ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΜΗΧΑΝΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΤΩΝ ΚΑΤΕΡΓΑΣΙΩΝ

Χωροταξική Διάταξη και Προσομοίωση Λειτουργίας Ευέλικτου Συστήματος Κατεργασιών με Χρήση Επαυξημένης Πραγματικότητας

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ανδρέας Ν. Κόκκας

Επιβλέπων Καθηγητής: Γεώργιος-Χριστόφορος Βοσνιάκος

Αθήνα, Φεβρουάριος 2018

Περιεχόμενα

Ευχαριστίες

Σε αυτό το σημείο, θα ήθελα να εκφράσω τις πολύ θερμές που ευχαριστίες στον κ. Γ.-Χ. Βοσνιάκο, καθηγητή του Ε.Μ.Π., για την ανάθεση της συγκεκριμένης διπλωματικής εργασίας, όπως επίσης για την καθοριστική υποστήριξη του κατά τη διάρκεια της εκπόνησης της.

Επίσης θα ήθελα να ευχαριστήσω όλο το έμψυχο δυναμικό του Εργαστηρίου Τεχνολογίας των Κατεργασιών και ιδιαίτερα τον Γιώργο Παπαζέτη, Υποψήφιο Διδάκτορα της σχολής Μηχανολόγων Μηχανικών Ε.Μ.Π., για την συμβολή του στην χρήση του εξοπλισμού του εργαστηρίου.

Τέλος, δεν θα μπορούσα να παραλείψω την οικογένεια μου, την οποία ευχαριστώ πολύ για την στήριξη που μου παρείχε καθ' όλη την διάρκεια των σπουδών μου.

Ανδρέας Κόκκας
Αθήνα, Φεβρουάριος 2018

Περίληψη

Η Επαυξημένη Πραγματικότητα (Augmented Reality-AR) αφορά μια τεχνολογία η οποία σήμερα φαίνεται να διανύει μια πολύ ώριμη περίοδο της εξέλιξης της καθώς επίσης να απευθύνεται σε ένα εξίσου ώριμο για εφαρμογές επαυξημένης πραγματικότητας κοινό. Ο εικονικός κόσμος και η αλληλεπίδραση του με τον πραγματικό αποτελούν ένα πολύ επίκαιρο θέμα, με εταιρίες παγκόσμιου βεληνεκούς να διεκδικούν την πρωτοκαθεδρία καθώς αποτελεί μια τεχνολογία συμβατή σχεδόν με οποιονδήποτε επαγγελματικό κλάδο και ιδιαίτερα με την βιομηχανία.

Το αντικείμενο της παρούσας εργασίας είναι ο σχεδιασμός μηχανουργικών διατάξεων και πιο συγκεκριμένα ενός Ευέλικτου Συστήματος Κατεργασιών (Flexible Manufacturing System - FMS) με την χρήση εργαλείων επαυξημένης πραγματικότητας.

Χρησιμοποιώντας τον CNC τόρνο-φρέζα EMCO του Εργαστηρίου Τεχνολογίας των Κατεργασιών του ΕΜΠ, επιχειρείται η υπέρθεση των στοιχείων που λείπουν από ένα πλήρες “κύτταρο” FMS προκειμένου αρχικά να γίνει η αποτύπωση των πιθανών χωροταξικών εναλλακτικών καθώς και ο καθορισμός της βέλτιστης τελικής χωροθέτησης (Facility Layout Planning-FLP).

Στόχος είναι ο χρήστης να μπορεί να αξιολογεί τις χωροταξικές διατάξεις που προέκυψαν σαν αποτέλεσμα αναλυτικών ή αριθμητικών προσεγγίσεων βελτιστοποίησης, λαμβάνοντας υπόψιν μή μετρήσιμους παράγοντες όπως είναι η εμπειρία των χρηστών αλλά και η επι τόπου εκτίμηση των εναλλακτικών. Ο χρήστης επαυξημένης πραγματικότητας κατά τον σχεδιασμό μιας μηχανουργικής διάταξης, είναι σε θέση να περιηγηθεί στον μελλοντικό χώρο παραγωγής και να λάβει υπόψιν παράγοντες που ακόμα και η προσομοίωση σε εικόνικό περιβάλλον αγνοούσε.

Ως δεύτερο βήμα επιχειρείται η λειτουργική σύνδεση και προσομοίωση της συνεργασίας των εικονικών μοντέλων με την εργαλειομηχανή EMCO καθώς και του ρομπότ Staubli του εργαστηρίου σε μια κατά το δυνατόν πειστική αποτύπωση της διάταξης που πρόκειται να κατασκευαστεί. Η προσομοίωση της παραγωγικής διαδικασίας μιας μονάδας με κινούμενα μέρη όπως ένα κύτταρο κατεργασίας, προσφέρουν την δυνατότητα εκτενούς ανάλυσης της επιλεχθείσας διάταξης ώστε να αποτελέσει την βέλτιστη, ασφαλέστερη αλλά και βολικότερη λύση.

Τέλος εξάγονται συμπεράσματα για την αξιολόγηση της διαδικασίας που ακολουθήθηκε και των εργαλείων που χρησιμοποιήθηκαν (ARKit) καθώς επίσης προτείνονται και σημεία βελτίωσης της εφαρμογής ώστε να αποτελεί χρήσιμο εργαλείο στα χέρια του σχεδιαστή μιας μονάδας κατεργασίας ή μίας βιομηχανικής μονάδας εν γένει.

Abstract

Augmented Reality-AR is a technology that now seems to be going through a very mature period of its evolution, as well as addressing an equally mature audience for Augmented Reality applications. The virtual world and its interaction with the real one is a very topical issue, with world-class companies claiming primacy, as it is a technology that is compatible with almost any professional field, especially industry.

The subject of the thesis is the planning of mechanical layouts and more specifically of a Flexible Manufacturing System (FMS) using augmented reality tools.

Using the EMCO CNC lathe of the NTUA's Laboratory of Manufacturing Technology, it is attempted to superimpose the missing elements of a complete FMS " cell ", in order to initially map out possible spatial alternatives as well as determine the optimum final positioning (Facility Layout Planning-FLP).

The goal is for the user to be able to evaluate spatial layouts resulting from analytical or numerical optimization approaches, taking into account non-measurable factors such as user experience and on-site assessment of alternative layouts. The augmented reality user during the design of the manufacturing layout is able to navigate the future production area and take into account factors that even the simulation in a virtual reality environment was unaware of.

The second step is the functional connection and simulation of the collaboration of the virtual models with the existing equipment of the laboratory (EMCO Lathe and Staubli robot) in the most convincing way of the layout to be constructed (simulation of production process). Simulating the production process of a unit with moving parts such as a manufacturing cell offers the possibility of extensive analysis of the selected layout in order to be the optimal, safer and the more convenient solution.

