

UNIVERSITY OF PATRAS - SCHOOL OF ENGINEERING
DEPARTMENT OF ELECTRICAL
AND COMPUTER ENGINEERING



ΠΑΝΕΠΙΣΤΗΜΙΟ
ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

DIVISION: ELECTRONICS AND COMPUTERS

DIPLOMA THESIS

of the Electrical and Computer Engineering department student
of the Engineering School of the University of Patras

ANGELOS KARDOUTSOS

REGISTRATION NUMBER: 1059372

Title

Development of a mixed reality system to support individuals with visual impairment

Supervisor

Nikolaos Avouris, Professor, University of Patras

Patras, July 2024

University of Patras, Department of Electrical and Computer Engineering.
Angelos Kardoutsos

© 2024 – All rights reserved

The whole work is an original work, produced by Angelos Kardoutsos, and does not violate the rights of third parties in any way. If the work contains material which has not been produced by him/her, this is clearly visible and is explicitly mentioned in the text of the work as a product of a third party, noting in a similarly clear way his/her identification data, while at the same time confirming that in case of using original graphics representations, images, graphs, etc., has obtained the unrestricted permission of the copyright holder for the inclusion and subsequent publication of this material.

CERTIFICATION

It is certified that the Diploma Thesis titled

Development of a mixed reality system to support individuals with visual impairment

of the Department of Electrical and Computer Engineering student

Angelos Kardoutsos

(Registration Number: 1059372)

was presented publicly at the Department of Electrical and Computer
Engineering at

10/07/2024

and was examined by the following examining committee:

Nikolaos Avouris, Professor, University of Patras

Christos Feidas, Associate Professor, University of Patras

Christos Sintoris, Laboratory Teaching Staff, University of Patras

The Supervisor

Nikolaos Avouris
Professor

The Director of the Division

George Theodoridis
Professor

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ - ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΠΟΛΟΓΙΣΤΩΝ



ΠΑΝΕΠΙΣΤΗΜΙΟ
ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

ΤΟΜΕΑΣ: ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του φοιτητή του Τμήματος Ηλεκτρολόγων Μηχανικών και Τεχνολογίας
Υπολογιστών της Πολυτεχνικής Σχολής του Πανεπιστημίου Πατρών

ΑΓΓΕΛΟΥ ΚΑΡΔΟΥΤΣΟΥ ΤΟΥ ΑΠΟΣΤΟΛΟΥ

ΑΡΙΘΜΟΣ ΜΗΤΡΟΥ: 1059372

Θέμα

Ανάπτυξη συστήματος μεικτής πραγματικότητας για υποστήριξη ατόμων
με προβλήματα όρασης

Επιβλέπων

Νικόλαος Αβούρης, Καθηγητής, Πανεπιστήμιο Πατρών

Πάτρα, Ιούλιος 2024

Πανεπιστήμιο Πατρών, Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας
Υπολογιστών.

Άγγελος Καρδούτσος του Απόστολου

© 2024 – Με την επιφύλαξη παντός δικαιώματος.

Το σύνολο της εργασίας αποτελεί πρωτότυπο έργο, παραχθέν από τον Άγγελο Καρδούτσο, και δεν παραβιάζει δικαιώματα τρίτων καθ' οιονδήποτε τρόπο. Αν η εργασία περιέχει υλικό, το οποίο δεν έχει παραχθεί από τον ίδιο, αυτό είναι ευδιάκριτο και αναφέρεται ρητώς εντός του κειμένου της εργασίας ως προϊόν εργασίας τρίτου, σημειώνοντας με παρομοίως σαφή τρόπο τα στοιχεία ταυτοποίησής του, ενώ παράλληλα βεβαιώνει πως στην περίπτωση χρήσης αυτούσιων γραφικών αναπαραστάσεων, εικόνων, γραφημάτων κ.λπ., έχει λάβει τη χωρίς περιορισμούς άδεια του κατόχου των πνευματικών δικαιωμάτων για την συμπερίληψη και επακόλουθη δημοσίευση του υλικού αυτού.

ΠΙΣΤΟΠΟΙΗΣΗ

Πιστοποιείται ότι η διπλωματική εργασία με θέμα

Ανάπτυξη συστήματος μεικτής πραγματικότητας για υποστήριξη ατόμων
με προβλήματα όρασης

του φοιτητή του Τμήματος Ηλεκτρολόγων Μηχανικών και Τεχνολογίας
Υπολογιστών

Άγγελου Καρδούτσου του Απόστολου

(Αριθμός Μητρώου: 1059372)

παρουσιάστηκε δημόσια στο τμήμα Ηλεκτρολόγων Μηχανικών και
Τεχνολογίας Υπολογιστών στις

10/07/2024

και εξετάστηκε από την ακόλουθη εξεταστική επιτροπή:

Νικόλαος Αβούρης, Καθηγητής, Πανεπιστήμιο Πατρών
Χρήστος Φείδας, Αναπληρωτής Καθηγητής, Πανεπιστήμιο Πατρών
Χρήστος Σιντόρης, Ε.ΔΙ.Π., Πανεπιστήμιο Πατρών

Ο Επιβλέπων

Νικόλαος Αβούρης
Καθηγητής

Ο Διευθυντής του Τομέα

Γεώργιος Θεοδωρίδης
Καθηγητής

Περίληψη

Η παρούσα διπλωματική εργασία αποσκοπεί στην δημιουργία μίας εφαρμογής για τη συσκευή μικτής πραγματικότητας Microsoft HoloLens 2. Στόχος της είναι να υποστηρίξει άτομα με προβλήματα όρασης, συμβάλλοντας στην ανεξαρτησία τους και τη προσβασιμότητά τους. Ειδικότερα, προσφέρει τη δυνατότητα να περιηγηθούν με ασφάλεια σε έναν άγνωστο, για αυτούς, χώρο.

Αρχικά, περιγράφεται το θεωρητικό και τεχνολογικό υπόβαθρο. Πιο συγκεκριμένα, παραθέτουμε τα αίτια που οδηγούν σε απώλεια όρασης, τον αντίκτυπο που έχει αυτή στο άτομο και τις διαθέσιμες λύσεις για την αντιμετώπισή της. Επιπλέον, επεξηγούνται οι έννοιες της εικονικής, επαυξημένης και μικτής πραγματικότητας και αναλύονται τα τεχνικά χαρακτηριστικά της συσκευής Microsoft HoloLens 2 και οι δυνατότητές της, όπως η χωρική χαρτογράφηση και ο χωρικός ήχος, καθώς και τα εργαλεία που αξιοποιήθηκαν για την ανάπτυξη της εφαρμογής, όπως η Unity.

Στη συνέχεια, παρουσιάζεται η υλοποίηση της εφαρμογής, ξεκινώντας με το σχεδιασμό αυτής και τους περιορισμούς που τέθηκαν και, έπειτα, με τον τρόπο οργάνωσης και προετοιμασίας του project. Ακολουθεί μια αναλυτική περιγραφή όλων των λειτουργιών της εφαρμογής, από τις οποίες κύριες είναι ο εντοπισμός εμποδίων και η προειδοποίηση χρηστών, και ο τρόπος με τον οποίο αναπτύχθηκαν.

Η αξιολόγηση της εφαρμογής πραγματοποιήθηκε με τη βοήθεια δέκα εθελοντών, οι οποίοι ακλήθηκαν να τη δοκιμάσουν παράλληλα με ένα αρκετά διαδεδομένο εργαλείο για άτομα με προβλήματα όρασης, το μπαστούνι. Από τις δοκιμές χρήσης και των δύο εργαλείων από τους εθελοντές, καταφέρουμε να συλλέξουμε σημαντικά συγκριτικά δεδομένα, τα οποία και αναλύονται μαζί με αυτά που αντλούμε από τις απαντήσεις των ερωτηματολογίων και τις παρατηρήσεις των χρηστών. Με βάση τα δεδομένα αυτά, συμπεραίνουμε ότι, παρά τη χρηστικότητα της εφαρμογής, αυτή δεν καταφέρνει να βοηθήσει τους εθελοντές να ολοκληρώσουν το task τους ταχύτερα και με μεγαλύτερη ασφάλεια σε σχέση με το μπαστούνι. Για το λόγο αυτό, θα αναλυθούν και τα ελαττώματα της εφαρμογής, όπως αυτά παρατηρήθηκαν από εμάς και αναφέρθηκαν από τους εθελοντές και πως αυτά οδήγησαν στο προαναφερθέν αποτελέσμα.

Τέλος, η εργασία ολοκληρώνεται παραθέτοντας πιθανές μελλοντικές προεκτάσεις και βελτιώσεις της εφαρμογής με στόχο μια ακόμη καλύτερη εμπειρία για τον τελικό χρήστη.

Λέξεις-κλειδιά: Εκτεταμένη Πραγματικότητα, Επαυξημένη Πραγματικότητα, Μικτή Πραγματικότητα, Microsoft HoloLens 2, Microsoft Visual Studio, Unity, χωρική χαρτογράφηση, χωρικός ήχος, απώλεια όρασης, προσβασιμότητα

Extensive English Summary

This thesis aims to create an application for the mixed reality device Microsoft HoloLens 2. Its goal is to support individuals with visual impairments, contributing to their independence and accessibility. Specifically, it offers the ability to safely navigate an unfamiliar environment.

Initially, the theoretical and technological background is described. More specifically, we present the causes that lead to vision loss, its impact on the individual, and the available solutions to address it. Additionally, the concepts of virtual, augmented, and mixed reality are explained, and the technical characteristics of the Microsoft HoloLens 2 device and its capabilities, such as spatial mapping and spatial sound, are analyzed, along with the tools used for the application development, such as Unity.

Next, the implementation of the application is presented, starting with its design and the constraints that were set, followed by the way the project was organized and prepared. A detailed description of all the functions of the application follows, with the main ones being obstacle detection and user warning, and the way they were developed.

The evaluation of the application was carried out with the help of ten volunteers, who were asked to test it alongside a widely used tool for visually impaired individuals, the cane. From the tests using both tools by the volunteers, we manage to collect significant comparative data, which are analyzed along with the responses from the questionnaires and the users' observations. Based on this data, we conclude that despite the application's usability, it fails to help volunteers complete their tasks faster and more safely compared to using a cane. The defects of the application, as observed by us and reported by the volunteers, are analyzed, as well as how these led to the aforementioned results.

Finally, the thesis was completed by outlining possible future extensions and improvements of the application to provide an even better experience for the end-user.

Keywords: Augmented Reality, Virtual Reality, Mixed Reality, Microsoft HoloLens 2, Microsoft Visual Studio, Unity, spatial mapping, spatial audio, visual impairment, accessibility

Ευχαριστίες

Θα ήθελα να ευχαριστήσω την οικογένεια και τους φίλους μου για την αδιάκοπη υποστήριξή τους, όχι μόνο κατά τη διάρκεια εκπόνησης αυτής της διπλωματικής εργασίας, αλλά όλα τα χρόνια των σπουδών μου.

Επιπλέον, θα ήθελα να ευχαριστήσω τον κ. Νικόλαο Αβούρη, που μου έδωσε την ευκαιρία να εργαστώ και να ασχοληθώ πάνω στο συγκεκριμένο θέμα, καθώς και τον Γιώργο Παπαδούλη για τη συνέχη βοήθειά του κατά την υλοποίηση της εργασίας.

Τέλος, πρέπει να ευχαριστήσω όλους τους εθελοντές που συμμετείχαν στην πειραματική διαδικασία και στην αξιολόγηση, αφού, χωρίς αυτούς, δεν θα ήταν δυνατή η ολοκλήρωση της διπλωματικής εργασίας.

ΠΕΡΙΕΧΟΜΕΝΑ

Κατάλογος σχημάτων	xiii
Κατάλογος πινάκων	xv
1 Εισαγωγή	1
1.1 Δομή Διπλωματικής Εργασίας	2
2 Θεωρητικό και Τεχνολογικό Υπόβαθρο	5
2.1 Όραση	6
2.1.1 Τρόπος Λειτουργίας	6
2.1.2 Απώλεια Όρασης: Κατηγοροποίηση και Αιτίες	6
2.1.3 Απώλεια Όρασης: Αυτίκτυπος	7
2.1.4 Απώλεια Όρασης: Λύσεις	7
2.2 Εκτεταμένη Πραγματικότητα	9
2.2.1 Εικονική Πραγματικότητα	9
2.2.2 Επαυξημένη Πραγματικότητα	10
2.2.3 Μικτή Πραγματικότητα	11
2.3 Microsoft HoloLens 2	11
2.3.1 Τεχνικά Χαρακτηριστικά	12
2.3.2 Τρόποι Αλληλεπίδρασης	14
2.3.3 Χωρική Χαρτογράφηση (Spatial Mapping)	17
2.3.4 Χωρικός Ήχος (Spatial Audio)	18
2.4 Εργαλεία ανάπτυξης για HoloLens	19
2.4.1 Unity	19
2.4.2 Microsoft Visual Studio	23
2.4.3 Mixed Reality Toolkit	25
3 Η υλοποίηση	27
3.1 Σενάριο Εφαρμογής	28
3.2 Σχεδιασμός και Περιορισμοί	29
3.3 Γλοποίηση	30

3.3.1	Προετοιμασία και Εγκατάσταση	30
3.3.2	Οργάνωση του Project	32
3.3.3	Εντοπισμός Εμποδίων	34
3.3.4	Προειδοποίηση Χρήστη	42
3.3.5	Φωνητικές Εντολές	45
4	Αξιολόγηση	47
4.1	Διαδικασία αξιολόγησης	48
4.1.1	Περιγραφή Πειράματος	48
4.1.2	Προδιαγραφές Αξιολόγησης	49
4.2	Αποτελέσματα	51
4.2.1	Καταγραφή Χρόνων και Ανάλυση	51
4.2.2	Καταγραφή Απαντήσεων Ερωτηματολογίου και Παρατηρήσεις .	56
5	Προεκτάσεις και Επίλογος	61
5.1	Μελλοντικές προεκτάσεις	62
5.2	Επίλογος	62
	Βιβλιογραφία	65
	Παράρτημα Α'	73
	Παράρτημα Β'	97

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

2.1	Η ανατομία του ματιού	6
2.2	Απτική πλακόστρωση	8
2.3	Κλίμακα Πραγματικού Κόσμου - Εικονικού Κόσμου	9
2.4	Χρήστης φορά τα γυαλιά εικονικής πραγματικότητας Meta Quest 3	10
2.5	Η συσκευή μικτής πραγματικότητας Microsoft HoloLens 2	12
2.6	Εξαρτήματα και αισθητήρες της συσκευής Microsoft HoloLens 2	12
2.7	Αισθητήρες στη προσωπίδα (visor) της συσκεύης Microsoft HoloLens 2	13
2.8	Εικονική αναπαράσταση χεριού	14
2.9	Τρόποι αλληλεπίδρασης με τους δείκτες των χεριών με 2D αντικείμενα	15
2.10	Πίεση (Press) εικονικού κουμπιού	15
2.11	Τρόποι αλληλεπίδρασης με το δείκτη και τον αντίχειρα με 3D αντικείμενα	16
2.12	Αλληλεπίδραση με εικονικό αντικείμενο σε μακρινή απόσταση	16
2.13	Υπόδειξη διαθέσιμης φωνητικής εντολής μέσω ετικέτας	17
2.14	Τριασδιάστατη ανάπαρασταση ένος χώρου	18
2.15	Περιοχές (Quads) που σχηματίστηκαν από το Scene Understanding	18
2.16	To περιβάλλον ανάπτυξης (Editor) της Unity	21
2.17	To Hierarchy window του Editor	21
2.18	To Game view του Editor	22
2.19	To Scene view του Editor	23
2.20	To Inspector window του Editor	24
2.21	To Project window του Editor	24
2.22	To Status bar του Editor	24
3.1	To Mixed Reality Feature Tool	30
3.2	To hierarchy window μετά την προσθήκη του MRTK	31
3.3	To Holographic Remoting Player σε αναμονή σύνδεσης	32
3.4	Η τελική μορφή του Hierarchy Window	33
3.5	To Inspector window για το variableAggObject	35
3.6	Οι ρυθμίσεις της λειτουργίας ‘Spatial Awareness’ και των ‘Spatial Surface Observers’	36

3.7 Απεικόνιση του εργαστηρίου με 3D μοντέλο	37
3.8 Χωρική χαρτογράφηση του χώρου του εργαστηρίου και απεικόνιση του mesh	38
3.9 Παράδειγμα εντοπισμού εμποδίων με τη μέθοδο BoxCast	41
3.10 Εντοπισμός εμποδίων με τη λειτουργία Hands Mode	42
3.11 Το component Audio Source	43
3.12 Το χειριστήριο Microsoft XBOX Series	44
3.13 Το component SpeechInputHandler και οι φωνητικές εντολές της εφαρμογής	45
 4.1 Χώρος διεξαγωγής του πειράματος	48
4.2 Εθελοντές χρησιμοποιούν τους δύο διαφορετικούς τρόπους περιήγησης στο χώρο	49
4.3 Απαντήσεις για το επίπεδο εξοικείωσης των εθελοντών με την επαυξημένη πραγματικότητα	50
4.4 Θηκόγραμμα της διαφοράς των συνολικών χρόνων	54
4.5 Έλεγχος δειγμάτων για κανονική κατανομή στα δείγματα χρόνων ολοκλήρωσης	54
4.6 Αποτελέσματα του paired-samples t test για τους χρόνους ολοκλήρωσης διαδρομής	55
4.7 Θηκόγραμμα της διαφοράς του πλήθους σφαλμάτων	55
4.8 Έλεγχος δειγμάτων για κανονική κατανομή στα δείγματα σφαλμάτων .	56
4.9 Αποτελέσματα του paired-samples t test για τους χρόνους ολοκλήρωσης διαδρομής	56
4.10 Συνολικό σκορ προτάσεων του ερωτηματολογίου SUS	58

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

2.1	Τεχνικές προδιαγραφές του headset Microsoft HoloLens 2	14
2.2	Διαθέσιμα engines και APIs για την ανάπτυξη εφαρμογών για το Microsoft HoloLens 2	19
3.1	Packages που ενσωματώθηκαν στο project με τη χρήση του Mixed Reality Feature Tool	31
4.1	Αποτελέσματα χρονομετρήσεων των δύο σκελών του πειράματος	51
4.2	Μέσος χρόνο ολοκλήρωσης τμημάτων του πειράματος και μέσο πλήθος σφαλμάτων ανά σκέλος	52
4.3	Μέση ποσοστιαία διαφορά χρόνων και ποσοστιαία διαφορά μέσων χρόνων ολοκλήρωσης τμημάτων του πειράματος	52
4.4	Απαντήσεις στο ερωτηματολόγιο SUS	57

ΚΕΦΑΛΑΙΟ

1

ΕΙΣΑΓΩΓΗ

Ο άνθρωπος, από τη γέννησή του, διαθέτει πέντε βασικές αισθήσεις: την αφή, την όραση, την ακοή, την όσφροση και την γεύση [1]. Οι αισθήσεις αυτές του δίνουν τη δυνατότητα να αντιληφθεί καλύτερα το περιβάλλον γύρω του και συμβάλλουν σημαντικά στην επιβίωσή του. Ωστόσο, εξαιτίας ποικίλων παραγόντων, είναι πιθανό να εμφανιστεί κάποια δυσλειτουργία στο αισθητήριο όργανο, οδηγώντας σε μερική ή πλήρη απώλεια της συγκεκριμένης αίσθησης. Στα πλαίσια της συγκεκριμένης διπλωματικής εργασίας, θα εστιάσουμε στο πρόβλημα της μερικής ή πλήρης απώλειας όρασης. Η απώλεια της συγκεκριμένης αίσθησης μπορεί να δυσχεράνει την υλοποίηση καθημερινών εργασιών από το άτομο, την πρόσβαση και πλοήγηση του σε χώρους, να επηρέασει την φυχολογία και την αυτοπεποίθησή του, οδηγώντας, τελικά, σε υποβάθμιση της ποιότητας ζωής του και έχοντας αρνητικό αντίκτυπο στην κοινωνική του ζωή και στην πνευματική του υγεία [2, 3]. Επόμενως, χρίνεται απαραίτητη η ανάπτυξη και η κατασκευή συσκευών, οι οποίες έχουν σκοπό να αναπληρώσουν την απούσα αίσθηση, εκμεταλλευόμενες τις ήδη υπάρχουσες αισθήσεις, με τελικό στόχο την διευκόλυνση της καθημερινότητας του ατόμου.

Παράλληλα, τα τελευταία χρόνια, η ανάπτυξη της τεχνολογίας αποδεικνύεται να είναι ραγδαία. Ειδικότερα, η τεχνολογία της επαυξημένης (Augmented Reality, AR), εικονικής (Virtual Reality, VR) και μικτής (Mixed Reality, MR) πραγματικότητας «εισβάλει» όλο και περισσότερο στην καθημερινότητά μας, βρίσκοντας εφαρμογή σε διάφορους τομείς αυτής, όπως είναι η διασκέδαση, η εκπαίδευση, η υγεία [4] κ.α. [5], διαθέτοντας συσκευές και εφαρμογές προσβάσιμες στο μέσο χρήστη, λόγω της απλότητας χρήσης τους και του κόστους τους. Παραδείγματα αυτών αποτελούν οι συσκευές Meta Quest, Valve Index (γυαλιά Εικονικής Πραγματικότητας), Microsoft HoloLens και Google Glass (γυαλιά Επαυξημένης και Μικτής Πραγματικότητας).

Λαμβάνοντας υπόψην, λοιπόν, την τρέχουσα τεχνολογική πρόοδο και το πλήθος συσκευών που βρίσκονται στη διάθεση του μέσου χρήστη, καθώς και τα προβλήματα προσβασιμότητας που αντιμετωπίζουν άτομα με μερική ή πλήρη απώλεια όρασης, τότε εύλογα τίθεται το ερώτημα:

Είναι εφικτή η ανάπτυξη μιας εφαρμογής, η οποία αξιοποιεί τις διαθέσιμες τεχνολογίες και hardware επαυξημένης/μικτής πραγματικότητας και παράλληλα βοηθά άτομα με αναπηρία να πραγματοποιήσουν καθημερινές εργασίες με ευκολία;

Σκοπός της διπλωματικής εργασίας είναι η υλοποίηση μιας τέτοιας εφαρμογής, η οποία θα εξυπηρετεί ειδικότερα άτομα με μερική ή πλήρη απώλεια όρασης. Η εφαρμογή στοχεύει στο να προσφέρει βοήθεια κατά την πρόσβαση και περιήγηση ενός τέτοιου ατόμου σε χώρους, ειδοποιώντας τον για πιθανά εμπόδια που μπορεί να συναντήσει στη διαδρομή του. Για τον εντοπισμό των εμποδίων και την ενημέρωση του χρήστη για αυτά, θα αναπτυχθεί λογισμικό, το οποίο θα αξιοποιεί τους αισθητήρες και τις ενσωματωμένες τεχνολογίες της συσκευής HoloLens 2 της Microsoft.

1.1 Διπλωματικής Εργασίας

Η παρούσα διπλωματική εργασία χωρίζεται σε 5 κεφάλαια. Στο 2ο κεφάλαιο παρουσιάζεται σε βάθος το θεωρητικό και τεχνολογικό υπόβαθρο, το οποίο σχετίζεται με την ανάπτυξη της εφαρμογής. Έπειτα, στο 3ο κεφάλαιο, περιγράφεται αναλυτικά η διαδικάσια σχεδίασμού και υλοποίησης της εφαρμογής, ενώ, στο 4ο κεφάλαιο, η εφαρμογή διατίθεται σε χρήστες, με σκοπό να την αξιολογήσουν. Παρατίθονται οι συνθήκες κάτω από τις οποίες έγινε η χρήση της εφαρμογής, ο τρόπος αξιολόγησης αυτής, καθώς

και τα αποτελέσματα που προέκυψαν από τα πειράματα. Τέλος, στον 5ο κεφάλαιο, αναφέρονται ορισμένες από τις δυνατότητες και προεκτάσεις, που θα μπορούσε να αποκτήσει η εφαρμογή με την περαιτέρω ανάπτυξή της και την ενσωμάτωση επιπλέον τεχνολογιών.

ΚΕΦΑΛΑΙΟ

2

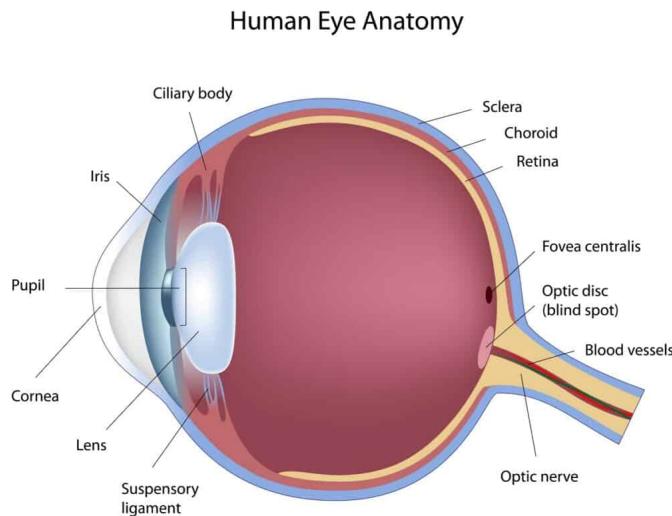
ΘΕΩΡΗΤΙΚΟ ΚΑΙ ΤΕΧΝΟΛΟΓΙΚΟ
ΥΠΟΒΑΘΡΟ

Στο παρόν κεφάλαιο θα πραγματοποιηθεί μια αναλυτική παρουσίαση του θεωρητικού και τεχνολογικού υπόβαθρου, που αποτέλεσε βάση για την υλοποίηση της εφαρμογής. Αρχικά, θα δοθεί μια εξήγηση για το τι εστί απώλεια όρασης, τι δυσκολίες αντιμετωπίζουν τα άτομα με αυτό το είδος αναπηρίας, καθώς και τις τρέχουσες λύσεις για την καταπολέμηση δυσκολιών προσβασιμότητας (Κεφάλαιο 2.1). Στη συνέχεια, θα δοθεί ο ορισμός της εκτεταμένης πραγματικότητας, καθώς και τις εφαρμογές που βρίσκει στην καθημερινότητα του ανθρώπου (Κεφάλαιο 2.2). Τέλος, θα δοθεί μια περιγραφή της συσκευής Microsoft HoloLens 2 και των λειτουργιών της (Κεφάλαιο 2.3), και των εργαλείων, τα οποία θα χρησιμοποιηθούν για την ανάπτυξη της εφαρμογής (Κεφάλαιο 2.4).

2.1 Όραση

2.1.1 Τρόπος Λειτουργίας

Η όραση αποτελεί μία από τις βασικές αισθήσεις του ανθρώπου. Η αίσθηση αυτή βασίζεται στη λειτουργία του ματιού, το οποίο αποτελεί το αισθητήριο όργανο και στο εσωτερικό του οποίου εισέρχεται το φως, διαπερνώντας αρχικά το κερατωειδή χιτώνα και την κόρη και προσπίπτει, τελικά, στον αμφιβληστροειδή χιτώνα (Σχήμα 2.1). Αυτό οδηγεί σε διέγερση των οπτικών νεύρων και τα οπτικά σήματα, που αποστέλλονται στον εγκέφαλο, μετατρέπονται σε εικόνα [6, 7].



Σχήμα 2.1: Η ανατομία του ματιού (Πηγή: dreamstime.com)

2.1.2 Απώλεια Όρασης: Κατηγοροποίηση και Αιτίες

Όταν η φυσιολογική λειτουργία του αισθητήριου οργάνου διαταραχθεί, τότε το άτομο έρχεται αντιμέτωπο με κάποιο τύπο προβλήματος όρασης. Με βάση τον Παγκόσμιο Οργανισμό Υγείας (Π.Ο.Υ.), τα άτομα αυτά κατηγοροποιούνται σε 6 κατηγορίες (Κατηγορία 0 έως Κατηγορία 5), ανάλογα την οπτική τους οξύτητα, δηλαδή την ικανότητά τους να διακρίνουν σχήματα και λεπτομέρειες από κάποια δεδομένη απόσταση:

- Στην κατηγορία 0 ανήκουν άτομα με πλήρως ή σχεδόν πλήρως λειτουργική όραση

- Στην κατηγορία 1 και 2 ανήκουν άτομα που έχουν υποστεί μερική απώλεια της όρασής τους
- Τέλος, στις κατηγορίες 3, 4 και 5 εντάσσονται τα άτομα με σχεδόν πλήρη ή πλήρη απώλεια όρασης

Αντιθέτως, για κοντινές αποστάσεις, τα άτομα με προβλήματα όρασης εντάσσονται σε μία μόνο κατηγορία [8].

Οι κυριότερες αιτίες, οι οποίες προκαλούν προβλήματα όρασης, αποτελούν:

- τα διαθλαστικά σφάλματα
- ο καταρράκτης
- η διαβητική αμφιβληστροειδοπάθεια
- το γλαιύκωμα
- η ηλικιακή εκφύλιση της ωχράς κηλίδας

Από τα ανωτέρω, η κυριότερη αιτία πλήρης απώλειας όρασης σε άτομα ηλικίας 50 ετών και άνω αποτελεί ο καταρράκτης, σε ποσοστό 46% των ατόμων που υπέστησαν πλήρη απώλεια όρασης το 2020. Αντιθέτως, για την ίδια ηλικιακή ομάδα, η οποία αντιμετωπίζει μερική απώλεια όρασης, κύριες αιτίες αποτελούν τα υποδιορθωμένα διαθλαστικά σφάλματα και ο καταρράκτης, σε ποσοστό 30% το καθένα για το ίδιο έτος [9].

2.1.3 Απώλεια Όρασης: Αντίκτυπος

Η απώλεια όρασης έχει ιδιαίτερο αντίκτυπο στην προσωπική, κοινωνική και οικονομική ζωή του ατόμου, ανεξάρτητα της ηλικίας του. Η εκπλήρωση απλών καθημερινών εργασιών μπορεί να αποδειχθεί ιδιαίτερα δύσκολη και χρονοβόρα, επιδεινώνοντας ιδιαίτερα την ποιότητα ζωής του ατόμου [10, 11], ακόμη και σε επίπεδο χαμηλότερο από αυτό ατόμων που αντιμετωπίζουν χρόνια νοσήματα [12].

Η εμφάνιση προβλημάτων όρασης σε αρκετά νεαρή ηλικία μπορεί να επηρεάσει αισθητά την ανάπτυξη ικανοτήτων, όπως είναι η κινητήρια και η γνωστική ικανότητα και η κοινωνική καλλιέργεια [2]. Αντιθέτως, στην ενήλικη ζωή του ατόμου, μπορεί να δυσκολέψει την εύρεση εργασίας και να επηρεάσει την φυσική του υγεία, προκαλώντας μέχρι και άγχος και κατάθλιψη [11]. Τέλος, σε άτομα αρκετά μεγάλης ηλικίας, η απώλεια όρασης μπορεί να αποτελέσει τροχοπέδη για βασικές λειτουργίες, όπως είναι το περπατήμα με εγγενές κίνδυνο την πτώση και τον τραυματισμό [2].

2.1.4 Απώλεια Όρασης: Λύσεις

Πλήθος λύσεων είναι διαθέσιμες στο μέσο άνθρωπο, οι οποίες στοχεύουν στην πρόληψη της απώλειας της όρασης ή στην επιδιόρθωση αυτής. Σε απλές περιπτώσεις, όπως είναι η μυωπία, πρεσβυωπία, κ.λ.π., το πρόβλημα μπορεί να επιλυθεί με χρήση διορθωτικών φακών ή εγχειρησης στο αισθητήριο όργανο [2]. Επίσης, για άτομα που δεν μπορούν να επιδιορθώσουν το πρόβλημα όρασής τους, έχουν αναπτυχθεί συσκευές και τεχνολογίες, οι οποίες συνεχώς εξελίσσονται και έχουν ως σκοπό την εξυπηρέτηση του ατόμου σε διάφορες πτυχές της καθημερινότητάς του. Επί δεκαετίες, το μπαστούνι αποτελεί μια αρκετά διαδεδομένη συσκευή, που βοηθά ένα άτομο να αναγνωρίζει εμπόδια, καθώς περιηγείται σε έναν χώρο. Σε συνδυασμό με την απτική πλακόστρωση (Σχήμα 2.2), το άτομο μπορεί να περιηγηθεί σε δημόσιο χώρο, όντας συνεχώς ενήμερο για την ύπαρξη εμποδίων, όπως δρόμοι, διάβαση πεζών, γραμμές τραμ, στη διαδρομή του, ανάλογα με το μοτίβο των πλακών του πεζοδρομίου [13].



Σχήμα 2.2: Απτική πλακόστρωση (Πηγή: vecteezy.com)

Ευρέως διαδεδομένη είναι και η χρήση σκύλων-βιογθών, οι οποίοι είναι ειδικά εκπαιδευμένοι σχετικά με την αναπηρία του ιδιοκτήτη τους με σκοπό να τους εξυπηρετήσουν στις ιδιαίτερες ανάγκες που μπορεί να έχουν. Στην περίπτωση των ατόμων με προβλήματα όρασης, σκοπός τους είναι να κατευθύνουν τον ιδιοκτήτη τους στο προορισμό του, ειδοποιώντας τον για πιθανά εμπόδια [14]. Τέλος, αναπτύχθηκε το σύστημα γραφής Braille, ώστε να είναι εφικτή η ανάγνωση κειμένων με τη βοήθεια της αφής.

Στη σύγχρονη εποχή, όλο και περισσότερες συσκευές κατασκευάζονται λαμβάνοντας εκ των προτέρων υπόψη τη δυνατότητα χρήσης αυτών από άτομα με περιορισμένη όραση. Οι κινητές συσκευές διαθέτουν λογισμικό screen reader (π.χ. TalkBack σε Android συσκεύες ή VoiceOver σε iOS συσκευές), διευκολύνοντας την πλοιήρηση του χρήστη στις εφαρμογές του κινητού, καθώς πραγματοποιούν καταγραφή και αναγνώριση των κειμένων και των λειτουργιών, που βρίσκονται στην οθόνη τη δεδομένη χρονική στιγμή, και ο χρήστης, με συγκεκριμένες χειρονομίες, επιλέγει αν επιθυμεί την ανάγνωση κάποιου κειμένου με χρήση text-to-speech ή την πραγματοποίηση κάποιας ενέργειας [15]. Επιπλέον, η γλώσσα προγραμματισμού HTML δίνει τη δυνανότητα στους προγραμματιστές να κατασκευάσουν ιστοσελίδες, οι οποίες θα είναι προσβάσιμες από άτομα με περιορισμένη όραση, καθώς οι screen readers θα μπορούν να αναγνωρίζουν τμήματα της ιστοσελίδας, όπως επικεφαλίδες, υπερσυνδέσμους, κουμπιά και το σκοπό που επιτελούν [16].

Όσον αφορά την περιήγηση ατόμων σε ανοιχτούς ή κλειστούς χώρους, πληθώρα συσκευών έχουν κατασκευαστεί, οι οποίες βελτιώνουν ήδη υπάρχουσες ή προσφέρουν έναν εναλλακτικό τρόπο λειτουργίας και παροχής βοήθειας. Παραδείγματα τέτοιων συσκευών αποτελούν:

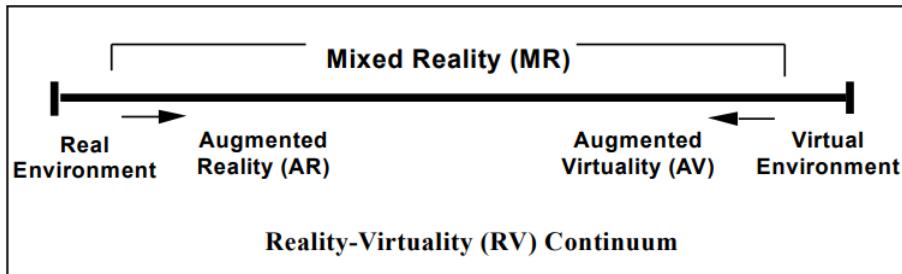
- **biped:** Συσκευή, η οποία φοριέται γύρω από το λαιμό του χρήστη. Διαθέτει 3 κάμερες, που προσφέρουν πεδίο ορατότητας 170°, και ενσωματώνουν λογισμικό για τον εντοπισμό και αναγνώριση εμποδίων. Ο χρήστης προειδοποιείται για εμπόδια με μια ηχητική προειδοποίηση. [17]
- **OrCam MyEye:** Φορητή συσκευή, η οποία προσφέρει τη δυνατότητα ανάγνωσης

κειμένων, αναγνώρισης αντικειμένων, χρωμάτων και προσώπων. [18]

- **BlindSquare:** Εφαρμογή πλούγησης, η οποία παρέχει αναλυτικες οδηγίες στο χρήστη, ώστε να κατευθυνθεί στο προορισμό του, παρέχοντας παράλληλα πληροφορίες για το περιβάλλον του και για σημεία ενδιαφέροντος. [19]
- **Be My Eyes:** Εφαρμογή, η οποία επιτρέπει σε χρήστες να έρθουν σε επικοινωνία με εθελοντές μέσω βιντεοκλήσης, με σκοπό να ζητήσουν βοήθεια. [20]
- **WeWALK:** Αποτελεί ένα εξάρτημα για μπαστούνια, το οποίο έχει τη δυνατότητα να εντοπίζει εμπόδια σε χαμηλό ύψος με χρήση υπερήχων και να ειδοποιεί το χρήστη με δονήσεις και ήχο. [21]
- **Seeing AI:** Εφαρμογή της Microsoft, η οποία, με χρήση τεχνητής νοημοσύνης, παρέχει περιγραφές των αντικειμένων, χρωμάτων, ανθρώπων και κειμένων, τα οποία στοχεύει ο χρήστης με την κάμερα του κινητού του τηλεφώνου. [22]

2.2 Εκτεταμένη Πραγματικότητα

Η εκτεταμένη πραγματικότητα (Extended Reality - XR) αποτελεί μια ιδιαιτέρως ευρεία έννοια, η οποία χρησιμοποιήθηκε πρώτη φορά το 1991 από τους Steve Mann και Charles Wyckoff, κατά την προσπάθεια κατασκευής συσκευών εικονικής/επαυξημένης πραγματικότητας [23, 24]. Αποτελεί «ομπρέλα» εννοιών και περιλαμβάνει τις έννοιες της Εικονικής (Virtual Reality - VR, Κεφάλαιο 2.2.3), της Επαυξημένης (Augmented Reality - AR, Κεφάλαιο 2.2.2) και της Μικτής Πραγματικότητας (Mixed Reality - MR, Κεφάλαιο 2.2.3) [25]. Το 1994, οι Paul Milgram και Fumio Kishino άρισαν ένα συνεχές μεταξύ πραγματικού και εικονικού κόσμου (Σχήμα 2.3). Αυτοί οι δύο κόσμοι αποτελούν τα άκρα της κλίμακας και μεταξύ αυτών υπάρχουν οι έννοιες της επαυξημένης και μικτής πραγματικότητας, καθώς και της επαυξημένης εικονικότητας.



Σχήμα 2.3: Κλίμακα Πραγματικού Κόσμου - Εικονικού Κόσμου [25]

2.2.1 Εικονική Πραγματικότητα

Η εικονική πραγματικότητα αποτελεί μια προσομοίωση, η οποία εντάσσει τον χρήστη σε έναν εικονικό κόσμο, κατασκευασμένος από υπολογιστή, διεγείροντας αισθήσεις όπως η όραση (μπορεί να δει τον κόσμο αυτόν), η ακοή (μπορεί να ακούσει ήχους που προέρχονται από τον κόσμο αυτόν) και η αφή (μπορεί να αντιληφθεί την επαφή με εικονικά αντικείμενα λαμβάνοντας απτική ανάδραση [26, 27]), δίνοντάς του την δυνατότητα να αλληλεπιδράσει με αυτόν [28]. Τα ανωτέρω επιτυγχάνονται με χρήση ειδικών συσκευών, όπως είναι τα γυαλιά εικονικής πραγματικότητας (VR headsets), τα οποία είτε ενσωματώνουν ειδικούς αισθητήρες με σκοπό να ανιχνεύσουν την κίνηση του κεφαλιού, είτε η ανίχνευση αυτή γίνεται με χρήση εξωτερικών

αισθητήρων, που τοποθετούνται σε διάφορα σημεία στον περιβάλλοντα χώρο. Για την ανίχνευση της κίνησης των χεριών, που επιτρέπουν και την αλληλεπίδραση με εικονικά αντικείμενα, απαιτείται η χρήση χειριστηρίων. Παραδείγματα τέτοιων συσκευών αποτελούν το Meta Quest (Σχήμα 2.4) και το Valve Index. Η VR τεχνολογία έχει εισβάλει πλέον και στην καθημερινότητα του μέσου χρήστη βρίσκοντας εφαρμογή σε ποικίλους τομείς:

- Στην ψυχαγωγία και στη διασκέδαση, μέσω των βιντεοπαιχνιδιών που έχουν αναπτυχθεί, την δυνατότητα παρακολούθησης πραγματικών αθλητικών αγώνων, καθώς και εικονικών ξεναγήσεων σε χώρους πολιτιστικού ενδιαφέροντος [29, 30, 31]
- Στην εκπαίδευση, προσφέροντας έναν εναλλακτικό και διαδραστικό τρόπο εκμάθησης [32, 33, 34]
- Στην ιατρική, με σκοπό την εκπαίδευση και προετοιμασία ιατρών, καθώς και για να προσφέρει βοήθεια σε ασθενείς [35, 32, 36, 37]



Σχήμα 2.4: Χρήστης φορά τα γυαλιά εικονικής πραγματικότητας Meta Quest 3

2.2.2 Επαυξημένη Πραγματικότητα

Η επαυξημένη πραγματικότητα αποτελεί μια τεχνολογία, η οποία «ενισχύει» τον πραγματικό κόσμο, ενσωματώνοντας τεχνητά δημιουργημένα, από υπολογιστή, στοιχεία (εικονικά αντικείμενα, ήχους) σε αυτόν. Τα στοιχεία αυτά βρίσκονται εντός ενός «στρώματος» (layer), το οποίο τοποθετείται «πάνω» από τον πραγματικό κόσμο, ο οποίος μπορεί να είναι μια φωτογραφία, ένα βίντεο ή ένα βίντεο πραγματικού χρόνου [38, 39]. Λόγω της εξέλιξης των επεξεργαστών και της απλότητας της τεχνολογίας (σε σχέση με την εικονική πραγματικότητα), ένας χρήστης μπορεί να βιώσει την εμπειρία της επαυξημένης πραγματικότητας χρησιμοποιώντας μια απλή, έξυπνη κινητή συσκευή (smartphone) [40]. Παράλληλα, είναι διαθέσιμα και γυαλιά επαυξημένης πραγματικότητας (AR headsets), όπως είναι τα Xreal Air AR Glasses και Magic Leap 2.

