστης να ενημερωθεί εγκαίρως για την επερχόμενη αλλαγή επιπέδου. Όταν ο χρήστης φτάσει στο σημείο της μετάβασης, παρέχεται πιο άμεση οδηγία για τη χρήση της σκάλας και την κατεύθυνση της μετάβασης. Με αυτόν τον τρόπο, η αλλαγή ορόφου αντιμετωπίζεται ως ειδικό στάδιο της διαδρομής και όχι ως μία απλή μετακίνηση μεταξύ δύο σημείων του ίδιου ορόφου. Παράδειγμα χρήσης παρουσιάζεται στο Σχήμα 5-54.



Σχήμα 5.52: Προειδοποίηση επερχόμενης σκάλας (αριστερά), ειδοποίηση για μετάβαση μέσω σκάλας άμεσα (δεξιά)

Η δεύτερη βασική κατάσταση της πλοήγησης αφορά την επιτυχή άφιξη στον προορισμό. Όταν η θέση του χρήστη προσεγγίσει επαρκώς τον τελικό κόμβο της υπολογισμένης διαδρομής, η εφαρμογή αναγνωρίζει την ολοκλήρωση της πλοήγησης και σταματά την κανονική παροχή οδηγιών. Παράλληλα, εμφανίζεται στην οθόνη επιβεβαιωτικό μήνυμα και παρέχεται αντίστοιχη φωνητική ενημέρωση, ώστε ο χρήστης να πληροφορηθεί ότι έφτασε στον προορισμό του.

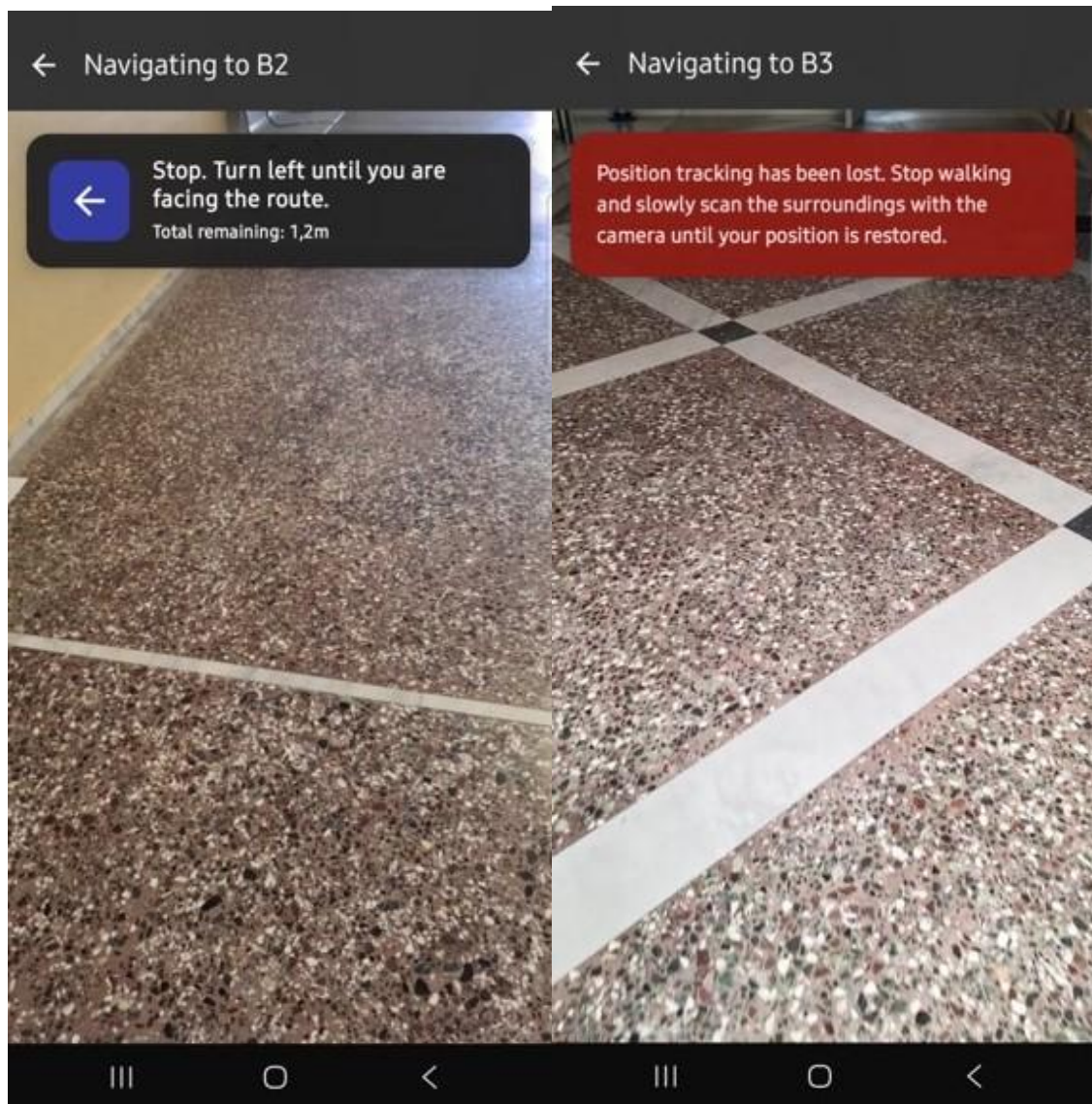


Σχήμα 5.53: Κατάσταση πλοήγησης β) Άφιξη στον προορισμό

Εκτός από την κανονική ολοκλήρωση της πλοήγησης, η εφαρμογή διαχειρίζεται δύο βασικές καταστάσεις σφάλματος, οι οποίες μπορούν να προκύψουν κατά τη μετακίνηση του χρήστη.

Η πρώτη αφορά την απόκλιση του χρήστη από την προτεινόμενη πορεία. Η εφαρμογή συγκρίνει τη θέση του χρήστη με την προβλεπόμενη διαδρομή και, όταν διαπιστωθεί ότι η απόσταση από την πορεία υπερβαίνει το επιτρεπόμενο όριο, μεταβαίνει στην αντίστοιχη κατάσταση ενημέρωσης. Ο χρήστης πληροφορείται ότι έχει αποκλίνει από την πορεία και λαμβάνει οδηγία για τη διόρθωση της κατεύθυνσής του. Η ενημέρωση παρέχεται τόσο φωνητικά όσο και οπτικά, διατηρώντας έτσι τον ίδιο τρόπο επικοινωνίας που χρησιμοποιείται και στην κανονική πλοήγηση. Αυτή η κατάσταση σφάλματος παρουσιάζεται στο Σχήμα 5-54 α.

Η δεύτερη κατάσταση σφάλματος αφορά την απώλεια της απαιτούμενης παρακολούθησης της κάμερας από το ARCore. Σε αυτή την περίπτωση, η χωρική θέση της συσκευής δεν μπορεί να θεωρηθεί επαρκώς αξιόπιστη για τη συνέχιση της πλοήγησης. Για τον λόγο αυτό, η εφαρμογή ενημερώνει τον χρήστη ότι η πλοήγηση δεν μπορεί να συνεχιστεί προσωρινά και του ζητά να σταματήσει και να σαρώσει προσεκτικά τον περιβάλλοντα χώρο. Στόχος της διαδικασίας είναι να αποκατασταθεί η χωρική παρακολούθηση και να καταστεί δυνατή η εκ νέου επίλυση ενός Cloud Anchor, ώστε να επαναπροσδιοριστεί η θέση της συσκευής. Αυτή η κατάσταση σφάλματος παρουσιάζεται στο Σχήμα 5-54 β.



Σχήμα 5.54: Καταστάσεις πλοήγησης γ) Ενημέρωση για απόκλιση από την πορεία (αριστερά) και δ) Απώλεια παρακολούθησης (δεξιά)

Κεφάλαιο 6ο: Αξιολόγηση εφαρμογής

Στο παρόν κεφάλαιο παρουσιάζεται η αξιολόγηση της εφαρμογής ως προς την εμπειρία χρήστη και την ευχρηστία. Αρχικά, περιγράφονται οι στόχοι και η μεθοδολογία αξιολόγησης, ενώ στη συνέχεια αναλύονται τα αποτελέσματα.

6.1 Στόχοι αξιολόγησης

Η αξιολόγηση αποσκοπεί στην διερεύνηση της λειτουργικότητας, αξιοπιστίας και ευχρηστίας της παρούσας εφαρμογής εσωτερικής πλοήγησης για άτομα με οπτική αναπηρία. Εξετάζεται το αν η παρούσα υλοποίηση ανταποκρίνεται στις απαιτήσεις που περιεγράφηκαν και επιδιώκεται ο εντοπισμός περιορισμών κατά τη χρήση της. Οι βασικοί στόχοι της αξιολόγησης είναι:

- Αξιολόγηση της καθοδήγησης: εξετάζεται η ορθότητα και σαφήνεια των φωνητικών οδηγιών και η χρησιμότητα της συμπληρωματικής απτικής ανάδρασης
- Διερεύνηση της ευχρηστίας κατά τη μη οπτική αλληλεπίδραση: εξετάζεται η δυνατότητα και ευκολία εκτέλεσης βασικών λειτουργιών, όπως η επιλογή κτηρίου και προορισμού μέσω φωνητικών εντολών και αναφέρονται δυσκολίες που εντοπίζονται κατά τη χρήση.
- Αξιολόγηση συνολικής εμπειρίας χρήστη: ερευνάται η ευκολία εκμάθησης, η προβλεψιμότητα και γενικότερα η εμπειρία του χρήστη.