Finally, conclusions are drawn on the evaluation of the process followed and the tools used (ARKit) as well as suggestions for improvements to the application to be a useful tool in the hands of the designer of a manufacturing cell or an industrial unit in general.

Πίνακας Εικόνων

1. Εισαγωγή

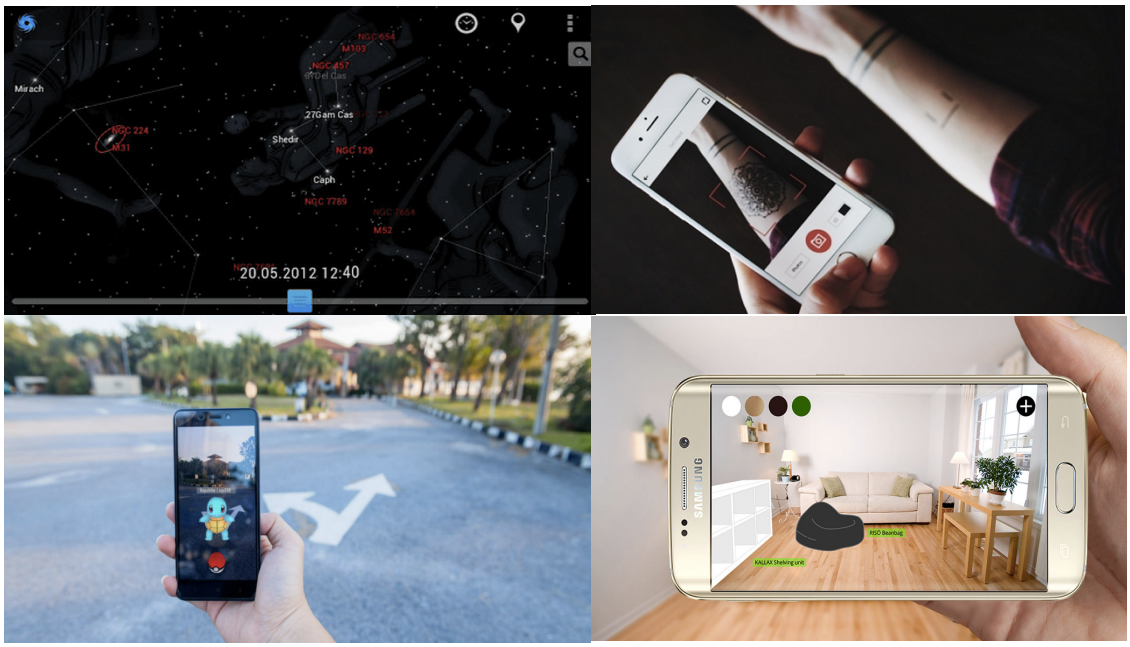
1.1. Πρόλογος

Επαυξημένη Πραγματικότητα (Augmented Reality-AR) είναι η τεχνολογία που κάνει υπέρθεση εικονικών μοντέλων στον πραγματικό χώρο και σε πραγματικό χρόνο, μέσω χαρακτηριστικών εφαρμογών σε ηλεκτρονικές συσκευές όπως κινητά, tablet, HMD(head mounted displays) κλπ. Η Επαυξημένη Πραγματικότητα στις μέρες μας είναι αρκετά ώριμη και έχει εισχωρήσει σε πολλούς κλάδους όπως η εκπαίδευση, η ιατρική, η αρχαιολογία, η διασκέδαση και ειδικότερα τα βιντεοπαιχνίδια, ο τουρισμός, ο αθλητισμός αλλά και η βιομηχανία ενώ αρκετές έρευνες προβλέπουν ότι θα υπάρξουν περίπου 1 δισεκατομμύριο χρήστες εφαρμογών Επαυξημένης Πραγματικότητας μέχρι το 2020.

Στη βιομηχανία η Επαυξημένη Πραγματικότητα έχει κάνει ήδη τα πρώτα της βήματα σε τομείς όπως η συντήρηση η συναρμολόγηση και η εκπαίδευση του προσωπικού αλλά και ο σχεδιασμός νέων γραμμών παραγωγής και προϊόντων.

Κατά την δημιουργία του περιβάλλοντος Επαυξημένης Πραγματικότητας η αντίστοιχη εφαρμογή εκμεταλλεύεται κάποιους από τους αισθητήρες που έχει ενσωματωμένους η συσκευή του χρήστη. Με βάσει αυτούς τους αισθητήρες γίνεται και η κατηγοριοποίηση των εφαρμογών επαυξημένης πραγματικότητας. Έτσι προκύπτουν οι παρακάτω τέσσερις (4) επικρατέστερες προσεγγίσεις:

- **Gyroscope-based AR**: Βασισμένες στο γυροσκόπιο της συσκευής, οι εφαρμογές μπορούν να τοποθετήσουν το εικονικό περιεχόμενο γύρω από τον χρήστη.
- **Marker-based AR**: Η εφαρμογή χρησιμοποιεί εικόνες ή αντικείμενα “στόχους” στον πραγματικό κόσμο. Η φωτογραφική μηχανή της συσκευής συλλαμβάνει τέτοιου είδους στόχους-ενεργοποιητές και στη συνέχεια, η εφαρμογή προβάλλει ένα συγκεκριμένο μοντέλο AR πάνω σε αυτούς.
- **Location-based AR**: Η εφαρμογή χρησιμοποιεί το GPS, την πυξίδα και το Wi-Fi για να προσθέσει τα εικονικά μοντέλα γύρω από τον χρήστη.
- **VIO (Visual Inertial Odometry) & SLAM (Simultaneous Localization And Mapping)**: Η εφαρμογή **ΔΕΝ** απαιτεί τον εντοπισμό κάποιου “στόχου” αλλά εκμεταλλεύεται τα δεδομένα του επιταχυνσιόμετρου του γυροσκόπιου και της κάμερας συγκρίνοντας την εικόνα που λαμβάνει σε κάθε frame με αυτήν που έλαβε στο αμέσως προηγούμενο.



