ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ: Ηλεκτρονικής και Υπολογιστών (Η.Υ.)

ΕΡΓΑΣΤΗΡΙΟ: Διαδραστικών Τεχνολογιών

«Σχεδίαση εξωτερικού ελεγκτή για τον έλεγχο περιβάλλοντος μικτής πραγματικότητας»

«Design of external controller for interacting with mixed reality environments»

Διπλωματική Εργασία

του

Παπαδούλη Γεωργίου

Αριθμός Μητρώου: 1020865

Αριθμός Διπλωματικής Εργασίας: (1020865 / 2022)

Επιβλέπων : Νικόλαος Αβούρης

Πάτρα, Σεπτέμβριος 2022

ΠΙΣΤΟΠΟΙΗΣΗ

Πιστοποιείται ότι η Διπλωματική Εργασία με θέμα

«Σχεδίαση εξωτερικού ελεγκτή για τον έλεγχο περιβάλλοντος μικτής πραγματικότητας»

«Design of external controller for interacting with mixed reality environments»

Του φοιτητή του Τμήματος Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών

Παπαδούλη Γεωργίου

Αριθμός Μητρώου: 1020865

Παρουσιάστηκε δημόσια και εξετάστηκε στο Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών στις

…….. / …….. / ……..

Ο Επιβλέπων	Ο φοιτητής	Ο Διευθυντής του Τομέα
Αβούρης Νικόλαος	Παπαδούλης Γεώργιος	Παλιουράς Βασίλειος
Καθηγητής		Καθηγητής

Ευχαριστίες

Πριν ξεκινήσει το κείμενο της διπλωματικής θα ήθελα να ευχαριστήσω από τα βάθη της καρδιάς μου την οικογένεια και τους φίλους μου οι οποίοι με έπεισαν πως άξιζε η προσπάθεια και πως πρέπει να πάρω το πτυχίο. Χωρίς αυτούς δε θα υπήρχε αυτή η διπλωματική.

Ευχαριστώ θερμά τον κ. Αβούρη ο οποίος με εμπιστεύτηκα με ένα ιδιαίτερο θέμα το οποίο χτίσαμε και συζητήσαμε μαζί. Επιπλέον ευχαριστώ πολύ τον κ. Φείδα Χρήστο για την υποστήριξη και το μεράκι που έδειξε τους τελευταίους μήνες εκπόνησης της διπλωματικής εργασίας.

Τέλος, ευχαριστώ όλα τα άτομα που συμμετείχαν στο πείραμα για το χρόνο τους και την υπομονή της.

Περίληψη

Στην παρούσα διπλωματική εργασία παρουσιάζεται ο σχεδιασμός και η ανάπτυξη ενός πλαισίου επικοινωνίας μεταξύ μίας ελεγχόμενης συσκευής μικτής πραγματικότητας και ενός ελεγκτή με χρήση του περιβάλλοντος της *Unity*. Συγκεκριμένα, η ελεγχόμενη συσκευή είναι ένα *Microsoft Hololens 2* και ο ελεγκτής ένα έξυπνο κινητό. Κατά το στάδιο της έρευνας πραγματοποιήθηκε αναζήτηση για το τι είναι μικτή πραγματικότητα καθώς και παρουσιάστηκαν βασικά θεωρητικά και τεχνολογικά εργαλεία ανάπτυξής εφαρμογών σε αυτή. Στη συνέχεια παρουσιάστηκε η σχεδίαση και υλοποίηση ενός γενικού πλαισίου επικοινωνίας μεταξύ ενός πελάτη και ενός διακομιστή. Με χρήση αυτού του πλαισίου αναπτύχθηκε το πλαίσιο επικοινωνίας που αφορά συγκεκριμένα την αποστολή δεδομένων εισόδου από τον ελεγκτή στην ελεγχόμενη συσκευή. Έπειτα σχεδιάστηκε ένας αλγόριθμος ο οποίος κάνει τον ελεγκτή να είναι χειριστήριο της συσκευής με 6 βαθμούς ελευθερίας. Τέλος, σχεδιάστηκε ένα πείραμα το οποίο αξιολογεί κατά πόσο μία καινούρια αλληλεπίδραση – στην προκειμένη ένα κλικ στο κινητό – μπορεί να βελτιώσει μια εγγενή αλληλεπίδραση του HoloLens 2 – στην προκειμένη το *air-tap*. Μέσω του πειράματος συγκεντρώθηκαν δεδομένα και προέκυψε μια στατιστική ανάλυση η οποία δείχνει πως αξίζει να ερευνηθούν περαιτέρω τρόποι αλληλεπίδρασης με συσκευής μικτής πραγματικότητας.

Abstract

This diploma thesis showcases the design and development of a communication framework between a controller and a controlled Mixed Reality device in Unity game engine. Specifically, the controller is a smartphone and the controlled device is a Microsoft HoloLens 2. Research was conducted on what constitutes Mixed Reality, as well as the theoretical and technological tools available in order to build MR applications. Afterwards, a framework of general communication between a client and a server was designed and developed. This general framework was used to develop a specific one whose role is to easily send input data from the controller to the controlled device. The next part showcases the design and implementation of an algorithm that makes the smartphone a 6DOF device to the HoloLens. Finally, an experiment was designed and conducted whose aim was to determine how valid and useful a new interaction – in this case a click on the touchscreen – may be compared to a native interaction of the HoloLens 2 - in this case the air-tap. Data was gathered from the experiment which resulted in a statistical analysis which shows that further research should be conducted on alternative ways to interact with a Mixed Reality Device.

Περιεχόμενα

Κατάλογος Σχημάτων

Πρόλογος

Η εικονική, επαυξημένη και εκτεταμένη πραγματικότητα είναι πεδία τα οποία τα τελευταία χρόνια αναπτύσσονται με γοργούς ρυθμούς. Έχουν σχεδιασθεί και αναπτυχθεί διάφορες τεχνολογίες που υλοποιούν τα παραπάνω πεδία, από *HMDs* που τοποθετούνται στο κεφάλι στα πιο απλά καθημερινά έξυπνα κινητά και τάμπλετ. Μάλιστα, η βελτίωση είναι εμφανής και από τη μεριά της προσβασιμότητας και της ταχύτητας, καθώς συνεχώς γίνεται φθηνότερη η παραγωγή καλύτερου υλικού και αναπτύσσονται περισσότερο αποδοτικοί αλγόριθμοι και λογισμικό.

Παρατηρούμε λοιπόν τη χρήση αυτών των τεχνολογιών σε διάφορους τομείς. Εφαρμογές εκπαίδευσης, προσομοίωσης, περιήγησης σε περιβάλλοντα ενδιαφέροντος – λόγου χάριν, πολιτισμικά – καθώς και βιντεοπαιχνίδια είναι κάποια παραδείγματα όπου η εικονική, επαυξημένη και εκτεταμένη πραγματικότητα προσφέρουν μια πιο ολοκληρωμένη, πιο κοντά στον άνθρωπο εμπειρία.

Συγκεκριμένα, η παρούσα διπλωματική ασχολείται περισσότερο με το πεδίο της εκτεταμένης πραγματικότητας, όπου ο χρήστης καλείται να περιηγηθεί και αλληλοεπιδράσει με ένα εικονικό περιβάλλον το οποίο όμως συνυπάρχει με τον πραγματικό κόσμο.

Ένα εύλογο ερώτημα που προκύπτει είναι το εξής: «Τι τρόπους / μηχανισμούς αλληλεπίδρασης θα προσφέρω στο χρήστη μίας συσκευής εκτεταμένης πραγματικότητας;» Συνηθέστερα, η αλληλεπίδραση γίνεται είτε με κάποιον ενσωματωμένο ελεγκτή – φωνητική εντολή, αναγνώριση χειρονομιών, εντοπισμός χεριού, εντοπισμός βλέμματος, εντοπισμός κεφαλής – είτε με κάποιο εξωτερικό ελεγκτή / χειριστήριο. Η πρώτη περίπτωση έχει το πλεονέκτημα λιγότερου υλικού, άρα και φθηνότερης υλοποίησης, με περιορισμούς το πεδίο όρασης της συσκευής και την υπολογιστική της ισχύ. Η δεύτερη περίπτωση αυξάνει το κόστος, αλλά προσφέρει έναν επιπλέον μηχανισμό ο οποίος επεκτείνει τις δυνατότητες αλληλεπίδρασης του χρήστη, καθώς και τη συνολική ευρωστία του συστήματος.

Ως μια λύση του υψηλότερου κόστους της χρήσης εξωτερικού ελεγκτή, καθώς και μία συσκευή πειραματισμού για διάφορους τρόπους ελέγχου της εκτεταμένης πραγματικότητας, προτείνεται το κινητό τηλέφωνο που χρησιμοποιούμε καθημερινά. Πλέον, τα κινητά ή έξυπνα τηλέφωνα αποτελούν ισχυρούς υπολογιστές και προσφέρουν πολλές δυνατότητες επέκτασης της αλληλεπίδρασης με ένα εξωτερικό σύστημα.

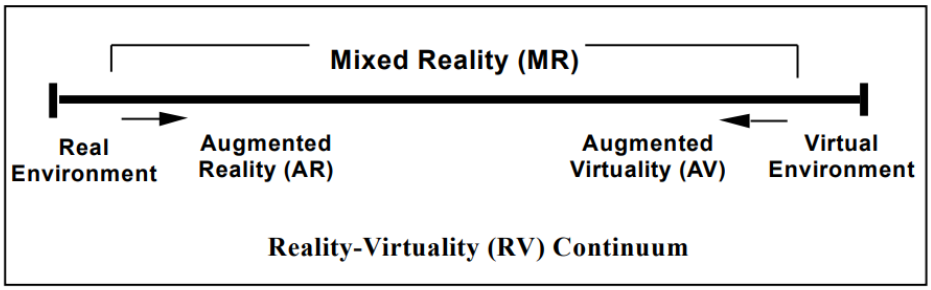
Αυτός είναι και ο σκοπός αυτής της διπλωματικής. Το *HMD* που χρησιμοποιείται στη διπλωματική είναι το *Microsoft Hololens 2* · αν και ο κώδικας επί το πλείστον λειτουργεί και στο *Hololens 1*. Παρουσιάζεται λοιπόν μία προσπάθεια επέκτασης του χειρισμού και της αλληλεπίδρασης που προσφέρει το *Hololens* χρησιμοποιώντας ως εξωτερικό χειριστήριο ένα καθημερινό έξυπνο τηλέφωνο. Επιπλέον, θα εξετάσουμε κατά πόσο είναι δυνατή η μετατροπή του κινητού σε ένα *6DOF Controller*, καθώς και τι άλλες δυνατότητες μπορεί να προσφέρει το κινητό μέσω της οθόνης αφής.

Τέλος, μέσω ενός απλού πειράματος θα συγκρίνουμε μία αλληλεπίδραση που εισαγάγαμε χρησιμοποιώντας την αναπτυχθείσα βιβλιοθήκη (*framework*) με τον εσωτερικό (*native*) τρόπο αλληλεπίδρασης.

Κεφάλαιο 1 – Θεωρητικό & Τεχνολογικό Υπόβαθρο

Στο κεφάλαιο αυτό παρουσιάζονται τα διάφορα θεωρητικά και τεχνολογικά εργαλεία που απαιτούνται για την κατανόηση και ολοκλήρωση του σκοπού της διπλωματικής εργασίας. Καταρχάς, γίνεται μια αναφορά στα πεδία της εικονικής, επαυξημένης και εκτεταμένης πραγματικότητας, καθώς και συσκευές που χρησιμοποιούνται για την υλοποίηση τους. Ακολουθεί συνοπτική ανάπτυξη χρήσιμων μαθηματικών περιγραφής τρισδιάστατου χώρου, μία εποπτική περιήγηση στην Επεξεργασία Εικόνας καθώς και επεξηγούνται οι όροι *Visual Inertial Odometry – Simultaneous Localization and Mapping*, βασικοί όροι για την κατανόηση της εσωτερικής λειτουργίας μιας εκ των αναφερθέντων συσκευών. Στη συνέχεια, παρουσιάζεται το *Hololens 2* και καταγράφονται τα διάφορα είδη ελεγκτών / χειριστηρίων για τη συσκευή αυτή και ο τρόπος αλληλεπίδρασης που προσφέρουν. Τέλος, αναφέρονται τα διάφορα περιβάλλοντα ανάπτυξης της λύσης που προτείνεται στη διπλωματική ·Το (H) Unity και το Visual Studio.

1.1 Εκτεταμένη Πραγματικότητα



ΕΙΚΟΝΑ 1 ΠΡΑΓΜΑΤΙΚΌ - ΕΙΚΟΝΙΚΟ ΣΥΝΕΧΕΣ - MILGRAM ET AL.'S CONTINUUM

Η εκτεταμένη πραγματικότητα αποτελεί μία έννοια που περικλείει την εικονική, την επαυξημένη και τη μικτή πραγματικότητα, καθώς και των ενδιάμεσων αυτών περιοχών. Εντούτοις, αποτελεί ένα υπερσύνολο των παραπάνω, με όρια από το «πλήρως αληθινό» στο «πλήρως εικονικό», όπως προτάθηκε από τον Paul Milgram κατά τα πλαίσια του «πραγματικού-εικονικού συνεχούς» (*Reality-Virtuality Continuum*) [1], [2] (Εικ. 1).

Η ενότητα αυτή εστιάζει στα υποσύνολα της εκτεταμένης πραγματικότητας – εικονική, επαυξημένη και μικτή - καθώς και στην καλύτερη αποσαφήνιση μεταξύ της επαυξημένης και μικτής πραγματικότητας, έννοιες που πολλές φορές χρησιμοποιούνται ως συνώνυμα.

1.1.1 Εικονική Πραγματικότητα (*VR*)

Η εικονική πραγματικότητα , σε αντίθεση με την επαυξημένη και τη μικτή είναι καλώς ορισμένη. Ως εικονική πραγματικότητα ορίζουμε «μία προσομοιωμένη εμπειρία ή οποία μπορεί να είναι παρόμοια ή τελείως διαφορετική σε σύγκριση με τον πραγματικό κόσμο» [3]. Χαρακτηριστικά της, μεταξύ άλλων, είναι η πλήρης εμβύθιση στον εικονικό κόσμο καθώς και η περιήγηση απομακρυσμένων περιοχών σε σχέση πάντα με την τοποθεσία του χρήστη. Επιπλέον, πολλοί υποστηρίζουν πως η χρήση ενός *HMD* είναι αναγκαία ώστε να θεωρηθεί η εμπειρία εικονική, έναντι ενός απλού τρισδιάστατου βίντεο [4]. Μία συσκευή *VR* είναι το *Oculus Rift*.

Η εικονική πραγματικότητα έχει εφαρμογή σε διάφορους τομείς:

- Στην εκπαίδευση ιατρικού προσωπικού και τη βοήθεια επανένταξης ασθενών, μέσω κατάλληλα σχεδιασμένων προσομοιώσεων. Επιπλέον, στην επέκταση της διαδικασίας της διάγνωσης ενός ασθενούς. [5], [6]
- Στη διασκέδαση και στην ψυχαγωγία, μέσω διάφορων βιντεοπαιχνιδιών. [7], [8]
- Στην εκπαίδευση ως εναλλακτική τεχνική εκμάθησης ή βελτίωση υπαρχόντων τεχνικών. [9]

1.1.2 Επαυξημένη Πραγματικότητα (AR)

Η επαυξημένη πραγματικότητα διαφέρει αισθητά από την εικονική. Ο χρήστης πλέον δεν είναι πλήρως εμβυθισμένος σε μια εικονική εμπειρία. Στα μάτια του συνεχίζει να αποτυπώνεται ο πραγματικός κόσμος, στον οποίο όμως ταυτόχρονα υπερτίθενται εικονικά αντικείμενα που διατηρούν τη θέση και τον προσανατολισμό τους στο χώρο. [4], [10] Τα αντικείμενα αυτά, όμως, δεν αλληλοεπιδρούν με τον πραγματικό κόσμο, απλά προβάλλονται πάνω σε αυτόν. Βέβαια, όπως προαναφέρθηκε, άλλες πηγές θεωρούν αναγκαία την αλληλεπίδρασή των αντικειμένων με τον πραγματικό κόσμο, γεγονός που προκαλεί σύγχυση μεταξύ των εννοιών της επαυξημένης και μικτής πραγματικότητας. [4] Μια AR συσκευή πλέον είναι οποιοδήποτε σύγχρονο έξυπνο κινητό.

Η επαυξημένη πραγματικότητα με τη σειρά της βρίσκει εφαρμογή σε διάφορους τομείς:

- Στον κόσμο των βιντεοπαιχνιδιών και της ψυχαγωγίας. Μία πολύ γνωστή εφαρμογή επαυξημένης πραγματικότητας είναι το παιχνίδι Pokémon Go. [11]
- Στον τομέα της αρχιτεκτονικής, του εσωτερικού σχεδιασμού και της διακόσμησης, όπου οι χρήστες μπορούν να προβάλλουν εικονικά αντικείμενα στο χώρο τους για να εκτιμήσουν καλύτερα μια πιθανή αγορά. [12], [13]
- Στον τομέα της πολιτισμικής κληρονομιάς, όπου συνεχώς αναζητούνται νέοι τρόποι προσέγγισης και βελτίωσης της εμπειρίας ενός επισκέπτη. [14]

1.1.3 Μικτή Πραγματικότητα (MR)

Η μικτή πραγματικότητα δεν είναι καλώς ορισμένη, γεγονός που γίνεται εύκολα αντιληπτό αν παρατηρήσουμε διαφορετικούς ορισμούς που έχουν δώσει ειδικοί όταν ερωτήθηκαν για αυτό. Συγκεκριμένα, παρατηρούνται 4 διαφορετικοί ορισμοί [4]:

- Η μικτή πραγματικότητα βάσει του *RV Continuum* των *Milgram et al.,* όπως αυτό παρουσιάζεται στην Εικ. 1. Σε αυτήν την περίπτωση, η μεικτή πραγματικότητα δεν είναι αναγκαίο να περιλαμβάνει την εικονική.
- Η μικτή πραγματικότητα ως συνδυασμός της εικονικής και της επαυξημένης. Η δυνατότητα δηλαδή συνδυασμού των δύο αυτών τεχνολογιών σε μία εφαρμογή ή συσκευή.
- Η μικτή πραγματικότητα ως επέκταση της επαυξημένης. Εδώ ορίζουμε τη μικτή ως μια ικανότερη εκδοχή της επαυξημένης · Λόγου χάρη, καλύτερη κατανόηση του φυσικού περιβάλλοντος στο οποίο βρίσκεται ο χρήστης.
- Η μικτή και η επαυξημένη πραγματικότητα ως συνώνυμα.

Λόγω αυτής της έλλειψης ενός καθολικά αποδεκτού ορισμού, πολλές εφαρμογές μπορούν να κατηγοριοποιηθούν και ως *AR* και ως *MR*. Στα πλαίσια αυτής της διπλωματικής θα αντιμετωπίσουμε τη μικτή πραγματικότητα ως την καλύτερη δυνατή ανίχνευση του περιβάλλοντος με σκοπό την προβολή ενός εικονικού κόσμου ο οποίος αλληλοεπιδρά με τον πραγματικό με μεγάλη ακρίβεια και σε πραγματικό χρόνο. Επιπλέον, θεωρείται αναγκαία η εμβύθιση του χρήστη στον κόσμο, μέσω παραδείγματος χάριν ενός *HMD,* έναντι μίας συσκευής *AR,* όπως τα έξυπνα κινητά.

Παραδείγματα συσκευών μικτής πραγματικότητας είναι τα Microsoft Hololens (1, 2), Magic Leap (1, 2)

Με βάση τα παραπάνω, έχουμε διάφορες εφαρμογές της μικτής πραγματικότητας. Παρακάτω αναφέρονται μερικές από αυτές, οι οποίες επιλέχθηκαν ώστε να ικανοποιούν τον παραπάνω ορισμό:

- Η χρήση της μικτής πραγματικότητας στον τομέα της ιατρικής (τηλε-ιατρική) για παροχή διάγνωσης και περίθαλψης ασθενών όταν χρειάζεται ομάδα ανθρώπων. Η εφαρμογή αυτή βρήκε χρήση ειδικά κατά την περίοδο του COVID-19, όπου ο επιβαλλόταν η αποφυγή του συνωστισμού ατόμων σε ένα μικρό χώρο αλλά ταυτόχρονα ήταν απαραίτητη η συνεργασία του ιατρικού προσωπικού [15]
- Στην αρχιτεκτονική και στην κατασκευαστική βιομηχανία, ειδικότερα σε περιπτώσεις συνεργασίας. Μια ομάδα ανθρώπων μπορεί μέσω διαφορετικών *HMDs* να συζητήσει / αναπτύξει / σχεδιάσει μία κατασκευή μέσω ενός εικονικού μοντέλου που τοποθετούν στον πραγματικό κόσμο. Επιπλέον, η μικτή πραγματικότητα είναι χρήσιμη στην εκπαίδευση νέου εργατικού δυναμικού. [16], [17]
- Σε πολιτισμικά θέματα. Η μικτή πραγματικότητα αποτελεί ένα πολύ δυνατό εργαλείο ώστε να βελτιωθεί και επαυξηθεί η εμπειρία περιήγησης ενός μουσείο ή γενικότερα πολιτισμικών εκθεμάτων. Είναι ιδιαίτερα σημαντικό να διεγείρεις το ενδιαφέρον τόσο των μικρών όσο και των μεγάλων σε πολιτισμικά θέματα. Μια

καλοστημένη εφαρμογή σε περιβάλλον μικτής πραγματικότητας ίσως είναι μια πολύ καλή λύση. [18], [19]

1.2 Γραφικά και αναπαράσταση στο χώρο[20], [21]

Στην ενότητα αυτή παρουσιάζονται συνοπτικά κάποιες σημαντικές έννοιες από την Προβολική Γεωμετρία και πως χρησιμοποιείται στα Γραφικά. Τα μαθηματικά που χρησιμοποιούνται στην Προβολική Γεωμετρία είναι αρκετά σημαντικά για οποιονδήποτε σχεδιαστή μιας τρισδιάστατης εφαρμογής, ανεξαρτήτως αν η εφαρμογή αυτή είναι σχεδιασμένη για εκτεταμένη πραγματικότητα ή απλά για προβολή σε μία τυπική οθόνη. Ειδικά για την υλοποίηση του ελεγκτή στη συγκεκριμένη διπλωματική, η παρούσα ενότητα είναι εξαιρετικά σημαντική.

Αν και κάποιος στις μέρες μας μπορεί να τα αντιμετωπίσει ως ένα «μαύρο κουτί» - δε χρειάζεται και ιδιαίτερη γνώση για να φτιάξεις μια απλή εφαρμογή – η κατανόηση τους εντούτοις μπορεί να κάνει πολύ ευκολότερη την ανάπτυξη της εφαρμογής και να λύσει αρκετά προβλήματα.

1.2.1 Γραφικά και τρισδιάστατος χώρος [20], [21]

Το κυριότερο πρόβλημα που πρέπει να λυθεί στα γραφικά είναι η ρεαλιστική απεικόνιση ενός τρισδιάστατου αντικειμένου, είτε αυτό αφορά το πεδίο της εκτεταμένης πραγματικότητας ή μία τυπική οθόνη. Ταυτόχρονα απαιτείται η δυνατότητα μεταφοράς (*translation*), περιστροφής (*rotation*) γύρω από έναν άξονα, αλλαγής του μεγέθους (*scale*) ενός αντικειμένου, αλλά και συνδυασμού των 3 παραπάνω πράξεων.

Ορίζουμε λοιπόν έναν τρισδιάστατο χώρο, όπου περιγράφουμε ένα σημείο του με τις συντεταγμένες (x, y, z). Αρχή του χώρο αποτελεί το σημείο $(0,0,0)$

1.2.2 Μεταφορά (*Translation*)

Η μεταφορά ενός σημείο στο χώρο είναι από τις πιο απλές πράξεις. Ουσιαστικά, αν έχουμε ένα σημείο στο χώρο, μία οποιαδήποτε μεταφορά εκφράζεται ως η αλγεβρική πρόσθεση των συντεταγμένων του σημείου αυτού με το ποσό κατά το οποίο επιθυμούμε να μετακινηθεί. Αυτό επεκτείνεται σε ένα σύνολο από σημεία, ώστε μπορούμε να πούμε πως η πρόσθεση αυτού του ποσού στις συντεταγμένες του κάθε σημείου προκαλεί ομοιόμορφη μεταφορά των σημείων στον τρισδιάστατο χώρο.

Ορίζουμε ένα γενικό τελεστή μεταφοράς $T(x, y,)$ ώστε:

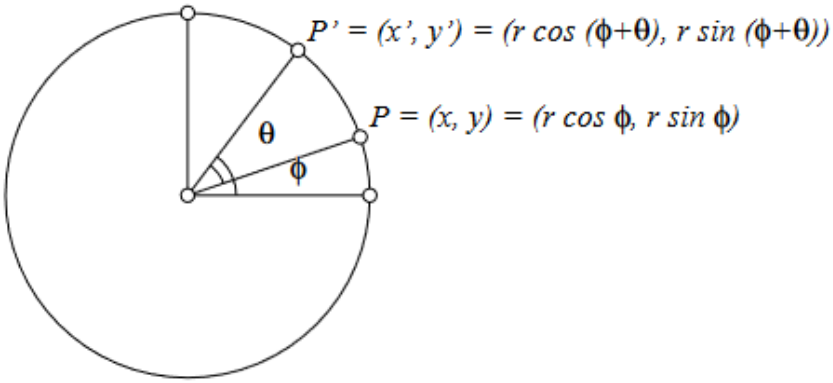
$$T_{t_x,t_y,t_z}(x, y, z) = (x + t_x, y + t_y, z + t_{z)}$$

,

όπου (t_x, t_y, t_z) το ποσό κατά το οποίο μεταφέρεται ένα σημείο στο χώρο.

1.2.3 Περιστροφή γύρω από άξονα (*Rotation*)

Θα ξεκινήσουμε περιγράφοντας την περιστροφή ενός σημείου στο επίπεδο x- y κατά γωνία θ με βάση το σημείο αναφοράς μας. Το σημείο $P = (x, y)$ μπορεί να γραφτεί με πολικές συντεταγμένες ως $P = (x, y) = (r\cos\varphi\,, r\sin\varphi)$. Μία μετατόπιση κατά γωνία θ περιγράφεται καλύτερα στο παρακάτω σχήμα.



ΕΙΚΟΝΑ 2 ΠΕΡΙΣΤΡΟΦΗ ΚΑΤΑ ΤΟΝ ΑΞΟΝΑ Z

Χρησιμοποιώντας την ταυτότητα για την πρόσθεση ημίτονων-συνημίτονων, καταλήγουμε πως

$$\begin{aligned}(x', y') &= (r\cos(\varphi+\theta)\,, r\sin(\varphi+\theta)) \\ &= (r\cos\varphi\cos\theta - r\sin\varphi\sin\theta\,, r\cos\varphi\sin\theta\ + r\sin\varphi\cos\theta) \\ &= (x\cos\theta - y\sin\theta\,, x\sin\theta + y\cos\theta\end{aligned}$$

ορίζοντας έτσι τους τελεστές περιστροφής γύρω από τους άξονες x, y, z ως

$$\begin{aligned}R_{x,\theta}(x, y, z) &= (x, y\cos\theta - z\sin\theta\,, y\sin\theta + z\cos\theta \\ R_{y,\theta}(x, y, z) &= (x\cos\theta + z\sin\theta\,, y, -x\sin\theta + z\cos\theta \\ R_{z,\theta}(x, y, z) &= (x\cos\theta - y\sin\theta\,, x\sin\theta + y\cos\theta\,, z\end{aligned}$$

Είναι σημαντικό να τονίσουμε εδώ πως τα διάφορα πρόσημα προκύπτουν έτσι επειδή το σύστημα συντεταγμένων μας είναι σύστημα «δεξιού χεριού». Τα συστήματα δεξιού-χεριού και αριστερού-χεριού έχουν να κάνουν με την κατεύθυνση του άξονα που αναφέρεται στο ύψος. Στην προκειμένη περίπτωση, έχουμε ορίσει αυτόν τον άξονα να είναι ο z. Αν τα x, y ορίζουν ένα επίπεδο σε ένα τραπέζι, τότε σε ένα σύστημα δεξιού-χεριού η θετική φορά του z δείχνει πάνω από το τραπέζι, ενώ σε ένα σύστημα αριστερού-χεριού δείχνει κάτω από το τραπέζι.

1.2.4 Γενικευμένη Περιστροφή

Στην προηγούμενη υπο-ενότητα ορίσαμε 3 διαφορετικούς τελεστές περιστροφής, έναν για κάθε διαφορετικό άξονα, οι οποίοι περιγράφουν την περιστροφή ενός σημείου με βάση την αναφορά μας (την αρχή των αξόνων). Συνεχίζοντας, θέλουμε να γενικεύσουμε την περιστροφή αυτή ώστε να γίνεται με βάση ένα αυθαίρετο άξονα.

Για να το πετύχουμε αυτό θα ακολουθήσουμε μια απλή τακτική. Επιλέγουμε έναν άξονα, έστω τον x. Θα περιστρέψουμε το σημείο μας ώστε ο άξονας περιστροφής που θέλουμε να χρησιμοποιήσουμε - ο οποίος περνάει από την αρχή των αξόνων - να είναι παράλληλος με τον άξονα x. Αυτό το πετυχαίνουμε εκτελώντας το πολύ δύο περιστροφές, έστω κατά γωνίες φ, ψ στους άξονες x, y αντίστοιχα. Έπειτα περιστρέφουμε κατά την επιθυμητή γωνία θ στον άξονα x και τέλος αντιστρέφουμε τις αρχικές περιστροφές που εκτελέσαμε. Η διαδικασία περιγράφεται από την παρακάτω εξίσωση

$$P' = R_{x,-\varphi}\left(R_{y,-\psi}\left(R_{x,-\theta}\left(R_{y,\psi}(R_{x,\varphi}(P))\right)\right)\right)$$

Θεωρώντας πλέον τον τελεστή R ως έναν γενικευμένο τελεστή περιστροφής γύρω από οποιονδήποτε άξονα περνάει από το σημείο αναφοράς, μπορούμε με τον τελεστή μεταφοράς να γράψουμε μία τελική εξίσωση η οποία περιγράφει περιστροφή γύρω από ένα αυθαίρετο άξονα:

$$P' = T_{x,y,z}\left(R\left(T_{-x,-y,-z}(P)\right)\right)$$

Η διαδικασία που περιγράφει η εξίσωση είναι απλή. Μεταφέρουμε το κέντρο της περιστροφής στην αρχή των αξόνων, εκτελούμε την περιστροφή και αντιστρέφουμε τη μεταφορά.

1.2.5 Αλλαγή μεγέθους (*Scale*)

Η αλλαγή του μεγέθους ενός αντικειμένου είναι απλή. Αν έχουμε ένα αντικείμενο που περιγράφεται από ένα σύνολο σημείων στο χώρο, τότε για να μικρύνουμε ή να μεγαλώσουμε το αντικείμενο αυτό, αρκεί να πολλαπλασιάσουμε τις συντεταγμένες του με ένα βαθμωτό αριθμό. Έτσι επιτυγχάνουμε οποιαδήποτε αλλαγή στο μέγεθος, πάντα με βάση τους τρεις άξονες. Παραμένει όμως το πρόβλημα που είχαμε και πριν. Ένας πολλαπλασιασμός των συντεταγμένων θα άλλαζε το μέγεθος του αντικειμένου αλλά βάσει του σημείου αναφοράς, της αρχής δηλαδή των αξόνων. Αν είχαμε για παράδειγμα ένα αμάξι και θέλαμε να το μικρύνουμε κατά ένα συντελεστή 0.1, τότε ναι μεν θα το αμάξι θα ήταν το 1/10 σε μέγεθος, αλλά θα κατέληγε και περίπου κατά το 1/10 από την αρχή των αξόνων, σε σχέση με το σημείο που βρισκόταν αρχικά.

Η λύση σε αυτό το πρόβλημα έχει ήδη προταθεί στην υπο-ενότητα για τη γενική περιστροφή. Αρκεί να μεταφέρουμε το αντικείμενο στην αρχή των αξόνων, να εκτελέσουμε την αλλαγή μεγέθους και να το μεταφέρουμε ξανά πίσω στην αρχική του θέση.

Έτσι, μία γενική εξίσωση για την αλλαγή μεγέθους ενός συνόλου σημείων με βάση την αρχή των αξόνων είναι:

$$S_{s_x,s_y,s_z}(x,y,z) = (s_x x, s_y y, s_z z),$$

όπου ο κάθε συντελεστής αφορά τον κάθε άξονα, ώστε το τελικό αποτέλεσμα μπορεί να είναι και ανομοιόμορφο. Στην περίπτωση μάλιστα που κάποιος συντελεστής είναι αρνητικός αυτό θα έχει ως αποτέλεσμα την «αντανάκλαση» του αντικειμένου κατά επίπεδο το οποίο είναι κάθετο στον εκάστοτε άξονα.

1.2.6 Συνδυασμός μεταφοράς, περιστροφής και αλλαγής μεγέθους

Συνήθως, αν όχι πάντα, συμφέρει να συνδυάσουμε τους διάφορους αυτούς μετασχηματισμούς ώστε να δημιουργήσουμε ένα πιο σύνθετο μετασχηματισμό ο οποίος περιγράφει ακριβώς αυτό που θέλουμε να κάνουμε. Φαίνεται όμως από την περιγραφή και τον ορισμό των τελεστών πως μια τέτοια διαδικασία θα γινόταν γρήγορα πολύπλοκη, όπως ήδη φαίνεται από τη γενικευμένη περιστροφή.

Η λύση στο παραπάνω πρόβλημα είναι η περιγραφή των 3 αυτών πράξεων με χρήση πινάκων. Η χρήση πινάκων, πέραν του ότι καθιστά πιο ευανάγνωστο και εύχρηστο το μετασχηματισμό, είναι βελτιστοποιημένη, αφού οι υπολογιστές σήμερα έρχονται με λογισμικό και υλικό το οποίο επιταχύνει τις πράξεις σε πίνακες.

Έστω λοιπόν πως τα σημεία $P = (x, y, z)$ αποτελούν διανύσματα στήλες. Μία οποιαδήποτε περιστροφή είναι απλά ένας πολλαπλασιασμός ενός πίνακα περιστροφής με το διάνυσμα στήλη. Έτσι, για τους 3 άξονες έχουμε

$$R_{z,\theta}\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x\cos\theta - y\sin\theta \\ x\sin\theta + y\cos\theta \\ z \end{pmatrix},$$

$$R_{y,\theta}\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix}\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x\cos\theta + z\sin\theta \\ y \\ -x\sin\theta + z\cos\theta \end{pmatrix}$$

και

$$R_{x,\theta}\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix}\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x \\ y\cos\theta - z\sin\theta \\ y\sin\theta + z\cos\theta \end{pmatrix}$$

Από αυτήν την περιγραφή προκύπτει κάτι πολύ χρήσιμο και όμορφο. Συνεχόμενες περιστροφές δεν είναι τίποτα άλλο πέραν πολλαπλασιασμοί μεταξύ πινάκων. Αν δηλαδή για οποιοδήποτε λόγο χρειαστεί να περιστρέψουμε το αντικείμενο κατά 5 διαφορετικούς τρόπους, τότε δεν έχουμε παρά να πολλαπλασιάσουμε 5 διαφορετικούς πίνακες περιστροφής και εν τέλη να πολλαπλασιάσουμε το αποτέλεσμα με τα σημεία του αντικειμένου.

Η αλλαγή μεγέθους είναι ακόμα ευκολότερη. Η μετατροπή του τελεστή σε μορφή πινάκων είναι η εξής:

$$S_{s_x,s_y,s_z}\begin{pmatrix}x\\y\\z\end{pmatrix}=\begin{pmatrix}s_x & 0 & 0\\0 & s_y & 0\\0 & 0 & s_z\end{pmatrix}\begin{pmatrix}x\\y\\z\end{pmatrix}=\begin{pmatrix}s_x x\\s_y y\\s_z z\end{pmatrix}$$

Όπως και πριν, πολλαπλασιασμός διαδοχικών πινάκων ισοδυναμεί με συνεχόμενους μετασχηματισμούς αλλαγής μεγέθους.

Προκύπτει, όμως, ένα πρόβλημα με αυτόν τον τρόπο περιγραφής των ανωτέρω μετασχηματισμών. Η περιγραφή της μεταφοράς με χρήση πινάκων, τουλάχιστον 3 x 3, είναι αδύνατη. Αυτό προκύπτει εύκολα αν εξετάσουμε τον πολλαπλασιασμό του σημείου $(0,0,0)$ με οποιονδήποτε 3 x 3 πίνακα. Το αποτέλεσμα πάντα θα είναι $(0,0,0)$. Αυτό αναστατώνει τον τρόπο περιγραφής που έχει αναπτυχθεί, διότι πλέον δε καθίσταται δυνατός ο συνδυασμός και της μεταφοράς με συνεχόμενους πολλαπλασιασμούς πινάκων.

Η λύση προέρχεται από την επέκταση των παραπάνω πινάκων ώστε να έχουν μία παραπάνω στήλη και γραμμή, ή καλύτερα, ένα επιπλέον διάνυσμα στήλη του οποίου ο τελευταίο στοιχείο ισοδυναμεί με 1. Πλέον, ο μετασχηματισμός, παραδείγματος χάρη της περιστροφής γύρω από τον άξονα z παίρνει τη μορφή:

$$R_{z,\theta}\begin{pmatrix}x\\y\\z\\1\end{pmatrix}=\begin{pmatrix}\cos\theta & -\sin\theta & 0 & 0\\\sin\theta & \cos\theta & 0 & 0\\0 & 0 & 1 & 0\\0 & 0 & 0 & 1\end{pmatrix}\begin{pmatrix}x\\y\\z\\1\end{pmatrix}=\begin{pmatrix}x\cos\theta - y\sin\theta\\x\sin\theta + y\cos\theta\\z\\1\end{pmatrix},$$

Πλέον, με αυτήν την σύνθετη συντεταγμένη, ακόμα και ο μετασχηματισμός της μεταφοράς μπορεί να γραφεί με τον εξής τρόπο:

$$T_{t_x,t_y,t_z}\begin{pmatrix}x\\y\\z\\1\end{pmatrix}=\begin{pmatrix}1 & 0 & 0 & t_x\\0 & 1 & 0 & t_y\\0 & 0 & 1 & t_z\\0 & 0 & 0 & 1\end{pmatrix}\begin{pmatrix}x\\y\\z\\1\end{pmatrix}=\begin{pmatrix}x+t_x\\y+t_y\\z+t_z\\1\end{pmatrix}$$

Συνεπώς, οποιοσδήποτε συνδυασμός της μεταφοράς, περιστροφής και αλλαγής μεγέθους δεν είναι τίποτα άλλο παρά συνεχόμενοι πολλαπλασιασμοί πινάκων. Μάλιστα, οι ανωτέρω συντεταγμένες με αυτή την επιπλέον συνιστώσα λέγονται ομοιογενείς. Χρησιμοποιούνται στην Προβολική Γεωμετρία όπως οι καρτεσιανές χρησιμοποιούνται στην Ευκλείδεια.

Η τελευταία συντεταγμένη δε χρειάζεται να είναι 1. Στις περισσότερες περιπτώσεις είναι γνωστή και ως w, εφόσον και η κυριότερη χρήση των ομοιογενών συντεταγμένων είναι για μονοδιάστατους, δισδιάστατους και τρισδιάστατους Ευκλείδειους χώρους, Είναι ευνόητο πως η τιμή της w δεν επηρεάζει το αποτέλεσμα των μετασχηματισμών.

1.2.7 Τετραδόνιο (*Quaternion)* [22], [23]

Μία άλλη χρήσιμη έννοια στο πως λειτουργούν τα περιβάλλοντα ανάπτυξης τρισδιάστατων εφαρμογών είναι το τετραδόνιο. Αποτελεί μία σχετικά δυσνόητη έννοια η οποία όμως χρησιμοποιείται αρκετά στα γραφικά. Στο πλαίσιο της διπλωματικής θα τα αντιμετωπίσουμε ως ένα «μαύρο κουτί», περιγράφοντας κάποιες σημαντικές ιδιότητες και τη χρησιμότητά τους.

Το τετραδόνιο στα γραφικά χρησιμοποιείται ως ένας εναλλακτικός τρόπος περιγραφής μιας περιστροφής στον τρισδιάστατο χώρο, έναντι των ευκλείδειων γωνιών τις οποίες περιγράψαμε στις προηγούμενες ενότητες. Οι Euler γωνίες είναι ευάλωτες στο φαινόμενο *gimbal lock*. Το φαινόμενο αυτό συμβαίνει όταν δύο από τους άξονες περιστροφές ευθυγραμμίζονται και έτσι χάνεται ένας βαθμός ελευθερίας. Επιπλέον καθιστούν δυσκολότερη και αυθαίρετη την παρεμβολή (*interpolation*) μεταξύ δύο διαφορετικών περιστροφών / προσανατολισμών. Τα προβλήματα αυτά λύνονται με τη χρήση των τετραδόνιων αριθμών.

Το τετραδόνιο ουσιαστικά είναι μία μη-αντιμεταθετική επέκταση της θεωρίας των μιγαδικών αριθμών. Βασικά στοιχεία αποτελούν τα i, j, k ώστε να ισχύει:

$$i^2 = j^2 = k^2 = ijk = -1$$

Ο αριθμός λοιπόν αυτός είναι ένας γραμμικός συνδυασμός των παραπάνω στοιχείων, των βασικών δηλαδή τετραδονίων $1, i, j, k$. Έτσι μπορεί να εκφρασθεί με μοναδικό τρόπο ως:

$$q = a + bi + cj + dk$$

Όπου a, b, c, d πραγματικοί αριθμοί.