6.2 Μεθοδολογία

Η αξιολόγηση πραγματοποιήθηκε με τη συμμετοχή 20 εθελοντών χωρίς προβλήματα όρασης. Η εφαρμογή δοκιμάστηκε σε ελεγχόμενο εσωτερικό χώρο, ο οποίος είχε νωρίτερα χαρτογραφηθεί. Για όλες τις δοκιμές χρησιμοποιήθηκε η ίδια κινητή συσκευή, Samsung Galaxy A54 5G, ώστε να διατηρηθούν σταθερές οι συνθήκες εκτέλεσης της εφαρμογής.

Πριν την έναρξη της αξιολόγησης, δόθηκαν στους συμμετέχοντες πληροφορίες σχετικά με τη λειτουργία της εφαρμογής. Δεν είχε προηγηθεί εκμάθηση των λειτουργιών ούτε είχε προηγηθεί η χρήση της νωρίτερα, ώστε να εξετασθεί η ευκολία κατανόησης της στην πρώτη επαφή.

Κατά την διάρκεια της δοκιμής, κάθε συμμετέχων κλήθηκε να επιλέξει το κτήριο και τον επιθυμητό προορισμό και να ακολουθήσει τις οδηγίες που παρείχε η εφαρμογή μέχρι την άφιξη στον προορισμό του. Για την επιλογή του κτηρίου χρησιμοποιήθηκε ο φωνητικός χειρισμός της εφαρμογής, ενώ κατά τη διάρκεια της πλοήγησης οι συμμετέχοντες ακολουθούσαν τις φωνητικές οδηγίες πλοήγησης και των συμπληρωματικών μοτίβων δόνησης.

Κατά τη διάρκεια της πλοήγησης, οι συμμετέχοντες διατηρούσαν τα μάτια τους κλειστά, ώστε να περιοριστεί η χρήση οπτικών πληροφοριών. Η πλοήγηση πραγματοποιήθηκε με τέτοιο τρόπο, ώστε να τονιστούν και εξεταστούν κυρίως οι φωνητικοί μηχανισμοί της εφαρμογής. Ωστόσο, η πειραματικές δοκιμές που πραγματοποιήθηκαν δεν αντικαθιστούν πλήρως την πραγματική εμπειρία ενός ατόμου με οπτική αναπηρία.

Αφού ολοκληρώθηκε η δοκιμή της εφαρμογής, οι συμμετέχοντες συμπλήρωσαν ανώνυμα ένα ηλεκτρονικό ερωτηματολόγιο. Το ερωτηματολόγιο περιλάμβανε τέσσερις ενότητες: δημογραφικά στοιχεία, ερωτήσεις σχετικά με την τεχνολογική εξοικείωση και την εμπειρία χρήσης των λειτουργιών προσβασιμότητας, αξιολόγηση της εμπειρίας χρήστη μέσω της ελληνικής μετάφρασης

του UEQ-S (User Experience Questionnaire – Short) και, τέλος, μία ανοιχτή ερώτηση σχετικά με παρατηρήσεις ή προτάσεις βελτίωσης.

Πιο συγκεκριμένα, στην ενότητα των δημογραφικών στοιχείων, καταγράφηκαν χαρακτηριστικά των συμμετεχόντων, όπως η ηλικία, το φύλο και το επίπεδο σπουδών. Παράλληλα, καταγράφηκε τυχόν προηγούμενη χρήση και εμπειρία των συμμετεχόντων με παρόμοιες εφαρμογές εσωτερικής πλοήγησης σε κινητές συσκευές, εμφανίζοντας τον βαθμό εξοικείωσης των συμμετεχόντων με παρόμοια συστήματα.

Στη συνέχεια, η ενότητα σχετικά με την τεχνολογική εξοικείωση και τις λειτουργίες προσβασιμότητας της εφαρμογής, είχε δύο ερωτήσεις. Η πρώτη αφορούσε τον βαθμό στον οποίο οι συμμετέχοντες θεωρούσαν κατανοητές τις φωνητικές οδηγίες κατά την διάρκεια της πλοήγησης, ενώ η δεύτερη αφορούσε την ευκολία του φωνητικού χειρισμού για την επιλογή κτηρίου και προορισμού. Οι συγκεκριμένες ερωτήσεις αποσκοπούν στην περεταίρω αποτίμηση των συγκεκριμένων λειτουργιών προσβασιμότητας.

Για την αξιολόγηση της εμπειρίας χρήστη χρησιμοποιήθηκε το UEQ-S. Το User Experience Questionnaire (UEQ) αποτελεί ένα τυποποιημένο ερωτηματολόγιο για την αξιολόγηση εμπειρίας χρήστη (User Experience – UX). Αναπτύχθηκε από τους Laugwitz, Schrepp και Held το 2006, με σκοπό την αποτίμηση διαφορετικών στοιχείων της εμπειρίας που αποκτά ένας χρήστης κατά την αλληλεπίδραση του με ένα προϊόν ή σύστημα [48]. Η αρχική, πλήρης μορφή του ερωτηματολογίου περιλαμβάνει 26 στοιχεία αξιολόγησης, τα οποία παρουσιάζονται ως ζεύγη αντίθετων χαρακτηρισμών.

Για την παρούσα αξιολόγηση χρησιμοποιήθηκε μία σύντομη έκδοση του ερωτηματολογίου, η UEQ-S, η οποία αποτελείται από οκτώ ζευγάρια αντίθετων χαρακτηρισμών και επιτρέπει την σύντομη αποτίμηση της εμπειρίας χρήστη. Χρησιμοποιήθηκε επταβάθμια κλίμακα. Οι απαντήσεις με τιμές από 1 έως 7. Η τιμή 4 αντιστοιχεί στην ουδέτερη θέση, ενώ οι τιμές 1 και 7 αντιστοιχούν στα άκρα του εκάστοτε ζεύγους. Τα ζεύγη χαρακτηρισμών που περιέχει το UEQ-S είναι τα παρακάτω:

- Αδιάφορη – Ενδιαφέρουσα
- Συμβατική – Εφευρετική
- Παρεμποδιστική – Υποστηρικτική
- Περίπλοκη – Εύκολη
- Βαρετή - Συναρπαστική
- Συνηθισμένη – Πρωτοπόρα
- Ανεπαρκής – Επαρκής
- Σαφής – Μπερδεμένη

Τέλος, το ερωτηματολόγιο περιείχε μία ανοιχτή ερώτηση, μέσω της οποίας οι συμμετέχοντες είχαν τη δυνατότητα να καταγράψουν σχόλια, παρατηρήσεις ή προτάσεις βελτίωσης σχετικά με κάποιο μέρος της εφαρμογής είτε το σύστημα ως σύνολο .

6.3 Αποτελέσματα

Στην παρούσα υπό-ενότητα παρουσιάζονται τα αποτελέσματα που προέκυψαν από την αξιολόγηση της εφαρμογής με συμμετοχή των 20 εθελοντών. Αρχικά, παρουσιάζονται τα βασικά χαρακτηριστικά του δείγματος και η προηγούμενη εμπειρία των συμμετεχόντων με παρόμοιες εφαρμογές. Έπειτα, τα αποτελέσματα των ερωτήσεων σχετικά με τη φωνητική καθοδήγηση και τον φωνητικό χειρισμό της

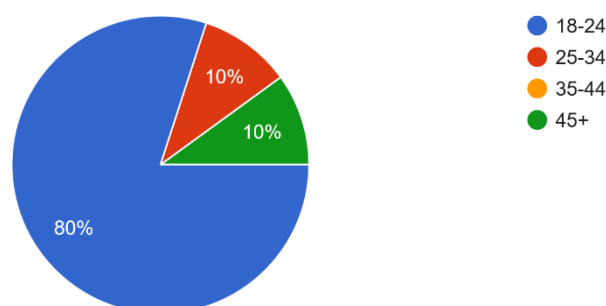
εφαρμογής. Ακολουθεί η παρουσίαση και ανάλυση των αποτελεσμάτων UEQ-S και τέλος, παρουσιάζονται προτάσεις βελτίωσης και παρατηρήσεις.

6.3.1 Δημογραφικά χαρακτηριστικά

6.3.1.1 Ηλικία συμμετεχόντων

Η πρώτη ερώτηση του ερωτηματολογίου καλούσε τους συμμετέχοντες να αναφέρουν την ηλικία τους. Οι διαθέσιμες ηλικιακές κατηγορίες ήταν: α) 18-24, β) 25-34, γ) 35-44, δ) 45+. Από τους 20 συμμετέχοντες, οι 16 (80%) ανήκαν στην ηλικιακή κατηγορία των 18-24, οι 2 (10%) στην κατηγορία 25-34, ενώ 2 άτομα (10%) δήλωσαν 45+.

Ηλικιακή Ομάδα
20 απαντήσεις

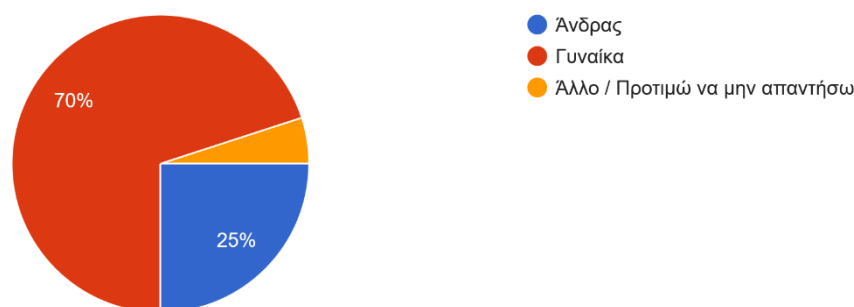


Σχήμα 6.1: Διάγραμμα αποτελεσμάτων στην ερώτηση "Σε ποια ηλικιακή ομάδα ανήκετε;"

6.3.1.2 Φύλο συμμετεχόντων

Η δεύτερη ερώτηση ρωτούσε τους συμμετέχοντες σχετικά με τον προσδιορισμό του φύλου τους. Τα 14 άτομα (70%) ήταν γυναίκες, τα 5 (25%) άνδρες, ενώ 1 άτομο (5%) επέλεξε να μην προσδιορίσει το φύλο του.