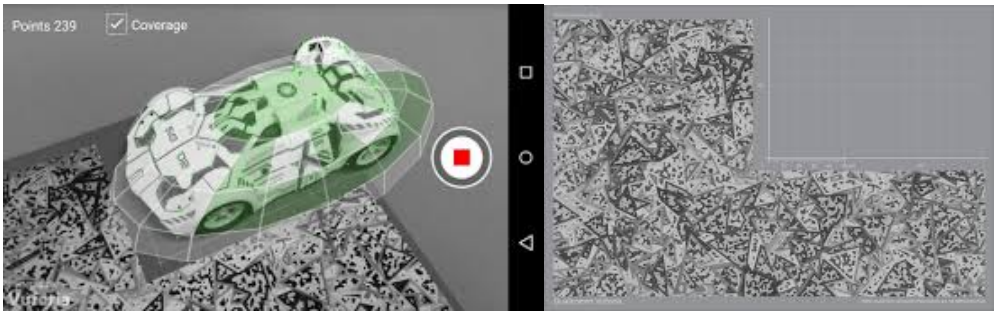
Εικόνα 1: Χαρακτηριστικά παραδείγματα εφαρμογών: Πάνω αριστερά: Vortex Planetarium - Gyroscope-based AR App, Πάνω δεξιά: Inkhunter-Marker-based AR App, Κάτω αριστερά: Pokemon Go-Location-based AR App, Κάτω δεξιά: IKEA AR App-VIO AR App.

Η ανάπτυξη εφαρμογών επαυξημένης πραγματικότητας, ιδιαίτερα πριν την ανακοίνωση της διεπαφής προγραμματισμού εφαρμογών (API) επαυξημένης πραγματικότητας, της Apple με όνομα "ARKit" τον Ιούνιο του 2017, στην συντριπτική της πλειοψηφία αφορούσε τον εντοπισμό μιας “Εικόνας Στόχου” (Marker-based AR) πάνω στην οποία κάποια κιτ ανάπτυξης λογισμικού (Software Development Kit-SDK) όπως το Vuforia SDK, αναλάμβαναν να κάνουν υπέρθεση των εικονικών μοντέλων που ο σχεδιαστής-προγραμματιστής είχε επιλέξει.

Παρά την μεγάλη δημοφιλία αυτών των ‘’εργαλείων’’, δεν ήταν εξίσου επιτυχημένη και αποδοτική η χρήση τους σε περιπτώσεις όπου το εικονικό μοντέλο ήταν αρκετά μεγαλύτερου μεγέθους σε σχέση με την εικόνα-στόχο. Έτσι σε αυτές τις περιπτώσεις έπρεπε ο σχεδιαστής της εφαρμογής επαυξημένης πραγματικότητας να γεμίσει το περιβάλλον με εικόνες στόχους και μάλιστα μεγάλου μεγέθους έτσι ώστε η κάμερα να κάνει αποδοτικό tracking των στόχων και να μην τους χάνει είτε λόγω απόστασης είτε λόγω μεγέθους. Και πάλι όλα αυτά για ένα μέτριο και ασυνεχές αποτέλεσμα.

Εικόνα 2: Εικόνες-Στόχοι

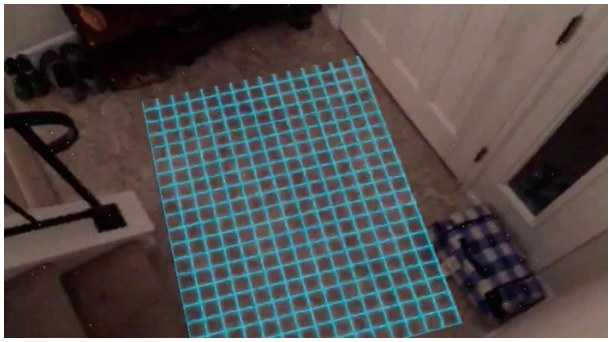
Το επόμενο βήμα ήταν τα ‘’Αντικείμενα-Στόχοι’’. Αυτά επέτρεπαν καλύτερο tracking γύρω από το αντικείμενο-στόχο και υπέρθεση μεγαλύτερων εικονικών αντικειμένων. Όμως το σκανάρισμα του αντικειμένου προκειμένου να το ορίσουμε ως στόχο ήταν μια χρονοβόρα διαδικασία και απαιτούσε πολύ συγκεκριμένο εξοπλισμό ενώ για κάθε διαφορετικό αντικείμενο στόχο η διαδικασία θα έπρεπε να επαναλαμβάνεται εξ αρχής.



Εικόνα 3: Αντικείμενο-Στόχος και εικόνα στόχος για σκανάρισμα του αντικειμένου

Βασικό μειονέκτημα των παραπάνω ήταν ότι πάντα η κάμερα έπρεπε να “βλέπει και να μην απομακρύνεται από τον στόχο είτε αυτός ήταν αντικείμενο είτε εικόνα.

Έτσι με την εμφάνιση του ARKit και έπειτα του ARCore της Google άνοιξε ένα καινούριο κεφάλαιο για την επαυξημένη πραγματικότητα και το που μπορεί να εφαρμοστεί. Η χρήση τεχνολογιών όπως Visual Inertial Odometry επέτρεψαν την υπέρθεση ενός εικονικού κόσμου οποιουδήποτε μεγέθους, ακριβώς πάνω στον πραγματικό, χωρίς καμία εξάρτηση από στόχους, παρά μόνο της αρχικής επίπεδης επιφάνειας (plane) την οποία χρησιμοποιούν σαν σημείο αναφοράς χωρίς βέβαια να χρειάζεται να την “βλέπουν” συνεχώς.



Εικόνα 4: Επίπεδη επιφάνεια (plane) του ARKit

Σήμερα, στην εποχή που πολλές μεγάλες εταιρίες τεχνολογίας (Microsoft, Facebook, Apple, Magic Leap κ.α.) προσπαθούν να αναπτύξουν τις δικές του πλατφόρμες επαυξημένης και μεικτής πραγματικότητας (AR, MR) παρουσιάζεται η ευκαιρία ανάπτυξης εφαρμογών για την βελτίωση κάποιων διαδικασιών που μέχρι πρότινος βασίζονταν σε εργαλεία VR. Ένας πολλά υποσχόμενος τομέας εφαρμογής αυτών των τεχνολογιών είναι και η βιομηχανία, η οποία φαίνεται να ασπάζεται με πολύ γρήγορο ρυθμό εργαλεία Επαυξημένης και Μεικτής πραγματικότητας.

1.2. Ανασκόπηση Βιβλιογραφίας

Η προσομοίωση μιας παραγωγικής διαδικασίας μέσα σε ένα περιβάλλον Εικονικής Πραγματικότητας (VR), όπου όλα τα αντικείμενα είναι εικονικά και καμία αλληλεπίδραση δεν υφίσταται με πραγματικό εξοπλισμό, είναι μια αρκετά ώριμη μέθοδος από τεχνολογικής άποψης. Δεδομένου ότι εργαλεία που χρησιμοποιήθηκαν κατά το παρελθόν για την ανάπτυξη βιντεοπαιχνιδιών, εύκολα επεκτάθηκαν στο να καλύψουν ανάγκες σχεδιασμού και προσομοίωσης στη βιομηχανία.