Χωρίς να επεκτείνουμε παραπάνω λόγω της πολυπλοκότητας των αριθμών αυτών, μπορούμε να ορίσουμε κάποιες σημαντικές ιδιότητες και κάποιους τελεστές που θα φανούν χρήσιμοι. Οι ορισμοί αυτοί δεν είναι μαθηματικά αυστηροί, αποτελούν απλώς μία προσπάθεια επεξήγησης της χρήσης και χρησιμότητας του τετραδόνιου.

1) Εάν έχουμε ένα τετραδόνιο q, τότε ορίζουμε ως αντίστροφό του το
$$q^{-1} = \frac{q^*}{|q|^2}$$
έτσι ώστε $q^{-1}q = qq^{-1} = 1$. Το q^* δεν είναι τίποτα άλλο παρά ο μιγαδικός συζυγής του q.
2) Ορίζουμε τελεστή $R_q(q1, q2)$ περιστροφών, όπως ακριβώς εξηγήθηκε για τους διαδοχικούς πολλαπλασιασμούς σε προηγούμενη ενότητα. Για χάριν συντομίας και ευκολίας, ταυτίζουμε τον τελεστή R_q :=∗, ώστε να μπορούμε να περιγράψουμε διαδοχικές περιστροφές ως $q_{final} = q_1 * q_2 * \dots * q_n$
3) Ορίζουμε τελεστή $T_q(q, v)$ όπου q τετραδόνιο και v ένα διάνυσμα τριών στοιχείων, ώστε το αποτέλεσμα να είναι ένα καινούριο διάνυσμα το οποίο έχει περιστραφεί κατά την περιστροφή που περιγράφει το τετραδόνιο. Για χάριν συντομίας και ευκολίας, ταυτίζουμε τον τελεστή R_q :=∗∗, ώστε $\boldsymbol{v_{final}} = q ** \boldsymbol{v}$ να είναι ένα διάνυσμα θέσης που προέκυψε από την περιστροφή του αρχικού διανύσματος κατά το τετραδόνιο.

4) Ορίζουμε τελεστή $E(q) = \boldsymbol{\theta}$, όπου $\boldsymbol{\theta}$ διάνυσμα στήλη το οποίο περιγράφει μία περιστροφή κατά γωνίες Euler. Επιπλέον, ισχύει και ο αντίθετος του, ώστε $E^{-1}(\boldsymbol{\theta}) = q$

Είναι σημαντικό να αναφερθεί πως ένα τετραδόνιο μπορεί να περιγράψει οποιουδήποτε είδους γενική περιστροφή, κατά οποιονδήποτε δηλαδή αυθαίρετο άξονα.

1.2.8 Συνάρτηση δημιουργίας πίνακα μετασχηματισμού

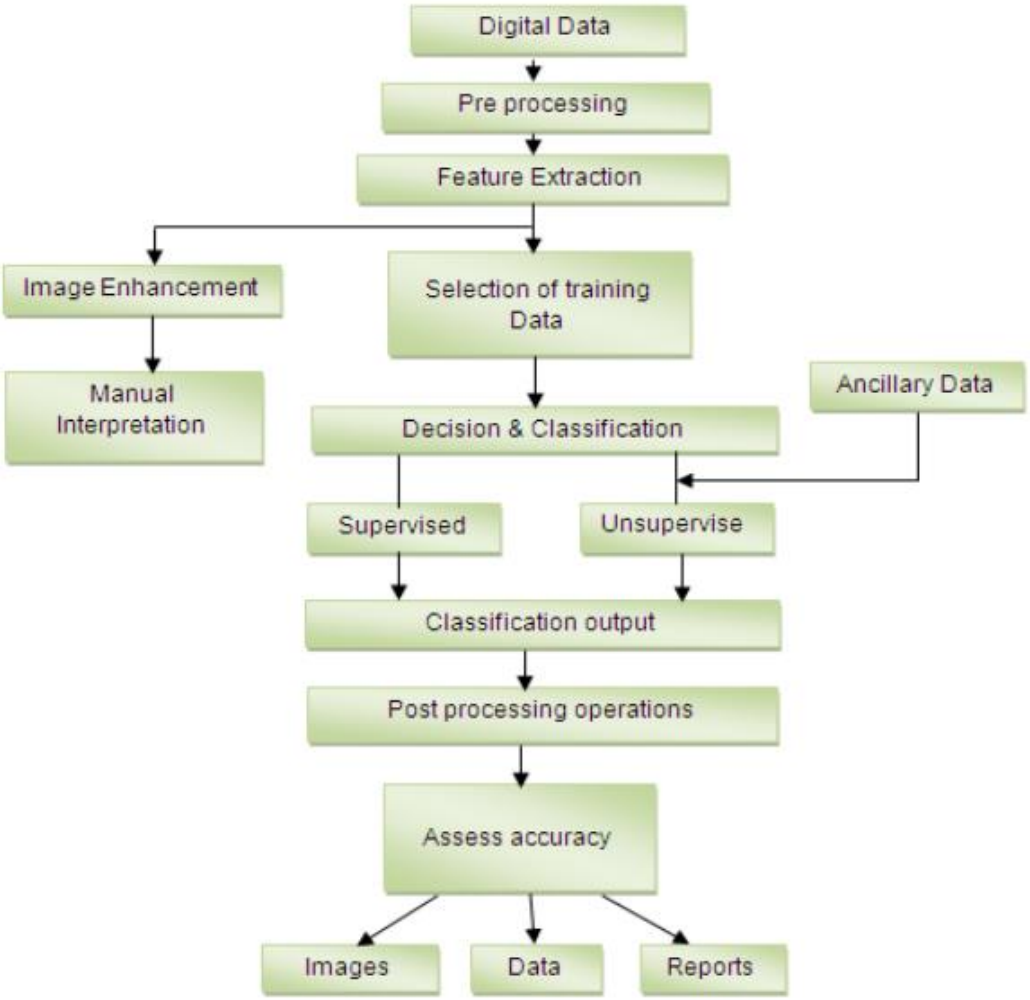
Τέλος, ορίζουμε μία βοηθητική συνάρτηση η οποία παίρνει ένα διάνυσμα στήλη που περιγράφει τη θέση, ένα τεταρδόνιο και ένα διάνυσμα στήλη που περιγράφει το μέγεθος και επιστρέφει τον πίνακα μετασχηματισμού ο οποίος περιγράφει το μετασχηματισμό αυτό.

$$TRS(p, q, s) = M$$

όπου M 4x4 πίνακας μετασχηματισμού ο οποίος επεξηγήθηκε στις προηγούμενες ενότητες.

1.3 Επεξεργασία εικόνας [24]

Στην ενότητα αυτή παρουσιάζεται ο τομέας της Επεξεργασίας Εικόνας. Συγκεκριμένα δίνεται ένας ορισμός, ο σκοπός της ύπαρξης αυτού του επιστημονικού πεδίου και η σημασία του στην υπολογιστική όραση, καθώς και η χρήση της σε περιβάλλοντα εκτεταμένης πραγματικότητας.



ΕΙΚΟΝΑ 3 ΔΙΑΓΡΑΜΜΑ ΡΟΗΣ ΕΝΟΣ ΓΕΝΙΚΟΥ ΣΥΣΤΗΜΑΤΟΣ ΕΠΕΞΕΡΓΑΣΙΑΣ ΕΙΚΟΝΑΣ

1.3.1 Ορισμός-σύντομη περιγραφή

Η επεξεργασία εικόνας είναι μία μέθοδος κατά την οποία μία εικόνα μετατρέπεται σε ψηφιακή μορφή και έπειτα επεξεργάζεται ώστε να προκύψει μία βελτιωμένη εκδοχή της ή να εξαχθεί χρήσιμη πληροφορία από αυτή. Ως εκ τούτων, ορίζουμε ένα σύστημα επεξεργασίας εικόνας το οποίο έχει ως είσοδο μία εικόνα – όπως λόγου χάρη μία φωτογραφία ή ένα στιγμιότυπο από ένα βίντεο – και ως έξοδο δύναται να έχει πάλι εικόνα ή χαρακτηριστικά που έχουν εξαχθεί από την αρχική εικόνα. Συνηθέστερα ένα τέτοιο σύστημα δέχεται την εικόνα ως ένα δισδιάστατο σήμα και την επεξεργάζεται με διάφορες μεθόδους.

Παρότι υπάρχουν και οι έννοιες της οπτικής και αναλογικής επεξεργασίας εικόνας, εδώ αναφερόμαστε εξ ολοκλήρου στην ψηφιακή.

Τα διάφορα συστήματα του πεδίου αυτού μπορούν να συνοψιστούν ως εξής:

- Επεξεργασία Εικόνας (είσοδος εικόνα => έξοδος εικόνα)
- Ανάλυση Εικόνας (είσοδος εικόνα => εξαγωγή μετρικών)
- Κατανόηση εικόνας (είσοδος εικόνα => σημασιολογική ερμηνεία)

Από την άλλη, ο σκοπός των συστημάτων αυτών μπορεί να κατηγοριοποιηθεί σε 5 διαφορετικές ομάδες

1) Απεικόνιση - Η δυνατότητα παρατήρησης αντικειμένων τα οποία δεν είναι ορατά
2) Βελτίωση και αποκατάσταση εικόνας – η δημιουργία μίας καλύτερης από την αρχική εικόνας
3) Ανάκτηση εικόνας - Η εύρεση μιας εικόνας υψηλού ενδιαφέροντος
4) Αναγνώριση προτύπων
5) Αναγνώριση Εικόνας – Κατηγοριοποίηση και αναγνώριση διάφορων αντικειμένων μέσα σε μία εικόνα.

1.3.2 Ανίχνευση ακμών [25]

Ένα από τα πολύ αναπτυγμένα πεδία της Επεξεργασίας Εικόνας είναι η ανίχνευση ακμών. Με τον όρο ανίχνευση ακμών εννοούμε τον εντοπισμό σημαντικών παραλλαγών σε μια ασπρόμαυρη εικόνα και η κατηγοριοποίηση των φυσικών φαινομένων που προκάλεσαν τις παραλλαγές αυτές. Η πληροφορία αυτή είναι πολλή σημαντική σε ποικίλα συστήματα, όπως τρισδιάστατη ανακατασκευή, κίνηση, αναγνώριση, βελτίωση, αποκατάσταση κ.ο.κ

Συγκεκριμένα, στα πλαίσια της διπλωματικής θα δούμε πως οι αλγόριθμοι που χρησιμοποιούν διάφορα συστήματα επαυξημένης – μικτής πραγματικότητας βασίζονται εσωτερικά σε κάποιου είδους ανίχνευσης σημείων ενδιαφέροντος για να προσδιορίσουν τη θέση της συσκευής στο χώρο.

ΕΙΚΟΝΑ 4 ΠΡΩΤΟΤΥΠΗ ΕΙΚΟΝΑ

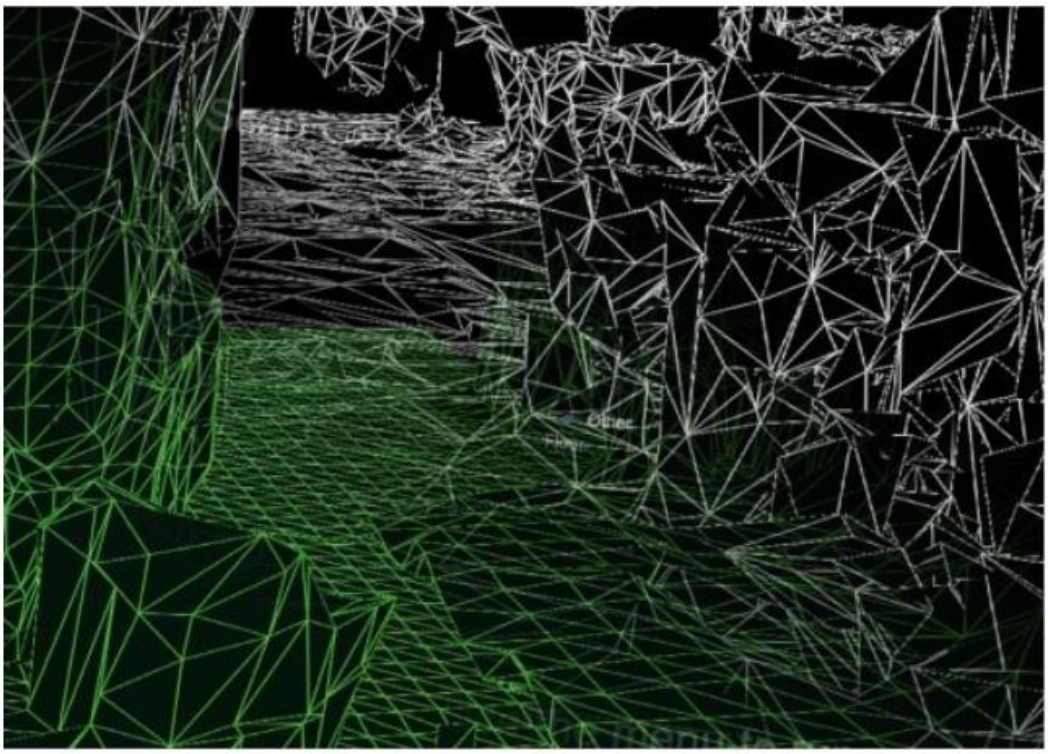
ΕΙΚΟΝΑ 5 ΑΠΟΤΕΛΕΣΜΑ ΑΝΙΧΝΕΥΣΗΣ

1.3.3 Υπολογιστική Όραση

Αποτελεί υψηλού επιπέδου επεξεργασία εικόνας κατά την οποία ένα μηχάνημα / υπολογιστής / λογισμικού προσπαθεί να αποκρυπτογραφήσει τα φυσικά περιεχόμενα μιας εικόνα. (παραδείγματος χάρη βίντεο ή τρισδιάστατες μαγνητικές τομογραφίες)

Είναι πλέον σημαντικός όρος για τα περιβάλλοντα μικτής πραγματικότητας, όπου η συσκευή πρέπει να αναγνωρίσει φυσικά στοιχεία του περιβάλλοντος ώστε ο εικονικός κόσμος να μπορεί να αλληλοεπιδρά με το φυσικό. Ουσιαστικά, μία συσκευή μικτής πραγματικότητας πρέπει:

- Να σαρώνει συνεχώς το περιβάλλον με σκοπό τη δημιουργία ενός σύννεφου σημείων (*point cloud*) ή αλλιώς πλέγματος (*mesh*) το οποίο είναι η εικονική αναπαράσταση του πραγματικού κόσμου στο περιβάλλον της συσκευής.
- Εκτός από τη εικονική αναπαράσταση του περιβάλλοντος, πρέπει να γίνεται σάρωση και αναγνώριση κινήσεων των χεριών του χρήστη, καθώς και χειρονομιών, ώστε να είναι ικανός να αλληλοεπιδράσει με το περιβάλλον.

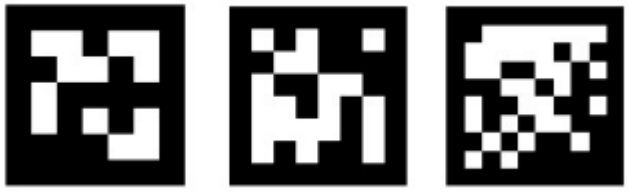


ΕΙΚΟΝΑ 6 ΤΟ ΠΛΕΓΜΑ ΠΟΥ ΔΗΜΙΟΥΡΓΕΙΤΑΙ ΜΕΤΑ ΤΗ ΣΑΡΩΣΗ ΤΟΥ ΠΕΡΙΒΑΛΛΟΝΤΟΣ [26]

1.3.4 ArUco – Ανίχνευση *δείκτη (marker)* [27]

Η βιβλιοθήκη *ArUco* εστιάζει στην ανίχνευση και τον εντοπισμό διάφορων «έμπιστων» δεικτών (***fiducial markers***) με σκοπό την εκτίμηση της πόζας μιας κάμερας και του εντοπισμού ενός ρομπότ στο χώρο. Επιπλέον, μπορεί να χρησιμοποιηθεί και ως απλή ανίχνευση μιας εικόνας όταν βρίσκεται στο πλαίσιο μιας κάμερας.

Ένα σύστημα εντοπισμού αυτών των δεικτών αποτελείται από ένα σύνολο από έγκυρους δείκτες και έναν αλγόριθμο ο οποίος εντοπίζει και πιθανά διορθώνει τα σφάλματα εντοπισμού σε μία εικόνα. Η συγκεκριμένη υλοποίηση βασίζεται σε τετράγωνους δείκτες οι οποίοι φέρουν κάποιον δυαδικό κωδικό. Η δημιουργία αυτών των δεικτών γίνεται αυτόματα μέσω ενός συστήματος παραγωγής ενός λεξικού.

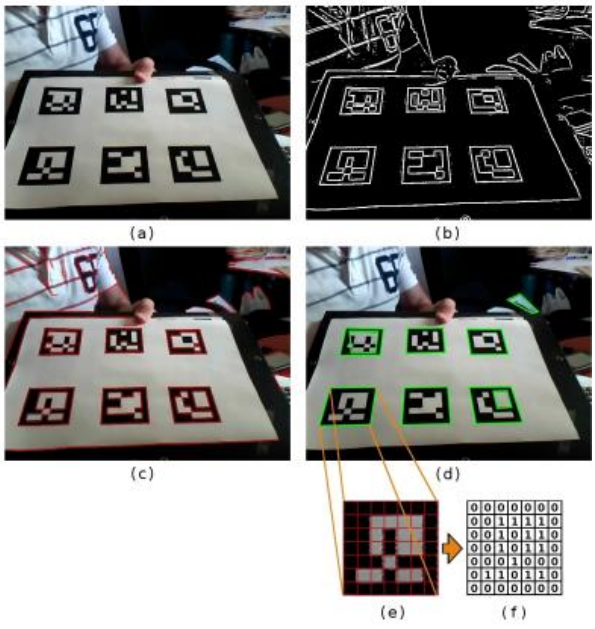


ΕΙΚΟΝΑ 7 ΠΑΡΑΔΕΙΓΜΑΤΑ ΔΕΙΚΤΩΝ ΔΙΑΦΟΡΕΤΙΚΩΝ ΜΕΓΕΘΩΝ, Ν. ΑΡΙΣΤΕΡΑ ΠΡΟΣ ΔΕΞΙΑ: Ν= 5, Ν=6, Ν=8

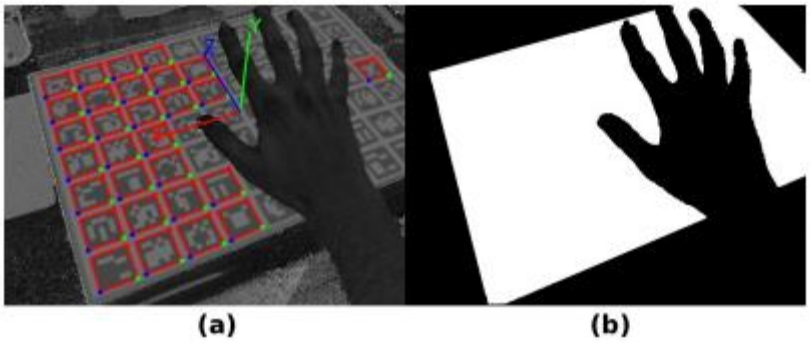
Η είσοδος είναι μία εικόνα σε γκρι κλίμακα και τα βασικά βήματα του αλγορίθμου είναι τα εξής:

1) ***Τμηματοποίηση εικόνας (Image Segmentation),*** όπου εξάγονται τα κύρια περιγράμματα της εικόνας. Η εξαγωγή γίνεται με προσαρμοστική προσέγγιση κατωφλίου για γρηγορότερα αποτελέσματα.
2) ***Εξαγωγή περιγραμμάτων και φιλτράρισμα (Contour extraction and filter),*** όπου εξάγονται πλήρως τα περιγράμματα. Ανάλογα με τα *bits* που χρησιμοποιούνται δίνεται η δυνατότητα καλύτερης διόρθωσης σφαλμάτων. Εδώ εντοπίζονται οι διάφοροι δείκτες οι οποίοι είναι ορατοί στο πλαίσιο της κάμερας.
3) ***Εξαγωγή των bits του αναγνωριστικού (Marker Code Extraction),*** όπου αναλύονται οι εσωτερικές περιοχές των περιγραμμάτων για την αποκωδικοποίηση του αναγνωριστικού.
4) ***Ανίχνευση του δείκτη και διόρθωση σφαλμάτων (Marker detection and error correction),*** όπου καθορίζεται αν ο δείκτης που ανιχνεύθηκε ανήκει στο λεξικό που δόθηκε στον αλγόριθμο. Για γρηγορότερη ανίχνευση, το λεξικό είναι ταξινομημένο ως ένα ισορροπημένο δυαδικό δένδρο.
5) ***Βελτίωση γωνιών και εκτίμηση πόζας (Corner refinement and pose estimation),*** όπου γίνεται η εκτίμηση της πόζας του δείκτη σε σχέση με την κάμερα. Για την εκτίμηση των γωνιών χρησιμοποιείται γραμμική παλινδρόμηση των ακριανών εικονοστοιχείων του δείκτη για τον υπολογισμό των διασταυρώσεων.

Ο αλγόριθμος αυτός βοηθάει και στην περίπτωση που κάποιο δείκτες δε φαίνονται, αφού είναι δυνατή η ανίχνευση παραπάνω από του ενός δείκτη. Συνεπώς, όχι μόνο γίνεται να εντοπίσουμε την πόζα κάθε δείκτη ξεχωριστά, αλλά και να εκτιμήσουμε την πόζα ενός συνόλου δεκτών, κάνοντας το σύστημα πιο εύρωστο σε περιπτώσεις που δε φαίνονται κάποιοι δείκτες.



ΕΙΚΟΝΑ 8 ΕΞΑΓΩΓΗ ΔΕΙΚΤΩΝ



ΕΙΚΟΝΑ 9 ΕΥΡΩΣΤΗ ΑΝΙΧΝΕΥΣΗ

1.4 Οπτική Αδρανειακή Οδομετρία – Ταυτόχρονος Εντοπισμός και Χαρτογράφηση

(*Visual Inertial Odometry [VIO] – Simultaneous Localization and Mapping [SLAM]*)

Η ενότητα αυτή παρουσιάζει τις έννοιες Οπτική Αδρανειακή Οδομετρία (*VIO*) και Ταυτόχρονος Εντοπισμός και Χαρτογράφηση (*SLAM).* Είναι ιδιαίτερα σημαντικές έννοιες όσον αφορά την επαυξημένη και μικτή πραγματικότητα, διότι χωρίς αυτές, η συσκευή δε θα είχε τρόπο να αναγνωρίσει τη θέση της και τον προσανατολισμό της σε σχέση με τον εικονικό – μικτό κόσμο που δημιουργείται. Επιπλέον, χρησιμοποιήθηκαν στη διπλωματική εκτός της ίδιας της συσκευής, στον εξωτερικό ελεγκτή για την ανάπτυξη ενός συστήματος που παρέχει 6 βαθμούς ελευθερίας στο χρήστη.

Λόγω του ότι είναι ταχέα αναπτυσσόμενα πεδία και υπάρχουν πολλές και διαφορετικές τεχνικές, θα εστιάσουμε περισσότερο σε μια γενική εποπτεία των όρων αυτών, ώστε να γίνει αντιληπτή περισσότερο η φιλοσοφία των τεχνικών αυτών, παρά η υλοποίηση καθαυτή.

1.4.1 Οδομετρία (*Odometry*)

Ως οδομετρία ορίζεται η χρήση δεδομένων τα οποία έχουν συλλεχθεί από αισθητήρες κίνησης ώστε να γίνει μία πρόβλεψη στη μεταβολή της θέσης ενός αντικειμένου. Ένα απλό οδόμετρο υπάρχει σε κάθε αμάξι, το οποίο μετρά τα συνολικά χιλιόμετρα που έχει διανύσει. Επιπλέον, βρίσκει χρήση στη ρομποτική σε δίποδα ή τροχοφόρα ρομπότ ώστε να γίνεται μία εκτίμηση της θέσης του ρομπότ σε σχέση με ένα αρχικό σημείο αναφοράς.

Μάλιστα, όσον φορά τη ρομποτική, η μέθοδος είναι ευαίσθητη στα σφάλματα λόγω της ολοκλήρωσης που απαιτείται από μετρήσεις τις επιτάχυνσης ή της ταχύτητας, που λόγω θορύβου δε δίνει σωστά αποτελέσματα. Μία καλή υλοποίηση της οδομετρίας απαιτεί τη συλλογή των μετρήσεων των αισθητήρων με ακρίβεια, τη σωστή βαθμονόμηση των οργάνων και την καλή επεξεργασία των δεδομένων.

Η εκτίμηση της θέσης και του προσανατολισμού μιας συσκευής στο χώρο με βάση κάποιο σημείο αναφοράς ονομάζεται ***ego-motion***.

1.4.2 Οπτική Οδομετρία (*Visual Odometry*) [28]–[30]

Με τον όρο **οπτική οδομετρία** αναφερόμαστε σε διάφορες τεχνικές οδομετρίας οι οποίες ως βάση έχουν την εξαγωγή στοιχείων μέσα από βίντεο (με μία ή περισσότερες κάμερες), ούτως ώστε με τη χρήση διάφορων τεχνικών επεξεργασίας εικόνας και υπολογιστικής όρασης, να προκύψει μία καλή εκτίμηση της θέσης και του προσανατολισμού - που τυπικά ονομάζουμε **στάση, πόζα** (**pose**) - του συστήματος με βάση ένα αρχικό σημείο αναφοράς.

Τέτοιες τεχνικές φέρουν πολύ καλά αποτελέσματα όσον αφορά την ακρίβεια, αλλά δεν είναι πάντα εύρωστες, διότι εξαρτώνται σημαντικά από τον εντοπισμό σημείων ενδιαφέροντος

(*feature points*) στην εικόνα-βίντεο. Αν λοιπόν τέτοια σημεία δε μπορούν να εντοπιστούν στο βίντεο - λόγου χάρη η κάμερα να κοιτάει έναν άσπρο τοίχο χωρίς ιδιαίτερα χαρακτηριστικά - τότε οι μέθοδοι αυτοί αδυνατούν να δώσουν ένα καλό αποτέλεσμα.

Η οπτική οδομετρία χωρίζεται *κυρίως* σε 2 διαφορετικούς τύπους:

- ***Monocular***, όπου το σύστημα περιέχει μόνο μία κάμερα.
- **Stereo**, όπου το σύστημα έχει δύο κάμερες.

Επιπλέον, υπάρχουν δύο κύριες κατηγορίες τεχνικών:

- **Βάσει σημείων ενδιαφέροντος (*Feature-Based*)**, όπου τα σημεία ενδιαφέροντος εξάγονται από το βίντεο και έπειτα εντοπίζονται μέσα στο βίντεο.
- **Άμεση μέθοδος (*Direct method*)**, όπου χρησιμοποιείται ή ένταση του κάθε εικονοστοιχείου (*pixel*) ως είσοδος στον αλγόριθμο.

Υπάρχουν και υβριδικές μέθοδοι που συνδυάζουν τους δύο τρόπους.

Ένας γενικός αλγόριθμος οπτικής οδομετρίας βάσει σημείων ενδιαφέροντος παρουσιάζεται παρακάτω. Είναι σημαντικό όμως να τονίσουμε πως το πεδίο είναι ραγδαία εξελισσόμενο και είναι αρκετά δύσκολη η κατηγοριοποίηση και παρουσίαση όλων των αλγορίθμων:

1) Εξαγωγή εικόνων εισόδου μέσω μονής κάμερας, *stereo* κάμερας ή παγκατευθυντικής (*omnidirectional*) κάμερας
2) Διόρθωση εικόνας: επεξεργασία της εικόνας ώστε να αφαιρεθούν διάφορες παραμορφώσεις, όπως η παραμόρφωση από το φακό
3) Ανίχνευση σημείων ενδιαφέροντος: Ορισμός τελεστών ενδιαφέροντος, η συσχέτιση των σημείων ενδιαφέροντος μεταξύ των διαφορετικών εικόνων (*frames*) στο βίντεο και η δημιουργία του πεδίου οπτικής ροής (*optical flow, Lucas-Kanade* [31]*)*
4) Ανίχνευση σφαλμάτων και διόρθωση με χρήση του πεδίου οπτικής ροής
5) Εκτίμηση της κίνησης της κάμερας από την οπτική ροή
   i. Είτε με φίλτρο Κάλμαν
   ii. Είτε με την ελαχιστοποίηση μιας συνάρτησης κόστους με σκοπό να βρεθούν οι γεωμετρικές και τρισδιάστατες ιδιότητες των σημείων ενδιαφέροντος.
6) Περιοδική ανασύσταση των εντοπισμένων σημείων ώστε να διατηρηθεί η ανίχνευση σε όλη την εικόνα.

FIGURE 1 MONOCULAR VIO FLOW CHART EXAMPLE[32]

1.4.3 Αδρανειακή Οδομετρία (Inertial Odometry) [33], [34]

Αδρανειακή Πλοήγηση (*Inertial Navigation*)

Με τον όρο **αδρανειακή οδομετρία** αναφερόμαστε στις τεχνικές οι οποίες ως βάση έχουν την επεξεργασία δεδομένων από διάφορους αισθητήρες αδράνειας (επιταχυνσιόμετρο, γυροσκόπιο, μαγνητόμετρο) ούτως ώστε να πάλι να προκύψει μία εκτίμηση της θέσης και του προσανατολισμού της συσκευής στο χώρο με βάση ένα αρχικό σημείο αναφοράς.

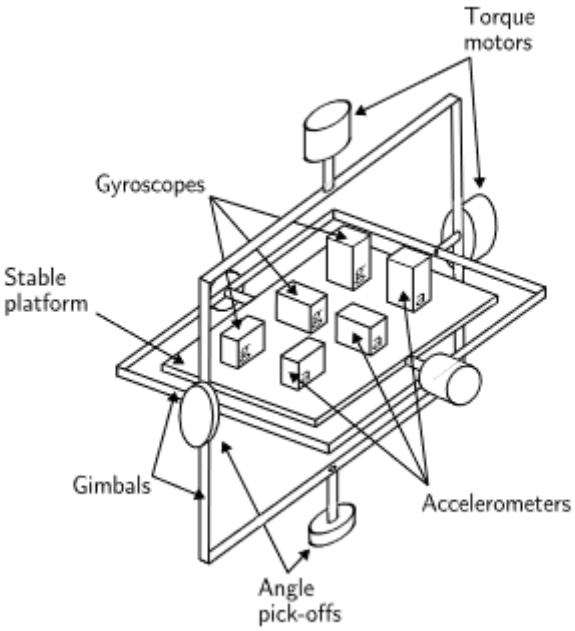
Τέτοιες τεχνικές είναι εύρωστες, διότι οι αισθητήρες παρέχουν συνεχώς δεδομένα, αλλά έχουν συνήθως μικρή ακρίβεια λόγω του θορύβου των μετρήσεων. Ένα κλασικό και εύκολο

παράδειγμα για να καταλάβουμε τη σημασία του θορύβου και την ανάγκη αφαίρεσής του - όσο το δυνατόν γίνεται-είναι το εξής:

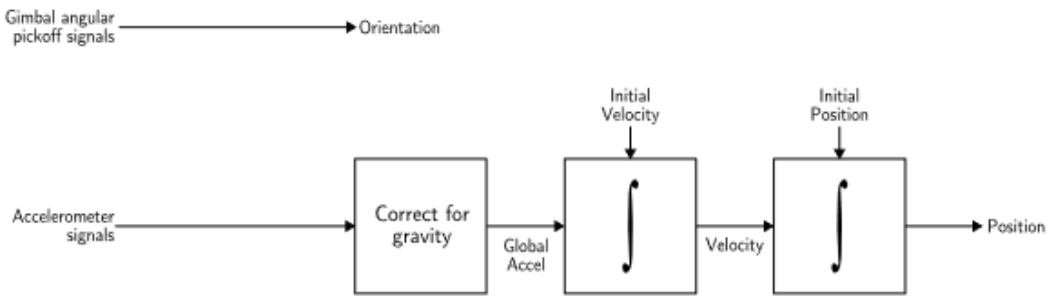
Έχοντας κάποιες μετρήσεις από ένα επιταχυνσιόμετρο και μία αρχική θέση θέλω να βρω τη θέση της συσκευής στο χώρο.

Μία άμεση λύση είναι η αριθμητική ολοκλήρωση της επιτάχυνσης ώστε να βρούμε την ταχύτητα και η αριθμητική ολοκλήρωση της ταχύτητας ώστε να βρούμε τη θέση. Ενώ η λύση θεωρητικά είναι άρτια, δε λαμβάνει υπόψιν τον θόρυβο που εισάγει στο σύστημα ένα επιταχυνσιόμετρο. Ένα τέτοιο όργανο, όσο καλό και να είναι, πάντα θα παρουσιάζει κάποιο θόρυβο στις μετρήσεις του. Αποτέλεσμα είναι να ολοκληρώνεται και ο θόρυβος - και μάλιστα στο τετράγωνο - και να έχουμε τη λεγόμενη **ολίσθηση** (*drift*), όπου η θέση στην αρχή φαίνεται να υπολογίζεται σωστά, αλλά μετά μετακινείται προς κάποια κατεύθυνση επιταχύνοντας ολοένα και περισσότερο. Εν ολίγης, ένα τέτοιο σύστημα δεν είναι ευσταθές και χρειάζεται επεξεργασία των μετρήσεων, φιλτράρισμα και καλή βαθμονόμηση ώστε να προκύψουν τα καλύτερα δυνατά αποτελέσματα.

Τα συστήματα αυτά υλοποιούνται με τη χρήση μίας **Μονάδας Αδρανειακής Μέτρησης** (***IMU***). Μία τέτοια μονάδα αποτελείται τυπικά από διάφορους αδρανειακούς αισθητήρες οι οποίο βασίζονται στην τεχνολογία *MEMS*. Παρακάτω απεικονίζεται σε σχεδιάγραμμα ένα τυπικό *IMU* καθώς και ένας απλουστευμένος αλγόριθμος της αδρανειακής οδομετρίας.



ΕΙΚΟΝΑ 10 STABLE PLATFORM IMU [31]

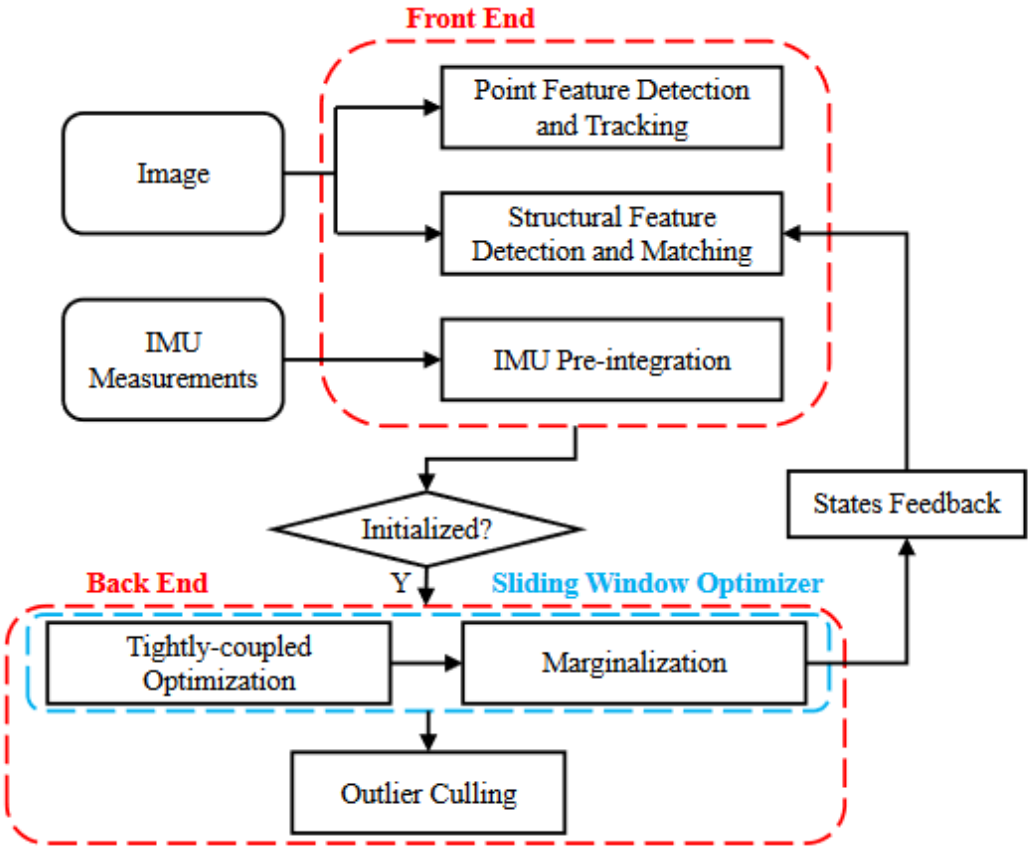


ΕΙΚΟΝΑ 11 ΣΥΣΤΗΜΑ ΑΔΡΑΝΕΙΑΚΗΣ ΠΛΟΗΓΗΣΗΣ [34]

1.4.4 Οπτική Αδρανειακή Οδομετρία (*Visual Inertial Odometry*) [28]

Όπως γίνεται εύκολα αντιληπτό από το όνομα, η οπτική αδρανειακή οδομετρία συνδυάζει την οπτική οδομετρία με την αδρανειακή πλοήγηση σε μια προσπάθεια εξάλειψης των αδυναμιών των δύο αυτών τεχνικών.

Με χρήση και των 2, το σύστημα γίνεται πιο εύρωστο και πιο ακριβές.

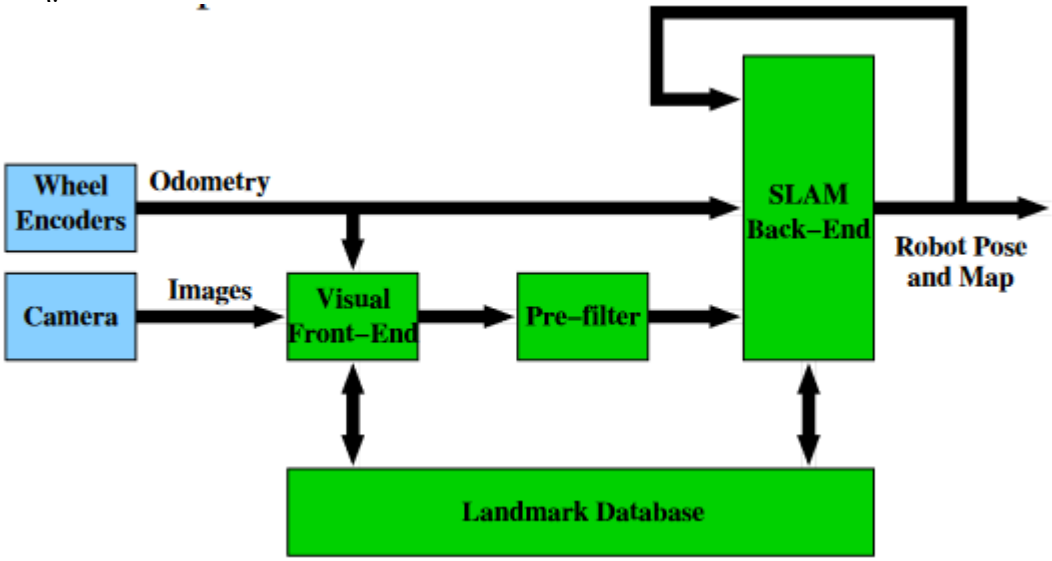


1.4.5 Ταυτόχρονος Εντοπισμός και Χαρτογράφηση (*Simultaneous Localization and Mapping*) – *SLAM* [35], [36]

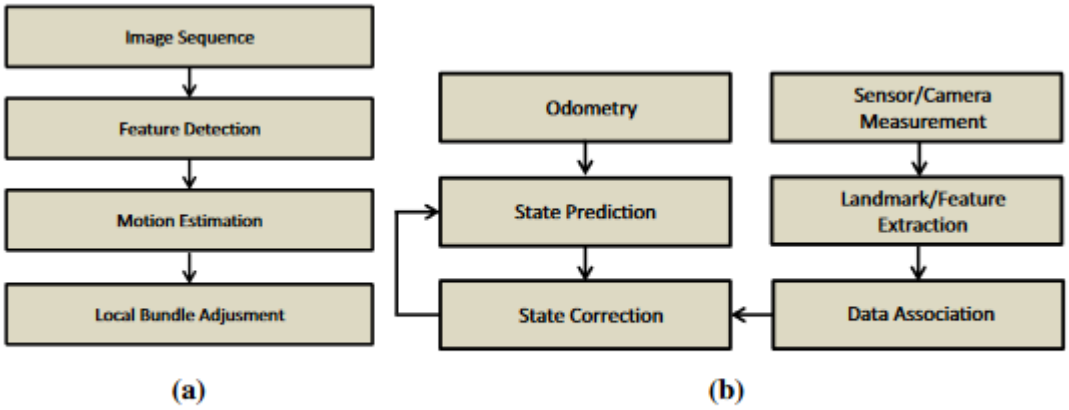
Οι παραπάνω τεχνικές είναι αρκετά επιτυχείς στο να προσδιορίσουν τη θέση και τον προσανατολισμό μέσω κυρίως της μεταβολής διάφορων ποσοτήτων, είτε αυτές είναι θέσεις των σημείων ενδιαφέροντος είτε η θέση που προέκυψε από την επεξεργασία των μετρήσεων του επιταχυνσιόμετρου. Οι τεχνικές αυτές περισσότερο πετυχαίνουν μια τοπική συνέχεια στη διαδρομή που φαίνεται να ακολουθεί η συσκευή.

Σκοπός του **SLAM** είναι η χαρτογράφηση του χώρου. Για να αποκτήσει πλήρη αυτονομία ένα ρομπότ πρέπει να έχει την ικανότητα να εξερευνά το χώρο χωρίς εξωτερική παρέμβαση από κάποιον χρήστη. Για να επιτευχθεί αυτό, το ρομπότ δημιουργεί έναν αξιόπιστο χάρτη και εντοπίζει τον εαυτό του στον χάρτη.