Φύλλο
20 απαντήσεις

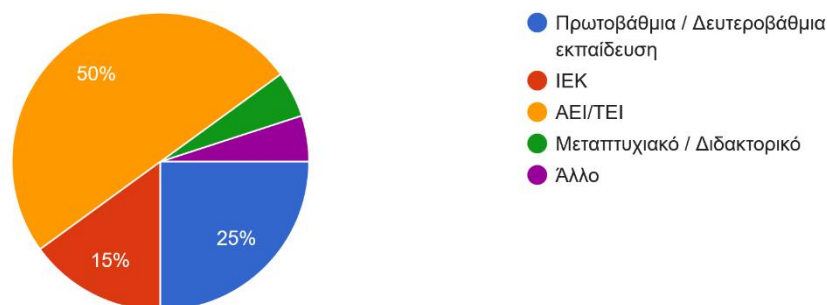


Σχήμα 6.2: Διάγραμμα απαντήσεων ερώτησης προσδιορισμού του φύλου των συμμετεχόντων

6.3.1.3 Επίπεδο σπουδών

Η Τρίτη ερώτηση του ερωτηματολογίου αφορούσε το επίπεδο σπουδών των συμμετεχόντων. Οι διαθέσιμες κατηγορίες ήταν α) Πρωτοβάθμια / Δευτεροβάθμια, β) ΙΕΚ, γ) ΑΕΙ/ΤΕΙ, δ) Μεταπτυχιακό / Διδακτορικό, ε) Άλλο. Από τους 20 συμμετέχοντες, οι 5 (25%) ανήκαν στην κατηγορία «Πρωτοβάθμια / Δευτεροβάθμια», οι 3 (15%) στην κατηγορία «ΙΕΚ», οι 10 (50%) ανήκαν στην κατηγορία «ΑΕΙ/ΤΕΙ», ένας (5%) στην κατηγορία «Μεταπτυχιακό / Διδακτορικό», ενώ ένας συμμετέχων (5%) επέλεξε την κατηγορία «Άλλο»,

Επίπεδο σπουδών
20 απαντήσεις



Σχήμα 6.3: Διάγραμμα απαντήσεων σχετικά με το επίπεδο σπουδών των συμμετεχόντων

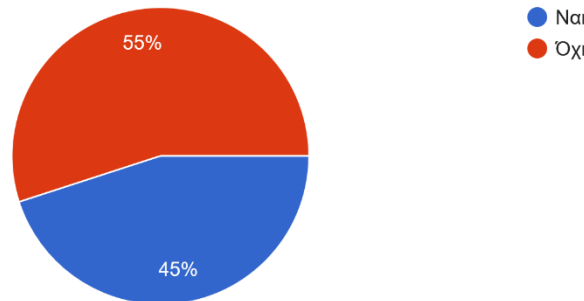
6.3.2 Τεχνολογική εξοικείωση και προσβασιμότητα

6.3.2.1 Προηγούμενη εμπειρία με παρόμοιες εφαρμογές

Η πρώτη ερώτηση αυτής της ενότητας, ρωτούσε τους συμμετέχοντες σχετικά με το αν είχαν χρησιμοποιήσει στο παρελθόν παρόμοιες εφαρμογές σε κινητό τηλέφωνο ή tablet. Από το σύνολο των 20 συμμετεχόντων, 9 άτομα (45%) δήλωσαν ότι είχαν προηγούμενη εμπειρία με παρόμοιες εφαρμογές, ενώ 11 άτομα (55%) δήλωσαν πως δεν είχαν χρησιμοποιήσει αντίστοιχη εφαρμογή στο παρελθόν.

Έχετε χρησιμοποιήσει στο παρελθόν παρόμοιες εφαρμογές σε κινητό ή tablet; (εφαρμογές εσωτερικής πλοήγησης)

20 απαντήσεις



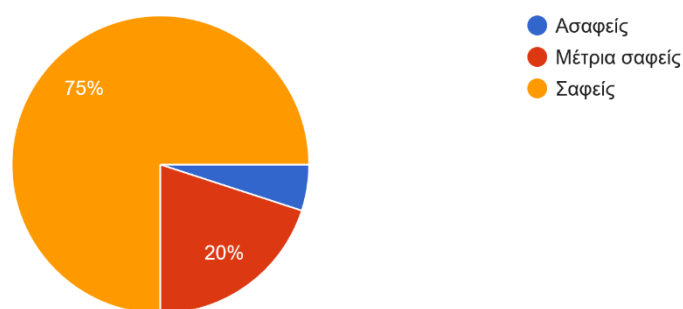
Σχήμα 6.4: Διάγραμμα απαντήσεων στην ερώτηση: "Έχετε χρησιμοποιήσει στο παρελθόν παρόμοιες εφαρμογές σε κινητό ή tablet; (εφαρμογές εσωτερικής πλοήγησης);"

6.3.2.2 Βαθμός κατανόησης φωνητικών οδηγιών κατά την πλοήγηση

Η δεύτερη ερώτηση αφορούσε τον βαθμό κατανόησης των φωνητικών οδηγιών, που παρείχε το TTS κατά την πλοήγηση. Οι συμμετέχοντες κλήθηκαν να απαντήσουν κατά πόσο οι οδηγίες προς τον προορισμό τους ήταν σαφείς. Από το σύνολο των 20 συμμετεχόντων, 15 άτομα (75%) χαρακτήρισαν τις φωνητικές οδηγίες ως «Σαφείς», 4 άτομα (20%) ως «Μέτρια σαφείς», ενώ 1 άτομο (5%) τις χαρακτήρισε «Ασαφείς».

Πόσο κατανοητές βρήκατε τις φωνητικές οδηγίες πλοήγησης της εφαρμογής (Text-to-Speech) κατά τη διάρκεια της πορείας σας;

20 απαντήσεις



Σχήμα 6.5: Διάγραμμα απαντήσεων στην ερώτηση: "Πόσο κατανοητές βρήκατε τις φωνητικές οδηγίες πλοήγησης της εφαρμογής (Text-to-Speech) κατά τη διάρκεια της πορείας σας; "

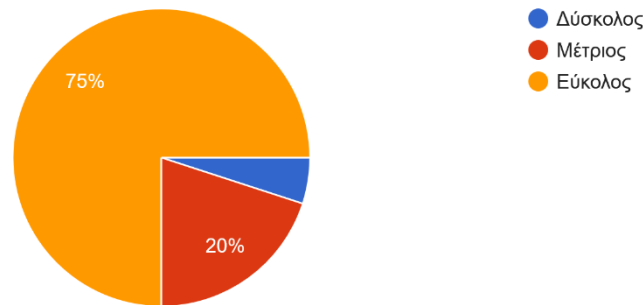
6.3.2.3 Βαθμός ευχρηστίας του φωνητικού χειρισμού της εφαρμογής

Η Τρίτη ερώτηση της ενότητας, αφορούσε την ευκολία χρήσης του φωνητικού χειρισμού της εφαρμογής για την επιλογή κτηρίου και προορισμού μέσω ομιλίας. Από το σύνολο των 20

συμμετεχόντων, 15 άτομα (75%) χαρακτήρισαν τον φωνητικό χειρισμό ως «Εύκολο», 4 άτομα (20%) ως «Μέτριο» και ένα άτομο (5%) ως «Δύσκολο».

Πόσο εύκολος σας φάνηκε ο φωνητικός χειρισμός της εφαρμογής (επιλογή κτηρίου/προορισμού μέσω ομιλίας) σε σύγκριση με τα οπτικά στοιχεία της οθόνης;

20 απαντήσεις



Σχήμα 6.6: Διάγραμμα απαντήσεων στην ερώτηση "Πόσο εύκολος σας φάνηκε ο φωνητικός χειρισμός της εφαρμογής (επιλογή κτηρίου/προορισμού μέσω ομιλίας) σε σύγκριση με τα οπτικά στοιχεία της οθόνης; "

6.3.3 Αποτελέσματα ερωτηματολογίου UEQ-S

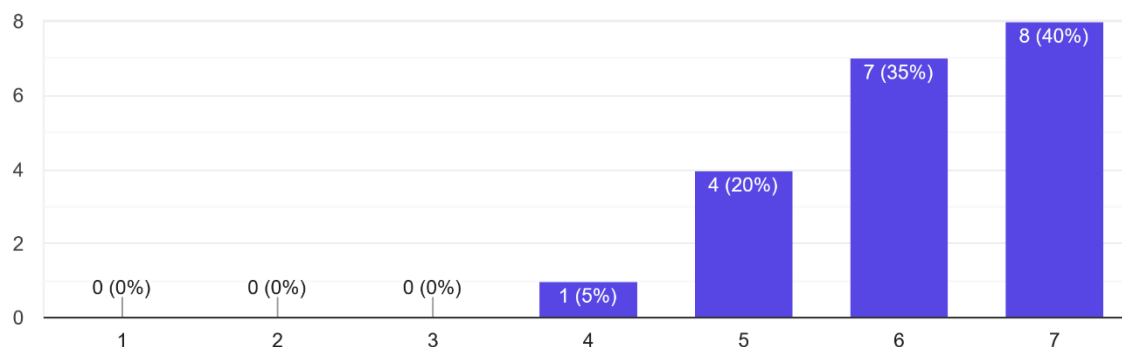
Έπειτα, οι χρήστες κλήθηκαν να αξιολογήσουν την συνολική εμπειρία χρήσης τους, μέσω του ερωτηματολογίου UEQ-S. Όπως αναφέρθηκε, η εφαρμογή αξιολογήθηκε μέσω οκτώ ζευγών αντίθετων χαρακτηρισμών, χρησιμοποιώντας επταβάθμια κλίμακα.