Ένα εικονικό περιβάλλον (Virtual Environment-VE) σε γενικές γραμμές αποτελείται από έναν εικονικό χώρο ή κόσμο και την ικανότητα αλληλεπίδρασης με τον χρήστη. Η δημιουργία ενός τέτοιου περιβάλλοντος συνεπάγεται την κατασκευή των αντικειμένων και σκηνών του εικονικού κόσμου και την εφαρμογή λειτουργιών αλληλεπίδρασης. Τα τελευταία χρόνια, χρησιμοποιήθηκαν πλατφόρμες λογισμικού που διευκολύνουν αυτά τα καθήκοντα ιδιαίτερα στον τομέα του Computer Integrated Manufacturing (CIM) [1]. Η συνεχής βελτίωση της λειτουργικότητας και η μείωση του κόστους των εικονικών περιβαλλόντων οδηγούν στην εξέταση της ενσωμάτωσής των συστημάτων CIM ως μια συνέχεια της Εικονικότητας-Πραγματικότητας (Reality-Virtuality), η οποία αναφέρεται τελευταία ως Virtual Manufacturing Systems [2].

Η χρήση Εικονικής Πραγματικότητας (VR) επεκτάθηκε περαιτέρω με την έλευση της έννοιας του ψηφιακού εργοστασίου. Η ποιοτική και ποσοτική ανάλυση της ροής υλικού [3], οι κίνδυνοι σύγκρουσης καθώς και ο χωροταξικός σχεδιασμός της εγκατάστασης [4] είναι μεταξύ των λειτουργιών που πρέπει να προσφέρει ένα σύστημα σχεδιασμού σε περιβάλλον VR. Η εμβύθιση στο VR εργοστάσιο (πχ με χρήση Head-Mounted Displays-HMD) αναφέρθηκε ότι διευκόλυνε την εκτίμηση των ελαττωμάτων σχεδιασμού διάταξης του εξοπλισμού σε σύγκριση με το μη εμβυθισμένο VR [5], ενώ η Επαυξημένη Πραγματικότητα (AR) [6] και η Μεικτή Πραγματικότητα (MR) [7] πιστεύεται ότι προσφέρουν ακόμα περισσότερες δυνατότητες στον χωροταξικό σχεδιασμό και αρχίζουν να αναγνωρίζονται και στη βιομηχανία.

Οι παραδοσιακές συμβάσεις προσομοίωσης και οι περιορισμοί εξομαλύνονται με τη χρήση εικονικού περιβάλλοντος [8]. Έχουν αναφερθεί ενδιαφέρουσες εφαρμογές στην κατασκευή και τη συναρμολόγηση των βιομηχανικών κυττάρων [9], στην ενσωμάτωση και παρακολούθηση δεδομένων στην αυτοκινητοβιομηχανία [10], στην εξέταση εναλλακτικών περιπτώσεων χωροθέτησης πριν την υλοποίηση των κυττάρων [11][12] με σύγκριση του πραγματικού και του εικονικού εργοστασίου και στην απεικόνιση των επιπτώσεων του συνολικού συστήματος [13]. Επιπλέον, έχουν προταθεί πλαίσια για τον σχεδιασμό εργαλείων εικονικών εργαλειομηχανών [14] και τον εικονικό σχεδιασμό του εργοστασίου [15], τον σχεδιασμό της λειτουργίας συστημάτων κατεργασίας [16] και την αξιολόγηση της διεργαστηριακής διεπιστημονικής διαδικασίας [17].

Γενικά, φαίνεται ότι τα VR μοντέλα για περιβάλλοντα με εργαλειομηχανές και ρομπότ είναι χρήσιμα για την διερεύνηση της αλληλεπίδρασης με τον χρήστη και για τον Χωροταξικό Σχεδιασμό (FLP), την λειτουργικότητα και την εργονομία μιας μονάδας παραγωγής. Για τον λόγο αυτό η μοντελοποίηση τους θα πρέπει να γίνει αρκετά εύρωστη ώστε όντως να μπορούν να προσομοιώσουν την λειτουργία των πραγματικών διατάξεων.

Πιο συγκεκριμένα, όσον αφορά τον έλεγχο, τα βιομηχανικά ρομπότ είναι παρόμοια με τις εργαλειομηχανές CNC αλλά πιο ευέλικτα, καθώς μπορούν να χειρίζονται υλικά και να συμμετέχουν ή να εκτελούν σε διαδικασίες κατεργασίας. Έτσι, έχουν εφαρμοστεί πιο προηγμένα μοντέλα VR και αυτό έγινε με βάση τις εμπορικά διαθέσιμες μηχανές ανάπτυξης [18] με έμφαση στην πολυτροπική αλληλεπίδραση [19] [20]. Η αντίστροφη κινηματική [21] και, πιο πρόσφατα, τα μοντέλα δυναμικής ανάλυσης [22] έχουν συνδεθεί με τις προσεγγίσεις προγραμματισμού ρομπότ με AR και VR.

Πλέον όμως οι προκλήσεις, η ακρίβεια, η ταχύτητα παραγωγής, η δυνατότητα απομακρυσμένου ελέγχου αλλά και η μείωση της ανοχής σε λάθη προδιαγράφει μια νέα κατάσταση στην παραγωγική διαδικασία προϊόντων. Αυτή η νέα εποχή απαρτίζεται από τεχνολογικούς παράγοντες όπως η 3D εκτύπωση, η αντίστροφη μηχανολογία (Reverse Engineering), η επαυξημένη πραγματικότητα (Augmented Reality), το Internet of Things (IoT), το Cloud Computing, η ανάλυση μεγάλου όγκου δεδομένων (Big Data Analysis) κ.α. Το σχεδόν πλήρως αυτοματοποιημένο βιομηχανικό περιβάλλον που εμπεριέχει τους περισσότερους, αν όχι όλους, τους παραπάνω τεχνολογικούς παράγοντες ονομάζεται **Industry 4.0**. [23]

‘Επίσης εργασίες που εκτελούνταν μέχρι τώρα από το προσωπικό, είτε βάσει εμπειρίας είτε βάσει σχεδίων όπως εργασίες συναρμολόγησης ή επίβλεψη των ενδείξεων λειτουργίας του εξοπλισμού (όπως πίεση, θερμοκρασία κλπ), στα πλαίσια μιας τέτοιας βιομηχανικής διάταξης είναι εφικτό να διενεργηθούν με την βοήθεια της επαυξημένης πραγματικότητας [24]. Τα σύγχρονα συστήματα κατεργασίας και ο εξοπλισμός CNC, ως ιδιαίτερα καινοτόμο υποσύνολο της βιομηχανίας, δεν θα μπορούσαν παρά να ακολουθήσουν αυτήν την τάση προκειμένου να γίνουν πιο εύρωστα και πιο ευέλικτα [25].