Η έννοια του ***vSLAM*** (*visual SLAM*) συνδέεται άμεσα με το *VO-VIO*. Αυτό συμβαίνει διότι οι αλγόριθμοι που αναπτύχθηκαν για την οδομετρία ενός ρομπότ ή μιας οποιασδήποτε συσκευής μπορούν να επεκταθούν ώστε να χαρτογραφήσουν το χώρο. Ταυτόχρονα όμως, αυτή η χαρτογράφηση βοηθά και το υπόλοιπο σύστημα να διορθώσει τυχόν σφάλματα που προέκυψαν από την οδομετρία. Ούτως ή άλλως, τα σημεία ενδιαφέροντος τα οποία εντοπίζει το *VIO* σύστημα αποτελούν σημεία ενδιαφέροντος που μπορεί να αποθηκεύσει το *vSLAM* σύστημα.



ΕΙΚΟΝΑ 12 BLOCK ΔΙΑΓΡΑΜΜΑ ΕΝΟΣ ΣΥΣΤΗΜΑΤΟΣ VSLAM [36]



ΕΙΚΟΝΑ 13 BLOCK ΔΙΑΓΡΑΜΜΑ ΓΙΑ (Α) VO ΚΑΙ (Β) SLAM

1.4.5 Εφαρμογές

Τα παραπάνω χρησιμοποιούνται σε διάφορες εφαρμογές. Μερικές από αυτές είναι:

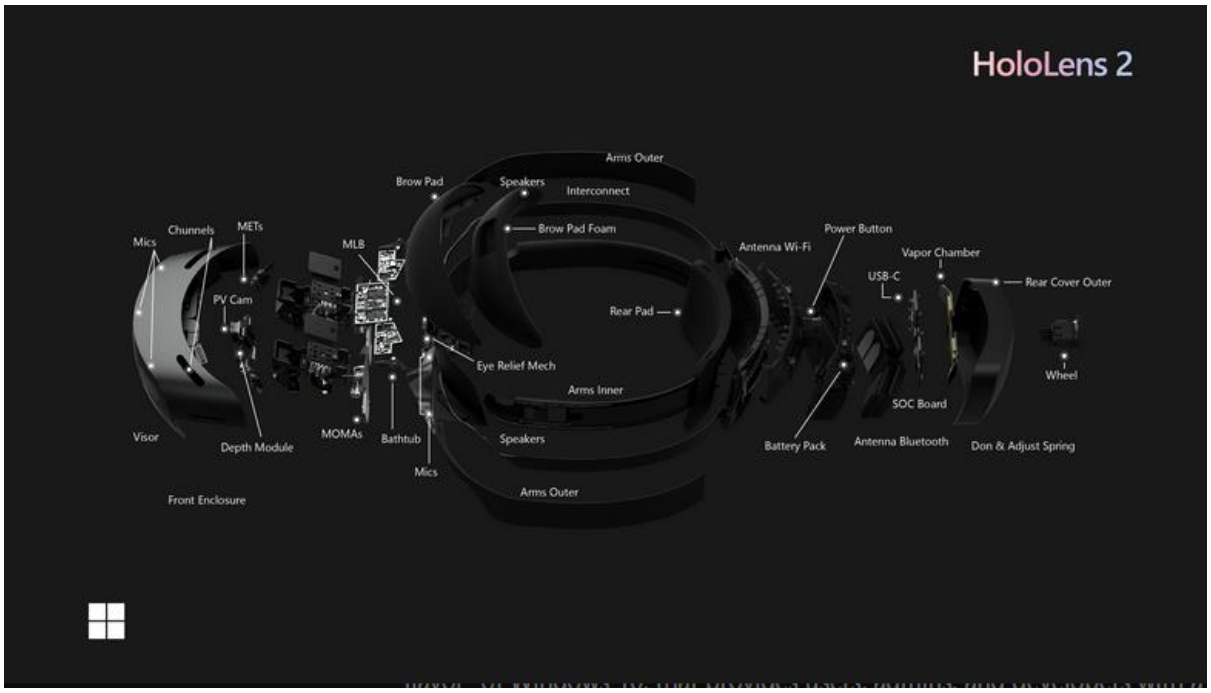
1) Στο *Mars Rovers*, όπου το αυτοκινούμενο όχημα χρησιμοποιεί τους αλγορίθμους ώστε να εξερευνήσει τον Άρη αυτόνομα. [30]
2) Στην εύρεση της τοποθεσίας μίας συσκευής σε περιπτώσεις όπου δεν υπάρχει καλό σήμα GPS. [37]
3) Στα έξυπνα κινητά με σκοπό τις εφαρμογές επαυξημένης πραγματικότητας.
4) Στα *HMDs* μικτής πραγματικότητας ώστε να γνωρίζουν τη θέση και τον προσανατολισμό τους στο χώρο. Ένα από αυτά τα *HMDs* είναι και το *Hololens.*

1.5 Microsoft Hololens 2 [38]

Τα *Hololens* είναι μια σειρά έξυπνων γυαλιών μικτής πραγματικότητας, τα οποία και παράγει η *Microsoft*. Ξεκίνησαν με τα *Hololens 1* το 2016 και το 2019 εξέδωσαν μία νέα έκδοση, τα *Hololens 2*, τα οποία χρησιμοποιούνται μέχρι και σήμερα. Δυστυχώς τα *Hololens 1* πλέον δεν έχουν ιδιαίτερη υποστήριξη από τη *Microsoft* ως προς το λογισμικό ανάπτυξης εφαρμογών.

Στην ενότητα αυτή θα γίνει μία σύντομη περιγραφή των σημαντικότερων εξαρτημάτων της συσκευής καθώς και των εξαρτημάτων τα οποία είναι ιδιαίτερης σημασίας λόγω του θέματος της διπλωματικής. Επιπλέον, καταγράφονται οι διάφοροι τρόποι αλληλεπίδρασης τους οποίους προσφέρει η συσκευή στο χρήστη.

1.5.1 Περιγραφή συσκευής



ΕΙΚΟΝΑ 14 ΑΠΟΔΟΜΗΣΗ ΤΗΣ ΣΥΣΚΕΥΗΣ

Το *Microsoft HoloLens 2* είναι ένα *HMD* το οποίο λειτουργεί ως ολογραφικός υπολογιστής. Είναι η εξέλιξη του *HoloLens 1*, με σκοπό να κάνει την εμπειρία πιο άνετη και πιο συναρπαστική, ώστε ο χρήστης να πιστεύει πως όντως βρίσκεται σε έναν εκτεταμένο κόσμο. Το λειτουργικό του είναι τα *Windows Holographic OS*, το οποίο βασίζεται στα *Windows 10*, δίνοντάς έτσι στον σχεδιαστή μια πλατφόρμα εύρωστη, ασφαλή και αποδοτική.

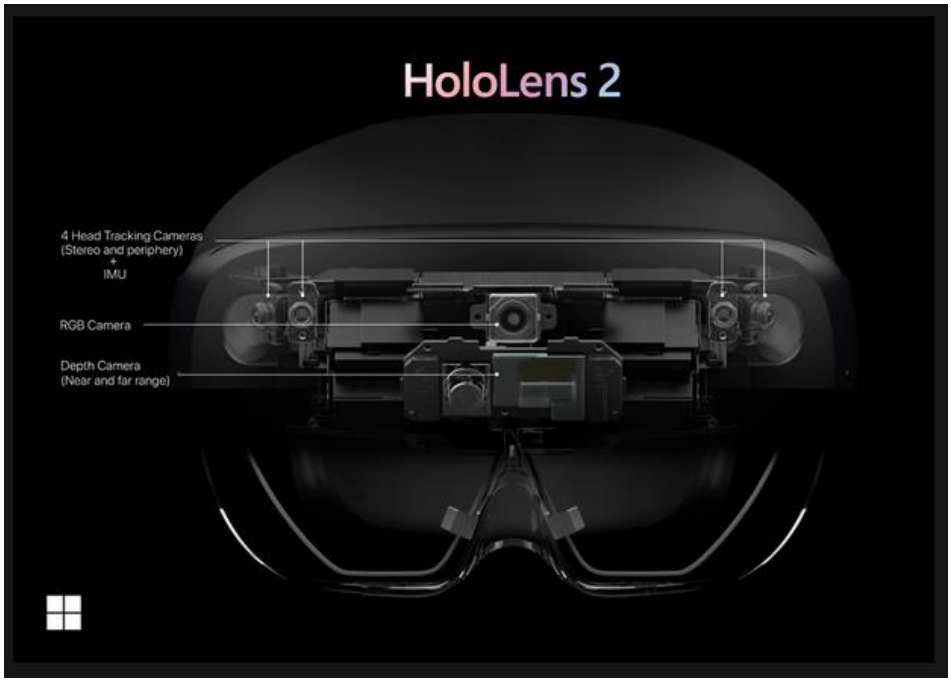
1.5.2 Βασικά εξαρτήματα

- ***Visor*:** Η λεγόμενη προσωπίδα. Περιέχει τους αισθητήρες και τις οθόνες. Δίνεται η δυνατότητα περιστροφής της ακόμα και όταν φοράει κάποιος χρήστης τη συσκευή.
- ***Headband:*** Μία ζώνη που περιβάλλει το κεφάλι και χρησιμοποιείται για να στηρίζεται η συσκευή πάνω σε αυτό. Ο χρήστης μπορεί να χρησιμοποιήσει μία ροδέλα που βρίσκεται στο πίσω μέρος της ώστε να σφίξει ή χαλαρώσει τη ζώνη αυτή, ώστε να τη νιώθει άνετα.
- ***Brightness buttons*:** Όταν ο χρήστης φοράει τη συσκευή, τα κουμπιά που ρυθμίζουν τη φωτεινότητα βρίσκονται στην αριστερή μεριά του *headband* κοντά στον κρόταφό του.
- ***Volume buttons*:** Με τη σειρά τους, τα κουμπιά που ρυθμίζουν την ένταση του ήχου βρίσκονται στη δεξιά μεριά του headband, κοντά στον κρόταφό του.
- ***Power button*:** Το κουμπί που ενεργοποιεί ή απενεργοποιεί τη συσκευή βρίσκεται στη δεξιά πλευρά του πίσω εξωτερικού καλύμματος.
- ***USB-C port*:** Μία θύρα *USB type C* βρίσκεται κάτω από το *power button*. Μέσω αυτής ο σχεδιαστής μπορεί να συνδέσει τη συσκευή στον υπολογιστή του εάν θέλει να ανεβάσει την εφαρμογή του.

1.5.3 Αισθητήρες

Οι αισθητήρες που βρίσκονται πάνω στη συσκευή αποτελούν πολύ σημαντικό κομμάτι της, γιατί όπως θα δούμε, είναι τα εξαρτήματα τα οποία υλοποιούν τους διάφορους αλγορίθμους επεξεργασίας εικόνας, υπολογιστικής όρασης και οπτικής οδομετρίας των οποίων η περιγραφή έγινε σε προηγούμενες ενότητες.

- ***Head tracking*:** Για τον εντοπισμό της κεφαλής, χρησιμοποιούνται 4 κάμερες ορατού φωτός. Αυτές οι κάμερες είναι που χρησιμοποιούνται για την υλοποίηση του *VIO-SLAM* ώστε η συσκευή να ξέρει τη θέση της και τον προσανατολισμός της στο χώρο.
- ***Eye tracking:*** 2 κάμερες υπέρυθρου φωτός χρησιμοποιούνται για τον εντοπισμό του βλέμματος του χρήστη.
- ***Depth:*** Ένας αισθητήρας βάθους, ο οποίος χρησιμοποιείται για τη χαρτογράφηση του χώρου.
- ***IMU*:** Μία μονάδα μέτρησης της αδράνειας, η οποία περιλαμβάνει επιταχυνσιόμετρο, γυροσκόπιο και μαγνητόμετρο. Όπως είδαμε παραπάνω, χρησιμοποιείται στην οπτική οδομετρία για τη βελτίωση της εκτίμησης της θέσης και του προσανατολισμού της συσκευής.
- ***Camera*:** Μία κάμερα με χαρακτηριστικά 8-MP, 1080p30 βίντεο.



ΕΙΚΟΝΑ 15 ΚΥΡΙΟΙ ΑΙΣΘΗΤΗΡΕΣ ΤΟΥ HOLOLENS 2

Επιπλέον, αναφέρεται πως τα ηχεία εφαρμόζουν τεχνολογία χωρικού ήχου, προσφέροντας έτσι μια πιο ρεαλιστική ακουστική εμπειρία, αφού ο ήχος δίνει την εντύπωση πως έρχεται από διάφορα σημεία στο χώρο, ακριβώς όπως συμβαίνει στην πραγματικότητα.

Τέλος, για τη χωρική χαρτογράφηση του χώρου τα *HoloLens* χρησιμοποιούν τον αισθητήρα βάθους και τις υπέρυθρες κάμερες. Ο αισθητήρας βάθους στέλνει ακτίνες στο χώρο, στις υπέρυθρες συχνότητες, οι οποίες και αντανακλώνται πάνω στα αντικείμενα. Έπειτα επιστρέφουν στα γυαλιά και συλλέγονται από τις υπέρυθρες κάμερες. Μέσω του χρόνου που έκανε να επιστρέψει μία ακτίνα, τα γυαλιά μπορούν να εκτιμήσουν την απόσταση των αντικειμένων από τον χρήστη και εν τέλει να δημιουργήσουν τον εικονικό χάρτη του περιβάλλοντος. Η διαδικασία αυτή είναι επαναληπτική.

1.5.4 Τρόποι αλληλεπίδρασης – ελέγχου του *HoloLens 2* [39]

Το *HoloLens 2* προσφέρει διάφορους τρόπους αλληλεπίδρασης με το μικτό περιβάλλον που δημιουργεί. Σε γενικές γραμμές, η χρήση μοιάζει περισσότερο με τη χρήση ενός κινητού τηλεφώνου, όπου ο χρήστης χρησιμοποιεί περισσότερο τα χέρια του και λιγότερο εξωτερικές συσκευές, όπως ένα ποντίκι ή ένα πληκτρολόγιο.

Παρακάτω παρουσιάζονται τα βασικά στοιχεία που έπειτα συνδέονται για τους διαφορετικούς τρόπους αλληλεπίδρασης.

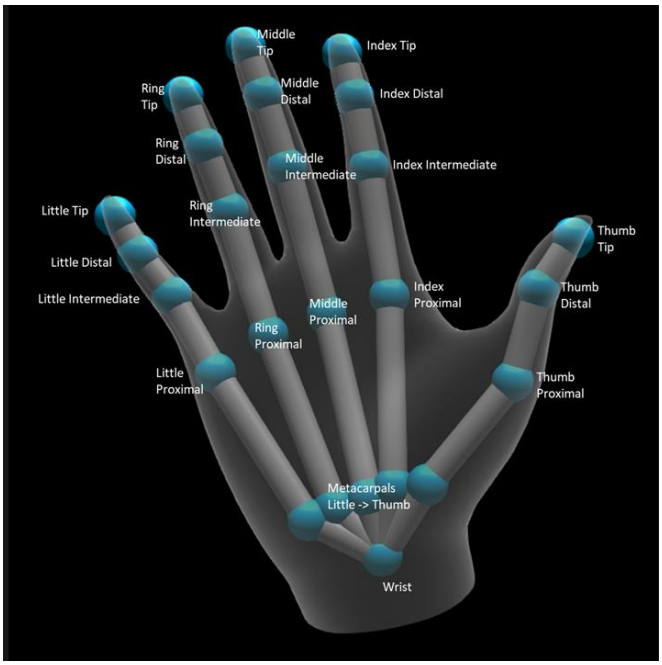
- ***Gaze*-Βλέμμα** [40], [41]**:** Το «βλέμμα» του χρήστη. Το εικονικό βλέμμα του χρήστη είναι ουσιαστικά μια εικονική κουκίδα η οποία αναγράφεται στο οπτικό του πεδίο. Η κουκίδα αυτή ελέγχεται με 2 διαφορετικούς τρόπους, και τυπικά μπορεί να θεωρηθεί πως είναι το αντίστοιχο του κέρσορα του υπολογιστή:

  - Η κουκίδα μετακινείται με βάση την κίνηση του κεφαλιού.
  - Η κουκίδα μετακινείται με βάση το που βλέπει ο χρήστης. Βέβαια σε αυτήν την περίπτωση η *Microsoft* προτείνει να μην είναι ορατή η κουκίδα, διότι η κίνησή της βάσει του βλέμματος καθιστά κουραστικό στο χρήστη το γραφικό που βλέπει.

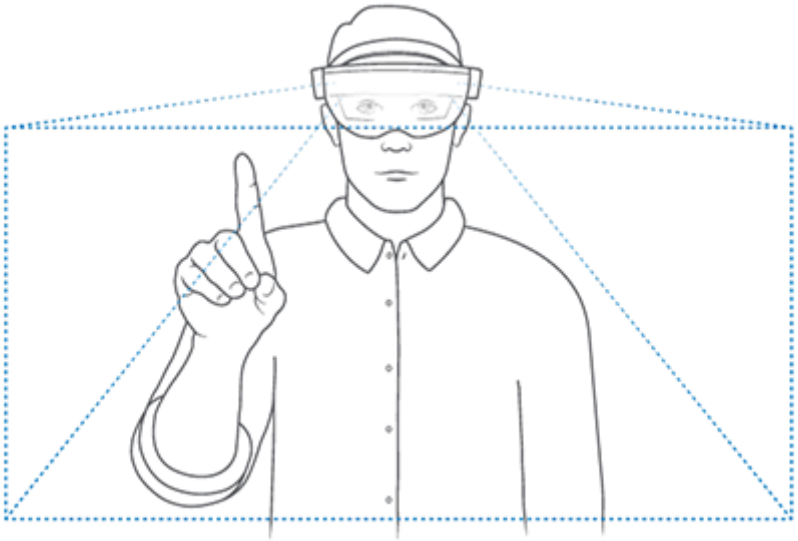
- ***Hand tracking*** [42]**:** Τα *HoloLens 2* προσφέρουν τη δυνατότητα ανίχνευσης και των 2 χεριών μέσω διάφορων τεχνικών υπολογιστικής όρασης και μηχανικής εκμάθησης. Ο χρήστης τοποθετεί τα χέρια του μπροστά από τις κάμερες της συσκευής ενώ η ίδια η συσκευή τρέχει συνεχώς έναν αλγόριθμο ανίχνευσης των χεριών. Όποτε εντοπιστεί ένα χέρι, το *HoloLens* δημιουργεί ένα πλέγμα – σύνολο σημείων στο χώρο τα οποία και αποτελούν το εικονικό χέρι. Παρότι η ανίχνευση είναι αρκετά εύρωστη, παρατηρείται μικρή καθυστέρηση στην απόκριση των κινήσεων του χρήστη.

  Το εικονικό χέρι που δημιουργείται μπορεί αν αλληλοεπιδράσει με τα εικονικά αντικείμενα με διάφορους τρόπους. Υπάρχει η δυνατότητα να τα αγγίξει, να τα περιστρέψει, να τα μετακινήσει με αρκετά οικείο τρόπο, περίπου όπως θα αντιμετώπιζε και ένα πραγματικό αντικείμενο. Με αυτόν τον τρόπο ο χρήστης μπορεί να πατήσει και σε διάφορα κουμπιά σε κάποιο μενού.

  Τέλος, πέραν από την αλληλεπίδραση από κοντά (*near interaction*), το χέρι παρέχει και τη δυνατότητα αλληλεπίδρασης από μακριά (*far interaction*). Σε αυτήν την περίπτωση εμφανίζεται μια ακτίνα σε μια προκαθορισμένη γωνία από το ανιχνευμένο χέρι με την οποία μπορεί σαν δείξεις σε απομακρυσμένα αντικείμενα.



ΕΙΚΟΝΑ 16 ΣΚΕΛΕΤΟΣ ΚΑΙ ΣΗΜΕΙΑ ΕΝΔΙΑΦΕΡΟΝΤΟΣ ΤΟΥ ΧΕΡΙΟΥ



ΕΙΚΟΝΑ 17 ΠΛΑΙΣΙΟ ΑΝΙΧΝΕΥΣΗΣ ΧΕΡΙΟΥ [43]

- **Χειρονομίες (*Gestures*):** Μέσω της ανίχνευσης του χεριού, καθώς και άλλων τεχνικών επεξεργασίας εικόνας, η συσκευή παρέχει τη δυνατότητα αναγνώρισης χειρονομιών. Παρακάτω θα αναφέρουμε τις βασικές χειρονομίες. Παρόλα αυτά, για κάποιον πιο εξειδικευμένο με την ανάπτυξη εφαρμογών στο *HoloLens 2*, υπάρχει και η δυνατότητα επέκτασης της αναγνώρισης αυτής με επιπλέον χειρονομίες.

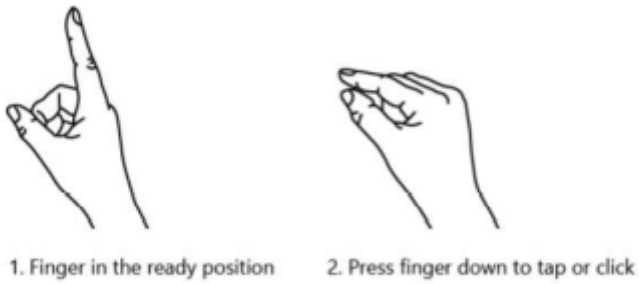
- **Ηχητικές εντολές**[44]**:** Το *HoloLens 2* παρέχει τη δυνατότητα αναγνώρισης μιας πληθώρας φωνητικών εντολών, σε αρκετές μάλιστα γλώσσες. Οι φωνητικές εντολές είναι πολύ χρήσιμες όταν ο χρήστης δε θέλει να κουράζει το χέρι του. Παρόλα αυτά, κατά τη διάρκεια της διπλωματικής παρατηρήθηκε πως μερικές φορές αναγνώριζε λέξεις στα ελληνικά ως κάποια φωνητική εντολή στα αγγλικά.

Επιπλέον, παρέχονται διάφορα γενικευμένα API ώστε να επεκταθούν αυτές οι λειτουργίες. Τα API αυτά δεν αφορούν μόνο το *HoloLens 2*, αλλά και άλλες συσκευές μικτής πραγματικότητας της *Microsoft*. [45]

Βασικές Χειρονομίες [43]

Στο *HoloLens 2* παρέχονται δύο βασικές χειρονομίες που πρέπει να γνωρίζει ο χρήστης για την περιήγηση στο μικτό περιβάλλον.

1) ***Air tap*** [39], κατά το οποίο ο χρήστης δείχνει με το δείκτη περίπου προς το ταβάνι και έπειτα να αγγίξει τον αντίχειρα με το δείκτη του, σηκώνοντάς το κατευθείαν. Αυτή η χειρονομία αντιστοιχεί στο αριστερό κλικ του ποντικιού. Υπάρχει προφανώς και η ενδιάμεση κατάσταση στην οποία ο χρήστης παραμένει στη θέση 2, σα να μην έχει απελευθερώσει το χέρι του από το αριστερό ποντίκι. Αυτό ονομάζεται ***tap and hold***.



ΕΙΚΟΝΑ 18 ΘΕΣΕΙΣ ΕΤΟΙΜΟΤΗΤΑΣ ΚΑΙ ΘΕΣΗ TAP

2) **Χειρονομία Έναρξης (*Start gesture*)**, η οποία επιτρέπει στο χρήστη να εμφανίσει το κύριο μενού καθώς και να το εξαφανίσει. Ο χρήστης πρέπει να βάλει το χέρι του στο πλαίσιο ανίχνευσης με τη μεριά της παλάμης να κοιτάει την κάμερα. Έπειτα πατάει στον καρπό του, στο σημείο όπου θα έχει εμφανιστεί ένα γραφικό που υποδηλώνει το μενού έναρξης. Έτσι το μενού εμφανίζεται ή εξαφανίζεται αναλόγως του τι κατάσταση βρισκόταν πριν. Το μενού αυτό ο χρήστης μπορεί να το διαχειριστεί πατώντας στα εικονικά κουμπιά που εμφανίζονται.

   Υπάρχει και η **χειρονομία έναρξης με το ένα χέρι**, όπου πάλι ο χρήστης πρέπει να έχει το χέρι του στο πλαίσιο ανίχνευσης με τη μεριά της παλάμης να κοιτάει την κάμερα και έπειτα να κλείσει αντίχειρα και δείκτη όπως στη χειρονομία *air tap*.

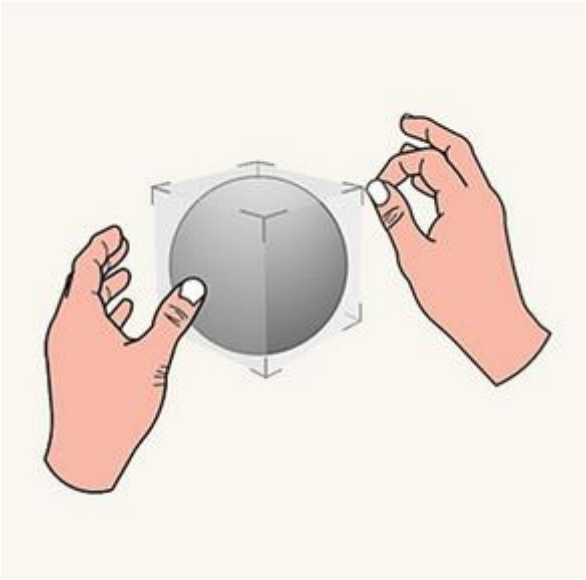
ΕΙΚΟΝΑ 19 START AND ONE HANDED START GESTURES

Δείκτες, επιλογή και βασική αλληλεπίδραση [46]

Σύμφωνα με τα παραπάνω, ο χρήστης στο *HoloLens 2* έχει τις εξής 2 κατηγορίες αλληλεπίδρασης:

1) **Χειρισμός από κοντά (*Near manipulation*)**

   Η αλληλεπίδραση αυτή ενεργοποιείται όταν το αντικείμενο με το οποίο γίνεται η αλληλεπίδραση βρίσκεται έως και περίπου 50 εκατοστά από τον χρήστη. Στην αλληλεπίδραση αυτή ο χρήστης μπορεί να πιάσει το αντικείμενο, να το μετακινήσει, να το περιστρέψει και να αλλάξει το μέγεθός του. Η ίδια αλληλεπίδραση χρησιμοποιείται ώστε ο χρήστης να πατήσει ένα κουμπί ή μία επιλογή που εμφανίζεται κοντά του.



ΕΙΚΟΝΑ 20 ΧΕΙΡΙΣΜΟΣ ΑΠΟ ΚΟΝΤΑ

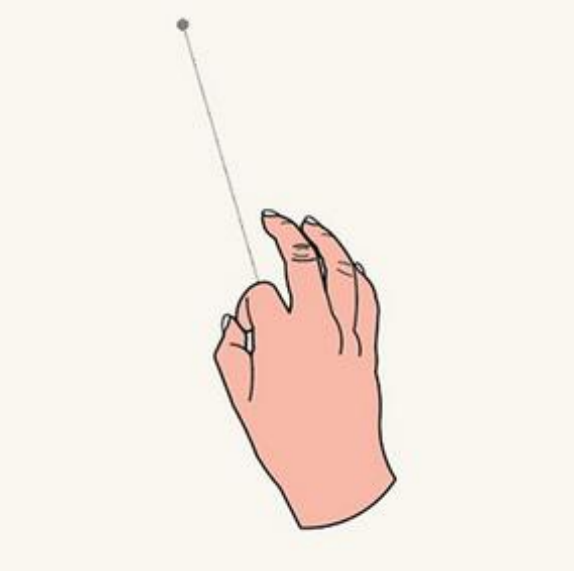
2) **Χειρισμός από μακριά (Far manipulation)**

Ο χειρισμός από μακριά ενεργοποιείται όταν το αντικείμενο με το οποίο γίνεται η αλληλεπίδραση βρίσκεται μακρύτερα των 50 εκατοστών από τον χρήστη. Στην αλληλεπίδραση αυτή έχουμε δύο τύπους δεικτών.

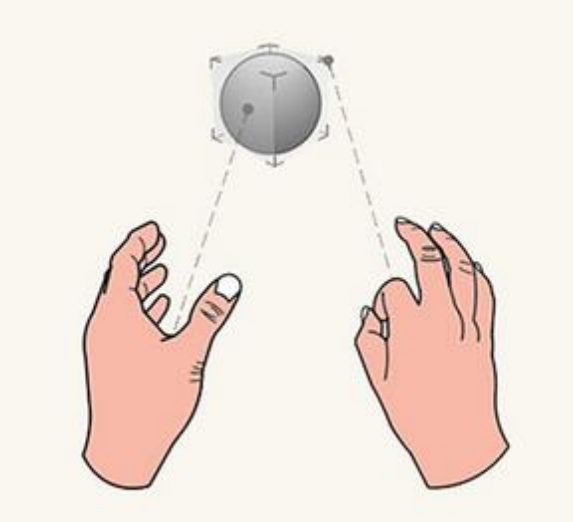
- Το κεφάλι ή το βλέμμα ως δείκτης.
- Μία ακτίνα που ξεκινά υπό μια μικρή γωνία από το κέντρο της παλάμης του χεριού και καταλήγει σε κάποιο αντικείμενο. Σημειωτέων πως εδώ μπορούν να αλληλοεπιδράσουν και τα 2 χέρια με το περιβάλλον.

Ανάλογα τώρα με το τι είδους αλληλεπίδραση είναι, ο χρήστης πάλι μπορεί να επιλέξει κάτι (με τη χρήση του *air tap*), να μετακινήσει, περιστρέφει ή αλλάξει μέγεθος σε ένα αντικείμενο καθώς και να το φέρει πιο κοντά.

ΕΙΚΟΝΑ 23 ΕΛΕΥΘΕΡΗ ΚΑΤΑΣΤΑΣΗ ΔΕΙΚΤΗ (POINTING STATE)



ΕΙΚΟΝΑ 21 ΚΑΤΑΣΤΑΣΗ ΕΠΙΚΥΡΩΣΗΣ (COMMIT STATE)



ΕΙΚΟΝΑ 22 ΧΕΙΡΙΣΜΟΣ ΑΠΟ ΚΟΝΤΑ

3) **Επιλογή με φωνητική εντολή**, κατά την οποία ο χρήστης κοιτάει μία επιλογή και την επικυρώνει τυπικά λέγοντας **Select** ή οποιαδήποτε άλλη εντολή έχει επιλέξει ο σχεδιαστής.
4) **Επιλογή με βλέμμα**, όπου ο χρήστης κοιτάει μία επιλογή για ένα χρονικό διάστημα – είτε με το δείκτη κεφαλής ή με το δείκτη βλέμματος – και η επιλογή επικυρώνεται μετά το πέρας του.

1.6 Πλατφόρμα ανάπτυξης Unity [47]

Η Unity αποτελεί μία πλατφόρμα ανάπτυξης δισδιάστατων (***2D***) και τρισδιάστατων (***3D***) εφαρμογών. Υποστηρίζει, μεταξύ άλλων, ανάπτυξη εφαρμογών για υπολογιστή (*Windows, Linux, MacOS*), για UWP (Universal Windows Applications), για Android και iOS. Διατίθεται από το 2005 και είναι γραμμένη σε C++ (backend-runtime) και σε C# (backend-scripts). Η χρήση της διευκολύνει την ανάπτυξη εφαρμογών τις οποίες πρέπει να υποστηρίζουν διαφορετικές πλατφόρμες, αφού με τον ίδιο κώδικα μπορεί η εφαρμογή να τρέξει σχεδόν σε οποιαδήποτε από τις παραπάνω πλατφόρμες αναφέρθηκαν, με ελάχιστες ή και καμία αλλαγή.

Στην παρακάτω ενότητα αναπτύσσονται κάποια βασικά στοιχεία της Unity, καθώς και κάποια πιο εξειδικευμένα που αφορούν την ανάπτυξη *XR* εφαρμογών και την επέκταση της Unity πέραν της C#.

1.6.1 Βασικά στοιχεία

Η Unity, καθώς χτίστηκε για να βοηθήσει εξαρχής στην ανάπτυξη παιχνιδιών και έπειτα και σε γενικότερες εφαρμογές, παρέχει ένα περιβάλλον ανάπτυξης υψηλού επιπέδου. Δε χρειάζεται ο προγραμματιστής / σχεδιαστής να προγραμματίσει την εφαρμογή του από την αρχή, αλλά δουλεύει πάνω σε ένα πλαίσιο το οποίο ήδη του προσφέρει έτοιμα πολλά πράγματα. Ένα από τα σημαντικότερα είναι πως δε χρειάζεται να γράψει ούτε υλοποίηση για το πως σχεδιάζονται τα γραφικά στην οθόνη, αλλά ούτε να ασχοληθεί ιδιαίτερα με τα μαθηματικά τρισδιάστατου χώρου, περιγραφή των οποίων έγινε σε προηγούμενη ενότητα.

Το βασικό δομικό στοιχείο της Unity είναι το ***GameObject***. Αποτελεί μία οντότητα του περιβάλλοντος της Unity της οποίας τη συμπεριφορά μπορεί να καθορίσει ο προγραμματιστής μέσω των ***C# Scripts*** που γράφει. Ένα τυπικό *C# Script* το οποίο τοποθετείται πάνω σε ένα *gameobject* κληρονομεί από την κλάση ***MonoBehaviour***, η οποία παρέχει κάποιες βασικές λειτουργίες που έχει οποιαδήποτε συμπεριφορά. Όταν αυτό το *script* τοποθετείται πάνω σε ένα *gameobject* τότε λέμε πως το gameobject έχει αποκτήσει ένα ***Component***. Η *Unity* παρέχει διάφορες συναρτήσεις ώστε τα *components* πάνω σε κάθε *gameobject* να έχουν τη δυνατότητα επικοινωνίας μεταξύ τους. Με αυτό τον τρόπο προωθείται το **αρθρωτό μοντέλο προγραμματισμού** (***component-oriented)***, το οποίο κατά μία έννοια είναι μία επέκταση του αντικειμενοστραφούς. Σε αυτό το μοντέλο προσπαθεί ο προγραμματιστής να περιγράψει το σύστημα το οποίο θέλει να φτιάξει με βάση τα βασικά δομικά στοιχεία του, τα οποία έπειτα συνεργάζονται μεταξύ τους για να εκτελέσουν μια

λειτουργία. Έτσι αποφεύγεται η ύπαρξη μονολιθικών κλάσεων οι οποίες εκτελούν πολλαπλές λειτουργίες και η κάθε λειτουργία περιγράφεται μεμονωμένα.

Μία επιπλέον καλή τακτική στη Unity είναι οι κλάσεις να κληρονομούν το πολύ από μία κλάση βάση και όχι παραπάνω.

Να σημειωθεί επιπλέον πως ένα οποιοδήποτε *gameobject* έρχεται πάντα με ένα προκαθορισμένο *component*, το ***Transform***, το οποίο περιγράφει τη θέση, την περιστροφή και το μέγεθος του *gameobject* στον χώρο.

Με όσα αναφέρθηκαν παραπάνω κάποιος μπορεί να σχεδιάσει *scripts* και εφαρμογές στη *Unity* τα οποία είναι ευκολότερα τροποποιήσιμα και ευανάγνωστα.

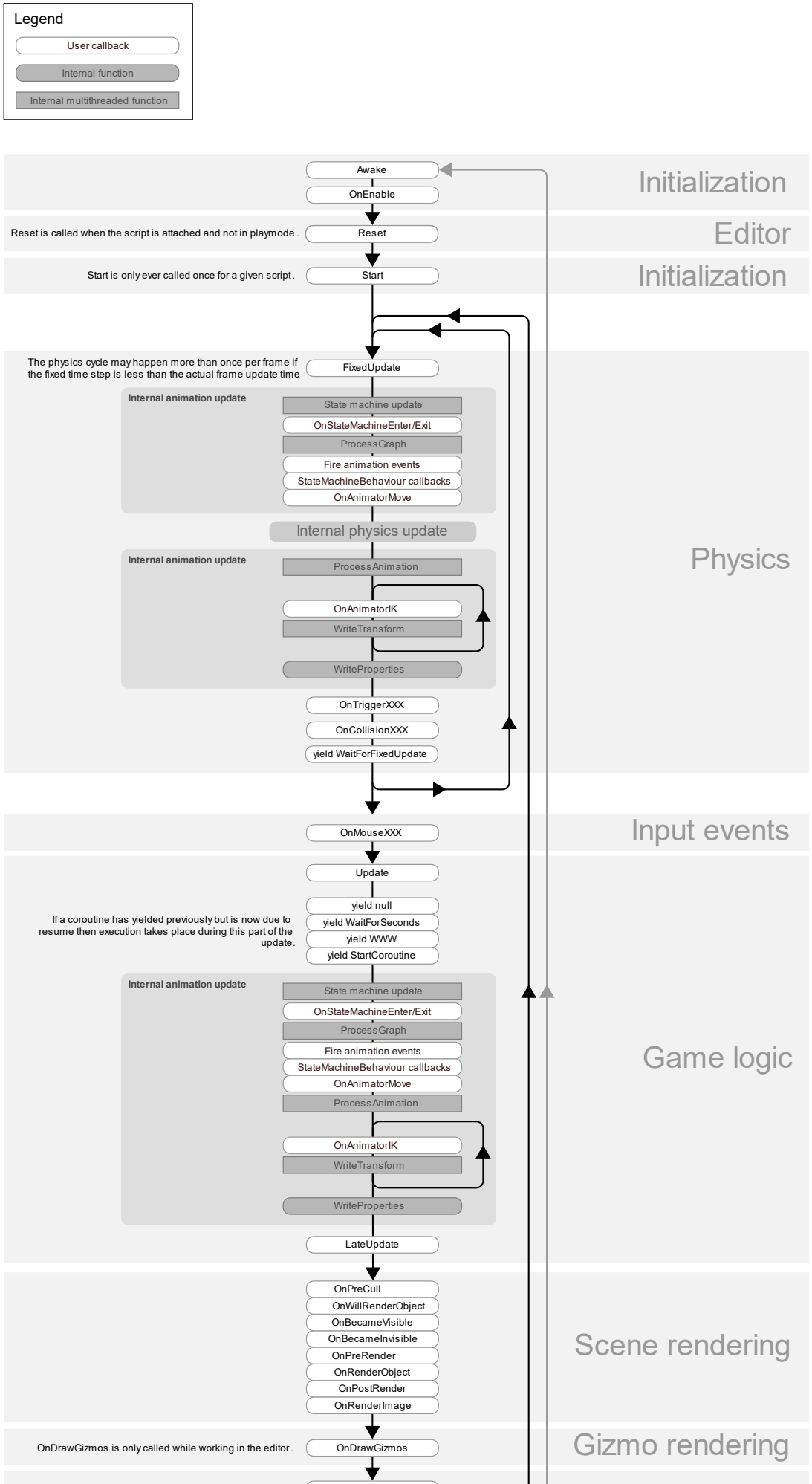
Τέλος, είναι σημαντικό να επισημάνουμε πως η πλατφόρμα πάνω στην οποία λειτουργεί το *Hololens 2* είναι η ***UWP (Universal Windows Platform)***, σχεδιασμένη από τη *Microsoft* ώστε να γράφονται εφαρμογές με τον ίδιο κώδικα για διαφορετικές συσκευές που υποστηρίζουν αυτήν την πλατφόρμα.

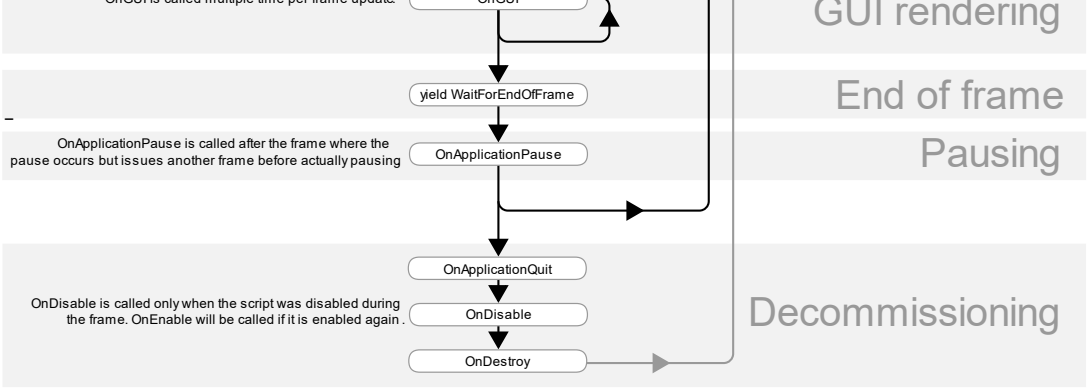
1.6.2 Κύκλος ζωής μιας εφαρμογής Unity [48]

Ένα οποιοδήποτε *component* έχει έναν κύκλο ζωής. Πέραν τον κύκλο ζωής του *component*, μια εφαρμογή στη *Unity* έχει διάφορα συστήματα τα οποία εκτελούν το δικό τους κύκλο ζωής. Παρακάτω αναφέρονται τα σημαντικότερα στάδια του κύκλου ζωής ενός *component* καθώς και παρουσιάζεται ένα αναλυτικό διάγραμμα ροής το οποίο περιγράφει πλήρως τα κύρια συστήματα της Unity.