6.3.3.1 Βαρετή – Συναρπαστική

Στο πρώτο ζεύγος χαρακτηριστικών, οι συμμετέχοντες αξιολόγησαν κατά πόσο θεωρούσαν την εφαρμογή βαρετή ή συναρπαστική. Από τους 20 συμμετέχοντες, 8 άτομα (40%) επέλεξαν την τιμή 7, 7 άτομα επέλεξαν την τιμή 6 (35%), 4 άτομα (20%) την τιμή 5, ενώ 1 άτομο (5%) επέλεξε την ουδέτερη τιμή 4.

Χαρακτηρίστε την εφαρμογή

20 απαντήσεις



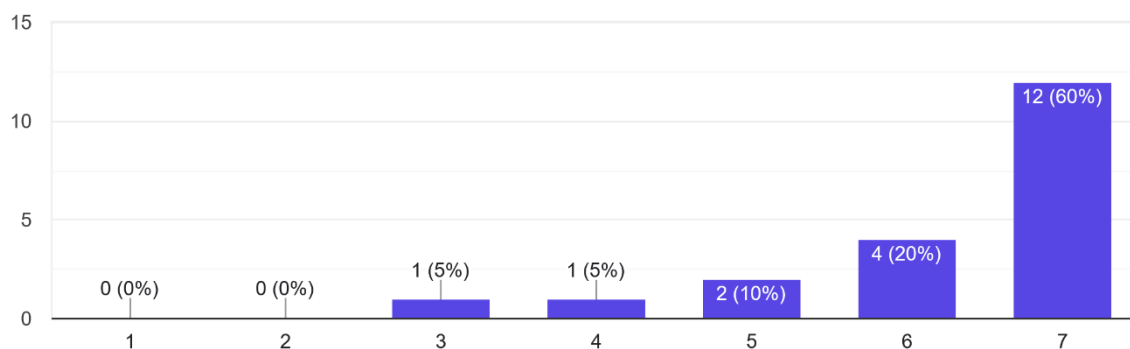
Σχήμα 6.7: Αποτελέσματα αξιολόγησης εφαρμογής ως προς το ζεύγος χαρακτηρισμών "Βαρετή - Συναρπαστική"

6.3.3.2 Αδιάφορη - Ενδιαφέρουσα

Η δεύτερη ερώτηση αφορούσε το πόσο αδιάφορη ή ενδιαφέρουσα βρήκαν οι συμμετέχοντες την εφαρμογή. Από τους 20 συμμετέχοντες, 12 άτομα (60%) επέλεξαν την τιμή 7, 4 άτομα (20%) την τιμή 6, 2 άτομα (10%) την τιμή 5, τις τιμές 4 και 3 επέλεξε από 1 άτομο (5%).

Χαρακτηρίστε την εφαρμογή

20 απαντήσεις



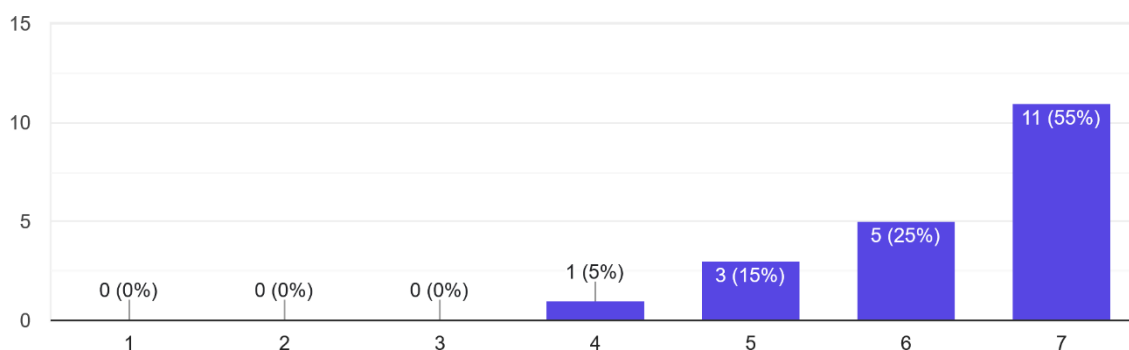
Σχήμα 6.8: Αποτελέσματα αξιολόγησης εφαρμογής ως προς το ζεύγος χαρακτηρισμών "Αδιάφορη - Ενδιαφέρουσα"

6.3.3.3 Συμβατική – Εφευρετική

Το τρίτο ζεύγος χαρακτηρισμών αφορούσε την αξιολόγηση της εφαρμογής ως προς την συμβατικότητα ή εφευρετικότητα της. Από τους 20 συμμετέχοντες, 11 άτομα (55%) επέλεξαν την τιμή 7, 5 άτομα (25%) την τιμή 6, 3 άτομα (15%) την τιμή 5, ενώ 1 άτομο (5%) επέλεξε την τιμή 4.

Χαρακτηρίστε την εφαρμογή

20 απαντήσεις



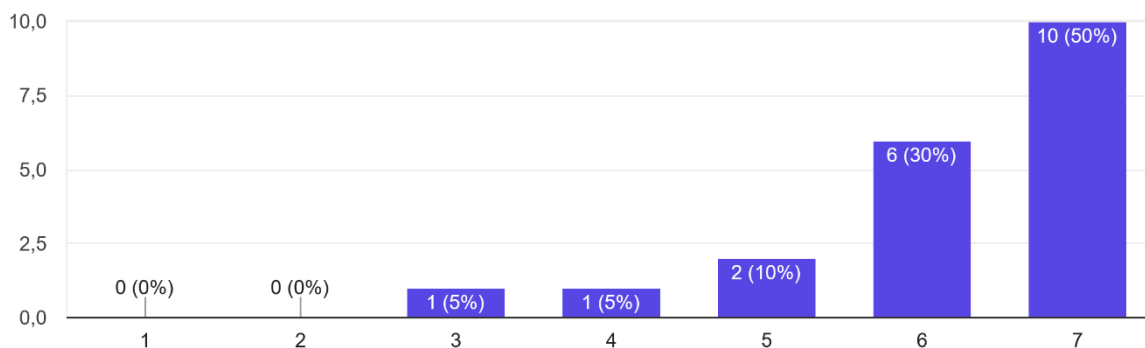
Σχήμα 6.9: Αποτελέσματα αξιολόγησης εφαρμογής ως προς το ζεύγος χαρακτηρισμών "Συμβατική - Εφευρετική"

6.3.3.4 Παρεμποδιστική – Υποστηρικτική

Στο τέταρτο ζεύγος εξετάστηκε το πόσο υποστηρικτική ήταν η εφαρμογή κατά τη χρήση της. Συγκεκριμένα, 10 άτομα (50%) επέλεξαν την τιμή 7, 6 άτομα (30%) την τιμή 6, 2 άτομα (10%) την τιμή 5, ενώ από 1 άτομο (5%) επέλεξε την τιμή 3 και την τιμή 4.

Χαρακτηρίστε την εφαρμογή

20 απαντήσεις

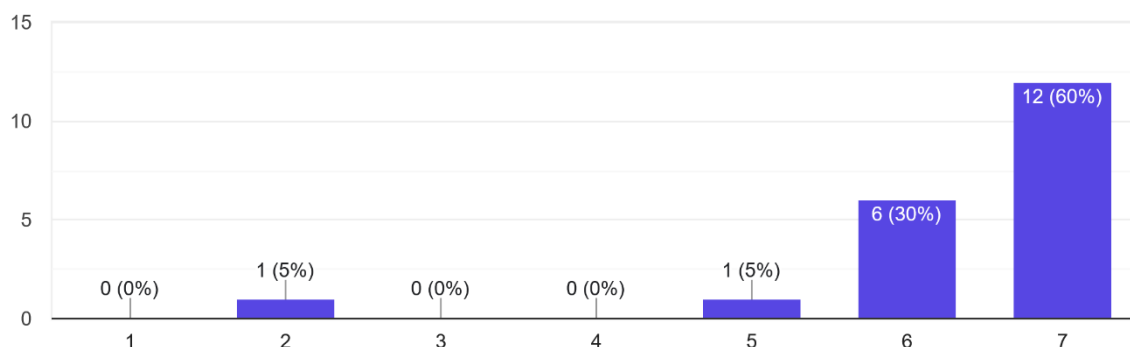


Σχήμα 6.10: Αποτελέσματα αξιολόγησης εφαρμογής ως προς το ζεύγος χαρακτηρισμών "Παρεμποδιστική - Υποστηρικτική"

6.3.3.5 Περίπλοκη – Εύκολη

Το πέμπτο ζεύγος χαρακτηρισμών αφορούσε την αξιολόγηση της εφαρμογής ως προς την περιπλοκότητα ή ευκολία της. Από τους 20 συμμετέχοντες, 12 άτομα (60%) επέλεξαν την τιμή 7, 6 άτομα (30%) την τιμή 6, 1 άτομο (5%) την τιμή 5 και επίσης 1 άτομο (5%) την τιμή 2.