Όσον αφορά τον σχεδιασμό των βιομηχανικών ή μηχανουργικών διατάξεων σαν σύνολο, η χρήση εργαλείων προσομοίωσης σε περιβάλλον εικονικής πραγματικότητας ήταν ιδιαίτερα δημοφιλής τα τελευταία χρόνια καθώς αυτά εξασφάλιζαν την αναλυτική εξέταση όλων των πιθανών χωροταξικών διατάξεων του εξοπλισμού. Αυτή η διαδικασία αποτελεί ένα ιδιαίτερα σημαντικό εργαλείο στα χέρια του σχεδιαστή της διάταξης του εξοπλισμού καθώς μικρές αλλαγές μπορεί να επιφέρουν μεγάλο οικονομικό και χρονικό όφελος για την επιχείρηση καθώς και ευελιξία κατεργαζόμενου προϊόντος και όγκου παραγωγής. [26], [27], [28]. Το επόμενο βήμα ήταν η προσομοίωση λειτουργίας, για αυτό η δημιουργία μιας κυψέλης κατεργασιών εικονικής πραγματικότητας με δυνατότητα

προγραμματισμού περίπλοκων κινήσεων των ρομπότ και ανίχνευση συγκρούσεων on-line προστέθηκε στα υφιστάμενα εργαλεία σχεδιασμού κυψελών σε περιβάλλον εικονικής πραγματικότητας [29], [30].

Η χρήση Επαυξημένης Πραγματικότητας κατά τον Χωροταξικό Σχεδιασμό του Εξοπλισμού (Facility Lay-out Planning-FLP) είναι το επόμενο βήμα προς την επιλογή της βέλτιστης διάταξης του εξοπλισμού μιας μονάδας. Αυτό γιατί όταν τοποθετηθούν τα εικονικά μοντέλα δίπλα στον υφιστάμενο εξοπλισμό μιας παραγωγικής μονάδας μπορούν να ληφθούν υπόψιν και μη μετρήσιμοι παράγοντες, όπως η εμπειρία, η άνεση και ασφάλεια του προσωπικού προκειμένου η επιλεχθείσα διάταξη να μην είναι απλά η βέλτιστη αλλά και οι πιο βολική για τις ανάγκες της συγκεκριμένης μονάδας [31], [32].

1.3. Στόχος της Διπλωματικής Εργασίας

Στόχος της εργασίας είναι να δειχθεί με χαρακτηριστικές εφαρμογές η δυνατότητα χρήσης τεχνικών επαυξημένης πραγματικότητας για την υποστήριξη του σχεδιασμού και της λειτουργίας συστημάτων κατεργασιών δηλαδή περιβάλλοντος βιομηχανικής παραγωγής.

Ειδικότερα, στοχεύεται η επιλογή και χωροθέτηση επιπρόσθετου εξοπλισμού στο Εργαστήριο Τεχνολογίας των Κατεργασιών του ΕΜΠ όπου ήδη βρίσκονται εγκατεστημένες εργαλειομηχανές και ένα βιομηχανικό ρομπότ. Ο επιπρόσθετος εξοπλισμός επιλέχθηκε με στόχο να συμπληρώνει τον υφιστάμενο εξοπλισμό του εργαστηρίου προς την δημιουργία ενός Κυττάρου Κατεργασιών.

2. Χωροταξικός Σχεδιασμός (Facility Layout Planning-FLP)

2.1. Εισαγωγή

Ο χωροταξικός σχεδιασμός αφορά την χωροταξική διάταξη των διαφόρων εργασιών στο εσωτερικό των μονάδων παραγωγής και γενικότερα στοχεύει στον αποδοτικότερο σχεδιασμό των παραγωγικών συστημάτων. Κυρίως αφορά τη χωροταξική διευθέτηση των μηχανημάτων, του πάσης φύσεως εξοπλισμού και του ανθρώπινου δυναμικού μέσα στον χώρο παραγωγής. Ο χωροταξικός σχεδιασμός θα καθορίσει επίσης την ροή του υλικού στην γραμμή παραγωγής οπότε η δυναμικότητα μιας παραγωγικής μονάδας εξαρτάται άμεσα από αυτόν.

Η ανάγκη χωροταξικού σχεδιασμού προκύπτει κυρίως κατά:

• Τον σχεδιασμό νέου παραγωγικού συστήματος.

• Τον ανασχεδιασμό των λειτουργιών στο χώρο σε υπάρχοντα συστήματα.

Κατά τον ανασχεδιασμό υφιστάμενου συστήματος λαμβάνεται υπόψιν τόσο η λειτουργικότητα της διάταξης που θα προκύψει, όσο και η κατά το δυνατόν μικρότερη επίπτωση λόγω της αναδιάταξης του εξοπλισμού που ήδη υπάρχει και πρέπει να μετακινηθεί. Η ανάγκη για ανασχεδιασμό προκύπτει συνήθως κατά την:

α) μεταβολή στο σχέδιο ενός εξαρτήματος ή ενός προϊόντος, που απαιτεί αλλαγή διαδικασίας παραγωγής,

β) αύξηση του όγκου παραγωγής, όταν απαιτείται προσθήκη παραγωγικού δυναμικού,

γ) μείωση του όγκου παραγωγής, οπότε προκύπτει το αντίστροφο πρόβλημα,

δ) έναρξη παραγωγής ενός προϊόντος, οπότε χρειάζεται είτε προσθήκη δυναμικού στα υπάρχοντα τμήματα, είτε δημιουργία ενός νέου τμήματος,

ε) μεταφορά ενός τμήματος από μια θέση σε μια άλλη, οπότε δίνεται η ευκαιρία να διορθωθούν ενδεχόμενα λάθη ή αδυναμίες στην υπάρχουσα χωροταξία,

στ) αντικατάσταση παλαιού εξοπλισμού,

ζ) αλλαγή στη μέθοδο παραγωγής. [33]

Ο χωροταξικός σχεδιασμός συμβάλει στην επίλυση ή βελτίωση προβλημάτων όπως:

• καθυστερήσεις και μη παραγωγικός χρόνος,

• διαχείρισης αποθεμάτων,

• ουρές αναμονής (συνωστισμός, μποτιλιαρίσματα σε ορισμένα σημεία των γραμμών παραγωγής),

• χρονικού προγραμματισμού,

• ροής υλικών.

2.2. Δραστηριότητες Χωροταξικού Σχεδιασμού

Το έργο του χωροταξικού σχεδιασμού περιλαμβάνει γενικά τις δραστηριότητες που ακολουθούν και που εκτελούνται με την αντίστοιχη σειρά:

• Συγκέντρωση και ανάλυση βασικών δεδομένων, όπως πληροφορίες για τις πωλήσεις, τεχνικά σχέδια, κατασκευαστικές προδιαγραφές, χρονικό πρόγραμμα της παραγωγής, διαθέσιμος χώρος κ.λπ..