Η επαυξημένη πραγματικότητα βρίσκει εφαρμογή σε παρόμοιους τομείς με αυτούς της εικονικής πραγματικότητας:

- Στην εκπαίδευση [41, 42]
- Στον εργασιακό χώρο, αποτελώντας ένα επιπλέον εργαλείο για τον εργαζόμενο με σκοπό να διευκολύνει το έργο και να βελτιώσει την απόδοσή του [43, 44, 45]
- Στην υγεία [46, 47, 37, 48]

- Στην ψυχαγωγία [49]
- Στον τουρισμό, παρέχοντας ξεναγήσεις, όπου ο χρήστης μπορεί να μάθει περισσότερες πληροφορίες για μνημεία απλά στοχεύοντας την κάμερα του κινητού του σε αυτά [50, 51]

2.2.3 Μικτή Πραγματικότητα

Η ‘μικτή πραγματικότητα’ αποτελεί μια έννοια, η οποία, μέχρι και πρόσφατα, δεν έχει προσδιοριστεί ξεκαθάρα και επιδέχεται πληθώρα ορισμών, οι οποίοι, ωστόσο, διαθέτουν μια κοινή βάση [52]. Βάση του ορισμού που δόθηκε από τους Milgram και Kishino με το Συνεχές Πραγματικού-Εικονικού Κόσμου (Σχήμα 2.3), η μικτή πραγματικότητα αποτελεί οτιδήποτε ανήκει μεταξύ των δύο άκρων της κλίμακας, όντας η επαυξημένη πραγματικότητα και η επαυξημένη εικονικότητα. Με τον όρο επαυξημένη εικονικότητα εννοείται ένας εικονικός κόσμος, στον οποίο ενσωματώνονται πραγματικά αντικείμενα του περιβάλλοντα χώρου [25]. Ορισμένοι ακόμη δημοφιλείς ορισμοί της μικτής πραγματικότητας είναι:

1. Η μικτή πραγματικότητα αποτελεί συνώνυμο της επαυξημένης πραγματικότητας
2. Η τεχνολογίας μικτής πραγματικότητας αποτελεί μια ενισχυμένη μορφή της τεχνολογίας επαυξημένης πραγματικότητας, όπου τα εικονικά αντικείμενα αλληλεπιδρούν με το πραγματικό περιβάλλον και τα αντικείμενα σε αυτόν
3. Η μικτή πραγματικότητα αποτελεί συνδυασμός της επαυξημένης και της εικονικής πραγματικότητας, ενσωματώνοντας στοιχεία και από τις δύο τεχνολογίες

Παράδειγμα συσκευών που αξιοποιούν την τεχνολογία μικτής πραγματικότητας αποτελούν τα Microsoft HoloLens, Apple Vision Pro και Meta Quest Pro. Η μικτή πραγματικότητα βρίσκεται και αυτή ευρεία εφαρμογή σε αρκετούς τομείς της σύγχρονης καθημερινότητας, όπως είναι η εκπαίδευση [53, 54], η υγεία [55, 56], η αρχιτεκτονική [57, 58] κ.α.

2.3 Microsoft HoloLens 2

Το 2015, η Microsoft παρουσίασε τα γυαλιά μικτής πραγματικότητας Microsoft HoloLens, όντας το πρώτο ‘ασύρματο’ ολογραφικό υπολογιστικό σύστημα, δηλαδή δεν ήταν απαραίτητη η χρήση καλωδιών για τη λειτουργία του, καθιστώντας το πλήρως φορητό [59]. Η πρώτη έκδοση του headset, Development Edition, έγινε διαθέσιμη το 2016. Η συσκευή αυτή, καθώς και οι εκδόσεις που ακολούθησαν, ενσωματώνουν την πλατφόρμα Windows Mixed Reality, η οποία βασίζεται στο λειτουργικό σύστημα Windows 10 [60]. Η συσκευή εντάχθηκε σε version ‘μακροχρόνιας υποστήριξης’ (Long-term support, LTS) λαμβάνοντας την τελευταία ενημέρωση τον Οκτώβριο του 2018 [61, 62]. Τρία χρόνια μετά την κυκλοφορία των Microsoft HoloLens, το 2019, παρουσιάζονται σε συνέδριο της Βαρκελώνης τα Microsoft HoloLens 2 (Σχήμα 2.5), τα οποία και τίθονται σε διαθεσιμότητα την ίδια χρονιά (Νοέμβριος 2019) [63]. Είναι διαθέσιμες προς αγορά τρεις εκδόσεις της συσκεύης, ανάλογα με το λόγο και το χώρο χρήσης της [64]:

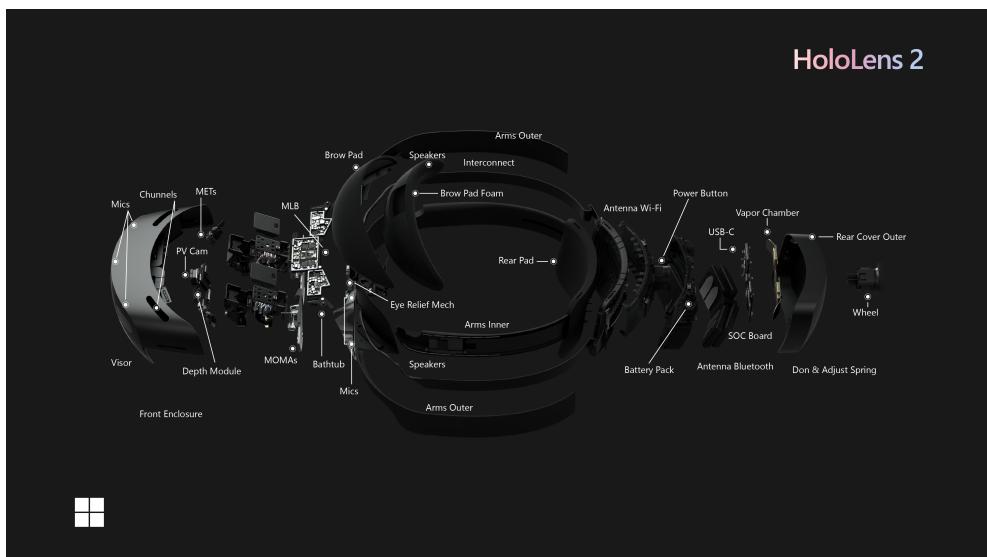
- **HoloLens 2**, για απλή, καθημερινή χρήση
- **HoloLens 2 Industrial Edition**, για χρήση σε ελεγχόμενους χώρους, όπως ένα αποστειρωμένο εργαστήριο
- **Trimble XR10 with HoloLens 2**, ένα προστατευτικό κράνος που ενσωματώνει τη συσκευή HoloLens για χώρους, όπου η ασφάλεια κρίνεται απαραίτητη



Σχήμα 2.5: Η συσκευή μικτής πραγματικότητας Microsoft HoloLens 2 (Πηγή: [theverge.com](https://www.theverge.com))

2.3.1 Τεχνικά Χαρακτηριστικά

Η συσκευή Microsoft HoloLens 2, όπως και ο προκάτοχός της, αποτελεί ένα ‘ασύρματο’ ολογραφικό υπολογιστικό σύστημα. Το λειτουργικό του σύστημα είναι το Windows Holographic OS, το οποίο βασίζεται στο λειτουργικό σύστημα Windows 10 [65]. Ωστόσο, από το πρώτο εξάμηνο του 2023, η Microsoft δίνει τη δυνατότητα στους χρήστες της συσκευής να την αναβαθμίσουν στο λειτουργικό σύστημα Windows 11 [66]. Το headset αποτελείται από πολλά τμήματα/εξαρτήματα και αισθητήρες, όπως μπορούμε να παρατηρήσουμε και στο Σχήμα 2.6.

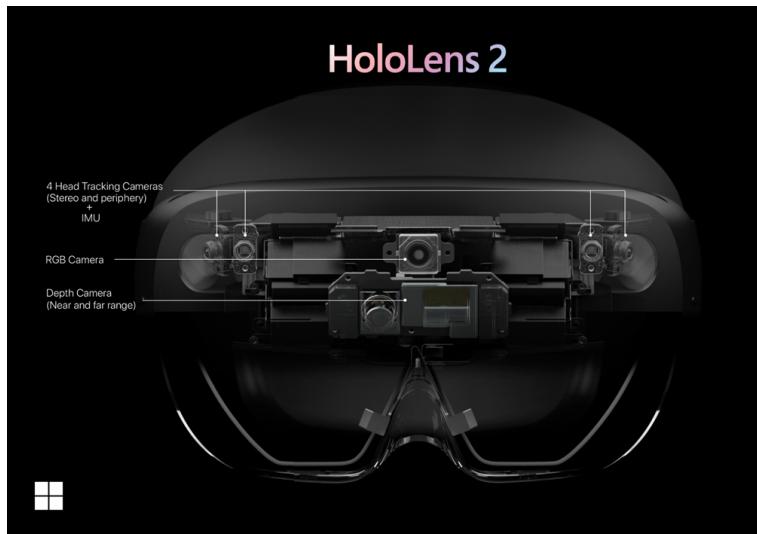


Σχήμα 2.6: Εξαρτήματα και αισθητήρες της συσκευής Microsoft HoloLens 2 (Πηγή: learn.microsoft.com)

Αρχικά, τα εξαρτήματα από τα οποία αποτελείται η συσκευή είναι τα κατώθι [65,

67]:

- **Visor** (Προσωπίδα): Η προσωπίδα διαθέτει τους αισθητήρες και τις οθόνες. Επιπλέον, μπορεί να περιστραφεί προς τα πάνω κατά της χρήσης της συσκευής, σε περίπτωση που ο χρήστης δεν χρησιμοποιεί τις οθόνες και θέλει να παρατηρήσει κάτι στον πραγματικό περιβάλλοντα χώρο.
- **Headband** (Ζώνη κεφαλής): Η ζώνη περιβάλλει το κεφάλι του χρήστη και εξασφαλίζει τη σωστή και σταθερή τοποθέτησή της. Με χρήση μιας ροδέλας, στο πίσω μέρος της συσκευής, ο χρήστης μπορεί να ρυθμίσει πόσο «σφιχτή» ή «χαλαρή» είναι η ζώνη, κατά την αρέσκειά του.
- **Brightness Buttons** (Κουμπιά φωτεινότητας): Με τα κουμπιά ρύθμισης φωτεινότητας, τα οποία βρίσκονται στην αριστερή πλευρά της προσωπίδας, ο χρήστης μπορεί να προσαρμόσει την φωτεινότητα των οθονών ανάλογα με το φως του περιβάλλοντα χώρου.
- **Volume Buttons** (Κουμπιά ήχου): Τα κουμπιά ρύθμισης της έντασης του ήχου βρίσκονται στη δεξιά πλευρά της προσωπίδας.
- **Power Button** (Κουμπί ενεργοποίησης): Το κουμπί ενεργοποίησης (και απενεργοποίησης) της συσκευής βρίσκεται στο δεξί μέρος του εξωτερικού καλύμματος στην πίσω πλευρά της συσκευής.
- **USB-C Port** (Θύρα τύπου USB-C): Η θύρα USB-C, η οποία βρίσκεται κάτω από το κουμπί ενεργοποίησης, χρησιμοποιείται για την φόρτιση της συσκευής και για τη σύνδεση αυτής με άλλες συσκευές.



Σχήμα 2.7: Αισθητήρες στη προσωπίδα (visor) της συσκεύης Microsoft HoloLens 2 (Πηγή: learn.microsoft.com)

Επιπλέον, η συσκευή διαθέτει ένα πλήθος από αισθητήρες, οι οποίοι, όπως αναφέρθηκε, βρίσκονται στο visor (Σχήμα 2.7). Ειδικότερα, διαθέτει [65]:

- 4 κάμερες ορατού φωτός, κάθε μία από τις οποίες έχει πεδίο ορατότητας (Field of View, FOV) 96.1°. Οι κάμερες αυτές χρησιμοποιούνται για τον εντοπισμό της θέσης και του προσανατολισμού του κεφαλιού (**Head Tracking**).
- 2 κάμερες υπέρυθρου φωτός, οι οποίες χρησιμοποιούνται για τον εντοπισμό της κίνησης των ματιών και του βλέμματος (**Eye Tracking**).
- 1 Time-of-Flight αισθητήρα βάθους (**Depth**) του 1 megapixels, δηλαδή ένα αι-

σθήρα που μετρά αποστάσεις βάσει του χρόνου που χρειάζονται τα φωτόνια, που εκπέμπει, να χτυπήσουν ένα στόχο και να επιστρέψουν στο δέκτη του αισθητήρα. Σε συνδυασμό με τις κάμερες υπέρυθρου φωτός, είναι εφικτή η χωρική χαρτογράφηση (Κεφάλαιο 2.3.3) του περιβάλλοντος χώρου.

- 1 μονάδα μέτρησης αδράνειας (**Inertia Measurement Unit**, IMU), η οποία περιλαμβάνει επιταχυνσιόμετρο, γυροσκόπιο και μαγνητόμετρο.
- 1 κάμερα των 8 megapixels, η οποία καταγράφει και βίντεο σε ανάλυση 1080p και 30 FPS.

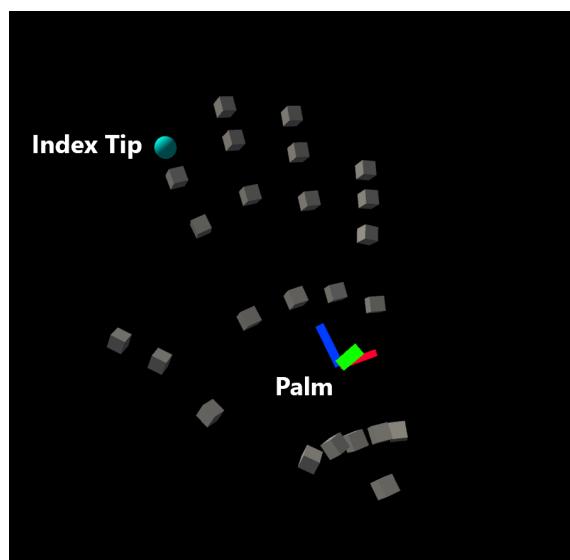
Τέλος, μερικές ακόμη σημαντικές προδιαγραφές της συσκευής αναφέρονται στον Πίνακα 2.1.

Πίνακας 2.1: Τεχνικές προδιαγραφές του headset Microsoft HoloLens 2

Μικρόφωνο	Συστοιχία 5 καναλιών
Ηχείο	Ενσωματώνει τεχνολογία χωρικού ήχου (Κεφάλαιο 2.3.4)
System-on-Chip	Qualcomm Snapdragon 850 Compute Platform
Holographic Processing Unit (Ολογραφική Μονάδα Επεξεργασίας)	Δεύτερης γενιάς, ειδικά κατασκευασμένη Holographic Processing Unit
Μνήμη	4GB
Αποθηκευτικός Χώρος	64GB
WiFi	802.11ac 2x2
Bluetooth	5.0
Βάρος	566 γραμμάρια

2.3.2 Τρόποι Αλληλεπίδρασης

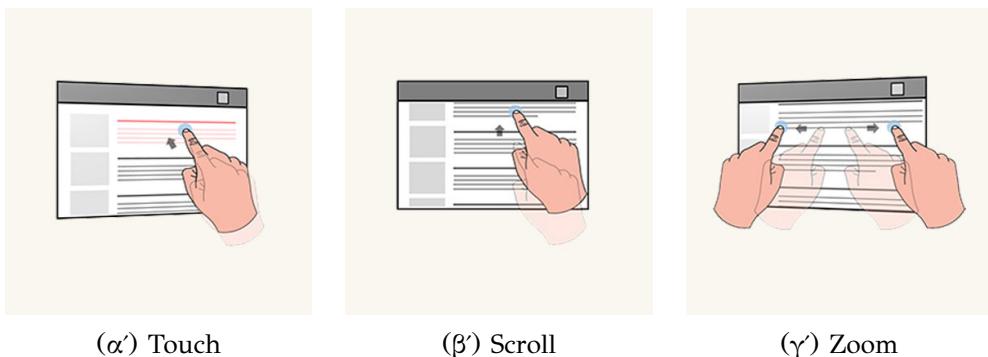
Ο χρήστης έχει στη διάθεση τρεις βασικούς τρόπους με τους οποίους μπορεί να αλληλεπιδράσει με τη συσκευή και τα ολογράμματα που δημιουργεί [65]:



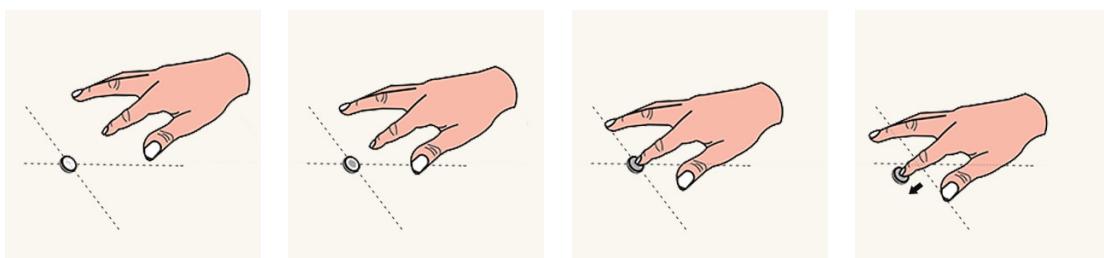
Σχήμα 2.8: Εικονική αναπαράσταση χεριού (Πηγή: learn.microsoft.com)

1. Με εντοπισμό και χρήση των χεριών (Hand Tracking): Είναι δυνατός ο εντοπισμός των κινήσεων και των δύο χεριών, καθώς και των δαχτύλων αυτών, όταν αυτά βρίσκονται εντός του πεδίου ορατότητας (FOV) της συσκευής. Τα εικονικά χέρια που δημιουργούνται για την αλληλεπίδραση αποτελούν ένα σύνολο σημείων (Σχήμα 2.8), όπου κάθε ένα αντιπροσωπεύει σύνδεσμο των χεριών ή σημεία αυτών [68].

Ο χρήστης μπορεί να αλληλεπιδράσει με τα εικονικά αντικείμενα, είτε από κοντινή [69], είτε από μακρινή απόσταση [70] και να πραγματοποιήσει συγκεκριμένες χειρονομίες (Gestures), όπως είναι το Air Tap, το Tap and Hold [71] και το Start Gesture (επιτρέπει την πρόσβαση στο κεντρικό μενού) [72], ώστε να υποδείξει την ενέργεια που επιθυμεί. Για παράδειγμα, με τους δείκτες των χεριών, ο χρήστης μπορεί να αγγίξει αντικείμενα (Touch) (Σχήμα 2.9α'), να πιέσει κουμπιά (Press) (Σχήμα 2.10), να κάνει ανακύλιση μιας σελίδας (Scroll) (Σχήμα 2.9β') ή να κάνει Zoom (Σχήμα 2.9γ') [69].



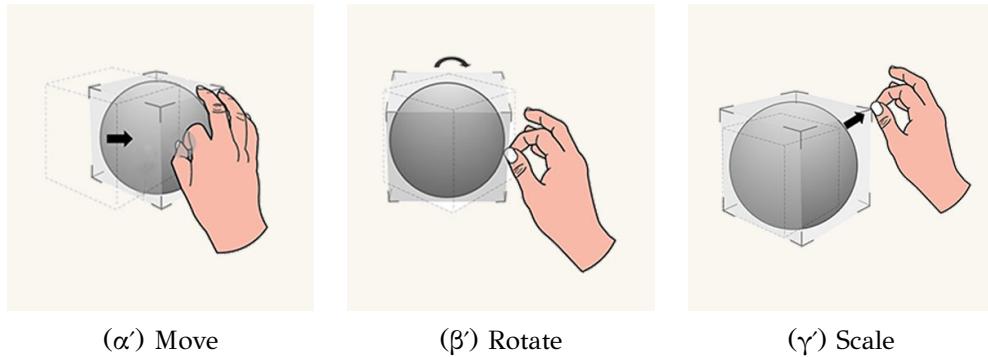
Σχήμα 2.9: Τρόποι αλληλεπίδρασης με τους δείκτες των χεριών με 2D αντικείμενα
(Πηγή: learn.microsoft.com)



Σχήμα 2.10: Πίεση (Press) εικονικού κουμπιού (Πηγή: learn.microsoft.com)

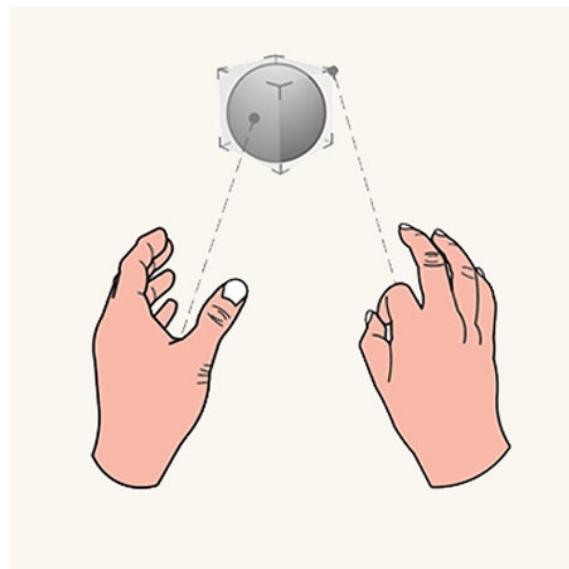
Αν ο χρήστης κρατήσει τον αντίχειρα και τον δείκτη ενωμένο, τότε μπορεί να μετακινήσει (Move) (Σχήμα 2.11α') και να αλλάξει την κλίμακα (Scale) (Σχήμα 2.11γ'), τόσο δισδιάστατων (2D), όσο και τρισδιάστατων (3D) αντικειμένων. Επιπλέον, για τα 3D εικονικά αντικείμενα, με την ίδια χειρονομία (gesture), μπορεί να τα περιστρέψει (Rotate) (Σχήμα 2.11β'). Το σημείο επαφής με το αντικείμενο είναι αυτό το οποίο καθορίζει ποια ένεργεια από τις τρεις ανωτέρω θα πραγματοποιήσει ο χρήστης [69].

Τέλος, οι ίδιες ενέργειες που αναφέρθηκαν ως τώρα, μπορούν να πραγματοποιηθούν και σε εικονικά αντικείμενα που βρίσκονται μακριά από το χρήστη (σε απόσταση μεγαλύτερη των 50 εκατοστών). Από το δάκτυλο του χρήστη εκτείνεται μια ακτίνα, στο τέλος της οποίας υπάρχει ένας κέρσορας, ώστε να διευ-



Σχήμα 2.11: Τρόποι αλληλεπίδρασης με το δείκτη και τον αντίχειρα με 3D αντικείμενα
(Πηγή: learn.microsoft.com)

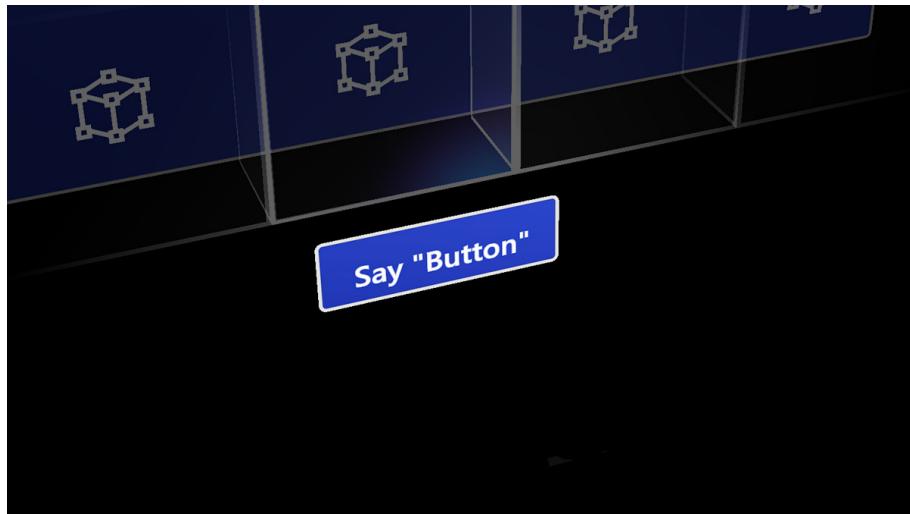
κολύνει τον χρήστη να αντιληφθεί ποιο είναι το σημείο αλληλεπίδρασης με το αντικείμενο [70]. Ο ίδιος δείκτης εμφανίζεται και κατά την αλληλεπίδραση με αντικείμενα σε κοντινή απόσταση.



Σχήμα 2.12: Αλληλεπίδραση με εικονικό αντικείμενο σε μακρινή απόσταση (Πηγή: learn.microsoft.com)

2. **Με το βλέμα και την κίνηση των ματιών ή του κεφαλιού (Gaze and Dwell)** [73]: Ο χρήστης ελέγχει έναν κέρσορα, τον οποίο μπορεί να μετακινήσει εντός του FOV, χρησιμοποιώντας είτε την κίνηση του κεφαλιού του [74], είτε την κίνηση των ματιών (Eye Tracking) [75]. Αυτός ο τρόπος αλληλεπίδρασης μπορεί να αποδειχθεί ιδιαίτερα χρήσιμος, σε περίπτωση που ο χρήστης αδυνατεί να χρησιμοποιήσει τα χέρια του, επειδή, για παράδειγμα, κρατά κάποιο εργαλείο ή αντιμετωπίζει κάποια δυσκολία ή πρόβλημα υγείας, ενώ έχει ακόμη μεγαλύτερη χρηστικότητα, αν συνδυαστεί με φωνητικές εντολές [71, 76].
 3. **Με φωνητικές εντολές (Voice Input):** Ο χρήστης έχει την δυνατότητα να χρησιμοποιήσει φωνητικές εντολές, ώστε να επιτελέσει ενέργειες. Η συσκευή ακολουθεί το μοντέλο ‘See It, Say It’, οπότε οι εντολές που μπορούν να χρησιμοποιηθούν υποδεικνύονται με ετικέτες (Σχήμα 2.13) και κουμπιά. Τέλος, αυτός ο τρόπος

αλληλεπίδρασης μπορεί να συνδυαστεί με οποιονδήποτε από τους δύο προηγούμενους για λόγους ευκολίας ή/και αύξησης απόδοσης [76].



Σχήμα 2.13: Υπόδειξη διαθέσιμης φωνητικής εντολής μέσω ετικέτας (Πηγή: learn.microsoft.com)

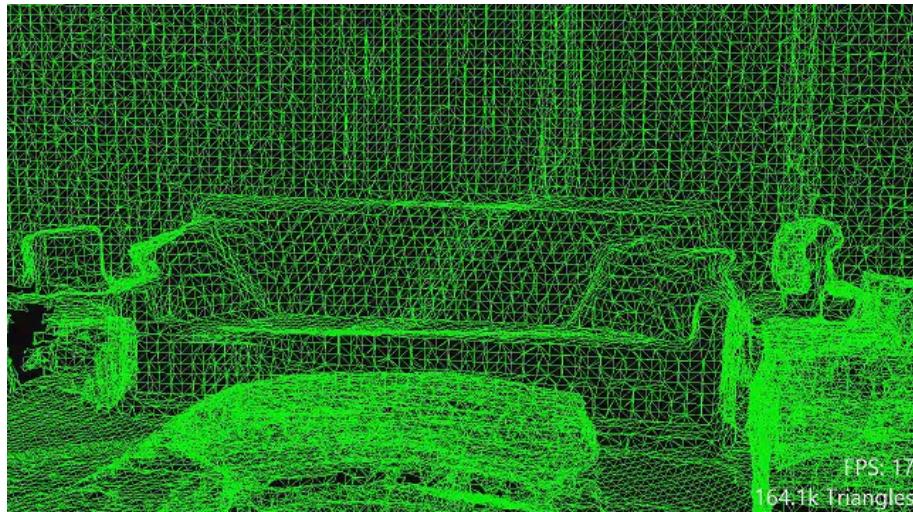
2.3.3 Χωρική Χαρτογράφηση (Spatial Mapping)

Η χωρική χαρτογράφηση (spatial mapping) αποτελεί μία από τις κύριες δυνατότητες της συσκευής Microsoft HoloLens και είναι το κύριο χαρακτηριστικό, στο οποίο στηρίζονται οι εφαρμογές που αναπτύσσονται για τη συσκευή για να υλοποιήσουν τις λειτουργίες μικτής πραγματικότητας. Χωρική χαρτογράφηση είναι η σάρωση και η δημιουργία μίας 3D αναπαράστασης του περιβάλλοντος χώρου του χρήστη (Σχήμα 2.14). Αυτό επιτρέπει στα εικονικά αντικείμενα να αλληλεπιδρούν με αντικείμενα του πραγματικού χώρου, κάθως στην πραγματικότητα αλληλεπιδρούν με το ‘πλέγμα’ (mesh), το οποίο είναι ένα σύνολο κορυφών και πολυγώνων (συνήθως τριγώνων) που αντιπροσωπεύουν το αντικείμενο [77], δρώντας ως το εικονικό του αντίγραφο, προσδίδοντας στο χρήστη την ψευδαίσθηση ότι τα αντικείμενα βρίσκονται όντως στο περιβάλλοντα χώρο [78].

Σε αντίθεση με το HoloLens 1, το HoloLens 2 μπορεί να αξιοποιήσει τα δεδομένα που λαμβάνονται από το spatial mapping και, σε συνδυασμό με τεχνητή νοημοσύνη, να σχηματίζει περιοχές (Quads), ομαδοποιώντας πολύγωνα που μπορεί να ανήκουν στο ίδιο πραγματικό αντικείμενο ή/και επιφάνεια (π.χ. τοίχος, πάτωμα, τραπέζι κ.λ.π.) (Σχήμα 2.15). Με αυτό τον τρόπο, τα δεδομένα αποκτούν μια δομή, καθιστώντας ’τα πιο κατανοητά στον προγραμματιστή, που επιθυμεί να τα χρησιμοποιήσει [79]. Αυτό επιτυγχάνεται με το Scene Understanding SDK, το οποίο, πέρα των πλεονεκτημάτων που περιγράφηκαν προηγουμένων, φέρει και ορισμένα μειονεκτήματα σε σχέση με το spatial mapping, όπως είναι η καθυστέρηση ανανέωσης των δεδομένων, καθώς τα δεδομένα που αξιοποιεί είναι στατικά [78].

Οι κυριότεροι λόγοι χρήσης της τεχνολογίας της χωρικής χαρτογράφησης είναι [78]:

- Η τοποθέτηση εικονικών αντικειμένων σε πραγματικές επιφάνειες.
- Η απόκρυψη εικονικών αντικειμένων, όταν βρίσκονται πίσω από πραγματικές επιφάνειες.



Σχήμα 2.14: Τριασδιάστατη ανάπταρασταση ένος χώρου (Πηγή: learn.microsoft.com)

- Η εφαρμογή φυσικής σε εικονικά αντικείμενα. Με αυτό τον τρόπο, τα ολογράμματα μπορούν να αλληλεπιδρούν με πιο ρεαλιστικό τρόπο με τα πραγματικά αντικείμενα.
- Η πλοιήγηση εικονικών αντικειμένων, καθώς και χρηστών, στον πραγματικό χώρο.

Τέλος, πρέπει να υπογραμμιστεί ότι υπάρχει ένας αριθμός παραγόντων, οι οποίοι μπορεί να επηρεάσουν την αποτελεσματικότητα και την ακρίβεια της χωρικής χαρτογράφησης. Τέτοιοι παράγοντες είναι ο φωτισμός του χώρου, η έλλειψη ύπαρξης αναγνωρίσιμων χαρακτηριστικών πάνω σε απλές επιφάνειες, η ύπαρξη wormholes (σημεία χώρων που φέρουν τα ίδια ή παρόμοια χαρακτηριστικά, με αποτέλεσμα, να θεωρούνται, εσφαλμένα, από τη συσκευή ως ο ίδιος χώρος), οι συχνές αλλαγές και κινήσεις στο χώρο, οι ανακλαστικές επιφάνειες κ.α. [80].



Σχήμα 2.15: Περιοχές (Quads) που σχηματίστηκαν από το Scene Understanding (Πηγή: learn.microsoft.com)

2.3.4 Χωρικός Ήχος (Spatial Audio)

Η συσκευή Microsoft HoloLens 2 ενσωματώνει, επίσης, την τεχνολογία χωρικού ήχου, η οποία συνεισφέρει ακόμη περισσότερο στη μίξη του πραγματικού και του εικονικού κόσμου, προσφέροντας μια ρεαλιστική εμπειρία στο χρήστη. Η τεχνολογία βασίζεται στην ικανότητα του ανθρώπου να αντιληφθεί την θέση και την κατεύθυνση από την οποία προήλθε κάποιος ήχος. Με τη χρήση των δύο ηχείων, τα οποία βρίσκονται εκατέρωθεν του κεφαλιού του χρήστη, και τεχνολογίας βασισμένη στη Head-related συνάρτηση μεταφοράς (Head-related Transfer Function, HRTF), είναι εφικτό η συσκευή να παράγει ήχο, για τον όποιο, θα είναι εφικτό ο χρήστης να υποδείξει την

κατεύθυνση προέλευσης αυτού [81]. Η HRTF αποτελεί το λόγο του μετασχηματισμού Fourier της πίεσης του ήχου στην είσοδου του καναλιού του αυτιού και της πίεσης αυτής στο μέσο του κεφαλιού [82]. Ανάλογα με το σχήμα και το μέγεθος του αυτιού του χρήστη, η συσκευή μπορεί να προσαρμόσει την HRTF με σκοπό να βελτιώσει την εμπειρία του χρήστη. Το spatial audio χρησιμοποιείται κυρίως για να τραβήξουμε την προσοχή του χρήστη προς κάποια συγκεκριμένη κατεύθυνση ή κάποιο συγκεκριμένο ολόγραμμα, το οποίο μπορεί να βρίσκεται εκτός του πεδίου ορατότητάς του [81].

2.4 Εργαλεία ανάπτυξης για HoloLens

Για την ανάπτυξη εφαρμογών για την συσκευή Microsoft HoloLens 2, είναι απαραίτητη η εγκατάσταση συγκεκριμένων προγραμμάτων και λογισμικών από μια λίστα εργαλείων που παρέχει η Microsoft [83, 84]. Η επιλογή των εργαλείων γίνεται από τον προγραμματιστή ανάλογα με την εξοικείωση του με τα εργαλεία αυτά και τις γλώσσες προγραμματισμού που χρησιμοποιούν, καθώς και το διαθέσιμο υλικό (documentation). Στο παρόν κεφάλαιο, θα δοθεί μια σύντομη περιγραφή των εργαλειών που θα αξιοποιηθούν για την ανάπτυξη της εφαρμογής, καθώς και των δυνατοτήτων που παρέχουν.

2.4.1 Unity

Η Microsoft δίνει την δυνατότητα στον προγραμματιστή να επιλέξει μεταξύ τεσσάρων μηχανών (software engines) και APIs στην οποία μπορεί να υλοποιήσει την εφαρμογή του [84], τα οποία παρατίθονται στον Πίνακα 2.2, μαζί με τη γλώσσα προγραμματισμού που χρησιμοποιεί κάθε engine ή API. Στα πλαίσια ανάπτυξης της εφαρμογής μας, ως μηχανή επιλέχθηκε η πλατφόρμα ανάπτυξης τρισδιάστατων εφαρμογών Unity, λόγω του πλούσιου documentation που παρέχεται από την Microsoft [85] και από την Unity Technologies, καθώς και του υλικού και των οδηγών, που είναι διαθέσιμα στο διαδίκτυο (forums, videos κ.λ.π.) και παρεχόνται από μια ιδιαιτερά ενεργή κοινότητα.

Πίνακας 2.2: Διαθέσιμα engines και APIs για την ανάπτυξη εφαρμογών για το Microsoft HoloLens 2

Engine και API	Γλώσσα Προγραμματισμού
Unity (Engine)	C#
Unreal Engine	C++
Custom Engine με χρήση του OpenXR API	C/C++
Web development με χρήση του WebXR API	JavaScript

Η Unity είναι μια πλατφόρμα ανάπτυξης τρισδιάστατων εφαρμογών, η οποία αναπτύχθηκε από την Over the Edge Entertainment (η οποία μετανομάστηκε σε Unity Technologies το 2007) και κυκλοφόρησε στις 8 Ιουνίου 2005. Η μηχανή, αρχικά, αναπτύχθηκε για την υλοποίηση εφαρμογών στο λειτουργικό σύστημα Mac OS X, ενώ στη συνέχεια έγινε δυνατή η ανάπτυξη σε πληθώρα πλατφορμών, όπως οι κινητές συσκευές (Android, iOS), υπολογιστές (Mac, Windows, Linux), παιχνιδοκονσόλες, πλατφόρμες εικονικής/εκτεταμένης πραγματικότητας κ.α. Η Unity επιτρέπει την ανάπτυξη τρισδιάστατων (3D), καθώς και δισδιάστατων (2D) εφαρμογών και χρησιμοποιείται σε κλάδους και βιομηχανίες, όπως η βιομηχανία βιντεοπαιχνιδιών, κινηματογράφου, ο

κλάδος της αρχιτεκτονικής κ.α. Η τρέχουσα LTS έκδοση της μηχανής παιχνιδιών είναι η Unity 2023.2.1. Ωστόσο, για λόγους συμβατότητας με άλλα εργαλεία που θα χρησιμοποιηθούν, κατά την ανάπτυξη της εφαρμογής προτιμήθηκε η έκδοση Unity 2020.3.48.

Η μηχανή παιχνιδιών Unity αναπτύχθηκε σε γλώσσα προγραμματισμού C++ (runtime) και αξιοποιεί τη C# (Unity Scripting API), μια αντικειμενοστρεφής γλώσσα προγραμματισμού, η οποία αναπτύχθηκε από τη Microsoft. Αποτελεί εφαρμογή υψηλού επιπέδου, δίνοντας στον προγραμματιστή τη δυνατότητα να υλοποιήσει μια εφαρμογή με ελάχιστη γραφή κώδικα (scripts), διαθέτοντας έτοιμες προς χρήση λειτουργίες και αντικείμενα, των οποίων η συμπεριφορά μπορεί να παραμετροποιηθεί.

Εφόσον η γλώσσα προγραμματισμού, που μπορεί να χρησιμοποιήσει ο προγραμματιστής για να αναπτύξει λειτουργίες, που δεν παρέχονται από την μηχανή, είναι η αντικειμενοστρεφής C#, τότε βασικό στοιχείο της Unity είναι η χρήση των κλάσεων (Classes) και των αντικειμένων (Objects), με κυριότερη κλάση να είναι το [GameObject](#) [86]. Όλα τα στοιχεία που μπορούμε να εντοπίσουμε σε μια εφαρμογή, όπως οι χαρακτήρες, η κάμερα, ο φωτισμός, αποτελούν αντικείμενο της κλάσης [GameObject](#). Από μόνα τους, τα αντικείμενα αυτά δεν διαθέτουν κάποια ιδιαίτερη λειτουργία ή συμπεριφορά, αλλά λειτουργούν ως «άδεια δοχεία», στα οποία προστίθενται [Components](#) [87]. Τα [Components](#) αποτελούν τα λειτουργικά μέρη των [GameObjects](#), καθώς είναι αυτά τα οποία καθορίζουν τη συμπεριφορά τους. Κάθε [GameObject](#) μπορεί να διαθέτει άπειρα [Components](#), τα οποία διαθέτουν παραμέτρους, τις οποίες ο προγραμματιστής μπορεί να προσαρμόσει ανάλογα με την επιθυμητή συμπεριφορά. Ωστόσο όλα τα [GameObjects](#) διαθέτουν πάντα ένα και μόνο ένα [Transform](#) component, το οποίο καθορίζει τη θέση, περιστροφή και κλίμακα του αντικειμένου. Επιπλέον, είναι δυνατή η δημιουργία custom [Component](#) από τον προγραμματιστή με τη χρήση του Unity Scripting API, δηλαδή με τη συγγραφή αρχείων κώδικα σε C# (C# scripts) [88]. Ένα [GameObject](#), επίσης, μπορεί να έχει πολλαπλά ‘children’ [GameObjects](#), το καθένα με τα δικά του [Components](#) και παραμετροποιήσεις. Άν ο χρήστης θέλει να χρησιμοποιήσει ένα τέτοιο σύνολο από [GameObjects](#) με τα [Components](#) που διαθέτουν, τότε, για λόγους ευκολίας, μπορεί να δημιουργήσει ένα [Prefab](#), με βάση το οποίο δημιουργεί πολλαπλά αντίγραφα με κοινή συμπεριφορά.

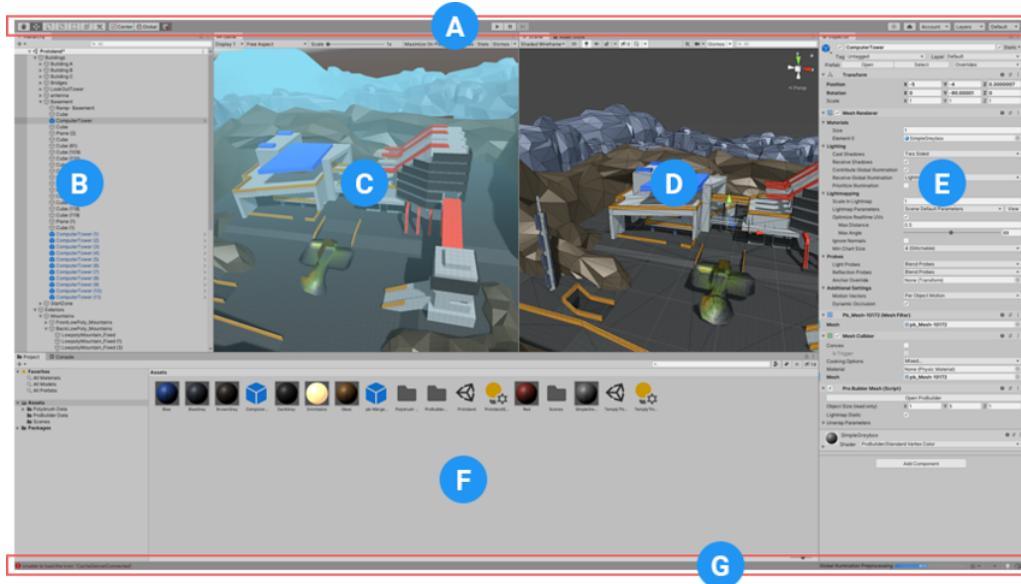
Η βασική διεπαφή της Unity, όπως μπορούμε να παρατηρήσουμε και στο Σχήμα 2.16, αποτελείται από 7 κύριες περιοχές/παράθυρα. Ειδικότερα, οι περιοχές αυτές είναι [89]:

(A) Toolbar: Το toolbar σου δίνει πρόσβαση σε ένα πλήθος εργαλείων για [90]:

- τη διαχείριση των [GameObjects](#) στο Scene view
- τη διαχείριση της ροής στο Game view
- τον έλεγχο και τη διαχείριση του λογαριασμού Unity του χρήστη
- την επιλογή του Layer, όπου τα αντικείμενα που ανήκουν σε αυτόν θα είναι ορατά στο Scene view
- την επιλογή της διάταξης των παραθύρων στη διεπαφή

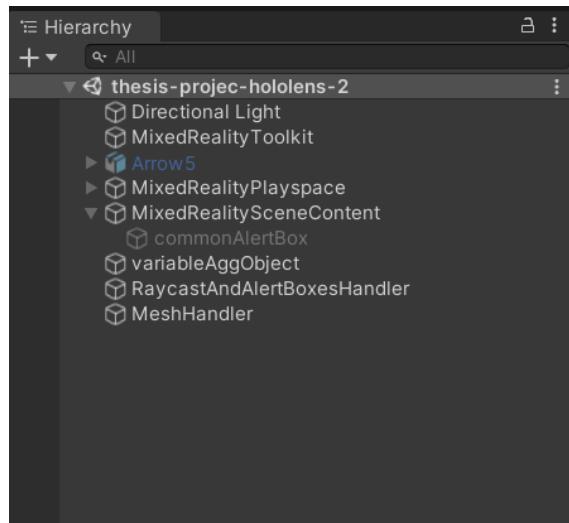
(B) Hierarchy Window: Το παράθυρο αυτό (Σχήμα 2.17) παρουσιάζει, σε μορφή λίστας, τα [GameObjects](#) που βρίσκονται σε μια σκηνή (Scene) και τη δομή αυτών. Σκηνή (Scene) αποτελεί το αρχείο που περιέχει το περιβάλλον, το μενού και τα αντικείμενα της εφαρμογής.