Χαρακτηρίστε την εφαρμογή
20 απαντήσεις

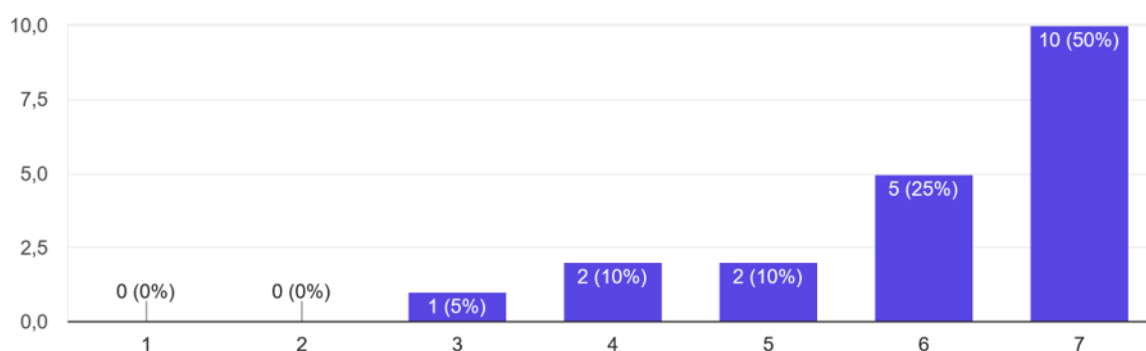


Σχήμα 6.11: Αποτελέσματα αξιολόγησης εφαρμογής ως προς το ζεύγος χαρακτηρισμών "Περίπλοκη - Εύκολη"

6.3.3.6 Συνηθισμένη – Πρωτοπόρα

Στο έκτο ζεύγος χαρακτηρισμών εξετάστηκε κατά πόσο η εφαρμογή θεωρήθηκε συνηθισμένη ή πρωτοπόρα. Από τους 20 συμμετέχοντες, 10 άτομα (50%) επέλεξαν την τιμή 7, 5 άτομα (25%) την τιμή 6, 2 άτομα (10%) τις τιμές 5 και 4, ενώ 1 άτομο επέλεξε την τιμή 3.

Χαρακτηρίστε την εφαρμογή
20 απαντήσεις



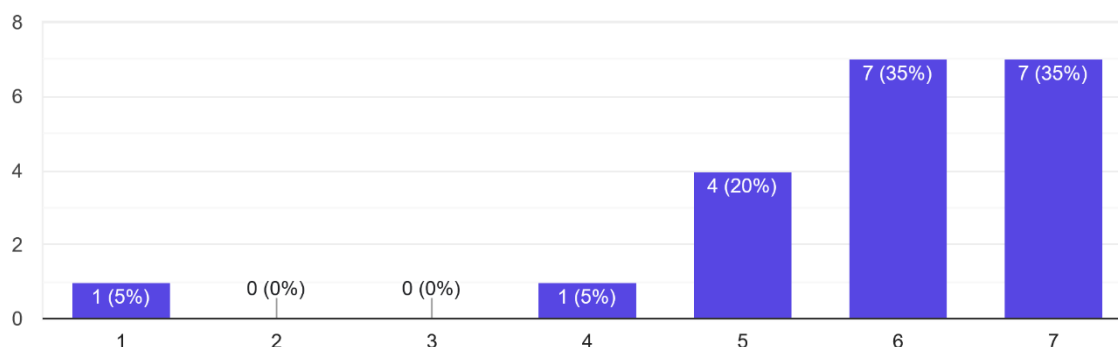
Σχήμα 6.12: Αποτελέσματα αξιολόγησης εφαρμογής ως προς το ζεύγος χαρακτηρισμών "Συνηθισμένη - Πρωτοπόρα"

6.3.3.7 Ανεπαρκής – Επαρκής

Στο έβδομο ζεύγος χαρακτηρισμών εξετάστηκε κατά πόσο οι συμμετέχοντες θεώρησαν την εφαρμογή ανεπαρκή ή επαρκή. Από τους 20 συμμετέχοντες, από 7 άτομα (35%) επέλεξαν τις τιμές 7 και 6 αντίστοιχα, 4 άτομα (20%) την τιμή 5, ενώ 1 άτομο (5%) επέλεξε την τιμή 4 και 1 άτομο (5%) επέλεξε την τιμή 1.

Χαρακτηρίστε την εφαρμογή

20 απαντήσεις



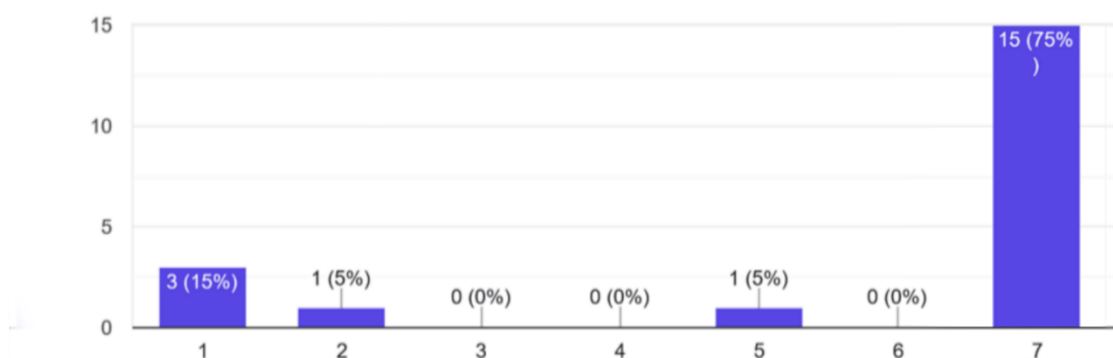
Σχήμα 6.13: Αποτελέσματα αξιολόγησης εφαρμογής ως προς το ζεύγος χαρακτηρισμών "Ανεπαρκής - Επαρκής"

6.3.3.8 Μπερδεμένη – Σαφής

Τέλος, το όγδοο ζεύγος χαρακτηρισμών αφορούσε την σαφήνεια της εφαρμογής. Από τους 20 συμμετέχοντες, 15 άτομα (75%) επέλεξαν την τιμή 7, από 1 άτομο (5%) τις τιμές 5 και 2, ενώ 3 άτομα επέλεξαν την τιμή 3.

Χαρακτηρίστε την εφαρμογή

20 απαντήσεις



Σχήμα 6.14: Αποτελέσματα αξιολόγησης εφαρμογής ως προς το ζεύγος χαρακτηρισμών "Σαφής - Μπερδεμένη"

6.3.4 Συνοπτική στατιστική ανάλυση UEQ-S

Από τις απαντήσεις στα οκτώ ζεύγη χαρακτηρισμών, υπολογίστηκαν στατιστικά μεγέθη για την συνοπτική αποτίμηση των αποτελεσμάτων του UEQ-S. Συγκεκριμένα, για κάθε ζεύγος χαρακτηρισμών, υπολογίστηκε η μέση τιμή και η τυπική απόκλιση των απαντήσεων. Η μέση τιμή εμφανίζει τη συνολική, κεντρική τάση των απαντήσεων, ενώ η τυπική απόκλιση παρουσιάζει τη διασπορά των απαντήσεων γύρω από τη μέση τιμή. Σύμφωνα με τη μεθοδολογία του UEQ-S, οι τιμές των επιμέρους χαρακτηρισμών μετασχηματίζονται σε κλίμακα από -3 έως +3, όπου οι αρνητικές τιμές υποδηλώνουν αρνητική αξιολόγηση και οι θετικές τιμές θετική αξιολόγηση. Τιμές μεταξύ -0,8 και +0,8 θεωρούνται ουδέτερες, ενώ τιμές μεγαλύτερες από +0,8 υποδηλώνουν θετική αξιολόγηση. Τα αποτελέσματα παρουσιάζονται στον Πίνακα 6-1.

Ζεύγος χαρακτηρισμών	Μέση τιμή	Τυπική απόκλιση
Βαρετή - Συναρπαστική	2,4	1,2
Αδιάφορη – Ενδιαφέρουσα	2,1	1,2
Συμβατική – Εφευρετική	1,8	1,4
Παρεμποδιστική – Υποστηρικτική	2,1	0,9
Περίπλοκη – Εύκολη	2,3	1,2
Συνηθισμένη – Πρωτοπόρα	1,8	2,4
Ανεπαρκής – Επαρκής	2,3	0,9
Μπερδεμένη - Σαφής	2,2	1,1

Πίνακας 6.1: Παρουσίαση μέσης τιμής και τυπικής απόκλισης για κάθε ζεύγος χαρακτηρισμών

Από τα παραπάνω αποτελέσματα προκύπτει πως η εφαρμογή αξιολογήθηκε συνολικά θετικά από τους συμμετέχοντες, με τη μέση τιμή κάθε ζεύγους χαρακτηρισμών βρίσκονται σημαντικά πάνω από το όριο του +0,8. Ως προς τα θετικά ευρήματα, η υψηλότερη μέση τιμή παρατηρείται στο ζεύγος «Βαρετή – Συναρπαστική» με τιμή 2,4 και τυπική απόκλιση 1,2, γεγονός που δείχνει πως για τους περισσότερους χρήστες η εφαρμογή ήταν ενδιαφέρουσα και ευχάριστη κατά τη χρήση. Επιπλέον, η μέση τιμή των ζευγών «Περίπλοκη – Εύκολη», με μέση τιμή 2,3, και «Ανεπαρκής – Επαρκής», επίσης με μέση τιμή 2,3, υποδεικνύουν ότι οι συμμετέχοντες θεωρούν την αλληλεπίδραση με την εφαρμογή σχετικά εύκολη και τις παρεχόμενες λειτουργίες επαρκείς για την εκτέλεση των βασικών ενεργειών της. Θετικά αποτελέσματα έλαβαν επίσης τα ζεύγη «Μπερδεμένη – Σαφής» με μέση τιμή 2,2, καθώς και «Αδιάφορη – Ενδιαφέρουσα» και «Παρεμποδιστική – Υποστηρικτική», με μέση τιμή 2,1.