Σημαντικά γεγονότα / μέθοδοι στα *components*

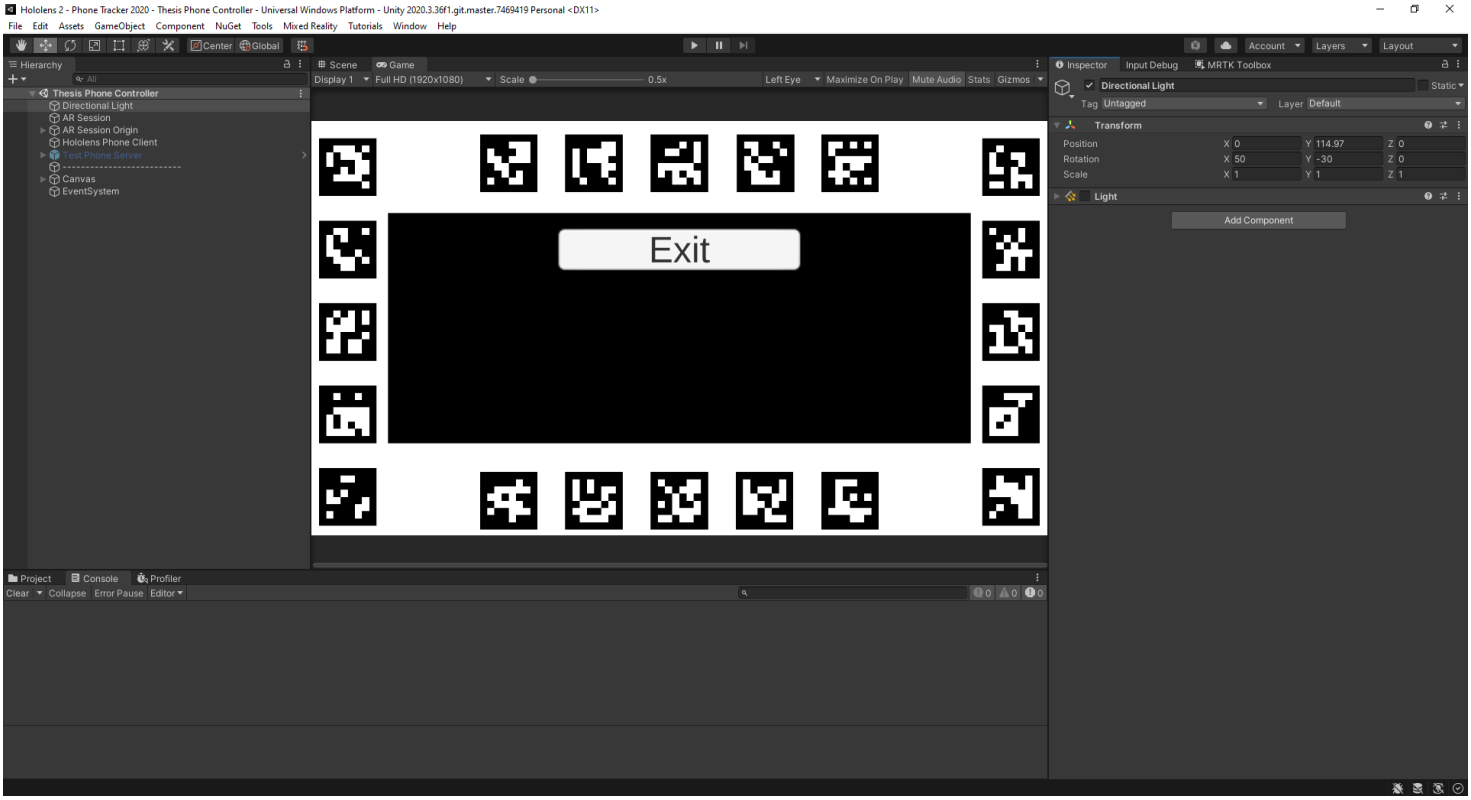
1) ***Awake:*** Η μέθοδος *Awake* καλείται όταν δημιουργείται για πρώτη φορά ένα *gameobject* το οποίο έχει πάνω ένα *component* με αυτή τη συνάρτηση. Δεν ξανακαλείται για το υπόλοιπο της ζωής του *gameobject*.
2) ***OnEnable:*** Η μέθοδος καλείται όταν δημιουργείται για πρώτη φορά ένα *gameobject* και είναι και ενεργό. Έπειτα καλείται όταν το *component* για οποιοδήποτε λόγο μεταβεί από την ανενεργό στην ενεργό κατάσταση.
3) ***OnDisable:*** Η μέθοδος καλείται όταν ένα *component* μεταβαίνει για οποιοδήποτε λόγο από την ενεργό στην ανενεργό κατάσταση
4) ***Start:*** Η μέθοδος καλείται όταν δημιουργηθεί ένα *gameobject* με τουλάχιστον ένα *component* που την υλοποιεί. Δεν ξανακαλείται κατά τη διάρκεια ζωής ενός *gameobject* και πάντα καλείται μετά την ***Awake***, ώστε τα διάφορα *components* να μπορούν να συνεργαστούν καλύτερα.
5) ***Update:*** Αφορά τη λογική του παιχνιδιού και τρέχει σε κάθε *frame*, με το ρολόι της εφαρμογής. Η μεταβολή του χρόνου δηλαδή μεταξύ κάθε *Update* δεν είναι σταθερή.
6) ***Fixed Update:*** Αφορά την προσομοίωση της φυσικής και τρέχει με σταθερό ρολόι, ώστε η προσομοίωση να παρέχει όσο το δυνατόν πιο έγκυρα αποτελέσματα γίνεται


Legend
User callback
Internal function
Internal multithreaded function
Initialization
Awake
OnEnable
Editor
Reset is called when the script is attached and not in playmode.
Reset
Initialization
Start is only ever called once for a given script.
Start
Physics
The physics cycle may happen more than once per frame if the fixed time step is less than the actual frame update time
FixedUpdate
Internal animation update
State machine update
OnStateMachineEnter/Exit
ProcessGraph
Fire animation events
StateMachineBehaviour callbacks
OnAnimatorMove
Internal physics update
Internal animation update
ProcessAnimation
OnAnimatorIK
WriteTransform
WriteProperties
OnTriggerXXX
OnCollisionXXX
yield WaitForFixedUpdate
Input events
OnMouseXXX
Game logic
Update
If a coroutine has yielded previously but is now due to resume then execution takes place during this part of the update.
yield null
yield WaitForSeconds
yield WWW
yield StartCoroutine
Internal animation update
State machine update
OnStateMachineEnter/Exit
ProcessGraph
Fire animation events
StateMachineBehaviour callbacks
OnAnimatorMove
ProcessAnimation
OnAnimatorIK
WriteTransform
WriteProperties
LateUpdate
Scene rendering
OnPreCull
OnWillRenderObject
OnBecameVisible
OnBecameInvisible
OnPreRender
OnRenderObject
OnPostRender
OnRenderImage
Gizmo rendering
OnDrawGizmos is only called while working in the editor.
OnDrawGizmos



ΕΙΚΟΝΑ 24 ΚΥΚΛΟΣ ΖΩΗΣ ΕΝΟΣ MONOBEHAVIOUR

1.6.3 Βασικό Περιβάλλον Ανάπτυξης (Editor)



ΕΙΚΟΝΑ 25 EDITOR UNITY

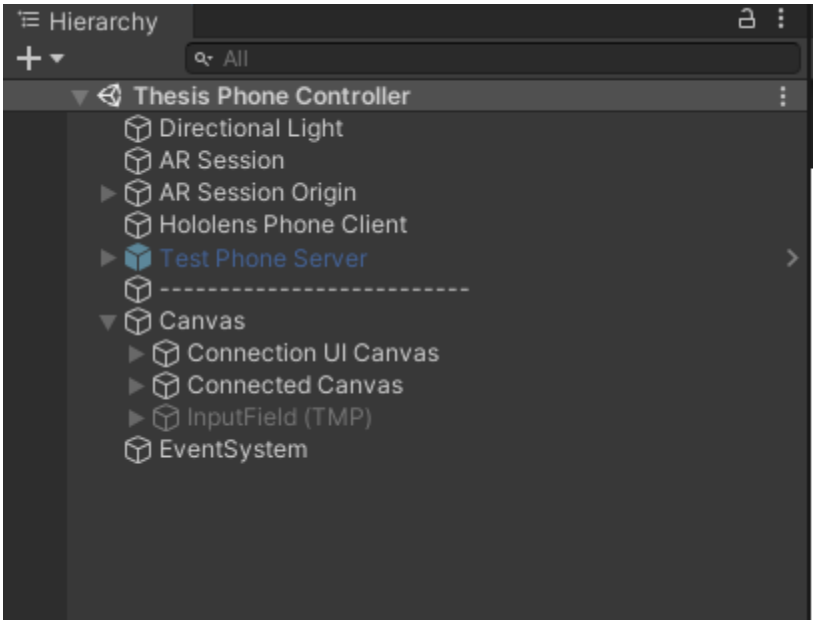
Στην παραπάνω εικόνα παρουσιάζεται το περιβάλλον ανάπτυξης (***Editor***) της Unity. Τα διάφορα παράθυρα μπορούν να επανατοποθετηθούν από τον χρήστη του *Editor* για την πιο εύκολη διαχείρισή του. Συνοπτικά τα πιο σημαντικά παράθυρα είναι:

- ***Project:*** Κάθε εφαρμογή στη *Unity* δημιουργεί ένα φάκελο που λέγεται ***Assets***. Μέσα σε αυτόν τον φάκελο βρίσκονται όλα τα αρχεία και τα δεδομένα, καθώς και τα *scripts* τα οποία χρησιμοποιεί ο προγραμματιστής / σχεδιαστής της εφαρμογής. Υπάρχει μία μπάρα αναζήτησης για διευκόλυνση εντοπισμού ενός επιθυμητού αρχείου και διάφορες άλλες δυνατότητες που στοχεύουν στη διευκόλυνση του χρήστη.

- ***Console (Κονσόλα):*** Αυτό το παράθυρο το χρησιμοποιεί ο σχεδιαστής ώστε να εκτυπώσει οτιδήποτε δεδομένα θέλει κατά τη διάρκεια της εφαρμογής. Είναι ουσιαστικά ένας τρόπος εντοπισμού σφαλμάτων (***debugging***), καθώς παρέχεται η δυνατότητα εποπτείας της εσωτερικής κατάστασης των διάφορων *components.*

- ***Profiler:*** Ο profiler χρησιμεύει στην επισκόπηση της λειτουργίας της εφαρμογής. Δείχνει πόση ώρα χρειάζονται διάφορα συστήματα της *Unity* για να εκπληρώσουν κάποιον κύκλο λειτουργίας καθώς και η λογική που έχει περιγράψει ο σχεδιαστής σε *components*.

- ***Hierarchy (Ιεραρχία):*** Η ιεραρχία είναι άκρως σημαντική, διότι δείχνει τα διάφορα *gameobjects* που έχουν τοποθετηθεί στην εφαρμογή. Παρατηρείται πως υπάρχει μια σχέση γονέα – παιδιού μεταξύ των *gameobjects*, σε μορφή δένδρου. Αυτή η σχέση υπάρχει και για την καλύτερη οργάνωση του σχεδιασμού αλλά και γιατί προσφέρει άλλα πλεονεκτήματα, όπως την αλλαγή του συστήματος αναφοράς του τρισδιάστατου χώρου σε ένα παιδί στο σύστημα αναφοράς του γονέα. Επιπλέον, δίνει τη δυνατότητα πρόσθεσης, αφαίρεσης και επιλογής *gameobjects*.



ΕΙΚΟΝΑ 26 Η ΙΕΡΑΡΧΙΑ ΤΩΝ GAMEOBJECTS

- ***Inspector:*** Ο *inspector* αποτελεί τον τρόπο εποπτείας των διάφορων *components* που βρίσκονται πάνω σε ένα *gameobject*. Εδώ μπορεί ο χρήστης της *Unity* να προσθαφαιρέσει components καθώς και να κάνει αλλαγή στην αρχικοποίησή τους.
- ***Scene View – Game View:*** Το *Scene View* δίνει τη δυνατότητα περιήγησης στη σκηνή. Παρέχονται οι δυνατότητες τροποποίησης βασικών στοιχείων ενός *gameobject* - θέσης, περιστροφής και μεγέθους – μέσω ενός γραφικού εργαλείου. Η κάμερα σε αυτό το view είναι ελεύθερη.0

Με την εναλλαγή στο Game View ο σχεδιαστής βλέπει ακριβώς το πως θα είναι το παιχνίδι κατά τη στιγμή της εκκίνησης, σύμφωνα πάντα με τον τρόπο που έχει ρυθμίσει την κάμερα.

Να σημειωθεί πως η γίνεται παρέχει πολλές από τις λειτουργίες της σε μορφή πακέτων, τα οποία και κατεβάζεις από τον ***Package Manager***.

1.6.4 *Plug-ins* [49]

Η *Unity* προσφέρει τη δυνατότητα στον σχεδιαστή της χρήσης εξωτερικών βιβλιοθηκών, γραμμένες ακόμα και σε διαφορετική γλώσσα προγραμματισμού. Αυτό είναι ένα πολύ χρήσιμο εργαλείο όταν δεν αρκεί η C# για την υλοποίηση ενός αλγορίθμου ή όταν δεν υπάρχουν οι κατάλληλες βιβλιοθήκες για την ανάπτυξη του. Τα *plug-ins* χωρίζονται σε 2 κατηγορίες:

1) ***Managed plug-ins***: Είναι *compiled* σε *.NET assemblies*. Περιέχουν ουσιαστικά μόνο *.NET* κώδικα, που σημαίνει πως δε μπορούν να χρησιμοποιήσουν περαιτέρω χαρακτηριστικά πέραν αυτών που παρέχει το *.NET*. Η διαφορά τους με τα απλά *scripts* της *Unity* είναι ό,τι γίνονται *compile* εξωτερικά από τη *Unity*, σε αντίθεση με τα scripts που αποθηκεύονται μέσα στη *Unity* ως πηγαίος κώδικας.
2) ***Native plug-ins***: Αποτελούν κώδικα γραμμένο σε μια άλλη συνήθως γλώσσα προγραμματισμού, όπως παραδείγματος χάρη C++. Γίνονται *compile* εξωτερικά της *Unity* και δίνουν, μεταξύ άλλων, τη δυνατότητα κλήσης στο λειτουργικό σύστημα καθώς και συγκεκριμένα για μια πλατφόρμα χαρακτηριστικά τα οποία αλλιώς δεν είναι διαθέσιμα στη *Unity*.

Τα *plug-ins* τοποθετούνται σε συγκεκριμένους υπο-φακέλους στη *Unity* με συγκεκριμένα ονόματα για την κάθε πλατφόρμα. Έρχονται σε διάφορες μορφές, όπως ***.dll, .so*** κ.ο.κ

1.6.5 Σύστημα εισόδου *Unity (Input System)* [50], [51]

Το σύστημα εισόδου αποτελεί ένα πακέτο το οποίο παρέχει η *Unity* μέσω του *Package Manager*. Σκοπός του είναι η χρήση οποιασδήποτε **Συσκευής Εισόδου *(Input Device)*** για τον έλεγχο περιεχομένου μέσα στην εφαρμογή. Είναι σχεδιασμένο ώστε να παρέχει μεγάλη ευελιξία και ταυτόχρονα να μειώνει τη συγγραφή κώδικα. Ο σχεδιασμός μίας εφαρμογής που χρησιμοποιεί κάποια είσοδο μπορεί επί το πλείστον να γίνει μέσω της γραφικής διεπαφής που προσφέρει το πακέτο.

Η φιλοσοφία του βασίζεται στο διαχωρισμό της συσκευής με την είσοδο που παρέχει καθώς και το διαχωρισμό της εισόδου με την ενέργεια που ενεργοποιεί. Έτσι, προκύπτουν οι εξής κατηγορίες:

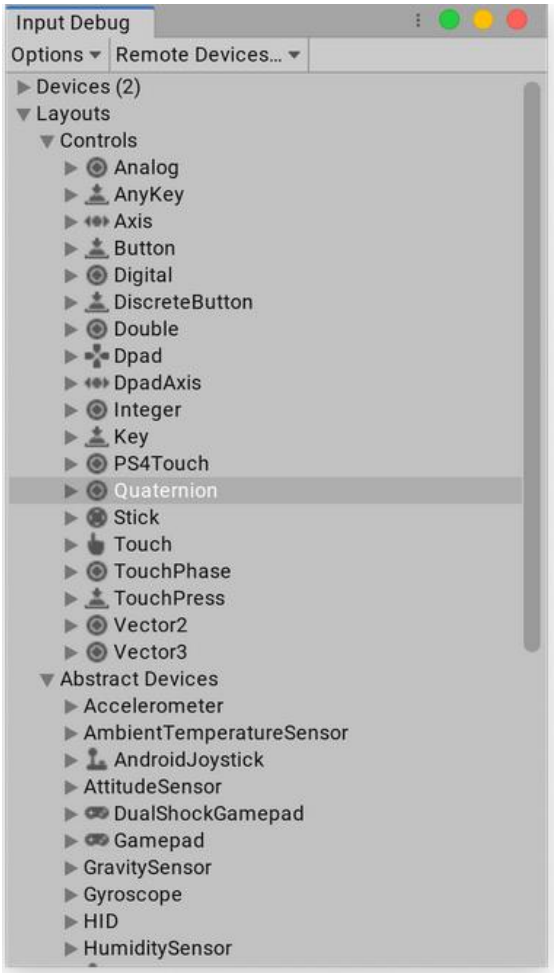
1) **Φυσική Συσκευή / Είσοδος**.
   Αποτελεί τη φυσική συσκευή που επικοινωνεί με την εφαρμογή, παραδείγματος χάρι ένα πληκτρολόγιο, ένα ποντίκι, μια οθόνη αφής. Στην κατηγορία αυτή εντάσσονται και οι διάφοροι αισθητήρες που μπορεί να έχει ένα κινητό. Γενικά οτιδήποτε μπορεί να στείλει σήμα εισόδου στην εφαρμογή.
2) **Ψηφιακή Συσκευή**.
   Η ψηφιακή συσκευή αποτελεί την αναπαράσταση της φυσικής συσκευής στον κόσμο της *Unity*. Μπορεί να είναι γραμμένη ως C# κλάση ή ως ένα ***layout***, από το οποίο μπορεί να παραχθεί αυτόματα η κλάση. Το *layout* περιγράφει ένα συγκεκριμένο τρόπο αποθήκευσης των δεδομένων εισόδου που προέρχονται από τη συσκευή, για βέλτιστη απόδοση. Επιπλέον, περιγράφει τις διαφορετικές εισόδους (**controls**) που μπορεί να έχει μία συσκευή · Για παράδειγμα, ένα χειριστήριο βιντεοπαιχνιδιών έχει δύο μοχλούς και διάφορα κουμπιά, καθένα από τα οποία παρέχουν μία είσοδο στην εφαρμογή.

   Η ύπαρξη της ψηφιακής συσκευής καθιστά δυνατή και εύκολα την προσομοίωση της φυσικής συσκευής σε μία εφαρμογή. Μπορεί να μην υπάρχει ένα φυσικό πληκτρολόγιο, αλλά δίνεται η δυνατότητα να δημιουργηθεί ένα εικονικό πληκτρολόγιο το οποίο μέσω ενός *API* χαμηλού επιπέδου θα δίνει εικονικά πατήματα κουμπιών στην εφαρμογή.

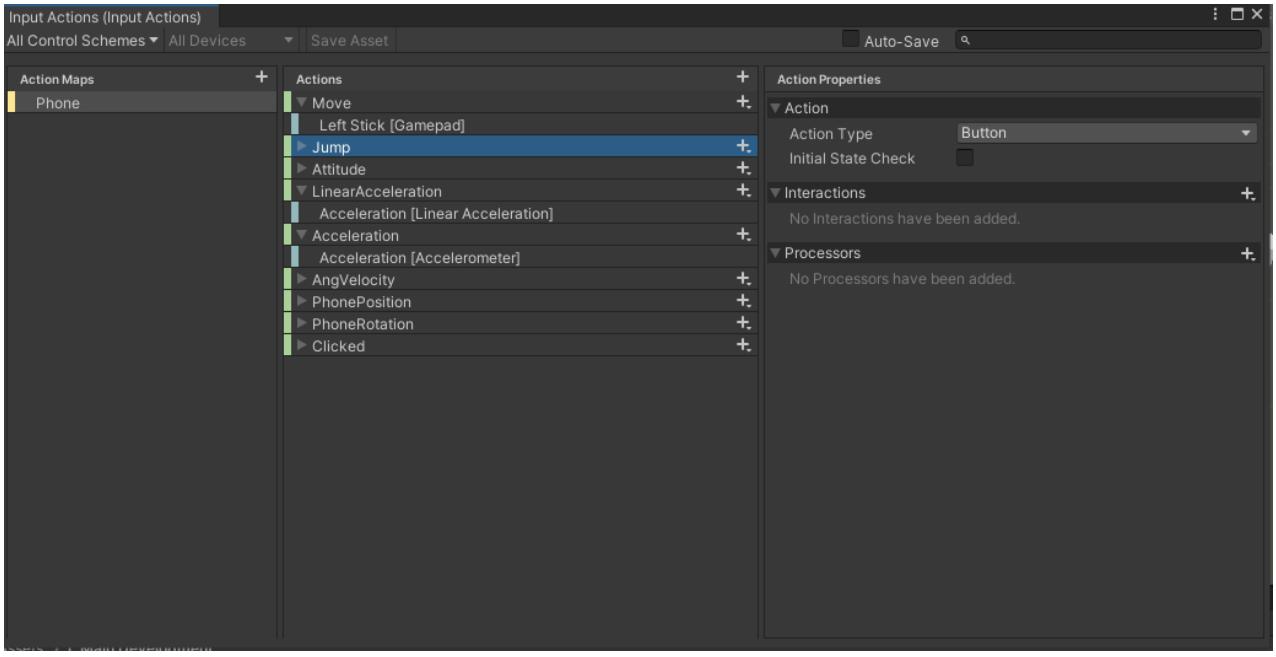
   Αναφέρεται επίσης πως ο σχεδιαστής μπορεί να ορίσει τις δικές του ψηφιακές συσκευές, τις οποίες μπορεί είτε να χρησιμοποιήσει για κάποια προσομοίωση είτε να τις συνδέσει με μια φυσική συσκευή.

   Τέλος, να σημειωθεί πως η αντιστοιχία φυσικής και ψηφιακής συσκευής δεν είναι πάντα ένα προς ένα. Ένα επιταχυνσιόμετρο, για παράδειγμα, υπάρχει και ως φυσική και ως ψηφιακή συσκευή αλλά η γραμμική επιτάχυνση δίνεται στο σύστημα ως μια επιπλέον ψηφιακή συσκευή η οποία δεν υπάρχει στον φυσικό κόσμο, αλλά προκύπτει από συνδυασμό και φιλτράρισμα εισόδου άλλων συσκευών.

3) Ενέργεια Εισόδου *(Input Action)*
   Παρότι υπάρχει η δυνατότητα απευθείας ανάγνωσης μίας εισόδου κάποιας συσκευής, οι σχεδιαστές του Συστήματος Εισόδου παρείχαν μια αρκετά πιο έξυπνη εναλλακτική · Χωρίς βέβαια να καθίσταται ασήμαντη η απευθείας ανάγνωση. Ο σχεδιαστής της εφαρμογής μπορεί να καθορίσει διάφορες ενέργειες τις οποίες και δεσμεύει σε διάφορες εισόδους. Έτσι δημιουργείται ένας χάρτης ενεργειών. Ο σχεδιαστής μπορεί να δημιουργήσει πολλούς χάρτες καθώς και να δεσμεύει την ίδια σε παραπάνω από μία είσοδο, αρκεί βέβαια η ενέργεια και η είσοδος να έχουν τον ίδιο τύπο δεδομένων.



ΕΙΚΟΝΑ 27 INPUT DEBUGGER



ΕΙΚΟΝΑ 28 EDITOR ΧΑΡΤΗ ΕΝΕΡΓΕΙΩΝ

1.6.5 *AR Foundation* [52]

Ένα από τα πακέτα που παρέχει η *Unity* είναι το *AR Foundation*. Είναι ένα πακέτο ανάπτυξης εφαρμογών για περιβάλλοντα επαυξημένης πραγματικότητας το οποίο στοχεύει στην ενοποίηση των διαφορετικών υλοποιήσεων λογισμικού *AR* γύρω από ένα ενιαίο *API*. Η δυνατότητα αυτή είναι εξαιρετικά σημαντική, διότι μπορεί κάποιος πολύ εύκολα να αλλάξει την υλοποίηση διάφορων αλγορίθμων (όπως του *VIO* που αναφέρθηκε σε προηγούμενη ενότητα) με ελάχιστο κόπο. Τα πακέτα λογισμικού τα οποία ενοποιεί το συγκεκριμένο *API* είναι:

1) *Google ARCore για Android*
2) *Apple ARKit για iOS*
3) *Magic Leap για το Magic Leap*
4) *WindowsXR ή OpenXR για προϊόντα της Microsoft.*

	ARCore	ARKit	OpenXR
Device tracking	✓	✓	✓
Plane tracking	✓	✓	
Point clouds	✓	✓	
Anchors	✓	✓	✓
Light estimation	✓	✓	
Environment probes	✓	✓	
Face tracking	✓	✓	
2D Image tracking	✓	✓	
3D Object tracking		✓	
Meshing		✓	✓
2D & 3D body tracking		✓	
Collaborative participants		✓	
Human segmentation		✓	
Raycast	✓	✓	
Pass-through video	✓	✓	
Session management	✓	✓	✓
Occlusion	✓	✓	

ΕΙΚΟΝΑ 29 ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΤΩΝ ΔΙΑΦΟΡΩΝ PLUG-IN

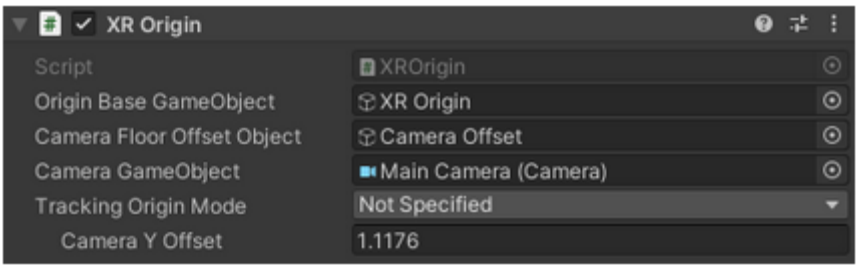
AR Session

Αποτελεί το *component* το οποίο ελέγχει τον κύκλο ζωής μιας εφαρμογής επαυξημένης πραγματικότητας μέσω του *AR Foundation*

ΕΙΚΟΝΑ 30 AR SESSION COMPONENT

XR Origin

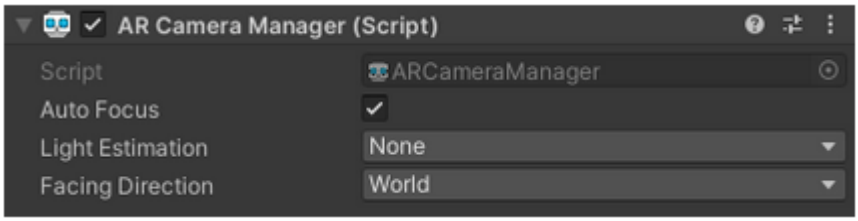
Το *component* αυτό παρέχει στα ανιχνεύσιμα χαρακτηριστικά του χώρου, όπως για παράδειγμα επίπεδες επιφάνειες και σημεία ενδιαφέροντος, την τελική θέση, περιστροφή και μέγεθος στη σκηνή ώστε να συνδυάζονται με τον πραγματικό κόσμο.



ΕΙΚΟΝΑ 31 XR ORIGIN COMPONENT

AR Camera Manager

Το *component* διαχειρίζεται κάποια χαρακτηριστικά της κάμερας της συσκευής επαυξημένης πραγματικότητας. Επιπλέον υπάρχει ένα component για απενεργοποίηση της εικόνας της κάμερας στη συσκευή (*AR Camera Background*). Η δυνατότητα αυτή χρησιμεύει στα έξυπνα κινητά και όχι σε *HMDs* όπως το *Hololens*.



ΕΙΚΟΝΑ 32 AR CAMERA COMPONENT

AR Input Manager

Το component αυτό ενεργοποιεί την ανίχνευση του κόσμου. Δεν έχει κάποια ιδιαίτερα χαρακτηριστικά.

1.6.6 MRTK (Mixed Reality Toolkit) [53]

Το *MRTK-Unity* είναι μία εργαλειοθήκη η οποία παρέχεται ως *package* για τη *Unity* για τη διευκόλυνση της ανάπτυξης και του σχεδιασμού εφαρμογών επαυξημένης πραγματικότητας. Προσφέρει στο σχεδιαστή μια συλλογή από *gameobjects* και *components*. Να σημειωθεί πως το *plug in* αυτό είναι παρόμοιο με το *AR Foundation,* αλλά είναι αποκλειστικά από τη *Microsoft* και ως εκ τούτου είναι καλύτερο να χρησιμοποιηθεί αυτό. Παρουσία και των 2 στην εφαρμογή δε φαίνεται να παρουσιάζει κάποιο πρόβλημα, παρόλα αυτά αρκεί η χρήση του ενός ώστε να μπορείς να χτίσεις μία *AR* εφαρμογή. Αναφερόμαστε συγκεκριμένα στο *MRTK 2.8* διότι αυτό χρησιμοποιήθηκε στα πλαίσια της διπλωματικής. Παρόλα αυτά γνωστοποιείται πως υπάρχει μία καινούρια έκδοση, το *MRTK 3.*

Οι πλατφόρμες και οι συσκευές που υποστηρίζονται είναι οι εξής:

Platform	Supported Devices
OpenXR (Unity 2020.3.8+)	Microsoft HoloLens 2 Windows Mixed Reality headsets
Windows Mixed Reality	Microsoft HoloLens Microsoft HoloLens 2 Windows Mixed Reality headsets
Oculus (Unity 2019.3 or newer)	Oculus Quest
OpenVR	Windows Mixed Reality headsets HTC Vive Oculus Rift
Ultraleap Hand Tracking	Ultraleap Leap Motion controller
Mobile	iOS and Android

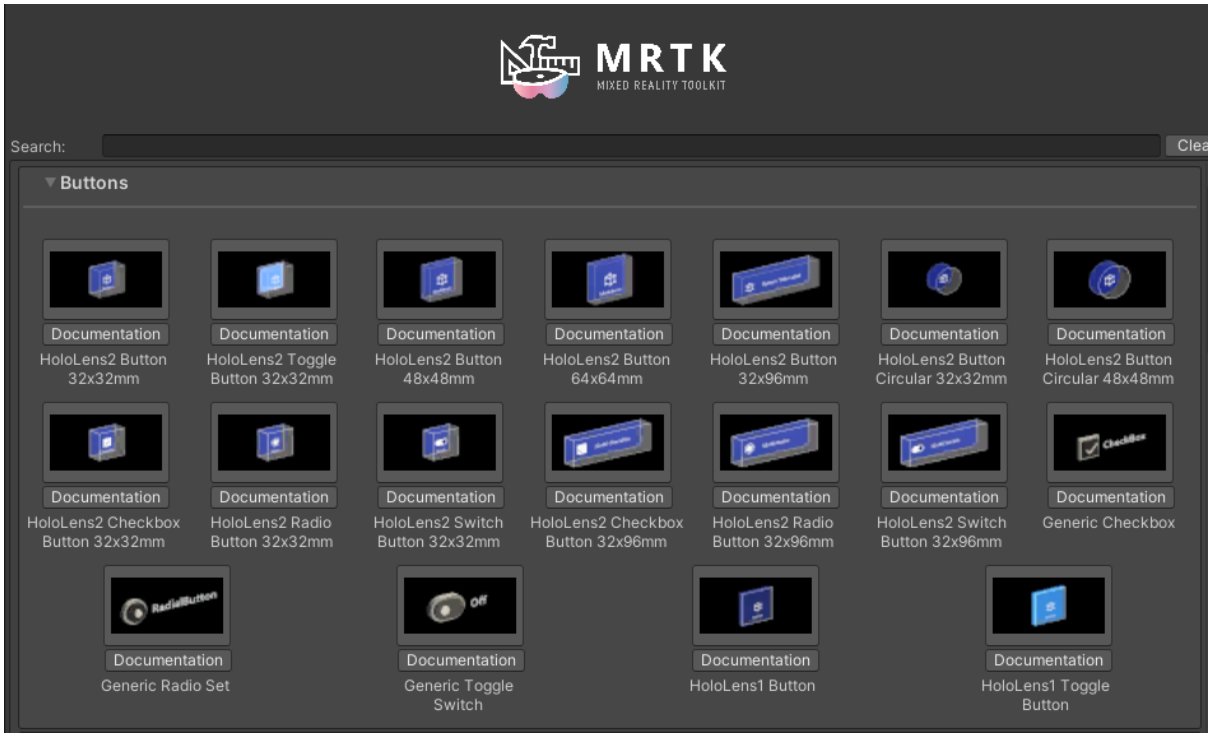
ΕΙΚΟΝΑ 33 ΠΛΑΤΦΟΡΜΕΣ ΚΑΙ ΣΥΣΚΕΥΕΣ MRTK

Η εγκατάστασή του γίνεται εύκολα από το *Mixed Reality Feature Tool*, ένα εργαλείο που παρέχει η *Microsoft* και όχι από τον *Package Manager* της *Unity*. Παρόλα αυτά θεωρείται ως ένα πακέτο το οποίο μετά εμφανίζεται στον *manager*.

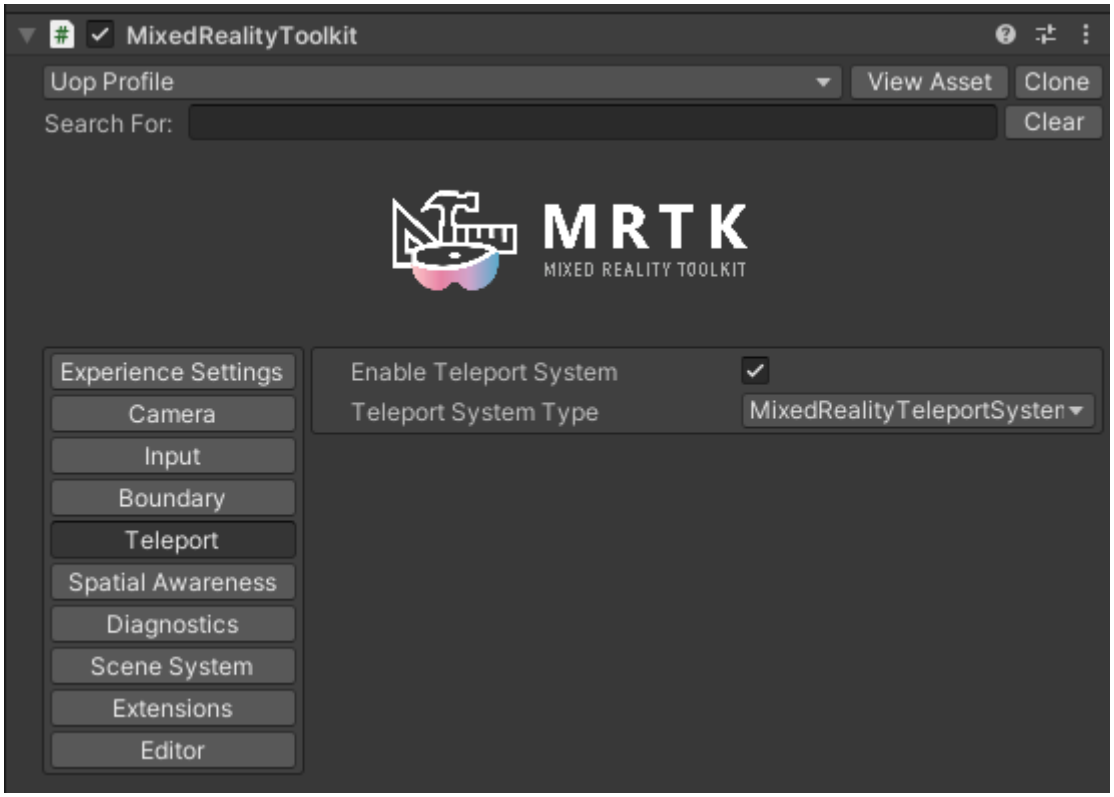
Μετά την εγκατάστασή του εμφανίζεται ένα *tab* στον *Unity Editor* το οποίο παρέχει αρκετά εργαλεία, καθώς και παρέχονται αρκετά components. Μερικά από αυτά είναι:

1) Διευκόλυνση του τελικού build της εφαρμογής από τη Unity στο Hololens.
2) Ρύθμιση του *project* με τις καταλληλότερες ρυθμίσεις για την εκάστοτε εφαρμογή.
3) Εύκολη εναλλαγή διάφορων προφίλ, ανάλογα με τον τύπο εφαρμογής και τη συσκευή.
4) Διάφορα έτοιμα *UI Components* για τη δημιουργία διεπαφής χρήστη.

5) Παροχή των *component Solver* τα οποία παρέχουν τη δυνατότητα τοποθέτησης διάφορων στοιχείων με βάση τη θέση, το κεφάλι, το χέρι του χρήστη κ.ο.κ



ΕΙΚΟΝΑ 34 MRTK TOOLBOX - COMPONENTS



ΕΙΚΟΝΑ 35 ΡΥΘΜΙΣΕΙΣ ΤΟΥ MRTK

1.6.7 *Microsoft Visual Studio* [54]

Το *Microsoft Visual Studio* είναι τυπικά ο *IDE* / code *editor* που χρησιμοποιείται για ανάπτυξη εφαρμογών στη *Unity*. Εκτός της *Unity*, μπορεί να χρησιμοποιηθεί και για την ανάπτυξη *plug-in*, αφού περιέχει διάφορους *compilers*.

Συγκεκριμένα για το *HoloLens*, η *Unity* παράγει εξωτερικά ένα *Visual Studio Project* το οποίο μετά ο σχεδιαστής μπορεί να κάνει *compile* και να το φορτώσει στη συσκευή, είτε με ασύρματη σύνδεση (*Wi-fi*) είτε ενσύρματα μέσω μίας θύρας *USB*. Η διαδικασία αυτή όμως δεν είναι σύντομη, γι' αυτό ο σχεδιαστής κρίνεται αναγκαίο να χρησιμοποιεί το ***Holographic Remoting***, ένα *API* το οποίο καθιστά ικανή την επικοινωνία *Unity Editor* με το HoloLens, ούτως ώστε η εφαρμογή τρέχει στον υπολογιστή και επικοινωνεί με τη συσκευή, ανταλλάσσοντας πληροφορίες σχετικά με το χώρο και την αναγνώριση χειρονομιών. Ο σχεδιαστής εν τέλη βλέπει μια προσομοίωση της εφαρμογής στο *Hololens* χωρίς να χρειάζεται να τη φορτώσει σε αυτό.

Κεφάλαιο 2 – Σχεδίαση πλαισίου επικοινωνίας μεταξύ ελεγχόμενης συσκευής και ελεγκτή

Στο παρόν κεφάλαιο αναπτύσσεται η σχεδίαση της επικοινωνίας μεταξύ ενός εξωτερικού ελεγκτή και μίας συσκευής την οποία επιθυμούμε να ελέγξουμε. Στη συγκεκριμένη περίπτωση, η ελεγχόμενη συσκευή θα είναι το HoloLens 2 και ο ελεγκτής ένα έξυπνο κινητό. Παρουσιάζεται στην αρχή ο σκοπός και η πιθανή χρησιμότητα μιας τέτοιας επικοινωνίας. Έπειτα δίνεται ο σχεδιασμός ενός συστήματος ***serialization*** και ενός συστήματος επικοινωνίας μεταξύ των δύο συσκευών. Στη συνέχεια γίνεται αναφορά στο *Input System* της *Unity*, το οποίο και χρησιμοποιείται για την εύκολη κωδικοποίηση σημάτων του ελεγκτή. Τέλος, συνδυάζονται όλα αυτά για τη δημιουργία ενός πελάτη (*client*) και ενός διακομιστή (server), που θα τρέχουν στον ελεγκτή και στην ελεγχόμενη συσκευή αντίστοιχα.

Η σχεδίαση έγινε στη *Unity 2020.3 (*προτείνεται για το HoloLens το 2022), με το *MRTK 2.8*.

2.1 Σκοπός

Όπως είδαμε στο Κεφάλαιο 1, το *HoloLens 2* προσφέρει πολλούς και διαφορετικούς τρόπους αλληλεπίδρασης. Ενώ είναι όλοι εξαιρετικά χρήσιμοι και σχεδιασμένοι ώστε ο χρήστης να νιώθει πως βρίσκεται σε ένα εκτεταμένο περιβάλλον, πέραν του πραγματικού, έρχονται με κάποια μειονεκτήματα.

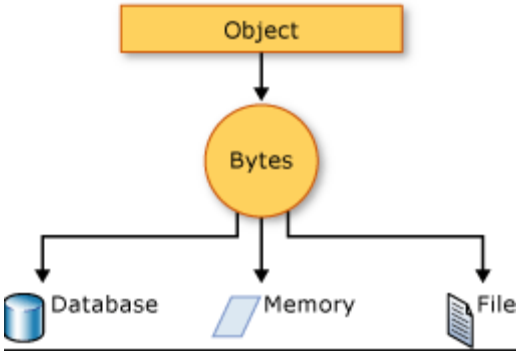
1) Οι τρόποι αλληλεπίδρασης που αφορούν την οποιαδήποτε ανίχνευση των χεριών επιβάλλουν τα χέρια να βρίσκονται μέσα στο πλαίσιο ανίχνευσης της κάμερας. Παρότι μεγάλο, αυτό σημαίνει πως ο χρήστης πρέπει να μετακινεί συνέχεια τα χέρια του και όλο του το σώμα, μία διαδικασία που ίσως αποβεί κουραστική.
2) Οι τρόπου αλληλεπίδρασης που αφορούν αναγνώριση φωνητικών εντολών ή επιλογή μέσω δείκτη κεφαλής – βλέμματος απαιτούν περισσότερο χρόνο, καθιστώντας την εμπειρία πιο αργή.
3) Επέκταση της αλληλεπίδρασης με την εισαγωγή λειτουργιών που το HoloLens δε μπορεί να υποστηρίξει. Το κινητό παρέχει μία οθόνη αφής, αισθητήρες, δόνηση και πληθώρα άλλων λειτουργιών που μπορούν αν συνδυαστούν για τη δημιουργία ενός καινούριου τύπου αλληλεπίδρασης.