• Σχεδίαση του τρόπου με τον οποίο γίνεται η ροή των υλικών.

• Διαμόρφωση γενικού σχεδίου διαχείρισης υλικών, έτσι ώστε να ικανοποιούνται εκτός από τις αρχές ροής των υλικών (αυτοματοποίηση κ.λπ.), οι στόχοι αλλά και οι ανάγκες του συστήματος (κανόνες ασφαλείας, συνθήκες εργασίας, παραγωγή ορισμένου όγκου προϊόντων κ.λπ..).

• Σχεδίαση των θέσεων εργασίας, με τη βοήθεια των διαγραμμάτων δραστηριοτήτων που χρησιμοποιούνται στη Μελέτη Μεθόδου.

• Επιλογή ειδικού εξοπλισμού για τη διαχείριση υλικού. Ο ειδικός εξοπλισμός περιλαμβάνει συνήθως μεταφορικές ταινίες, ανυψωτικά μηχανήματα, κεκλιμένα επίπεδα, γερανογέφυρες κ.λπ..

• Προσδιορισμός των απαιτήσεων αποθηκευτικού χώρου, για τις πρώτες ύλες, τα ενδιάμεσα και τα τελικά προϊόντα.

• Συντονισμός ομάδων που εκτελούν σχετιζόμενες λειτουργίες.

• Σχεδίαση των θέσεων των παροχών. Οι διάφορες θέσεις εργασίας για να λειτουργήσουν χρειάζονται υποστήριξη από δίκτυα παροχών, με τα οποία συνδέονται σε συγκεκριμένα σημεία στο χώρο.

• Κατανομή του διαθέσιμου χώρου στις θέσεις εργασίας, καθώς και στις άλλες χρήσεις (διάδρομοι επικοινωνίας κ.λπ.).

• Εκπόνηση του κύριου χωροταξικού σχεδίου. Είναι η σύνθεση των παραπάνω επιμέρους σχεδίων και υπολογισμών και πραγματοποιείται με τη βοήθεια εικονικών προτύπων. Εκπονούνται τα διάφορα εναλλακτικά σχέδια και γίνεται η επιλογή του καταλληλότερου.

• Έλεγχος του σχεδίου από κατάλληλα πρόσωπα, στελέχη που κάνουν χρήσιμες υποδείξεις για την υλοποίησή του.

• Έγκριση σχεδίου, από το αρμόδιο όργανο. Διοίκηση Παραγωγής & Συστημάτων Υπηρεσιών 5

• Συμμετοχή στην εκπόνηση κατασκευαστικών σχεδίων.

• Παρακολούθηση των κατασκευών και της εγκατάστασης του χωροταξικού σχεδίου, με σκοπό την εξασφάλιση της εφαρμογής του χωροταξικού σχεδίου με συνέπεια.

• Παρακολούθηση της εφαρμογής του χωροταξικού σχεδίου. Έτσι, διαπιστώνεται η αποτελεσματικότητα του σχεδίου και τα σημεία στα οποία μπορούν να γίνουν βελτιώσεις [33].

Όλες οι τεχνικές χωροταξικού σχεδιασμού είτε αυτές είναι αυτοματοποιημένες με τη βοήθεια υπολογιστή (όπως CRAFT, ALDEP, CORELAP) είτε χωρίς υπολογιστή (όπως Systematic Layout Planning-SLP) χρησιμοποιούν ένα μέτρο αποτελεσματικότητας προκειμένου να αξιολογηθεί κάθε εναλλακτικό σενάριο χωροθέτησης και να επιλεγεί το καλύτερο.

2.3. Χωροταξικός Σχεδιασμός και Επαυξημένη Πραγματικότητα

Μια καλά σχεδιασμένη διάταξη παραγωγής μπορεί να μειώσει έως και 50% το λειτουργικό κόστος [34]. Παραδοσιακά, ο χωροταξικός σχεδιασμός πραγματοποιείται κατά το στάδιο σχεδιασμού της παραγωγικής μονάδας. Η αλγοριθμική επίλυση και η επίλυση με εργαλεία εικονικής πραγματικότητας είναι οι δύο ευρέως εφαρμοζόμενες προσεγγίσεις. Οι αλγοριθμικές προσεγγίσεις επικεντρώνονται στη μαθηματική αποτύπωση των χωροταξικών εναλλακτικών με τη χρήση διάφορων μοντέλων όπως το μοντέλο Quadratic Assignment Problem, το μοντέλο Mixed Integer Problem και την ανάπτυξη αποτελεσματικών αλγορίθμων για την επίλυση αυτών των μοντέλων, όπως GA (γενετικός αλγόριθμος), SA (simulated annealing) κ.α. Ωστόσο, λόγω της συνδυαστικής πολυπλοκότητας του προβλήματος του χωροταξικού σχεδιασμού, είναι σχεδόν αδύνατο να βρεθεί η καλύτερη λύση. Παρέχοντας πληροφορίες σε πραγματικό χρόνο για το πραγματικό περιβάλλον εργαλεία επαυξημένης πραγματικότητας μπορούν να προσφέρουν μια εφικτή λύση στο πρόβλημα του χωροταξικού σχεδιασμού. [32]

Στην παρούσα εργασία επιχειρείται ο ανασχεδιασμός των λειτουργιών στο χώρο στο υπάρχον σύστημα του εργαστηρίου κατεργασιών με στόχο τον βέλτιστο σχεδιασμό ενός κυττάρου κατεργασιών. Σκοπός είναι η αύξηση του όγκου παραγωγής ενός συγκεκριμένου τεμαχίου που επιλέχθηκε για τις ανάγκες της εργασίας. Για τον λόγο αυτό κρίνεται απαραίτητη η προσθήκη παραγωγικού δυναμικού αλλά και η αλλαγή της μεθόδου παραγωγής η οποία μέχρι τώρα ήταν ημί-αυτοματοποιημένη και πλέον επιχειρείται η πλήρης αυτοματοποίηση της. Ο χωροταξικός σχεδιασμός

αποσκοπούσε επίσης κυρίως στην εξάλειψη των καθυστερήσεων και του μη παραγωγικού χρόνου καθώς επίσης και στην βέλτιστη διαχείριση των αποθεμάτων.

Μια από τις τελευταίες αλλά ταυτόχρονα βασικότερες δραστηριότητες του χωροταξικού σχεδιασμού είναι η εκπόνηση των διάφορων εναλλακτικών σχεδίων προκειμένου να γίνει η επιλογή του καταλληλότερου.

Με την βοήθεια εργαλείων επαυξημένης πραγματικότητας η επιλογή της βέλτιστης λύσης χωροταξικού σχεδιασμού σε μια παραγωγική μονάδα, μπορεί να λάβει υπόψιν διαισθητικούς και εμπειρικούς παράγοντες οι οποίοι συνήθως δεν μπορούν να μοντελοποιηθούν εύκολα προκειμένου η επιλεχθείσα λύση να μην είναι μόνο βέλτιστη αλλά βέλτιστη και βολική.