Επιπλέον, έχουμε τη δυνατότητα να αλλάξουμε τη σειρά και το ‘parenting’ των [GameObjects](#), καθώς και να δημιουργήσουμε, να αντιγράψουμε ή να διαγρά-



Σχήμα 2.16: Το περιβάλλον ανάπτυξης (Editor) της Unity (Πηγή: docs.unity3d.com)

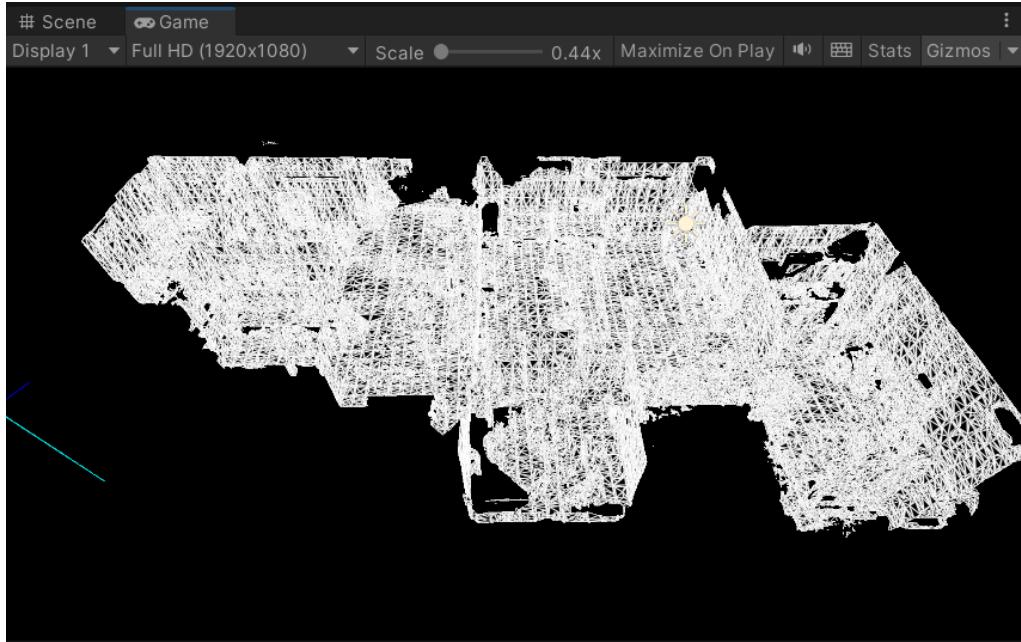
φουμε ένα **GameObject**. Τέλος, μπορούμε και να επιλέξουμε ποια **GameObjects** θα είναι εμφανή στο Scene view [91].



Σχήμα 2.17: Το Hierarchy window του Editor

- (C) Game View: Το Game view (Σχήμα 2.18) αποτελεί το παράθυρο στο οποίο παρουσιάζεται η τελική μορφή την οποία έχει η εφαρμογή μας, όπως αυτή γίνεται render από τις κάμερες (**Camera** component), που έχουμε στη σκηνή. Από τα κουμπιά που βρίσκονται στο toolbar μπορούμε να ορίσουμε τη ροή εκτέλεσης της εφαρμογής (έναρξη, παύση, τερματισμός κ.α.) στο Play mode. Play mode ονομάζεται η λειτουργία κατά την οποία εκτελείται η εφαρμογή και παρουσιάζεται στο Game view. Κατά την λειτουργία, μπορούν να πραγματοποιηθούν αλλαγές σε διάφορα στοιχεία, όπως **GameObjects**, παραμέτρους από **Components** κ.α., ωστόσο οι αλλαγές αυτές είναι προσωρινές και αναιρούνται, όταν ολοκληρωθεί η εκτέλεση της εφαρμογής. Τέλος, στην κορυφή του παραθύρου, υπάρχει μια

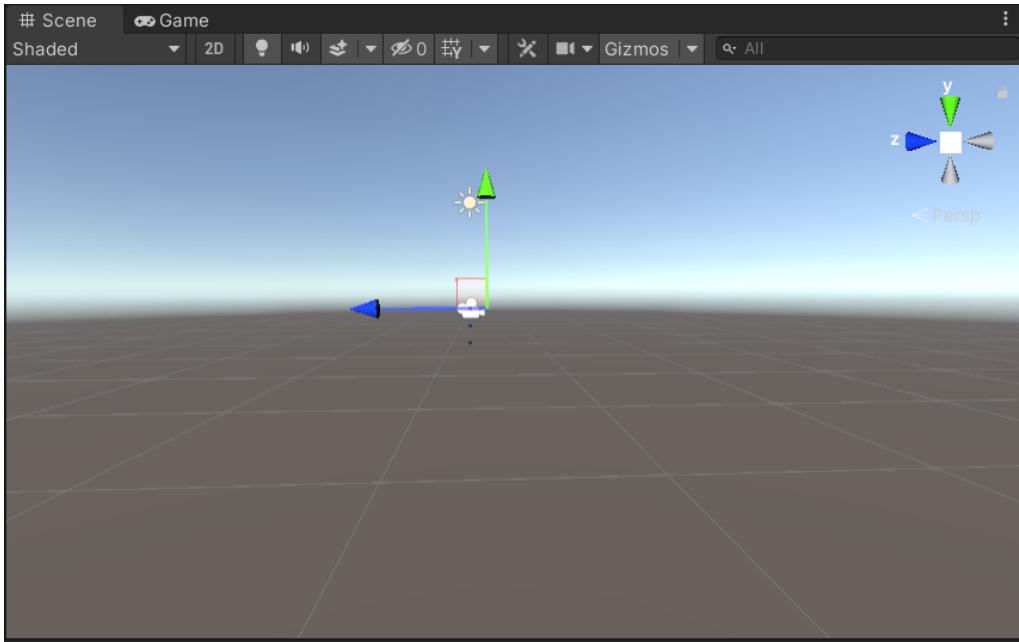
εργαλειοθήκη ώστε να προσαρμόσει ο χρήστης το Game view παράθυρο και το περιεχόμενο αυτού [92].



Σχήμα 2.18: Το Game view του Editor

- (D) Scene View: Αποτελεί το παράθυρο (Σχήμα 2.19), όπου ο χρήστης μπορεί να αλληλεπιδράσει με τα Game Objects τα οποία έχει εισάγει σε μια σκηνή. Στη Scene view, έχουμε μια τρισδιάστατη ή δισδιάστατη προοπτική της εφαρμογής, όπως αυτή εμφανίζεται πριν την εκτέλεση κάποιου κώδικα. Τέλος, όπως και στο Game view, στην κορυφή του παραθύρου, υπάρχει μια εργαλειοθήκη όπου ο χρήστης μπορεί να τροποποιήσει το Scene view με βάση τις ανάγκες, όπως είναι η απόκρυψη των Gizmos, η αφαίρεση του φωτισμού και η αλλαγή από 3D σε 2D προοπτική και αντίστροφα [93].
- (E) Inspector Window: Από το παράθυρο αυτό (Σχήμα 2.20), ο χρήστης μπορεί να δει και να τροποποιήσει παραμέτρους πολλών στοιχείων του Editor, όπως είναι τα **GameObjects**, τα **Components**, τα υλικά (Materials), τα Assets, καθώς και ρυθμίσεις του Editor. Επιλέγοντας κάποιο από τα ανωτέρω στοιχεία, στον Inspector, εμφανίζονται όλοι οι παράμετροι, σχετικές με το το στοιχείο αυτό, τις οποίες μπορεί να αλλάξει ο χρήστης. Για παράδειγμα, αν επιλεχθεί ένα **GameObject**, τότε εμφανίζονται όλα τα **Components** που έχουν προστεθεί σε αυτόν. Αν κάποιο από τα **Components** αποτελεί Script που έχει δημιουργήσει ο χρήστης, τότε στον Inspector παρουσιάζονται οι public μεταβλητές του αρχείου αυτού [94].
- (F) Project Window: Στο συγκεκριμένο παράθυρο εντοπίζονται όλα τα αρχεία, τα οποία σχετίζονται με την εφαρμογή, που υλοποιεί ο χρήστης. Ο χρήστης μπορεί να ανατρέξει στο παράθυρο αυτό σε περίπτωση που αναζητεί κάποιο Asset, το οποίο έχει ξαναχρησιμοποιήσει ή έχει αποθηκεύσει σε κάποιο φάκελο του project [95].

Το Asset είναι οποιοδήποτε αντικείμενο χρησιμοποιείται κατά την ανάπτυξη της εφαρμογής, όπως είναι ένα αρχείο ήχου, μια εικόνα, ένα 3D μοντέλο [96]. Αυτά τα assets μπορούν να έχουν δημιουργηθεί από το χρήστη ή από ένα τρίτο άτομο και έχουν αποκτήθει από το χρήστη από καταστήματα, όπως το Unity Asset



Σχήμα 2.19: To Scene view του Editor

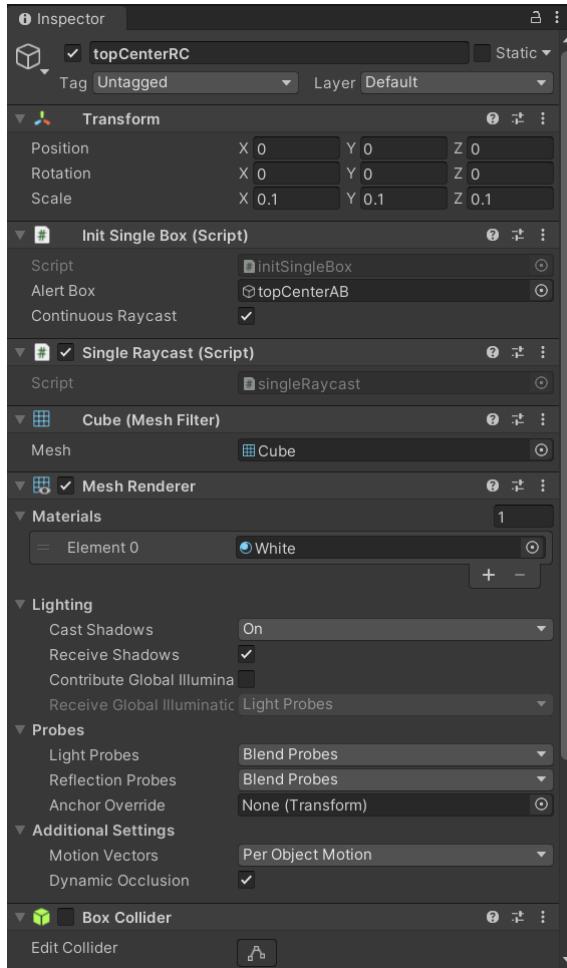
Store.

Τέλος, όπως και σε άλλες περιοχές του Editor, στην κορυφή του παραθύρου υπάρχει μια εργαλειοθήκη, που επιτρέπει στο χρήστη να προσθέσει Assets στον τρέχοντα φάκελο ή να αναζητήσει κάποιο Asset με χρήση διάφορων φίλτρων [95].

- (G) Status Bar: Στο κάτω μέρος του Editor εντοπίζεται η μπάρα κατάστασης (Status Bar) (Σχήμα 2.22), η οποία ενημερώνει σχετικά με την κατάσταση του Editor ή διαφόρων εργασιών που επιτελεί. Ειδικότερα, παρουσιάζεται μια προεπισκόπηση του τελευταίου μηνύματος, προειδοποίησης ή σφάλματος που καταγράφηκε στην κονσόλα (Console). Επίσης, αν επιτελείται κάποια εργασία, εμφανίζεται μια μπάρα προόδου της εργασίας αυτής. Τέλος, στο δεξί μέρος του status bar, υπάρχουν ορισμένα κουμπιά και ενδείξεις, που επιτρέπουν στο χρήστη [97]:
- να αλλάξει ανάμεσα σε λειτουργία Debug και Release, το οποίο επηρεάζει την βελτιστοποίηση του κώδικα
 - να ελέγξει την κατάσταση του cache server
 - να ελέγξει την κατάστηση του Global illumination
 - να ελέγξει την τρέχουσα κατάσταση του Editor, σε περίπτωση που αυτός πραγματοποιεί compilation αρχείων C# ή άλλες ασύγχρονες εργασίες

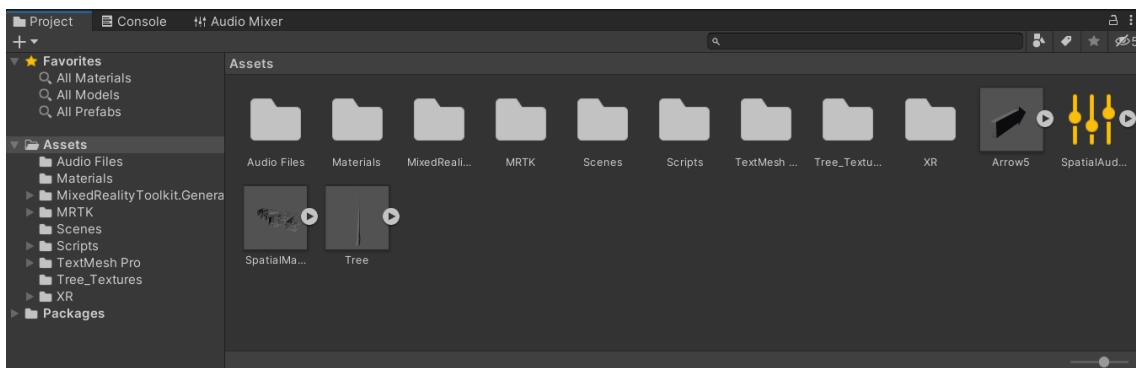
2.4.2 Microsoft Visual Studio

To Microsoft Visual Studio αποτελεί το κύριο ολοκληρωμένο περιβάλλον ανάπτυξης (Integrated Development Environment, IDE) για την ανάπτυξη εφαρμογών στη Unity, όπως είναι οι εφαρμογές για τη συσκευή HoloLens [83]. Κατά τη δημιουργία ενός project στη Unity, δημιουργείται ένα Visual Studio Project. Ο συγκεκριμένος editor μπορεί να χρησιμοποιηθεί για τη συγγραφή κώδικα σε γλώσσα C#, για τη compilation των αρχείων κώδικα και το build του project, αλλά και για την απο-



Σχήμα 2.20: To Inspector window του Editor

σφαλμάτωση (debugging) του κώδικα. Τέλος, μέσω του Visual Studio, είναι δυνατό η φόρτωση του project στη συσκευή [98]. Ωστόσο, αυτή είναι μια ιδιαιτέρως χρονοβόρα διαδικασία, η οποία δημιουργεί επιπλέον καθυστερήσεις στην ανάπτυξη της εφαρμο-



Σχήμα 2.21: To Project window του Editor

[AsyncCoroutineRunner] There is no AsyncCoroutineRunner in the scene. Adding a GameObject with AsyncCoroutineRunner

Σχήμα 2.22: To Status bar του Editor

γής, όπου οι συχνές αλλαγές του κώδικα και οι δοκιμές αποτελούν συχνό φαινόμενο. Για το λόγο αυτό, η μεταφόρτωση της εφαρμογής πραγματοποιείται μέσω του εργαλείου Holographic Remoting που προσφέρει το MRTK (Κεφάλαιο 2.4.3) μέσω της Unity.

Στα πλαίσια της παρούσας διπλωματικής εργασίας, εγκαταστάθηκε η έκδοση του προγράμματος Visual Studio 2019, καθώς και τα απαιραίτητα workloads για την ανάπτυξη εφαρμογών μικτής πραγματικότητας στη Unity.

2.4.3 Mixed Reality Toolkit

Το Mixed Reality Toolkit (MRTK) αποτελεί μια εργαλειοθήκη, η οποία αναπτύχθηκε από την Microsoft, ενσωματώνεται σε project στη Unity ως package και έχει ως σκόπο να διευκολύνει τον προγραμματισμό στην ανάπτυξη εφαρμογών μικτής πραγματικότητας, παρέχοντας μια σειρά από εργαλεία. Πιο συγκεκριμένα, παρέχει σημαντικά [GameObjects](#) και [Components](#), τα οποία μπορούν να χρησιμοποιηθούν σε ένα εικονικό περιβάλλον ή σε αντικείμενα που υπάρχουν σε αυτόν.

Επίσης, δίνει τη δυνατότητα στο χρήστη να πραγματοποιήσει μια προσομοίωση της χρήσης της εφαρμογής μέσω του editor της Unity, χωρίς τη φόρτωση του project σε συσκευή HoloLens. Ωστόσο, στην περίπτωση αυτή, δεν είναι πάντα λειτουργικά όλα τα στοιχεία μικτής πραγματικότητας που έχουμε προσθέσει στο project και προσφέρει το MRTK [99].

Όπως αναφέρθηκε, το MRTK προστίθεται ως package σε ένα Unity project. Αυτό με τη χρήση του εργαλείου Mixed Reality Feature Tool, το οποίο αναπτύχθηκε επίσης από τη Microsoft. Μέσω του εργαλείου αυτού, μπορεί ο χρήστης να αναζητήσει και να προσθέσει στο Unity project του εργαλεία και components χρήσιμα για εφαρμογές μικτής πραγματικότητας [100].

Για την ανάπτυξη της εφαρμογής μας εγκαταστήσαμε, μέσω του Mixed Reality Feature Tool, το MRTK 2.7.3, καθώς και τα packages Mixed Reality OpenXR plugin και Microsoft Spatializer.

ΚΕΦΑΛΑΙΟ

3

Η ΥΛΟΠΟΙΗΣΗ

Σκοπός του κεφαλαίου αποτελεί η παρουσίαση της διαδικασίας ανάπτυξης της εφαρμογής για τη συσκευή Microsoft HoloLens 2. Πιο συγκεκριμένα, θα παρουσιάσουμε αρχικά τον κύριο στόχο της εφαρμογής, ποιος είναι ο σκοπός που εξυπηρέτει και σε ποιο πρόβλημα προσπαθεί να προσφέρει λύση (Κεφάλαιο 3.1). Έπειτα, θα γίνει αναφορά στις αποφάσεις που λήφθηκαν κάτα την σχεδίαση και υλοποίηση της εφαρμογής με βάση και τους περιορισμούς που τέθηκαν (Κεφάλαιο 3.2). Το κεφάλαιο θα ολοκληρωθεί με μια αναλυτική περιγραφή της διαδικασίας υλοποίησης της εφαρμογής και επεξήγηση του κώδικα που αναπτύχθηκε, καθώς και με την παρουσίαση όλων των κύριων λειτουργιών (Κεφάλαιο 3.3).

3.1 Σενάριο Εφαρμογής

Τα άτομα με προβλήματα όρασης έρχονται καθημερινά αντιμέτωπα με ένα πρόβλημα, το οποίο, για το μέσο πληθυσμό αποτελεί μια απλή καθημερινή ενέργεια: η απροβλημάτιστη μετακίνηση και η περιήγηση σε δημόσιους και ιδιωτικούς χώρους. Ως τώρα, παρά την τεχνολογική εξέλιξη και την ανάπτυξη σύγχρονων λύσεων, όπως αυτές περιγράφηκαν στο Κεφάλαιο 2.1.4, για την πλειονότητα αυτών των ατόμων, κύριος υποστηρικτής στην προσπάθεια αυτή αποτελεί το μπαστούνι, λόγω της απλότητας χρήσης του και του κόστους του.

Βασικός στόχος της εφαρμογής αποτελεί η παροχή βοήθειας σε τυφλούς και άτομα με προβλήματα όρασης, κατά την περιήγησή τους σε ένα αγνωστό προς αυτούς χώρο με ευκολία και ασφάλεια, χωρίς τη βοήθεια κάποιου επιπλέον οργάνου. Δηλαδή, η εφαρμογή πρέπει να προσφέρει ένα απλό και εύκολα κατανοητό τρόπο λειτουργίας, η οποία θα προσομοιάζει τον τρόπο χρήσης ενός μπαστονιού.

Αρχικά, ο χρήστης φορά το headset μικτής πραγματικότητας Microsoft HoloLens 2 και επιλέγει την εφαρμογή που αναπτύξαμε. Κατά την εκκίνηση, καθώς και σε τακτά χρονικά διαστήματα κατά την εκτέλεση της εφαρμογής, πραγματοποιείται χωρική χαρτογράφηση του περιβάλλοντα χώρου. Με αυτό τον τρόπο δημιουργείται ένα mesh, ένα 3D μοντέλο του χώρου.

Στη συνέχεια, καθώς ο χρήστης περπατά στο χώρο, προσπαθούμε να εντοπίσουμε, σε περιοχή γύρω από αυτόν, πιθανά εμπόδια. Σε περίπτωση που η συσκευή ανιχνεύσει κάποιο εμπόδιο, προειδοποιεί τον χρήστη με τρεις διαφορετικούς τρόπους. Αναπαράγει, αρχικά, μια σύντομη νηστική ειδοποίηση, η οποία διαθέτει ιδιότητες χωρικού ύχου. Έτσι, ο χρήστης μπορεί να αναγνωρίσει την «εικονική» θέση προέλευσης του ύχου, που αποτελεί και θέση του εμποδίου, καθώς και την απόστασή του από αυτήν. Παράλληλα, προβάλλονται οπτικά προειδοποιητικά σήματα στο πεδίο ορατότητας του ατόμου που υποδεικνύουν την θέση του εμποδίου. Τα σήματα αυτά μπορούν να αξιοποιηθούν από άτομα που αντιμετωπίζουν μερική απώλεια όρασης. Τέλος, προαιρετικά, προσφέρεται απτική ανάδραση προς το χρήστη, το οποίο του επιτρέπει να αντιληφθεί μόνο την θέση του εμποδίου στον οριζόντιο άξονα, αν θεωρήσουμε σύστημα αξόνων με αρχή τον χρήστη, δηλαδή πόσο αριστερά ή δεξιά του βρίσκεται.

Τέλος, αναπτύχθηκε ένας πρόσθετος τρόπος εντοπισμού εμποδίων. Ο εντοπισμός δε γίνεται σε συγκεκριμένη περιοχή γύρω από το χρήστη, όπως περιγράφηκε προηγουμένως. Αντιθέτως, ο ίδιος προτάσσει τα χεριά του προς την κατεύθυνση που επιθυμεί να ελέγξει.

Τέλος, πρέπει να τονιστεί ότι, λόγω της ιδιαιτερότητας των χρηστών, όλες οι λειτουργίες εντός της εφαρμογής πραγματοποιούνται με χρήση φωνητικών εντολών.

3.2 Σχεδιασμός και Περιορισμοί

Κατά τον σχεδιασμό της εφαρμογής, καθώς και κατά τη διάρκεια υλοποίησης αυτής, υπήρξαν πληθώρα περιπτώσεων, όπου κληθήκαμε να λάβουμε ιδιαίτερα σημαντικές αποφάσεις, που επηρέασαν σημαντικά την πορεία ανάπτυξης, λόγων των συνθηκών και προδιαφραφών που τέθηκαν ή των περιορισμών και δυσκολιών που αντιμετωπίσαμε.

Αρχικά, όσον αφορά την προετοιμασία και την εγκατάσταση των εργαλείων, ήρθαμε αντιμέτωποι με ένα ιδιαίτερα λειψό εγχειρίδιο (documentation) [101], το οποίο παρέχει η Microsoft. Ειδικότερα, το documentation είναι πλούσιο σε πληροφοριές, που βοηθούν προγραμματιστές, οι οποίοι έρχονται πρώτη φορά σε επαφή με το headset, να κατανοήσουν πλήρως έννοιες και τεχνολογίες που αξιοποοιούνται από τη συσκευή, καθώς και τον τρόπο λειτουργίας της. Ωστόσο, σε προγραμματιστικό επίπεδο, οι πληροφορίες είναι ελάχιστες, ενώ οι επεξηγήσεις αντικειμένων, κλάσεων και συναρτήσεων είναι ιδιαίτερα ελλιπείς, καθώς περιορίζονται σε επιδερμικές περιγραφές αυτών και την παρουσιασή παραδειγμάτων και σε ένα πολύ μικρό δείγμα αυτών. Παράλληλα, η Microsoft παρέχει μαθήματα (courses) για την πρακτική εκμάθηση των προγραμματιστών στην ανάπτυξη εφαρμογών για τη συσκευή με τη χρήση του περιβάλλοντος Unity. Όμως, και σε αυτή την περίπτωση, τα courses περιορίζονται στη χρήση έτοιμων projects, όπου η μόνη συμβολή του χρήστη στην ανάπτυξη αυτών είναι η αλλαγή ορισμένων απλών ρυθμίσεων. Επομένως, κατά τη διάρκεια της ανάπτυξης (development), αναγκαστήκαμε να καταφύγουμε σε οδηγούς (tutorials) και forums, τα οποία αναπτύχθηκαν από τρίτους, από κοινότητες προγραμματιστών που διέθεταν εμπειρία πάνω στη συγκεκριμένη τεχνολογία. Τέλος, ιδιαίτερο πρόβλημα αποτέλεσε η ασυμβατότητα (incompatibility) μεταξύ των διαφορετικών εκδόσεων των εργαλείων που χρησιμοποιήθηκαν (Unity, MRTK, Visual Studio), το οποίο καθυστέρησε την έναρξη της φάσης ανάπτυξης (development phase).

Επιπλέον, λόγω της ιδιαιτερότητας των χρηστών προς τους οποίους απευθύνεται η εφαρμογή, εκ πρώτης όψεως, φαίνεται αρκέτα οξύμωρη η επιλογή μίας συσκευής που ενσωματώνει την τεχνολογία Μικτής Πραγματικότητας, όπου, επί τον πλείστον, οι εφαρμογές απαιτούν από τους χρήστες να αλληλεπιδράσουν με εικονικά αντικείμενα και η όραση αποδεικνύεται εξαιρετικά σημαντική. Παρ’ όλα αυτά, η συσκευή επιλέχθηκε λόγω compact σχεδιασμού, ενσωματώνοντας πληθώρα αισθητήρων σε μια μικρή, φορητή συσκευή. Επίσης, ο προγραμματιστής έχει τη δυνατότητα να ενσωματώσει φωνητικές εντολές, όπου οι χρήστες μπορούν χρησιμοποιήσουν για να αλληλεπιδράσουν με τις λειτουργίες της εφαρμογής.

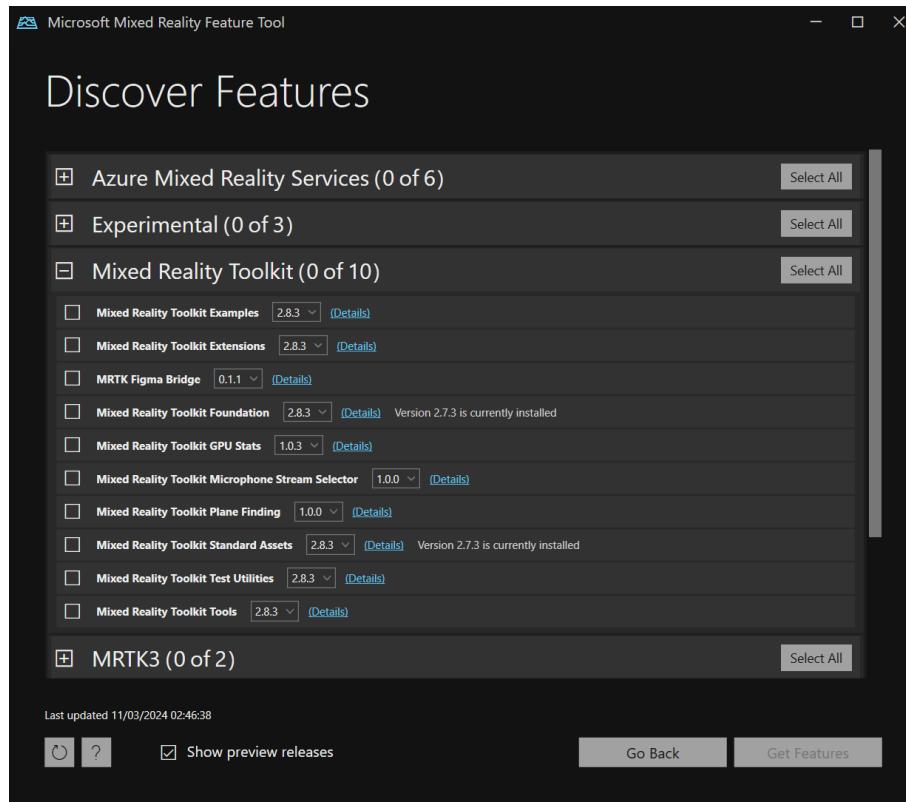
Πέρα από τις δυνατότητες που προσφέρει η συσκευή, καθιστώντας την ιδανική επιλογή, θέτει, παράλληλα, έχει και ορισμένους περιορισμούς. Συγκεκριμένα, οι περιορισμοί αφορούν τις συνθήκες του περιβάλλοντος, όπου χρησιμοποιείται η συσκευή [80]. Για την αποδοτική λειτουργία της συσκευής, πρέπει ο χώρος να είναι καλά φωτιζόμενος, να μην είναι ιδιαίτερα απλός, αλλά, αντιθέτως, να διαθέτει πολλά, μοναδικά αντικείμενα, τα οποία θα εξυπηρετήσουν στην καλύτερη αναγνώριση των αντικειμένων και των εμποδίων κατά τη χωρική χαρτογράφηση. Τέλος, οι συγχέεις αλλαγές και μετακινήσεις στο χώρο, καθώς και η ύπαρξη ανακλαστικών επιφανειών μπορούν να επηρεάσουν αρνητικά το tracking και τη χαρτογράφηση της συσκευής. Για τους ανωτέρω λόγους, αποφασίστηκε η χρήση της εφαρμογής να περιοριστεί σε εσωτερικούς χώρους.

3.3 Υλοποίηση

Στο παρόν κεφάλαιο, θα πραγματοποιηθεί μια αναλυτική περιγραφή όλων των σταδίων της φάσης ανάπτυξης (development phase) της εφαρμογής. Παράλληλα, θα εξηγήσουμε όλες τις λειτουργίες της εφαρμογής μαζί με τη λογική αυτών, παρουσίαζοντας τμήματα του κώδικα που αναπτύχθηκε. Ολόκληρος ο κώδικας της εργασίας βρίσκεται στο Παράρτημα A' μαζί με σχόλια σχετικά με τη λειτουργία κάθε μεθόδου.

3.3.1 Προετοιμασία και Εγκατάσταση

Αρχικά εγκαταστήσαμε όλα τα απαραίτητα εργαλεία, τα οποία συνιστά η Microsoft για την ανάπτυξη εφαρμογών για το headset HoloLens 2. Αφού η συνιστώμενη γλώσσα προγραμματισμού, την οποία και επιλέξαμε, είναι η C#, έπρεπε να εγκαταστήσουμε το IDE Visual Studio, τη μηχανή γραφικών Unity, καθώς και το εργαλείο Mixed Reality Feature Tool (Σχήμα 3.1) για την προσθήκη του MRTK στο project.



Σχήμα 3.1: To Mixed Reality Feature Tool

Όπως αναφέρθηκε και στο προηγούμενο κεφάλαιο (Κεφάλαιο 3.2), ένα σημαντικό πρόβλημα, που δημιούργησε το δύσχρηστο documentation, ήταν η έλλειψη καθοδήγησης όσον αφορά τη συμβατότητα των διάφορων εκδόσεων των προγραμμάτων. Για το λόγο αυτό, δεν επιλέχθηκαν οι πιο πρόσφατες εκδόσεις, αλλά εκδόσεις, οι οποίες χρησιμοποιούνταν κατά κόρον στους διάφορους διαθέσιμους οδηγούς. Ειδικότερα, επιλέχθηκε η Unity 2020.3.48f1 και εγκαταστήθηκε το Visual Studio Community 2019 μαζί με τα απαραίτητα components για την ανάπτυξη εφαρμογών μικτής πραγματικότητας, καθώς και την τελευταία έκδοση του Mixed Reality Feature Tool (version 1.0.2209.0). Επίσης, για τη διαχείριση των εκδόσεων του κώδικα, που αναπτύχαμε,

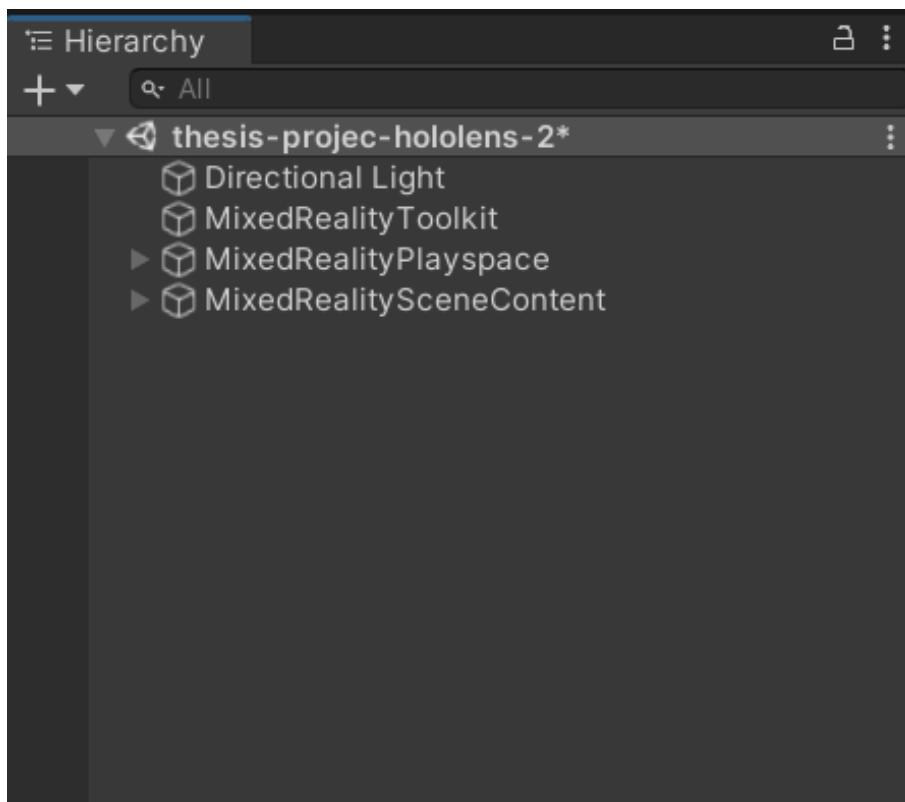
χρησιμοποιήθηκε το λογισμικό Git, ενώ, για την αποθήκευση αυτού, έγινε χρήση της πλατφόρμας GitHub.

Μετά την ολοκλήρωση της εγκατάστασης, πρώτο βήμα στην ανάπτυξη της εφαρμογής ήταν η δημιουργία ενός project στη Unity, το οποίο διαθέτει μια σκηνή με τα αρχικώς απαραίτητα **GameObjects**, όπως είναι η Camera, ενώ παρέχει πληθώρα από **Components** και Assets. Ωστόσο, αυτές οι «πρώτες ύλες» δεν επαρκούν για την ανάπτυξη εφαρμογών μικτής πραγματικότητας. Για το λόγο αυτό, θα αξιοποιήσουμε το Mixed Reality Feature Tool για την προσθήκη των αναγκαίων packages, όπως αυτά αναφέρονται στον Πίνακα 3.1, με σημαντικότερο από όλα το MRTK.

Πίνακας 3.1: Packages που ενσωματώθηκαν στο project με τη χρήση του Mixed Reality Feature Tool

Mixed Reality Toolkit Foundation	version 2.7.3
Mixed Reality Toolkit Standard Assets	version 2.7.3
Mixed Reality OpenXR Plugin	version 1.4.0
Microsoft Spatializer	version 2.0.37

Με την ενσωμάτωση του MRTK, προστίθενται στη σκηνή ορισμένα βασικά **GameObjects**, ενώ στο project συμπεριλαμβάνονται απαραίτητα components για την βελτίωση της συμπεριφοράς των εικονικών αντικειμένων και την καλύτερη αλληλεπίδραση του χρήστη με αυτά, assets για τη διαμόρφωση του UI (User Interface) της εφαρμογής, καθώς και λειτουργίες, οι οποίες θα διευκολύνουν σημαντικά την ανάπτυξη και δοκιμή της εφαρμογής μας [99].



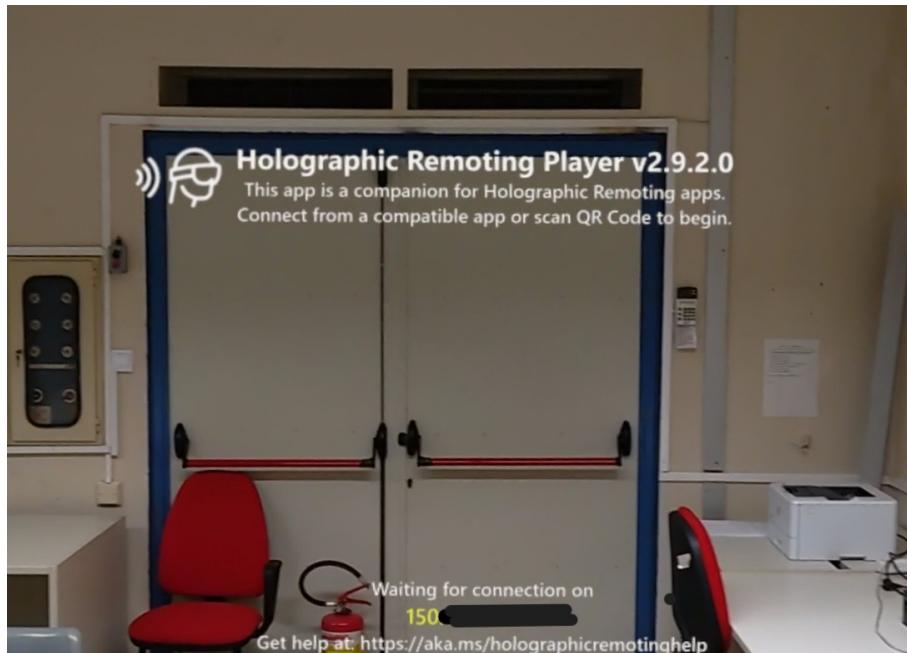
Σχήμα 3.2: Το hierarchy window μετά την προσθήκη του MRTK

Πλέον, η ιεραρχία των **GameObjects** στη σκηνή διαμορφώνεται με τον τρόπο που φαίνεται στο Σχήμα 3.2.

Τα νέα **GameObjects**, που προστέθηκαν, είναι:

- **MixedRealityToolkit**: Αποτελεί το κύριο **GameObject** της εργαλειοθήκης, καθώς διαθέτει όλες τις ρυθμίσεις της, όπως για το σύστημα εισόδου (Input System), την κάμερα, την χωρική χαρτογράφηση και άλλα.
- **MixedRealityPlaySpace**: Είναι το **GameObject** που αντιπροσωπεύει το headset στη σκηνή. Εντός αυτού του αντικειμένου, υπάρχει και το αντικείμενο της κάμερας.
- **MixedRealitySceneContent**: Αποτελεί τον χωρό των εικονικών αντικειμένων. Τα αντικείμενα τοποθετούνται εντός αυτού τους **GameObject**, με σκοπό να εξασφαλιστεί η σωστή κλίμακα αυτών, όταν προβάλλονται στον πραγματικό χώρο.

Επιπλέον, το MRTK μας δίνει τη δυνατότητα να κάνουμε προσομοίωση χρήσης της εφαρμογής μέσα από τον Editor της Unity, το οποίο ήταν καθοριστικό στα πρώτα στάδια ανάπτυξης της εφαρμογής για το debugging και τις δοκιμές αυτής. Τέλος, η λειτουργία Holographic Remoting του MRTK επιτρέπει στη Unity να συνδεθεί με το HoloLens 2 μέσω του Holographic Remoting Player (Σχήμα 3.3), ώστε το headset να «τρέξει» την εφαρμογή μας και να δοκιμαστεί υπό πραγματικές συνθήκες [102]. Η λειτουργία αυτή αξιοποιήθηκε στα τελευταία στάδια ανάπτυξης, με σκοπό να ελεγχθεί ταχύτατα η λειτουργικότητα της εφαρμογής, αλλά και για να γίνουν οι απαραίτητες αλλαγές σε προβλήματα που δεν θα μπορούσαμε να εντοπίσουμε κατά την προσομοίωση.



Σχήμα 3.3: Το Holographic Remoting Player σε αναμονή σύνδεσης

3.3.2 Οργάνωση του Project

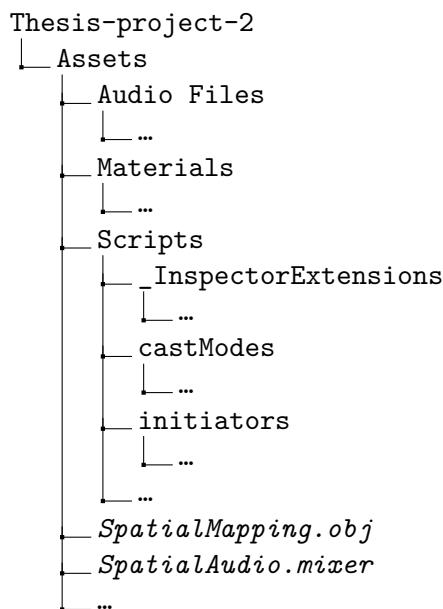
Σε αυτό το κεφάλαιο, θα πραγματοποιήσουμε μια μικρή περιήγηση στη σκηνή της εφαρμογής μας. Θα παρουσιάσουμε τα **GameObjects** που δημιουργήθηκαν κατά

την αναπτυξή, τον τροπό οργάνωσης αυτών και των **Components**, τα οποία συντάξαμε και θα περιγράψουμε συνοπτικά τη λειτουργία τους. Μια αναλύτική περιγραφή όλων των παραπάνω θα δοθεί στα ακόλουθα κεφάλαια.