Σημειώνεται πως το πλαίσιο επικοινωνίας είναι γραμμένο σε *C#* εντός *Unity* και για τον διακομιστή αλλά και για τον πελάτη. Επιπλέον, η σχεδίαση χρησιμοποιεί μία τροποποιημένη εκδοχή ενός πλαισίου δικτύωσης ελεύθερου κώδικα (*open source*), του **MASTERSERVER FRAMEWORK** [55].

2.2 Κωδικοποίηση (*Serialization*) [56]

Για να είναι ικανή η οποιαδήποτε επικοινωνία μεταξύ δύο διαφορετικών συστημάτων, πρέπει να υπάρχει ένα ενιαίο πλαίσιο επικοινωνίας. Αυτό σημαίνει πως πρέπει από το εσωτερικό

περιβάλλον μιας εφαρμογής πελάτη να κωδικοποιηθεί ένα αντικείμενο σε μια άλλη μορφή (**ροή από bytes**), η οποία θα σταλεί μέσω ενός δικτύου στην άλλη συσκευή, θα ληφθεί από την εφαρμογή διακομιστή και έπειτα θα αποκωδικοποιηθεί (*deserialize*) σε κάτι χρήσιμο για την εφαρμογή αυτή.



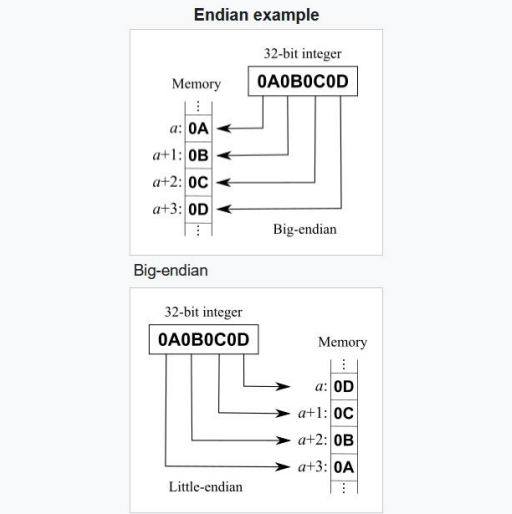
ΕΙΚΟΝΑ 36 ΔΙΑΓΡΑΜΜΑ ΡΟΗΣ ΚΩΔΙΚΟΠΟΙΗΣΗΣ

2.2.1 Η κλάση *EndianBitConverter*

Ορίζουμε την κλάση βάση `EndianBitConverter`, της οποίας ο σκοπός είναι να παρέχει μεθόδους οι οποίες μετατρέπουν βασικούς τύπους δεδομένων, π.χ. έναν ακέραιο, σε έναν πίνακα από bytes. Επιπλέον, παρέχουν μεθόδους και για τον αντίστροφο μετασχηματισμό, τη μετατροπή δηλαδή ενός πίνακα από bytes σε ένα βασικό τύπο δεδομένων.

Η κλάση αυτή είναι αφαιρετική κλάση διότι έγινε η επιλογή παροχής της μετατροπής σε bytes κατά δύο τρόπους, με *little-endian* και με *big-endian*. Η εφαρμογή χρησιμοποιεί τον *big-endian* τρόπο. Αυτό σημαίνει πως η αποθήκευση ξεκινά από το ***least significant bit (LSB)*** και τελειώνει στο ***most significant bit (MSB)***. Οι υλοποιήσεις τις κλάσης αυτής είναι αντίστοιχα ο `LittleEndianBitConverter` και ο `BigEndianBitConverter`.

Η ανάγκη υλοποίησης των δύο αυτών διαφορετικών τρόπων αποθήκευσης προέκυψε επειδή ο ήδη υπάρχον `BitConverter` του *.NET* παίρνει το *endianness* [57] από την αρχιτεκτονική του υπολογιστή, ενώ η κωδικοποίηση απαιτεί να είναι σταθερό και στις συσκευές για λόγους επικοινωνίας.



ΕΙΚΟΝΑ 37 ENDIANNESS

The following table lists the C# built-in value types:

C# type keyword	.NET type
bool	System.Boolean
byte	System.Byte
sbyte	System.SByte
char	System.Char
decimal	System.Decimal
double	System.Double
float	System.Single
int	System.Int32
uint	System.UInt32
nint	System.IntPtr
nuint	System.UIntPtr
long	System.Int64
ulong	System.UInt64
short	System.Int16
ushort	System.UInt16

The following table lists the C# built-in reference types:

C# type keyword	.NET type
object	System.Object
string	System.String
dynamic	System.Object

ΕΙΚΟΝΑ 38 ΒΑΣΙΚΟΙ ΤΥΠΟΙ ΔΕΔΟΜΕΝΩΝ ΣΤΗ C# [58]

2.2.2 Η κλάση *EndianBinaryWriter*

Η κλάση αυτή παρέχει μεθόδους για τη μετατροπή μιας δεδομένων σε μια σειρά από bytes. Παίρνει ως ορίσματα έναν τύπο `EndianBitConverter`, ένα `Encoding` και ένα `Stream`. Το `Encoding` αφορά τον τρόπο κωδικοποίησης των αλφαριθμητικών, ενώ το `Stream` είναι μία κλάση την οποία μπορείς να χρησιμοποιήσεις για να αποθηκεύσεις και να αναζητήσεις μια αλληλουχία από bytes. Η κλάση `Stream` είναι μία αφαιρετική κλάση. Στην περίπτωσή μας χρησιμοποιούμε την υλοποίηση `MemoryStream` που δίνεται από τη βασική βιβλιοθήκη.

Οι διάφορες μέθοδοι που υλοποιούνται είναι *overloads* με χαρακτηριστική ονομασία *Write*, η οποία παίρνει ως ορίσματα όλα τους βασικούς τύπους που περιγράφηκαν παραπάνω και τους γράφει στο *Stream* μέσω του `EndianBitConverter`.

2.2.3 Η κλάση *EndianBinaryReader*

Αντίστοιχα η κλάση αυτή χρησιμοποιείται για τη μετατροπή μιας αλληλουχίας από bytes στους βασικούς τύπους της C#. Πάλι παίρνει ως ορίσματα έναν τύπο `EndianBitConverter`, ένα `Encoding` και ένα `Stream` και παρέχει μεθόδους για την αποκωδικοποίηση ενός μέρους μιας αλληλουχίας από *bytes*.

2.2.4 Η διεπαφή *ISerializablePacket*

Η διεπαφή *ISerializablePacket* ορίζει τρεις μεθόδους:

1) Μία μέθοδο για την κωδικοποίηση ενός αντικειμένου αυτής της κλάσης σε έναν *EndianBinaryWriter*.
2) Μία μέθοδο για την αποκωδικοποίηση ενός αντικειμένου αυτής της κλάσης από έναν *EndianBinaryReader*.
3) Μία μέθοδο που επιστρέφει την κλάση ως έναν πίνακα από bytes. Η μέθοδος αυτή είναι παρωχημένη, αφού μπορεί η υλοποίηση να υπάρχει πάντα από τον *EndianBinaryWriter*. Παρόλα αυτά, για λόγους πληρότητας παρέμεινε στον ορισμό της διεπαφής.

Ο σκοπός της διεπαφής είναι ο σχεδιαστής / προγραμματιστής να μπορεί να δημιουργήσει δικά του πακέτα, συγκεντρώνοντας την πληροφορία σε ένα αντικείμενο. Αυτό κάνει το σχεδιασμό πιο αποδοτικό, καθώς και τον κώδικα πιο ευανάγνωστο.

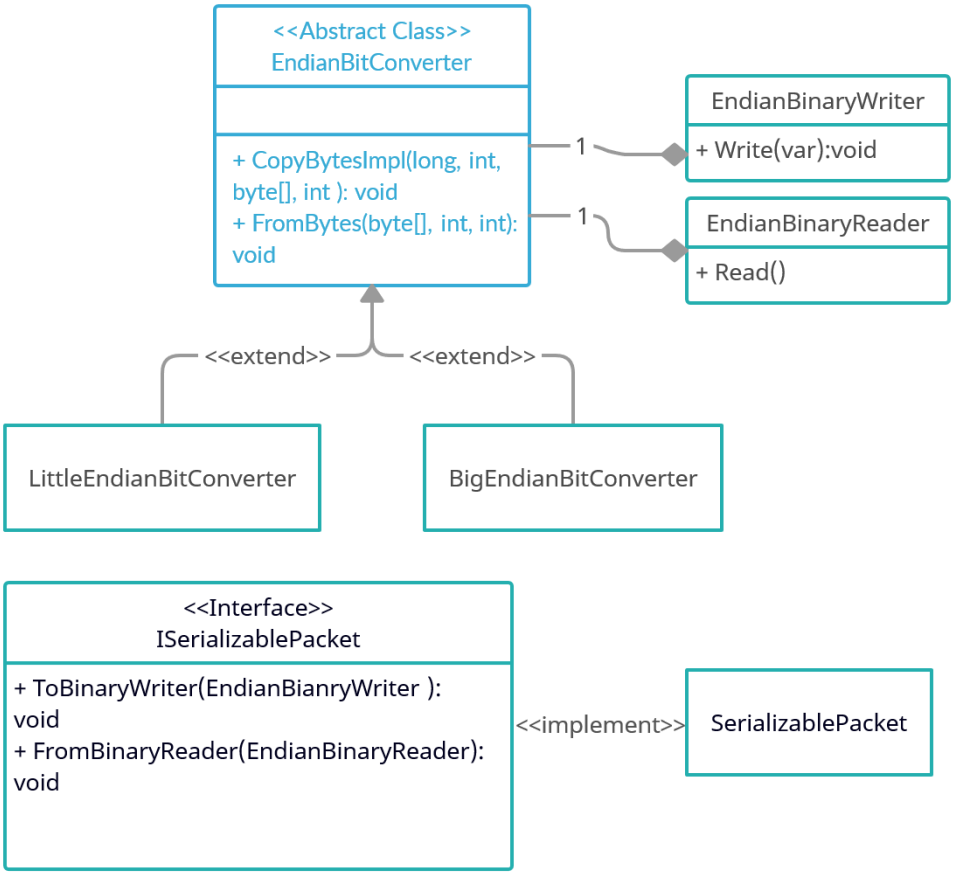
Να σημειωθεί εδώ πως ένα *ISerializablePacket* πρέπει να υλοποιηθεί από τον προγραμματιστή. Αυτό σημαίνει πως δεν υπάρχει κάποια αυτόματη μετατροπή οποιασδήποτε κλάσης υλοποιεί αυτή τη διεπαφή. Μία βελτίωση του συγκεκριμένου συστήματος κωδικοποίησης θα ήταν η εξής:

Μέσω *Reflection* [59] να διαβάζονται οι βασικοί τύποι που ορίζονται ως μέλη μίας κλάσης που υλοποιεί μια εφαρμογή και μετά μέσω τεχνικών *weaving* [60] να υλοποιείται η κλάση κατά το *compile time*. Η υλοποίηση όμως αυτής της τεχνικής είναι σχετικά πολύπλοκη και ξεφεύγει από τα πλαίσια της διπλωματικής.

Τέλος, υπάρχει μία υλοποίηση της διεπαφής, η κλάση ***SerializablePacket***, η οποία είναι αφαιρετική κλάση η οποία υλοποιεί όμως τη μέθοδο 3). Αυτό συμβαίνει διότι είχαν γραφτεί ήδη αρκετά πακέτα με αυτόν τον τρόπο, και παρόλο που η μέθοδος είναι παρωχημένη, δεν ήταν επιθυμητό να δημιουργηθεί πρόβλημα στο σύστημα.

```
[Serializable]
6 references
public class PongMessage : SerializablePacket
{
    public double clientTime;
    public double serverTime;
    1 reference
    public PongMessage() { }
    1 reference
    public PongMessage(double clientTime, double serverTime)
    {
        this.clientTime = clientTime;
        this.serverTime = serverTime;
    }
    6 references
    public override void ToBinaryWriter(EndianBinaryWriter writer)
    {
        writer.Write(clientTime);
        writer.Write(serverTime);
    }
    6 references
    public override void FromBinaryReader(EndianBinaryReader reader)
    {
        clientTime = reader.ReadDouble();
        serverTime = reader.ReadDouble();
    }
}
```

ΕΙΚΟΝΑ 39 ΑΠΛΗ ΥΛΟΠΟΙΗΣΗ ΕΝΟΣ PONG ΜΗΝΥΜΑΤΟΣ



ΕΙΚΟΝΑ 40 ΒΑΣΙΚΟ UML ΚΩΔΙΚΟΠΟΙΗΣΗΣ – ΠΕΡΙΕΧΕΙ ΤΙΣ ΣΗΜΑΝΤΙΚΟΤΕΡΕΣ ΜΕΘΟΔΟΥΣ

2.3 Αρχιτεκτονική Επικοινωνίας

Στην ενότητα αυτή αναλύεται η σχεδίαση των κλάσεων και components που αποτελούν ένα γενικό σύστημα επικοινωνίας μεταξύ ενός πελάτη και ενός διακομιστή. Ο τύπος της επικοινωνίας που υλοποιείται είναι αμφίδρομος · και ο διακομιστής και ο πελάτης μπορούν να στέλνουν και να δέχονται μηνύματα. Είναι σημαντικό να αναφερθεί πως ο σχεδιασμός έγινε βάσει διεπαφών και *components*, ώστε να είναι όσο το δυνατόν ευκολότερη η αλλαγή της υλοποίησης, κρατώντας σταθερό όμως το *API*.

2.3.1 Η διεπαφή *IMessage*

Η διεπαφή *IMessage*, όπως υποδηλώνει και το όνομα, αναπαριστά ένα εξερχόμενο μήνυμα. Τα σημαντικότερα μέλη αυτής της διεπαφής είναι ο κωδικός λειτουργίας (*opcode*), τον οποίο μπορεί να χρησιμοποιήσει το σύστημα που λαμβάνει το μήνυμα για να εκτελέσει μια ενέργεια και ένας πίνακας από byte, την κωδικοποιημένη μορφή του πακέτου που στείλαμε.

Επιπλέον ορίζουμε ένα *enum*, το *ResponseStatus*, το οποίο ορίζει την κατάσταση ενός μηνύματος στον αποστολέα, αν ζητηθεί. Οι τιμές που μπορεί να πάρει φαίνονται παρακάτω.

```
public enum ResponseStatus
{
    Default = 0,
    Success = 1,
    Timeout = 2,
    Error = 3,
    Unauthorized = 4,
    Invalid = 5,
    Failed = 6,
    NotConnected = 7,
    NotHandled = 8,
}
```

ΕΙΚΟΝΑ 41 RESPONSE STATUS

Μία βασική υλοποίηση της διεπαφής αυτής αποτελεί το **Message**.

2.3.2 Η διεπαφή *IMsgDispatcher*

Η διεπαφή αυτή ορίζει διάφορες *overload* μεθόδους αποστολής ενός μηνύματος. Στη γενικευμένη μορφή της, η αποστολή μηνύματος αποτελείται από:

1) Κωδικό λειτουργίας (*opcode*)
2) Μια αλληλουχία από bytes ή ένας τύπος ο οποίος μπορεί να μετατραπεί σε bytes, όπως σχεδιάστηκε στην προηγούμενη ενότητα.
3) Ένα *ResponseCallback*, ένα *event* που συμβαίνει αν ο παραλήπτης απαντήσει στο εισερχόμενο μήνυμα.

Κάθε *IMsgDispatcher* έχει ένα *reference* σε μια διεπαφή *IPeer*.

2.3.3 Η διεπαφή *IPeer*

Η διεπαφή αυτή επεκτείνει τη διεπαφή *IMsgDispatcher* και ταυτόχρονα αντιπροσωπεύει τη σύνδεση μεταξύ πελάτη. Σε αυτήν ο προγραμματιστής μπορεί να εγγραφή σε ένα *event*, το *MessageReceived*, το οποίο καλείται όταν ληφθεί ένα μήνυμα από αυτή τη σύνδεση. Η διεπαφή αυτή επιπλέον έχει *getters* για τη μοναδική ταυτότητά της (*Id:integer*) και για το αν είναι η σύνδεση είναι ενεργή (IsConnected:Boolean);

2.3.4 Η διεπαφή IIncommingMessage

Η διεπαφή αυτή αναπαριστά ένα εισερχόμενο μήνυμα σε έναν οποιονδήποτε παραλήπτη, είτε αυτός είναι πελάτης είτε διακομιστής. Περιέχει περίπου ό,τι έχει οριστεί και για τη διεπαφή *IMessage*, αλλά ορίζει και διάφορες μεθόδους οι οποίες μπορούν να χρησιμοποιηθούν ώστε ο παραλήπτης να απαντήσει άμεσα στο μήνυμα. Επιπλέον γνωρίζει τον *IPeer*, τη σύνδεση δηλαδή από την οποία προήλθε.

Μία γενική υλοποίηση της διεπαφής είναι το *IncomingMessage*.

2.3.5 Η διεπαφή *IClientSocket*

Η διεπαφή αυτή αναπαριστά έναν πελάτη. Επεκτείνει τη διεπαφή *IMsgDispatcher*, πράγμα που σημαίνει πως η υλοποίησή της πρέπει να περιέχει μεθόδους αποστολής μηνυμάτων. Επιπλέον, ορίζονται *events* τα οποία καλούνται κατά τη σύνδεση και αποσύνδεση του πελάτη από το διακομιστή, καθώς και μια μέθοδος *Connect(string ip, int port)* η οποία ξεκινά τη διαδικασία σύνδεσης.

Τέλος, δίνει τη δυνατότητα πρόσθεσης ενός *IPacketHandler*, μίας διεπαφής η οποία λειτουργεί ως *event* το οποίο καλείται όταν ένα μήνυμα με συγκεκριμένο κωδικό ληφθεί από τον πελάτη.

2.3.6 Η διεπαφή *IServerSocket*

Η διεπαφή αυτή αναπαριστά έναν διακομιστή. Ορίζει σημαντικά *events* τα οποία καλούνται κατά της διάρκεια λειτουργίας του διακομιστή. Αυτά είναι:

1) ***Connected***, για όταν δημιουργείται μια σύνδεση *IPeer*.
2) ***Disconnected***, για όταν καταστρέφεται μία σύνδεση *IPeer*.

Και στις 2 περιπτώσεις τα events καλούνται με όρισμα μια διεπαφή IPeer. Επιπλέον, διαθέτει μεθόδους για την εκκίνηση του διακομιστή [**void Listen(int port)**] και για την παύση του [**void Stop()**].

2.3.7 *Mirror Networking* [61]

Το *Mirror* αποτελεί μία βιβλιοθήκη ελεύθερου πηγαίου κώδικα (*open source*) για τη *Unity*. Ορίζει διάφορες χρήσιμες κλάσεις και *components* για τη διευκόλυνση της ανάπτυξης ενός παιχνιδιού πολλαπλών παικτών. Ως εκ τούτου, είναι γραμμένη με συγκεκριμένο τρόπο και δεν εξυπηρετεί ακριβώς τις προδιαγραφές του σχεδιασμού, γι' αυτό και ορίσαμε τις παραπάνω διεπαφές.

Παρόλα αυτά, προσφέρει έτοιμες υλοποιήσεις διάφορων πρωτοκόλλων επικοινωνίας, γεγονός που την καθιστά εξαιρετικά χρήσιμη. Ένα πρωτόκολλο επικοινωνίας υλοποιεί την αφαιρετική κλάση Transport, η οποία περιέχει παρόμοια *events* με αυτά που αναφέραμε στις διεπαφές και είναι επέκταση της `MonoBehaviour` της *Unity*. Ένα Transport όμως μπορεί να λειτουργήσει και ως πελάτης και ως διακομιστής, γεγονός που μας επιτρέπει εύκολα να το προσαρμόσουμε στις παραπάνω διεπαφές.

Έτσι, έχουμε τις κλάσεις `BaseTransportSocket`, `TransportServerSocket`, `TransportServerPeer`, `TransportClientSocket`, `TransportClientPeer`. Η `BaseTransportSocket` αποτελεί ένα `MonoBehaviour` *το* οποίο συνεργάζεται με ένα

οποιοδήποτε *Transport component*. Οι κλάσεις Transport Server και Transport Client αποτελούν επεκτάσεις της *BaseTransportSocket*, αλλά ταυτόχρονα υλοποιούν τις διεπαφές *IServerSocket* και *IClientSocket* αντίστοιχα.

Είναι χρήσιμο να αναφέρουμε τις υλοποιήσεις διάφορων πρωτοκόλλων σε Transport components [62]:

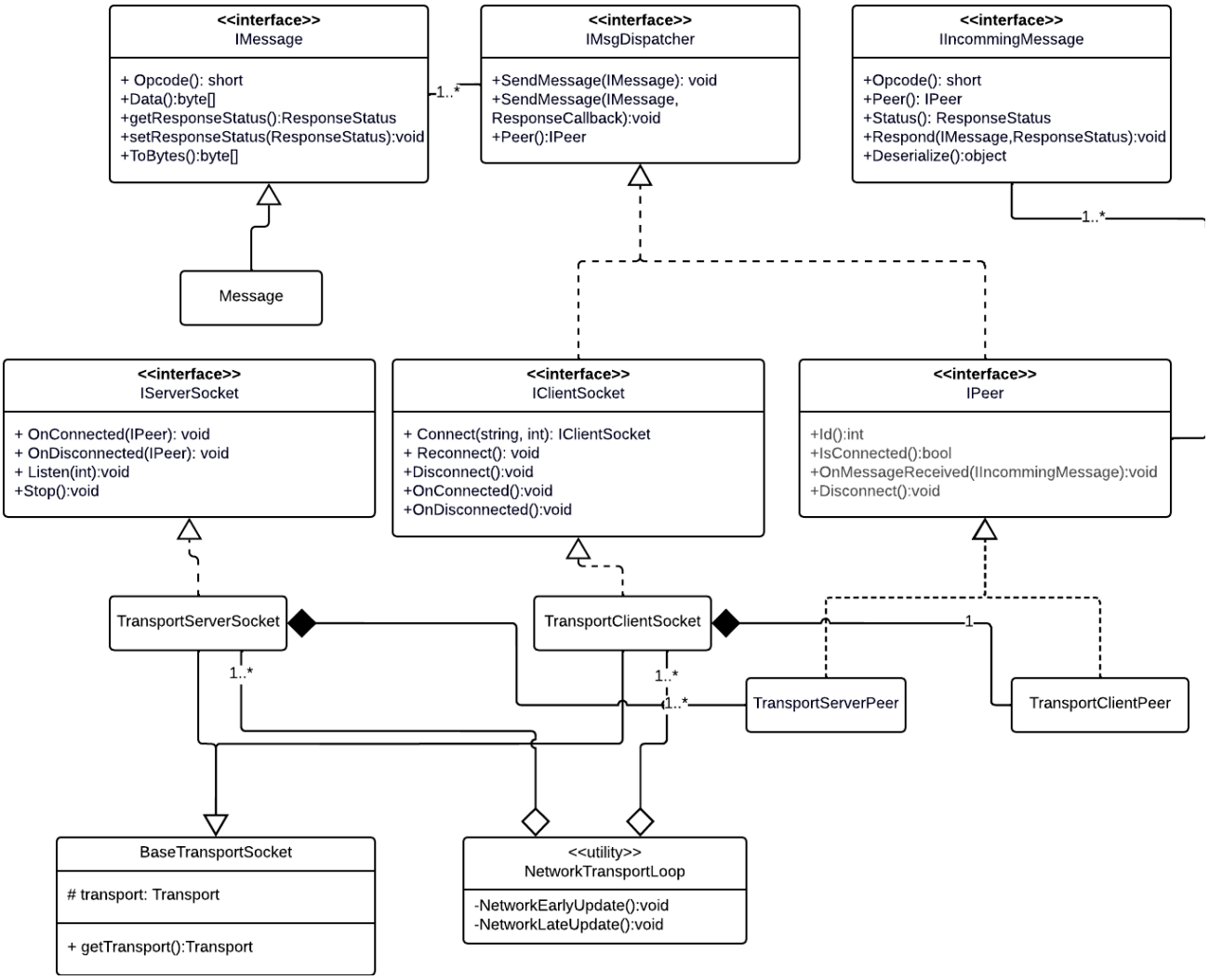
1) ***Telepathy***, ένα Transport που υλοποιεί το πρωτόκολλο *TCP*. Είναι φτιαγμένο για χρήση σε *MMOs* (*Massive Multiplayer Online games*) και γραμμένο εξολοκλήρου στη C#.
2) ***Simple Web Sockets***, πρωτόκολλο για χρήση σε εφαρμογές που θα τρέξουν σε φυλλομετρητή.
3) ***KCP***, ένα γρήγορο και αξιόπιστο πρωτόκολλο το οποίο επιτυγχάνει τη μείωση του μέσου όρου της καθυστέρησης (*latency*) κατά 30-40% με κόστος 10-20% παραπάνω εύρος ζώνης (bandwidth) σε σχέση με το *TCP*. Το πρωτόκολλο δεν είναι υπεύθυνο για την υλοποίηση του εσωτερικού πρωτοκόλλου αποστολής και λήψης (όπως το *UDP*) . Αυτό παρέχεται στο *KCP* εξωτερικά. [63]

Κατά την υλοποίηση της εφαρμογής χρησιμοποιήθηκε το *KCP Transport* διότι αυτό πρότειναν οι σχεδιαστές του Mirror. Να σημειωθεί επιπλέον πως όλες αυτές οι υλοποιήσεις αφορούν σύνδεση μέσω δικτύου Wi-fi, αν και κάποιος θα μπορούσε να τροποποιήσει τις διεπαφές ώστε να υποστηρίζει και άλλα πρωτόκολλα, όπως Bluetooth ή Wi-fi Direct.

2.3.8 Κατανάλωση των μηνυμάτων

Παραπάνω σχεδιάσαμε την αρχιτεκτονική ενός *API* επικοινωνίας. Πλέον το μόνο που μένει για να ολοκληρωθεί αυτό το γενικό σύστημα είναι η κατανάλωση εισερχόμενων – εξερχόμενων μηνυμάτων από διακομιστή και πελάτη. Η κατανάλωση αυτή θα επιτευχθεί με τη δημιουργία ενός συστήματος εντός της Unity το οποίο θα καταναλώνει τα μηνύματα σε κάθε ανανέωση της εφαρμογής. Για την υλοποίηση των παραπάνω χρησιμοποιείται το *PlayerLoopSystem*, ένα *low level API* της *Unity* το οποίο δίνει τη δυνατότητα εισαγωγής συστημάτων σε καίρια σημεία του κύκλου ζωής της *Unity* και όχι μόνο στον κύκλο ζωής ενός *component / monobehaviour*.

Η κλάση που εκτελεί αυτή τη διαδικασία λέγεται *NetworkTransportLoop*. Συγκεντρώνει όλα τα *BaseTransportSocket* και εκτελεί μία πρώιμη ανανέωση (*early update*), κατά την οποία το σύστημα επεξεργάζεται και καταναλώνει τα εισερχόμενα μηνύματα και μία καθυστερημένη ανανέωση (*late update*), κατά την οποία το σύστημα επεξεργάζεται και αποστέλλει τα εξερχόμενα μηνύματα. Το σύστημα είναι σχεδιασμένο έτσι ώστε να αποφεύγονται τυχόν καθυστερήσεις μεταξύ εισερχόμενων και εξερχόμενων. Η *Unity* εκτελεί την *Update* μέθοδο των *MonoBehaviour* στο ενδιάμεσο των δύο αυτών ανανεώσεων, εξού και οι όροι *early update, late update*.



ΕΙΚΟΝΑ 42 ΒΑΣΙΚΟ UML ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΔΙΚΤΥΟΥ

2.4 Υλοποίηση πλαισίου (*framework*) επικοινωνίας

Πλέον έχουν σχεδιαστεί τα συστήματα με τα οποία κωδικοποιούνται, αποκωδικοποιούνται και στέλνονται τα δεδομένα. Η παρούσα ενότητα παρουσιάζει το σχεδιασμό και την υλοποίηση, μέσω των προηγούμενων συστημάτων ενός πλαισίου επικοινωνίας κατά το οποίο η συσκευή ελεγκτής - στην περίπτωσή μας, το κινητό – αποστέλλει δεδομένα από «συσκευές» εισόδου της στην ελεγχόμενη συσκευή – στην περίπτωσή μας το *HoloLens*. Επεξηγείται πως το Σύστημα Εισόδου της *Unity* βοήθησε στην κωδικοποίηση – αποκωδικοποίηση των δεδομένων εισόδου του κινητού. Έπειτα αναπτύσσονται ο διακομιστής που τρέχει στο *HoloLens* και ο πελάτης που τρέχει στο κινητό.

2.4.1 Σκοπός σχεδίασης πλαισίου

Ο σκοπός του πλαισίου είναι η παροχή της δυνατότητας στον σχεδιαστή μίας εφαρμογής να αξιοποιήσει εύκολα τις διάφορες εισόδους μίας συσκευής σα να ήταν είσοδοι μιας άλλης. Για να επιτευχθεί αυτό, χρησιμοποιείται το Σύστημα Εισόδου της *Unity*, ούτως ώστε να συσσωρευτούν όλοι οι είσοδοι της συσκευής ελεγκτή, να αποσταλούν στο δίκτυο και να αποκωδικοποιηθούν στην ελεγχόμενη συσκευή ως είσοδοι εικονικών συσκευών ίδιου τύπου. Με αυτόν τον τρόπο, ο προγραμματιστής που χρησιμοποιεί αυτό το πλαίσιο δε χρειάζεται να κάνει τίποτα άλλο πέραν του να αντιμετωπίσει τις συσκευές αυτές ακριβώς όπως θα αντιμετώπιζε μια οποιαδήποτε άλλη συσκευή μέσω του Συστήματος Εισόδου.

Αναφέρεται η ταύτιση των παρακάτω όρων για διευκόλυνση της ανάγνωσης:

- **Πελάτης – Συσκευή Ελεγκτής – Κινητό**
- **Διακομιστής – Ελεγχόμενη Συσκευή – *Hololens* 2**

2.4.1 Κωδικοποίηση των εισόδων του ελεγκτή-κινητού

Η κωδικοποίηση των εισόδων του ελεγκτή-κινητού για την αποστολή τους στο δίκτυο έγινε με το σύστημα κωδικοποίησης το οποίο αναπτύχθηκε παραπάνω με τη βοήθεια του Συστήματος Εισόδου της *Unity*. Συγκεκριμένα, έχουμε τις εξής σημαντικές κλάσεις:

- `BaseInput`, η οποία είναι αφαιρετική και επεκτείνει την κλάση *SerializablePacket*, ώστε να μπορεί να μεταδοθεί στο δίκτυο. Η κλάση αυτή ορίζει αφαιρετικές μεθόδους *SetUp*, η οποία χρησιμοποιείται από τον πελάτη ώστε να αντλήσει τα δεδομένα εισόδου από μία συσκευή, και *QueueInput*, η οποία καλείται στον διακομιστή ώστε να περάσει την είσοδο στις αντίστοιχες εικονικές συσκευές που έχουν δημιουργηθεί.

  Ο συσχετισμός των υλοποιήσεων της κλάσης αυτής – μία υλοποίηση της *BaseInput* για κάθε συσκευή παρέχεται από μία στατική κλάση, την `InputFactory` η οποία και επιστρέφει συγκεκριμένη υλοποίηση ανάλογα με τον τύπο της συσκευής. Έχουμε λοιπόν τους εξής συσχετισμούς:

Τύπος Συσκευής	**Υλοποίηση *BaseInput***
Accelerometer	*AccelerometerInput*
AmbientTemperatureSensor	*AmbientTemperatureInput*
AttitudeSensor	*AttitudeInput*
Gamepad	*GamepadInput*
GravitySensor	*GravityInput*
Gyroscope	*GyroscopeInput*
HumiditySensor	*HumidityInput*
Keyboard	*KeyboardInput*
LightSensor	*LightInput*
LinearAccelerationSensor	*LinearAccelerationInput*
MagneticFieldSensor	*MagneticFieldinput*
Mouse	*MouseInput*
PressureSensor	*PressureInput*
ProximitySensor	*ProximityInput*
StepCounter	*StepCounterInput*
Touchscreen	*TouchscreenInput*
TrackedDevice	*TrackedDeviceInput*

Είναι σημαντικό να αναφέρουμε πως αυτή η διαδικασία μπορεί να αυτοματοποιηθεί μέσω *Reflection* και *Metadata*, ώστε να μη χρειάζεται να επεκτείνεται χειροκίνητα η κλάση *InputFactory*. Οι συσκευές όμως είναι συνήθως συγκεκριμένες και σχετικά λίγες, οπότε επιλέχθηκε ο πιο άμεσος τρόπος.

- `DeviceDescription`, η οποία επεκτείνει την κλάση `SerializablePacket`. Η κλάση αυτή χρησιμοποιείται για την περιγραφή μίας συσκευής εάν ο σχεδιαστής θέλει να την εντάξει στο σύστημα ελεγκτή-ελεγχόμενης συσκευής (Κινητό-*HoloLens*). Είναι σχεδιασμένη έτσι ώστε να φαίνεται στον *Editor* της *Unity*, και άρα ο οποιασδήποτε να μπορεί να επιλέξει τι συσκευές θα συμμετέχουν στο σύστημα χωρίς να χρειάζεται να γράψει κώδικα.
- `InputData`**,** η οποία επεκτείνει την κλάση `SerializablePacket`. Λειτουργεί ως πακέτο δεδομένων το οποίο περιέχει την περιγραφή της συσκευής, τη συνολική είσοδο της συσκευής καθώς και την κατάστασή της – εάν δηλαδή η συσκευή μόλις προστέθηκε στο σύστημα, αφαιρέθηκε κ.ο.κ. Έτσι παρέχεται η δυνατότητα σε πραγματικό χρόνο της προσθαφαίρεσης μιας οποιασδήποτε συσκευής από τον ελεγκτή στην ελεγχόμενη συσκευή.
- `SubscriptionData`, η οποία επεκτείνει την κλάση `SerializablePacket`. Αποτελεί το μήνυμα που στέλνεται από τον ελεγκτή στην ελεγχόμενη συσκευή όταν εγκαθιδρύεται η σύνδεση. Περιέχει μια λίστα από την περιγραφή των συσκευών που θα αποστέλλονται στο HoloLens. Επιπλέον, περιέχει και κάποια άλλα δεδομένα που εξετάζονται σε παρακάτω ενότητα και αφορούν την υλοποίηση ενός ελεγκτή με ελευθερία 6 βαθμών (*6DOF Controller*).

- `DeviceData`, η οποία επεκτείνει την κλάση `SerializablePacket`. Στέλνεται από το κινητό στο *HoloLens* κάθε *frame* και περιέχει όλα τα δεδομένα εισόδου που έχουν συλλεχθεί από το κινητό, καθώς και τα διάφορα *events* που έχει αναπτύξει ο σχεδιαστής της εφαρμογής. Τέλος, εμπεριέχει ένα μήνυμα *ping* στο οποίο απαντάει το *HoloLens* με ένα μήνυμα *pong*, στην περίπτωση που χρειάζεται η γνώση της καθυστέρησης (*latency, delay*) μεταξύ των δύο για καλύτερο συγχρονισμό του συνολικού συστήματος.

2.4.2 Υλοποίηση ενός συστήματος με *events*

Παρόλο που η παραπάνω υλοποίηση της κωδικοποίησης εν γένει υλοποιεί και ένα σύστημα με *events* λόγω της φύσης του Συστήματος Εισόδου της *Unity*, στην παρούσα ενότητα παρουσιάζεται ένα επιπλέον σύστημα για *events*.

Καταρχάς, να σημειωθεί πως το σύστημα με *events* που προκύπτει φυσικά από το Σύστημα Εισόδου βασίζεται στη δυνατότητα που παρέχεται της δημιουργίας μιας οποιασδήποτε εικονικής συσκευής με αόριστο αριθμών εισόδων και της δέσμευσης των εισόδων αυτών σε κάποια ενέργεια μέσω του Χάρτη Ενεργειών. Η υλοποίηση ενός διαφορετικού από αυτού συστήματος προέκυψε περισσότερο ως παροχή ενός εναλλακτικού τρόπου αποστολής ενός *event*, διότι η δημιουργία μιας εικονικής συσκευής απαιτεί περισσότερη προσοχή και γνώση.

Παρακάτω παρουσιάζονται οι πλέον σημαντικές κλάσεις του συστήματος αυτού καθώς και η λειτουργία τους.

1) `EventIdentifier`, κλάση η οποία δίνει μία μοναδική ταυτότητα σε ένα event. Η ταυτότητα αυτή δεν είναι τίποτα άλλο παρά ένας ακέραιος αριθμός ο οποίος στέλνεται μέσω του συστήματος δικτύωσης ως *opcode*. Να σημειωθεί πως επεκτείνει την κλάση *ScriptableObject* της *Unity*, η οποία βοηθάει στη δημιουργία *asset* που ζουν μέσα στη *Unity* έναντι `MonoBehaviour` το οποίο ζει σε ένα *GameObject*.
2) `EventManager`, η οποία είναι αφαιρετική κλάση και επεκτείνει την `MonoBehaviour`. Πρέπει να υλοποιηθούν 2 συναρτήσεις. Η πρώτη είναι η *SetEventHandler*, η οποία ορίζει στο σύστημα ένα *event* με συγκεκριμένο *EventIdentifier* καθώς και τρόπο να επεξεργαστεί το εισερχόμενο event ως *IIncommingMessage*. Η δεύτερη είναι η *RemoveEventHandler*, η οποία αφαιρεί το *event* με συγκεκριμένο *EventIdentifier*. Υλοποιήσεις του *EventManager* παρέχονται και για τον πελάτη και για τον διακομιστή, ώστε να επιτευχθεί η αμφίδρομη επικοινωνία.

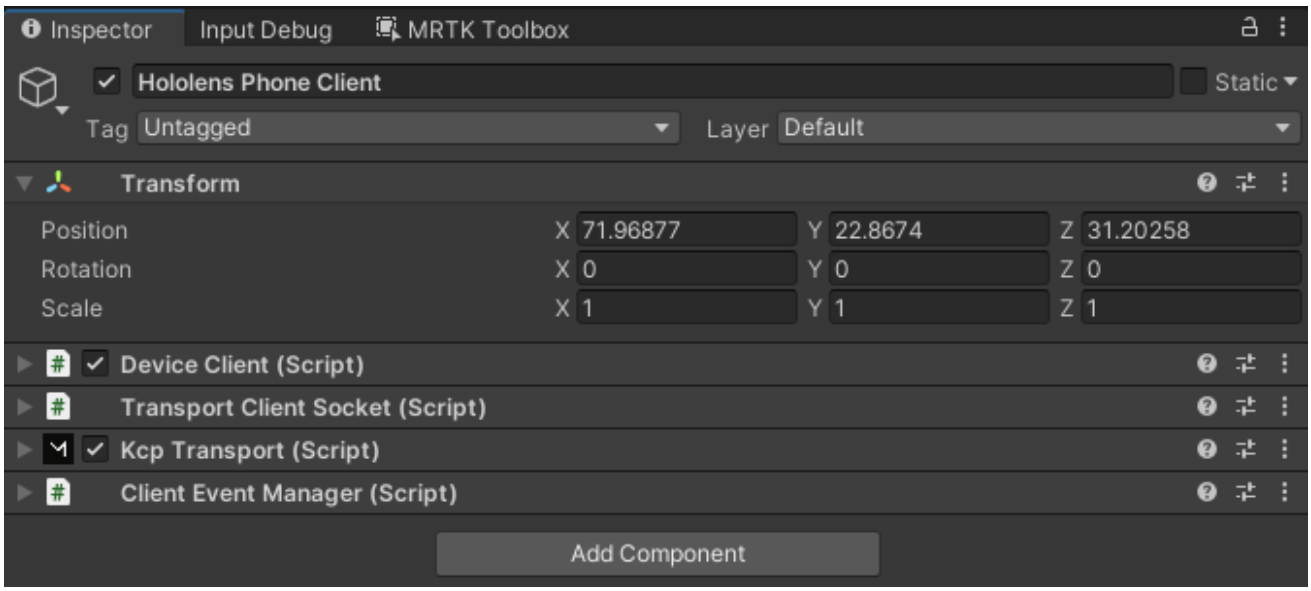
Η αποστολή και η κατανάλωση των *events* γίνεται μέσω του συστήματος δικτύωσης το οποίο υλοποιήθηκε παραπάνω, ως μηνύματα τα οποία εκτελούν ένα *event* όταν παραληφθούν.