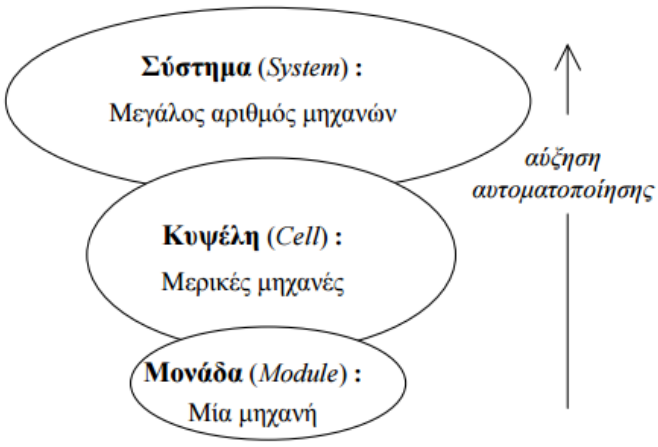
3. Ευέλικτο Σύστημα Κατεργασιών (Flexible Manufacturing System-FMS)

Η έννοια των Ευέλικτων Συστημάτων Κατεργασίας (FMS) εξελίχθηκε κατά τη διάρκεια της δεκαετίας του 1960, όταν τα ρομπότ, οι προγραμματιζόμενοι ελεγκτές, και οι αριθμητικά ελεγχόμενες εργαλειομηχανές (CNC) έφεραν ένα αριθμητικά ελεγχόμενο οικοσύστημα στο περιβάλλον του εργοστασίου.

Ένα ευέλικτο σύστημα κατεργασίας θεωρείται μια βελτιωμένη έκδοση του συστήματος DNC (Distributed Numerical Control) στο οποίο μια ομάδα από αριθμητικά ελεγχόμενες εργαλειομηχανές, οι οποίες αλληλοσυνδέονται με ένα κεντρικό σύστημα ελέγχου τροφοδοτείται από ένα σύστημα παροχής και μετακίνησης ακατέργαστων ή και κατεργασμένων τεμαχίων [35].

Ο στόχος ενός ευέλικτου συστήματος κατεργασίας είναι η μεγιστοποίηση, της χρήσης κάθε εργαλειομηχανής μεμονωμένα. Επιτρέπει πλήθος γεωμετρικών αλλαγών στη σχεδίαση των προς κατεργασία κομματιών, την προσθήκη ή την αφαίρεση λειτουργικών χαρακτηριστικών του τεμαχίου και τη ρύθμιση του ρυθμού παραγωγής [37].

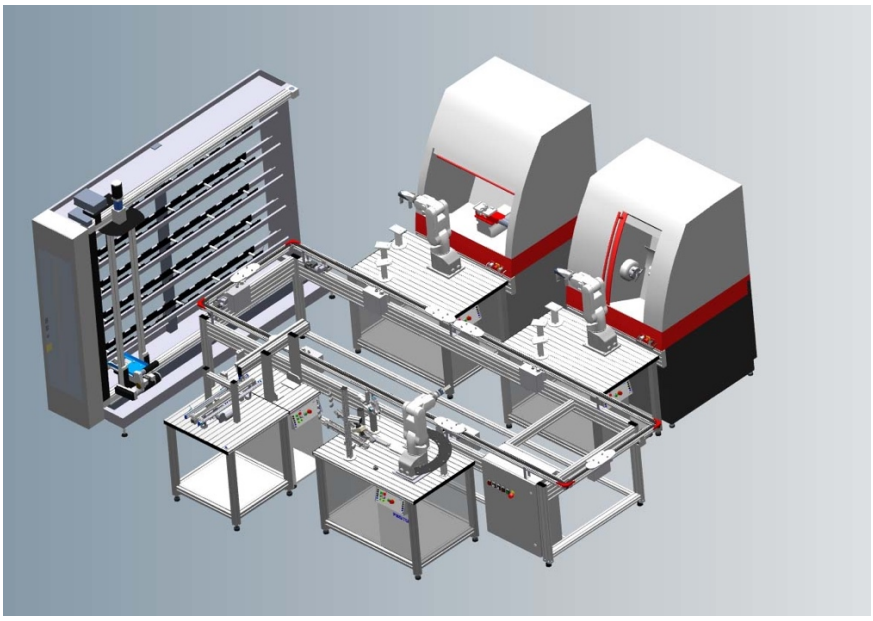
Το βασικό χαρακτηριστικό των Ευέλικτων Συστημάτων Κατεργασιών είναι η αυτοματοποίηση. Το "Σύστημα" αποτελείται από πολλές κυψέλες (Cells), και η κυψέλη από πολλές μονάδες (Modules). Η καρδιά της μονάδας είναι π.χ. ένα Κέντρο Τόρνευσης ή ένα Κέντρο Κατεργασίας με ενσωματωμένο μικροϋπολογιστή που περιλαμβάνει συσκευή αυτόματης αλλαγής εργαλείων.



Εικόνα 5: Δομή ενός ευέλικτου συστήματος κατεργασίας

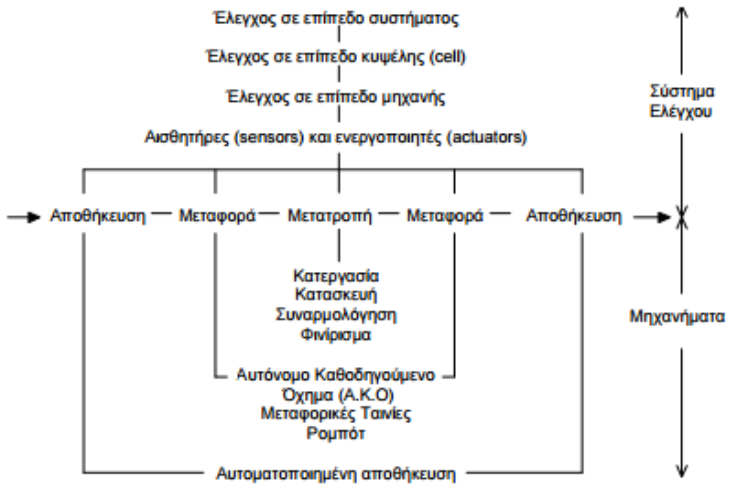
Το στοιχείο που πλαισιώνει την κάθε μονάδα σε ένα ευέλικτο σύστημα κατεργασίας είναι το αυτόματο σύστημα φόρτωσης/εκφόρτωσης και αποθήκευσης των προς κατεργασία αλλά και των έτοιμων κομματιών. Η φόρτωση και η εκφόρτωση των προς κατεργασία εξαρτημάτων γίνεται είτε με μεταφορική ταινία είτε με ρομποτικό βραχίονα.