Σχήμα 3.4: Η τελική μορφή του Hierarchy Window

Αρχικά, οι φάκελοι του project έχουν την ακόλουθη δομή:



Ειδικότερα, στο φάκελο **Audio Files**, τοποθετήθηκαν αρχεία ήχου σε μορφή **.mp3**, ενώ στο φάκελο **Materials**, τοποθετήθηκαν τα υλικά, τα οποία εφαρμόστηκαν στα **GameObjects** που δημιουργήσαμε και τους προσδίδουν υφή και χρώμα. Ο φάκελος **Scripts**, καθώς και οι υποφακέλοι αυτού, περιέχουν αρχεία κώδικα σε μορφή **.cs** και αποτελούν ουσιαστικά τα **Components** που αναπτύξαμε για τα **GameObjects** μας, αλλά και μερικά αρχεία για την καλύτερη απεικόνιση πληροφοριών στο Inspector window.

Τέλος, εντός του φακέλου **Assets** υπάρχει το αρχείο **SpatialMapping**, το οποίο είναι ένα 3D μοντέλο κλειστού χώρου, που χρησιμοποιήθηκε στις προσομοιώσεις και ένα mixer για τον έλεγχο του ήχου.

Σχετικά με τα **GameObjects** και τη θέση αυτών (Σχήμα 3.4), παρατηρούμε ότι ορισμένα από αυτά είναι ‘εμφωλευμένα’, εντός των **GameObjects MixedRealityPlayspace** και **MixedRealitySceneContent**. Αυτά αποτελούν τα εικονικά αντικείμενα, τα οποία θα τοποθετούνται και απεικονίζονται στο πραγματικό χώρο. Το ίδιο συμβαίνει και με τα **GameObjects**, τα οποία είναι ‘children’ (‘παιδιά’) της **Main Camera** με μόνη διαφορά ότι η θέση τους μεταβάλλεται σύνεχως σύμφωνα με την κίνηση του headset, δηλαδή την κίνηση του κεφαλιού του χρήστη. Τα **GameObjects**, που δεν βρίσκονται εντός των ειδικά διαμορφωμένων **GameObjects** από το MRTK, επιτελούν βοηθητικό ρόλο, καθώς ενσωματώνουν scripts που βοηθούν στη διαχείριση των λειτουργιών της εφαρμογής και των λοιπών **GameObjects** στη σκηνή.

Όσον αφορά την ονοματοδοσία των **GameObjects**, αυτή καθορίζεται από το σκοπό τους. Συγκειμένα, αντικείμενα, τα οποία χρησιμοποιούνται για την αναπαραγωγή προειδοποιητικών ήχων, περιέχουν τα γράμματα ‘AB’ στο τέλος του ονόματος, ενώ στην αρχή αυτού αναφέρεται η θέση των αντικειμένων αυτών κατά την έναρξη εκτέλεσης της εφαρμογής, σε CamelCase μορφή. Κάτι αντίστοιχο ισχύει και στην περίπτωση των **GameObjects**, τα οποία ενσωματώνουν components για τον εντοπισμό εμποδίων στη διαδρομή του χρήστη. Για την διαφοροποίηση των αντικειμένων αυτών, η κατάλληλη της ονομασίας τους είναι ‘RC’. Αντικείμενα, τα οποία επιτελούν βοηθητικό ρόλο, όπως αναφέρθηκε και προηγουμένως, διαθέτουν την λέξη ‘Handler’ στην ονομασία τους (από εδώ και στο εξής τα αντικείμενα αυτά μαζί με τα components τους θα αποκαλούνται ‘handlers’).

Τέλος,

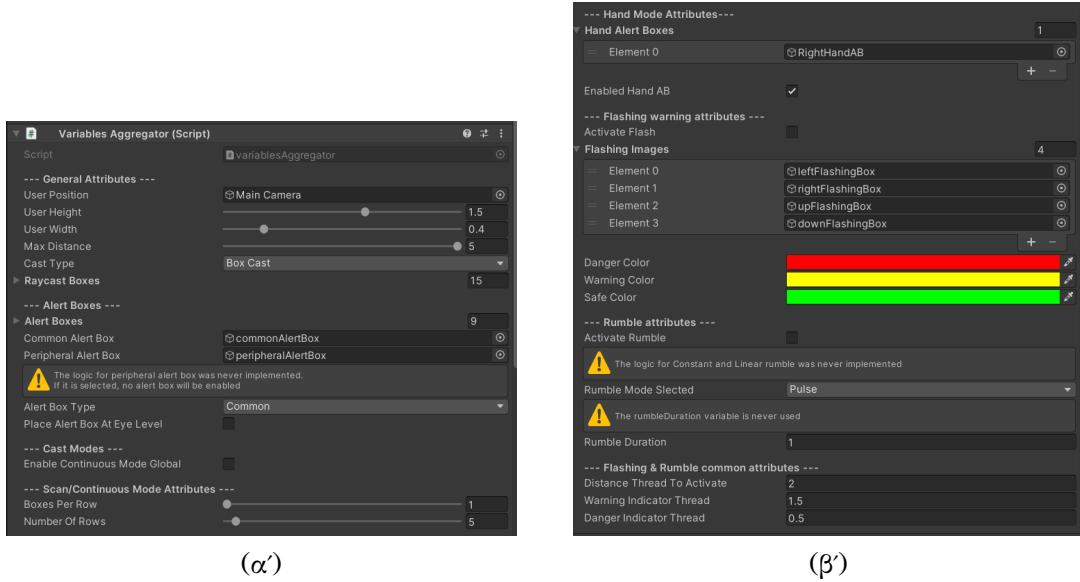
ει-

δική αναφορά πρέπει να πραγματοποιηθεί στο **GameObject variableAggObject**, στο όποιο έχει προστεθεί το component **variablesAggregator.cs** (*Assets/Scripts*). Σκοπός του αρχείου είναι να συγκεντρώνει την πλειοψηφία των κοινών μεταβλητών που χρειάζονται περισσότερα του ενός scripts, ορίζοντας αυτές ως **public**, δηλαδή να είναι προσβάσιμες από άλλες κλάσεις πέρα αυτής στην οποία ορίστηκαν. Με τον τρόπο αυτό, είναι ευκολότερο για εμάς να διαχειριστούμε ένα μεγάλο αριθμό μεταβλητών, τις οποίες συχνά καλούμαστε να αλλάξουμε πριν ή κατά την εκτέλεση της εφαρμογής για λόγους δοκιμής ή διορθώσεων.

3.3.3 Εντοπισμός Εμποδίων

Η σημαντικότερη λειτουργία της εφαρμογής αποτελεί ο εντοπισμός των εμποδίων στη διαδρομή ενός χρήστη. Ο χρήστης μπορεί να περιγγηθεί σε ένα κλειστό χώρο, γνωστό ή άγνωστο προς αυτόν. Η εφαρμογή οφείλει να καταγράψει το χώρο και να αναζητήσει προς την κατεύθυνση πορείας του χρήστη πιθανά εμπόδια. Σε περίπτωση που εντοπίσει κάποιο εμπόδιο, πρέπει να προειδοποιήσει το χρήστη με διάφορους τρόπους (Κεφάλαιο 3.3.4), ώστε αυτός, αναλογά με τη θέση και την απόσταση του εμποδίου, να επιλέξει αν επιθυμεί να αλλάξει πορεία κατεύθυνσης ή να προσπεράσει το εμπόδιο.

Για να επιτευχθεί αυτή η λειτουργία, αρχικά, θα πρέπει να καταγραφεί ο περιβάλλον χώρος του χρήστη. Για αυτό θα αξιοποιήσουμε την τεχνολογία της χωρικής χαρτογράφησης (spatial mapping) που προσφέρει το headset. Ειδικότερα, καθώς η συγκεκριμένη λειτουργία είναι ενεργοποιημένη, ο χρήστης περιστρέφει το κεφάλι του μαζί με το headset προς διάφορες κατεύθυνσεις. Με τον τρόπο αυτό συλλέγονται



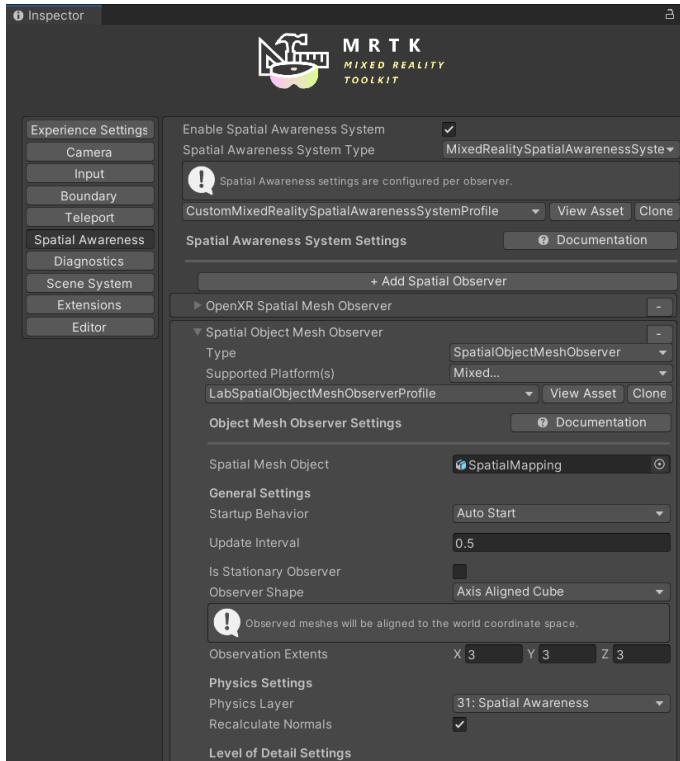
Σχήμα 3.5: Το Inspector window για το variableAggObject

πλήθος σημείων, δημιουργώντας ένα νέφος (point cloud) και με βάση τα σημεία αυτά δημιουργείται ένα σύνολο τριγώνων, που αποτελούν το mesh. Διάφοροι παράμετροι σχετικά με τον τρόπο λειτουργίας της χαρτογράφησης μπορούν να τροποποιηθούν.

Η ενεργοποίηση της χωρικής χαρτογράφησης γίνεται μέσα από τις ρυθμίσεις που διαθέτει το **GameObject MixedRealityToolkit**, στην καρτέλα ‘Spatial Awareness’ (Σχήμα 3.6). Αφού ενεργοποιήσουμε τη λειτουργία και δημιουργήσουμε τα custom profiles μας για την αποθήκευση των τροποποιήσεών μας, πρέπει να δημιουργήσουμε ένα ‘Spatial Surface Observer’, το οποίο αποτελεί ουσιαστικά το αντικείμενο που διαχειρίζεται τον τρόπο δημιουργίας και απεικόνισης των meshes [78]. Στην δική μας περίπτωση, χρησιμοποιήσαμε δύο Observers, ένας ο οποίος χρησιμοποιήθηκε κατά την προσομοίωση της εφαρμογής (Spatial Object Mesh Observer [103]) και ένας ο οποίος χρησιμοποιήθηκε κατά τις δοκιμές της εφαρμογής με χρήση του headset (OpenXR Spatial Mesh Observer). Ωστόσο η πλειονότητα των αλλαγών, που πραγματοποιήθηκαν στο setup αυτών, ήταν ίδιες και στις δύο περιπτώσεις.

Θα περιγράψουμε περιληπτικά τις κύριες παραμέτρους που αλλάξαμε, καθώς και τις τιμές που θέσαμε [104]:

- **Startup Behavior:** Καθορίζει αν η καταγραφή του χώρου θα ξεκινήσει αμέσως μετά την αρχικοποίηση της λειτουργίας ή αν θα καθοριστεί από τον προγραμματιστή. Θέσαμε την τιμή **Auto Start**, ώστε η καταγραφή να αρχίσει με την έναρξη χρήσης της εφαρμογής.
- **Update Interval:** Είναι ο ρυθμός ανανέωσης σε δευτερόλεπτα των δεδομένων του mesh. Για την προσομοίωση, επιλέξαμε το mesh να ανανεώνεται κάθε **0.5** δευτερόλεπτα, ενώ στις πραγματικές δοκιμές κάθε **0.1**, αφού οι αλλαγές στο χώρο είναι πιο συχνές.
- **Is Stationary Observer:** Καθορίζει αν ο Observer, που πραγματοποιεί την καταγραφή, θα βρίσκεται σταθερά στο χώρο ή θα κινείται μαζί με τον χρήστη. Επιλέξαμε την τιμή **False**, καθώς ο χρήστης μπορεί να κινείται σε πολλά δωμάτια, τα οποία δεν έχουν καταγραφεί.
- **Observation Extents:** Ορίζουμε τις τιμές **x**, **y**, **z**, οι οποίες είναι οι διαστάσεις του



Σχήμα 3.6: Οι ρυθμίσεις της λειτουργίας ‘Spatial Awareness’ και των ‘Spatial Surface Observers’

όγκου που αποτελεί την περιοχή καταγραφής. Για την προσομοίωση, θέσαμε την τιμή 5 για όλες τις διαστάσεις, ενώ για τις πραγματικές δοκιμές την τιμή 8, ώστε να διαθέτουμε περισσότερα δεδομένα του χώρου εκ των προτέρων, δηλαδή πριν προλάβει ο χρήστης να φθάσει στο σημείο.

- **Level of Detail:** Αποτελεί το βαθμό λεπτομέρειας του mesh, δηλαδή την πυκνότητα των σημείων και των τριγώνων. Για την προσομοίωση, θέσαμε την τιμή **Coarse**, ενώ για τις πραγματικές δοκιμές την τιμή **Unlimited**, ώστε να έχουμε όσο το δυνατόν πιο λεπτομέρες πλέγμα.
- **Display Settings:** Η κατηγορία αυτή διαθέτει τρεις μεταβλητές (**Display Option**, **Visible Material**, **Occlusion Material**), οι οποίες καθορίζουν τον τρόπο απεικόνισης του mesh. Και στις δύο περιπτώσεις, το mesh ήταν εμφανές προς τον χρήστη.
- **Runtime Spatial Mesh Prefab:** Πρόκειται για ένα prefab, το οποίο θα χρησιμοποιηθεί ως mesh κατά την εκτέλεση της εφαρμογής. Το prefab αυτό ορίστικε μόνο στην περίπτωση της προσομοίωσης, όπου θέσαμε το αρχείο **SpatialMapping.obj** (**Assets**), που αποτελεί ένα 3D μοντέλο του χώρου του εργαστηρίου (Σχήμα 3.7), το οποίο καταγράφηκε από το headset HoloLens 2.

Τέλος, δημιουργήθηκε ο handler **meshHandler**, στον οποίο προσθέσαμε το Component **hideMesh.cs** (**Assets/Scripts**), όπου μπορούμε με χρήση φωνητικών εντολών να κρύψουμε ή να εμφανίσουμε το mesh.

Αφού ολοκληρώσαμε τη διαδικασία χωρικής χαρτογράφησης του χώρου (Σχήμα 3.8), πρέπει να υλοποιήσουμε τη λογική για την ανίχνευση των εμποδίων με βάση το mesh που διαθέτουμε. Θα αξιοποιήσουμε την κλάση **Physics** του Physics



(β') Εσωτερική απεικόνιση του κύριου χώρου του εργαστηρίου

Σχήμα 3.7: Απεικόνιση του εργαστηρίου με 3D μοντέλο

module της Unity, το οποίο χρησιμοποιείται για την εφαρμογή φυσικής σε 3D αντικείμενα. Η κλάση αυτή διαθέτει ποικίλες μεθόδους, οι οποίες μας επιτρέπουν να ανιχνεύσουμε τη σύγχρονη δύο αντικειμένων [105]. Συγκεκριμένα, θα προσπαθήσουμε να εντοπίσουμε το σημείο τομής μιας ακτίνας (ray), που θα αποστέλλουμε από τη θέση του χρήστη, με κάποιο σημείο του mesh.

Οι μέθοδοι αυτοί είναι τα `RayCast`, `BoxCast`, `CapsuleCast`, `SphereCast`, των οποίων η διαφορά τους έγκειται στο σχήμα της ακτίνας, η οποία «εκπέμπεται». Στα πλαίσια της εφαρμογής μας, χρησιμοποιήθηκαν δύο μέθοδοι, το `RayCast`, που χρησιμοποιήθηκε στην αρχή του development, και το `BoxCast`, που χρησιμοποιείται στην τελική έκδοση της εφαρμογής. Η δεύτερη μέθοδος επιλέχθηκε λόγω του σχήματος της ακτίνας και του μεγέθους, το οποίο μπορούμε να το ορίσουμε έτσι ώστε να καλύπτει μια ικανοποιητική επιφάνεια μπροστά από το χρήστη. Η μέθοδος δέχεται ως παραμέτρους την αρχική θέση από όπου θα εκπεμψεί η ακτίνα, την κλίμακα αυτής, καθώς έχει το σχήμα κύβου, την περιστροφή της, την κατεύθυνση εκπομπής και τη μέγιστη απόσταση μέχρι την οποία θα γίνεται έλεγχος επαφής της ακτίνας με κάποια επιφάνεια που διαθέτει collider. Η κλήση της μεθόδου `RayCast` χρειάζεται τις ίδιες παραμέτρους, με μόνη εξαίρεση ότι δεν είναι απαραίτητη η κλίμακα και η περιστροφή της ακτίνας, αφού αυτή είναι δισδιάστατη. Τέλος, η μέθοδος επιστρέφει ένα flag σχετικά με το



Σχήμα 3.8: Χωρική χαρτογράφηση του χώρου του εργαστηρίου και απεικόνιση του mesh

αν υπήρξε επαφή με αντικείμενο και πληροφορίες σχετικά με αυτή τη σύγκρουση. Στην περίπτωση που υπήρξε επαφή με περισσότερα του ενός αντικείμενα, τότε επιστρέφονται πληροφορίες σχετικά με την πρώτη επαφή, καθώς αυτή θα είναι και η πλησιέστερη από το σημείο έναρξης (Origin Point).

Όπως αντιλαμβανόμαστε από τις παραμέτρους που απαιτούνται από τις δύο μεθόδους, πρέπει να ορίσουμε ένα σημείο έναρξης, από όπου θα ξεκινήσει η εκπομπή των ακτινών. Ως σημείο έναρξης ορίζουμε τη θέση του χρήστη, η οποία πρέπει να αλλάζει όταν αυτός θα μετακινείται. Η κατεύθυνση, που επιλέχθηκε, είναι αυτή προς την οποία κοιτά ο χρήστης, καθώς, συνήθως, αυτή είναι και η κατεύθυνση προς την οποία κινείται. Δεδομένου των ανωτέρω, τοποθετήσαμε ορισμένα **GameObjects** ως children του **GameObject** Main Camera, διότι, σε αυτή την περίπτωση, μαζί με τη θέση και την περιστροφή της κάμερας, που ακολουθεί την κίνηση του κεφαλιού, ανανεώνονται και η θέση και η περιστροφή των **GameObjects**. Η ονομασία των αντικειμένων αυτών διαθέτει την κατάληξη 'RC' και όλα τους ενσωματώνουν δύο components, το *initSingleBox.cs* (*Assets/Scripts/initiators*) και το *singleRaycast.cs* (*Assets/Scripts*). Το πλήθος των origin points, επομένως και των ακτινών, το μέγεθος και η διάταξη αυτών μπορούν να παραμετροποιηθούν από τις public μεταβλητές του **variableAggObject** (Κώδικας 3.1), ενώ η διαδικασία τοποθέτησης και διαμόρφωσης των διαστάσεων πραγματοποιείται από τη συνάρτηση **positionAndScaleBoxes** (Κώδικας 3.2), η οποία καλείται κατά την εκτέλεση της εφαρμογής και ανήκει στο component **Init Boxes** (*Assets/Scripts/initiators*), το οποίο και ενσωματώνεται στο **GameObject RaycastAndAlertBoxesHandler**. Έπειτα από δοκιμές, επιλέχθηκε η χρήση πέντε σημείων έναρξης εκπομπής ακτινών, τοποθετημένα κάθετα, με τέτοιο τρόπο και διαστάσεις, όπου το πρώτο origin point βρίσκεται στο ύψος των ματιών του χρήστη και το τελευταίο σε ύψος λίγο χαμηλότερο του γονάτου.

```

1  public GameObject voiceCommandSound;
2  // Variables for debugging purposes
3  [Header("---- General Attributes ----")]
4  public GameObject userPosition;
5  [Tooltip("User's height in meters (m)"), Range(0.0f, 2.5f)]
6  public float userHeight = 0.0f;

```

```

7     [Tooltip("User's width in meters (m)"), Range(0.0f, 2.5f)]
8     [Tooltip("The list of the hand alert boxes")]
9     public GameObject[] handAlertBoxes;
10    public enum lightPositionEnum
11    {
12        Up,

```

Κώδικας 3.1: Μεταβλητές για τη διάταξη των origin points και το μέγεθος των ακτινών

```

1
2     /// <summary>
3     /// A method which places the castBoxes and their alert boxes in the
4     /// correct position based on the parameters provided.
5     /// </summary>
6     /// <param name="raycstBoxesToPosition">An array of the castboxes</
7     /// param>
8     /// <param name="boxesPerRow">The number of castBoxes which will be
9     /// placed in each row</param>
10    /// <param name="numberofRows">The number of rows, in which the cast
11    /// boxes will be placed</param>
12    /// <param name="userHeight">The height of the user in meters</param>
13    public void positionAndScaleBoxes
14    (
15        GameObject[] raycstBoxesToPosition,
16        int boxesPerRow,
17        int numberofRows,
18        float userHeight
19    ) {
20        float boxLength = userWidth / boxesPerRow;
21        float boxHeight = userHeight / numberofRows;
22
23        float boxPlacementX = -boxLength * (boxesPerRow - 1);
24        float boxPlacementY = 0.0f;
25        for (int i = 0; i < raycstBoxesToPosition.Length; i++) {
26            if (i % boxesPerRow == 0) {
27                boxPlacementX = -boxLength * (boxesPerRow - 1);
28
29                if (i != 0) boxPlacementY -= boxHeight;
30
31                raycstBoxesToPosition[i].transform.position = new Vector3(
32                    boxPlacementX, boxPlacementY, -0.1f);
33                raycstBoxesToPosition[i].transform.localScale = new Vector3(
34                    boxLength, boxHeight, boxLength);
35
36                raycstBoxesToPosition[i].GetComponent<initSingleBox>().alertBox
37                    .transform.localScale = new Vector3(boxLength, boxHeight,
0.1f);
38
39                raycstBoxesToPosition[i].SetActive(true);
40
41                boxPlacementX += boxLength * (boxesPerRow - 1);
42            }
43        }
44    }

```

Κώδικας 3.2: Η συνάρτηση **positionAndScaleBoxes** για την τοποθέτηση και αλλαγή διαστάσεων αντικειμένων

Σχετικά με τα components, το **Init Single Box** διαθέτει ορισμένες public μεταβλητές που αφορούν χάθε αντικείμενο και χρησιμοποιούνται αποκλειστικά από άλλα

components. Αντιθέτως, το script **Single Raycast** περιέχει τη συνάρτηση, η οποία ξεκινά την εκπομπή της ακτίνας και επιστρέφει τις πληροφορίες για το σημείο επαφής αυτής με το mesh (Κώδικας 3.3). Είναι δυνατό επίσης να επιλέξουμε αν θα γίνει χρήση της μεθόδου **RayCast** ή **BoxCast** θέτοντας την αντίστοιχη τιμή στη μεταβλητή **castType** του **variableAggObject**.

```

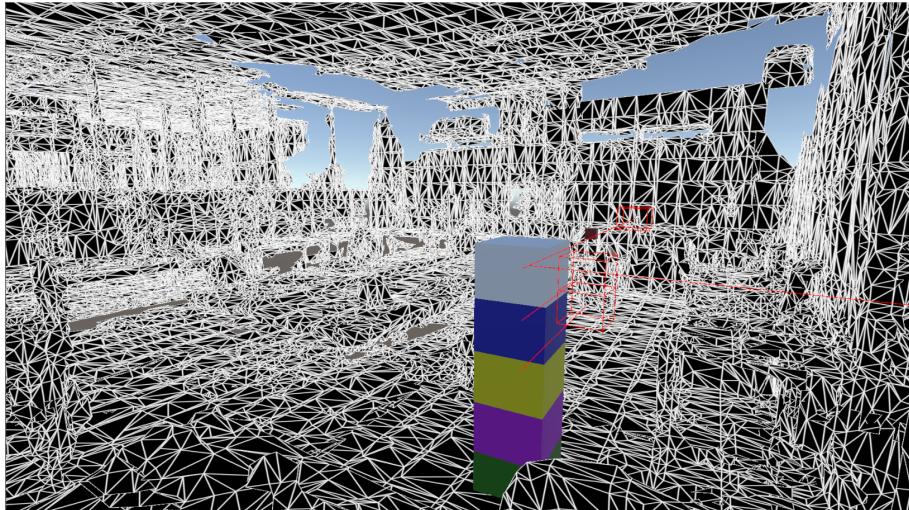
1      float maxDistance = variableAggInstance.maxDistance;
2
3      variablesAggregator.CastTypeEnum castType = variableAggInstance.
4          castType;
5
6      switch (castType) {
7          case variablesAggregator.CastTypeEnum.Raycast:
8              m_HitDetect = Physics.Raycast(transform.position, fwd, out
9                  hitInfo, 5.0f);
10             break;
11
12         case variablesAggregator.CastTypeEnum.BoxCast:
13             m_HitDetect = Physics.BoxCast(transform.position, transform
14                 .localScale, fwd, out hitInfo, transform.rotation,
15                 maxDistance);
16             break;
17     }
18 }
```

Κώδικας 3.3: Η συνάρτηση **SingleRaycastFunc** για τον εντοπισμό εμποδίων

Επομένως, από πέντε σημεία έναρξης, των οποίων η θέση και η περιστροφή επηρεάζεται από την κίνηση του κεφαλιού του χρήστη, εκπέμπονται ισάριθμες ακτίνες σε σχήμα κύβου (μέθοδος **BoxCast**). Αν οι ακτίνες αυτές έρθουν σε επαφή με κάποιο σημείο του πλέγματος, το οποίο ισοδυναμεί με την ύπαρξη εμποδίου προς την κατεύθυνση κίνησης του χρήστη, τότε καταγράφουμε τις πληροφορίες για το σημείο επαφής. Για παράδειγμα, στο Σχήμα 3.9, μπορούμε να παρατηρήσουμε πέντε κύβους, των οποίων τα κέντρα αποτελούν τα σημεία έναρξης. Οι ακτίνες εκπέμπονται από τα κέντρα και οι πορείες τους αναπαρίστανται με κόκκινες γραμμές, οι οποίες καταλήγουν σε έναν κύβο με κόκκινες ακμές σε περίπτωση που υπήρξε επαφή με το πλέγμα της χωρικής χαρτογράφησης.

Τα τελευταία ερωτήματα, τα οποία καλούμαστε να απαντήσουμε, με σκοπό να ολοκληρώσουμε τη λειτουργία εντοπισμού εμποδίων αφορούν τη συχνότητα κλήσης της συνάρτησης **SingleRaycastFunc** και τον τρόπο διαχείρισης πολλαπλών σημείων επαφής. Για τα ερωτήματα αυτά, αναπτύχθηκαν τρεις λύσεις, κάθε μία από τις οποίες έχει τα πλεονεκτήματα και τα μειονεκτήματά της. Κάθε λύση αναπτύχθηκε σε ξεχωριστό component και όλα προστέθηκαν στο **GameObject RaycastAndAlertBoxesHandler**

Η πρώτη, χρονικά, λύση, που αναπτύχθηκε, ονομάστηκε **Scan Mode**. Η εκπομπή των ακτινών ενεργοποιείται «χειροκίνητα», δηλαδή κάθε φορά που ο χρήστης χρησιμοποιεί μια συγκεκριμένη φωνητική εντολή. Τότε, από όλα τα origin points εκπέμπονται τα rays (με μέθοδο **BoxCast**). Συγκεντρώνονται σε ένα **List** όλα τα σημεία επαφής και επιλέγεται το πλησιέστερο (με χρήση της συνάρτησης **findClosestHitPointIndex** του component **Init Boxes**). Η λογική αυτή περιλαμβάνεται στο component **Scan Cast** (*Assets/Scripts/castModes*). Με αυτή τη λύση, ο χρήστης μπορεί να διαχειριστεί τη συ-

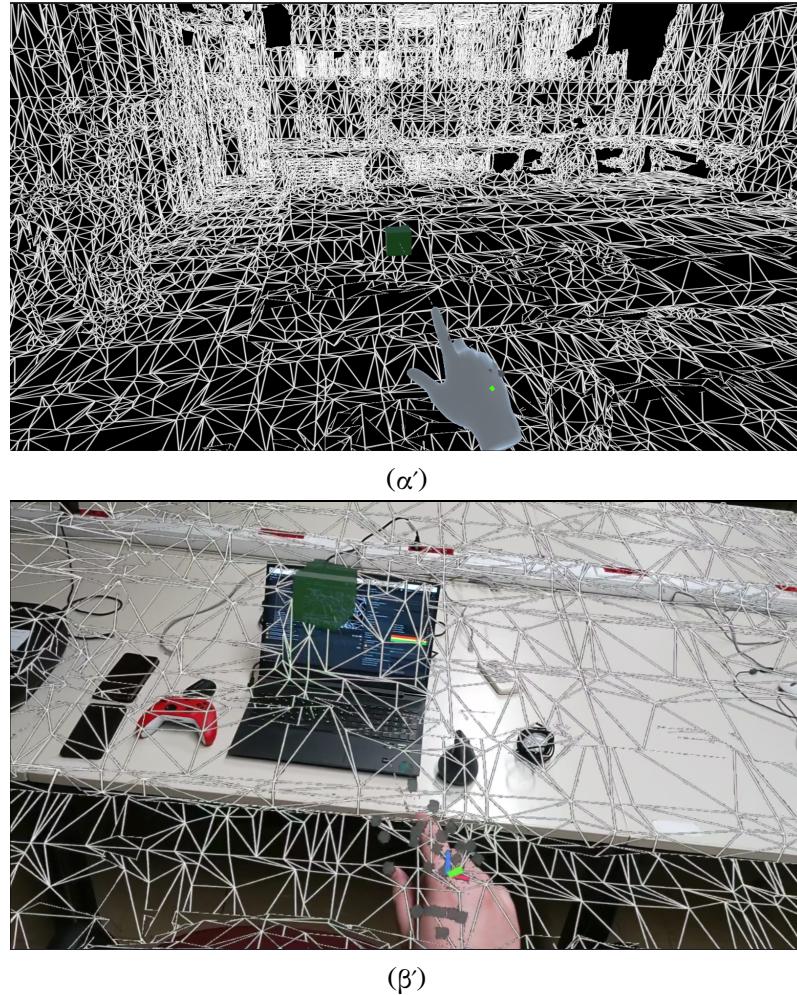


Σχήμα 3.9: Παράδειγμα εντοπισμού εμποδίων με τη μέθοδο **BoxCast**

χνότητα λήψης ειδοποιήσεων για εμπόδια και να τις λαμβάνει, οπότε αυτός κρίνει απαραίτητο. Ωστόσο, αυτός ο τρόπος ενεργοποίησης μπορεί να αποδειχθεί κουραστικός, ενώ, σε περίπτωση αλλαγής του περιβάλλοντα χώρου, μπορεί να είναι και εσφαλμένος.

Η δεύτερη λύση είναι η **Continuous Mode** και το component της είναι το **Continuous Cast** (*Assets/Scripts/castModes*). Η λογική σχετικά με την εκπομπή των rays και διαχείριση των πολλαπλών σημείων επαφής είναι με αυτή της προηγούμενης λύσης. Ωστόσο, στην περίπτωση αυτή, ο εντοπισμός εμποδίων πραγματοποιείται σε κάθε frame. Έτσι, ο χρήστης παραμένει πάντα ενήμερος για πιθανά εμπόδια στο χώρο του, ακόμη και αν πραγματοποιηθεί κάποια αλλαγή σε αυτόν.

Τέλος, η τρίτη λύση αποτελεί και την πιο ιδιάζουσα. Γλοποιείται εντός του component **Hand Raycast** (*Assets/Scripts/initiators*) και ονομάζεται **Hands Mode**, διότι το σημείο έναρξης των rays αποτελούν το χέρι του χρήστη. Ειδικότερα, θα εκμεταλλευτούμε το γεγονός ότι τα χέρια του χρήστη είναι Pointers [106], που του επιτρέπουν να αλληλεπιδρά με εικονικά αντικείμενα σε κοντινή ή μακρινή απόσταση. Στη περίπτωση της αλληλεπίδρασης από μακρινή απόσταση, ο τρόπος εντοπισμού των εικονικών αντικειμένων είναι παρόμοιος με αυτόν που περιγράφηκε στις προηγούμενες λύσεις. Με σημείο έναρξης το χέρι, εκπέμπεται ένα **RayCast**, το οποίο αναζητά σημείο επαφής με άλλα εικονικά αντικείμενα, ώστε να αλληλεπιδράσει με αυτά. Επομένως, στο αντικείμενο Pointer αποθηκεύονται πληροφορίες για το σημείο επαφής, όπως είναι η απόσταση από το χέρι και οι συντεταγμένες του στο χώρο. Επίσης, πρέπει να τονίσουμε ότι τα αντικείμενα Pointers δημιουργούνται και διατηρούνται κάθε φορά που τα χεριά του χρήστη βρίσκονται εντός του πεδίου ορατότητας (FOV) του headset. Με βάση τα ανωτέρω, κάθε φορά που τα χέρια του χρήστη βρίσκονται εντός του FOV, μπορούμε να αναζητήσουμε το σημείο επαφής των raycasts που εκπέμπονται από τα χεριά με το mesh που έχει δημιουργηθεί από τη χωρική χαρτογράφηση και, αν το σημείο αυτό βρίσκεται εντός μιας συγκεκριμένης απόστασης, να ειδοποιήσουμε το χρήστη για εμπόδιο (Σχήμα 3.10). Ουσιαστικά, τα raycasts λειτουργούν σαν ένα εικονικό μπαστούνι, το οποίο επιστρέφει feedback στο χειριστή του, όταν «έρθει σε επαφή» με εικονικά αντικείμενα, δηλαδή το mesh.



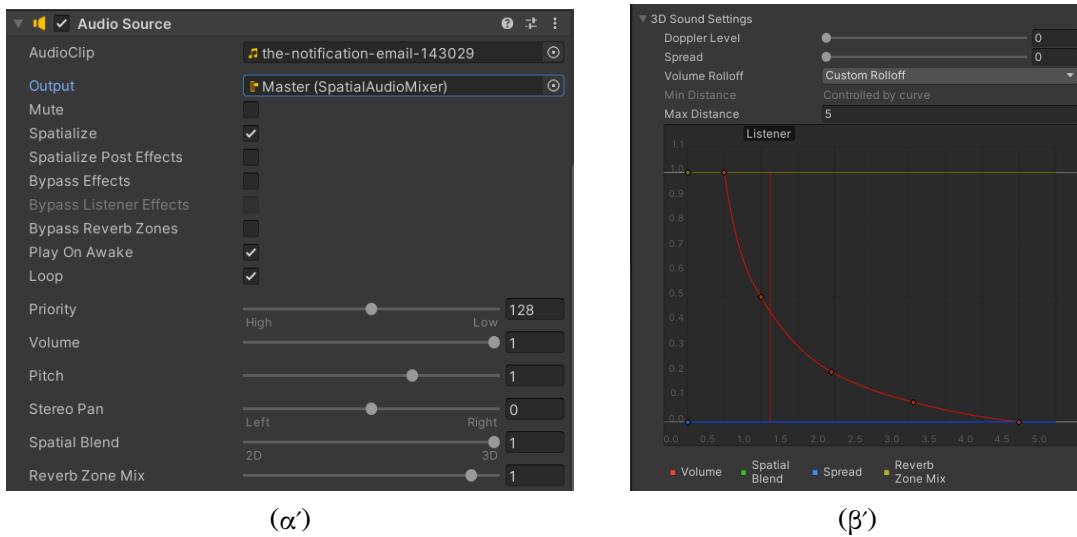
Σχήμα 3.10: Εντοπισμός εμποδίων με τη λειτουργία **Hands Mode**

3.3.4 Προειδοποίηση Χρήστη

Μετά τον εντοπισμό εμποδίων, εξίσου ουσιαστική είναι και η έγκαιρη προειδοποίηση του χρήστη για την ύπαρξη αυτών στη διαδρομή του. Σημαντικό ρόλο στην ανάπτυξη αυτής της λειτουργίας θα παίξει η τεχνολογία χωρικού ήχου (spatial sound), που διαθέτει το headset. Η συγκεκριμένη λειτουργία επιτρέπει τη δημιουργία ενός 3D ήχου, ο οποίος θα βοηθήσει το χρήστη να αντιληφθεί τη θέση του στο χώρο και ως προς αυτόν.

Αρχικά, θα πρέπει να τοποθετήσουμε την πηγή ήχου στο σημείο που βρίσκεται το εμπόδιο, έτσι ώστε να μπορέσει ο χρήστης να αντιληφθεί που βρίσκεται αυτό. Βασιζόμενοι στη λογική που αναπτύχθηκε και περιγράφηκε στο Κεφάλαιο 3.3.3, γνωρίζουμε ήδη τη θέση του πλησιέστερου προς το χρήστη σημείου επαφής με το πλέγμα σημείων, το οποίο αντιπροσωπεύει το εμπόδιο. Επομένως, τοποθετούμε στις συντεταγμένες του σημείου αυτού ένα **GameObject**, στο οποίο προσθέτουμε το component **Audio Source**. Με το συγκεκριμένο component, μπορούμε να προσθέσουμε ένα αρχείο, ώστε, από το σημείο όπου τοποθετήσαμε το **GameObject** και το οποίο ονομάσαμε **CommonAlertBox**, να αναπαράγεται ένας προειδοποιητικός ήχος.

Για να προσδώσουμε στον ήχο τις ιδιότητες ενός χωρικού ήχου, προσθέτουμε ένα **Audio Mixer**, στο οποίο έχουμε ενσωματώσει το Microsoft Spatializer Mixer effect. Επί-

Σχήμα 3.11: To component **Audio Source**

σης, ενεργοποιούμε στο **Audio Source** το spatialization, θέτοντας το flag **Spatialize** σε **True**, ενώ, παράλληλα, θέτουμε την τιμή **1** για τη μεταβλητή **Spatial Blend**. Ορισμένες ακόμη αλλαγές που πραγματοποιήσαμε στο component είναι να ενεργοποιήσουμε τις λειτουργίες **Play On Awake** και **Loop**, ώστε ο ήχος να ενεργοποιηθεί με την έναρξη της εφαρμογής και να μην σταματήσει η αναπαραγωγή μέχρι αυτό να υποδειχθεί από τον κώδικα, αντίστοιχα. Τέλος, διαμορφώσαμε την καμπύλη έντασης του ήχου ως προς την απόσταση του χρήστη από την πηγή, έτσι ώστε ο ήχος να γίνεται αισθητός σε απόσταση **5¹** από την πηγή, να αυξάνεται εκθετικά και να φθάνει το μέγιστο σε απόσταση **1**.

Όσον αφορά την τοποθέτηση της πηγής ήχου και τον έλεγχο αναπαραγωγής αυτού στις διάφορες λειτουργίες εντοπισμού εμποδίων, ισχύουν τα κατώθι:

- Στη λειτουργία **Continuous Mode**, ο εντοπισμός πλησιέστερου εμποδίου πραγματοποιείται σε κάθε ανανέωση του frame. Αν εντοπιστεί κάποιο εμπόδιο, τότε η πηγή τοποθετείται στις συντεταγμένες του σημείου και ξεκινά η αναπαραγωγή του προειδοποιητικού ήχου. Αν σε κάποιο update, βρέθει άλλο εμπόδιο, πλησιέστερο προς το χρήστη, τότε η πηγή του ήχου μετακινείται. Αν δεν εντοπιστεί κάποιο εμπόδιο, η αναπαραγωγή του ήχου σταματά.
- Στην περίπτωση του **Scan Mode**, η πηγή ήχου τοποθετείται στο πλησιέστερο εμπόδιο και παραμένει στο σημείο, ακόμη και αν ο χρήστης έχει μετακινηθεί ή αλλάξει κατεύθυνση. Η πηγή ήχου μετακινείται μόνο αν ο χρήστης ενεργοποιήσει ξανά τη λειτουργία με φωνητική εντολή και εντοπιστεί κάποιο νέο εμπόδιο, ενώ, αν δεν εντοπιστεί, τότε ο ήχος απενεργοποιείται.
- Στη λειτουργία **Hands Mode**, η λογική μετακίνησης πηγής και αναπαραγωγής/παύσης του ήχου είναι ίδια με αυτή της λειτουργίας **Continuous Mode**. Επιπλέον, ο χρήστης έχει τη δυνατότητα, προσωρινά, να παύσει την αναπαραγωγή του ήχου για όσο διάστημα πραγματοποιεί το Pinch gesture. Τέλος, ο ήχος απενεργοποιείται και στην περίπτωση που τα χέρια βρεθούν εκτός του field-of-view του headset.

¹Θεωρούμε ότι η αναλογία απόστασης σε μονάδα του περιβάλλοντος Unity προς μέτρα στον πραγματικό κόσμο είναι κατά προσέγγιση 1:1

Αξίζει να αναφερθούμε στο γεγονός ότι υλοποιήθηκε και η ιδέα χρήσης πολλαπλών πηγών και, πιο συγκεκριμένα, μία πηγή για κάθε ray που θα χρησιμοποιούταν για τον εντοπισμό εμποδίων. Αυτή η τεχνική αξιοποιήθηκε στην περίπτωση του **Scan Mode**, ώστε ο χρήστης να είναι ενήμερος για περισσότερα του ενός εμποδίων και να περιορίσουμε τη συχνότητα χρήσης της φωνητικής εντολής για την ενεργοποίηση της λειτουργίας. Ωστόσο, κατά τις δοκιμές, η ταυτόχρονη ενεργοποίηση πολλαπλών πηγών ήχου δυσκόλεψε στη διάκριση της κάθε πηγής ήχου και επομένως της θέσης του εμποδίου, λόγω της ιδιαίτερα κοντινής απόστασης στην οποία τοποθετούνταν.

Τέλος, εκτός από την ηχητική προειδοποίηση του χρήστη, αναπτύχθηκαν δύο ακόμη μέθοδοι: μία οπτική και μία με απτική ανάδραση. Η οπτική προειδοποίηση απευθύνεται σε άτομα που δεν αντιμετωπίζουν πλήρη απώλεια όρασης και μπορούν να αντιληφθούν χρώματα και σχήματα. Σε αυτό τον τρόπο προειδοποίησης, τοποθετούμε τέσσερα σχήματα μπροστά στο οπτικό πεδίο του χρήστη, το καθένα σε μια διαφορετική σχέση: στο πάνω, δεξί, κάτω και αριστερό μέρος του πεδίου του. Όταν εντοπιστεί κάποιο εμπόδιο, ανάλογα με τη θέση του εμποδίου ως προς τη κατεύθυνση που κοιτά ο χρήστης, τα αντίστοιχα σχήματα αρχίζουν να αναβοσβήνουν. Το χρώμα των σχημάτων αλλάζει ανάλογα με την απόσταση του χρήστη από το εμπόδιο. Ξεκινώντας από το πράσινο και όσο πλησιάζει προς το εμπόδιο, το χρώμα αλλάζει σε κίτρινο και τέλος σε κόκκινο. Κάθε σχήμα αποτελεί ένα **GameObject**, το οποίο διαθέτει το component **Flashing Light (Assets/Scripts)**, που περιέχει τη λογική για το flashing effect και τη θέση του αντικειμένου στο πεδίο ορατότητας του χρήστη. Ο κώδικας για την επιλογή ενεργοποίησης των κατάλληλων αντικειμένων ανάλογα με την θέση των εμποδίων είναι στο component **Init Flash Handler (Assets/Scripts/initiators)** του **RaycastAndAlertBoxesHandler**.