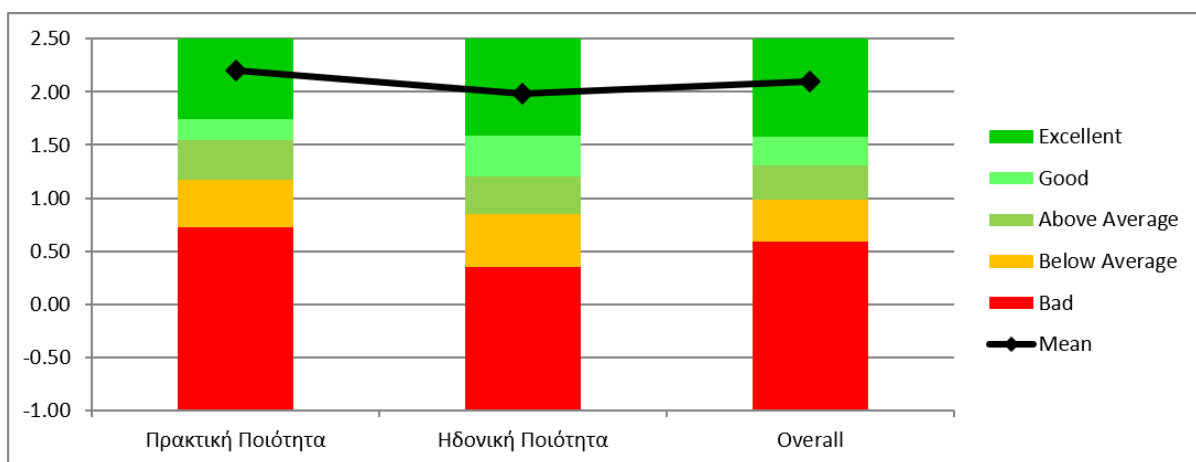
Οι χαμηλότερες, αν και σαφώς θετικές, μέσες τιμές παρατηρούνται στα ζεύγη «Συμβατική – Εφευρετική» και «Συνηθισμένη – Πρωτοπόρα», με τιμή 1,8. Συγκεκριμένα, η τυπική απόκλιση στο δεύτερο ζεύγος είναι 2,3, γεγονός που υποδεικνύει μεγάλη διαφοροποίηση στις απόψεις των συμμετεχόντων σχετικά με τον χαρακτηρισμό της εφαρμογής ως πρωτοποριακής. Επομένως, το συγκεκριμένο εύρημα μπορεί να αποτελέσει πεδίο περαιτέρω βελτίωσης, κυρίως ως προς τον τρόπο με τον οποίο η εφαρμογή ξεχωρίζει από αντίστοιχες λύσεις.

Συνολικά, τα αποτελέσματα του ερωτηματολογίου επιβεβαιώνουν την θετική εμπειρία χρήστη. Υψηλές αξιολογήσεις καταγράφηκαν ως προς το ενδιαφέρον, την πρακτικότητα και τον υποστηρικτικό χαρακτήρα της εφαρμογής. Παράλληλα, οι συμμετέχοντες αξιολόγησαν θετικά την κατανόηση και την καινοτομία της εφαρμογής. Ωστόσο, λόγω του περιορισμένου αριθμού συμμετεχόντων, καθώς και της έλλειψης συμμετεχόντων με προβλήματα όρασης, τα αποτελέσματα αποτυπώνουν ενδεικτικά τις απόψεις των συμμετεχόντων για τα βασικά μέρη της εφαρμογής και δεν πρέπει να αποτελέσουν γενικεύσεις για τις προτιμήσεις και απαιτήσεις της ομάδας για την οποία προορίζεται το σύστημα, δηλαδή άτομα με οπτική αναπηρία.

6.3.5 Σύγκριση με το UEQ Benchmark

Με σκοπό την περαιτέρω αξιολόγηση της εμπειρίας χρήστη, πραγματοποιήθηκε σύγκριση των αποτελεσμάτων της εφαρμογής με το ενσωματωμένο benchmark του UEQ. Η σύγκριση πραγματοποιείται σε επίπεδο Πρακτικής Ποιότητας (Pragmatic Quality), Ηδονικής Ποιότητας (Hedonic Quality) και συνολικής αξιολόγησης (Overall).

Όπως παρουσιάζεται στο Σχήμα 6-X και οι τρεις δείκτες της εφαρμογής κατατάσσονται στην κατηγορία «Excellent». Η Πρακτική Ποιότητα παρουσιάζει μέση τιμή 2,20, η Ηδονική Ποιότητα μέση τιμή 1,99, ενώ η συνολική αξιολόγηση διαμορφώνεται στο 2,09. Η υψηλή επίδοση στην Πρακτική Ποιότητα υποδεικνύει ότι η εφαρμογή παρουσιάζει ιδιαίτερα θετικά χαρακτηριστικά ως προς την ευχρηστία, τη σαφήνεια και την υποστήριξη του χρήστη κατά την αλληλεπίδραση με το σύστημα. Παράλληλα, η κατάταξη της Ηδονικής Ποιότητας στην κατηγορία «Excellent» δείχνει ότι η εμπειρία χρήσης αξιολογήθηκε θετικά και ως προς στοιχεία όπως το ενδιαφέρον και ο ευχάριστος χαρακτήρας της εφαρμογής.



Σχήμα 6.15: Διάγραμμα σύγκρισης αποτελεσμάτων UEQ-S με το UEQ Benchmark

Κεφάλαιο 7ο: Συμπεράσματα και μελλοντικές επεκτάσεις

7.1 Συμπεράσματα

Συνοψίζοντας, στο πλαίσιο της παρούσας εργασίας υλοποιήθηκε μία εφαρμογή πλοήγησης σε εσωτερικούς χώρους για κινητές συσκευές Android. Στόχος της εφαρμογής ήταν η ασφαλής και ανεξάρτητη μετακίνηση και πλοήγηση ατόμων με οπτική αναπηρία, προσφέροντας καθοδήγηση μέσω φωνητικών οδηγιών. Η εφαρμογή αξιοποιεί τις δυνατότητες της επαυξημένης πραγματικότητας, μέσω του ARCore, για τον εντοπισμό και καθοδήγηση του χρήστη.

Μετά από δοκιμές διαπιστώθηκε η δυνατότητα του συστήματος να αναγνωρίζει αποθηκευμένα σημεία αναφοράς, να υπολογίζει την διαδρομή προς τον επιλεγμένο προορισμό και να καθοδηγήσει τον χρήστη προς αυτόν, μέσω φωνητικών οδηγιών και συμπληρωματικής απτικής ανάδρασης. Μέσα από τα αποτελέσματα της αξιολόγησης διαπιστώθηκε πως οι παραπάνω τρόποι ανάδρασης παρείχαν στους συμμετέχοντες επαρκείς πληροφορίες για την πλοήγηση τους. Παρατηρήθηκε, επιπλέον, συνολικά θετική εμπειρία χρήσης, με τις βασικές λειτουργίες και τον τρόπο αλληλεπίδρασης με την εφαρμογή να χαρακτηρίζονται ως κατανοητά και σαφή.

Συμπεραίνοντας, η προτεινόμενη εφαρμογή απέδειξε πως η επαυξημένη πραγματικότητα, και συγκεκριμένα το ARCore μπορεί να χρησιμοποιηθεί για την ανάπτυξη ενός ολοκληρωμένου συστήματος εσωτερικής πλοήγησης για άτομα με οπτική αναπηρία, χωρίς να απαιτείται η εγκατάσταση και συντήρηση ειδικού εξοπλισμού, όπως σε πολλές προτεινόμενες λύσεις στη βιβλιογραφία. Επιπλέον, η διπλή λειτουργία χαρτογράφησης και πλοήγησης στην εφαρμογή, επιτρέπει τη δημιουργία και διαχείριση χαρτών για πληθώρα κτηρίων, χωρίς να προϋποθέτει την απόκτηση έτοιμων ψηφιακών κατόψεων, συμβάλλοντας στην επεκτασιμότητα της.

7.2 Περιορισμοί συστήματος

Παρά την δυνατότητα της εφαρμογής να ανταποκρίνεται επαρκώς στην πλοήγηση προς έναν επιλεγμένο προορισμό, η υλοποίηση παρουσιάζει περιορισμούς, οι οποίοι σχετίζονται κυρίως με το ARCore.

Η ακρίβεια και η σταθερότητα του ARCore επηρεάζονται άμεσα από τις συνθήκες του περιβάλλοντος. Συγκεκριμένα, παράγοντες όπως ο φωτισμός, η έλλειψη διακριτών υφών σε επιφάνειες και η απότομη ή γρήγορη κίνηση της συσκευής δυσχεραίνουν την αναγνώριση οπτικών χαρακτηριστικών και, συνεπώς, μπορεί να οδηγήσουν σε ανεπαρκή παρακολούθηση του χώρου [50]. Επίσης, η χρήση του ARCore περιορίζεται σε συμβατές με αυτό συσκευές [36]. Έτσι, η αποτελεσματικότητα της λειτουργίας του σχετίζεται με τις συνθήκες του περιβάλλοντος χώρου και τις δυνατότητες της συσκευής.