2.4.3 Υλοποίηση πελάτη-διακομιστή · Οι κλάσεις *DeviceClient* και *DeviceServer*

DeviceClient

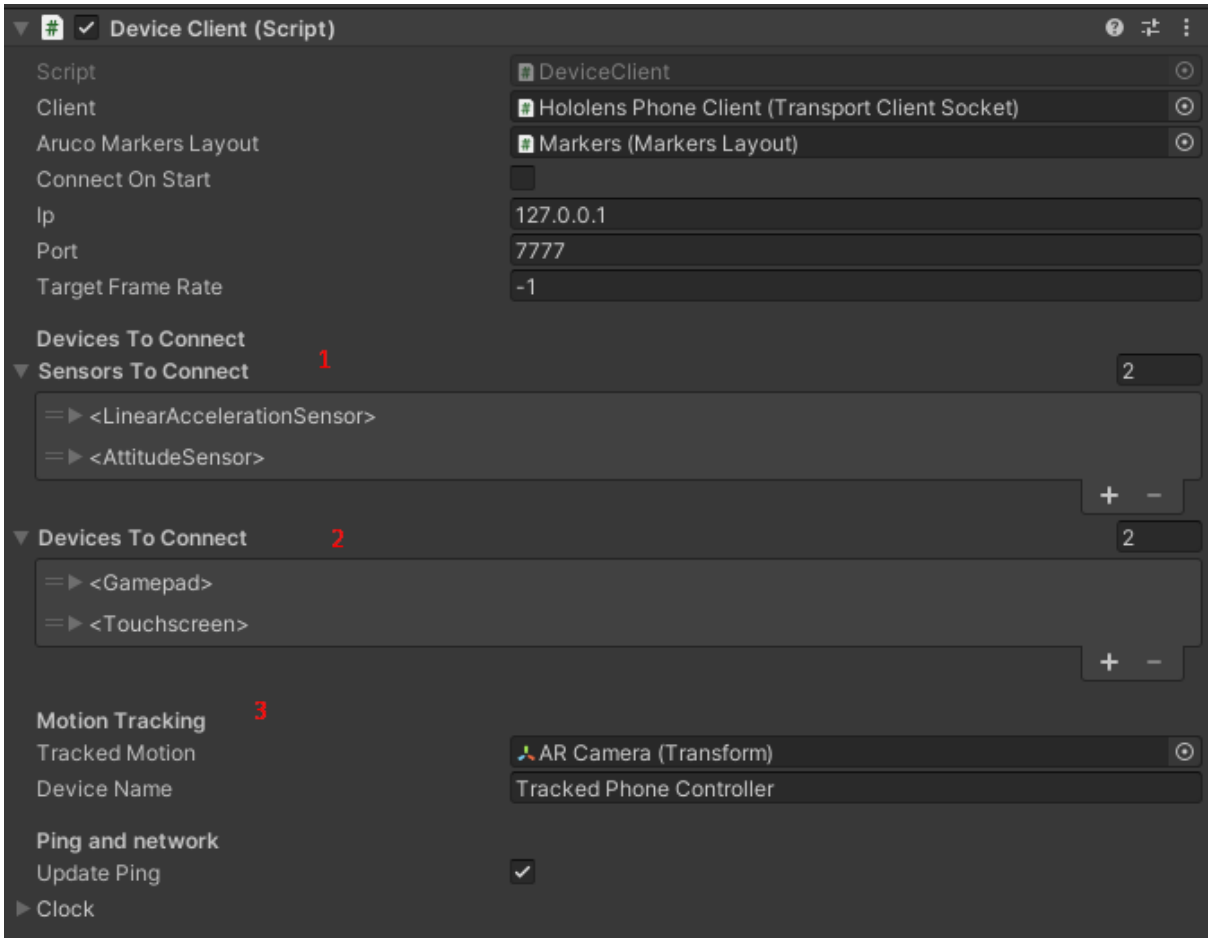
Η κλάση `DeviceClient` επεκτείνει την `MonoBehaviour` της *Unity* ώστε να λειτουργεί ως *component* και να καθίσταται δυνατή η αλλαγή των ρυθμίσεών της από τον *editor*. Αρμοδιότητές της είναι η σύνδεση με την ελεγχόμενη συσκευή και η συλλογή και αποστολή δεδομένων και ενεργειών από τις – κατ' επιλογή του σχεδιαστή – συσκευές του ελεγκτή.

Τα δεδομένα αυτά συλλέγονται μέσω του Συστήματος Εισόδου της *Unity* και στέλνονται μέσω του συστήματος δικτύωσης στην ελεγχόμενη συσκευή. Στις παρακάτω εικόνες παρουσιάζονται συνολικά τα *components* που πρέπει να βρίσκονται σε ένα *gameobject* ώστε αυτό να εκτελεί τη λειτουργία του *DeviceClient* καθώς και οι διάφορες ρυθμίσεις που ο σχεδιαστής μπορεί να αλλάξει.



ΕΙΚΟΝΑ 43 COMPONENTS ΤΟΥ DEVICECLIENT

Εδώ πλέον φαίνεται και η φιλοσοφία της σχεδίασης του πλαισίου, όπου η τελική συμπεριφορά της συσκευής πελάτη εξαρτάται από τη συνεργασία των διάφορων αυτών *component*. Αν για παράδειγμα κάποιος ήθελε να χρησιμοποιήσει άλλο πρωτόκολλο επικοινωνίας, αρκεί να αφαιρούσε το `KcpTransport` και να το αντικαθιστούσε με μια δική του υλοποίηση ή ένα άλλο έτοιμο *component*.



ΕΙΚΟΝΑ 44 ΡΥΘΜΙΣΕΙΣ ΤΟΥ CLIENT

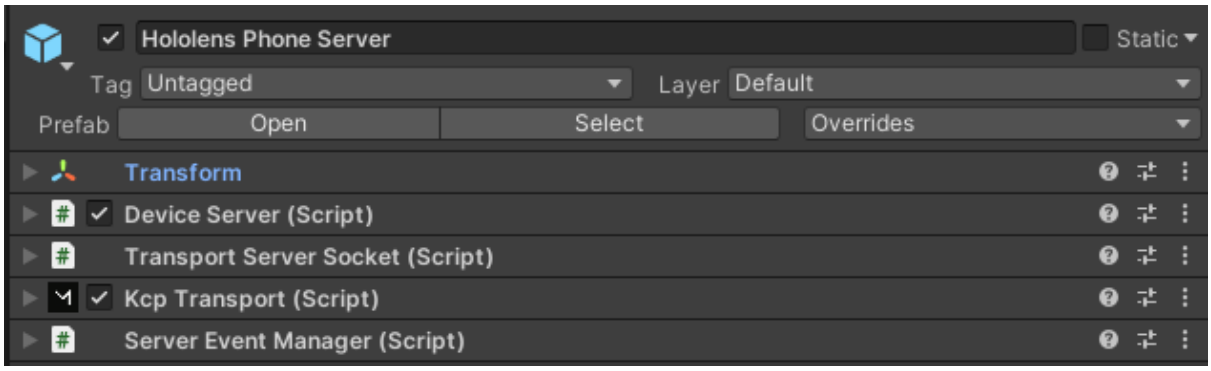
Παραπάνω φαίνονται οι ρυθμίσεις που μπορεί να αλλάξει κάποιος στον πελάτη – ελεγκτή. Τα σημεία ενδιαφέροντος αφορούν:

1. Τους αισθητήρες που θέλει κάποιος να στείλει στον διακομιστή
2. Τις συσκευές που θέλει κάποιος να στείλει στον διακομιστή
3. Το *Transform* που δίνει το ***pose*** (θέση και προσανατολισμός), αν αυτό υπάρχει. Θα αναπτυχθεί περαιτέρω στην επόμενη ενότητα.

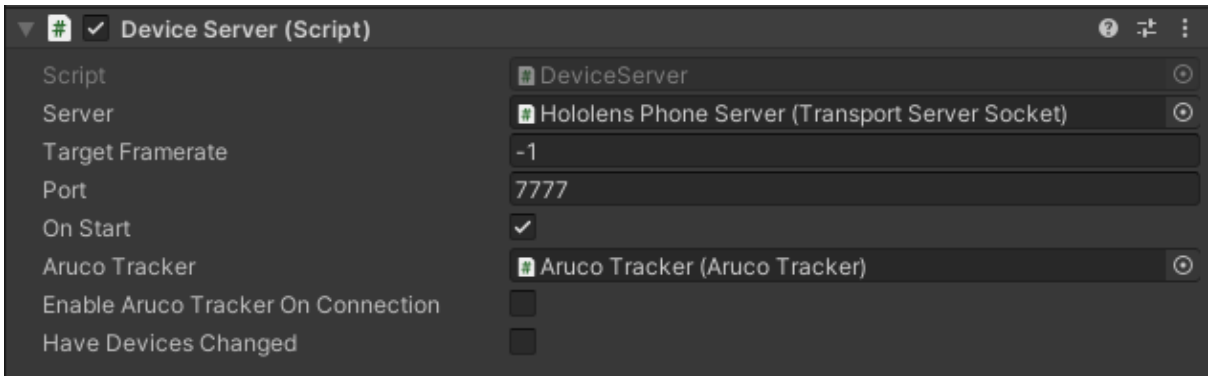
DeviceServer

Η κλάση `DeviceServer`, όπως η `DeviceClient`, επεκτείνει την `MonoBehaviour` της *Unity*. Λειτουργεί ως διακομιστής στην ελεγχόμενη συσκευή όπου και συνδέεται ο ελεγκτής. Όταν συνδέεται ένας ελεγκτής, τότε δημιουργούνται εικονικές συσκευές στην ελεγχόμενη συσκευή με βάση τα στοιχεία που στάλθηκαν. Έπειτα ο διακομιστής αναλαμβάνει την αποκωδικοποίηση των μηνυμάτων που λαμβάνει και την αλλαγή της κατάστασης των συσκευών μέσω του Συστήματος Εισόδου. Τέλος, αποστέλλει πίσω στον πελάτη ένα μήνυμα *pong* ώστε να συγχρονιστεί το δικτυακό ρολόι του συστήματος.

Στις παρακάτω εικόνες παρουσιάζονται συνολικά τα *components* που πρέπει να βρίσκονται σε ένα *gameobject* ώστε αυτό να εκτελεί τη λειτουργία του *DeviceServer* καθώς και οι διάφορες ρυθμίσεις που ο σχεδιαστής μπορεί να αλλάξει.

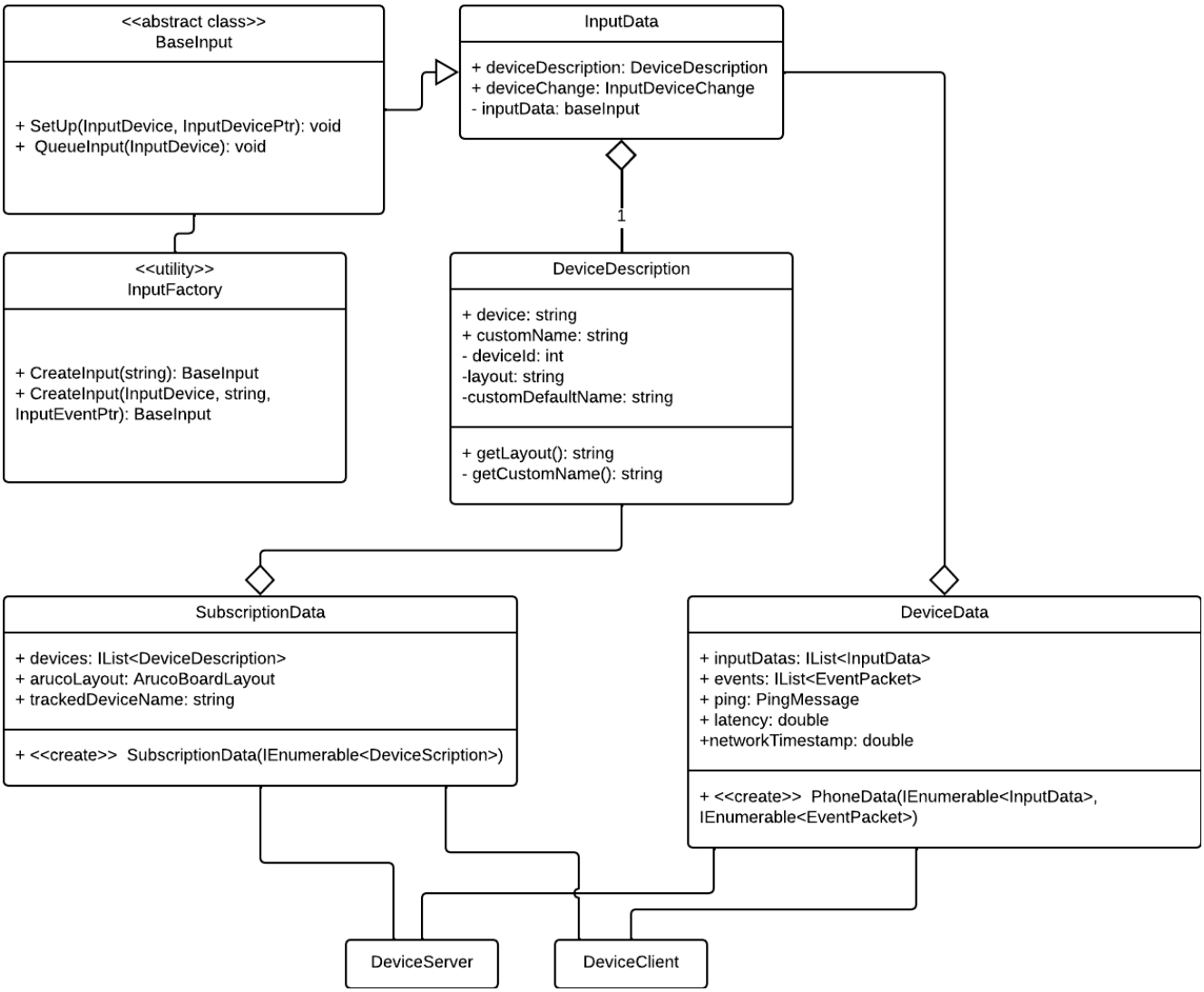


ΕΙΚΟΝΑ 45 COMPONENTS ΤΟΥ DEVICESERVER



ΕΙΚΟΝΑ 46 ΡΥΘΜΙΣΕΙΣ ΤΟΥ SERVER

Παρακάτω δίνονται ένα βασικό *UML* με τις κυριότερες κλάσεις που αναφέρθηκαν παραπάνω καθώς. Σημειώνεται πως δεν καταγράφεται καμία συσχέτιση των κλάσεων του συστήματος δικτύου και των κλάσεων του πλαισίου προς αποφυγή συμφόρησης των διαγραμμάτων. Παρόλα αυτά είναι προφανές πως το πλαίσιο χρησιμοποιεί το σύστημα δικτύου για την υλοποίηση της επικοινωνίας.



EIKONA 47 FRAMEWORK UML

Κεφάλαιο 3 – Σχεδίαση ελεγκτή με 6 βαθμούς ελευθερίας (*6DOF Controller*)

Η ενότητα αυτή περιγράφει το σχεδιασμό και την υλοποίηση ενός *6DOF Controller* με χρήση του πλαισίου που αναπτύχθηκε. Επιπλέον, στο τέλος της ενότητας παρουσιάζονται συνοπτικά 2 επιπλέον αλληλεπιδράσεις που προσφέρει μπορεί κάποιος να σχεδιάσει.

3.1 Σκοπός

Στην ενότητα 1 παρουσιάστηκαν οι διάφοροι τρόποι αλληλεπίδρασης που προσφέρει το HoloLens. Αν εξαιρέσουμε τις φωνητικής εντολές και την αλληλεπίδραση μόνο μέσω δείκτη κεφαλής – βλέμματος, οι υπόλοιποι τύποι αλληλεπίδρασης αφορούν κυρίως κίνηση των χεριών ή χειρονομίες. Τα χέρια όμως και στις 2 περιπτώσεις πρέπει να βρίσκονται μέσα στο πλαίσιο ανίχνευσης του HoloLens 2. Παρόλο δηλαδή που αποτελούν *6DOF* ελεγκτές, η δυνατότητα που προσφέρουν είναι περιορισμένη με βάση τι βλέπει η συσκευή.

Σκοπός αυτής της ενότητας είναι ο σχεδιασμός και υλοποίηση ενός αλγορίθμου ο οποίος, χρησιμοποιώντας το παραπάνω πλαίσιο επικοινωνίας, θα μετατρέψει το κινητό σε έναν *6DOF* ελεγκτή για το HoloLens, μειώνοντας όσο το δυνατόν περισσότερο την ανάγκη η ίδια η συσκευή να βλέπει το κινητό.

3.2 Αλγόριθμος ανίχνευσης του κινητού 6DOF

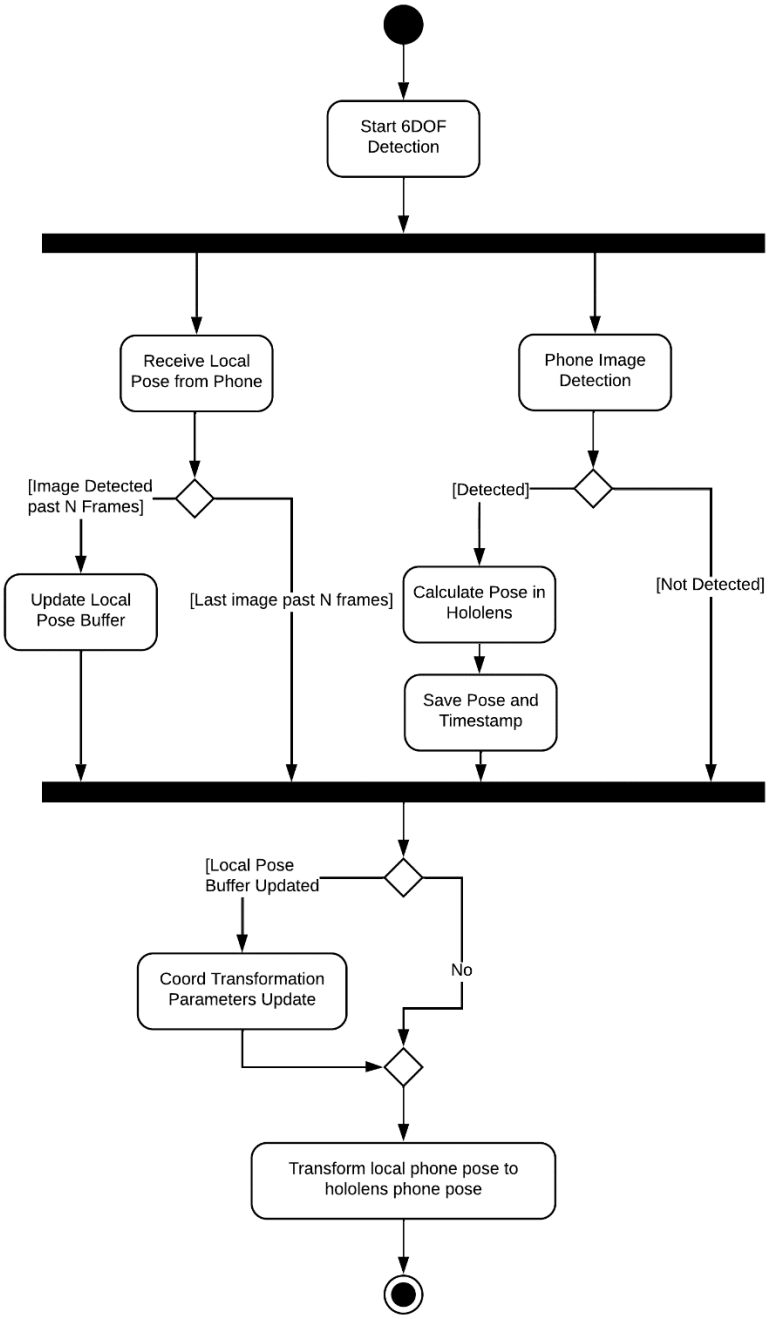
3.2.1 Σχεδιασμός του αλγορίθμου

Ο αλγόριθμος ο οποίος υλοποιεί την παραπάνω συμπεριφορά βασίζεται στη δυνατότητα που παρέχεται στα κινητά της ανάπτυξης εφαρμογών επαυξημένης πραγματικότητας. Ουσιαστικά η συσκευή πρέπει να μπορεί να τρέξει κάποιον αλγόριθμο οπτικής οδομετρίας, ώστε:

- Το κινητό εντοπίζει τη θέση του και τον προσανατολισμό του στο χώρο (**πόζα**) πάντα με βάση το δικό του σημείο αναφοράς. Συνήθως το σημείο αναφοράς είναι η θέση και ο προσανατολισμός του κινητού όταν ενεργοποιήθηκε ο αλγόριθμος. Παρόλα αυτά, το σημείο αναφοράς μπορεί να αλλάξει όσο ο αλγόριθμος οπτικής οδομετρίας βελτιώνει την «αντίληψη» που έχει για το περιβάλλον.
- Η πόζα του κινητού καταγράφεται, κωδικοποιείται και στέλνεται στο HoloLens.
- Στην οθόνη του κινητού προβάλλεται μία εικόνα. Το HoloLens τρέχει έναν αλγόριθμο επεξεργασίας εικόνας, ο οποίος ανιχνεύει την εικόνα. Έπειτα κατασκευάζει την πόζα του κινητού στο δικό του σύστημα αναφοράς, εφόσον εντοπίσει την εικόνα στο πλαίσιο της κάμεράς του.
- Τη στιγμή του εντοπισμού, καταγράφεται η πόζα του κινητού βάσει του δικού του συστήματος και η πόζα του κινητού βάσει την εικόνα που προβάλλεται στην οθόνη του. Λόγω καθυστερήσεων του δικτύου, καθίσταται αναγκαίος ο συγχρονισμός δεδομένων.

- Από τη στιγμή του εντοπισμού και μετά, γίνεται συνεχώς ένας μετασχηματισμός μεταξύ του συστήματος αναφοράς του κινητού, το οποίο παρέχει τη δική του πόζα και του συστήματος αναφοράς του HoloLens, σύμφωνα με τις πόζες που κατεγράφησαν. Σε θεωρητικό επίπεδο πλέον δε χρειάζεται ο εντοπισμός του κινητού μέσω της κάμερας του HoloLens, αλλά αρκεί ο μετασχηματισμός της πόζας του κινητού ώστε το HoloLens να ξέρει πάντα που βρίσκεται το κινητό στο δικό του σύστημα.

Τα παραπάνω περιγράφονται καλύτερα με το παρακάτω διάγραμμα δραστηριότητας (*activity diagram*), το οποίο περιγράφει τον αλγόριθμο εντοπισμού που τρέχει στο HoloLens.



ΕΙΚΟΝΑ 48 ΔΙΑΓΡΑΜΜΑ ΔΡΑΣΤΗΡΙΟΤΗΤΑΣ ΓΙΑ ΤΗΝ ΥΛΟΠΟΙΗΣΗ 6DOF CONTROLLER

Για τον παραπάνω σχεδιασμό, είναι σημαντικό να αναφερθεί πως οι χρονικές στιγμές εντοπισμού της εικόνας στο κινητό και της πόζας που στέλνει το κινητό είναι σίγουρα διαφορετικές, διότι προέρχονται από 2 διαφορετικά συστήματα. Ως εκ τούτου, αν η πόζα που εντοπίστηκε από την ανίχνευση της εικόνας στο κινητό και η πόζα που στέλνει το ίδιο το κινητό στο σύστημα είναι διαφορετικές επειδή τα συστήματα είναι ασυγχρόνιστα, θα έχει ως αποτέλεσμα τον λανθασμένο υπολογισμό των παραμέτρων που απαιτούνται για την αλλαγή του συστήματος αναφοράς.

Η λύση που προτείνεται είναι ο συγχρονισμός του δικτυακού ρολογιού των 2 συστημάτων και η αποθήκευση της πόζας που στέλνει το κινητό στο HoloLens σε έναν *buffer* ώστε να είναι δυνατός ο εντοπισμός των πιο συμβατών καταστάσεων για να δουλεύει άρτια ο αλγόριθμος.

3.3 Ανίχνευση της εικόνας στο κινητό

Για την ανίχνευση της εικόνας στο κινητό απαιτείται η πρόσβαση στην κάμερα του HoloLens, με σκοπό την ανάγνωση ενός *frame* και τη χρήση του στην ανίχνευση. Υπάρχουν βιβλιοθήκες οι οποίες εκτελούν ακριβώς την παραπάνω διαδικασία [64], [65], αλλά δυστυχώς τα αποτελέσματα της ανίχνευσης δεν ήταν ακριβή και επιπλέον δε δίνεται η δυνατότητα αλλαγής του ρυθμού με τον οποίο γίνεται η ανίχνευση, με αποτέλεσμα την μείωση των *frames* με τα οποία έτρεχε η εφαρμογή. Κάτι τέτοιο δεν είναι καθόλου επιθυμητό, ειδικά σε εφαρμογές μικτής πραγματικότητας όπου το μικτό περιβάλλον πρέπει να ανανεώνεται με ένα σταθερό και σχετικά γρήγορο ρυθμό για τη βέλτιστη εμπειρία του χρήστη. Εν τέλη, η ανίχνευση έγινε με τη χρήση της βιβλιοθήκης *OpenCV*, μίας βιβλιοθήκης ελεύθερου κώδικα που περιέχει διάφορους αλγορίθμους επεξεργασίας εικόνας.

3.3.1 *OpenCV* ως *plug-in* στη *Unity*

Για τη χρήση της *OpenCV* στη *Unity* χρειάστηκε το *compilation* της βιβλιοθήκης για την πλατφόρμα *UWP* ως *plug-in*, όπως αναφέρθηκε στην ενότητα 1.6.4. Αυτό είναι αρκετά σύνθετη διαδικασία, διότι η βιβλιοθήκη έρχεται με αρκετές παραμέτρους για το *compilation*.

Στην περίπτωσή μας θέλαμε να δημιουργηθεί ένα *Visual Studio Project* για *C++/WinRT* [66], μία έκδοση της *C++* η οποία αναπτύσσεται και διατηρείται από τη *Microsoft* με σκοπό την καλή συνεργασία μεταξύ γλωσσών του *.NET* περιβάλλοντος και της *C++*. Μάλιστα, η δημιουργία ενός *.dll* αυτής της μορφής επιτρέπει στον προγραμματιστή να χρησιμοποιήσει κλάσεις και μεθόδους του *.dll* απευθείας σε άλλη γλώσσα του *.NET*, όπως για παράδειγμα τη *C#*, σαν να είχε γραφτεί η βιβλιοθήκη εξαρχής στη γλώσσα αυτή.

Δυστυχώς, η διαδικασία του *compilation* της *OpenCV* για *WinRT* δεν οδήγησε κάπου. Έγιναν αρκετές δοκιμές μέσω *CMake* [67], αλλά το τελικό παραγόμενο *project* δεν παρείχε τη δυνατότητα χρήσης σε περιβάλλον *WinRT*. Εν τέλη χρησιμοποιήθηκε μία παλαιότερη έκδοση της *OpenCV* από ένα *GitHub repository* [68] το οποίο λειτούργησε και ως πρότυπο για την ανάπτυξη της ανίχνευσης της εικόνας.

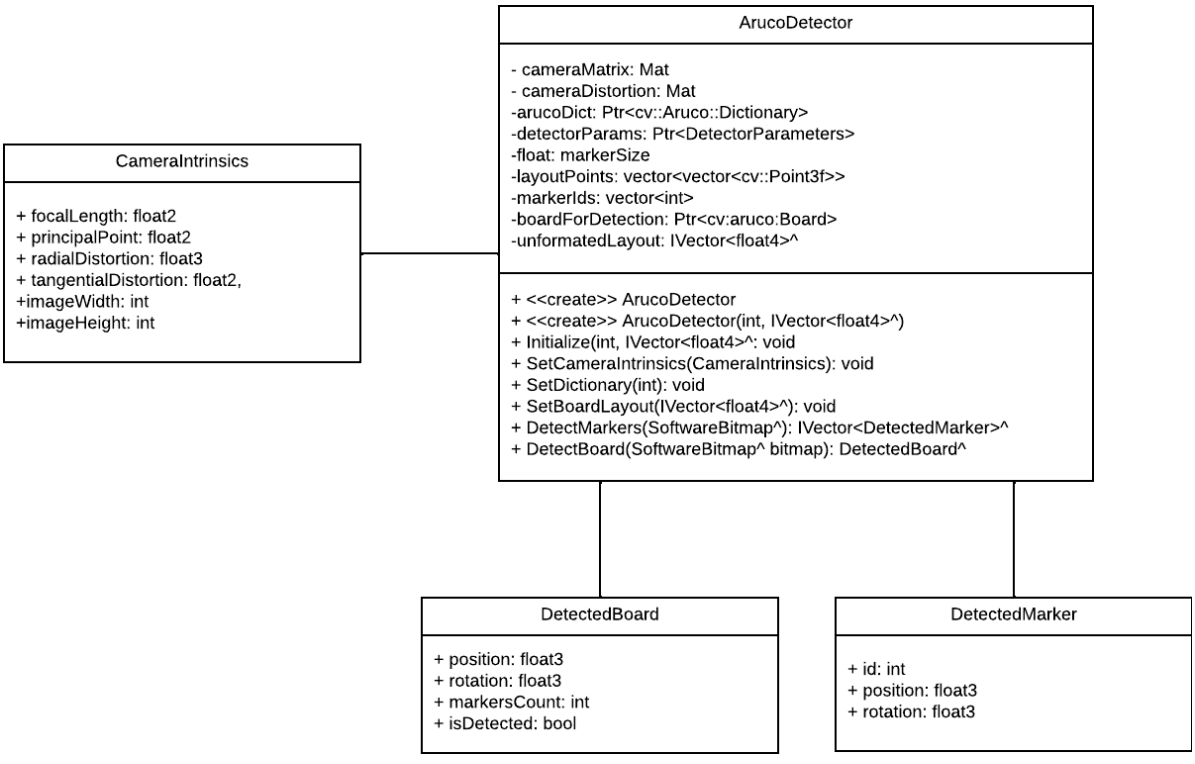
3.3.2 *ArUco Tracking σε OpenCV*

Για την τελική ανίχνευση της εικόνας χρησιμοποιήθηκε το *ArUco module* της *OpenCV*. Δημιουργήθηκε μια βιβλιοθήκη για *WinRT* η οποία υλοποιεί την ανίχνευση δεικτών είτε ξεχωριστά είτε ως ένα σύνολο (*board*). Για να λειτουργήσει σωστά η ανίχνευση πρέπει να περαστεί το *board* ως μια διάταξη πίνακα [69], ώστε:

- Μια λίστα λιστών, όπου η εσωτερική λίστα έχει τις θέσεις των γωνιών των προς ανίχνευση δεικτών στο τρισδιάστατο σύστημα αναφοράς. Ως μονάδα μέτρησης χρησιμοποιούνται τα μέτρα. Οι γωνιές αποθηκεύονται ξεκινώντας από την πάνω αριστερή και ακολουθώντας ωρολογιακή φορά.
- Το λεξικό των δεικτών για τη σωστή λειτουργία του αλγορίθμου.
- Μια λίστα με το αναγνωριστικό του κάθε δείκτη βάσει το εκάστοτε λεξικό, ένα προς ένα ώστε να αντιστοιχεί κάθε αναγνωριστικό στο αντίστοιχο δείκτη, η θέση του οποίου δίνεται από τη λίστα των θέσεων.
- Οι εγγενείς ιδιότητες της κάμερας, ήτοι η **εστιακή απόσταση (*focal length*)**, το **κύριο σημείο (*principal point*)**, **η ακτινική παραμόρφωση (*radial distortion*)** και η **εφαπτομενική παραμόρφωση (*tangential distortion*).**

Σύμφωνα με τα παραπάνω, προκύπτουν οι εξής κλάσεις σε *C++*, οι οποίες έγιναν *compile* σε *WinRT .dll* για χρήση εντός *Unity*:

- ***CVUtlities***, η οποία είναι βοηθητική κλάση που μετατρέπει ένα *bitmap* (την εικόνα) σε μορφή που να την αναγνωρίζει η *OpenCV*.
- ***CameraIntrinsics***, η οποία εμπεριέχει τις εγγενείς ιδιότητες της κάμερας.
- ***DetectedMarker***, η οποία υποδηλώνει έναν εντοπισμένο δείκτη στο χώρο
- ***DetectedBoard,*** η οποία αναπαριστά ένα εντοπισμένο σύνολο δεικτών στο χώρο και την πόζα του.
- ***ArucoDetector***, η οποία χρησιμοποιεί όλα τα παραπάνω για τον σωστό εντοπισμό των δεικτών ή ενός συνόλου δεικτών σε ένα πλαίσιο της κάμερας.



ΕΙΚΟΝΑ 49 ARUCO MODULE CLASS UML

3.3.3 Χρήση του *ArUco* στη *Unity*

Για να χρησιμοποιήσουμε τη βιβλιοθήκη *ArUco* η οποία αναπτύχθηκε παραπάνω χρειάζονται κάποια επιπλέον βήματα. Στη *Unity* ναι μεν μπορείς να γράψεις *scripts* σε *C#*, αλλά δεν έχεις πάντα όλες τις δυνατότητες τις κάθε πλατφόρμας. Πολλές φορές χρειάζεται η *Unity* να έχει προσφέρει δυνατότητες που αφορούν μόνο μία πλατφόρμα – π.χ. *Android* – είτε πρέπει ο ίδιος ο σχεδιαστής να γράψει μία εξωτερική βιβλιοθήκη σε μορφή *.dll* ή κάτι παρόμοιο ώστε να χρησιμοποιήσει επιπλέον δυνατότητες της κάθε πλατφόρμας.

Στην περίπτωση της διπλωματικής είναι αναγκαία η πρόσβαση σε δυνατότητες της πλατφόρμας *UWP*, ώστε να καθίσταται δυνατή η ανάγνωση πλαισίων της κάμερας καθώς και επεξεργασίας τους με τη βιβλιοθήκη *ArUco*. Ευτυχώς στη *Unity* μπορεί να χρησιμοποιηθεί άμεσα κώδικας *UWP*, αρκεί να τυλίξουμε τον κώδικα σε μία *#if-#endif* ντιρεκτίβα.

Έτσι αποκτούμε άμεση πρόσβαση στην κάμερα του *HoloLens*. Τα πλαίσια της κάμερας έρχονται στη μορφή *SoftwareBitmap*, την οποία και μπορεί να χρησιμοποιήσει η βιβλιοθήκη *ArUco*.

3.3.4 Συντεταγμένες από *OpenCV* σε *Unity*

Υπάρχουν κάποιες διαφορές όσον αφορά τα συστήματα συντεταγμένων της *OpenCV* και της *Unity*. Το σύστημα συντεταγμένων της *OpenCV* είναι δεξιού χεριού, ενώ της *Unity* είναι αριστερού χεριού. Αυτό σημαίνει πως για την αλλαγή μεταξύ των δύο συστημάτων αρκεί απλά να πολλαπλασιάσουμε την 3η σειρά ενός πίνακα μετασχηματισμού που δίνεται σε συντεταγμένης της *OpenCV* με -1 για να μετατρέψουμε τις συντεταγμένες σε σύστημα στη *Unity*.

Πριν όμως γίνει αυτή η μετατροπή, πρέπει να εξάγουμε τον πίνακα μετασχηματισμού από την πόζα που δίνει η *OpenCV*. Η *Unity* προσφέρει μια πολύ εύχρηστη συνάρτηση που παίρνει ως ορίσματα θέση, περιστροφή και μέγεθος και επιστρέφει έναν 4x4 πίνακα που περιγράφει το μετασχηματισμό αυτό. Συναρτησιακά αυτό προκύπτει – σύμφωνα με την υποενότητα 1.2.8 ως:

$$M_{aruco} = TRS(p, q, [1\ 1\ 1]^T)$$

, όπου p η θέση της πόζας, q η περιστροφή και $[1\ 1\ 1]^T$ η μοναδιαία κλίμακα μεγέθους.

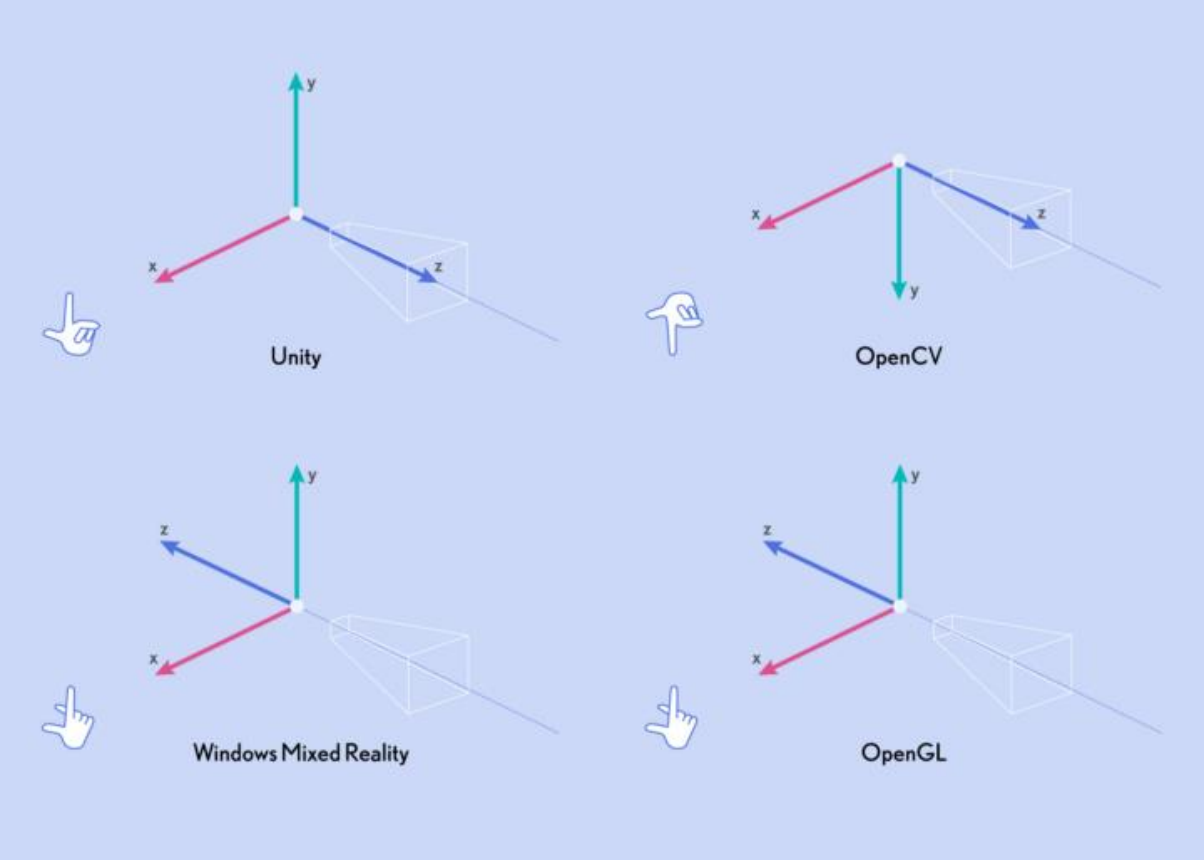
Το M_{aruco} περιγράφει την πόζα του δείκτη, αλλά ως προς σύστημα αναφοράς την κάμερα. Επειδή όμως σύστημα αναφοράς της εφαρμογής που τρέχει στο HoloLens είναι συνήθως η πόζα της συσκευής στην αρχή της εφαρμογής, είναι αναγκαίος ένας επιπλέον μετασχηματισμός ο οποίος θα μετατρέπει τις συντεταγμένες από το παγκόσμιο σύστημα αναφοράς της εφαρμογής στο σύστημα αναφοράς της κάμερας.

Για αυτό το σκοπό η πλατφόρμα *UWP* προσφέρει τη δυνατότητα εξαγωγής του πίνακα μετασχηματισμού της κάμερας, στον οποίο αναφερόμαστε ως M_{camera}. Ο τελικός πίνακας μετασχηματισμού που δίνει τη θέση, την περιστροφή και το μέγεθος του εντοπισμένου δείκτη ή συνόλου δεικτών δίνεται από:

$$M_{arucoCamera} = M_{camera} * M_{aruco}$$

Παρακάτω παρατίθενται τα συστήματα αναφοράς διαφόρων εφαρμογών και η γραφική απεικόνισή.

	Unity	**OpenCV**	**WMR**	**OpenGL**
χέρι αναφοράς	αριστερό	δεξί	δεξί	δεξί
κάμερα (δεξιά / πάνω / μπροστά)	(x, y, z)	(x, -y, z)	(x, y,-z)	(x, y,-z)
βιβλιογραφία	Unity Docs	OpenCV Docs	WMR Docs	OpenGL Boo

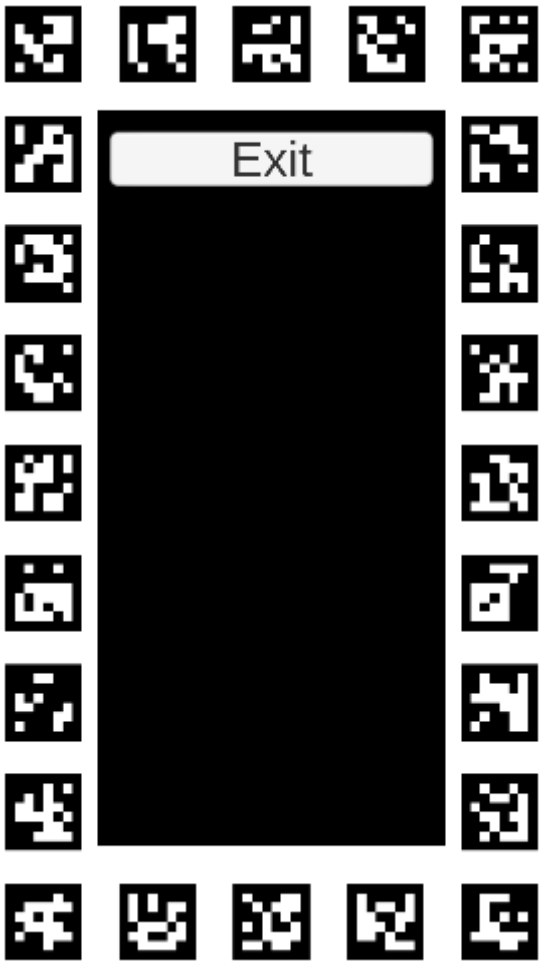


ΕΙΚΟΝΑ 50 ΠΕΔΙΟ ΤΗΣ ΚΑΜΕΡΑΣ ΚΑΙ ΣΥΣΤΗΜΑ ΣΥΝΤΕΤΑΓΜΕΝΩΝ

3.3.5 Διαμόρφωση του πίνακα δεικτών στην οθόνη του κινητού

Πριν την υλοποίηση του αλγορίθμου *6DOF*, απαιτείται η διαμόρφωση του πίνακα δεικτών τον οποίο θα εντοπίζει η ανίχνευση *ArUco*. Για το σκοπό αυτό χρησιμοποιήθηκε μια διαδικτυακή γεννήτρια δεικτών *ArUco*, από την οποία δημιουργήθηκαν εικόνες από τους διάφορους δείκτες. Το λεξικό που επιλέχθηκε ήταν 6x6 (DCT_6x6). Έπειτα, έγινε εισαγωγή των δεικτών ως *assets* μέσα στη *Unity*.

Οι δείκτες τοποθετήθηκαν στην οθόνη του κινητού όπως φαίνεται στην παρακάτω εικόνα, με χρήση του *Editor* της *Unity*.