Τέλος, αναπτύχθηκε και μέθοδος της απτικής ανάδρασης για την προειδοποίηση του χρήστη. Για τη μέθοδο αυτή, χρησιμοποιήθηκε η συσκευή Microsoft XBOX Series controller (Σχήμα 3.12). Πρόκειται για ένα χειριστήριο, το οποίο επιλέχθηκε λόγω της απλότητας της ασύρματης σύνδεσης αυτού με τον υπολογιστή, για την ανάπτυξη του κώδικα, και του headset, για την πρακτική χρήση αυτού. Επιπλέον, η Unity παρέχει έτοιμες βιβλιοθήκες με κλάσεις, οι οποίες μας επιτρέπουν να αναγνωρίσουμε την σύνδεση/αποσύνδεση του χειριστηρίου, το input του χρήστη μέσω αυτού, καθώς και να ελέγχουμε τμήματα αυτού. Η απτική ανάδραση επιτυγχάνεται με την ενεργοποίηση



Σχήμα 3.12: Το χειριστήριο Microsoft XBOX Series (Πηγή: microsoft.com)

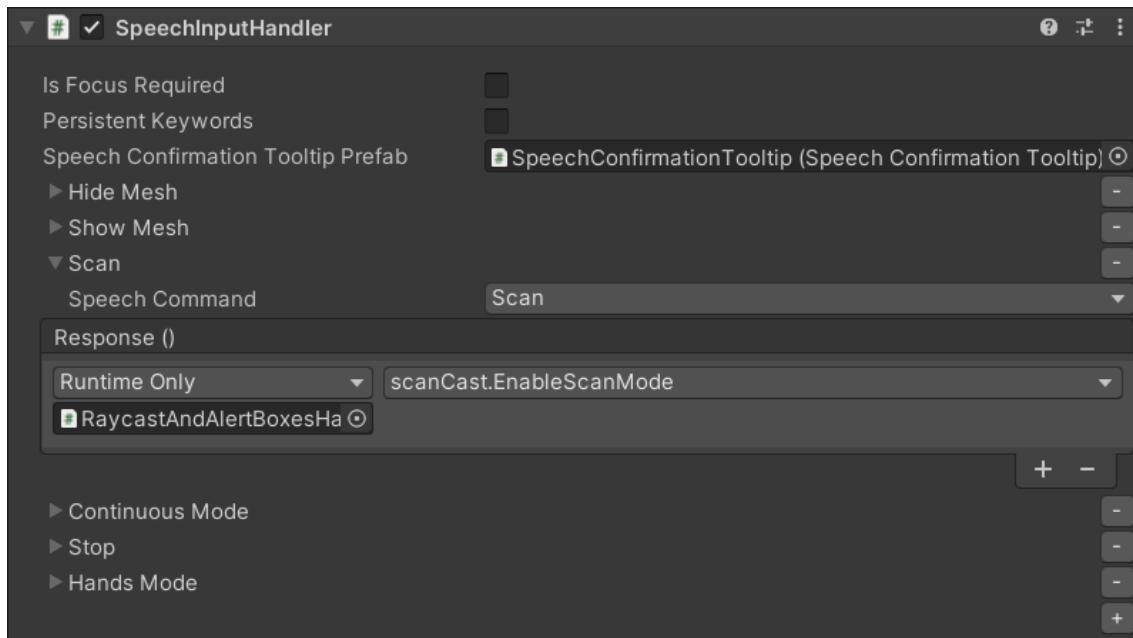
ενός εκ των δύο μοτέρ που διαθέτει το controller, ένα σε κάθε πλευρά αυτού. Ανάλογα με τη θέση του εμποδίου ως προς την κατεύθυνση που κοιτά ο χρήστης, ενεργοποιείται το αντίστοιχο μοτέρ παράγοντας μια παλμική δόνηση. Όσο πιο πολύ πλησιάζει ο

χρήστης στο εμπόδιο, τόσο η ένταση της δόνησης και η συχνότητα των παλμών αυξάνονται. Τέλος, ο χρήστης μπορεί να ενεργοποιήσει και να απενεργοποιήσει αυτόν τον τρόπο προειδοποίησης πιέζοντας τα κουμπιά ‘A’ και ‘B’ αντίστοιχα. Η διαχείριση του χειριστηρίου πραγματοποιείται από το component **Init Rumble Handler** (*Assets/Scripts/initiators*) του αντικειμένου **RaycastAndAlertBoxesHandler**.

3.3.5 Φωνητικές Εντολές

Αφού ολοκληρώσαμε την ανάπτυξη της λογικής για τον εντοπισμό εμποδίων και την προειδοποίηση του χρήστη, καλούμαστε τώρα να δώσουμε στο χρήστη τη δυνατότητα ελέγχου αυτών των λειτουργιών. Αυτό καθίσταται εφικτό με τη χρήση φωνητικών εντολών.

Η δυνατότητα χρήσης φωνητικών εντολών μπορεί να ενεργοποιηθεί από το **MixedRealityToolkit**, στην καρτέλα ‘Input’. Εκεί μπορούμε να προσθέσουμε τις λέξεις-κλειδιά, εκφρασμένες στην Αγγλική γλώσσα, οι οποίες θα αποτελέσουν τις εντολές της εφαρμογής. Στη συνέχεια δημιουργούμε το **GameObject SpeechInputHandler_Global**, στο οποίο προσθέτουμε το component **SpeechInputHandler** (Σχήμα 3.13), το οποίο προσφέρει το MRTK [107] για τη διαχείριση των φωνητικών εντολών. Συγκεκριμένα, όταν προσθέτουμε μια νέα φωνητική εντολή, επιλέγουμε την λέξη-κλειδί, η οποία θα την ενεργοποιεί, και ένα **GameObject**. Το **GameObject** διαθέτει components, από τα οποία επιλέγουμε μία συνάρτησή τους, η οποία θα εκτελείται, όταν θα χρησιμοποιείται η εντολή.



Σχήμα 3.13: Το component **SpeechInputHandler** και οι φωνητικές εντολές της εφαρμογής

Οι διαθέσιμες φωνητικές εντολές, καθώς και οι λειτουργίες τους, είναι οι εξής:

- **Hide Mesh:**
 - Συνάρτηση: `hideMeshFunc` (**Component:** `hideMesh.cs`)
 - Λειτουργία: Απόκρυψη απεικόνισης του mesh
- **Show Mesh:**

- Συνάρτηση: showMeshFunc (**Component:** *hideMesh.cs*)
 - Λειτουργία: Απεικόνιση του mesh
- Scan:
 - Συνάρτηση: EnableScanMode (**Component:** *scanCast.cs*)
 - Λειτουργία: Ενεργοποίηση του **Scan Mode** για τον εντοπισμό εμποδίων
- Continuous Mode:
 - Συνάρτηση: continuousModeGlobalToggle (**Component:** *continuousCast.cs*)
 - Λειτουργία: Ενεργοποίηση/απενεργοποίηση του Continuous Mode για τον εντοπισμό εμποδίων
- Stop:
 - Συνάρτηση: EnableStopMode (**Component:** *stopCast.cs*)
 - Λειτουργία: Απενεργοποίηση του **Continuous Mode** και παύση όλων των προειδοποιητικών ήχων
- Hands Mode:
 - Συνάρτηση: enabledHandABToggle (**Component:** *handRaycast.cs*)
 - Λειτουργία: Ενεργοποίηση/απενεργοποίηση της λειτουργίας **Hands Mode**

ΚΕΦΑΛΑΙΟ

4

ΑΞΙΟΛΟΓΗΣΗ

Μετά την ολοκλήρωση της υλοποίησης, ήρθε η ώρα να πραγματοποιηθεί η αξιολόγηση της εφαρμογής. Κατά την διαδικασία αυτή, χρήστες καλούνται να χρησιμοποιηθούν την εφαρμογή υπό πραγματικές ή/και ελεγχόμενες συνθήκες. Σκοπός της αξιολόγησης είναι να γίνει αντιληπτό αν η εφαρμογή μπορεί να ανταποκριθεί στις απαιτήσεις των πιθανών μελλοντικών χρηστών, καθώς και να αναδειχθούν προβλήματα και αδυναμίες της εφαρμογής, τα οποία χρίζουν αντιμετώπιση. Στα ακόλουθα κεφάλαια θα περιγράψουμε αναλυτικά τη διαδικασία που ακολουθήσαμε για την αξιολόγηση της εφαρμογής (Κεφάλαιο 4.1), καθώς και τα αποτελέσματα αυτής (Κεφάλαιο 4.2).

4.1 Διαδικασία αξιολόγησης

4.1.1 Περιγραφή Πειράματος

Το πείραμα διεξήχθη σε εσωτερικό κλειστό χώρο, σε αίθουσα του εργαστηρίου (Σχήμα 4.1). Κάθε εθέλοντης προσέρχοταν στο εργαστήριο και, προτού εισέλθει στην αίθουσα, λάμβανε ορισμένες σύντομες οδηγίες. Ειδικότερα, ενημερώναμε τους χρήστες για τα σκέλη του πειράματος και τον τρόπο διεξαγωγής τους. Επιπλέον, εξηγούσαμε τον τρόπο χρήσης της εφαρμογής, δηλαδή τις διαθέσιμες φωνητικές εντολές, καθώς τα διάφορα είδη ειδοποιήσεων, τα οποία θα λάμβαναν, αλλά τονίσαμε, επίσης, ορισμένα σφάλματα και λανθασμένες συμπεριφόρες της εφαρμογής, τις οποίες εντοπίσαμε κατά τις δοκιμές μας και γνωρίζαμε ότι θα συναντούσαν οι χρήστες, και τον τρόπο αντιμετώπισή τους. Στη συνέχεια, αφού καλύπταμε τα μάτια του εθελοντή με ένα πανί, εισέρχοταν στο χώρο διεξαγωγής του πειράματος. Με αυτό τον τρόπο εξασφλίζαμε ότι ο χρήστης δε γνωρίζει τη διάταξη του χώρου, καθώς και το πλήθος των εμποδίων, το είδος αυτών και τη θέση τους.



(α') Τμήμα 1

(β') Τμήμα 2

(γ') Τμήμα 3

(δ') Τμήμα 4

Σχήμα 4.1: Χώρος διεξαγωγής του πειράματος

Το πείραμα χωρίζεται σε δύο σκέλη, όπου, και στα δύο, ο εθελοντής καλείται να περιγγηθεί στην αίθουσα με όσο τον δυνατό μεγαλύτερη ασφάλεια (δηλαδή χωρίς να «συγκρουστεί» με κάποιο εμπόδιο) μέχρι να φθάσει στο τέλος της διαδρομής, το οποίο, λόγω της διάταξης του χώρου, ταυτίζεται με το σημείο έναρξης. Κατά το πρώτο σκέλος, ο χρήστης περιγγούταν στο χώρο χρησιμοποιώντας ένα μπαστούνι τυφλών (Σχήμα 4.2α'), η οποία αποτελεί και τη πιο συνηθισμένη λύση για άτομα με απώλεια όρασης. Αφού ολοκληρώσει τη διαδρομή, επαναλαμβάνει το πείραμα, αυτή τη φορά χρησιμοποιώντας την συσκευή HoloLens 2 και την εφαρμογή που αναπτύξαμε για αυτήν (Σχήμα 4.2β'). Μετά το πέρας και του δεύτερου σκέλους, ο εθελοντής συμπληρώνει ένα ερωτηματολογίο, απαντά, προφορικά ή γραπτώς, σε μερικές ερωτήσεις

ανοικτού τύπου και η διαδικασία ολοκληρώνεται.



(α') Χρήση του μπαστουνιού



(β') Χρήση εφαρμογής στο HoloLens 2

Σχήμα 4.2: Εθελοντές χρησιμοποιούν τους δύο διαφορετικούς τρόπους περιήγησης στο χώρο

4.1.2 Προδιαγραφές Αξιολόγησης

Το πείραμα έλαβε χώρο στην κύρια αίθουσα του εργαστηρίου Συστημάτων Υπολογιστών. Επιλέχθηκε ένας εσωτερικός κλειστός χώρος, λόγω των περιορισμών που θέτει το ίδιο το HoloLens 2 για την ορθή χαρτογράφηση του χώρου, η οποία εξαρτάται από τον φωτισμό και τις επιφάνειές του [78]. Επιπλέον, υπήρχε μεγαλύτερος έλεγχος στις συνθήκες διεξαγωγής του πειράματος, καθώς μπορούσαμε να επιλέξουμε τα εμπόδια και τη θέση τους στο χώρο, ενώ το ρίσκο για κάποιο αναπάντεχο γεγονός κατά την διεξαγωγή ήταν πολύ μικρότερο σε αντίθεση με το αν γινόταν σε έναν ανοιχτό, εξωτερικό χώρο. Πριν εισέλθουν οι εθελοντές στο χώρο, παρείχαμε οδηγίες για τον τρόπο χρήσης της εφαρμογής, ενώ υπήρξε και ιδιαίτερη μνεία σε δύο άλυτα προβλήματα της εφαρμογής:

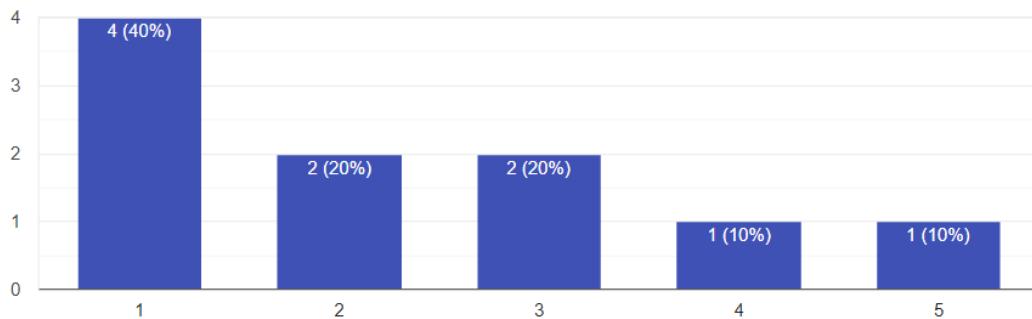
- Μίξη των προειδοποιητικών ήχων: κατά τη χρήση της εφαρμογής, λόγω της κίνησης του κεφαλιού του χρήστη όσο αυτός περπατά, οπότε και του headset, η εφαρμογή μπορεί να εντοπίσει κάποιο διαφορετικό σημείο του mesh ως πλησιέστερο προς το χρήστη. Ως αποτέλεσμα, κατά την κίνηση του χρήστη, είναι πιθανό να αναπαράγονται πολλοί, διαφορετικοί προειδοποιητικοί ήχοι σε πολύ σύντομο χρονικό διάστημα. Προτάθηκε στους χρήστες, αν αυτό συμβεί, να παραμείνουν σταθεροί στο σημείο εως ότου η συσκευή εντοπίσει το σωστό πλησιέστερο σημείο του mesh και να αναπαράξει το σωστό προειδοποιητικό ήχο.
- Δημιουργία ‘hallucinations’: είναι πιθανό, σε δυναμικά περιβάλλοντα, όπου άτομα ή αντικείμενα μετακινούνται, να δημιουργηθούν ‘hallucinations’, δηλαδή 3D αναπαραστάσεις αντικειμένων να τοποθετηθούν σε σημείο, στο οποίο δε βρίσκεται πλέον το αντικείμενο [78]. Προσπαθήσαμε να επιλύσουμε το θέμα αυτό, θέτοντας την τιμή για το Update Interval της χωρικής χαρτογράφησης στη μικρότερη δυνατή τιμή [104]. Ωστόσο, και σε αυτή την περίπτωση, απαιτούνται περίπου 3-4 δευτερόλεπτα, ώστε να αντικατροπτιστεί μια αλλαγή του περιβάλλοντος στο mesh. Για το λόγο αυτό, πριν ο εθελοντής χρησιμοποιήσει την εφαρμογή, πραγματοποιούμε μια αρχική χαρτογράφηση του χώρου, ώστε να αφαιρέσουμε πιθανά hallucinations, που έχουν παραμείνει.

Στο πείραμα συμμετείχαν δέκα άτομα (πέντε άντρες και πέντε γυναίκες), ηλικιών από 21 έως 47 ετών και με μέση ηλικία τα 25.7 έτη. Από τα άτομα αυτά, τέσσερις είχαν κάποια προηγούμενη εμπειρία με τεχνολογία εκτεταμένης πραγματικότητας και,

ειδικότερα, ένας είχε εμπειρία με την επαυξημένη πραγματικότητα (AR), ένας με την εικονική (VR) και δύο και με τα δύο είδη πραγματικότητας. Η πλειονότητα των εθελοντών ένιωθε ότι έχει ελάχιστη εξοικείωση με την τεχνολογία AR (Σχήμα 4.3).

Πόσο εξοικιωμένος/η θεωρείτε ότι είστε με την τεχνολογία επαυξημένης πραγματικότητας;

10 responses



Σχήμα 4.3: Απαντήσεις για το επίπεδο εξοικείωσης των εθελοντών με την επαυξημένη πραγματικότητα

Τέλος, έξι στους δέκα ανέφεραν ότι ήταν προηγουμένως εξοικιωμένοι με τη διάταξη του χώρου. Ωστόσο, καλύπτοντας τα μάτια των εθελοντών πριν εισέλθουν στο χώρο, εξασφαλίζαμε ότι κανένας εθελοντής δε γνώριζε τη θέση των εμποδίων σε αυτόν.

Για την εκτέλεση του πειράματος, αποφασίσαμε όλοι οι εθελοντές να πραγματοποιήσουν και τα δύο σκέλη/tasks αυτού, δηλαδή πραγματοποιήσαμε ένα πείραμα within-subjects [108]. Επιλέξαμε αυτό το τύπο πειράματος, λόγω του μικρού αριθμού εθελοντών που διαθέταμε και της πολυπλοκότητας των tasks [109]. Για να μειώσουμε το learning effect, όπου οι εθελοντές μαθαίνουν και είναι περισσότερο προετοιμασμένοι μετά την ολοκλήρωση του πρώτου task, αναθέσαμε στους μισούς εθελοντές να χρησιμοποιήσουν πρώτα το μπαστούνι τυφλών (πρώτο σκέλος) και έπειτα την εφαρμογή (δεύτερο σκέλος), ενώ οι υπόλοιποι εθελοντές πραγματοποίησαν τα σκέλη με την αντίστροφη σειρά.

Κατά την εκτέλεση και των δύο σκελών, καταγράφαμε το χρόνο που απαιτείται για να ολοκληρωθεί κάθε task, συνολικά και ανά τμήμα, καθώς είχαμε χωρίσει την περιήγηση σε τέσσερα τμήματα. Επίσης, σημειώναμε το πλήθος σφαλμάτων που έκαναν οι εθελοντές κατά την πραγματοποίηση ενός task. Ως σφάλμα σημειώσαμε κάθε σύγκρουση ενός εθελοντή με ένα εμπόδιο, καθώς και όταν κατευθυνόταν προς τη λάθος κατεύθυνση και όχι προς το τέλος της διαδρομής.

Τέλος, για την αξιολόγηση της εφαρμογής και της χρηστικότητάς της, χρησιμοποιήθηκε το ερωτηματολογίο System Usability Scale (SUS), το οποίο περιέχει δέκα προτάσεις [110]. Σε κάθε πρόταση, ο χρήστης επιλέγει έναν ακέραιο αριθμό από την κλίμακα 1 έως 5, ανάλογα με το πόσο συμφωνεί με όσα αναφέρει η πρόταση. Ο αριθμός 1 στην κλίμακα αντιστοιχεί στην απάντηση ‘Διαφωνώ Απόλυτα’, ενώ το 5 στην απάντηση ‘Συμφωνώ Απόλυτα’. Για να προκύψει η τελική βαθμολογία, η οποία είναι ένας αριθμός μεταξύ του 0 και του 100, εφαρμόζουμε τον ακόλουθο μαθηματικό

τύπο, όπου n είναι ο αριθμός της ερώτησης και x_n η τιμή της κλίμακας που επέλεξε ο χρήστης για την ερώτηση n :

$$\text{SUS Score} = 2.5 \times \left(\sum_{n=1}^5 (x_{2n-1} - 1) + \sum_{n=1}^5 (5 - x_{2n}) \right)$$

Στο τέλος του ερωτηματολογίου, προσθέσαμε τρεις ερωτήσεις ανοικτού τύπου, έτσι ώστε να καταγράψουμε καλύτερα την εμπειρία των εθελοντών με την εφαρμογή, δυσκολίες που αντιμετώπισαν, καθώς και τις προτάσεις τους για τη βελτίωσή της. Το ερωτηματολόγιο SUS και οι ανοικτού τύπου ερωτήσεις παρατίθενται στο Παράρτημα B.

4.2 Αποτελέσματα

4.2.1 Καταγραφή Χρόνων και Ανάλυση

Όπως αναφέραμε και στο τέλος του προηγούμενου κεφαλαίου, το πείραμα υλοποιήθηκε σε δύο σκέλη και για κάθε ένα από αυτά, καταγράφηκε ο χρόνος που απαιτούταν μέχρι οι εθελοντές να ολοκληρώσουν τη διαδρομή τους, καθώς και το πλήθος των σφαλμάτων που πραγματοποίησαν κατά τη διάρκεια αυτών. Οι χρόνοι αυτοί και τα σφάλματα έχουν αποτυπωθεί στον παρακάτω πίνακα (Πίνακας 4.1).

Πίνακας 4.1: Αποτελέσματα χρονομετρήσεων των δύο σκελών του πειράματος

Εθελοντές	Σκέλος	Τμήμα 1	Τμήμα 2	Τμήμα 3	Τμήμα 4	Συνολικός Χρόνος	Σφάλματα
Εθελοντής 1	(1) Μπαστούνι	1:51.80	1:11.81	1:30.48	1:20.17	5:54.26	2
	(2) Εφαρμογή	3:29.77	2:19.55	2:48.95	3:11.17	11:49.44	2
	—	+1:37.97	+1:07.74	+1:18.47	+1:51.00	+5:54.18	0
Εθελοντής 2	(2) Μπαστούνι	00:53.73	00:42.71	00:59.75	1:10.66	3:46.85	3
	(1) Εφαρμογή	1:17.59	1:02.52	00:40.52	00:58.57	3:59.20	2
	—	+00:23.86	+00:19.81	-00:19.23	-00:12.09	+00:12.62	-1
Εθελοντής 3	(1) Μπαστούνι	1:22.74	00:51.50	1:16.73	1:41.63	5:12.60	3
	(2) Εφαρμογή	00:56.14	00:47.72	1:07.44	1:19.26	4:10.56	2
	—	-00:26.60	-00:03.78	-00:09.29	-00:22.37	-1:02.04	-1
Εθελοντής 4	(2) Μπαστούνι	1:36.75	1:38.75	00:49.24	00:33.51	4:38.25	1
	(1) Εφαρμογή	3:12.21	1:43.52	2:31.27	00:50.25	8:19.60	9
	—	+1:35.46	+00:04.77	+1:41.98	+00:16.74	+3:41.35	+8
Εθελοντής 5	(1) Μπαστούνι	1:39.21	00:58.62	00:22.51	00:48.58	3:48.92	7
	(2) Εφαρμογή	1:45.98	1:42.39	2:56.42	1:46.22	8:11.01 (*)	9
	—	+00:06.77	+00:43.77	+2:33.91	+00:57.64	+4:22.19	+2
Εθελοντής 6	(2) Μπαστούνι	1:12.96	00:30.83	00:47.95	00:30.06	3:01.80	3
	(1) Εφαρμογή	00:58.70	1:19.82	1:22.17	00:46.21	4:26.90	5
	—	-00:14.26	+00:48.99	+00:34.22	+00:16.15	+1:25.10	+2
Εθελοντής 7	(1) Μπαστούνι	00:25.97	00:17.66	00:38.77	00:50.15	2:12.55	1
	(2) Εφαρμογή	00:47.70	00:24.86	1:18.71	00:37.80	3:09.07	4
	—	+00:21.73	+00:07.20	+00:39.44	-00:12.35	+00:56.53	+3
Εθελοντής 8	(2) Μπαστούνι	1:01.52	00:36.82	00:47.83	00:39.70	3:05.87	2
	(1) Εφαρμογή	1:53.22	1:17.36	1:38.00	00:42.01	5:30.59	4
	—	+00:51.30	+00:40.56	+00:50.17	+00:02.31	+2:24.72	+2
Εθελοντής 9	(1) Μπαστούνι	00:17.62	1:12.71	00:30.90	00:34.61	2:35.84 (*)	1
	(2) Εφαρμογή	00:40.85	00:47.85	1:07.61	1:33.24	4:09.55 (*)	3
	—	+00:23.23	-00:24.86	+00:36.71	+0:58.83	+1:33.71	+2
Εθελοντής 10	(2) Μπαστούνι	00:54.46	00:24.92	00:27.38	00:23.54	2:10.30	2
	(1) Εφαρμογή	1:54.47	00:51.47	1:18.27	00:28.25	4:32.46	3
	—	+1:00.01	+00:26.55	+00:50.89	+00:04.71	+2:22.16	+1

Για κάθε εθελοντή, σημειώνουμε το χρόνο, ανά τμήματα της διαδρομής, καθώς και τον συνολικό, μέχρι να ολοκληρώσει το task του σε κάθε σκέλος του πειράματος και το πλήθος των σφαλμάτων στα οποιά υπέπεσε. Στη στήλη **Σκέλος**, δίπλα από κάθε συσκευή, που κάθε εθελοντής χρησιμοποίησε για να περιηγηθεί στον χώρο, καταγράφεται η σείρα με την οποία υλοποίησε τα δύο σκέλη του πειράματος. Στη τρίτη γραμμή κάθε εθελοντή, σημειώνονται οι διαφορές των χρόνων/σφαλμάτων των δύο προηγούμενων γραμμών της ίδια στήλης. Με κόκκινο χρωματίζεται το κελί, όπου η χρήση της συσκεύης HoloLens 2 αποδείχθηκε πιο χρονοβόρα από τη χρήση του μπαστούνιού ή οδήγησε σε περισσότερα σφάλματα, ενώ με πράσινο χρώμα σημειώνεται η αντίθετη περίπτωση. Τέλος, παρατηρούμε ότι ορισμένες διαδρομές των εθελοντών 5 και 9 έχουν σημειωθεί με ένα αστερίσκο (*). Ο λόγος είναι ότι οι συγκεκριμένοι εθελοντές, κατά το ένα ή και τα δύο σκέλη, περιηγήθηκαν στο χώρο κατά την αντίθετη φορά από τους υπόλοιπους. Ωστόσο, οι χρόνοι ανά τμήμα καταγράφηκαν έτσι ώστε να αναφέρονται στο ίδιο τμήμα.

Μπορούμε να παρατηρήσουμε ότι, για την πλειονότητα των εθελοντών, η χρήση της συσκεύης HoloLens 2 οδήγησε σε αύξηση του χρόνου ολοκλήρωσης της προδιαγεγραμμένης διαδρομής. Εξαίρεση αποτελούν οι εθελοντές 2, 6, 7 και 9, οι οποίοι πραγματοποίησαν ταχύτερους χρόνους με τη χρήση της εφαρμογής σε τμήματα της διαδρομής, καθώς και ο εθελοντής 3, ο οποίος παρουσίασε βελτιωμένη επίδοση στο σύνολο αυτής. Επίσης, τρεις στους δέκα εθελοντές είχαν μικρότερο αριθμό σφαλμάτων με τη χρήση της εφαρμογής σε σχέση με τη χρήση του μπαστούνιού. Ωστόπο, πρέπει να αντιληφθούμε ότι η προσέγγιση του πειράματος από τους εθελοντές παρουσιάζει πολλές διακυμάνσεις. Ειδικότερα, το εύρος του χρόνου ολοκλήρωσης της διαδρομής με το μπαστούνιο είναι από 2:10.30 έως 5:54.26, ενώ με την εφαρμογή ξεκινά από 3:09.07 έως και 11:49.44. Επομένως, αρχικά, υπολογίσαμε τους μέσους χρόνους ολοκλήρωσης του task.

Πίνακας 4.2: Μέσος χρόνος ολοκλήρωσης τμημάτων του πειράματος και μέσο πλήθος σφαλμάτων ανά σκέλος

Μέσος Χρόνος	Τμήμα 1	Τμήμα 2	Τμήμα 3	Τμήμα 4	Συνολικός Χρόνος	Σφάλματα
Μπαστούνι	01:07.68	00:50.63	00:49.15	00:51.26	03:38.72	2.5
Εφαρμογή	01:50.29	01:13.71	01:40.94	01:13.30	05:49.84	4.3
—	-00:42.61	-00:23.08	-00:51.81	-00:22.04	-02:11.12	-1.8

Πίνακας 4.3: Μέση ποσοστιαία διαφορά χρόνων και ποσοστιαία διαφορά μέσων χρόνων ολοκλήρωσης τμημάτων του πειράματος

	Τμήμα 1	Τμήμα 2	Τμήμα 3	Τμήμα 4	Συνολικός Χρόνος
Μέση Ποσοστιαία Διαφορά Χρόνων	+68.25%	+59.5%	+151.73%	+49.22%	+61.64%
Ποσοστιαία Διαφορά Μέσων Χρόνων	+62.96%	+45.56%	+105.34%	+42.98%	+59.94%

Με τη χρήση του μπαστούνιού, ο μέσος χρόνος ολοκλήρωσης της διαδρομής είναι 3:38.72 με 2.5 σφάλματα. Αντίθετα, με τη χρήση της εφαρμογής, οι εθελοντές ολοκλήρωναν τη διαδρομή με μέσο χρόνο τα 5:49.84 και 4.3 σφάλματα (Πίνακας 4.2). Δηλαδή παρατηρούμε αύξηση κατά δύο λεπτά περίπου στο μέσο χρόνο ολοκλήρωσης.

Επιπλέον υπολογίσαμε δύο ακόμη μεγέθη (Πίνακας 4.3). Αρχικά, υπολογίσαμε

τη μέση ποσοστιαία διαφορά των χρόνων εθελοντών, το οποίο μας επιτρέπει να αντιληφθούμε σε ποια τμήματα παρουσιαζόταν η μεγαλύτερη διαφορά μεταξύ των χρόνων των δύο μεθόδων περιήγησης. Όπως παρατηρούμε, στο τμήμα 3 της διαδρομής, η διαφορά αυξάνεται ραγδαία, το οποίο δηλώνει ότι δυσκολεύτηκαν ιδιαίτερα με τη χρήση της εφαρμογής. Η αύξηση, όπως παρατηρήσαμε και κατά το πείραμα, οφείλεται στο γεγονός ότι στο συγκεκριμένο τμήμα, στη διαδρομή εμφανιζόταν ένα στένο πέρασμα, το οποίο δυσκολεύτηκαν να αντιληφθούν οι εθελοντές με την εφαρμογή. Στη συνέχεια, καταγράφουμε τη ποσοστιαία διαφορά των μέσων χρόνων, με το οποίο μπορούμε να αντιληφθούμε πόσο σημαντική ήταν η αύξηση του χρόνου στην περίπτωση της εφαρμογής. Στην πλεινότητα των τμημάτων, ο χρόνος χρήσης της εφαρμογής ήταν 1.5 φορές μεγαλύτερη από αυτόν με τη χρήση του μπαστούνιού. Εξαίρεση και σε αυτή την περίπτωση αποτελεί το τμήμα 3, όπου, λόγω της ίδιας δυσκολίας, οι εθελοντές χρειάστηκαν τον διπλάσιο χρόνο. Η ποσοστιαία διαφορά των χρόνων ενός εθελοντή ή των μέσων χρόνων ενός τμήματος προέκυψε με βάση τον τύπο:

$$\Pi\Delta\% = \frac{t_{\text{Εφαρμογή}} - t_{\text{Μπαστούνι}}}{t_{\text{Μπαστούνι}}} \times 100\%$$

Για μια πιο πλήρη παρουσίαση των ανωτέρων δεδομένων, θα πραγματοποιήσουμε στατιστική ανάλυση αυτών. Πρωταρχικός μας στόχος με την ανάλυση αυτή είναι να δώσουμε απάντηση στα ακόλουθα ερωτήματα για την εφαρμογή, τα οποία αποτέλεσαν και τους κύριους στόχους προς επίτευξη κατά την υλοποίηση αυτής:

- Μπορεί ένας χρήστης, χρησιμοποιώντας την εφαρμογή, να περιηγηθεί το ίδιο γρήγορα ή γρηγορότερα σε έναν εσωτερικό χώρο, σε σχέση με την περίπτωση που χρησιμοποιεί το μπαστούνι;
- Μπορεί ένας χρήστης να περιηγηθεί σε έναν χώρο με περισσότερη ασφάλεια, όταν χρησιμοποιεί την εφαρμογή αντί για το μπαστούνι;

Επομένως, και εφόσον διεξήχθη ένα within-subjects πείραμα, όπου καταγράφηκαν οι επιδόσεις των εθελοντών υπό δύο συνθήκες (χρήση μπαστούνιο και χρήση εφαρμογής), θα πραγματοποιήσουμε ένα paired-samples t test με ανεξάρτητη μεταβλητή τον τρόπο υποβοήθησης των χρηστών για την περιήγηση τους στο χώρο [109]. Απαραίτητες προϋποθέσεις για να εφαρμόσουμε ένα paired-samples t test είναι:

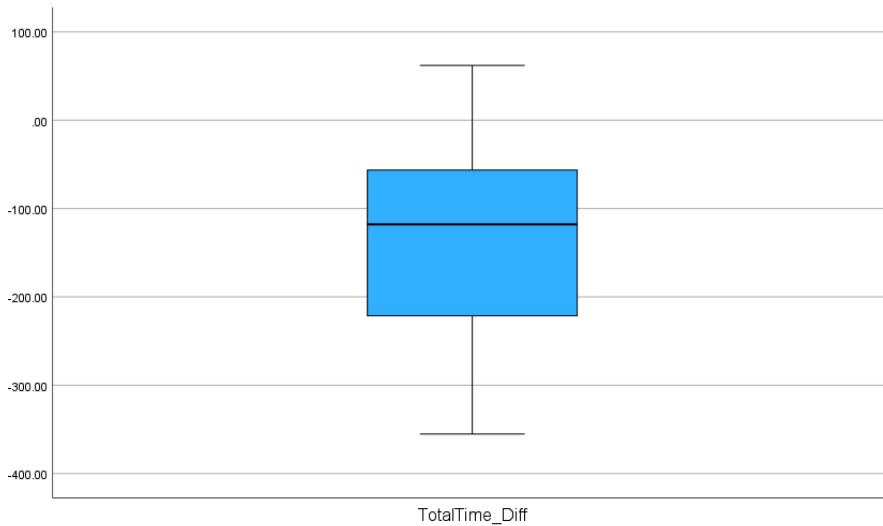
- Η εξαρτημένη μεταβλητή να έχει συνεχείς τιμές.
- Οι παρατηρήσεις μας να είναι ανεξάρτητες η μία από την άλλην.
- Δεν υπάρχουν ακραίες τιμές
- Οι τιμές της εξαρτημένης μεταβλητής είναι κανονικά κατανεμημένες

Αρχικά, για να απαντήσουμε το πρώτο ερώτημα που θέσαμε, θα μελετήσουμε τα δείγματα συνολικού χρόνου ολοκλήρωσης της διαδρομής που έπρεπε να ακολουθήσουν οι εθελοντές. Οι δύο πρώτες προϋποθέσεις γνωρίζουμε ότι ισχύουν, οπότε πρέπει να ελέγχουμε τις τελευταίες δύο, ελέγχοντας τη διαφορά των χρόνων για τους εθελοντές. Η διαφορά ορίζεται ως:

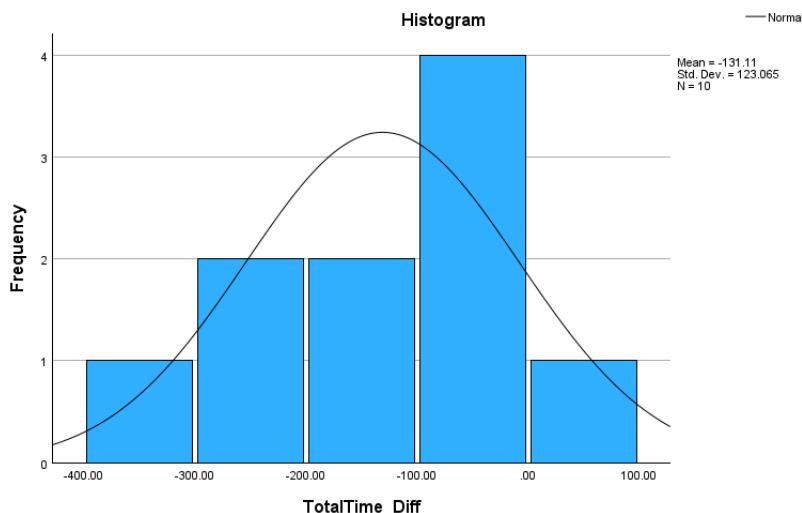
$$t_{\text{Diff}} = t_{\text{Μπαστούνι}} - t_{\text{Εφαρμογή}}$$

Με βάση το Σχήμα 4.4, δεν παρατηρείται κάποια ακραία τιμή στο σύνολο των δειγμάτων μας.

Παρατηρώντας το Σχήμα 4.5, διαπιστώνουμε ότι τα δεδομένα μας είναι κανονικά κατανεμημένα, λόγω της μορφής του ιστογράμματος, καθώς και ότι $p = 0.981 > 0.05$, που προκύπτει από το test Shapiro-Wilk [111].



Σχήμα 4.4: Θηκόγραμμα της διαφοράς των συνολικών χρόνων



(α') Ιστόγραμμα της διαφοράς των συνολικών χρόνων

Tests of Normality						
	Kolmogorov-Smirnov ^a			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
TotalTime_Diff	.156	10	.200*	.983	10	.981

*. This is a lower bound of the true significance.
a. Lilliefors Significance Correction

(β') Αποτελέσματα Normality Test

Σχήμα 4.5: Έλεγχος δειγμάτων για κανονική κατανομή στα δείγματα χρόνων ολοκλήρωσης

Επομένως, με βάση τα ανωτέρω, καλύπτουμε τις προϋποθέσεις για να πραγματοποιήσουμε ένα paired-samples t test. Αρχικά, εκφράζουμε τις υποθέσεις μας:

- H_0 : Η χρήση της συσκευής HoloLens 2 και της εφαρμογής δεν οδηγεί σε μικρό-

τερο χρόνο ολοκλήρωσης της διαδρομής ($\mu_{Μπαστούνι} - \mu_{Εφαρμογή} \leq 0$)

- H_0 : Η χρήση της συσκευής HoloLens 2 και της εφαρμογής οδηγεί σε μικρότερο χρόνο ολοκλήρωσης της διαδρομής ($\mu_{Μπαστούνι} - \mu_{Εφαρμογή} > 0$)

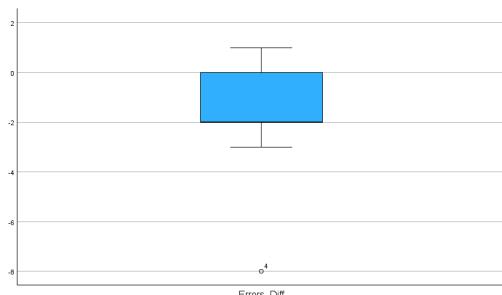
Για να απορρίψουμε την μηδενική υπόθεση, πρέπει να είναι $t > t_{9,0.05}$, όπου $t_{9,0.05} = 1.833$ και για να θεωρηθούν τα αποτελέσματά μας στατιστικά σημαντικά, πρέπει $p < 0.05$.

Από τα αποτελέσματα (Σχήμα 4.6) προκύπτει ότι το αποτέλεσμα ήταν στατιστικά σημαντικό, καθώς $p = 0.004 < 0.05$. Επομένως, παρατηρείται σημαντική διαφορά μεταξύ των χρόνων που καταγράφηκαν κάτω από τις δύο συνθήκες. Ωστόσο δεν μπορούμε να απορρίψουμε την μηδενική υπόθεση, καθώς ισχύει: $t = -3.369 < t_{9,0.05}$. Συμπεραίνουμε ότι η χρήση της εφαρμογής αύξησε τον συνολικό χρόνο ολοκλήρωσης της διαδρομής κατά 131.11 ± 38.92 δευτερόλεπτα.

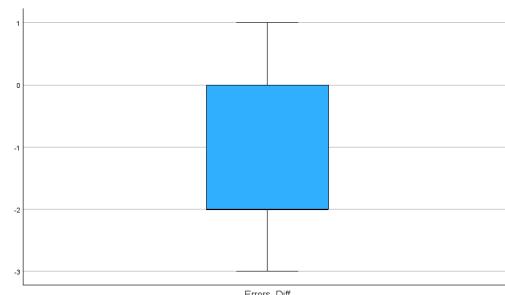
Paired Samples Test										
	Paired Differences					Significance				
	Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference			t	df	One-Sided p	Two-Sided p
Pair 1 TotalTime_C - TotalTime_H	-131.11400	123.06454	38.91642	-219.14907	-43.07893	-3.369	9	.004	.008	

Σχήμα 4.6: Αποτελέσματα του paired-samples t test για τους χρόνους ολοκλήρωσης διαδρομής

Στη συνέχεια, εφαρμόζουμε μια παρόμοια διαδικασία και για τα δείγματα που αφορούν το σύνολο σφαλμάτων που πραγματοποίησαν οι εθελοντές κατά το πείραμα, ώστε να δοθεί απάντηση και στο δεύτερο ερώτημα που θέσαμε. Αρχικά ελέγχουμε αν τα δείγματα μας έχουν κάποια ακραία τιμή. Σε αυτήν την περίπτωση (Σχήμα 4.7α'), παρατηρούμε ότι υπάρχουν ακραίες τιμές και ειδικότερα αυτές που αντλήσαμε από τον Εθελοντή 4. Όπως θα παρατηρήσουμε και στον (Πίνακα 4.1), ενώ η διαφορά σφαλμάτων των εθελοντών κυμαίνεται στο εύρος $[-1, 3]$, για τον εθελοντή 4, η διαφορά αυτή είναι $+8$, δηλαδή αρκετά υψηλότερη από τη μέγιστη τιμή του εύρους που ορίσαμε. Οπότε, στη συνέχεια της ανάλυσης μας, θα αφαιρέσουμε το συγκεκριμένο δείγμα.