Ένα ακόμη μειονέκτημα λόγω της χρήσης του ARCore αποτελεί η υψηλή ενεργειακή κατανάλωση. Η συνεχής λειτουργία της κάμερας, των αδρανειακών αισθητήρων, της φωνητικής καθοδήγησης και της σύνδεσης δικτύου επηρεάζει έντονα την ενεργειακή κατανάλωση της συσκευής. Συγκεκριμένα, κατά τη δοκιμή της εφαρμογής σε διάφορα σενάρια πλοήγησης, η μπαταρία της κινητής συσκευής δοκιμής παρατηρήθηκε πως μειώθηκε σημαντικά. Αυτό μπορεί να αποδειχθεί ακόμη και επικίνδυνο σε περίπτωση αποφόρτισης της μπαταρίας για ένα άτομο με οπτική αναπηρία, καθώς η κινητή του συσκευή συμβάλλει στην αυτόνομη μετακίνηση του σε άγνωστα περιβάλλοντα, αλλά και γενικότερα κατά την καθημερινότητα του.

Ένας ακόμη περιορισμός είναι η εξάρτηση του συστήματος από σύνδεση στο διαδίκτυο. Όπως αναλύθηκε, απαιτείται σύνδεση για την φιλοξενία των Cloud Anchors, την αποθήκευση του χάρτη στο Cloud Firestore και ανάκτηση του χάρτη και των απαραίτητων δεδομένων κατά την πλοήγηση. Συνεπώς, η απουσία σύνδεσης ή αστάθεια του δικτύου μπορεί να επιφέρει καθυστερήσεις ή ακόμη και αδυναμία χρήσης της εφαρμογής.

Τέλος, ο σημαντικότερος περιορισμός του συστήματος είναι η απουσία μηχανισμού ανίχνευσης δυναμικών εμποδίων στο περιβάλλον. Το εσωτερικό ενός κτηρίου μπορεί να χαρτογραφηθεί με ακρίβεια, αφού οι αίθουσες και γενικότερα τα σημεία ενδιαφέροντος παραμένουν σταθερά. Ωστόσο, σε πραγματικές συνθήκες, σε αντίθεση με το ελεγχόμενο περιβάλλον στο οποίο πραγματοποιήθηκε η δοκιμή της εφαρμογής, είναι πιθανό ένα άτομο με οπτική αναπηρία να έρθει αντιμέτωπο με δυναμικά εμπόδια, για τα οποία η εφαρμογή δεν προβλέπει την αναγνώρισή τους. Επομένως, λειτουργεί ως συμπληρωματικό μέσο πλοήγησης και δεν μπορεί να αντικαταστήσει βοηθήματα ασφαλούς πλοήγησης, όπως το λευκό μπαστούνι.

7.3 Προτάσεις Βελτίωσης - Μελλοντικές επεκτάσεις

Για την αντιμετώπιση ορισμένων από τους παραπάνω περιορισμούς, προτείνονται ενδεικτικές λύσεις. Αρχικά, η τοπική αποθήκευση χαρτών στη συσκευή θα μπορούσε να περιορίσει την εξάρτηση της εφαρμογής από το δίκτυο. Παρόλο που για την επίλυση των Cloud Anchors απαιτείται σταθερή σύνδεση στο διαδίκτυο, η αλλαγή αυτή θα συμβάλλει και στη μείωση της ενεργειακής κατανάλωσης.

Απαραίτητη είναι η δοκιμή της εφαρμογής σε διαφορετικές συσκευές, κτήρια και περιβαλλοντικές συνθήκες. Σε μελλοντική αξιολόγηση θα ήταν χρήσιμο να συμμετέχουν περισσότεροι εθελοντές και κυρίως άτομα με οπτική αναπηρία, οι οποίοι αποτελούν και το βασικό κοινό της συγκεκριμένης εφαρμογής. Επίσης, θα ήταν σημαντικό να καταγραφούν περισσότερες πληροφορίες για την εμπειρία χρήστη και τη λειτουργία της εφαρμογής, όπως ο χρόνος ολοκλήρωσης μιας διαδρομής, ο αριθμός σφαλμάτων κατά την πλοήγηση, η απόκλιση από την προγραμματισμένη διαδρομή και η κατανάλωση της μπαταρίας.

Η σημαντικότερη μελλοντική επέκταση είναι η ενσωμάτωση μηχανισμού αναγνώρισης αντικειμένων και εμποδίων μέσω της κάμερας κατά τη διάρκεια της πλοήγησης, μέσω μηχανικής μάθησης. Η εφαρμογή θα μπορούσε να αναγνωρίζει εμπόδια μέσα στο δυναμικά μεταβαλλόμενο περιβάλλον και να προειδοποιεί τον χρήστη για αυτά, μέσω φωνητικής και απτικής ανάδρασης, με στόχο την ασφαλή μετακίνηση των χρηστών και την πλήρη ανεξαρτησία τους κατά την πλοήγηση. Επιπλέον, θα μπορούσε να αναπτυχθεί μία λειτουργία συνεργασίας, κατά την οποία ένας χρήστης με οπτική αναπηρία θα είχε τη δυνατότητα να υποβάλλει αίτημα χαρτογράφησης κάποιου κτηρίου και ένας διαθέσιμος βλέπων εθελοντής, ο οποίος έχει κατεβάσει ήδη την εφαρμογή, να αναλάβει τη δημιουργία του χάρτη.

ΒΙΒΛΙΟΓΡΑΦΙΑ

- [1] E. D Kaplan and C Hegarty, Understanding GPS/GNSS : Principles and Applications. Boston ; London: Artech House, 2017.
- [2] M. D. Messaoudi, B.-A. J. Menelas, and H. Mcheick, “Review of Navigation Assistive Tools and Technologies for the Visually Impaired,” *Sensors*, vol. 22, no. 20, p. 7888, Oct. 2022, doi: 10.3390/s22207888.
- [3] Ekaterina Sukhareva, T. Tomchinskaya, and Ilya Serov, “SLAM-based Indoor Navigation in University Buildings,” pp. 611–617, Jan. 2021, doi: 10.20948/graphicon-2021-3027-611-617.
- [4] L. Arora, A. Choudhary, M. Bhatt, J. Kaliappan, and K. Srinivasan, “A Comprehensive Review on NUI, Multi-Sensory Interfaces and UX Design for Applications and Devices for Visually Impaired Users,” *Frontiers in Public Health*, vol. 12, Oct. 2024, doi: 10.3389/fpubh.2024.1357160.
- [5] World Health Organization, “Blindness and Vision Impairment,” world health organization. Accessed: Sep. 03, 2026. [Online]. Available: <https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment>
- [6] W. Jeamwattthanachai, M. Wald, and G. Wills, “Indoor Navigation by Blind People: Behaviors and Challenges in Unfamiliar Spaces and Buildings - ePrints Soton,” Soton.ac.uk, May 2019, doi: https://eprints.soton.ac.uk/427172/1/indoor_navigation_problems.pdf.
- [7] D. Belli, Paolo Barsocchi, G. Lombardi, A. Coffrini, and A. Crivello, “Navigation Solutions for Blind and Visually Impaired Persons: A State-of-the-Art Survey,” *International Journal of Human-Computer Interaction*, pp. 1–30, Mar. 2026, doi: 10.1080/10447318.2026.2643358.
- [8] F. Alessio, G. Fabio, C. Limongelli, M. Stefano, G. Sansonetti, and J. Torres-Sospedra, “Indoor Positioning, Localization, and Navigation Systems for Cultural Heritage: A Five-Year Systematic Review,” *Internet of Things*, vol. 39, p. 102011, Sep. 2026, doi: 10.1016/j.iot.2026.102011.
- [9] P. S. Farahsari, A. Farahzadi, J. Rezazadeh, and A. Bagheri, “A Survey on Indoor Positioning Systems for IoT-based Applications,” *IEEE Internet of Things Journal*, pp. 1–1, 2022, doi: 10.1109/jiot.2022.3149048.
- [10] N. Syazwani C. J, A. Wahab, Noorhazirah Sunar, H. Syed, Keng Yinn Wong, and Y. Aun, “Indoor Positioning System: A Review,” *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 6, Jan. 2022, doi: 10.14569/ijacsa.2022.0130659.
- [11] Si. Tomažič and I. Škrjanc, “An Automated Indoor Localization System for Online Bluetooth Signal Strength Modeling Using Visual-Inertial SLAM,” *Mdpi.com*. Accessed: Sep. 03, 2026. [Online]. Available: <https://www.mdpi.com/1424-8220/21/8/2857>
- [12] F. Zafari, A. Gkelias, and K. K. Leung, “A Survey of Indoor Localization Systems and Technologies,” *IEEE Communications Surveys Tutorials*, vol. 21, no. 3, pp. 2568–2599, 2019, doi: 10.1109/COMST.2019.2911558.
- [13] C. S. Mouhammad, A. Allam, M. Abdel-Raouf, E. Shenouda, and M. Elsabrouty, “BLE Indoor Localization Based on Improved RSSI and Trilateration,” *IEEE Xplore*. Accessed: Sep. 03, 2026. [Online]. Available: <https://ieeexplore.ieee.org/abstract/document/9051304>