ΕΙΚΟΝΑ 51 ARUCO BOARD LAYOUT

Όπως αναφέρθηκε παραπάνω, ο αλγόριθμος για την ανίχνευση του συνόλου των δεικτών (*layout*) απαιτεί τη θέση και το μέγεθος του κάθε σημείου στον φυσικό χώρο, με σύστημα αναφοράς το κέντρο του συνόλου.

Για την αυτοματοποίηση της παραπάνω διαδικασίας – ώστε να μη χρειάζεται ο σχεδιαστής – να τοποθετεί τις τιμές με το χέρι κάθε φορά στον *editor* για κάθε διαφορετικό *layout* χρησιμοποιήθηκε το ***dpi*** *(dots per inch)* [70], ένα μέγεθος που δείχνει πόσες κουκίδες (στην

περίπτωσή μας εικονοστοιχεία) χωράνε σε μια γραμμή μήκους μίας ίντσας. Η *Unity* δίνει τη δυνατότητα ανάγνωσης του *dpi* συσκευής [71]. Παρόλα αυτά, να αναφερθεί πως σε συσκευές *Android* υπάρχει η περίπτωση η τιμή να μην είναι σωστή, διότι εκεί επιστρέφεται η τιμή *densityDpi*, η οποία προέρχεται από μία έκταση διαφορετικών τιμών *dpi*. Δίνεται λοιπόν η δυνατότητα στον σχεδιαστή να βάλει ο ίδιος την τιμή *dpi* και με τη σειρά του να το προσφέρει ως δυνατότητα της εφαρμογής του κινητού, σε περίπτωση που πιστεύει πως χρειάζεται. Για το κινητό της διπλωματικής, το οποίο είναι συσκευή *Android*, η τιμή *dpi* που παρείχε η *Unity* ήταν σωστή και αναγράφεται στα 420.

Εφόσον υπάρχει η σωστή τιμή *dpi*, η εύρεση του φυσικού σημείου του δείκτη πάνω στο κινητό γίνεται ως εξής:

- Ένας δείκτης στο κινητό έχει πάνω το *component RectTransform*, το οποίο στη *Unity* χρησιμοποιείται για την τοποθέτηση *UI* στην οθόνη. Ουσιαστικά αποτελεί ένα τετράγωνο του οποίου γνωρίζουμε τη συντεταγμένη του κέντρου, το πλάτος και κατ' επέκταση των 4 γωνιών του στο σύστημα συντεταγμένων της *Unity*.
- Η *Unity* επιπλέον παρέχει τη δυνατότητα να μετατρέψεις τις συντεταγμένες με αρχή αναφοράς τον κόσμο σε συντεταγμένες με αρχή αναφοράς την οθόνη, όπου και σου επιστρέφει περίπου σε πιο εικονοιστοιχείο βρίσκεται η θέση αυτή βάσει της ανάλυσης της οθόνης σου. Έπειτα διαιρούμε με το *dpi* για να πάρουμε τη θέση των γωνιών του κάθε δείκτη και το μέγεθός του, με αρχή αναφοράς το κέντρο της οθόνης και μονάδα μέτρησης τις ίντσες.
- Τέλος, πολλαπλασιάζουμε με 0.0254 ώστε να μετατρέψουμε τις ίντσες σε μέτρα για να ικανοποιήσουμε τις προδιαγραφές της βιβλιοθήκης ArUco.

```
2 references
private Vector3 GetFromCenterInMeters(Vector3 worldPos, Camera camera)
{
    var inScreen = camera.WorldToScreenPoint(worldPos);
    var dpi = dpiFetcher.GetDpi();
    inScreen -= new Vector3(Screen.width / 2, Screen.height / 2, 0);
    inScreen.z = 0;
    inScreen = inScreen / dpi;
    inScreen *= 0.0254f;
    return inScreen;
}
```

ΕΙΚΟΝΑ 52 ΚΩΔΙΚΑΣ ΥΠΟΛΟΓΙΣΜΟΥ ΘΕΣΗΣ ΣΥΜΦΩΝΑ ΜΕ ΤΟ ΚΕΝΤΡΟ ΤΗΣ ΟΘΟΝΗΣ

```
    var topLeftInScreen = GetFromCenterInMeters(topLeft, camera);
    var topRightInScreen = GetFromCenterInMeters(topRight, camera);
    var width = (topLeftInScreen - topRightInScreen).magnitude;

    var item = new ArucoBoardLayoutItem
    {
        topLeftCorner = topLeftInScreen,
        size = width
    };
```

ΕΙΚΟΝΑ 53 ΚΩΔΙΚΑΣ ΥΠΟΛΟΓΙΣΜΟΥ ΤΗΣ ΑΡΙΣΤΕΡΗΣ ΠΑΝΩ ΓΩΝΙΑΣ ΚΑΙ ΤΟΥ ΜΕΓΕΘΟΥΣ ΕΝΟΣ ΔΕΙΚΤΗ

3.4 Υλοποίηση του αλγορίθμου 6DOF

Η τελική υλοποίηση του αλγορίθμου έγινε με χρήση της βιβλιοθήκης *ARCore* της *Google* σε μια συσκευή *Android*. Παρόλα αυτά, η υλοποίηση σε οποιοδήποτε κινητό γίνεται εύκολα και άμεσα, όπως εξηγήθηκε στην ενότητα 1.6.

3.4.1 Μετασχηματισμός τοπικής πόζας του κινητού στο σύστημα αναφοράς της κάμερας.

Εφόσον έχει υπολογιστεί η τοπική πόζα του κινητού από τον αλγόριθμο, το μόνο που μένει είναι ο μετασχηματισμός της πόζας αυτής στο σύστημα αναφοράς της κάμερας. Ορίζουμε τα εξής μεγέθη:

p_{Vio}: $Τοπική\ θέση\ κινητού\ τη\ στιγμή\ ανίχνευσης\ του\ layout$
r_{Vio}: $Τοπικός\ προσανατολισμός\ (τετραδόνιο)\ του\ κινητού\ τη\ στιγμή\ ανίχνευσης\ του\ layout$
p_{board}: $Θέση\ του\ κινητού\ από\ την\ ανίχνευση\ του\ layout$
r_{board}: $Προσανατολισμός\ (τετραδόνιο)\ του\ κινητού\ από\ την\ ανίχνευση\ του\ layout$

Θέλουμε να βρούμε δύο εξισώσεις που να μετασχηματίζουν την πόζα. Σύμφωνα με τα παραπάνω μεγέθη, μπορούμε να πούμε πως:

$$r_{hololens} = R * r_{phone} => R = r_{hololens} * r_{phone}^{-1}$$
$$p_{hololens} = T + R * p_{phone} => T = p_{hololens} - R * p_{phone}$$

όπου R, T περιστροφή και θέση μετασχηματισμού αντίστοιχα. Για τον υπολογισμό τους αρκεί να αντικαταστήσουμε στις παραπάνω εξισώσεις τα μεγέθη ανίχνευσης που ορίσαμε παραπάνω.

3.4.2 Κλάσεις που αφορούν το κινητό

Οι κλάσεις που αφορούν το κινητό είναι οι εξής:

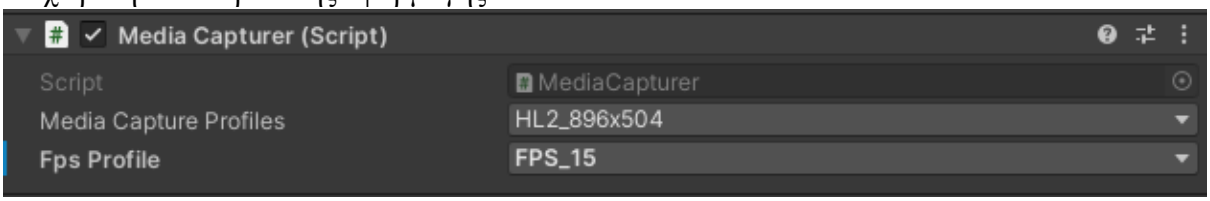
- ***MarkersLayout***, η οποία επεκτείνει τη *MonoBehaviour* της Unity. Υλοποιεί τον υπολογισμό της θέσης και του μεγέθους των δεικτών με σημείο αναφοράς την οθόνη του κινητού, όπως περιγράφηκε στην προηγούμενη ενότητα. Τα δεδομένα στέλνονται μέσω του συστήματος δικτύωσης στο HoloLens.

3.4.3 Κλάσεις που αφορούν το *HoloLens*

Οι κλάσεις που αφορούν το *HoloLens* είναι οι εξής:

- ***MediaCapturer***, η οποία επεκτείνει τη *MonoBehaviour* της Unity. Η κλάση αυτή χρησιμοποιεί κώδικα γραμμένο σε *UWP* για τον εντοπισμό των πλαισίων της κάμερας, καθώς και των εγγενών ιδιοτήτων της. Έπειτα παρέχει τα πλαίσια αυτά σε οποιαδήποτε άλλη κλάση για επεξεργασία. Οποιαδήποτε στιγμή παρέχεται η δυνατότητα να ξεκινήσει ή να σταματήσει η παροχή πλαισίων, για καλύτερη

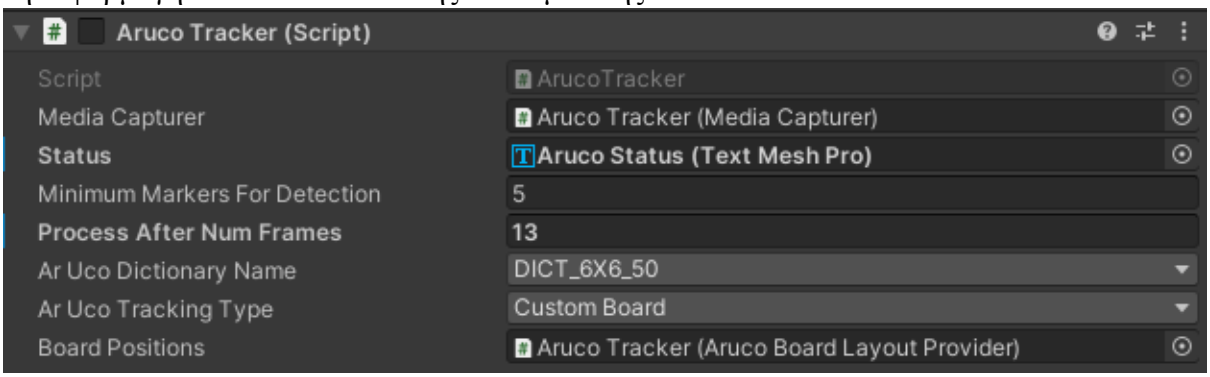
διαχείριση των πόρων της εφαρμογής.



ΕΙΚΟΝΑ 54 ΤΟ COMPONENT MEDIA CAPTURER

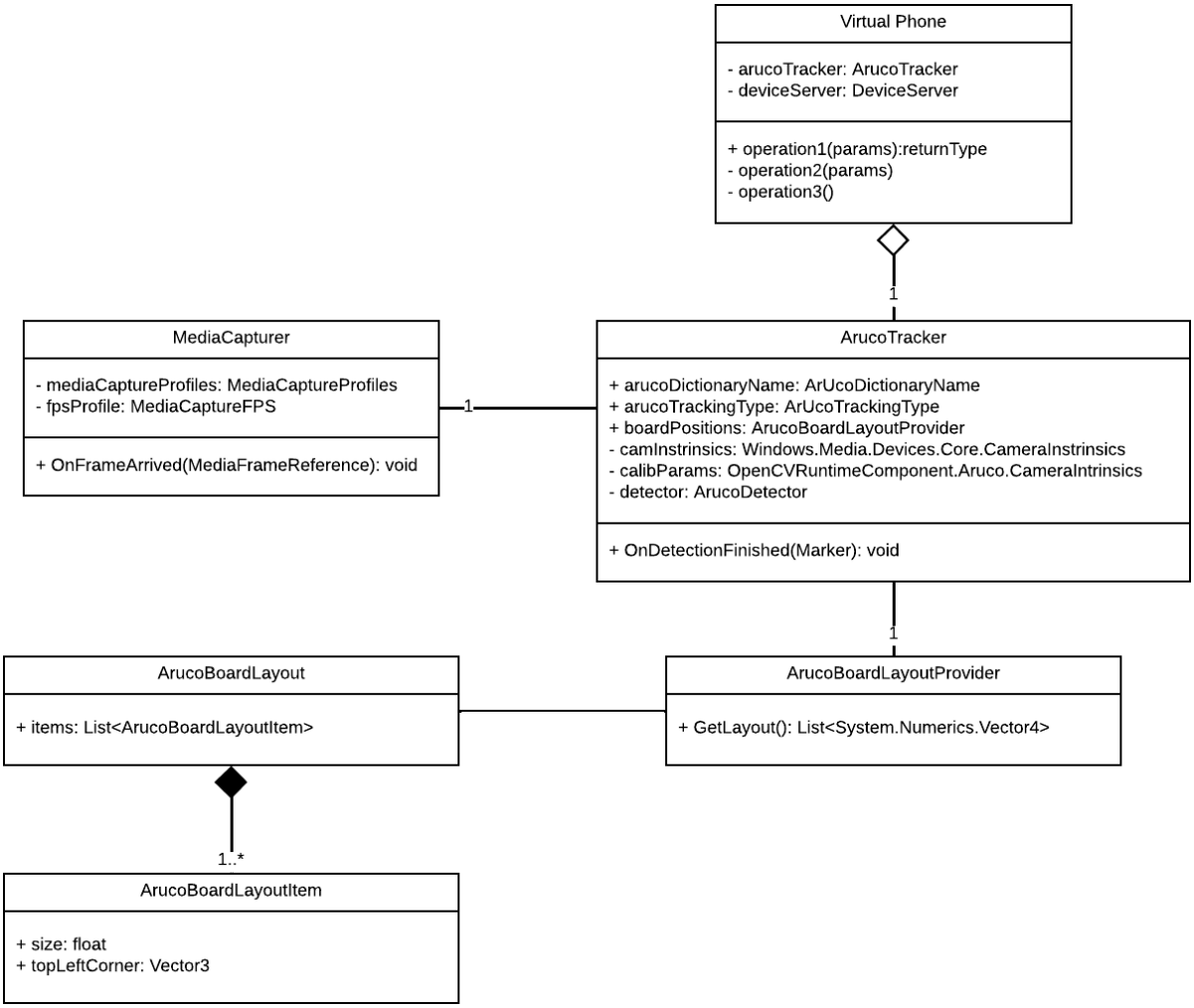
Στην παραπάνω εικόνα φαίνονται οι ρυθμίσεις που δέχεται αυτή η κλάση. Υπάρχει η δυνατότητα επιλογής διαφορετικής ανάλυσης καθώς και διαφορετικού προφίλ *fps*, με επιλογές 15 και 30.

- ***ArucoBoardLayout***. Η κλάση αυτή χρησιμοποιείται για να περαστεί το *layout* των δεικτών στη βιβλιοθήκη *ArUco*.
- ***ArucoBoardLayoutProvider***, η οποία επεκτείνει τη *MonoBehaviour* της Unity. Η κλάση αυτή δέχεται το *layout* από το κινητό κατά τη σύνδεσή του με το *HoloLens*. Έπειτα το μετατρέπει στη μορφή *ArucoBoardLayout*.
- ***ArucoTracker***, η οποία επεκτείνει τη *MonoBehaviour* της Unity. Η κλάση αυτή αξιοποιεί την *MediaCapturer*, απ' όπου και παίρνει τα στιγμιότυπα της κάμερας. Κάνει αρχικοποίηση της βιβλιοθήκης *ArUco* με το επιλεγμένο από το σχεδιαστή λεξικό και το *layout* που δόθηκε από το κινητό. Έπειτα, εκτελεί την ανίχνευση *ArUco* μετά από κάθε n στιγμιότυπα εικόνας (δηλαδή αν είχαμε 30 στιγμιότυπα και n=15, θα είχαμε ρυθμό ανίχνευσης 2). Η επιλογή αυτή δίνεται διότι η ανίχνευση είναι αρκετά βαριά διαδικασία και αφήνεται στην κρίση του σχεδιαστή το κατά πόσο επηρεάζει την εφαρμογή του. Στα πλαίσια της διπλωματικής το n = 5-15.



ΕΙΚΟΝΑ 55 ΤΟ COMPONENT ARUCOTRACKER ΚΑΙ ΟΙ ΡΥΘΜΙΣΕΙΣ ΤΟΥ

- ***VirtualPhone***, η οποία επεκτείνει τη *MonoBehaviour* της *Unity*. Σκοπός της είναι η τοποθέτηση του εικονικού κινητού πάνω στο φυσικό κινητό και η εκτέλεση του αλγορίθμου *6DOF*.

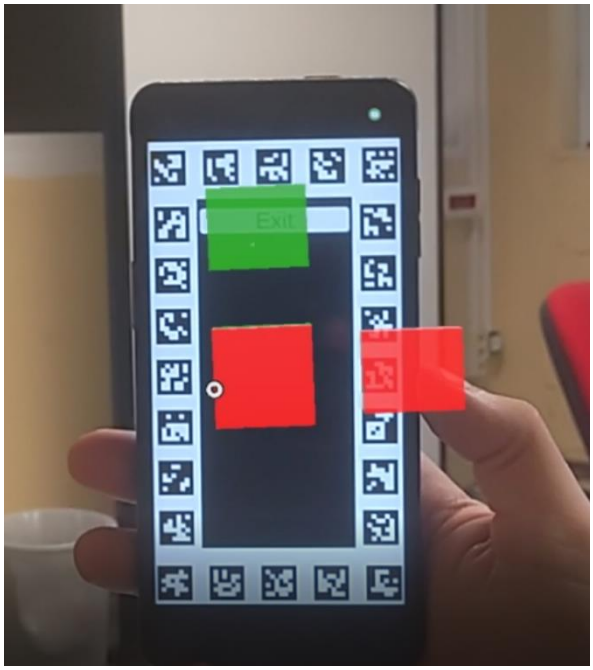


ΕΙΚΟΝΑ 56 UML ΔΙΑΓΡΑΜΜΑ ΤΩΝ COMPONENT ΤΟΥ ΑΛΓΟΡΙΘΜΟΥ ΜΕΣΑ ΣΤΗ UNITY

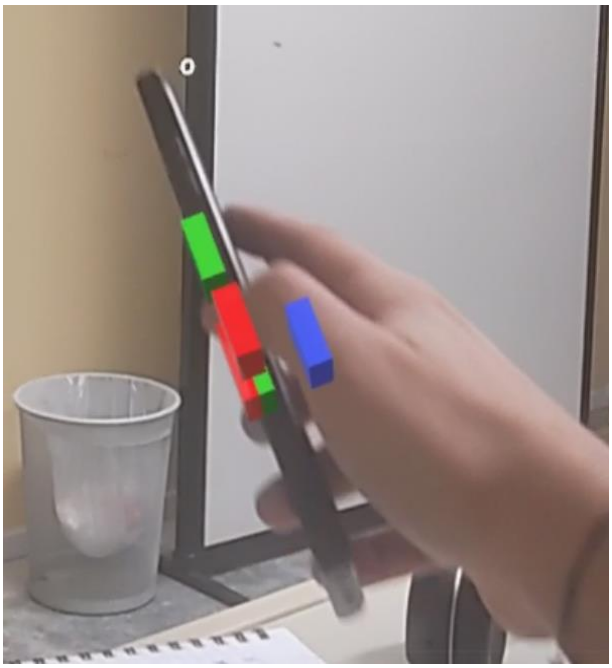
3.4.4 Αποτελέσματα

Ο αλγόριθμος *6DOF* λειτουργεί, αλλά το σύστημα δεν είναι αρκετά εύρωστο ώστε να θεωρηθεί χρήσιμο. Πολλές φορές η οπτική οδομετρία του κινητού από τη βιβλιοθήκη *ARCore* δεν είναι επαρκης · Η αρχή αναφοράς του κινητού αλλάζει ή χάνεται ο εντοπισμός της πόζας του κινητού. Βελτίωση αυτού προβλήματος μπορεί να γίνει με καλύτερη υλοποίηση της οπτικής οδομετρίας καθώς και καλύτερη βαθμονόμηση της συσκευής. Δυστυχώς, για συσκευές *Android*, η βαθμονόμηση είναι πιο γενική επειδή το λειτουργικό αυτό καθώς και οι συσκευές παράγονται μαζικά από διαφορετικές εταιρίες και εργοστάσια. Έχει παρατηρηθεί πως η υλοποίηση του *ARKit* είναι πιο εύρωστη και αποτελεσματική.

Τα αποτελέσματα φαίνονται στις παρακάτω εικόνες.



ΕΙΚΟΝΑ 58 ΕΝΤΟΠΙΣΜΟΣ ΤΟΥ BOARD



ΕΙΚΟΝΑ 57 TRACKING

Τέλος, όπως αναφέρθηκε και προηγουμένως, υπάρχει καθυστέρηση λόγω του δικτύου. Η ανίχνευση της πόζας του κινητού πιθανότατα θα βελτιωνόταν αν αλλάζαμε το γενικό πρωτόκολλο επικοινωνίας από *Wi-fi* σε *Bluetooth* ή *Wi-fi Direct*, για μείωση της καθυστέρησης (*latency*).

Κεφάλαιο 4 – Αξιολόγηση αλληλεπίδρασης που προσφέρει ο εξωτερικός ελεγκτής

4.1 Σκοπός

Στο κεφάλαιο παρουσιάζεται η σχεδίαση και η υλοποίηση ενός πειράματος με σκοπό την αξιολόγηση μιας αλληλεπίδρασης που προσφέρει ο ελεγκτής – κινητό σε σύγκριση με την αντίστοιχη εγγενή προσέγγιση του HoloLens. (Σημείωση πως όταν χρησιμοποιείται ο όρος χρήστης αναφερόμαστε είτε στο υποκείμενο είτε στο σχεδιαστή.)

4.2 Το πείραμα [72]

4.2.1 Περιγραφή του πειράματος

Το πείραμα χωρίζεται σε 2 σκέλη. Το υποκείμενο τοποθετείται σε ένα συγκεκριμένο σημείο στο χώρο. Έπειτα καλείται να επιλέξει μια σειρά από τυχαίους απομακρυσμένους στόχους. Οι στόχοι αυτοί βρίσκονται σε συγκεκριμένα σημεία στο χώρο, σε ένα σχετικά μικρό δωμάτιο περίπου των 5 τετραγωνικών μέτρων. Η διαδικασία αυτή διεξάγεται χρησιμοποιώντας 2 διαφορετικούς τρόπους αλληλεπίδρασης, οι οποίοι είναι:

- Ο εγγενής τρόπος που προσφέρει το HoloLens. Ως εγγενής τρόπος αλληλεπίδρασης επιλέχθηκε ο συνδυασμός του δείκτη κεφαλής ώστε το υποκείμενο να μπορεί να διαλέξει το στόχο και έπειτα το *air-tap* ώστε να επικυρώσει την επιλογή του.
- Ο εξωτερικός τρόπος αλληλεπίδρασης που προσφέρει το πλαίσιο που αναπτύχθηκε στη διπλωματική. Στην περίπτωση αυτή το υποκείμενο πάλι διαλέγει το στόχο με τη χρήση του δείκτη κεφαλής και έπειτα επικυρώνει την επιλογή του πατώντας στην οθόνη του κινητού του, όπως θα έκανε και με ένα ποντίκι.

Συγκεκριμένα, οι 2 τρόποι αλληλεπίδρασης αξιολογούνται σε 2 παρόμοιες δραστηριότητες, οι οποίες είναι:

- **Απόκριση στόχων:** Το υποκείμενο καλείται να επιλέξει όσους περισσότερους στόχους μπορεί σε **45** δευτερόλεπτα. Δε μας ενδιαφέρει ο ενδιάμεσος χρόνος μεταξύ των στόχων ούτε πόσο γρήγορη είναι η απόκριση που αφορά την επικύρωση.
- **Απόκριση χρόνου:** Το υποκείμενο καλείται να επιλέξει 20 στόχους, σε όσο χρόνο θέλει. Σημαντικά μεγέθη που μετρούνται κατά τη διάρκεια αυτής της δραστηριότητας είναι **ο χρόνος που χρειάστηκε ώστε να κοιτάξει τον επόμενο στόχο** και **ο χρόνος που χρειάστηκε για να επικυρώσει την επιλογή του**.

4.2.2 Προδιαγραφές απαιτήσεων της εφαρμογής του πειράματος

Για τη σωστή υλοποίηση του πειράματος καταγράφηκαν οι εξής απαιτήσεις της εφαρμογής.

- Τοποθέτηση από τον σχεδιαστή στο χώρο των στόχων και αποθήκευσής τους ώστε να μη χρειάζεται η εκ νέου αρχικοποίηση της εφαρμογής του πειράματος.
- Εύκολος εντοπισμός του επόμενου στόχου καθώς και οπτική ένδειξη πως ο δείκτης κεφαλή δείχνει πάνω στο στόχο.
- Πειραματισμός του υποκειμένου στο περιβάλλον του πειράματος πριν ξεκινήσει το πείραμα. Σκοπός αυτής της απαίτησης είναι να αποκτήσει μία εξοικείωση με το περιβάλλον και κυρίως με τον εγγενή τρόπο αλληλεπίδρασης, το *air-tap*. Αυτή η απαίτηση προέκυψε διότι οι περισσότεροι άνθρωποι δεν έχουν κάποια εξοικείωση με συσκευές μικτής πραγματικότητας και εν γένει τα αποτελέσματα του πειράματος δε θα ήταν αξιόπιστα.
- Εισαγωγή των στοιχείων του υποκειμένου· Ενός ψευδωνύμου και της ηλικίας του.
- Εύκολης διεξαγωγή των 2 διαφορετικών δραστηριοτήτων με χρήση κατάλληλης γραφικής διεπαφής χρήστη (*user interface*)
- Ακύρωση του εν ενεργεία πειράματος σε περίπτωση που η διαδικασία δε διεκπεραιωνόταν σωστά.
- Αυτόματη αποθήκευση από την εφαρμογή των αποτελεσμάτων των δραστηριοτήτων.

4.2.3 Σχεδιαστικές προδιαγραφές της εφαρμογής του πειράματος

Εν συνεχεία καταγράφηκαν οι εξής σχεδιαστικές προδιαγραφές οι οποίες αναλύουν περαιτέρω την υλοποίηση των απαιτήσεων:

1) Ο χρήστης βλέπει ένα μήνυμα καλωσορίσματος το οποίο θα τον παραπέμπει για το βασικό τρόπο αλληλεπίδρασης με τη διεπαφή της εφαρμογής.
2) Ο βασικός τρόπος αλληλεπίδρασης με τη διεπαφή αποτελείται από 2 διαφορετικές ενέργειες. Ο χρήστης συναντά διάφορα εικονικά κουμπιά τα οποία μπορεί να πατήσει με το δείκτη του. Επιπλέον, τα μενού της εφαρμογής εμφανίζονται στο χρήστη αν φέρει την αριστερή του παλάμη ανοιχτή ώστε να φαίνεται στο πλαίσιο της κάμερας, απ' όπου και μπορεί να αλληλεπιδράσει με τις διάφορες επιλογές που του δίνει η διεπαφή. Ο σχεδιασμός αυτός προέκυψε από την ανάγκη γρήγορης διεκπεραίωσης του πειράματος, ώστε να αποφευχθεί η σπατάλη χρόνου στην περιήγηση της διεπαφής.
3) Το γενικό μενού θα περιέχει τις εξής επιλογές:
    - ***Demo***, όπου ο χρήστης περιηγείται στο περιβάλλον του πειράματος.
    - ***Air-tap***, όπου ο χρήστης εκτελεί τις 2 δραστηριότητες με χρήση του εγγενούς τρόπου αλληλεπίδρασης.
    - ***Phone-tap***, όπου ο χρήστης εκτελεί τις 2 δραστηριότητες με χρήση του εξωτερικού τρόπου αλληλεπίδρασης.
    - ***Ρυθμίσεις***, όπου ο χρήστης μπορεί να εισάγει ψευδώνυμο και ηλικία ή να τοποθετήσει τους στόχους στο χώρο.

Αναφέρεται πως υπάρχει και μία επιπλέον επιλογή, η επιλογή ***Other Demos***. Δεν αποτελεί μέρος του πειράματος αλλά υπάρχει για την ανάπτυξη και πειραματισμό άλλων τρόπων αλληλεπίδρασης κατά τη διάρκεια της διπλωματικής.

4) Ο χρήστης – υποκείμενο καθοδηγείται κατά τη διαδικασία των δραστηριοτήτων από γραφικές διεπαφές οι οποίες επισημαίνουν τα καίρια σημεία της δραστηριότητας.
5) Οι στόχοι είναι τρισδιάστατοι κύβοι. Το υποκείμενο καθοδηγείται στον επόμενο στόχο μέσω ενός βέλους στον τρισδιάστατο χώρο. Ο επόμενος στόχος φέρει διαφορετικό χρώμα από τους υπόλοιπους.
6) Μόλις το υποκείμενο τοποθετήσει τον δείκτη πάνω στο στόχο εμφανίζεται ένα γραφικό το οποίο υποδηλώνει πως ο στόχος μπορεί να επιλεχθεί με κάποια από τις 2 αλληλεπιδράσεις.
7) Η επιλογή του στόχου παίζει ένα μικρό κλιπ ήχου καθώς και ένα *animation* πάνω στο στόχο ώστε να γίνεται εύκολα αντιληπτή η επιλογή του.
8) Όταν τελειώσει μια δραστηριότητα παίζεται ένα μικρό κομμάτι μουσικής ώστε να περαστεί στο υποκείμενο η αίσθηση της εκπλήρωσης κάποιου σκοπού.

4.2.4 Υλοποίηση της εφαρμογής του πειράματος

Από την υλοποίηση της εφαρμογής προέκυψαν οι εξής κλάσεις:

- `TestPhaseHelper`, η οποία επεκτείνει τη `MonoBehaviour` της *Unity*. Χρησιμοποιείται για την περιγραφή μίας φάσης των δραστηριοτήτων και παρέχει τη δυνατότητα απεικόνισης μιας διεπαφής πληροφοριών στο χρήστη και αντίστροφης μέτρησης πριν ξεκινήσει η επόμενη φάση. Οι φάσεις μιας δραστηριότητας είναι οι εξής:
  1) ***Start Phase***, κατά την οποία εμφανίζεται ένα εισαγωγικό μήνυμα που περιγράφει τη δραστηριότητα.
  2) ***Target Response Phase,*** κατά την οποία διεξάγεται η δραστηριότητα «Απόκριση Στόχων».
  3) ***Middle Phase***, στην περίπτωση που χρειαστεί κάποιο ενδιάμεσο μήνυμα μεταξύ των δραστηριοτήτων.
  4) ***Time Response Phase***, κατά την οποία διεξάγεται η δραστηριότητα «Απόκριση Χρόνου»
  5) ***End Phase***, όπου εμφανίζεται ένα μήνυμα τέλους της δραστηριότητας.
- `BaseTester`, αφαιρετική κλάση η οποία επεκτείνει τη `MonoBehaviour` της *Unity*. Συγκεντρώνει τις διαφορετικές φάσεις μιας δραστηριότητας και εκτελεί το πείραμα. Οι υλοποιήσεις που αφορούν το πείραμα είναι η `DemoTester`, η οποία τρέχει την προσομοίωση του πειράματος και η `TimeAndTargetTester`,η οποία παρέχει ρυθμίσεις για τη διεκπεραίωση και των 2 δραστηριοτήτων.
- `EvaluationResults`, η οποία μοντελοποιεί τα αποτελέσματα μιας δραστηριότητας.
- `EvaluationTest`, η οποία μοντελοποιεί τα αποτελέσματα μιας των πειραμάτων.
- `EvaluationPlayer`, η οποία μοντελοποιεί τα αποτελέσματα ενός υποκειμένου για τα πειράματα.

- `SimpleFirebaseClient`, η οποία επεκτείνει τη *MonoBehaviour* της *Unity*. Χρησιμοποιείται για την επικοινωνία με τη βάση δεδομένων.

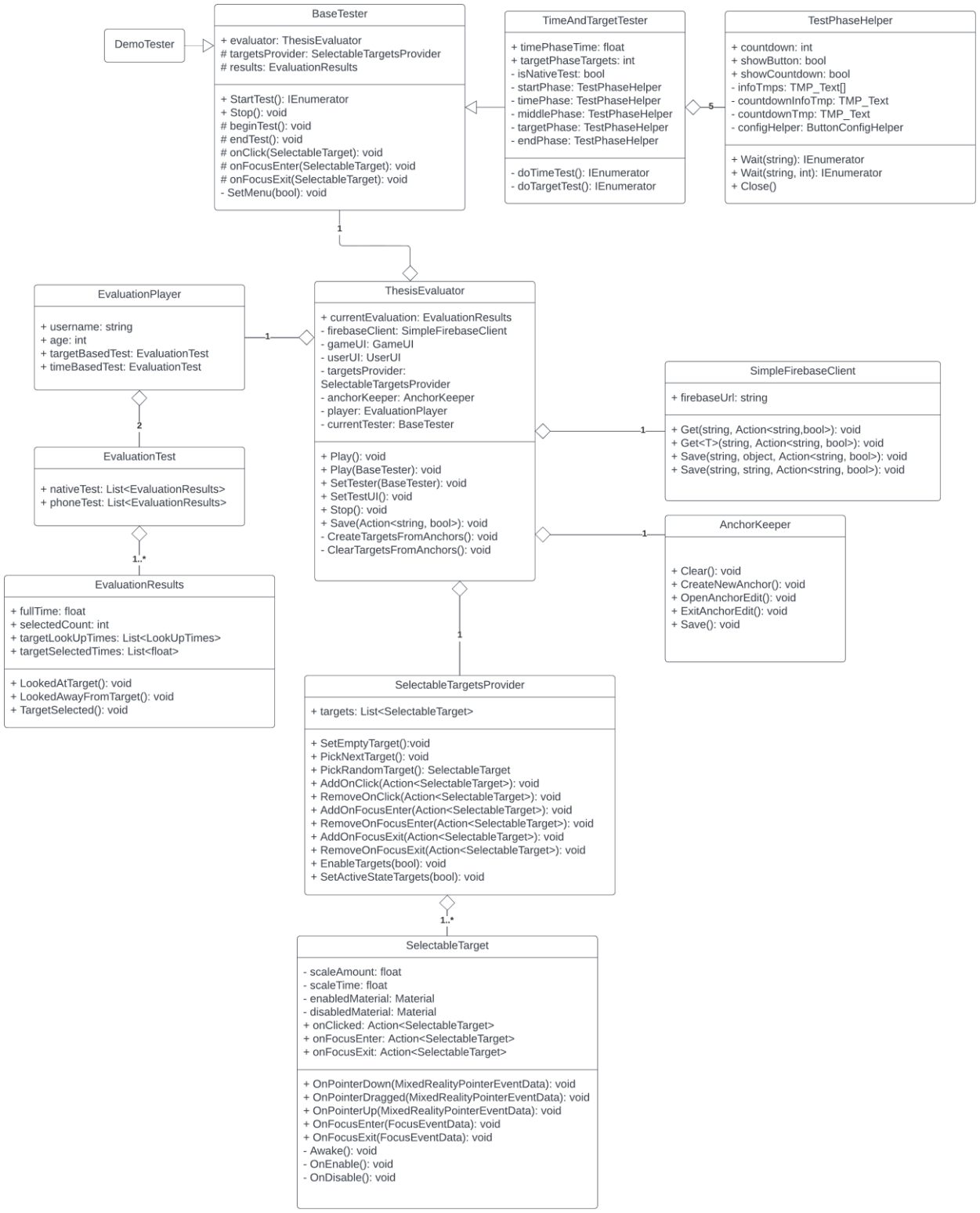
  Η βάση δεδομένων που χρησιμοποιήθηκε είναι η *Firebase Realtime Database* από την *Google* [73]. Αποτελεί μία μη σειριακή βάση δεδομένων (*NoSQL*) η οποία αποθηκεύει τα δεδομένα σε μορφή *JSON*. Προσφέρει δυνατότητα επικοινωνίας και μέσω μιας βιβλιοθήκης αλλά και μέσω απλών αιτημάτων *HTTP*.

4. `SelectableTarget`, η οποία επεκτείνει τη `MonoBehaviour` της *Unity*. Αυτή η κλάση είναι *component* το οποίο τοποθετείται πάνω στο *gameobject* ενός στόχου και αντιδρά αναλόγως όταν ο στόχος δειχθεί και επιλεχθεί.
5. `SelectableTargetsProvider`, η οποία επεκτείνει τη `MonoBehaviour` της *Unity*. Χρησιμοποιείται για την παροχή των στόχων στις δραστηριότητες της εφαρμογής.
6. `AnchorKeeper`, η οποία επεκτείνει τη `MonoBehaviour` της *Unity*. Ρόλος της είναι η αποθήκευση και ανάκτηση των στόχων που έχει τοποθετήσει ο σχεδιαστής.

   Αναφέρεται πως ο τρόπος αποθήκευσης που προσφέρει η *Unity* σημείων στο χώρο ώστε να παραμείνουν παρά την επανεκκίνηση της εφαρμογής δε δούλευε. Το μήνυμα σφάλματος δεν ήταν επαρκές για να λυθεί το πρόβλημα, οπότε σημειώθηκαν οι στόχοι σε ένα χαρτί και τοποθετούνταν εκ νέου όποτε χρειαζόταν.
7. `GameUI`, η οποία επεκτείνει τη `MonoBehaviour` της *Unity*. Η γραφική διεπαφή που βλέπει το υποκείμενο την ώρα του πειράματος.
8. `UserUI`, η οποία επεκτείνει τη `MonoBehaviour` της *Unity*. Η γραφική διεπαφή που βλέπει ο χρήστης πριν μπει στο πείραμα.
9. `ClickEvent`, η οποία μοντελοποιεί ένα κλικ που προέρχεται από το κινητό και λαμβάνεται από το HoloLens. Σχεδιάστηκε με βάσει το πλαίσιο που αναπτύχθηκε παραπάνω.
10. `ClickEventSender`, της οποίας αρμοδιότητα είναι η αποστολή του κλικ από το κινητό.
11. `ClickEventReceiver`, της οποίας αρμοδιότητα είναι η λήψη του κλικ στο HoloLens.
12. `ThesisEvaluator`, η οποία επεκτείνει τη `MonoBehaviour` της *Unity*. Αρμοδιότητά της είναι να συντονίζει τα υπόλοιπα *components* για να διεκπεραιωθεί το πείραμα.

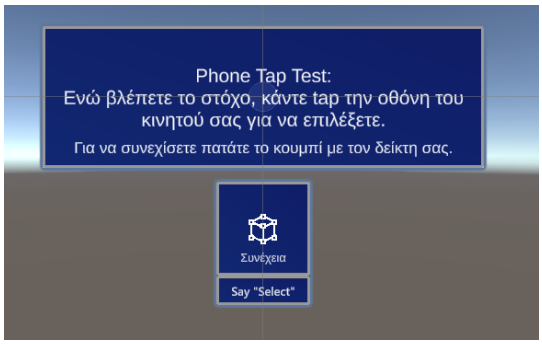
Σημειώνεται πως η υλοποίηση του κατευθυντικού βέλους έγινε με χρήση της βιβλιοθήκης *MRTK*, με χρήση της κλάσης `Solver`. [74]

Παρακάτω παρουσιάζονται το διάγραμμα *UML* το οποίο δείχνει τις σχέσεις μεταξύ των διάφορων *components* καθώς και διάφορες εικόνες από την εφαρμογή του πειράματος.



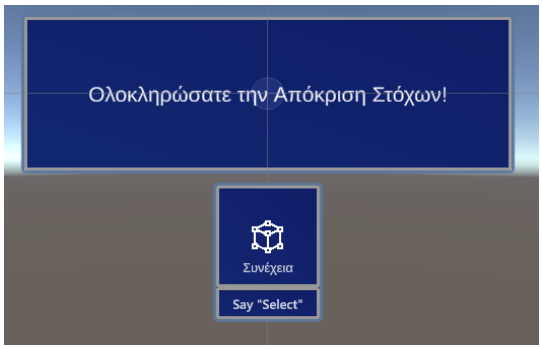
ΕΙΚΟΝΑ 59 UML ΤΩΝ COMPONENTS ΤΟΥ ΠΕΙΡΑΜΑΤΟΣ

ΕΙΚΟΝΑ 60 ΕΙΣΑΓΩΓΙΚΟ ΚΑΙ ΚΥΡΙΟ ΜΕΝΟΥ

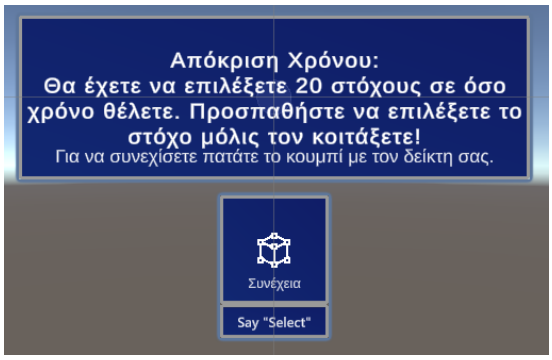


ΕΙΚΟΝΑ 61 START PHASE

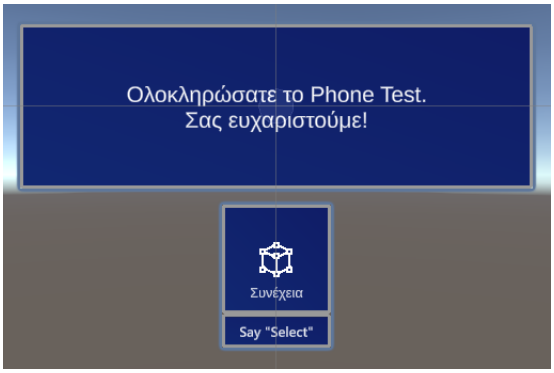
ΕΙΚΟΝΑ 62 TARGET RESPONSE PHASE



ΕΙΚΟΝΑ 63 MIDDLE PHASE

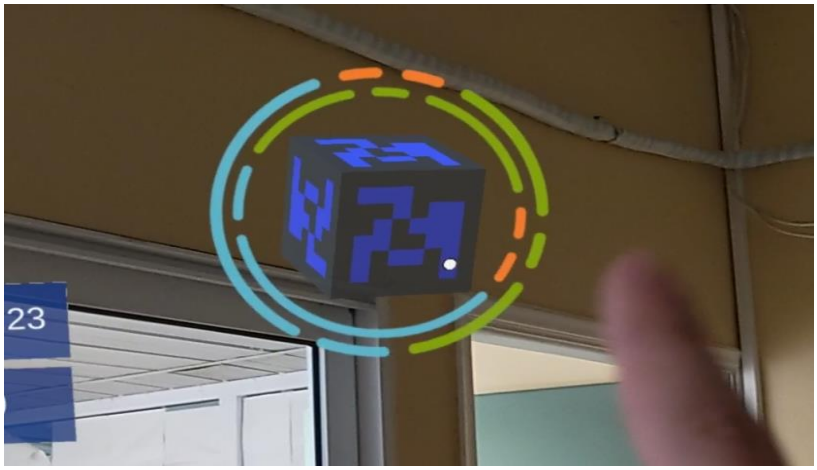


ΕΙΚΟΝΑ 64 TIME RESPONSE PHASE



ΕΙΚΟΝΑ 65 END PHASE

ΕΙΚΟΝΑ 66 ΕΝΤΟΠΙΣΜΟΣ ΣΤΟΧΟΥ



ΕΙΚΟΝΑ 67 ΕΠΙΛΟΓΗ ΣΤΟΧΟΥ

4.3 Διεξαγωγή του πειράματος

Το πείραμα έγινε στο Πανεπιστήμιο Πατρών, στο τμήμα Ηλεκτρολόγων Μηχανικών, τις περιόδους Ιουνίου και αρχές Σεπτεμβρίου 2022. Τα υποκείμενα ήταν φίλοι ή προσεγγίστηκαν μέσα από το τμήμα. Συνολικός αριθμός των συμμετεχόντων ήταν 25.