(α') Θηκόγραμμα με ακραία τιμή

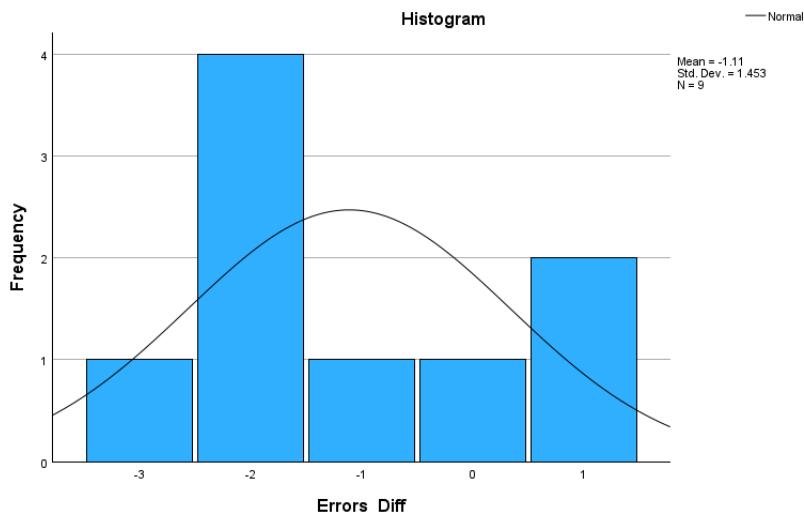


(β') Θηκόγραμμα χωρίς ακραία τιμή

Σχήμα 4.7: Θηκόγραμμα της διαφοράς του πλήθους σφαλμάτων

Αφού ελέγξαμε ότι τα δεδομένα παρουσιάζουν κανονική κατανομή (Σχήμα 4.8), πραγματοποιούμε το paired-samples t test με τις υποθέσεις:

- H_0 : Η χρήση της συσκευής HoloLens 2 και της εφαρμογής δεν οδηγεί σε μικρότερο αριθμό σφαλμάτων ($\mu_{Μπαστούνι} - \mu_{Εφαρμογή} \leq 0$)
- H_a : Η χρήση της συσκευής HoloLens 2 και της εφαρμογής οδηγεί σε μικρότερο αριθμό σφαλμάτων ($\mu_{Μπαστούνι} - \mu_{Εφαρμογή} > 0$)



(α') Ιστόγραμμα της διαφοράς των σφαλμάτων

Tests of Normality

	Kolmogorov-Smirnov ^a			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
Errors_Diff	.285	9	.033	.864	9	.105

a. Lilliefors Significance Correction

(β') Αποτελέσματα Normality Test

Σχήμα 4.8: Έλεγχος δειγμάτων για κανονική κατανομή στα δείγματα σφαλμάτων

Με βάση τα αποτελέσματα, όπως αυτά παρουσιάζονται στο Σχήμα 4.9, πρόκειται για στατιστικά σημαντικά αποτελέσματα ($p = 0.025 < 0.05$), τα οποία, ωστόσο, δεν οδηγούν σε απόρριψη της μηδενικής υπόθεσης, καθώς ισχύει: $t = -2.294 < t_{8,0.05}$. Η μηδενική υπόθεση θα είχε απορριφθεί αν ίσχυε: $t > t_{8,0.05}$, όπου $t_{8,0.05} = 1.860$. Ειδικότερα, η χρήση της εφαρμογής οδηγεί σε 1.11 ± 0.48 περισσότερα σφάλματα από τη χρήση του μπαστουνιού.

	Paired Samples Test									
	Paired Differences			95% Confidence Interval of the Difference				Significance		
	Mean	Std. Deviation	Std. Error Mean	Lower	Upper	t	df	One-Sided p	Two-Sided p	
Pair 1 TotalErrors_C - TotalErrors_H	-1.111	1.453	.484	-2.228	.006	-2.294	8	.025	.051	

Σχήμα 4.9: Αποτελέσματα του paired-samples t test για τους χρόνους ολοκλήρωσης διαδρομής

4.2.2 Καταγραφή Απαντήσεων Ερωτηματολογίου και Παρατηρήσεις

Μετά την ολοκλήρωση της πειραματικής διαδικασίας, οι εθελοντές έπρεπε να συμπληρώσουν ένα ερωτηματολόγιο. Το ερωτηματολόγιο αποτελούταν από μερικές εισαγωγικές ερωτήσεις, από τις οποίες προσπαθήσαμε να αντλήσουμε πληροφορίες σχετικά με προηγούμενες εμπειρίες των χρηστών με την τεχνολογία που μελετούσαμε. Στη συνέχεια συμπλήρωναν ένα ερωτηματολόγιο, το οποίο βασίστηκε στο ερωτηματολόγιο System Usability Scale (SUS), ενώ στο τέλος απάντησαν σε τρεις ερωτήσεις

ανοικτού τύπου. Στο παρόν κεφάλαιο, θα αναλύσουμε τα δύο τελευταία τμήματα του ερωτηματολογίου.

Το SUS είναι ουσιαστικά μια κλίμακα Likert 5 σημείων [112] αποτελούμενη από δέκα προτάσεις. Οι εθελοντές επιλέγούν για κάθε πρόταση έναν αριθμό από το 1 έως το από 5 ανάλογα πόσο συμφωνούν ή διαφωνούν με την πρόταση αυτή. Το SUS ερωτηματολόγιο δημιουργήθηκε με σκόπο να αποτέλεσει ένα σύντομο, εύκολο στη χρήση ερωτηματολόγιο, το οποίο περιγράφει την αντικειμενική χρηστικότητα ενός συστήματος ή μιας εφαρμογής. Οι προτάσεις έχουν διατυπωθεί με τέτοιο τρόπο, ώστε οι εθελοντές να επιλέγουν με σχετική ευκολία κάποιο άκρο της κλίμακας, έτσι ώστε να περιοριστούν ασαφείς απαντήσεις, δηλαδή απαντήσεις στη μέση της κλίμακας. Επίσης, οι θετικές και οι αρνητικές προτάσεις εναλλάσσονται έτσι ώστε ο εθελοντές να διατηρούν την προσοχή τους καθώς διαβάζουν τις προτάσεις [110]. Οι προτάσεις του ερωτηματολογίου παρατίθενται στο Παράρτημα B, ενώ οι απαντήσεις των συμμετεχόντων ανά πρόταση στον Πίνακα 4.4. Συγκεκριμένα, στον πίνακα καταγράφουμε για κάθε πρόταση πόσοι εθελοντές επέλεξαν μία συγκεκριμένη βαθμίδα, ενώ στην τελευταία στήλη σημειώνουμε την μέση τιμή για τη συγκεκριμένη πρόταση.

Πίνακας 4.4: Απαντήσεις στο ερωτηματολόγιο SUS

Προτάσεις	1	2	3	4	5	Μέση Τιμή
Πρόταση 1	0	2	3	3	2	3.5
Πρόταση 2	5	4	1	0	0	1.6
Πρόταση 3	0	0	3	3	4	4.1
Πρόταση 4	5	2	2	1	0	1.9
Πρόταση 5	0	2	1	3	4	3.9
Πρόταση 6	5	3	2	0	0	1.7
Πρόταση 7	0	0	1	3	6	4.5
Πρόταση 8	7	3	0	0	0	1.3
Πρόταση 9	0	2	2	4	2	3.6
Πρόταση 10	8	2	0	0	0	1.2

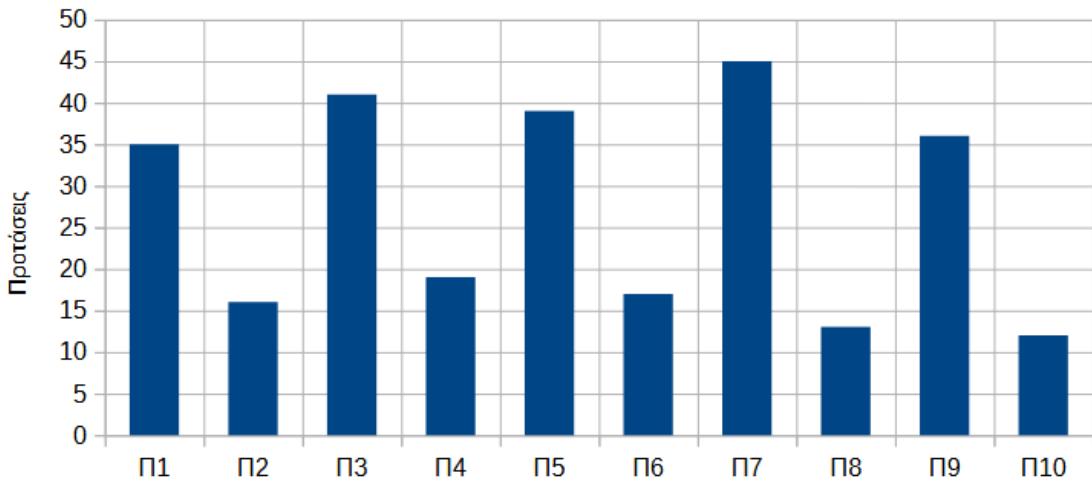
Παράλληλα, στο Σχήμα 4.10 παρουσιάζουμε το συνολικό σκορ που καταγράφηκε για κάθε ερώτηση, δηλαδή το άθροισμα των βαθμών στην κλίμακα, που επέλεξαν οι εθελοντές για κάθε πρόταση. Όπως αναφέραμε προηγουμένως, οι θετικές και οι αρνητικές προτάσεις εναλλάσσονται. Επομένως, είναι σημαντικό οι ερωτήσεις με περιττό αύξοντα αριθμό να έχουν υψηλό σκορ, ενώ αυτές με ζυγό αύξοντα αριθμό να έχουν χαμηλό σκορ, ώστε το τελικό αποτέλεσμα να είναι θετικό. Αυτό γίνεται ξεχάθαρο και από τη σχέση με βάση την οποία υπολογίζεται το τελικό SUS Score:

$$\text{SUS Score} = 2.5 \times \left(\sum_{n=1}^5 (x_{2n-1} - 1) + \sum_{n=1}^5 (5 - x_{2n}) \right)$$

Με βάση τις μέσες τιμές για κάθε πρόταση και την ανωτέρω σχέση, υπολογίζουμε ότι το τελικό SUS Score είναι 79.75/100. Επομένως, παρά τις χαμηλές επιδόσεις της

Ερωτηματολόγιο SUS

Συνολικό Σκορ



Σχήμα 4.10: Συνολικό σκορ προτάσεων του ερωτηματολογίου SUS

εφαρμογής όσον αφορά το χρόνο ολοκλήρωσης μιας διαδρομής και το πλήθος σφαλμάτων, οι εθελοντές αναγνωρίζουν τη χρηστικότητά της.

Οι λόγοι αυτής της αντίφασης γίνονται ξεκάθαροι με τη βοήθεια των ερωτήσεων ανοικτού τύπου, όπου οι εθελοντές εξηγούν τους λόγους για τους οποίους δυσκολεύτηκαν να ολοκληρώσουν το task σε χρόνο παρόμοιο ή ταχύτερο με αυτό του μπαστουνιού, καθώς και τι ήταν αυτό που ξεχώρισαν στην εφαρμογή. Οι ερωτήσεις ανοικτού τύπου απαντήθηκαν επί τον πλείστον με προφορικό τρόπο, ενώ ελάχιστοι ήταν που παρείχαν τις απαντήσεις τους γραπτά. Λόγω του πλήθους των απαντήσεων, στη συνέχεια θα παραθέσουμε τα σημαντικότερα σημεία από αυτές, καθώς και ορισμένες από τις παρατηρήσεις μας.

Ερώτηση 1: Σας βοήθησε η εφαρμογή να αντιληφθείτε τη διάταξη του χώρου και να περιηγηθείτε σε αυτόν και με ποιον τρόπο;

Η πλειονότητα των εθελοντών ανέφεραν ότι η εφαρμογή τους βοήθησε να αντιληφθούν τη διάταξη του χώρου, καθώς με μια απλή περιστρόφη, γνώριζαν που βρίσκονταν τα εμπόδια και σε ποια απόσταση, ενώ σε συνδυασμό με την αίσθηση του προσανατολισμού, γνώριζαν που βρίσκονταν στο χώρο. Ωστόσο υπήρχαν δύο εθελοντές, οι οποίοι ένιωσαν περισσότερη σιγουριά με το μπαστούνι και, όπως τονίζει ο ένας από αυτούς, ο ήχος των ανάγκαζε να επικεντρωθεί κυρίως στο τώρα και στο εμπόδιο που αντιμετωπίζει, παρά να συγκεντρωθεί να σχηματίσει μια εικόνα του χώρου στο μυαλό του. Τέλος, ένας ακόμη εθελοντής τόνισε ότι το μπαστούνι τον βοήθησε να αντιληφθεί καλύτερα εμπόδια μικρά και αρκετά χαμηλά στο έδαφος.

Ερώτηση 2: Τι ήταν αυτό το οποίο σας δυσκόλεψε κάτια τη χρήση της εφαρμογής;

Στο ερώτημα αυτό, όλοι οι εθελοντές ανέφεραν ελαττώματα σχετικά με τους

προειδοποιητικούς ήχους. Ειδικότερα, εφτά στους δέκα εθελοντές σχολίασαν τη μίξη των προειδοποιητικών ήχων, το οποίο τους δυσκόλεψε να αντιληφθούν ποια είναι η σωστή απόσταση του εμποδίου. Το πρόβλημα είχε αναγνωριστεί πριν από το πείραμα και αναφέρθηκε στο Κεφάλαιο 4.1.2. Οι υπόλοιποι ανέφεραν ότι το πλήθος των προειδοποιητικών ήχων, οι οποίοι διαφέρουν ως προς τη συχνότητα, τους δυσκόλεψε να διακρίνουν μεταξύ τους τα διαφορετικά επίπεδα προειδοποίησης. Τέλος, ένας εθελοντής σχολίασε ότι ο χωρικός ήχος δεν ήταν ιδιαίτερα αισθητός και δεν τον βοήθησε να αντιληφθεί την κατεύθυνση του ήχου.

Ερώτηση 3: Ήταν κάτι το οποίο θεωρείτε ότι έλειπε από την εφαρμογή και θα σας διευκόλυνε την εμπειρία;

Αρκετοί ήταν οι εθελοντές που ανέφεραν ότι θα προτιμούσαν λιγότερους και πιο ευδιάκριτους προειδοποιητικούς ήχους. Μερικοί πρόσθεσαν ότι σε κάποιο επίπεδο προειδοποίησης θα μπορούσε να αντικατασταθεί από άλλο τρόπο προειδοποίησης, όπως η απτική ανάδραση, οι φωτεινές σημάνσεις ή η εκφώνηση αναλυτικών οδηγιών ως προς την ακριβή απόσταση των εμποδίων ή/και την κατεύθυνση που πρέπει να ακολουθήσει. Ορισμένες ακόμη ιδέες που προτάθηκαν ήταν η ενσωμάτωση αναγνώρισης εμποδίων, καθώς και η δυναμική αύξηση και μείωση του εύρους του cast, έτσι ώστε να εντοπίζονται ευκολότερα στενά περάσματα.

Όπως μπορούμε να αντιληφθούμε από τις ανωτέρω απαντήσεις, οι προειδοποιητικοί ήχοι ήταν αυτοί που αποτέλεσαν το κυριότερο μειονέκτημα της εφαρμογής και, όπως φάνηκε, επηρέασαν σημαντικά τις επιδόσεις των εθελοντών σε σχέση με τη χρήση του μπαστονιού, καθώς οι εθελοντές αναγκάζονταν συχνά να σταματήσουν και να περιμένουν μέχρις ότου γίνει ξεκάθαρο ποιος είναι ο σωστός προειδοποιητικός ήχος. Αιτία αυτού αποτελούν οι ασυνέπειες που δημιουργούνται στο mesh και η συχνή αλλαγή αυτού σε κάθε interval. Λύση σε αυτό θα μπορούσε να αποτελέσει η αντικατάσταση του *BoxCast*, το οποίο είναι ευαίσθητο στις ασυνέπειες αυτές, με collision detection, δηλαδή η δημιουργία ζωνών γύρω από το χρήστη και η αναγνώριση συγκρούσεων του mesh εμποδίων με τις ζώνες αυτές. Ωστόσο, πέρα αυτού του προβλήματος, οι εθελοντές χαρακτήρισαν τη συσκευή ιδιαίτερα όνετη και τη χρήση της εφαρμογής αρκετά απλή και κατανοητή και ιδιαίτερα χρηστική στο σύνολό της. Επίσης, με βάση τις απαντήσεις τους στο τελευταίο ερώτημα, φαίνεται ότι γίνεται ιδιαίτερα αντιληπτό η δυνατότητα συνεχής βελτίωσης της εφαρμογής και η ενσωμάτωση πολλών νέων δυνατοτήτων που εκμεταλλεύονται πληθώρα άλλων τεχνολογιών και αισθήσεων, κάτι το οποίο δύσκολα υλοποιείται σε ένα μπαστούνι.

ΚΕΦΑΛΑΙΟ

5

ΠΡΟΕΚΤΑΣΕΙΣ ΚΑΙ ΕΠΙΛΟΓΟΣ

5.1 Μελλοντικές προεκτάσεις

Ο κύριος στόχος της διπλωματικής εργασίας ήταν η υλοποίηση μιας εφαρμογής, η οποία θα προσέφερε ουσιαστική βοήθεια σε άτομα με προβλήματα όρασης στην περιήγηση τους σε έναν άγνωστο χώρο. Η ανάπτυξή της βασίζεται στην αξιοποίηση σύγχρονων τεχνολογιών, όπως είναι η Μικτή Πραγματικότητα, η οποία, μέχρι και σήμερα, δεν έχει καταφέρει να εισέλθει σε μεγάλο βαθμό στην καθημερινότητα του μέσου ανθρώπου. Η χρήση της συσκευής Microsoft HoloLens 2 αποτέλεσε ιδανική επιλογή, διότι διαθέτει πληθώρα αισθητήρων απαραίτητων για τους σκοπούς της εφαρμογής μας σε μια φορητή συσκευή, μικρού σχετικά μεγέθους. Έτσι μπορέσαμε να επικεντρωθούμε στην ανάπτυξη λογισμικού χωρίς να υπάρχει η ανάγκη δημιουργίας εξειδικευμένου hardware. Επιπλέον, αποτελεί ευκαιρία ώστε άτομα με προβλήματα όρασης να βιώσουν - εν μέρει - την εμπειρία της Μικτής Πραγματικότητας, η οποία γίνεται περισσότερο προσβάσιμη προς αυτούς.

Η εφαρμογή, η οποία δημιουργήθηκε, διαθέτει τις απαραίτητες λειτουργίες, ώστε να αποτελέσει ένα πρωτότυπο και να παρουσιαστεί η χρηστικότητά της, ωστόσο βρίσκεται σε ένα αρχικό στάδιο ανάπτυξης. Ιδιαίτερη σημαντική κρίνεται η βελτίωση των προειδοποιητικών ήχων, το οποίο θα βελτίωνε δραματικά την εμπειρία των χρηστών. Όπως αναφέρθηκε και η προηγουμένως, η δημιουργία 'hitboxes-ζωνών' και η αναγνώριση συγκρούσεων (collision detection) αυτών με το mesh του χώρου, θα μπορούσε να συμβάλλει σε καλύτερο εντοπισμό εμποδίων, οπότε και σε πιο ευδιάκριτους προειδοποιητικούς ήχους, χωρίς αυτοί να «μπερδεύονται». Επιπλέον, η εφαρμογή επιδέχεται πλήθος αναβαθμίσεων, οι οποίες θα μπορούσαν να βελτιώσουν την απόδοσή της, καθώς και την εμπειρία του χρήστη. Μεταξύ των άλλων, μεγαλύτερο ενδιαφέρον παρουσιάζει η χρήση τεχνητής νοημοσύνης σε συνδυασμό με υπολογιστική όραση με σκοπό, όχι μόνο τον καλύτερο εντοπισμό εμποδίων, αλλά και για την πρόβλεψη πιθανών συγκρούσεων όπως και για την αναγνώριση αντικειμένων και εμποδίων. Προσφέροντας στο χρήστη τη γνώση για το εμπόδιο που βρίσκεται στο δρόμο του, δηλαδή αν πρόκειται για ένα τοίχο, μια καρέκλα, έναν άνρθωπο ή μια πόρτα, του δίνεται η δυνατότητα να προσαρμόσει ανάλογα τον τρόπο αντιμετώπισης αυτής της δυσκολίας, καθώς και να κινείται με μεγαλύτερη ελευθερία στο χώρο, γνωρίζοντας τι υπάρχει σε αυτόν ή από που να μεταβεί σε κάποιο άλλο χώρο. Επίσης, χρήσιμη θα ήταν η δυνατότητα σύνδεσης του κινητού τηλεφώνου του χρήστη με την εφαρμογή μέσω ενός companion app, καθιστώντας τη διαχείριση αυτής και των λειτουργιών της ευκολότερη λόγω των επιλογών προσβασιμότητας που προσφέρει, όπως είναι οι screen readers. Με αυτό τον τρόπο, ο χρήστης δε θα χρειάζεται να βασίζεται αποκλειστικά στη χρήση φωνητικών εντολών. Παράλληλα, το κινητό μπορεί ακόμη να προσφέρει απτική ανάδραση και να ενημερώνει το χρήστη, όταν πλησιάζει αρκετά σε κάποιο εμπόδιο.

5.2 Επίλογος

Στην παρούσα διπλωματική εργασία, παρουσιάσαμε, αρχικά, το θεωρητικό και τεχνολογικό υπόβαθρο στο και, έπειτα, τον τρόπο υλοποίησης μιας εφαρμογής αξιοποιώντας τις δυνατότητες της συσκευή μικτής πραγματικότητας Microsoft HoloLens 2. Με την ανάπτυξή της, προσπαθήσαμε να ευαισθητοποιήσουμε σχετικά με ένα καθημερινό πρόβλημα που αντιμετωπίζει μια ιδιαίτερα ευάλωτη, προσφέροντας, παράλληλα, μία λύση σε αυτό. Η λύση αυτή εκμεταλλεύεται νέες τεχνολογίες, οι οποίες δεν είχαν αξιοποιηθεί ευρύτατα για παρόμοια ζητήματα, με σκοπό να εκσυγχρονίσει ή να χρησιμοποιηθεί συνδυαστικά με ήδη υπάρχουσες λύσεις, ώστε να προσφέρει στο χρήστη

περισσότερη ελευθερία και ανεξαρτησία.

ΒΙΒΛΙΟΓΡΑΦΙΑ

- [1] A. Bradford and A. Harvey, “The five (and more) senses,” Live Science, 2017. [Online]. Available: <https://www.livescience.com/60752-human-senses.html>
- [2] W. H. Organization, “Blindness and vision impairment,” World Health Organization, 08 2023. [Online]. Available: <https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment>
- [3] M. Langelaan, Quality of Life of Visually Impaired Working Age Adults, 2007. [Online]. Available: https://www.researchgate.net/publication/239845719_Quality_of_life_of_visually_impaired_working_age_adults
- [4] K. K. Morgan, “The role of augmented reality in medicine,” WebMD, 05 2021. [Online]. Available: <https://www.webmd.com/a-to-z-guides/features/augmented-reality-medicine>
- [5] W. Contributors, “Mixed reality,” Wikipedia, 04 2019. [Online]. Available: https://en.wikipedia.org/wiki/Mixed_reality
- [6] N. E. Institute, “How the eyes work | national eye institute,” Nih.gov, 04 2022. [Online]. Available: <https://www.nei.nih.gov/learn-about-eye-health/healthy-vision/how-eyes-work>
- [7] K. Anspaugh, S. Goncalves, E. Jackson-Osagie, and S. Q. Smith, “Vision,” louis.pressbooks.pub, 08 2022. [Online]. Available: <https://louis.pressbooks.pub/medicalterminology/chapter/vision/>
- [8] W. H. Organization, “World report on vision,” www.who.int, 10 2019. [Online]. Available: <https://www.who.int/publications/i/item/9789241516570>
- [9] J. D. Adelson et al., “Causes of blindness and vision impairment in 2020 and trends over 30 years, and prevalence of avoidable blindness in relation to vision 2020: the right to sight: an analysis for the global burden of disease study,” The Lancet Global Health, vol. 9, p. e144–e160, 02 2021. [Online]. Available: [https://www.thelancet.com/journals/langlo/article/PIIS2214-109X\(20\)30489-7/fulltext](https://www.thelancet.com/journals/langlo/article/PIIS2214-109X(20)30489-7/fulltext)
- [10] S. K. West, G. S. Rubin, A. T. Broman, B. Muñoz, K. Bandeen-Roche, K. Turano, and S. Project, “How does visual impairment affect performance on tasks of everyday life?: the see project,” Archives of Ophthalmology, vol. 120, no. 6, pp.

- 774–780, 06 2002. [Online]. Available: <https://doi.org/10.1001/archopht.120.6.774>
- [11] M. KHORRAMI-NEJAD, A. SARABANDI, M.-R. AKBARI, and F. ASKARIZADEH, “The impact of visual impairment on quality of life,” *Medical Hypothesis, Discovery and Innovation in Ophthalmology*, vol. 5, p. 96–103, 2016. [Online]. Available: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5347211/>
- [12] M. Langelaan, M. R. de Boer, R. M. A. van Nispen, B. Wouters, A. C. Moll, and G. H. M. B. van Rens, “Impact of visual impairment on quality of life: A comparison with quality of life in the general population and with other chronic conditions,” *Ophthalmic Epidemiology*, vol. 14, pp. 119–126, 01 2007.
- [13] M. Mashiata, T. Ali, P. Das, Z. Tasneem, F. R. Badal, S. K. Sarker, M. Hasan, S. H. Abhi, M. R. Islam, F. Ali, H. Ahamed, M. Islam, and S. K. Das, “Towards assisting visually impaired individuals: A review on current status and future prospects,” *Biosensors and Bioelectronics: X*, vol. 12, p. 100265, 10 2022.
- [14] I. U. Library, “Libguides: Blind/visual impairment: Common assistive technologies,” Illinois.edu, 2013. [Online]. Available: <https://guides.library.illinois.edu/c.php?g=526852&p=3602299>
- [15] A. F. for the Blind, “Screen readers | american foundation for the blind,” Afb.org, 2019. [Online]. Available: <https://www.afb.org/blindness-and-low-vision/using-technology/assistive-technology-products/screen-readers>
- [16] W3Schools, “Html accessibility,” W3schools.com, 2020. [Online]. Available: https://www.w3schools.com/html/html_accessibility.asp
- [17] “biped | smart copilot for blind and visually impaired people,” biped. [Online]. Available: <https://www.biped.ai>
- [18] A. Ghebali, “Orcam myeye - the ultimate visual aid device for the people with visual impairment,” OrCam Technologies, 02 2023. [Online]. Available: <https://www.orcam.com/en-us/orcam-myeye>
- [19] “Blindsight,” Blindsight. [Online]. Available: <https://www.blindsight.com>
- [20] “Be my eyes - bringing sight to blind and low-vision people,” Bemyeyes.com, 2019. [Online]. Available: <https://www.bemyeyes.com>
- [21] “WeWalk smart cane,” WeWALK Smart Cane. [Online]. Available: <https://wewalk.io/en/>
- [22] “Seeing ai,” Microsoft Garage. [Online]. Available: <https://www.microsoft.com/en-us/garage/wall-of-fame/seeing-ai/>
- [23] S. Mann, V. Phillip, T. A. Furness, Y. C. Yuan, J. Iorio, and Z. Wang, “Fundamentals of all the realities: Virtual, augmented, mediated, multimediated, and beyond,” *Springer eBooks*, pp. 3–34, 01 2023.
- [24] S. Mann and C. Wyckoff, “Extended reality,” Wearcam.org, 1991. [Online]. Available: <http://wearcam.org/xr.txt>
- [25] P. Milgram, H. Takemura, A. Utsumi, and F. Kishino, “Augmented reality: A class of displays on the reality-virtuality continuum,” *Telemanipulator and Telepresence Technologies*, vol. 2351, 01 1994.
- [26] J. Weiler, “Phantom touch: Vr reveals new insights in human perception,” Neuroscience News, 11 2023. [Online]. Available: <https://neurosciencenews.com/vr-tactile-perception-phantom-touch-25208/>
- [27] *Touching Virtual Reality: A Review of Haptic Gloves*, ACTUATOR 2018; 16th International Conference on New Actuators. VDE, 2018.

- [28] H. E. Lowood, *Virtual Reality | Computer Science*, 11 2018. [Online]. Available: <https://www.britannica.com/technology/virtual-reality>
- [29] C. E. Loeffler, “Distributed virtual reality: Applications for education, entertainment and industry,” *Telektronikk*, vol. 89, p. 83–88, 1993.
- [30] Meta, “Xtadium on meta quest: Get closer to sports you love in vr,” Meta, 11 2022. [Online]. Available: <https://about.fb.com/news/2022/11/xtadium-quest-sports-in-vr/>
- [31] A. Ansari, V. K. Shukla, K. Saxena, and B. Filomeno, “Implementing virtual reality in entertainment industry,” *Springer eBooks*, pp. 561–570, 09 2021.
- [32] A. Hamad and B. Jia, “How virtual reality technology has changed our lives: An overview of the current and potential applications and limitations,” *International Journal of Environmental Research and Public Health*, vol. 19, p. 11278, 09 2022. [Online]. Available: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9517547/>
- [33] S. Kavanagh, A. Luxton-Reilly, B. Wuensche, and B. Plimmer, “A systematic review of virtual reality in education,” *Themes in Science and Technology Education*, vol. 10, p. 85–119, 12 2017.
- [34] *A literature review on immersive virtual reality in education: state of the art and perspectives*, vol. 1, no. 133, 2015.
- [35] A. Chirico, F. Lucidi, M. De Laurentiis, C. Milanese, A. Napoli, and A. Giordano, “Virtual reality in health system: Beyond entertainment. a mini-review on the efficacy of vr during cancer treatment,” *Journal of Cellular Physiology*, vol. 231, pp. 275–287, 10 2015.
- [36] A. J. Snoswell and C. L. Snoswell, “Immersive virtual reality in health care: Systematic review of technology and disease states,” *JMIR Biomedical Engineering*, vol. 4, p. e15025, 09 2019.
- [37] J. Gerup, C. B. Soerensen, and P. Dieckmann, “Augmented reality and mixed reality for healthcare education beyond surgery: an integrative review,” *International Journal of Medical Education*, vol. 11, pp. 1–18, 01 2020.
- [38] W. L. Hosch, *Augmented Reality | Computer Science*, 2020. [Online]. Available: <https://www.britannica.com/technology/augmented-reality>
- [39] J. Carmigniani and B. Furht, “Augmented reality: An overview,” *Handbook of Augmented Reality*, pp. 3–46, 2011.
- [40] S. M. Ko, W. S. Chang, and Y. G. Ji, “Usability principles for augmented reality applications in a smartphone environment,” *International Journal of Human-Computer Interaction*, vol. 29, pp. 501–515, 08 2013.
- [41] H.-K. Wu, S. W.-Y. Lee, H.-Y. Chang, and J.-C. Liang, “Current status, opportunities and challenges of augmented reality in education,” *Computers and Education*, vol. 62, pp. 41–49, 03 2013.
- [42] K. Lee, “Augmented reality in education and training,” *TechTrends*, vol. 56, pp. 13–21, 02 2012.
- [43] S. Kim, M. A. Nussbaum, and J. L. Gabbard, “Augmented reality “smart glasses” in the workplace: Industry perspectives and challenges for worker safety and health,” *IIE Transactions on Occupational Ergonomics and Human Factors*, vol. 4, pp. 253–258, 07 2016.
- [44] M. Funk, A. Bächler, L. Bächler, T. Kosch, T. Heidenreich, and A. Schmidt, “Working with augmented reality?” *Proceedings of the 10th International*

- Conference on PErvasive Technologies Related to Assistive Environments, 06 2017.
- [45] A. C. Pereira, A. C. Alves, and P. Arezes, “Augmented reality in a lean workplace at smart factories: A case study,” *Applied Sciences*, vol. 13, p. 9120, 01 2023. [Online]. Available: <https://www.mdpi.com/2076-3417/13/16/9120>
 - [46] K. Klinker, M. Wiesche, and H. Krcmar, “Digital transformation in health care: Augmented reality for hands-free service innovation,” *Information Systems Frontiers*, 06 2019.
 - [47] E. Zhu, A. Lilienthal, L. A. Shluzas, I. Masiello, and N. Zary, “Design of mobile augmented reality in health care education: A theory-driven framework,” *JMIR Medical Education*, vol. 1, p. e10, 09 2015.
 - [48] L. Solbiati, N. Gennaro, and R. Muglia, “Augmented reality: From video games to medical clinical practice,” *CardioVascular and Interventional Radiology*, 07 2020.
 - [49] S.-W. Hung, C.-W. Chang, and Y.-C. Ma, “A new reality: Exploring continuance intention to use mobile augmented reality for entertainment purposes,” *Technology in Society*, vol. 67, p. 101757, 11 2021.
 - [50] Z. Yovcheva, D. Buhalis, and C. Gatzidis, “Smartphone augmented reality applications for tourism,” *e-Review of Tourism Research (eRTR)*, vol. 10, p. 63–66, 2012. [Online]. Available: <http://eprints.bournemouth.ac.uk/20219/>
 - [51] C. D. Kounavis, A. E. Kasimati, and E. D. Zamani, “Enhancing the tourism experience through mobile augmented reality: Challenges and prospects,” *International Journal of Engineering Business Management*, vol. 4, p. 10, 01 2012.
 - [52] M. Speicher, B. D. Hall, and M. Nebeling, “What is mixed reality?” *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pp. 1–15, 05 2019.
 - [53] P. Knierim, T. Kosch, M. Hoppe, and A. Schmidt, “Challenges and opportunities of mixed reality systems in education,” *dl.gi.de*, 2018.
 - [54] Mixed reality classroom: Learning from entertainment. Association for Computing Machinery, 2007. [Online]. Available: <https://doi.org/10.1145/1306813.1306833>
 - [55] L. Chen, T. W. Day, W. Tang, and N. W. John, “Recent developments and future challenges in medical mixed reality,” *2017 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 10 2017. [Online]. Available: <https://ieeexplore.ieee.org/abstract/document/8115411>
 - [56] O. M. Tepper, H. L. Rudy, A. Lefkowitz, K. A. Weimer, S. M. Marks, C. S. Stern, and E. S. Garfein, “Mixed reality with hololens: Where virtual reality meets augmented reality in the operating room,” *www.ingentaconnect.com*, 11 2017. [Online]. Available: <https://www.ingentaconnect.com/content/wk/prs/2017/00000140/00000005/art00063>
 - [57] X. Wang and M. A. Schnabel, *Mixed Reality in Architecture, Design, and Construction*. Springer Science & Business Media, 12 2008.
 - [58] P. S. Dunston and X. Wang, “Mixed reality-based visualization interfaces for architecture, engineering, and construction industry,” *Journal of Construction Engineering and Management*, vol. 131, no. 12, pp. 1301–1309, 2005. [Online]. Available: <https://ascelibrary.org/doi/abs/10.1061/%28ASCE%290733-9364%282005%29131%3A12%281301%29>

- [59] “Hololens (1st gen) hardware,” Microsoft Learn, 11 2021. [Online]. Available: <https://learn.microsoft.com/en-us/hololens/hololens1-hardware>
- [60] A. Kipman, “Announcing microsoft hololens development edition open for pre-order, shipping march 30,” Microsoft Devices Blog, 02 2016. [Online]. Available: <https://blogs.windows.com/devices/2016/02/29/announcing-microsoft-hololens-development-edition-open-for-pre-order-shipping-march-30/>
- [61] “Hololens 1st (gen) release notes,” Microsoft Learn, 10 2023. [Online]. Available: <https://learn.microsoft.com/en-us/hololens/hololens1-release-notes>
- [62] Z. Bowden, “The original hololens will no longer receive major os updates,” Windows Central, 07 2019. [Online]. Available: <https://www.windowscentral.com/original-hololens-will-no-longer-receive-major-os-updates>
- [63] J. White, “Microsoft at mwc barcelona: Introducing microsoft hololens 2,” The Official Microsoft Blog, 02 2019. [Online]. Available: <https://blogs.microsoft.com/blog/2019/02/24/microsoft-at-mwc-barcelona-introducing-microsoft-hololens-2/>
- [64] Microsoft, “Hololens 2—pricing and options | microsoft hololens,” Microsoft.com, 2019. [Online]. Available: <https://www.microsoft.com/en-us/hololens/buy>
- [65] scooley, “Hololens 2 hardware,” Microsoft Learn, 03 2023. [Online]. Available: <https://learn.microsoft.com/en-us/hololens/hololens2-hardware>
- [66] R. Seiler, “Microsoft brings windows 11 to hololens 2,” Windows Experience Blog, 04 2023. [Online]. Available: <https://blogs.windows.com/windowsexperience/2023/04/13/microsoft-brings-windows-11-to-hololens-2/>
- [67] Microsoft, “Hololens 2—overview, features, and specs | microsoft hololens,” www.microsoft.com. [Online]. Available: <https://www.microsoft.com/en-us/hololens/hardware#document-experiences>
- [68] keveleigh, “Hand tracking - mrtk 2,” Microsoft Learn, 02 2022. [Online]. Available: <https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/features/input/hand-tracking?view=mrtkunity-2022-05>
- [69] caseymeeckhof, “Direct manipulation with hands - mixed reality,” Microsoft Learn, 09 2022. [Online]. Available: <https://learn.microsoft.com/en-us/windows/mixed-reality/design/direct-manipulation>
- [70] ——, “Point and commit with hands - mixed reality,” Microsoft Learn, 08 2022. [Online]. Available: <https://learn.microsoft.com/en-us/windows/mixed-reality/design/point-and-commit>
- [71] sostel, “Gaze and commit - mixed reality,” Microsoft Learn, 03 2023. [Online]. Available: <https://learn.microsoft.com/en-us/windows/mixed-reality/design/gaze-and-commit>
- [72] shengkait, “Start gesture - mixed reality,” Microsoft Learn, 02 2022. [Online]. Available: <https://learn.microsoft.com/en-us/windows/mixed-reality/design/system-gesture>
- [73] sostel, “Gaze and dwell - mixed reality,” Microsoft Learn, 07 2022. [Online]. Available: <https://learn.microsoft.com/en-us/windows/mixed-reality/design/gaze-and-dwell>
- [74] Sean-Kerawala, “Head-gaze and dwell - mixed reality,” Microsoft Learn, 07 2022. [Online]. Available: <https://learn.microsoft.com/en-us/windows/mixed-reality/design/gaze-and-dwell-head>
- [75] sostel, “Eye-gaze and dwell - mixed reality,” Microsoft Learn, 03 2023.

- [Online]. Available: <https://learn.microsoft.com/en-us/windows/mixed-reality/design/gaze-and-dwell-eyes>
- [76] HakOn, “Voice input - mixed reality,” Microsoft Learn, 03 2022. [Online]. Available: <https://learn.microsoft.com/en-us/windows/mixed-reality/design/voice-input>
- [77] B. Furht, Ed., Mesh, 3D, ser. Encyclopedia of Multimedia. Springer US, 2006, p. 406–407. [Online]. Available: https://doi.org/10.1007/0-387-30038-4_26
- [78] mattzmsft, “Spatial mapping - mixed reality,” Microsoft Learn, 02 2023. [Online]. Available: <https://learn.microsoft.com/en-us/windows/mixed-reality/design/spatial-mapping>
- [79] SzymonS, “Scene understanding - mixed reality,” Microsoft Learn, 09 2022. [Online]. Available: <https://learn.microsoft.com/en-us/windows/mixed-reality/design/scene-understanding>
- [80] dorreneb, “Hololens environment considerations,” Microsoft Learn, 03 2022. [Online]. Available: <https://learn.microsoft.com/en-us/hololens/hololens-environment-considerations>
- [81] kegodin, “Spatial sound overview - mixed reality,” Microsoft Learn, 02 2022. [Online]. Available: <https://learn.microsoft.com/en-us/windows/mixed-reality/design/spatial-sound>
- [82] H. Møller, “Fundamentals of binaural technology,” Applied Acoustics, vol. 36, no. 3, pp. 171–218, 1992. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/0003682X9290046U>
- [83] thetuvix, “Install the tools - mixed reality,” Microsoft Learn, 01 2023. [Online]. Available: <https://learn.microsoft.com/en-us/windows/mixed-reality/develop/install-the-tools>
- [84] qianw211, “Choosing your engine - mixed reality,” Microsoft Learn, 06 2022. [Online]. Available: <https://learn.microsoft.com/en-us/windows/mixed-reality/develop/choosing-an-engine>
- [85] ——, “Unity development for hololens - mixed reality,” Microsoft Learn, 09 2022. [Online]. Available: <https://learn.microsoft.com/en-us/windows/mixed-reality/develop/unity/unity-development-overview>
- [86] U. Technologies, “Gameobjects (unity - manual),” docs.unity3d.com, 04 2023. [Online]. Available: <https://docs.unity3d.com/2020.3/Documentation/Manual/GameObjects.html>
- [87] ——, “Introduction to components (unity - manual),” docs.unity3d.com, 04 2023. [Online]. Available: <https://docs.unity3d.com/2020.3/Documentation/Manual/Components.html>
- [88] ——, “Creating and using scripts (unity - manual),” docs.unity3d.com, 04 2023. [Online]. Available: <https://docs.unity3d.com/2020.3/Documentation/Manual/CreatingAndUsingScripts.html>
- [89] ——, “Unity’s interface (unity - manual),” docs.unity3d.com, 04 2023. [Online]. Available: <https://docs.unity3d.com/2020.3/Documentation/Manual/UsingTheEditor.html>
- [90] ——, “Toolbar (unity - manual),” docs.unity3d.com, 04 2023. [Online]. Available: <https://docs.unity3d.com/2020.3/Documentation/Manual/Toolbar.html>
- [91] ——, “Hierarchy window (unity - manual),” docs.unity3d.com, 04 2023.