- [14] Inpixon, “Wi-Fi Location Tracking & Positioning Systems | Inpixon,” www.inpixon.com. Accessed: Sep. 03, 2026. [Online]. Available: <https://www.inpixon.com/technology/standards/wifi>
- [15] N. S. Aminah, A. S. Ichwanda, D. D. Djamal, Y. B. W. Budiharto, and M. Budiman, “A Low-Cost Indoor Navigation and Tracking System Based on Wi-Fi-RSSI,” *Wireless Personal Communications*, vol. 136, no. 3, pp. 1791–1809, Jun. 2024, doi: 10.1007/s11277-024-11361-3.
- [16] W. C. S. S. Simões, G. S. Machado, A. M. A. Sales, M. M. de Lucena, N. Jazdi, and V. F. de Lucena, “A Review of Technologies and Techniques for Indoor Navigation Systems for the Visually Impaired,” *Sensors*, vol. 20, no. 14, p. 3935, Jul. 2020, doi: 10.3390/s20143935.
- [17] A. Abraham and V. Namboodiri, “An Accessible BLE Beacon-Based Indoor Wayfinding System,” scholarworks.calstate.edu. Accessed: Sep. 03, 2026. [Online]. Available: <https://scholarworks.calstate.edu/concern/publications/j6731b477>
- [18] Э. И. Кальметьев and А. Е. Сергеева, “Обзор Современных Методов И Технологий Навигации Внутри Помещений OVERVIEW of MODERN METHODS and TECHNOLOGIES of INDOOR NAVIGATION,” Mar. 2025. Accessed: Sep. 03, 2026. [Online]. Available: <https://s.eduherald.ru/pdf/2025/3/21869.pdf>
- [19] A. Idrees, Z. Iqbal, and M. Ishfaq, “An Efficient Indoor Navigation Technique to Find Optimal Route for Blinds Using QR Codes,” *IEEE Xplore*. Accessed: Sep. 03, 2026. [Online]. Available: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7334197>
- [20] Y. Alkady, R. Rizk, D. M. Alsekait, A. S. Alluhaidan, and D. S. Abdelminaam, “SINS_AR: An Efficient Smart Indoor Navigation System Based on Augmented Reality,” *IEEE Access*, vol. 12, pp. 109171–109183, 2024, doi: 10.1109/access.2024.3439357.
- [21] A. G. Karkar, S. Al-Maadeed, J. Kunhoth, and A. Bouridane, “CamNav: A Computer-Vision Indoor Navigation System,” *The Journal of Supercomputing*, vol. 77, no. 7, pp. 7737–7756, Jan. 2021, doi: 10.1007/s11227-020-03568-5.
- [22] J. Kunhoth, A. Karkar, S. Al-Maadeed, and A. Al-Attiyah, “Comparative Analysis of Computer-Vision and BLE Technology Based Indoor Navigation Systems for People With Visual Impairments,” *International Journal of Health Geographics*, vol. 18, no. 1, Dec. 2019, doi: 10.1186/s12942-019-0193-9.
- [23] M. Mehta, K. Mehta, K. Chauhan, M. Mangla, M. Sankhe, and P. Hole, “IndoorNavX: ARCore-Powered Markerless Indoor Navigation,” in *2025 IEEE International Conference on Smart Power, Energy, Renewables, and Transportation (SPERT)*, 2025, pp. 1–7. doi: 10.1109/SPERT67079.2025.11469721.
- [24] C. Lin and Y. Shin, “Deep Learning-Based Multifloor Indoor Tracking Scheme Using Smartphone Sensors,” *IEEE Access*, vol. 10, pp. 63049–63062, 2022, doi: 10.1109/ACCESS.2022.3183072.
- [25] C. H. Tsai, F. Elyasi, P. Ren, and R. Manduchi, “All the Way There and Back: Inertial-Based, Phone-in-Pocket Indoor Wayfinding and Backtracking Apps for Blind Travelers,” *arXiv.org*. Accessed: Sep. 03, 2026. [Online]. Available: <https://arxiv.org/abs/2401.08021>
- [26] D. Plikynas, A. Žvironas, M. Gudauskis, A. Budrionis, P. Daniušis, and I. Sliesoraitytė, “Research Advances of Indoor Navigation for Blind People: A Brief Review of Technological Instrumentation,” *IEEE Instrumentation & Measurement Magazine*, vol. 23, no. 4, pp. 22–32, 2020, doi: 10.1109/MIM.2020.9126068.

- [27] D. Plikynas, A. Žvironas, A. Budrionis, and M. Gudauskis, “Indoor Navigation Systems for Visually Impaired Persons: Mapping the Features of Existing Technologies to User Needs,” *Sensors*, vol. 20, no. 3, p. 636, Jan. 2020, doi: 10.3390/s20030636.
- [28] R. T. Azuma, “A Survey of Augmented Reality,” *Presence: Teleoperators and Virtual Environments*, vol. 6, no. 4, pp. 355–385, 1997, doi: 10.1162/pres.1997.6.4.355.
- [29] D. Schmalstieg and T. Hollerer, *Augmented Reality*. Addison-Wesley Professional, 2016.
- [30] Z. Oufqir, A. El Abderrahmani, and K. Satori, “ARKit and ARCore in Serve to Augmented Reality,” in *2020 International Conference on Intelligent Systems and Computer Vision (ISCV)*, 2020, pp. 1–7. doi: 10.1109/ISCV49265.2020.9204243.
- [31] V. V. Satyadev, *Building AR Apps With ARCore and ARKit*. Orange Education Pvt. Ltd., 2025.
- [32] J. Zhu et al., “Semantics-Based Connectivity Graph for Indoor Pathfinding Powered by IFC-Graph,” *Automation in Construction*, vol. 171, p. 106019, Mar. 2025, doi: 10.1016/j.autcon.2025.106019.
- [33] L. Georgiadis, S. Nikolopoulos, and L. Palios, *Data Structures*. 2016. Accessed: Sep. 03, 2026. [Online]. Available: <https://dx.doi.org/10.57713/kallipos-710>
- [34] K. Georgouli, *Artificial Intelligence*. 2015. Accessed: Sep. 03, 2026. [Online]. Available: <https://dx.doi.org/10.57713/kallipos-666>
- [35] “Android Developers,” Android Developers. Accessed: Sep. 03, 2026. [Online]. Available: <https://developer.android.com>
- [36] “Choose Your Development Environment | ARCore,” Google Developers. Accessed: Sep. 03, 2026. [Online]. Available: <https://developers.google.com/ar/develop>
- [37] N. Hardiyanti, A. Lawi, Diaraya, and F. Aziz, “Classification of Human Activity Based on Sensor Accelerometer and Gyroscope Using Ensemble SVM Method,” *2018 2nd East Indonesia Conference on Computer and Information Technology (EIconCIT)*, pp. 304–307, Nov. 2018, doi: 10.1109/eiconcit.2018.8878627.
- [38] SceneView, “SceneView,” SceneView. Accessed: Sep. 03, 2026. [Online]. Available: <https://sceneview.github.io/>
- [39] Firebase, “Cloud Firestore | Firebase,” Firebase. Accessed: Sep. 03, 2026. [Online]. Available: <https://firebase.google.com/docs/firestore>
- [40] “TextToSpeech | Android Developers,” Android Developers. Accessed: Sep. 03, 2026. [Online]. Available: <https://developer.android.com/reference/android/speech/tts/TextToSpeech>
- [41] “SpeechRecognizer | Android Developers,” Android Developers. Accessed: Sep. 03, 2026. [Online]. Available: <https://developer.android.com/reference/android/speech/SpeechRecognizer>
- [42] “Vibrator | API Reference | Android Developers,” Android Developers. Accessed: Sep. 03, 2026. [Online]. Available: <https://developer.android.com/reference/android/os/Vibrator>
- [43] C. Murad, H. Candello, and C. Munteanu, “What’s the Talk on VUI Guidelines? A Meta-Analysis of Guidelines for Voice User Interface Design,” *Proceedings of the 5th International Conference on Conversational User Interfaces*, pp. 1–16, Jul. 2023, doi: 10.1145/3571884.3597129.

- [44] “Make Apps More Accessible,” Android Developers. Accessed: Sep. 03, 2026. [Online]. Available: <https://developer.android.com/guide/topics/ui/accessibility/apps>
- [45] D. Plikynas, A. Indriulionis, A. Laukaitis, and L. Sakalauskas, “Indoor-Guided Navigation for People Who Are Blind: Crowdsourcing for Route Mapping and Assistance,” *Applied Sciences*, vol. 12, no. 1, p. 523, Jan. 2022, doi: 10.3390/app12010523.
- [46] S. Khusro, B. Shah, I. Khan, and S. Rahman, “Haptic Feedback to Assist Blind People in Indoor Environment Using Vibration Patterns,” *Sensors*, vol. 22, no. 1, p. 361, Jan. 2022, doi: 10.3390/s22010361.
- [47] “Com.Google.Ar.Core,” Google for Developers. Accessed: Sep. 03, 2026. [Online]. Available: <https://developers.google.com/ar/reference/java/com/google/ar/core/package-summary>
- [48] UEQ, “User Experience Questionnaire (UEQ),” Ueq-online.org. Accessed: Sep. 03, 2026. [Online]. Available: <https://www.ueq-online.org/>
- [49] G. Melissourgos and C. Katsanos, “UEQ-GR and UEQ-S-GR: Towards a Greek Adaptation of the User Experience Questionnaire and Its Short Version,” *Proceedings of the 3rd International Conference of the ACM Greek SIGCHI Chapter*, pp. 203–211, Sep. 2025, doi: 10.1145/3749012.3749059.
- [50] T. Feigl, A. Porada, S. Steiner, C. Löffler, C. Mutschler, and M. Philippsen, “Localization Limitations of ARCore, ARKit, and Hololens in Dynamic Large-Scale Industry Environments,” *Proceedings of the 15th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, 2020, doi: 10.5220/0008989903070318.