Ο συμμετέχοντας ερχόταν στο δωμάτιο διεξαγωγής του πειράματος. Του παρουσιαζόταν μια συνοπτική περιγραφή της επαυξημένης πραγματικότητας και του HoloLens. Έπειτα, γινόταν εξήγηση του πειράματος καθώς και των 2 δραστηριοτήτων. Ο συμμετέχοντας φοράει το HoloLens και ξεκινάει την εφαρμογή. Εισάγει ψευδώνυμο και ηλικία και στη συνέχεια του ζητείται να περιφερθεί λίγο στο περιβάλλον προσομοίωσης του πειράματος. Μετά από λίγη ώρα επιλέγεται ένας από τους 2 τρόπους αλληλεπίδρασης (ώστε πάντα ο επόμενος συμμετέχοντας να ξεκινάει με διαφορετικό τρόπο) και διεξάγεται το πείραμα.

Με το πέρας του πειράματος ο συμμετέχοντας περιηγείται σε 2 άλλα *demos*, ένα που χρησιμοποιεί ένα εικονικό χειριστήριο στο κινητό για να μετακινήσει έναν εικονικό κύβο και ένα που παρουσιάζει τον *6DOF Controller*.

Τέλος, γίνονται κάποιες ερωτήσεις που αφορούν την εμπειρία του από τη διαδικασία.

4.4 Απόκριση χρόνου

Σε αυτήν την ενότητα παρουσιάζονται τα αποτελέσματα της ανάλυσης της δραστηριότητας Απόκριση Χρόνου.

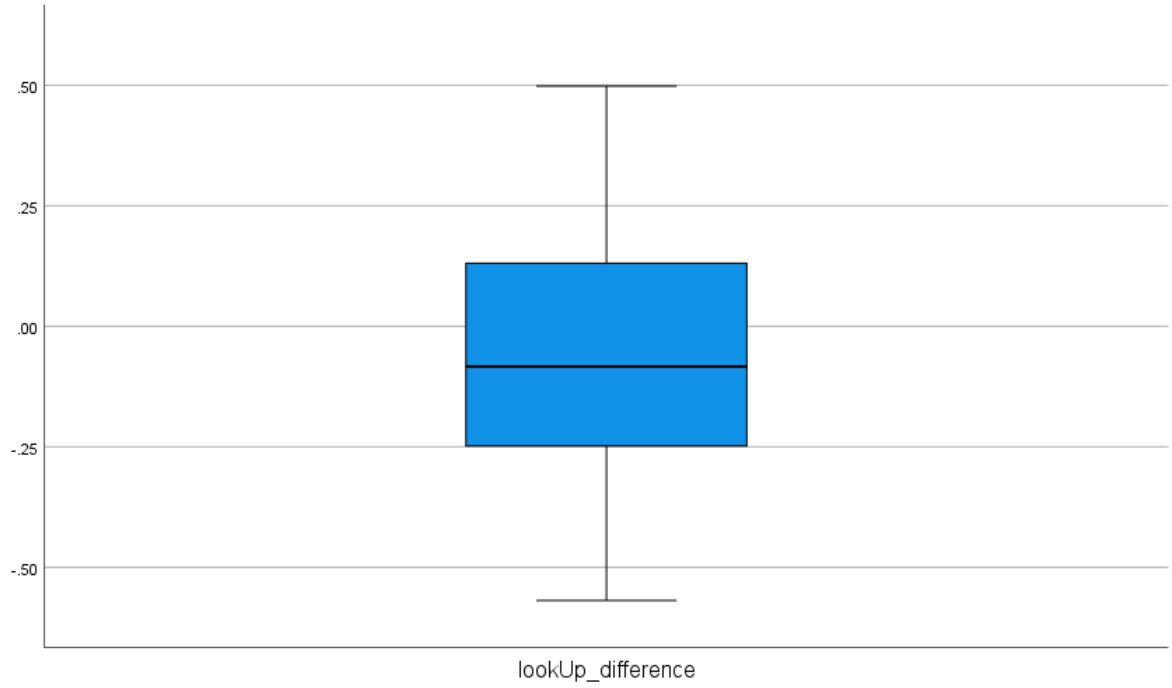
4.4.1 Χρόνος εντοπισμού του επόμενου στόχου

Ένα από τα μεγέθη που μας ενδιαφέρουν στις 2 διαφορετικές περιπτώσεις (*air-tap*, *κινητό*) όσον αφορά την Απόκριση Χρόνου είναι εάν επηρεάζεται η ικανότητα του χρήστη να κοιτάξει τον επόμενο στόχο ανάλογο τον τρόπο αλληλεπίδρασης που χρησιμοποιεί. Για τη στατιστική ανάλυση επιλέχθηκε να γίνει το ***paired-samples-t-test***, το οποίο εντοπίζει αν υπάρχει διαφορά στις μέσες τιμές δύο εξαρτημένων ομάδων σε μια εξαρτημένη συνεχή μεταβλητή.

Ορίζουμε λοιπόν $lookUp_{Difference} = lookUp_{phone} - lookUp_{native}$

Για να συνεχίσουμε, πρέπει να βεβαιωθούμε πως για τη διαφορά:

- Δεν υπάρχουν ακραίες τιμές
- Είναι κανονικά κατανεμημένα τα δείγματα.



ΕΙΚΟΝΑ 68 LOOK UP BOXPLOT

Σύμφωνα με την παραπάνω εικόνα, δεν παρατηρούμε κάποιες ακραίες τιμές.

Tests of Normality

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
lookUp_difference	.100	25	.200*	.953	25	.295

*. This is a lower bound of the true significance.

a. Lilliefors Significance Correction

ΕΙΚΟΝΑ 69 ΤΕΣΤ ΚΑΝΟΝΙΚΟΤΗΤΑΣ LOOK UP

Σύμφωνα με τα παραπάνω, τα δεδομένα φαίνεται να είναι κανονικά κατανεμημένα, λόγω του ότι η τιμή p = 0.295 > 0.05 στη στήλη Sig. του *Shapiro-Wilk*.

Πλέον είμαστε σε θέση να εκτελέσουμε το ***paired-samples-t-test***. Τα αποτελέσματα θεωρούνται στατιστικά σημαντικά εάν $p < 0.05$. Θεωρούμε ως υπόθεση την εξής:

Ο χρήστης κοιτάει γρηγορότερα το στόχο όταν χρησιμοποιεί ως ελεγκτή το κινητό αντί για το air-tap.

Παρακάτω παρουσιάζονται τα αποτελέσματα του τεστ:

Paired Samples Statistics

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	Look Up Phone	1.24693	25	.392219	.078444
	Look Up Native	1.31144	25	.349299	.069860

ΕΙΚΟΝΑ 70

Paired Samples Correlations

		N	Correlation	Sig.
Pair 1	Look Up Phone & Look Up Native	25	.660	<.001

ΕΙΚΟΝΑ 71

Paired Samples Test

		Paired Differences					t	df	Sig. (2-tailed)
		Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference Lower	Upper			
Pair 1	Look Up Phone - Look Up Native	-.064510	.308228	.061646	-.191741	.062720	-1.046	24	.306

ΕΙΚΟΝΑ 72

Συνεπώς, μπορούμε να πούμε πως:

Η χρήση του κινητού ως ελεγκτή δε φαίνεται να μείωσε το χρόνο που χρειάζεται ο χρήστης να κοιτάξει τον επόμενο στόχο (-0.65 ± 0.062 s [μέση τιμή ± τυπικό σφάλμα]), διότι το αποτέλεσμα δεν ήταν στατιστικά σημαντικό (95% CI), t(24) = -1.045, p = 0.306.

4.4.2 Χρόνος επιλογής του επόμενου στόχου

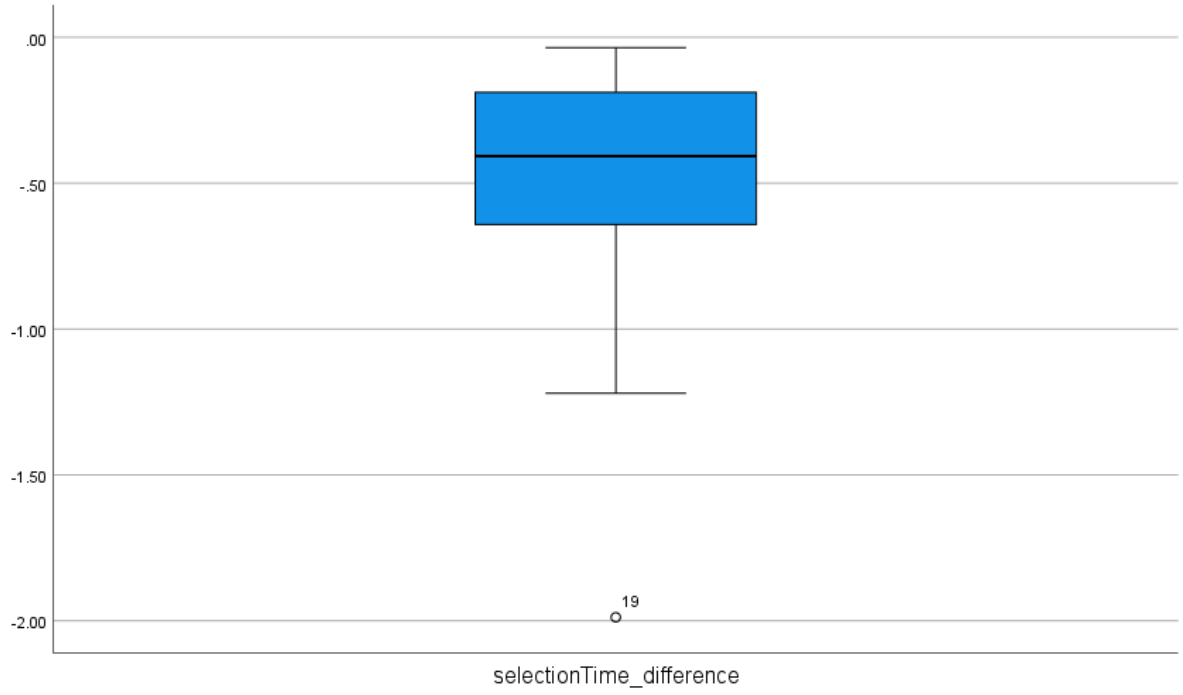
Το άλλο μέγεθος που μας ενδιαφέρει στις 2 διαφορετικές περιπτώσεις (*air-tap*, *κινητό*) όσον αφορά την Απόκριση Χρόνου είναι εάν επηρεάζεται η ικανότητα του χρήστη να επιλέξει τον επόμενο στόχο ανάλογο τον τρόπο αλληλεπίδρασης που χρησιμοποιεί. Για τη στατιστική ανάλυση επιλέχθηκε να γίνει ξανά το ***paired-samples-t-test***.

Ορίζουμε λοιπόν

$$selectionTime_{Difference} = selectionTime_{phone} - selectionTime_{native}$$

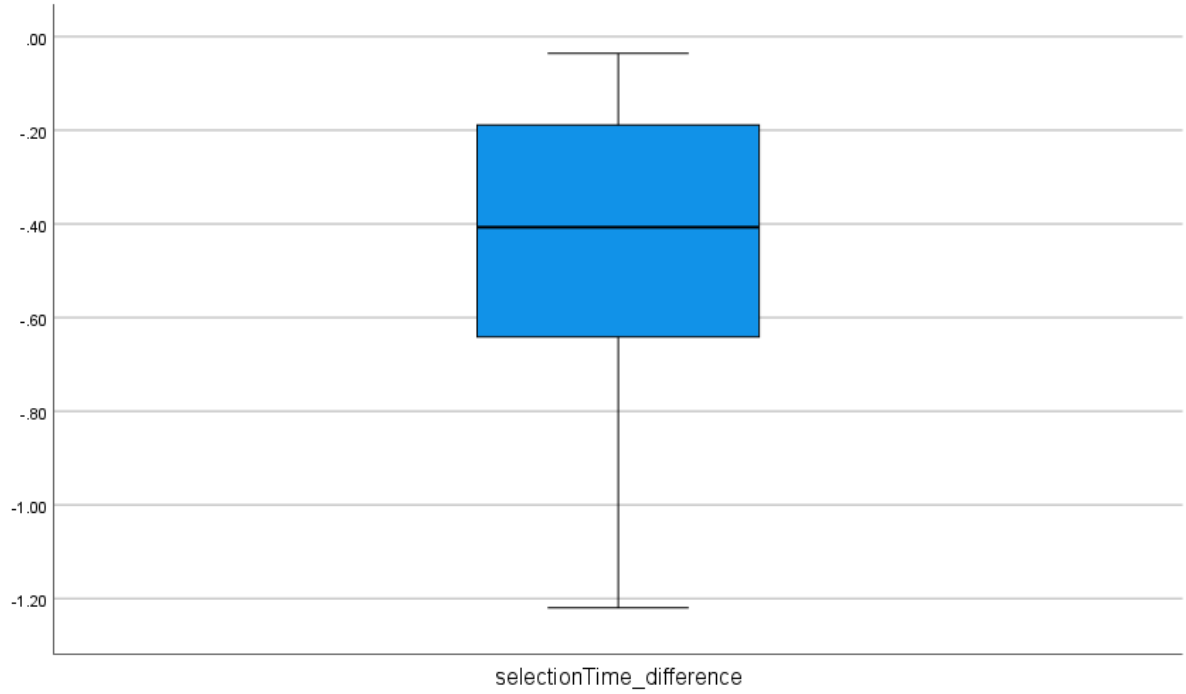
Για να συνεχίσουμε, πρέπει να βεβαιωθούμε πως για τη διαφορά:

- Δεν υπάρχουν ακραίες τιμές
- Είναι κανονικά κατανεμημένα τα δείγματα.



ΕΙΚΟΝΑ 73 SELECTION TIME BOXPLOT WITH OUTLIER

Με βάση την παραπάνω εικόνα, παρατηρούμε μία ακραία τιμή στο δείγμα 19. Επιλέγουμε να πειράξουμε το δείγμα αυτό θέτοντας την τιμή του *selectionTimeNative* από 2.9 σε 1.66, το αμέσως μεγαλύτερο. Η αλλαγή αυτή εξαφάνισε την ακραία τιμή, η οποία θεωρήθηκε πως ήταν μια πολύ ειδική περίπτωση του πειράματος.



ΕΙΚΟΝΑ 74 MODIFIED SELECTION TIME BOXPLOT

Tests of Normality

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
selectionTime_difference	.119	25	.200*	.935	25	.111

*. This is a lower bound of the true significance.

a. Lilliefors Significance Correction

ΕΙΚΟΝΑ 75 ΤΕΣΤ ΚΑΝΟΝΙΚΟΤΗΤΑΣ SELECTION TIME

Σύμφωνα με τα παραπάνω, τα δεδομένα φαίνεται να είναι κανονικά κατανεμημένα, λόγω του ότι η τιμή p = 0.111 > 0.05 στη στήλη Sig. του *Shapiro-Wilk*.

Πλέον είμαστε σε θέση να εκτελέσουμε το ***paired-samples-t-test***. Τα αποτελέσματα θεωρούνται στατιστικά σημαντικά εάν $p < 0.05$. Θεωρούμε ως υπόθεση την εξής:

Ο χρήστης επιλέγει γρηγορότερα το στόχο όταν χρησιμοποιεί ως ελεγκτή το κινητό αντί για το air-tap.

Παρακάτω παρουσιάζονται τα αποτελέσματα του τεστ:

Paired Samples Statistics

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	Selection Time Phone	.40651	25	.178041	.035608
	Selection Time Native	.86206	25	.374917	.074983

ΕΙΚΟΝΑ 76

Paired Samples Correlations

		N	Correlation	Sig.
Pair 1	Selection Time Phone & Selection Time Native	25	.553	.004

ΕΙΚΟΝΑ 77

Paired Samples Test

		Paired Differences					t	df	Sig. (2-tailed)
		Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference Lower	Upper			
Pair 1	Selection Time Phone - Selection Time Native	-.455549	.313818	.062764	-.585087	-.326011	-7.258	24	<.001

ΕΙΚΟΝΑ 78

Συνεπώς, μπορούμε να πούμε πως:

Η χρήση του κινητού ως ελεγκτή μείωσε το χρόνο που χρειάζεται ο χρήστης να επιλέξει τον επόμενο στόχο κατά -0.46 ± 0.063 s [μέση τιμή ± τυπικό σφάλμα], αποτέλεσμα το οποίο είναι στατιστικά σημαντικό (95% CI), t(24) = -7.258, p < 0.001

4.5 Απόκριση στόχων

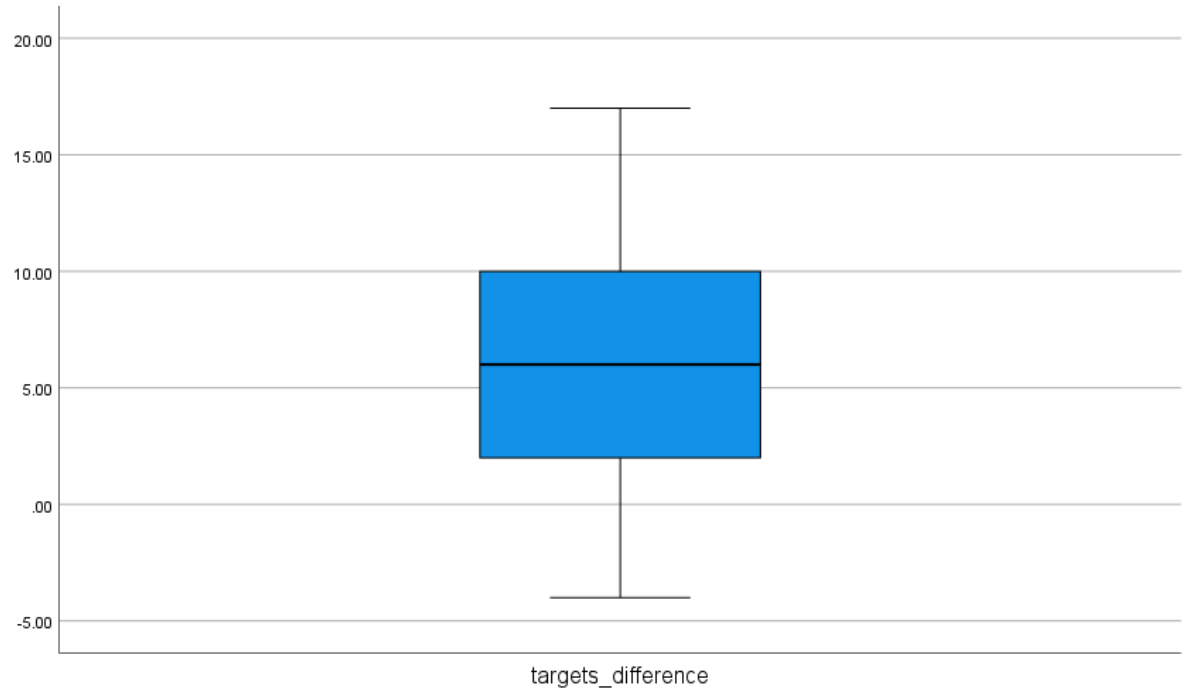
Στην Απόκριση Στόχων μας ενδιαφέρει πόσους στόχους πέτυχε ο χρήστης σε 45 δευτερόλεπτα στις 2 διαφορετικές περιπτώσεις (*air-tap, κινητό*). Για τη στατιστική ανάλυση επιλέχθηκε να γίνει ξανά το ***paired-samples-t-test***.

Ορίζουμε λοιπόν

$$targets_{Difference} = targets_{phone} - targets_{native}$$

Για να συνεχίσουμε, πρέπει να βεβαιωθούμε πως για τη διαφορά:

- Δεν υπάρχουν ακραίες τιμές
- Είναι κανονικά κατανεμημένα τα δείγματα.



ΕΙΚΟΝΑ 79 TARGETS BOXPLOT

Σύμφωνα με την παραπάνω εικόνα, δεν παρατηρούνται ακραίες τιμές στα δεδομένα.

Tests of Normality

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
targets_difference	.122	25	.200*	.971	25	.677

*. This is a lower bound of the true significance.

a. Lilliefors Significance Correction

ΕΙΚΟΝΑ 80 ΤΕΣΤ ΚΑΝΟΝΙΚΟΤΗΤΑΣ TARGETS

Επιπλέον, τα δεδομένα φαίνεται να είναι κανονικά κατανεμημένα, λόγω του ότι η τιμή p = 0.677 > 0.05 στη στήλη Sig. του *Shapiro-Wilk*.

Πλέον είμαστε σε θέση να εκτελέσουμε το ***paired-samples-t-test***. Τα αποτελέσματα θεωρούνται στατιστικά σημαντικά εάν $p < 0.05$. Θεωρούμε ως υπόθεση την εξής:

Ο χρήστης επιλέγει περισσότερους στόχους όταν χρησιμοποιεί ως ελεγκτή το κινητό αντί για το air-tap.

Παρακάτω παρουσιάζονται τα αποτελέσματα του τεστ:

Paired Samples Statistics

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	Targets Phone	22.64	25	8.031	1.606
	Targets Native	16.36	25	7.141	1.428

ΕΙΚΟΝΑ 81

Paired Samples Correlations

		N	Correlation	Sig.
Pair 1	Targets Phone & Targets Native	25	.778	<.001

ΕΙΚΟΝΑ 82

Paired Samples Test

		Paired Differences					t	df	Sig. (2-tailed)
		Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference Lower	Upper			
Pair 1	Targets Phone - Targets Native	6.280	5.128	1.026	4.163	8.397	6.124	24	<.001

ΕΙΚΟΝΑ 83

Συνεπώς, μπορούμε να πούμε πως:

Η χρήση του κινητού ως ελεγκτή αύξησε τους στόχους που επιλέγει ο χρήστης κατά -6.28 ± 1.03 [μέση τιμή ± τυπικό σφάλμα], αποτέλεσμα το οποίο είναι στατιστικά σημαντικό (95% CI), t(24) = 6.124, p < 0.001

4.6 Ερωτηματολόγιο

Αφού τελείωσε το πείραμα, δόθηκε ένα ερωτηματολόγιο. Δεν αποτελεί κάποιο επίσημο ερωτηματολόγιο αλλά βασίστηκε στο ερωτηματολόγιο NASA – TLX [75]

Οι ερωτήσεις βαθμολογήθηκαν σε κλίμακα από 1-5, με 1 να είναι το κινητό και 5 το *air-tap*. Παρακάτω σημειώνουμε τις ερωτήσεις:

- Νοητικά ποιος τρόπος αλληλεπίδρασης σας κούρασε παραπάνω;
- Σωματικά ποιος τρόπος αλληλεπίδρασης σας κούρασε παραπάνω;
- Με ποιον τρόπο αλληλεπίδρασης νιώσατε πως πετύχατε καλύτερα;

Επιπλέον, έγινε μια τελευταία ερώτηση, η κλίμακα της οποίας ήταν από 1-5, αλλά αυτή τη φορά δεν αφορούσε τους τρόπους αλληλεπίδρασης:

- Ενοχληθήκατε από την εμπειρία;

Δίνεται ο μέσος όρων των αποτελεσμάτων υπό μορφή πίνακα:

Ερώτηση	Αποτέλεσμα
Νοητική κούραση	3.08
Σωματική κούραση	3.68
Επίτευξη στόχου	1.16
Ενόχληση / ανασφάλεια	1

Προκύπτει λοιπόν πως οι χρήστες δεν αισθάνθηκαν πως κουράστηκαν το ίδιο και στις 2 περιπτώσεις, ενώ πιστεύουν πως πέτυχαν τον στόχο των δραστηριοτήτων πολύ καλύτερα στο κινητό. Επιπλέον, κανείς δεν ένιωσε ενόχληση ή ανασφάλεια.

Τέλος, να αναφερθεί πως στους περισσότερους άρεσε η εφαρμογή και πως ήταν σχεδιασμένη, καθώς και γενικά η μικτή πραγματικότητα. Μερικοί όμως θεωρούν πως η χρήση του κινητού δεν είναι τόσο φυσική όσο η χρήση των χεριών.

Κεφάλαιο 5 – Επίλογος

Τα γυαλιά μικτής πραγματικότητας HoloLens 2 προσφέρουν αρκετούς τρόπους εγγενούς αλληλεπίδρασης, αλλά πολλές φορές περιορίζουν το χρήστη λόγω του σχετικά μικρού πλαισίου της κάμερας στο οποίο ανιχνεύονται τα χέρια. Κατά τη διάρκεια της διπλωματικής αναπτύχθηκε ένα πλαίσιο επικοινωνίας, με βάση του οποίου δίνονται νέοι τρόπο αλληλεπίδρασης στο χρήστη. Επιπλέον αξιολογήθηκε ένας νέος τρόπος αλληλεπίδρασης, βάσει του οποίου ο χρήστης επιλέγει απομακρυσμένους στόχους χρησιμοποιώντας το δείκτη κεφαλή και το κινητό. Τα αποτελέσματα ήταν τα εξής:

1) Όσον αφορά το πόσο γρήγορα βλέπει ο χρήστης τον επόμενο στόχο, δε φάνηκε κάποια ιδιαίτερη διαφορά από τον εγγενή τρόπο.
2) Όσον αφορά το πόσο γρήγορα ο χρήστης επιλέγει το στόχο, φάνηκε στατιστική διαφορά, με το νέο τρόπο αλληλεπίδρασης να είναι πιο γρήγορος.
3) Ο χρήστης πετυχαίνει περισσότερους στόχους σε δεδομένο χρονικό διάστημα χρησιμοποιώντας το κινητό.

Με βάση το απλό αυτό πείραμα δείξαμε πως αξίζει η επιπλέον έρευνα άλλων τρόπων αλληλεπίδρασης που ίσως διευκολύνουν το χρήστη ή ακόμα και κάνουν την εμπειρία πιο ευχάριστη και πιο όμορφη.

Βιβλιογραφία

[1] P. Milgram, H. Takemura, A. Utsumi, and F. Kishino, “Augmented reality: A class of displays on the reality-virtuality continuum,” *Telemanipulator Telepresence Technol.*, vol. 2351, Jan. 1994, https://doi.org/10.1117/12.197321

[2] “Extended reality,” *Wikipedia*. Sep. 18, 2022. Accessed: Sep. 21, 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Extended_reality&oldid=1110892164

[3] “Virtual reality,” *Wikipedia*. Sep. 13, 2022. Accessed: Sep. 21, 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Virtual_reality&oldid=1110016947

[4] M. Speicher, B. Hall, and M. Nebeling, “What is Mixed Reality?,” May 2019. https://doi.org/10.1145/3290605.3300767

[5] M. Javaid and A. Haleem, “Virtual Reality applications toward medical field,” *Clin. Epidemiol. Glob. Health*, vol. 8, Dec. 2019, https://doi.org/10.1016/j.cegh.2019.12.010

[6] H. O. Barros, M. M. Soares, E. L. R. Filho, W. Correia, and F. Campos, “Virtual Reality Immersion: An Important Tool for Diagnostic Analysis and Rehabilitation of People with Disabilities,” *Lect. Notes Comput. Sci.*, pp. 337–344, 2013.

[7] “Half-Life: Alyx,” *Half-Life*. https://www.half-life.com/en/alyx (accessed Sep. 21, 2022).

[8] L. digital studio, “Beat Saber - VR rhythm game.” https://www.beatsaber.com/ (accessed Sep. 21, 2022).

[9] H. Luo, G. Li, Q. Feng, Y. Yang, and M. Zuo, “Virtual reality in K-12 and higher education: A systematic review of the literature from 2000 to 2019,” *J. Comput. Assist. Learn.*, vol. 37, no. 3, pp. 887–901, 2021, https://doi.org/10.1111/jcal.12538

[10] “Augmented reality,” *Wikipedia*. Sep. 20, 2022. Accessed: Sep. 21, 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Augmented_reality&oldid=1111308728

[11] “Pokémon GO,” *Pokémon GO*. https://pokemongolive.com/ (accessed Sep. 21, 2022).

[12] H. Soneria, “AR Decor: Decoration using markerless Augmented Reality,” *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 8, no. 5, p. 2145, 2020.

[13] “Houzz - Home Design & Remodel - Εφαρμογές στο Google Play.” https://play.google.com/store/apps/details?id=com.houzz.app&hl=el&gl=GR (accessed Sep. 21, 2022).

[14] F. Tscheu and D. Buhalis, “Augmented Reality at Cultural Heritage sites,” 2016, pp. 607–619. https://doi.org/10.1007/978-3-319-28231-2_44

[15] G. Martin *et al.*, “Use of the HoloLens2 Mixed Reality Headset for Protecting Health Care Workers During the COVID-19 Pandemic: Prospective, Observational Evaluation,” *J. Med. Internet Res.*, vol. 22, no. 8, p. e21486, Aug. 2020, https://doi.org/10.2196/21486

[16] D. F. Bosché, D. M. Abdel-Wahab, and D. L. Carozza, “Towards a Mixed Reality System for Construction Trade 1 Training 2,” 2015. https://www.semanticscholar.org/paper/Towards-a-Mixed-Reality-System-for-Construction-1-2-Bosch%C3%A9-Abdel-Wahab/f3de52d5e7f998fce95cbe7be9c73c2c0ed04607 (accessed Sep. 21, 2022).

[17] Yilei Huang, Samjhana Shakya, and Temitope Odeleye, “Comparing the Functionality between Virtual Reality and Mixed Reality for Architecture and Construction Uses,” *J. Civ. Eng. Archit.*, vol. 13, no. 7, Jul. 2019, https://doi.org/10.17265/1934-7359/2019.07.001

[18] A. Muñoz and A. Martí Testón, “HOLOMUSEUM: A HOLOLENS APPLICATION FOR CREATING EXTENSIBLE AND CUSTOMIZABLE HOLOGRAPHIC EXHIBITIONS,” Jul. 2018, pp. 2303–2310. https://doi.org/10.21125/edulearn.2018.0625

[19] W. Hou, “Augmented Reality Museum Visiting Application based on the Microsoft HoloLens,” *J. Phys. Conf. Ser.*, vol. 1237, p. 052018, Jun. 2019, https://doi.org/10.1088/1742-6596/1237/5/052018

[20] T. Davis, “Homogeneous coordinates and computer graphics,” Dec. 2001.

[21] “Homogeneous coordinates,” *Wikipedia*. May 26, 2022. Accessed: Sep. 21, 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Homogeneous_coordinates&oldid=1089902994

[22] Y.-B. Jia, “Quaternions and Rotations,” p. 12.

[23] G. S. and B. Eater, “Visualizing quaternions, an explorable video series.” https://eater.net/quaternions (accessed Sep. 22, 2022).

[24] R. Saroha, R. Bala, and S. Siwach, “Review paper on Overview of Image Processing and Image Segmentation,” p. 13, 2013.

[25] D. Ziou and S. Tabbone, “Edge Detection Techniques - An Overview,” vol. 8, Jun. 2000.

[26] “How does the Microsoft HoloLens work?,” *Quora*. https://www.quora.com/How-does-the-Microsoft-HoloLens-work (accessed Sep. 22, 2022).

[27] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, “Automatic generation and detection of highly reliable fiducial markers under occlusion,” *Pattern Recognit.*, vol. 47, no. 6, pp. 2280–2292, Jun. 2014, https://doi.org/10.1016/j.patcog.2014.01.005

[28] N. A. Ismail, T. C. Wen, S. Salam, A. M. Nawi, and S. E. N. Mohamed, “A Review Of Visual Inertial Odometry For Object Tracking And Measurement,” vol. 9, no. 02, p. 7, 2020.

[29] “Visual odometry,” *Wikipedia*. Jul. 23, 2022. Accessed: Sep. 23, 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Visual_odometry&oldid=1100024244

[30] M. Maimone, Y. Cheng, and L. Matthies, “Two years of Visual Odometry on the Mars Exploration Rovers,” *J. Field Robot.*, vol. 24, no. 3, pp. 169–186, Mar. 2007, https://doi.org/10.1002/rob.20184

[31] “OpenCV: Optical Flow.” https://docs.opencv.org/3.4/db/d7f/tutorial_js_lucas_kanade.html (accessed Sep. 23, 2022).

[32] Y. Wang, L. Chen, P. Wei, and X. Lu, “Visual-Inertial Odometry of Smartphone under Manhattan World,” *Remote Sens.*, vol. 12, no. 22, p. 3818, Nov. 2020, https://doi.org/10.3390/rs12223818

[33] D. Ksentini, A. R. Elhadi, and N. Lasla, “Inertial Measurement Unit: Evaluation for Indoor Positioning,” in *2014 International Conference on Advanced Networking Distributed Systems and Applications*, Bejaia, Algeria, Jun. 2014, pp. 25–30. https://doi.org/10.1109/INDS.2014.12

[34] O. J. Woodman, “An introduction to inertial navigation,” p. 37.

[35] K. Yousif, A. Bab-Hadiashar, and R. Hoseinnezhad, “An Overview to Visual Odometry and Visual SLAM: Applications to Mobile Robotics,” *Intell. Ind. Syst.*, vol. 1, no. 4, pp. 289–311, Dec. 2015, https://doi.org/10.1007/s40903-015-0032-7

[36] N. Karlsson, E. di Bernardo, J. Ostrowski, L. Goncalves, P. Pirjanian, and M. E. Munich, “The vSLAM Algorithm for Robust Localization and Mapping,” in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, Barcelona, Spain, 2005, pp. 24–29. https://doi.org/10.1109/ROBOT.2005.1570091

[37] D. Dornellas, F. Rosa, A. Bernardino, R. Ribeiro, and J. Santos-Victor, “GPS emulation via visual-inertial odometry for inspection drones,” Dec. 2019, pp. 755–760. https://doi.org/10.1109/ICAR46387.2019.8981597

[38] scooley, “HoloLens 2 hardware.” https://learn.microsoft.com/en-us/hololens/hololens2-hardware (accessed Sep. 24, 2022).

[39] Mamaylya, “HoloLens 2 gestures (for example, gaze and air tap) for navigating a guide in Dynamics 365 Guides - Dynamics 365 Mixed Reality.” https://learn.microsoft.com/en-us/dynamics365/mixed-reality/guides/operator-gestures-hl2 (accessed Sep. 24, 2022).

[40] sostel, “Eye tracking overview - Mixed Reality.” https://learn.microsoft.com/en-us/windows/mixed-reality/design/eye-tracking (accessed Sep. 24, 2022).

[41] sostel, “Eye-gaze-based interaction - Mixed Reality.” https://learn.microsoft.com/en-us/windows/mixed-reality/design/eye-gaze-interaction (accessed Sep. 24, 2022).

[42] Sean-Kerawala, “Hand tracking in Unreal - Mixed Reality.” https://learn.microsoft.com/en-us/windows/mixed-reality/develop/unreal/unreal-hand-tracking (accessed Sep. 25, 2022).

[43] scooley, “Getting around HoloLens 2.” https://learn.microsoft.com/en-us/hololens/hololens2-basic-usage (accessed Sep. 24, 2022).

[44] evmill, “Use your voice to operate HoloLens.” https://learn.microsoft.com/en-us/hololens/hololens-cortana (accessed Sep. 25, 2022).

[45] thetuvix, “Voice input in Unity - Mixed Reality.” https://learn.microsoft.com/en-us/windows/mixed-reality/develop/unity/voice-input-in-unity (accessed Sep. 25, 2022).

[46] caseymeekhof, “Point and commit with hands - Mixed Reality.” https://learn.microsoft.com/en-us/windows/mixed-reality/design/point-and-commit (accessed Sep. 25, 2022).

[47] U. Technologies, “Unity - Manual: Unity User Manual 2021.3 (LTS).” https://docs.unity3d.com/Manual/index.html (accessed Sep. 25, 2022).

[48] U. Technologies, “Unity - Manual: Order of execution for event functions.” https://docs.unity3d.com/Manual/ExecutionOrder.html (accessed Sep. 25, 2022).

[49] U. Technologies, “Unity - Manual: Plug-ins.” https://docs.unity3d.com/Manual/Plugins.html (accessed Sep. 26, 2022).

[50] “Input System | Input System | 1.4.2.” https://docs.unity3d.com/Packages/com.unity.inputsystem@1.4/manual/index.html (accessed Sep. 26, 2022).

[51] “Layouts | Input System | 1.4.2.” https://docs.unity3d.com/Packages/com.unity.inputsystem@1.4/manual/Layouts.html (accessed Sep. 26, 2022).

[52] “About AR Foundation | AR Foundation | 4.1.12.” https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.1/manual/index.html (accessed Sep. 25, 2022).

[53] polar-kev, “MRTK2-Unity Developer Documentation - MRTK 2.” https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/ (accessed Sep. 25, 2022).

[54] TerryGLee, “Code editor features - Visual Studio (Windows).” https://learn.microsoft.com/en-us/visualstudio/ide/index-writing-code (accessed Sep. 26, 2022).

[55] A. Poškevičius, “Barebones Master Server.” Sep. 20, 2022. Accessed: Sep. 26, 2022. [Online]. Available: https://github.com/alvyxaz/barebones-masterserver

[56] BillWagner, “Serialization (C#).” https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/serialization/ (accessed Sep. 26, 2022).

[57] “Endianness,” *Wikipedia*. Sep. 25, 2022. Accessed: Sep. 26, 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Endianness&oldid=1112276452

[58] BillWagner, “Built-in types - C# reference.” https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/builtin-types/built-in-types (accessed Sep. 26, 2022).

[59] BillWagner, “Reflection (C#).” https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/reflection (accessed Sep. 26, 2022).

[60] M. Blackstock, “Aspect Weaving with C# and .NET,” Jan. 2004.

[61] “Mirror Networking – Open Source Networking for Unity.” https://mirror-networking.com/ (accessed Sep. 26, 2022).

[62] “Transports.” https://mirror-networking.gitbook.io/docs/transports (accessed Sep. 26, 2022).

[63] Linwei, “KCP - A Fast and Reliable ARQ Protocol.” Sep. 26, 2022. Accessed: Sep. 26, 2022. [Online]. Available: https://github.com/skywind3000/kcp/blob/f553df7afc46fbad599bb6d92484baa02dc83f86/README.en.md

[64] “Vuforia Developer Portal |.” https://developer.vuforia.com/ (accessed Sep. 28, 2022).

[65] “EasyAR|Augmented Reality & AR SDK.” https://www.easyar.com/ (accessed Sep. 28, 2022).

[66] stevewhims, “C++/WinRT - UWP applications.” https://learn.microsoft.com/en-us/windows/uwp/cpp-and-winrt-apis/ (accessed Sep. 28, 2022).

[67] “CMake.” https://cmake.org/ (accessed Sep. 28, 2022).

[68] M. Doughty, “ArUcoDetectionHoloLens-Unity.” Sep. 09, 2022. Accessed: Sep. 28, 2022. [Online]. Available: https://github.com/doughtmw/ArUcoDetectionHoloLens-Unity

[69] “OpenCV: Detection of ArUco Boards.” https://docs.opencv.org/3.4/db/da9/tutorial_aruco_board_detection.html (accessed Sep. 28, 2022).

[70] “Dots per inch,” *Wikipedia*. Aug. 12, 2022. Accessed: Sep. 28, 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Dots_per_inch&oldid=1104072407

[71] U. Technologies, “Unity - Scripting API: Screen.dpi.” https://docs.unity3d.com/ScriptReference/Screen-dpi.html (accessed Sep. 28, 2022).

[72] Ν. Αβούρης, Χ. Κατσάνος, Ν. Τσέλιος, and Κ. Μουστάκας, *Εισαγωγή στην Αλληλεπίδραση Ανθρώπου-Υπολογιστή*, 2nd ed. ΕΚΔΟΣΕΙΣ ΠΑΝΕΠΙΣΤΗΜΙΟΥ ΠΑΤΡΩΝ, 2018.

[73] “Firebase Realtime Database,” *Firebase*. https://firebase.google.com/docs/database (accessed Sep. 29, 2022).

[74] CDiaz-MS, “Solver overview - MRTK 2.” https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/features/ux-building-blocks/solvers/solver (accessed Sep. 29, 2022).

[75] S. G. Hart, “nasa task load index (tlx),” Jan. 1986, Accessed: Sep. 30, 2022. [Online]. Available: https://ntrs.nasa.gov/citations/20000021487