- [Online]. Available: <https://docs.unity3d.com/2020.3/Documentation/Manual/Hierarchy.html>
- [92] ——, “Game view (unity - manual),” docs.unity3d.com, 04 2023. [Online]. Available: <https://docs.unity3d.com/2020.3/Documentation/Manual/GameView.html>
- [93] ——, “Scene view (unity - manual),” docs.unity3d.com, 04 2023. [Online]. Available: <https://docs.unity3d.com/2020.3/Documentation/Manual/UsingTheSceneView.html>
- [94] ——, “Inspector window (unity - manual),” docs.unity3d.com, 04 2023. [Online]. Available: <https://docs.unity3d.com/2020.3/Documentation/Manual/UsingTheInspector.html>
- [95] ——, “Project window (unity - manual),” docs.unity3d.com, 04 2023. [Online]. Available: <https://docs.unity3d.com/2020.3/Documentation/Manual/ProjectView.html>
- [96] ——, “Asset workflow (unity - manual),” docs.unity3d.com, 04 2023. [Online]. Available: <https://docs.unity3d.com/2020.3/Documentation/Manual/AssetWorkflow.html>
- [97] ——, “Status bar (unity - manual),” docs.unity3d.com, 04 2023. [Online]. Available: <https://docs.unity3d.com/2020.3/Documentation/Manual/StatusBar.html>
- [98] vtieto, “Using visual studio to deploy and debug - mixed reality,” Microsoft Learn, 09 2022. [Online]. Available: <https://learn.microsoft.com/en-us/windows/mixed-reality/develop/advanced-concepts/using-visual-studio?tabs=hl2>
- [99] polar kev, “Mrkt2-unity developer documentation - mrkt 2,” Microsoft Learn, 12 2022. [Online]. Available: <https://learn.microsoft.com/en-us/windows/mixed-reality/mrkt-unity/mrkt2/?view=mrktunity-2022-05>
- [100] Sean-Kerawala, “Welcome to the mixed reality feature tool - mixed reality,” Microsoft Learn, 12 2022. [Online]. Available: <https://learn.microsoft.com/en-us/windows/mixed-reality/develop/unity/welcome-to-mr-feature-tool>
- [101] hferrone, “Mixed reality documentation - mixed reality,” learn.microsoft.com. [Online]. Available: <https://learn.microsoft.com/en-us/windows/mixed-reality/>
- [102] FlorianBagarMicrosoft, “Holographic remoting player - mixed reality,” learn.microsoft.com, 02 2023. [Online]. Available: <https://learn.microsoft.com/en-us/windows/mixed-reality/develop/native/holographic-remoting-player>
- [103] davidkline ms, “Spatial object mesh observer - mrkt 2,” learn.microsoft.com, 08 2022. [Online]. Available: <https://learn.microsoft.com/en-us/windows/mixed-reality/mrkt-unity/mrkt2/features/spatial-awareness/spatial-object-mesh-observer?view=mrktunity-2022-05>
- [104] ——, “Configuring mesh observers for device - mrkt 2,” learn.microsoft.com, 08 2022. [Online]. Available: <https://learn.microsoft.com/en-us/windows/mixed-reality/mrkt-unity/mrkt2/features/spatial-awareness/configuring-spatial-awareness-mesh-observer?view=mrktunity-2022-05>
- [105] U. Technologies, “Physics (unity - scripting api),” docs.unity3d.com. [Online]. Available: <https://docs.unity3d.com/ScriptReference/Physics.html>
- [106] keveleigh, “Pointers - mrkt 2,” learn.microsoft.com, 08 2022. [Online]. Available: <https://learn.microsoft.com/en-us/windows/mixed-reality/mrkt-unity/>

- mrtk2/features/input/pointers?view=mrtkunity-2022-05
- [107] “Speechinpuhandler class (microsoft.mixedreality.toolkit.input),” learn.microsoft.com. [Online]. Available: <https://learn.microsoft.com/en-us/dotnet/api/microsoft.mixedreality.toolkit.input.speechinpuhandler?view=mixed-reality-toolkit-unity-2020-dotnet-2.8.0>
- [108] P. Bhandari, “Within-subjects design | explanation, approaches, examples,” Scribbr, 03 2021. [Online]. Available: <https://www.scribbr.com/methodology/within-subjects-design/>
- [109] J. Lazar, J. Heidi Feng, and H. Hochheiser, Research Methods in Human Computer Interaction | ScienceDirect, 2017.
- [110] J. Brooke, “Sus: a quick and dirty usability scale,” ResearchGate, 11 1995. [Online]. Available: https://www.researchgate.net/publication/228593520_SUS_A_quick_and_dirty_usability_scale
- [111] S. S. SHAPIRO and M. B. WILK, “An analysis of variance test for normality (complete samples),” Biometrika, vol. 52, pp. 591–611, 12 1965.
- [112] R. Likert, “A technique for the measurement of attitudes.” psycnet.apa.org, 1932.

ПАРАРТНМА А'

Κώδικας Εφαρμογής

Κώδικας A.1: Αρχείο variablesAggregator.cs

```

1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class variablesAggregator : MonoBehaviour
6 {
7     public GameObject voiceCommandSound;
8     // Variables for debugging purposes
9     [Header(" --- General Attributes ---")]
10    public GameObject userPosition;
11    [Tooltip("User's height in meters (m)"), Range(0.0f, 2.5f)]
12    public float userHeight = 0.0f;
13    [Tooltip("User's width in meters (m)"), Range(0.0f, 2.5f)]
14    public float userWidth = 0.0f;
15    [Tooltip("The max distance of the raycasts"), Range(0.0f, 5.0f)]
16    public float maxDistance = 5.0f;
17    public enum CastTypeEnum
18    {
19        RayCast,
20        BoxCast
21    };
22    [Tooltip("The cast type which will be used")]
23    public CastTypeEnum castType;
24    [Tooltip("The list of the raycast boxes")]
25    public GameObject[] raycastBoxes;
26    [Header(" --- Alert Boxes ---")]
27    [Tooltip("The common alert box")]
28    public GameObject commonAlertBox;
29    public enum AlertBoxTypeEnum
30    {
31        BoxSpecific,
32        Common
33    }
34    public AlertBoxTypeEnum alertBoxType;
35    [Tooltip("The alert box is placed at the height level of the eyes
            instead of the height level of the hitPoint")]
36    public bool placeAlertBoxAtEyeLevel = true;
37    public enum CastModeEnum
38    {
39        ContinuousMode,
40        ScanMode,
41        HandsMode,
42        StopMode
43    }
44    [Header(" --- Cast Modes ---")]
45    [Tooltip("The cast mode which will be used")]
46    public CastModeEnum enabledModeGlobal;
47
48    [Header(" --- Scan/Continuous Mode Attributes ---")]
49    [Tooltip("The number of cast boxes placed in each row"), Range(1.0f,
            100.0f)]
50    public int boxesPerRow=3;
51    [Tooltip("The number of cast boxes placed in each column"), Range(1.0f,
            100.0f)]
52    public int numberOfRows=3;

```

```

53
54     [Header(" --- Hand Mode Attributes ---")]
55     [Tooltip("The list of the hand alert boxes")]
56     public GameObject[] handAlertBoxes;
57     public enum lightPositionEnum
58     {
59         Up,
60         Right,
61         Down,
62         Left
63     }
64     [Header(" --- Flashing warning attributes ---")]
65     [Tooltip("Enables the flash effect")]
66     public bool activateFlash = false;
67     [Tooltip("The gameObjects which have the flashing effect")]
68     public GameObject[] flashingImages;
69     [Tooltip("The color of the flash for dangerous distance"), ColorUsage(
70         false)]
71     public Color dangerColor;
72     [Tooltip("The color of the flash for warning"), ColorUsage(false)]
73     public Color warningColor;
74     [Tooltip("The color of the flash for safe distance"), ColorUsage(false)
75         ]
76     public Color safeColor;
77
78     [Header(" --- Rumble attributes ---")]
79     [Tooltip("Enbales the rumble effect")]
80     public bool activateRumble;
81     public enum RumbleModes
82     {
83         Constant,
84         Linear,
85         Pulse
86     }
87     #if UNITY_EDITOR
88         [Help("The logic for Constant and Linear rumble was never
89             implemented", UnityEditor.MessageType.Warning)]
90     #endif
91     public RumbleModes rumbleModeSlected;
92     [Header(" --- Pitch Change attributes ---")]
93     [Tooltip("The pitch of the alert sound changes based on the distance of
94             the obstacle from the user")]
95     public bool changeAlertPitch = true;
96     [Tooltip("The value of the pitch for a safe distance"), Range(1.0f, 3.0
97         f)]
98     public float pitchChangeSafeDist = 1.0f;
99     [Tooltip("The value of the pitch for a cautious distance"), Range(1.0f,
100         3.0f)]
101    public float pitchChangeWarningDist = 1.15f;
102    [Tooltip("The value of the pitch for a dangerous distance"), Range(1.0f
103        , 3.0f)]
104    public float pitchChangeDangerousDist = 1.5f;
105    [Tooltip("The value of the pitch for a dangerous distance"), Range(1.0f
106        , 3.0f)]
107    public float pitchChangeEndDist = 2.5f;
108    [Header(" --- Flashing, Rumble & Pitch Change common attributes ---")]
109    [Tooltip("The maximum distance considered safe"), Min(0.0f)]
110    public float distanceThreadToActivate = 2.0f;
111    [Tooltip("The maximum distance for which a user should be cautious"),
112        Min(0.0f)]
113    public float warningIndicatorThread = 1.5f;

```

```

105     [Tooltip("The maximum distance considered dangerous"), Min(0.0f)]
106     public float dangerIndicatorThread = 0.5f;
107
108 }
```

Κώδικας A.2: Αρχείο initiators/initCastAndBoxesHandler.cs

```

1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class initCastAndBoxesHandler : MonoBehaviour
6 {
7     public GameObject variableAggregatorObject;
8 }
```

Κώδικας A.3: Αρχείο initiators/initBoxes.cs

```

1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.InputSystem;
5 using UnityEngine.Serialization;
6
7 [RequireComponent(typeof(initCastAndBoxesHandler)), RequireComponent(typeof(
8     initRumbleHandler)), RequireComponent(typeof(initFlashHandler))]
9 public class initBoxes : MonoBehaviour {
10     [HideInInspector()]
11     public float userHeight = 0.0f;
12     private float userWidth = 0.0f;
13     private variablesAggregator variableAggInstance;
14     private GameObject[] raycastBoxes;
15     private int boxesPerRow;
16     private int numberOfRows;
17
18     void Start() {
19         variableAggInstance = this.GetComponent<initCastAndBoxesHandler>();
20         variableAggInstance = variableAggInstance.GetComponent<variablesAggregator>();
21         userHeight = variableAggInstance.userHeight;
22         userWidth = variableAggInstance.userWidth;
23         raycastBoxes = variableAggInstance.raycastBoxes;
24         boxesPerRow = variableAggInstance.boxesPerRow;
25         numberOfRows = variableAggInstance.numberOfRows;
26
27         if (userHeight <= 0) {
28             userHeight = 1.8f;
29         }
30
31         initializePosition();
32     }
33
34     /// <summary>
35     /// Based on the mode that has been selected, the cast boxes are placed
36     /// and activated accordingly
37     /// </summary>
38     public void initializePosition() {
39         GameObject[] castBoxesToUse;
40         switch (variableAggInstance.enabledModeGlobal) {
41             // Scan Mode
42             case variablesAggregator.CastModeEnum.ScanMode:
```

```

40             this.positionAndScaleBoxes(raycastBoxes, 3, 3, userHeight);
41             break;
42         // Continuous Mode
43         case variablesAggregator.CastModeEnum.ContinuousMode:
44             castBoxesToUse = this.findCastBoxesWithContinuousEnabled(
45                 raycastBoxes);
46             this.positionAndScaleBoxes(castBoxesToUse, boxesPerRow,
47                 numberOfRows, userHeight);
48             break;
49         case variablesAggregator.CastModeEnum.StopMode:
50             castBoxesToUse = this.findCastBoxesWithContinuousEnabled(
51                 raycastBoxes);
52             this.positionAndScaleBoxes(castBoxesToUse, boxesPerRow,
53                 numberOfRows, userHeight);
54             break;
55     }
56 }
57
58 /**
59 * A method which places the castBoxes and their alert boxes in the
60 * correct position based on the parameters provided.
61 */
62 /**
63 * An array of the castboxes</param>
64 * The number of castBoxes which will be
65 * placed in each row</param>
66 * The number of rows, in which the cast
67 * boxes will be placed</param>
68 * The height of the user in meters</param>
69 */
70 public void positionAndScaleBoxes
71 (
72     GameObject[] raycstBoxesToPosition,
73     int boxesPerRow,
74     int numberOfRows,
75     float userHeight
76 ) {
77     float boxLength = userWidth / boxesPerRow;
78     float boxHeight = userHeight / numberOfRows;
79
80     float boxPlacementX = -boxLength * (boxesPerRow - 1);
81     float boxPlacementY = 0.0f;
82     for (int i = 0; i < raycstBoxesToPosition.Length; i++) {
83         if (i % boxesPerRow == 0) {
84             boxPlacementX = -boxLength * (boxesPerRow - 1);
85
86             if (i != 0) boxPlacementY -= boxHeight;
87         }
88
89         raycstBoxesToPosition[i].transform.position = new Vector3(
90             boxPlacementX, boxPlacementY, -0.1f);
91         raycstBoxesToPosition[i].transform.localScale = new Vector3(
92             boxLength, boxHeight, boxLength);
93
94         raycstBoxesToPosition[i].GetComponent<initSingleBox>().alertBox
95             .transform.localScale = new Vector3(boxLength, boxHeight,
96             0.1f);
97
98         raycstBoxesToPosition[i].SetActive(true);
99
100        boxPlacementX += boxLength * (boxesPerRow - 1);
101    }
102 }
```

```

89     }
90
91     /// <summary>
92     /// In an array of RaycastHit (<paramref name="hitPointsList"/>), it
93     /// locates the index of the hitPoint, whose distance is closer to the
94     /// user.
95     /// </summary>
96     /// <param name="hitPointsList">An array of hitPoints</param>
97     /// <returns>It returns the index of the hitPoint closer to the user</
98     /// returns>
99     public int findClosestHitPointIndex(RaycastHit[] hitPointsList) {
100         int closestHitPointIndex = -1;
101         float minDistance = 999.0f;
102         List<Vector3> relPosList = new List<Vector3>();
103         for (int i = 0; i < hitPointsList.Length; i++) {
104             if (hitPointsList[i].distance != 0.0f && hitPointsList[i].
105                 collider != null) {
106                 Vector3 relativePosition = variableAggInstance.userPosition
107                     .transform.InverseTransformPoint(hitPointsList[i].point)
108                     ;
109                 relPosList.Add(relativePosition);
110                 if (hitPointsList[i].distance < minDistance) {
111                     minDistance = hitPointsList[i].distance;
112                     closestHitPointIndex = i;
113                 }
114             }
115         }
116         return closestHitPointIndex;
117     }
118
119     /// <summary>
120     /// Finds the castBoxes for which the continuous mode has been
121     /// enabled..
122     /// </summary>
123     /// <param name="allCastBoxes">An array of all the castBoxes in the
124     /// scene</param>
125     /// <returns>The castBoxes that meet the criteria</returns>
126     public GameObject[] findCastBoxesWithContinuousEnabled(GameObject[]
127         allCastBoxes) {
128         List<GameObject> castBoxesWithContinous = new List<GameObject>();
129
130         for (int i = 0; i < allCastBoxes.Length; i++) {
131             if (allCastBoxes[i].GetComponent<initSingleBox>().
132                 continuousRaycast == true) {
133                 castBoxesWithContinous.Add(allCastBoxes[i]);
134             }
135         }
136
137         return castBoxesWithContinous.ToArray();
138     }
139
140     /// <summary>
141     /// Places the alert box of a <paramref name="castBox"/> in the
142     /// position of the <paramref name="hitPoint"/>.
143     /// </summary>
144     /// <param name="castBox">The castBox whose alert box will be
145     /// transformed</param>
146     /// <param name="hitPoint">The hitPoint to which the alert box will be
147     /// placed</param>
148     public void placeAlertBox(GameObject castBox, RaycastHit hitPoint) {

```

```

137         castBox.GetComponent<initSingleBox>().alertBox.SetActive(true);
138         if (!castBox.GetComponent<initSingleBox>().alertBox.GetComponent<
139             AudioSource>().isPlaying)
140             castBox.GetComponent<initSingleBox>().alertBox.GetComponent<
141                 AudioSource>().Play();
140         castBox.GetComponent<initSingleBox>().alertBox.transform.position =
141             hitPoint.point;
141         castBox.GetComponent<initSingleBox>().alertBox.transform.rotation =
142             castBox.transform.rotation;
142     }
143
144     /// <summary>
145     /// Places an <paramref name="alertView"/> in the position of a <
146     /// paramref name="hitPoint"/>.
146     /// </summary>
147     /// <param name="alertView">The alert box which will be transformed</
148     /// param>
148     /// <param name="castBox">The castBox whose rotation will be used</
149     /// param>
149     /// <param name="hitPoint">The hitPoint to which the alert box will be
150     placed</param>
150     public void placeAlertBox(GameObject alertBox, GameObject castBox,
151         RaycastHit hitPoint) {
151         alertBox.SetActive(true);
152         var alertTrans = alertBox.transform;
153         if (!alertView.GetComponent<AudioSource>().isPlaying)
154             alertBox.GetComponent<AudioSource>().Play();
155
156         if (variableAggInstance.enabledModeGlobal == variablesAggregator.
157             CastModeEnum.ScanMode) {
157             alertBox.GetComponent<AudioSource>().loop = true;
158         } else {
159             alertBox.GetComponent<AudioSource>().loop = false;
160         }
161
162         var hitPointVec = hitPoint.point;
163         var userPositionTrans = variableAggInstance.userPosition.transform;
164         if (variableAggInstance.placeAlertBoxAtEyeLevel) {
165             alertTrans.position = new Vector3(hitPointVec.x,
166                 userPositionTrans.position.y, hitPointVec.z);
166         } else {
167             alertTrans.position = hitPointVec;
168         }
169         alertBox.transform.rotation = castBox.transform.rotation;
170     }
171
172     /// <summary>
173     /// Deactivates all the alert boxes, based on an array of castBoxes
174     /// </summary>
175     /// <param name="castBoxesArray">The array of castBoxes, whose alert
176     /// boxes will be deactivated</param>
176     public void deactivateAlertBox(GameObject[] castBoxesArray) {
177         for (int i = 0; i < castBoxesArray.Length; i++) {
178             if (castBoxesArray[i].GetComponent<initSingleBox>().alertView.
179                 GetComponent<AudioSource>().isPlaying)
180                 castBoxesArray[i].GetComponent<initSingleBox>().alertView.
181                     GetComponent<AudioSource>().Pause();
180         }
181     }
182
183     /// <summary>

```

```
184     /// Deactivates an alert box
185     /// </summary>
186     /// <param name="alertView">The alert box which will be deactivated</param>
187     public void deactivateAlertBox(GameObject alertBox) {
188         if (alertView.GetComponent<AudioSource>().isPlaying)
189             if (alertView.GetComponent<AudioSource>().loop)
190                 alertBox.GetComponent<AudioSource>().Pause();
191     }
192
193     /// <summary>
194     /// Deactivates all the alert boxes, which have been defined and are present in the scene
195     /// </summary>
196     public void deactivateAllAlertBoxes() {
197         deactivateAlertBox(variableAggInstance.raycastBoxes);
198         deactivateAlertBox(variableAggInstance.commonAlertBox);
199         for (int i = 0; i < variableAggInstance.handAlertBoxes.Length; i++)
200             {
201                 deactivateAlertBox(variableAggInstance.handAlertBoxes[i]);
202             }
203     }
204
205     /// <summary>
206     /// Changes the pitch of the sound, based on the distance of the user from the obstacle.
207     /// Three thresholds have been defined for different pitch levels
208     /// </summary>
209     /// <param name="alertView">The alert box containing an AudioSource</param>
210     /// <param name="hitPoint">The position of the obstacle</param>
211     public void changeAlertPitch(GameObject alertBox, RaycastHit hitPoint)
212     {
213         changeAlertPitch(alertBox, hitPoint.distance);
214     }
215
216     /// <summary>
217     /// Changes the pitch of the sound, based on the distance of the user from the obstacle.
218     /// Three thresholds have been defined for different pitch levels
219     /// <param name="alertView">The alert box containing an AudioSource</param>
220     /// <param name="obsDistance">The distance of the obstacle from the user</param>
221     public void changeAlertPitch(GameObject alertBox, float obsDistance) {
222         if (obsDistance > variableAggInstance.maxDistance
223             || (obsDistance <= variableAggInstance.maxDistance &&
224                 obsDistance > variableAggInstance.warningIndicatorThread)) {
225             alertBox.GetComponent<AudioSource>().pitch =
226                 variableAggInstance.pitchChangeSafeDist;
227         } else if (obsDistance <= variableAggInstance.
228             warningIndicatorThread && obsDistance > variableAggInstance.
229             dangerIndicatorThread) {
230             alertBox.GetComponent<AudioSource>().pitch =
231                 variableAggInstance.pitchChangeWarningDist;
232         } else if (obsDistance <= variableAggInstance.dangerIndicatorThread
233             && obsDistance > 0.2f) {
234             alertBox.GetComponent<AudioSource>().pitch =
235                 variableAggInstance.pitchChangeDangerousDist;
236         } else {
237             alertBox.GetComponent<AudioSource>().pitch =
238                 variableAggInstance.pitchChangeDefaultDist;
239         }
240     }
241 }
```

```

229         alertBox.GetComponent<AudioSource>().pitch =
230             variableAggInstance.pitchChangeEndDist;
231     }
232
233     /// <summary>
234     /// Set the pitch of the sound to orginal value
235     /// </summary>
236     /// <param name="alertBox">The alert box containing an AudioSource</
237     param>
238     public void changeAlertPitch(GameObject alertBox) {
239         alertBox.GetComponent<AudioSource>().pitch = variableAggInstance.
240             pitchChangeSafeDist;
241     }
242
243     /// <summary>
244     /// Disables the alert boxes, the flashes, the rumble and the pitch
245     /// change.
246     /// Moreover, positions the cast boxes based on the mode that has been
247     /// activated
248     /// </summary>
249     public void resetStatus() {
250         variableAggInstance.voiceCommandSound.GetComponent<AudioSource>().
251             Play();
252         this.GetComponent<initBoxes>().deactivateAllAlertBoxes();
253         this.GetComponent<initFlashHandler>().StopFlashes();
254         this.GetComponent<initRumbleHandler>().StopRumble();
255         this.GetComponent<initBoxes>().changeAlertPitch(variableAggInstance
256             .commonAlertBox);
257         // this.GetComponent<initBoxes>().initializePosition();
258     }
259 }

```

Κώδικας A.4: Αρχείο initiators/initSingleBox.cs

```

1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class initSingleBox : MonoBehaviour
6 {
7     [Tooltip("The object which will operate as an alert box for the
8         specific raycast.")]
9     public GameObject alertBox;
10    [Tooltip("If it is set to 'True', rays will be casted in every frame."]
11    ]
12    public bool continuousRaycast = false;
13 }

```

Κώδικας A.5: Αρχείο initiators/initFlashHandler.cs

```

1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.UI;
5
6 [RequireComponent(typeof(initCastAndBoxesHandler))]
7 public class initFlashHandler : MonoBehaviour {
8     private variablesAggregator variableAggInstance;
9

```

```
1 void Start() {
2     variableAggInstance = this.GetComponent<initCastAndBoxesHandler>();
3         variableAggregatorObject.GetComponent<variablesAggregator>();
4 }
5
6 /// <summary>
7 /// Disables the flash effect of all the gameObjects
8 /// </summary>
9 public void StopFlashes() {
10     GameObject[] flashImagesArray = variableAggInstance.flashingImages;
11
12     for (int i = 0; i < flashImagesArray.Length; i++) {
13         flashImagesArray[i].GetComponent<flashingLight>().StopFlash();
14     }
15 }
16
17 /// <summary>
18 /// Enables the flash effect for specific gameObjects, based on the
19 /// position of the hitPoint
20 /// </summary>
21 /// <param name="lightPositions">The position of the hitPoint relative
22 /// to the user</param>
23 /// <param name="flashColor">The color of the flash</param>
24 public void StartMultipleFlashes(variablesAggregator.lightPositionEnum
25     [] lightPositions, Color flashColor) {
26     GameObject[] flashImagesArray = variableAggInstance.flashingImages;
27
28     for (int i = 0; i < flashImagesArray.Length; i++) {
29         bool shouldBeEnabled = false;
30
31         for (int j = 0; j < lightPositions.Length; j++) {
32             if (flashImagesArray[i].GetComponent<flashingLight>().
33                 lightPosition == lightPositions[j]) shouldBeEnabled =
34                 true;
35         }
36
37         if (shouldBeEnabled) {
38             flashImagesArray[i].GetComponent<flashingLight>().
39                 StartFlash(1, 0.75f, flashColor);
40         } else {
41             flashImagesArray[i].GetComponent<flashingLight>().StopFlash();
42         }
43     }
44 }
45
46 /// <summary>
47 /// Detects the position of the hitPoint relative to the user
48 /// </summary>
49 /// <param name="hitPoint">The closest point of the mesh to the user</
50 /// param>
51 /// <returns>The position of the hitPoint (<see cref="
52 /// variablesAggregator.lightPositionEnum"/>)</returns>
53 public variablesAggregator.lightPositionEnum[] FindHitPointPosition(
54     RaycastHit hitPoint) {
55     List<variablesAggregator.lightPositionEnum> lightPositionToReturn =
56         new List<variablesAggregator.lightPositionEnum>();
57     Vector3 relativePosition = variableAggInstance.userPosition.
58         transform.InverseTransformPoint(hitPoint.point);
59
60     if (relativePosition.x > 0.3f) {
```

```

58         lightPositionToreturn.Add(variablesAggregator.lightPositionEnum
59             .Right);
60     } else if (relativePosition.x < -0.3f) {
61         lightPositionToreturn.Add(variablesAggregator.lightPositionEnum
62             .Left);
63     }
64
65     if (relativePosition.y > 0) {
66         lightPositionToreturn.Add(variablesAggregator.lightPositionEnum
67             .Up);
68     } else {
69         lightPositionToreturn.Add(variablesAggregator.lightPositionEnum
70             .Down);
71     }
72
73     return lightPositionToreturn.ToArray();
74 }
75
76 /// <summary>
77 /// Based on the distance, it returns the color of the flash which will
78 /// be applied to the gameObjects.
79 /// There are three possible colors to be applied:
80 /// <list type="bullet">
81 ///     <item>
82 ///         <description>Red: If the distance of the hitPoint is less
83 ///             than <see cref="variablesAggregator.dangerIndicatorThread">
84 ///                 dangerIndicatorThread</see></description>
85 ///         </item>
86 ///         <item>
87 ///             <description>
88 ///                 Yellow: If the distance of the hitPoint is more than <
89 ///                     see cref="variablesAggregator.dangerIndicatorThread">
90 ///                         dangerIndicatorThread</see>,
91 ///                     but less than <see cref="variablesAggregator.
92 ///                         warningIndicatorThread">warningIndicatorThread</see>
93 ///             </description>
94 ///         </item>
95 ///         <item>
96 ///             <description>
97 ///                 Yellow: If the distance of the hitPoint is more than <
98 ///                     see cref="variablesAggregator.warningIndicatorThread">
99 ///                         warningIndicatorThread</see>,
100 ///                     but less than <see cref="variablesAggregator.
101 ///                         distanceThreadToActivate">distanceThreadToActivate</see>
102 ///             </description>
103 ///         </item>
104     </list>
105     </summary>
106     <param name="distance"></param>
107     <returns></returns>
108     public Color FindColorBasedOnDistance(float distance) {
109         Color dangerColor = variableAggInstance.dangerColor;
110         Color warningColor = Color.yellow;
111         Color safeColor = Color.green;
112
113         if (distance >= 0 && distance <= variableAggInstance.
114             dangerIndicatorThread) {
115             return dangerColor;
116         } else if (distance > variableAggInstance.dangerIndicatorThread &&
117             distance <= variableAggInstance.warningIndicatorThread) {
118             return warningColor;
119         }
120     }
121 }
```

```

104         } else if (distance > variableAggInstance.warningIndicatorThread &&
105             distance < variableAggInstance.distanceThreadToActivate) {
106             return safeColor;
107         } else {
108             return new Color(0, 0, 0, 0);
109         }
110     }

```

Κώδικας A.6: Αρχείο initiators/initRumbleHandler.cs

```

1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.InputSystem;
5
6 [RequireComponent(typeof(initCastAndBoxesHandler))]
7 public class initRumbleHandler : MonoBehaviour {
8
9     private variablesAggregator variableAggInstance;
10    private variablesAggregator.RumbleModes rumbleModeSelected;
11    private Gamepad gamepad = null;
12    private float pulseDuration = 0.0f;
13    private float stopDuration = 0.0f;
14    private float lowA = 0.0f;
15    private float highA = 0.0f;
16    private float rumbleStep = 0.0f;
17    private float stopStep = 0.0f;
18    private bool isMotorActive = false;
19    private bool isRumbleCurrentlyActive = false;
20
21    void Start() {
22        variableAggInstance = this.GetComponent<initCastAndBoxesHandler>().
23            variableAgggregatorObject.GetComponent<variablesAggregator>();
24        rumbleModeSelected = variableAggInstance.rumbleModeSelected;
25        gamepad = GetGamepad();
26    }
27
28    /// <summary>
29    /// Detects and returns the current gamepad
30    /// </summary>
31    /// <returns>The current gamepad</returns>
32    public Gamepad GetGamepad() {
33        return Gamepad.current;
34    }
35
36    /// <summary>
37    /// Disables the rumble effect in the current gamepad
38    /// </summary>
39    public void StopRumble() {
40        Gamepad gamepad = GetGamepad();
41
42        if (gamepad != null && isRumbleCurrentlyActive) {
43            isRumbleCurrentlyActive = false;
44            gamepad.SetMotorSpeeds(0, 0);
45            lowA = 0;
46            highA = 0;
47            rumbleStep = 0;
48            stopStep = 0;
49            pulseDuration = 0;
50            stopDuration = 0;

```

```
51     }
52 }
53
54 /// <summary>
55 /// Activates the rumble with a pulse effect in the current gamepad
56 /// </summary>
57 /// <param name="low">Speed of low-frequency motor</param>
58 /// <param name="high">Speed of high-frequency motor</param>
59 /// <param name="stopTime">The time between each rumble (in seconds)</param>
60 /// <param name="burstTime">The duration of the rumble (in seconds)</param>
61 public void RumblePulse(float low, float high, float stopTime = 0.5f,
62                         float burstTime = 0.5f) {
63     if (rumbleModeSelected == variablesAggregator.RumbleModes.Pulse) {
64         isRumbleCurrentlyActive = true;
65         lowA = low;
66         highA = high;
67         rumbleStep = burstTime;
68         stopStep = stopTime;
69         if (pulseDuration == 0.0f) {
70             pulseDuration = Time.time + burstTime;
71             isMotorActive = true;
72             if (gamepad != null) gamepad.SetMotorSpeeds(lowA, highA);
73         }
74     }
75
76 /// <summary>
77 /// Based on the distance of the hitPoint, the speed of the motors are
78 /// calculated.
79 /// The speed is inversely proportional of the distance of the user from
80 /// the hitPoint
81 /// </summary>
82 /// <param name="distance"></param>
83 /// <returns></returns>
84 public float FindMotorsLowAndHigh (float distance) {
85     return 1.0f - Mathf.Floor((distance / variableAggInstance.
86                             maxDistance) * 100.0f) / 100.0f;
87 }
88
89 /// <summary>
90 /// Based on the distance of the hitPoint, it calculates the duration
91 /// between each rumble
92 /// </summary>
93 /// <param name="hitPointDistance">The distance of the hitPoint from
94 /// the user</param>
95 /// <returns>The stop time</returns>
96 public float GetStopFrequency (float hitPointDistance) {
97     float stopStep = 0.0f;
98
99     if (0 <= hitPointDistance && hitPointDistance <=
100         variableAggInstance.dangerIndicatorThread) {
101         stopStep = 0.25f;
102     } else if (variableAggInstance.dangerIndicatorThread <
103                hitPointDistance && hitPointDistance <= variableAggInstance.
104                warningIndicatorThread) {
105         stopStep = 1.0f;
106     } else if (variableAggInstance.warningIndicatorThread <
107                hitPointDistance && hitPointDistance <= variableAggInstance.
108                distanceThreadToActivate) {
```

```

98             stopStep = 2.0f;
99         }
100
101     return stopStep;
102 }
103
104 /// <summary>
105 /// Applies the speed to the correct motor based on the position of the
106 /// hitPoint relative to the user
107 /// </summary>
108 /// <param name="hitPoint">The closest point of an obstacle relative to
109 /// the user</param>
110 /// <param name="motorSpeed">The speed, which will be applied to one of
111 /// the motors</param>
112 /// <returns>A dictionary which dictates what the speed of each motor
113 /// should be</returns>
114 public Dictionary<string, float> GetSpeedOfEachMotor(RaycastHit
115     hitPoint, float motorSpeed) {
116     Vector3 relativePosition = variableAggInstance.userPosition.
117         transform.InverseTransformPoint(hitPoint.point);
118     Dictionary<string, float> motorToEnable = new Dictionary<string,
119         float>()
120     {
121         {"low", 0.0f},
122         {"high", 0.0f}
123     };
124
125     if (relativePosition.x >= 0.0f) {
126         motorToEnable["high"] = motorSpeed;
127     }
128     if (relativePosition.x <= 0.0f) {
129         motorToEnable["low"] = motorSpeed;
130     }
131
132     return motorToEnable;
133 }
134
135 /// <summary>
136 /// Toggles the activation of the rumble effect based on the button
137 /// pressed by the user on the gamepad
138 /// </summary>
139 /// <remarks>
140 /// <para>If the user presses A on the gamepad, the rumble effect will
141 /// be activated (if it was deactivated before).</para>
142 /// <para>If the user presses B on the gamepad, the rumble effect will
143 /// be deactivated (if it was activated before).</para>
144 /// </remarks>
145 private void rumbleToggleGamepad() {
146     if (gamepad.bButton.wasPressedThisFrame && variableAggInstance.
147         activateRumble) {
148         variableAggInstance.activateRumble = false;
149         StopRumble();
150     } else if (gamepad.aButton.wasPressedThisFrame && !
151         variableAggInstance.activateRumble) {
152         variableAggInstance.activateRumble = true;
153     }
154 }
155
156 private void Update() {
157     gamepad = GetGamepad();
158     if (gamepad == null) return;

```

```

147         rumbleToggleGamepad();
148
149         if (!variableAggInstance.activateRumble) return;
150
151         if (!isRumbleCurrentlyActive) return;
152
153         switch (rumbleModeSelected) {
154             case variablesAggregator.RumbleModes.Pulse:
155                 if (isMotorActive && Time.time > pulseDuration) {
156                     isMotorActive = !isMotorActive;
157                     stopDuration = Time.time + stopStep;
158                     gamepad.SetMotorSpeeds(0, 0);
159                 } else if (!isMotorActive && Time.time > stopDuration) {
160                     isMotorActive = !isMotorActive;
161                     pulseDuration = Time.time + rumbleStep; // Update
162                     pulseDuration
163                     gamepad.SetMotorSpeeds(lowA, highA);
164
165                 }
166                 break;
167             }
168         }
169     }

```

Kώδικας A.7: Αρχείο castModes/scanCast.cs

```

1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 [RequireComponent(typeof(initCastAndBoxesHandler)), RequireComponent(typeof(
6     initBoxes))]
7 public class scanCast : MonoBehaviour {
8     private variablesAggregator variableAggInstance;
9
10    void Start() {
11        variableAggInstance = this.GetComponent<initCastAndBoxesHandler>().
12            variableAggInstance.GetComponent<variablesAggregator>();
13    }
14
15    public void EnableScanMode() {
16
17        checkCurrentMode();
18
19        GameObject[] allCastBoxes = variableAggInstance.raycastBoxes;
20        List<RaycastHit> castHitPointsList = new List<RaycastHit>();
21
22        for (int i = 0; i < allCastBoxes.Length; i++) {
23            RaycastHit hitPoint = allCastBoxes[i].GetComponent<
24                singleRaycast>().SingleRaycastFunc(variableAggInstance);
25            castHitPointsList.Add(hitPoint);
26        }
27
28        //ToArray doesn't change the order of the elements
29        int hitPointIndex = this.GetComponent<initBoxes>().
30            findClosestHitPointIndex(castHitPointsList.ToArray());
31        if (hitPointIndex != -1) {
32            switch (variableAggInstance.alertBoxType) {
33                case variablesAggregator.AlertBoxTypeEnum.BoxSpecific:
34                    // Use the alert box of each castBox
35            }
36        }
37    }
38
39    void checkCurrentMode() {
40        if (variableAggInstance.currentMode == Mode.Scan) {
41            enableScanMode();
42        }
43    }
44
45    void enableScanMode() {
46        variableAggInstance.currentMode = Mode.Scan;
47        variableAggInstance.raycastBoxes = GetCastBoxes();
48    }
49
50    void GetCastBoxes() {
51        List<GameObject> castBoxes = new List<GameObject>();
52
53        foreach (var box in initBoxes) {
54            if (box.activeSelf) {
55                castBoxes.Add(box.gameObject);
56            }
57        }
58
59        return castBoxes;
60    }
61
62    void findClosestHitPointIndex(GameObject[] raycastBoxes) {
63        int closestIndex = -1;
64        float closestDistance = float.MaxValue;
65
66        for (int i = 0; i < raycastBoxes.Length; i++) {
67            RaycastHit hitPoint = raycastBoxes[i].GetComponent<
68                singleRaycast>().SingleRaycastFunc(variableAggInstance);
69
70            if (hitPoint.distance < closestDistance) {
71                closestIndex = i;
72                closestDistance = hitPoint.distance;
73            }
74        }
75
76        return closestIndex;
77    }
78
79    void findClosestHitPointIndex(List<RaycastHit> castHitPointsList) {
80        int closestIndex = -1;
81        float closestDistance = float.MaxValue;
82
83        for (int i = 0; i < castHitPointsList.Count; i++) {
84            if (castHitPointsList[i].distance < closestDistance) {
85                closestIndex = i;
86                closestDistance = castHitPointsList[i].distance;
87            }
88        }
89
90        return closestIndex;
91    }
92
93    void findClosestHitPointIndex(int[] raycastBoxes) {
94        int closestIndex = -1;
95        float closestDistance = float.MaxValue;
96
97        for (int i = 0; i < raycastBoxes.Length; i++) {
98            if (raycastBoxes[i] < closestDistance) {
99                closestIndex = i;
100               closestDistance = raycastBoxes[i];
101           }
102       }
103
104       return closestIndex;
105   }
106
107   void findClosestHitPointIndex(GameObject raycastBox) {
108       int closestIndex = -1;
109       float closestDistance = float.MaxValue;
110
111       for (int i = 0; i < raycastBox.Length; i++) {
112           if (raycastBox[i] < closestDistance) {
113               closestIndex = i;
114               closestDistance = raycastBox[i];
115           }
116       }
117
118       return closestIndex;
119   }
120
121   void findClosestHitPointIndex(RaycastHit raycastBox) {
122       int closestIndex = -1;
123       float closestDistance = float.MaxValue;
124
125       for (int i = 0; i < raycastBox.Length; i++) {
126           if (raycastBox[i].distance < closestDistance) {
127               closestIndex = i;
128               closestDistance = raycastBox[i].distance;
129           }
130       }
131
132       return closestIndex;
133   }
134
135   void findClosestHitPointIndex(RaycastHit[] raycastBox) {
136       int closestIndex = -1;
137       float closestDistance = float.MaxValue;
138
139       for (int i = 0; i < raycastBox.Length; i++) {
140           if (raycastBox[i].distance < closestDistance) {
141               closestIndex = i;
142               closestDistance = raycastBox[i].distance;
143           }
144       }
145
146       return closestIndex;
147   }
148
149   void findClosestHitPointIndex(RaycastHit raycastBox) {
150       int closestIndex = -1;
151       float closestDistance = float.MaxValue;
152
153       for (int i = 0; i < raycastBox.Length; i++) {
154           if (raycastBox[i].distance < closestDistance) {
155               closestIndex = i;
156               closestDistance = raycastBox[i].distance;
157           }
158       }
159
160       return closestIndex;
161   }
162
163   void findClosestHitPointIndex(RaycastHit raycastBox) {
164       int closestIndex = -1;
165       float closestDistance = float.MaxValue;
166
167       for (int i = 0; i < raycastBox.Length; i++) {
168           if (raycastBox[i].distance < closestDistance) {
169               closestIndex = i;
170               closestDistance = raycastBox[i].distance;
171           }
172       }
173
174       return closestIndex;
175   }
176
177   void findClosestHitPointIndex(RaycastHit raycastBox) {
178       int closestIndex = -1;
179       float closestDistance = float.MaxValue;
180
181       for (int i = 0; i < raycastBox.Length; i++) {
182           if (raycastBox[i].distance < closestDistance) {
183               closestIndex = i;
184               closestDistance = raycastBox[i].distance;
185           }
186       }
187
188       return closestIndex;
189   }
190
191   void findClosestHitPointIndex(RaycastHit raycastBox) {
192       int closestIndex = -1;
193       float closestDistance = float.MaxValue;
194
195       for (int i = 0; i < raycastBox.Length; i++) {
196           if (raycastBox[i].distance < closestDistance) {
197               closestIndex = i;
198               closestDistance = raycastBox[i].distance;
199           }
200       }
201
202       return closestIndex;
203   }
204
205   void findClosestHitPointIndex(RaycastHit raycastBox) {
206       int closestIndex = -1;
207       float closestDistance = float.MaxValue;
208
209       for (int i = 0; i < raycastBox.Length; i++) {
210           if (raycastBox[i].distance < closestDistance) {
211               closestIndex = i;
212               closestDistance = raycastBox[i].distance;
213           }
214       }
215
216       return closestIndex;
217   }
218
219   void findClosestHitPointIndex(RaycastHit raycastBox) {
220       int closestIndex = -1;
221       float closestDistance = float.MaxValue;
222
223       for (int i = 0; i < raycastBox.Length; i++) {
224           if (raycastBox[i].distance < closestDistance) {
225               closestIndex = i;
226               closestDistance = raycastBox[i].distance;
227           }
228       }
229
230       return closestIndex;
231   }
232
233   void findClosestHitPointIndex(RaycastHit raycastBox) {
234       int closestIndex = -1;
235       float closestDistance = float.MaxValue;
236
237       for (int i = 0; i < raycastBox.Length; i++) {
238           if (raycastBox[i].distance < closestDistance) {
239               closestIndex = i;
240               closestDistance = raycastBox[i].distance;
241           }
242       }
243
244       return closestIndex;
245   }
246
247   void findClosestHitPointIndex(RaycastHit raycastBox) {
248       int closestIndex = -1;
249       float closestDistance = float.MaxValue;
250
251       for (int i = 0; i < raycastBox.Length; i++) {
252           if (raycastBox[i].distance < closestDistance) {
253               closestIndex = i;
254               closestDistance = raycastBox[i].distance;
255           }
256       }
257
258       return closestIndex;
259   }
260
261   void findClosestHitPointIndex(RaycastHit raycastBox) {
262       int closestIndex = -1;
263       float closestDistance = float.MaxValue;
264
265       for (int i = 0; i < raycastBox.Length; i++) {
266           if (raycastBox[i].distance < closestDistance) {
267               closestIndex = i;
268               closestDistance = raycastBox[i].distance;
269           }
270       }
271
272       return closestIndex;
273   }
274
275   void findClosestHitPointIndex(RaycastHit raycastBox) {
276       int closestIndex = -1;
277       float closestDistance = float.MaxValue;
278
279       for (int i = 0; i < raycastBox.Length; i++) {
280           if (raycastBox[i].distance < closestDistance) {
281               closestIndex = i;
282               closestDistance = raycastBox[i].distance;
283           }
284       }
285
286       return closestIndex;
287   }
288
289   void findClosestHitPointIndex(RaycastHit raycastBox) {
290       int closestIndex = -1;
291       float closestDistance = float.MaxValue;
292
293       for (int i = 0; i < raycastBox.Length; i++) {
294           if (raycastBox[i].distance < closestDistance) {
295               closestIndex = i;
296               closestDistance = raycastBox[i].distance;
297           }
298       }
299
300       return closestIndex;
301   }
302
303   void findClosestHitPointIndex(RaycastHit raycastBox) {
304       int closestIndex = -1;
305       float closestDistance = float.MaxValue;
306
307       for (int i = 0; i < raycastBox.Length; i++) {
308           if (raycastBox[i].distance < closestDistance) {
309               closestIndex = i;
310               closestDistance = raycastBox[i].distance;
311           }
312       }
313
314       return closestIndex;
315   }
316
317   void findClosestHitPointIndex(RaycastHit raycastBox) {
318       int closestIndex = -1;
319       float closestDistance = float.MaxValue;
320
321       for (int i = 0; i < raycastBox.Length; i++) {
322           if (raycastBox[i].distance < closestDistance) {
323               closestIndex = i;
324               closestDistance = raycastBox[i].distance;
325           }
326       }
327
328       return closestIndex;
329   }
330
331   void findClosestHitPointIndex(RaycastHit raycastBox) {
332       int closestIndex = -1;
333       float closestDistance = float.MaxValue;
334
335       for (int i = 0; i < raycastBox.Length; i++) {
336           if (raycastBox[i].distance < closestDistance) {
337               closestIndex = i;
338               closestDistance = raycastBox[i].distance;
339           }
340       }
341
342       return closestIndex;
343   }
344
345   void findClosestHitPointIndex(RaycastHit raycastBox) {
346       int closestIndex = -1;
347       float closestDistance = float.MaxValue;
348
349       for (int i = 0; i < raycastBox.Length; i++) {
350           if (raycastBox[i].distance < closestDistance) {
351               closestIndex = i;
352               closestDistance = raycastBox[i].distance;
353           }
354       }
355
356       return closestIndex;
357   }
358
359   void findClosestHitPointIndex(RaycastHit raycastBox) {
360       int closestIndex = -1;
361       float closestDistance = float.MaxValue;
362
363       for (int i = 0; i < raycastBox.Length; i++) {
364           if (raycastBox[i].distance < closestDistance) {
365               closestIndex = i;
366               closestDistance = raycastBox[i].distance;
367           }
368       }
369
370       return closestIndex;
371   }
372
373   void findClosestHitPointIndex(RaycastHit raycastBox) {
374       int closestIndex = -1;
375       float closestDistance = float.MaxValue;
376
377       for (int i = 0; i < raycastBox.Length; i++) {
378           if (raycastBox[i].distance < closestDistance) {
379               closestIndex = i;
380               closestDistance = raycastBox[i].distance;
381           }
382       }
383
384       return closestIndex;
385   }
386
387   void findClosestHitPointIndex(RaycastHit raycastBox) {
388       int closestIndex = -1;
389       float closestDistance = float.MaxValue;
390
391       for (int i = 0; i < raycastBox.Length; i++) {
392           if (raycastBox[i].distance < closestDistance) {
393               closestIndex = i;
394               closestDistance = raycastBox[i].distance;
395           }
396       }
397
398       return closestIndex;
399   }
400
401   void findClosestHitPointIndex(RaycastHit raycastBox) {
402       int closestIndex = -1;
403       float closestDistance = float.MaxValue;
404
405       for (int i = 0; i < raycastBox.Length; i++) {
406           if (raycastBox[i].distance < closestDistance) {
407               closestIndex = i;
408               closestDistance = raycastBox[i].distance;
409           }
410       }
411
412       return closestIndex;
413   }
414
415   void findClosestHitPointIndex(RaycastHit raycastBox) {
416       int closestIndex = -1;
417       float closestDistance = float.MaxValue;
418
419       for (int i = 0; i < raycastBox.Length; i++) {
420           if (raycastBox[i].distance < closestDistance) {
421               closestIndex = i;
422               closestDistance = raycastBox[i].distance;
423           }
424       }
425
426       return closestIndex;
427   }
428
429   void findClosestHitPointIndex(RaycastHit raycastBox) {
430       int closestIndex = -1;
431       float closestDistance = float.MaxValue;
432
433       for (int i = 0; i < raycastBox.Length; i++) {
434           if (raycastBox[i].distance < closestDistance) {
435               closestIndex = i;
436               closestDistance = raycastBox[i].distance;
437           }
438       }
439
440       return closestIndex;
441   }
442
443   void findClosestHitPointIndex(RaycastHit raycastBox) {
444       int closestIndex = -1;
445       float closestDistance = float.MaxValue;
446
447       for (int i = 0; i < raycastBox.Length; i++) {
448           if (raycastBox[i].distance < closestDistance) {
449               closestIndex = i;
450               closestDistance = raycastBox[i].distance;
451           }
452       }
453
454       return closestIndex;
455   }
456
457   void findClosestHitPointIndex(RaycastHit raycastBox) {
458       int closestIndex = -1;
459       float closestDistance = float.MaxValue;
460
461       for (int i = 0; i < raycastBox.Length; i++) {
462           if (raycastBox[i].distance < closestDistance) {
463               closestIndex = i;
464               closestDistance = raycastBox[i].distance;
465           }
466       }
467
468       return closestIndex;
469   }
470
471   void findClosestHitPointIndex(RaycastHit raycastBox) {
472       int closestIndex = -1;
473       float closestDistance = float.MaxValue;
474
475       for (int i = 0; i < raycastBox.Length; i++) {
476           if (raycastBox[i].distance < closestDistance) {
477               closestIndex = i;
478               closestDistance = raycastBox[i].distance;
479           }
480       }
481
482       return closestIndex;
483   }
484
485   void findClosestHitPointIndex(RaycastHit raycastBox) {
486       int closestIndex = -1;
487       float closestDistance = float.MaxValue;
488
489       for (int i = 0; i < raycastBox.Length; i++) {
490           if (raycastBox[i].distance < closestDistance) {
491               closestIndex = i;
492               closestDistance = raycastBox[i].distance;
493           }
494       }
495
496       return closestIndex;
497   }
498
499   void findClosestHitPointIndex(RaycastHit raycastBox) {
500       int closestIndex = -1;
501       float closestDistance = float.MaxValue;
502
503       for (int i = 0; i < raycastBox.Length; i++) {
504           if (raycastBox[i].distance < closestDistance) {
505               closestIndex = i;
506               closestDistance = raycastBox[i].distance;
507           }
508       }
509
510       return closestIndex;
511   }
512
513   void findClosestHitPointIndex(RaycastHit raycastBox) {
514       int closestIndex = -1;
515       float closestDistance = float.MaxValue;
516
517       for (int i = 0; i < raycastBox.Length; i++) {
518           if (raycastBox[i].distance < closestDistance) {
519               closestIndex = i;
520               closestDistance = raycastBox[i].distance;
521           }
522       }
523
524       return closestIndex;
525   }
526
527   void findClosestHitPointIndex(RaycastHit raycastBox) {
528       int closestIndex = -1;
529       float closestDistance = float.MaxValue;
530
531       for (int i = 0; i < raycastBox.Length; i++) {
532           if (raycastBox[i].distance < closestDistance) {
533               closestIndex = i;
534               closestDistance = raycastBox[i].distance;
535           }
536       }
537
538       return closestIndex;
539   }
540
541   void findClosestHitPointIndex(RaycastHit raycastBox) {
542       int closestIndex = -1;
543       float closestDistance = float.MaxValue;
544
545       for (int i = 0; i < raycastBox.Length; i++) {
546           if (raycastBox[i].distance < closestDistance) {
547               closestIndex = i;
548               closestDistance = raycastBox[i].distance;
549           }
550       }
551
552       return closestIndex;
553   }
554
555   void findClosestHitPointIndex(RaycastHit raycastBox) {
556       int closestIndex = -1;
557       float closestDistance = float.MaxValue;
558
559       for (int i = 0; i < raycastBox.Length; i++) {
560           if (raycastBox[i].distance < closestDistance) {
561               closestIndex = i;
562               closestDistance = raycastBox[i].distance;
563           }
564       }
565
566       return closestIndex;
567   }
568
569   void findClosestHitPointIndex(RaycastHit raycastBox) {
570       int closestIndex = -1;
571       float closestDistance = float.MaxValue;
572
573       for (int i = 0; i < raycastBox.Length; i++) {
574           if (raycastBox[i].distance < closestDistance) {
575               closestIndex = i;
576               closestDistance = raycastBox[i].distance;
577           }
578       }
579
580       return closestIndex;
581   }
582
583   void findClosestHitPointIndex(RaycastHit raycastBox) {
584       int closestIndex = -1;
585       float closestDistance = float.MaxValue;
586
587       for (int i = 0; i < raycastBox.Length; i++) {
588           if (raycastBox[i].distance < closestDistance) {
589               closestIndex = i;
590               closestDistance = raycastBox[i].distance;
591           }
592       }
593
594       return closestIndex;
595   }
596
597   void findClosestHitPointIndex(RaycastHit raycastBox) {
598       int closestIndex = -1;
599       float closestDistance = float.MaxValue;
600
601       for (int i = 0; i < raycastBox.Length; i++) {
602           if (raycastBox[i].distance < closestDistance) {
603               closestIndex = i;
604               closestDistance = raycastBox[i].distance;
605           }
606       }
607
608       return closestIndex;
609   }
610
611   void findClosestHitPointIndex(RaycastHit raycastBox) {
612       int closestIndex = -1;
613       float closestDistance = float.MaxValue;
614
615       for (int i = 0; i < raycastBox.Length; i++) {
616           if (raycastBox[i].distance < closestDistance) {
617               closestIndex = i;
618               closestDistance = raycastBox[i].distance;
619           }
620       }
621
622       return closestIndex;
623   }
624
625   void findClosestHitPointIndex(RaycastHit raycastBox) {
626       int closestIndex = -1;
627       float closestDistance = float.MaxValue;
628
629       for (int i = 0; i < raycastBox.Length; i++) {
630           if (raycastBox[i].distance < closestDistance) {
631               closestIndex = i;
632               closestDistance = raycastBox[i].distance;
633           }
634       }
635
636       return closestIndex;
637   }
638
639   void findClosestHitPointIndex(RaycastHit raycastBox) {
640       int closestIndex = -1;
641       float closestDistance = float.MaxValue;
642
643       for (int i = 0; i < raycastBox.Length; i++) {
644           if (raycastBox[i].distance < closestDistance) {
645               closestIndex = i;
646               closestDistance = raycastBox[i].distance;
647           }
648       }
649
650       return closestIndex;
651   }
652
653   void findClosestHitPointIndex(RaycastHit raycastBox) {
654       int closestIndex = -1;
655       float closestDistance = float.MaxValue;
656
657       for (int i = 0; i < raycastBox.Length; i++) {
658           if (raycastBox[i].distance < closestDistance) {
659               closestIndex = i;
660               closestDistance = raycastBox[i].distance;
661           }
662       }
663
664       return closestIndex;
665   }
666
667   void findClosestHitPointIndex(RaycastHit raycastBox) {
668       int closestIndex = -1;
669       float closestDistance = float.MaxValue;
670
671       for (int i = 0; i < raycastBox.Length; i++) {
672           if (raycastBox[i].distance < closestDistance) {
673               closestIndex = i;
674               closestDistance = raycastBox[i].distance;
675           }
676       }
677
678       return closestIndex;
679   }
680
681   void findClosestHitPointIndex(RaycastHit raycastBox) {
682       int closestIndex = -1;
683       float closestDistance = float.MaxValue;
684
685       for (int i = 0; i < raycastBox.Length; i++) {
686           if (raycastBox[i].distance < closestDistance) {
687               closestIndex = i;
688               closestDistance = raycastBox[i].distance;
689           }
690       }
691
692       return closestIndex;
693   }
694
695   void findClosestHitPointIndex(RaycastHit raycastBox) {
696       int closestIndex = -1;
697       float closestDistance = float.MaxValue;
698
699       for (int i = 0; i < raycastBox.Length; i++) {
700           if (raycastBox[i].distance < closestDistance) {
701               closestIndex = i;
702               closestDistance = raycastBox[i].distance;
703           }
704       }
705
706       return closestIndex;
707   }
708
709   void findClosestHitPointIndex(RaycastHit raycastBox) {
710       int closestIndex = -1;
711       float closestDistance = float.MaxValue;
712
713       for (int i = 0; i < raycastBox.Length; i++) {
714           if (raycastBox[i].distance < closestDistance) {
715               closestIndex = i;
716               closestDistance = raycastBox[i].distance;
717           }
718       }
719
720       return closestIndex;
721   }
722
723   void findClosestHitPointIndex(RaycastHit raycastBox) {
724       int closestIndex = -1;
725       float closestDistance = float.MaxValue;
726
727       for (int i = 0; i < raycastBox.Length; i++) {
728           if (raycastBox[i].distance < closestDistance) {
729               closestIndex = i;
730               closestDistance = raycastBox[i].distance;
731           }
732       }
733
734       return closestIndex;
735   }
736
737   void findClosestHitPointIndex(RaycastHit raycastBox) {
738       int closestIndex = -1;
739       float closestDistance = float.MaxValue;
740
741       for (int i = 0; i < raycastBox.Length; i++) {
742           if (raycastBox[i].distance < closestDistance) {
743               closestIndex = i;
744               closestDistance = raycastBox[i].distance;
745           }
746       }
747
748       return closestIndex;
749   }
750
751   void findClosestHitPointIndex(RaycastHit raycastBox) {
752       int closestIndex = -1;
753       float closestDistance = float.MaxValue;
754
755       for (int i = 0; i < raycastBox.Length; i++) {
756           if (raycastBox[i].distance < closestDistance) {
757               closestIndex = i;
758               closestDistance = raycastBox[i].distance;
759           }
760       }
761
762       return closestIndex;
763   }
764
765   void findClosestHitPointIndex(RaycastHit raycastBox) {
766       int closestIndex = -1;
767       float closestDistance = float.MaxValue;
768
769       for (int i = 0; i < raycastBox.Length; i++) {
770           if (raycastBox[i].distance < closestDistance) {
771               closestIndex = i;
772               closestDistance = raycastBox[i].distance;
773           }
774       }
775
776       return closestIndex;
777   }
778
779   void findClosestHitPointIndex(RaycastHit raycastBox) {
780       int closestIndex = -1;
781       float closestDistance = float.MaxValue;
782
783       for (int i = 0; i < raycastBox.Length; i++) {
784           if (raycastBox[i].distance < closestDistance) {
785               closestIndex = i;
786               closestDistance = raycastBox[i].distance;
787           }
788       }
789
790       return closestIndex;
791   }
792
793   void findClosestHitPointIndex(RaycastHit raycastBox) {
794       int closestIndex = -1;
795       float closestDistance = float.MaxValue;
796
797       for (int i = 0; i < raycastBox.Length; i++) {
798           if (raycastBox[i].distance < closestDistance) {
799               closestIndex = i;
800               closestDistance = raycastBox[i].distance;
801           }
802       }
803
804       return closestIndex;
805   }
806
807   void findClosestHitPointIndex(RaycastHit raycastBox) {
808       int closestIndex = -1;
809       float closestDistance = float.MaxValue;
810
811       for (int i = 0; i < raycastBox.Length; i++) {
812           if (raycastBox[i].distance < closestDistance) {
813               closestIndex = i;
814               closestDistance = raycastBox[i].distance;
815           }
816       }
817
818       return closestIndex;
819   }
820
821   void findClosestHitPointIndex(RaycastHit raycastBox) {
822       int closestIndex = -1;
823       float closestDistance = float.MaxValue;
824
825       for (int i = 0; i < raycastBox.Length; i++) {
826           if (raycastBox[i].distance < closestDistance) {
827               closestIndex = i;
828               closestDistance = raycastBox[i].distance;
829           }
830       }
831
832       return closestIndex;
833   }
834
835   void findClosestHitPointIndex(RaycastHit raycastBox) {
836       int closestIndex = -1;
837       float closestDistance = float.MaxValue;
838
839       for (int i = 0; i < raycastBox.Length; i++) {
840           if (raycastBox[i].distance < closestDistance) {
841               closestIndex = i;
842               closestDistance = raycastBox[i].distance;
843           }
844       }
845
846       return closestIndex;
847   }
848
849   void findClosestHitPointIndex(RaycastHit raycastBox) {
850       int closestIndex = -1;
851       float closestDistance = float.MaxValue;
852
853       for (int i = 0; i < raycastBox.Length; i++) {
854           if (raycastBox[i].distance < closestDistance) {
855               closestIndex = i;
856               closestDistance = raycastBox[i].distance;
857           }
858       }
859
860       return closestIndex;
861   }
862
863   void findClosestHitPointIndex(RaycastHit raycastBox) {
864       int closestIndex = -1;
865       float closestDistance = float.MaxValue;
866
867       for (int i = 0; i < raycastBox.Length; i++) {
868           if (raycastBox[i].distance < closestDistance) {
869               closestIndex = i;
870               closestDistance = raycastBox[i].distance;
871           }
872       }
873
874       return closestIndex;
875   }
876
877   void findClosestHitPointIndex(RaycastHit raycastBox) {
878       int closestIndex = -1;
879       float closestDistance = float.MaxValue;
880
881       for (int i = 0; i < raycastBox.Length; i++) {
882           if (raycastBox[i].distance < closestDistance) {
883               closestIndex = i;
884               closestDistance = raycastBox[i].distance;
885           }
886       }
887
888       return closestIndex;
889   }
890
891   void findClosestHitPointIndex(RaycastHit raycastBox) {
892       int closestIndex = -1;
893       float closestDistance = float.MaxValue;
894
895       for (int i = 0; i < raycastBox.Length; i++) {
896           if (raycastBox[i].distance < closestDistance) {
897               closestIndex = i;
898               closestDistance = raycastBox[i].distance;
899           }
900       }
901
902       return closestIndex;
903   }
904
905   void findClosestHitPointIndex(RaycastHit raycastBox) {
906       int closestIndex = -1;
907       float closestDistance = float.MaxValue;
908
909       for (int i = 0; i < raycastBox.Length; i++) {
910           if (raycastBox[i].distance < closestDistance) {
911               closestIndex = i;
912               closestDistance = raycastBox[i].distance;
913           }
914       }
915
916       return closestIndex;
917   }
918
919   void findClosestHitPointIndex(RaycastHit raycastBox) {
920       int closestIndex = -1;
921       float closestDistance = float.MaxValue;
922
923       for (int i = 0; i < raycastBox.Length; i++) {
924           if (raycastBox[i].distance < closestDistance) {
925               closestIndex = i;
926               closestDistance = raycastBox[i].distance;
927           }
928       }
929
930       return closestIndex;
931   }
932
933   void findClosestHitPointIndex(RaycastHit raycastBox) {
934       int closestIndex = -1;
935       float closestDistance = float.MaxValue;
936
937       for (int i = 0; i < raycastBox.Length; i++) {
938           if (raycastBox[i].distance < closestDistance) {
939               closestIndex = i;
940               closestDistance = raycastBox[i].distance;
941           }
942       }
943
944       return closestIndex;
945   }
946
947   void findClosestHitPointIndex(RaycastHit raycastBox) {
948       int closestIndex = -1;
949       float closestDistance = float.MaxValue;
950
951       for (int i = 0; i < raycastBox.Length; i++) {
952           if (raycastBox[i].distance < closestDistance) {
953               closestIndex = i;
954               closestDistance = raycastBox[i].distance;
955           }
956       }
957
958       return closestIndex;
959   }
960
961   void findClosestHitPointIndex(RaycastHit raycastBox) {
962       int closestIndex = -1;
963       float closestDistance = float.MaxValue;
964
965       for (int i = 0; i < raycastBox.Length; i++) {
966           if (raycastBox[i].distance < closestDistance) {
967               closestIndex = i;
968               closestDistance = raycastBox[i].distance;
969           }
970       }
971
972       return closestIndex;
973   }
974
975   void findClosestHitPointIndex(RaycastHit raycastBox) {
976       int closestIndex = -1;
977       float closestDistance = float.MaxValue;
978
979       for (int i = 0; i < raycastBox.Length; i++) {
980           if (raycastBox[i].distance < closestDistance) {
981               closestIndex = i;
982               closestDistance = raycastBox[i].distance;
983           }
984       }
985
986       return closestIndex;
987   }
988
989   void findClosestHitPointIndex(RaycastHit raycastBox) {
990       int closestIndex = -1;
991       float closestDistance = float.MaxValue;
992
993       for (int i = 0; i < raycastBox.Length; i++) {
994           if (raycastBox[i].distance < closestDistance) {
995               closestIndex = i;
996               closestDistance = raycastBox[i].distance;
997           }
998       }
999
1000      return closestIndex;
1001  }
1002
1003  void findClosestHitPointIndex(RaycastHit raycastBox) {
1004      int closestIndex = -1;
1005      float closestDistance = float.MaxValue;
1006
1007      for (int i = 0; i < raycastBox.Length; i++) {
1008          if (raycastBox[i].distance < closestDistance) {
1009              closestIndex = i;
1010              closestDistance = raycastBox[i].distance;
1011          }
1012      }
1013
1014      return closestIndex;
1015  }
1016
1017  void findClosestHitPointIndex(RaycastHit raycastBox) {
1018      int closestIndex = -1;
1019      float closestDistance = float.MaxValue;
1020
1021      for (int i = 0; i < raycastBox.Length; i++) {
1022          if (raycastBox[i].distance < closestDistance) {
1023              closestIndex = i;
1024              closestDistance = raycastBox[i].distance;
1025          }
1026      }
1027
1028      return closestIndex;
1029  }
1030
1031  void findClosestHitPointIndex(RaycastHit raycastBox) {
1032      int closestIndex = -1;
1033      float closestDistance = float.MaxValue;
1034
1035      for (int i = 0; i < raycastBox.Length; i++) {
1036          if (raycastBox[i].distance < closestDistance) {
1037              closestIndex = i;
1038              closestDistance = raycastBox[i].distance;
1039          }
1040      }
1041
1042      return closestIndex;
1043  }
1044
1045  void findClosestHitPointIndex(RaycastHit raycastBox) {
1046      int closestIndex = -1;
1047      float closestDistance = float.MaxValue;
1048
1049      for (int i = 0; i < raycastBox.Length; i++) {
1050          if (raycastBox[i].distance < closestDistance) {
1051              closestIndex = i;
1052              closestDistance = raycastBox[i].distance;
1053          }
1054      }
1055
1056      return closestIndex;
1057  }
1058
1059  void findClosestHitPointIndex(RaycastHit raycastBox) {
1060      int closestIndex = -1;
1061      float closestDistance = float.MaxValue;
1062
1063      for (int i = 0; i < raycastBox.Length; i++) {
1064          if (raycastBox[i].distance < closestDistance) {
1065              closestIndex
```

```

31             break;
32
33         case variablesAggregator.AlertBoxTypeEnum.Common:
34             // Use a common alert for all castBoxes
35             this.GetComponent<initBoxes>().placeAlertBox(
36                 variableAggInstance.commonAlertBox, allCastBoxes[
37                     hitPointIndex], castHitPointsList[hitPointIndex]);
38             break;
39     } else {
40         switch (variableAggInstance.alertBoxType) {
41             case variablesAggregator.AlertBoxTypeEnum.BoxSpecific:
42                 // Use the alert box of each castBox
43                 this.GetComponent<initBoxes>().deactivateAlertBox(
44                     variableAggInstance.raycastBoxes);
45                 break;
46
47             case variablesAggregator.AlertBoxTypeEnum.Common:
48                 // Use a common alert for all castBoxes
49                 this.GetComponent<initBoxes>().deactivateAlertBox(
50                     variableAggInstance.commonAlertBox);
51                 break;
52         }
53     }
54
55     public void checkCurrentMode() {
56         variablesAggregator.CastModeEnum previousMode = variableAggInstance.
57             enabledModeGlobal;
58
59         if (previousMode != variablesAggregator.CastModeEnum.ScanMode) {
60             variableAggInstance.enabledModeGlobal = variablesAggregator.
61                 CastModeEnum.ScanMode;
62             this.GetComponent<initBoxes>().resetStatus();
63         }
64     }
65 }
```

Κώδικας A.8: Αρχείο castModes/continuousCast.cs

```

1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.UI;
5
6 [RequireComponent(typeof(initCastAndBoxesHandler)), RequireComponent(typeof(
7     initBoxes)),
8  RequireComponent(typeof(initFlashHandler)), RequireComponent(typeof(
9     initRumbleHandler))]
10 public class continuousCast : MonoBehaviour {
11     private variablesAggregator variableAggInstance;
12     private GameObject[] castBoxesWithContinuous;
13     private initFlashHandler flashHandler;
14
15     void Start() {
16         variableAggInstance = this.GetComponent<initCastAndBoxesHandler>().
17             variableAggInstanceObject.GetComponent<variablesAggregator>();
18         castBoxesWithContinuous = this.GetComponent<initBoxes>().
19             findCastBoxesWithContinuousEnabled(variableAggInstance.
20                 raycastBoxes);
21         flashHandler = this.GetComponent<initFlashHandler>();
```

```

17     }
18
19     void Update() {
20         if (variableAggInstance.enabledModeGlobal == variablesAggregator.
21             CastModeEnum.ContinuousMode) {
22             List<RaycastHit> hitPointsList = new List<RaycastHit>();
23
24             for (int i = 0; i < castBoxesWithContinuous.Length; i++) {
25                 RaycastHit hitPoint = castBoxesWithContinuous[i].
26                     GetComponent<singleRaycast>().SingleRaycastFunc(
27                         variableAggInstance);
28                 hitPointsList.Add(hitPoint);
29             }
30             int hitPointIndex = this.GetComponent<initBoxes>().
31                 findClosestHitPointIndex(hitPointsList.ToArray());
32             if (hitPointIndex != -1) {
33                 switch (variableAggInstance.alertBoxType) {
34                     case variablesAggregator.AlertBoxTypeEnum.BoxSpecific:
35                         // Use the alert box of each castBox
36                         this.GetComponent<initBoxes>().placeAlertBox(
37                             variableAggInstance.commonAlertBox,
38                             castBoxesWithContinuous[hitPointIndex],
39                             hitPointsList[hitPointIndex]);
40                         break;
41
42                     case variablesAggregator.AlertBoxTypeEnum.Common:
43                         // Use a common alert for all castBoxes
44                         this.GetComponent<initBoxes>().placeAlertBox(
45                             variableAggInstance.commonAlertBox,
46                             castBoxesWithContinuous[hitPointIndex],
47                             hitPointsList[hitPointIndex]);
48                         break;
49                 }
50             }
51
52             if (variableAggInstance.activateRumble) {
53                 float motorSpeed = this.GetComponent<initRumbleHandler>().
54                     FindMotorsLowAndHigh(hitPointsList[hitPointIndex].
55                         distance);
56                 float stopFreq = this.GetComponent<initRumbleHandler>().
57                     GetStopFrequency(hitPointsList[hitPointIndex].
58                         distance);
59                 Dictionary<string, float> motorSpeedDict = this.
60                     GetComponent<initRumbleHandler>().
61                     GetSpeedOfEachMotor(hitPointsList[hitPointIndex],
62                         motorSpeed);
63
64                 print("Motor speed: " + motorSpeed);
65                 if (motorSpeed > 0.2f) {
66                     this.GetComponent<initRumbleHandler>().RumblePulse(
67                         motorSpeedDict["low"], motorSpeedDict["high"],
68                         stopFreq, 0.5f);
69                 } else {
70                     this.GetComponent<initRumbleHandler>().StopRumble()
71                         ;
72                 }
73             }
74
75             if (variableAggInstance.changeAlertPitch) {
76                 switch (variableAggInstance.alertBoxType) {
77                     case variablesAggregator.AlertBoxTypeEnum.
78                         BoxSpecific:
79                         break;
80                 }
81             }
82         }
83     }

```

```

58
59             case variablesAggregator.AlertBoxTypeEnum.Common:
60                 this.GetComponent<initBoxes>().changeAlertPitch(
61                     variableAggInstance.commonAlertBox,
62                     hitPointsList[hitPointIndex]);
63             break;
64         }
65     }
66
67     if (variableAggInstance.activateFlash) {
68         if (hitPointsList[hitPointIndex].distance <=
69             variableAggInstance.distanceThreadToActivate) {
70             variablesAggregator.lightPositionEnum []
71                 lightPositionsToStart = flashHandler.
72                 FindHitPointPosition(hitPointsList[hitPointIndex]
73                 );
74             Color colorToUse = flashHandler.
75                 FindColorBasedOnDistance(hitPointsList[
76                     hitPointIndex].distance);
77             flashHandler.StartMultipleFlashes(
78                 lightPositionsToStart, colorToUse);
79         } else {
80             flashHandler.StopFlashes();
81         }
82     }
83     } else {
84         this.GetComponent<initRumbleHandler>().StopRumble();
85         switch (variableAggInstance.alertBoxType) {
86             case variablesAggregator.AlertBoxTypeEnum.BoxSpecific:
87                 // Use the alert box of each castBox
88                 this.GetComponent<initBoxes>().deactivateAlertBox(
89                     castBoxesWithContinuous);
90             break;
91
92             case variablesAggregator.AlertBoxTypeEnum.Common:
93                 // Use a common alert for all castBoxes
94                 // this.GetComponent<initBoxes>().
95                 deactivateAlertBox(variableAggInstance.
96                     commonAlertBox);
97             break;
98         }
99     }
100 }
101
102 /**
103  * Toggles the continuous mode on
104  */
105 public void continuousModeGlobalToggle() {
106
107     variablesAggregator.CastModeEnum previousMode = variableAggInstance
108         .enabledModeGlobal;
109
110     if (previousMode != variablesAggregator.CastModeEnum.ContinuousMode
111         ) {
112         variableAggInstance.enabledModeGlobal = variablesAggregator.
113             CastModeEnum.ContinuousMode;
114     }
115 }
116
117 /**
118  * The function is called with the voice input <c>"Continuous Mode
119  * "</c>
120  */
121 
```

```

103         this.GetComponent<initBoxes>().resetStatus();
104     }
105 }
106 }
```

Κώδικας A.9: Αρχείο castModes/handRaycast.cs

```

1 using Microsoft.MixedReality.Toolkit;
2 using Microsoft.MixedReality.Toolkit.Input;
3 using System.Collections.Generic;
4 using UnityEngine;
5
6 [RequireComponent(typeof(initCastAndBoxesHandler)), RequireComponent(typeof(
7     initBoxes))]
8 public class handRaycast : MonoBehaviour {
9     private variablesAggregator variableAggInstance;
10    private GameObject[] handAlertBoxes;
11    private Dictionary<string, GameObject> handAlertBoxesDict = new
12        Dictionary<string, GameObject>();
13
14    public void Start() {
15        variableAggInstance = this.GetComponent<initCastAndBoxesHandler>();
16        variableAggInstance.GetComponent<variablesAggregator>();
17        handAlertBoxes = variableAggInstance.handAlertBoxes;
18
19        foreach (var handAlertBox in handAlertBoxes) {
20            if (!handAlertBox.CompareTag("Untagged")) {
21                handAlertBoxesDict.Add(handAlertBox.tag, handAlertBox);
22            }
23        }
24    }
25
26    void Update() {
27        if (variableAggInstance.enabledModeGlobal == variablesAggregator.
28            CastModeEnum.HandsMode) {
29            var isHandPresent = false;
30
31            foreach (var source in CoreServices.InputSystem.
32                DetectedInputSources) {
33                // Ignore anything that is not a hand because we want
34                // articulated hands
35                if (source.SourceType == InputSourceType.Hand && source.
36                    SourceName != "None Hand") {
37                    isHandPresent = true;
38
39                    foreach (var p in source.Pointers) {
40                        if (p is IMixedRealityNearPointer) {
41                            // Ignore near pointers, we only want the rays
42                            continue;
43                        }
44                        if (p.Result != null) {
45                            //var startPoint = p.Position;
46                            var endPoint = p.Result.Details.Point;
47                            var rayDist = p.Result.Details.RayDistance;
48                            //var hitObject = p.Result.Details.Object;
49                            if (rayDist <= variableAggInstance.maxDistance)
50                            {
51                                if (handAlertBoxesDict.ContainsKey(source.
52                                    SourceName)) {
53                                    handAlertBoxesDict[source.SourceName].
54                                        SetActive(true);
55                                }
56                            }
57                        }
58                    }
59                }
60            }
61        }
62    }
63}
```

```

44                     handAlertBoxesDict[source.SourceName].
45                     transform.position = endPoint;
46
47             if (variableAggInstance.
48                 changeAlertPitch) {
49                 this.GetComponent<initBoxes>().
50                     changeAlertPitch(
51                         handAlertBoxesDict[source.
52                             SourceName], p.Result.Details.
53                             RayDistance);
54
55         }
56     }
57
58     if (p.Controller.Interactions[2].BoolData == true)
59     {
60         if (handAlertBoxesDict.ContainsKey(source.
61             SourceName)) {
62             if (handAlertBoxesDict[source.SourceName].
63                 GetComponent< AudioSource >().isPlaying ==
64                 true) {
65                 handAlertBoxesDict[source.SourceName].
66                     GetComponent< AudioSource >().Pause();
67             }
68         }
69     }
70 }
71
72 if (!isHandPresent) {
73     foreach (var handAlertBox in handAlertBoxes) {
74         handAlertBox.SetActive(false);
75     }
76 }
77 }
78 }
79
80 /// <summary>
81 ///     Toggles Hands Mode on.
82 /// </summary>
83 public void enabledHandABToggle() {
84     variablesAggregator.CastModeEnum previousMode = variableAggInstance
85         .enabledModeGlobal;
86
87     if (previousMode != variablesAggregator.CastModeEnum.HandsMode) {
88         variableAggInstance.enabledModeGlobal = variablesAggregator.

```

```

                CastModeEnum.HandsMode;
88         this.GetComponent<initBoxes>().resetStatus();
89     }
90 }
91 }
```

Kώδικας A.10: Αρχείο castModes/stopCast.cs

```

1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 [RequireComponent(typeof(initCastAndBoxesHandler)), RequireComponent(typeof(
6     continuousCast))]
7 public class stopCast : MonoBehaviour {
8     private variablesAggregator variableAggInstance;
9
10    void Start() {
11        variableAggInstance = this.GetComponent<initCastAndBoxesHandler>().
12            variableAggregatorObject.GetComponent<variablesAggregator>();
13    }
14
15    /// <summary>
16    /// Toggle the stop mode. In stop mode, all alert boxes are disabled.
17    /// </summary>
18    /// <remarks>
19    ///     The function is called with the voice input <c>"Stop Mode"</c>
20    /// </remarks>
21    public void EnableStopMode() {
22        variablesAggregator.CastModeEnum previousMode = variableAggInstance
23            .enabledModeGlobal;
24
25        if (previousMode != variablesAggregator.CastModeEnum.StopMode) {
26            variableAggInstance.enabledModeGlobal = variablesAggregator.
27                CastModeEnum.StopMode;
28            this.GetComponent<initBoxes>().resetStatus();
29        }
30    }
31 }
```

Kώδικας A.11: Αρχείο singleRaycast.cs

```

1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 [RequireComponent(typeof(initSingleBox))]
6 public class singleRaycast : MonoBehaviour
7 {
8     private bool m_HitDetect;
9     private RaycastHit hitInfo;
10
11    /// <summary>
12    ///     Casts a raycast or boxcast from the position of a castBox and
13    ///     detects a hitPoint.
14    /// </summary>
15    /// <param name="variableAggInstance">The variable aggregator
16    /// GameObjetc</param>
17    /// <returns>
18    ///     The hitPoint of the cast
```

```

17     /// </returns>
18     public RaycastHit SingleRaycastFunc(variablesAggregator
19         variableAggInstance) {
20         Vector3 fwd = transform.TransformDirection(Vector3.forward); // 
21             forward direction
22
23         float maxDistance = variableAggInstance.maxDistance;
24
25         variablesAggregator.CastTypeEnum castType = variableAggInstance.
26             castType;
27
28         switch (castType) {
29             case variablesAggregator.CastTypeEnum.Raycast:
30                 m_HitDetect = Physics.Raycast(transform.position, fwd, out
31                     hitInfo, 5.0f);
32                 break;
33
34             case variablesAggregator.CastTypeEnum.BoxCast:
35                 m_HitDetect = Physics.BoxCast(transform.position, transform
36                     .localScale, fwd, out hitInfo, transform.rotation,
37                     maxDistance);
38                 break;
39             }
40
41         return hitInfo;
42     }
43 }

```

Κώδικας A.12: Αρχείο flashingLight.cs

```

1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.UI;
5
6 public class flashingLight : MonoBehaviour
7 {
8     public GameObject variableAggregatorObject;
9     private variablesAggregator variableAggInstance;
10    [Tooltip("Position of the GameObject relative to the user")]
11    public variablesAggregator.lightPositionEnum lightPosition;
12    private IEnumerator currentFlashRoutine = null;
13    private bool shouldFlashStop = false;
14    private MaterialPropertyBlock newColor;
15
16    void Start() {
17        variableAggInstance = variableAggregatorObject.GetComponent<
18            variablesAggregator>();
19        newColor = new MaterialPropertyBlock();
20    }
21
22    /// <summary>
23    /// Starts the flash effect for the specific gameObject
24    /// </summary>
25    /// <param name="flashInterval">The duration of the flash effect</param>
26    /// <param name="maxAlpha">The max value of the alpha (max opacity)</param>
27    /// <param name="colorOfTheFlash">The color of the flash</param>
28    public void StartFlash(float flashInterval, float maxAlpha, Color
29        colorOfTheFlash) {

```

```

28
29     newColor.SetColor("_Color", colorOfTheFlash);
30     this.GetComponent<Renderer>().SetPropertyBlock(newColor);
31
32     // 0 <= maxAlpha <= 1
33     maxAlpha = Mathf.Clamp(maxAlpha, 0, 1);
34
35     if (currentFlashRoutine == null) {
36         currentFlashRoutine = Flash(flashInterval, maxAlpha);
37         shouldFlashStop = false;
38         StartCoroutine(currentFlashRoutine);
39     }
40 }
41 }
42
43 /// <summary>
44 /// Stops the flash effect for the specific gameObject
45 /// </summary>
46 public void StopFlash() {
47     shouldFlashStop = true;
48     if (currentFlashRoutine != null) {
49         StopCoroutine(currentFlashRoutine);
50         currentFlashRoutine = null;
51
52         newColor.SetColor("_Color", new Color(0, 0, 0));
53         this.GetComponent<Renderer>().SetPropertyBlock(newColor);
54     }
55 }
56
57 /// <summary>
58 /// Applies the flash effect to the gameObject. The effect is looped.
59 /// </summary>
60 /// <param name="flashInterval">The duration of one flash (in seconds)
61 /// </param>
62 /// <param name="maxAlpha">The max value of the alpha</param>
63 /// <returns></returns>
64 IEnumerator Flash(float flashInterval, float maxAlpha) {
65     // Flash In
66     float flashInDuration = flashInterval / 2;
67
68     while(!shouldFlashStop) {
69         for (float t = 0; t <= flashInDuration; t += Time.deltaTime) {
70             Color colorThisFrame = newColor.GetColor("_Color");
71             colorThisFrame.a = Mathf.Lerp(0, maxAlpha, t /
72                 flashInDuration);
73             newColor.SetColor("_Color", colorThisFrame);
74             this.GetComponent<Renderer>().SetPropertyBlock(newColor);
75
76             // wait until next frame
77             yield return null;
78         }
79
80         // Flash Out
81         float flashOutDuration = flashInterval / 2;
82         for (float t = 0; t <= flashOutDuration; t += Time.deltaTime) {
83             Color colorThisFrame = newColor.GetColor("_Color");
84             colorThisFrame.a = Mathf.Lerp(maxAlpha, 0, t /
85                 flashOutDuration);
86             newColor.SetColor("_Color", colorThisFrame);
87             this.GetComponent<Renderer>().SetPropertyBlock(newColor);
88             yield return null;
89     }
90 }
```

```

86         }
87     }
88
89     // ensure alpha is 0
90     newColor.SetColor("_Color", new Color(0, 0, 0, 0));
91     this.GetComponent<Renderer>().SetPropertyBlock(newColor);
92 }
93 }
```

Κώδικας A.13: Αρχείο hideMesh.cs

```

1 using System.Collections;
2 using System.Collections.Generic;
3 using Microsoft.MixedReality.Toolkit;
4 using Microsoft.MixedReality.Toolkit.SpatialAwareness;
5 using UnityEngine;
6
7 public class hideMesh : MonoBehaviour {
8     public bool hideMeshAtStart = false;
9
10    private void Start() {
11        if (hideMeshAtStart == true) {
12            hideMeshFunc();
13        }
14    }
15
16    public void hideMeshFunc() {
17        var observer = CoreServices.GetSpatialAwarenessSystemDataProvider<
18            IMixedRealitySpatialAwarenessMeshObserver>();
19        observer.DisplayOption = SpatialAwarenessMeshDisplayOptions.None;
20    }
21
22    public void showMeshFunc() {
23        var observer = CoreServices.GetSpatialAwarenessSystemDataProvider<
24            IMixedRealitySpatialAwarenessMeshObserver>();
25        observer.DisplayOption = SpatialAwarenessMeshDisplayOptions.Visible
26    }
27}
```

ПАРАРТНМА В'

Ερωτηματολόγιο System Usability Scale

1. Νομίζω ότι θα ήθελα να χρησιμοποιώ αυτή την εφαρμογή συχνά.
2. Βρήκα την εφαρμογή αδικαιολόγητα πολύπλοκη.
3. Σκέφτηκα ότι αυτή η εφαρμογή ήταν εύκολη στη χρήση.
4. Νομίζω ότι θα χρειαστώ βοήθεια από κάποιον τεχνικό για να είμαι σε θέση να χρησιμοποιήσω αυτή την εφαρμογή.
5. Βρήκα τις διάφορες λειτουργίες σε αυτή την εφαρμογή ήταν καλά ολοκληρωμένες.
6. Σκέφτηκα ότι υπήρχε μεγάλη ασυνέπεια σε αυτήν την εφαρμογή.
7. Φαντάζομαι ότι οι περισσότεροι άνθρωποι θα μάθουν να χρησιμοποιούν αυτή την εφαρμογή πολύ γρήγορα.
8. Βρήκα αυτή την εφαρμογή πολύ δύσκολη/περίπλοκη στη χρήση.
9. Ένιωθα πολύ σίγουρος/η χρησιμοποιώντας την εφαρμογή.
10. Χρειάστηκε να μάθω πολλά πράγματα πριν μπορέσω να ξεκινήσω με αυτή την εφαρμογή.

Ερωτήσεις ανοικτού τύπου

1. Σας βοήθησε η εφαρμογή να αντιληφθείτε τη διάταξη του χώρου και να περιηγηθείτε σε αυτόν και με ποιον τρόπο;
2. Τι ήταν αυτό το οποίο σας δυσκόλεψε κάτα τη χρήση της εφαρμογής;
3. Ήταν κάτι το οποίο θεωρείτε ότι έλειπε από την εφαρμογή και θα σας διευκόλυνε την εμπειρία;

Πανεπιστήμιο Πατρών, Πολυτεχνική Σχολή
Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών
Άγγελος Καρδούτσος
© Ιούλιος 2024 – Με την επιφύλαξη παντός δικαιώματος.