Μετά την ολοκλήρωση της κατεργασίας των κομματιών, ακολουθεί η συσκευασία / παλετοποίηση/ αποθήκευση τους, η οποία ενώ παλαιότερα γινόταν με την βοήθεια Κλαρκ υπό την καθοδήγηση του χειριστή του, πλέον η διαδικασία αυτή σε πολλά συστήματα FMS έχει αυτοματοποιηθεί εξίσου, δεδομένου ότι εισήχθησαν στο σύστημα τα Συστήματα Αυτόματης Αποθήκευσης και Ανάκτησης (Automated storage and retrieval system – ASRS) τα οποία αφού παραλάβουν το έτοιμο κομμάτι ή παλέτα είναι υπεύθυνα για την αποθήκευση τους και τα Αυτομάτως οδηγούμενα οχήματα (Automatic Guided Vehicles - AGVs) τα οποία είναι υπεύθυνα για την μεταφορά των κομματιών μεταξύ των σταθμών κατεργασίας και αποθήκευσης.



Εικόνα 6: Παράδειγμα ευέλικτου συστήματος κατεργασίας

Σε γενικές γραμμές, ένα σύστημα FMS αποτελείται από κέντρα κατεργασίας, κέντρα αλλαγής εργαλείων, συστήματα φόρτωσης/εκφόρτωσης παλετών, εργαλεία κοπής, παλέτες, εξαρτήματα, συστήματα αποθήκευσης, μηχανές προσωρινής αποθήκευσης, και συστήματα μετακίνησης υλικών.

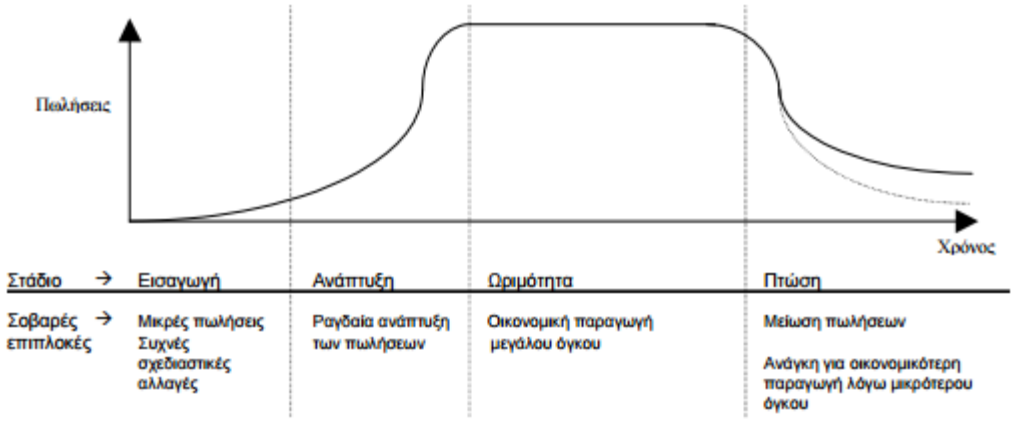


Εικόνα 7: Εποπτική λειτουργία ενός ευέλικτου συστήματος κατεργασίας

Η ευέλικτη αυτοματοποίηση βρίσκει εφαρμογή στους τομείς της επεξεργασίας, της κατασκευής, της συναρμολόγησης, του φινιρίσματος, της αποθήκευσης, της διαχείρισης και της μεταφοράς πρώτων υλών. Τα ευέλικτα συστήματα αυτοματοποίησης περιλαμβάνουν εξελιγμένες μηχανές παραγωγής και επιθεώρησης (ποιοτικού ελέγχου), συστήματα ελέγχου με υπολογιστές, συστήματα επικοινωνίας καθώς και συστήματα διαχείρισης υλικών.

3.1. Ευελιξία Συστήματος Παραγωγής

Ευελιξία καλούμε την ικανότητα ενός συστήματος κατεργασίας να προσαρμόζεται στις απαιτήσεις ενός συγκεκριμένου προϊόντος. Στο παρακάτω διάγραμμα φαίνονται τα τέσσερα στάδια του κύκλου ζωής ενός προϊόντος: Εισαγωγή, Ανάπτυξη, Ωρίμανση, Πτώση. Σε κάθε στάδιο του κύκλου ζωής του προϊόντος θα πρέπει να καθορίσουμε τους διάφορους τύπους ευελιξίας που πρέπει να έχει το σύστημα κατεργασίας.



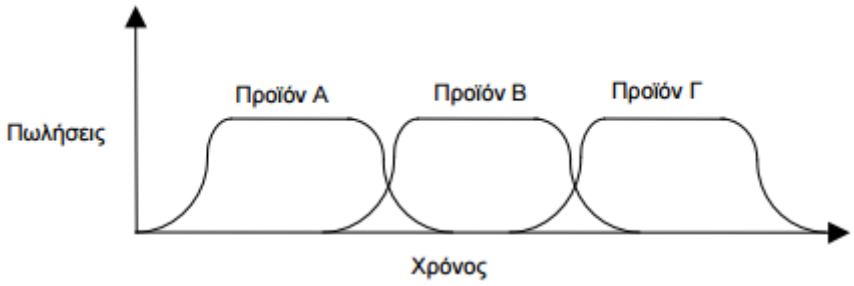
Εικόνα 8: Κύκλος ζωής προϊόντος και επίδραση στην ευελιξία της παραγωγής

3.2. Ευελιξία Ποικιλίας Προϊόντων

Η ευελιξία ποικιλίας επικεντρώνεται:

- Στον αριθμό των διαφορετικών προϊόντων που παράγονται από το σύστημα σε οποιαδήποτε χρονική στιγμή.
- Στη δυνατότητα του συστήματος κατεργασίας να εναλλάσσεται μεταξύ δύο προϊόντων γρήγορα και με μικρό κόστος.
- Στις διάφορες παραλλαγές μεγέθους, σχήματος, πρώτων υλών, απαιτούμενων κατεργασιών κτλ.

Είναι σημαντικό, τα συστήματα παραγωγής να αντεπεξέρχονται δυναμικά στην εισαγωγή νέων προϊόντων και στην απόσυρση παλαιοτέρων με τις λιγότερες δυνατές διαταραχές. Η ικανότητα ταυτόχρονης διεκπεραίωσης πολλών εργασιών αναφέρεται και ως “ευελιξία έργου”.



Εικόνα 9: Επικαλυπτόμενοι κύκλοι ζωής προϊόντος

3.3. Ευελιξία Μεθόδων Παραγωγής

Η ευελιξία μεθόδων παραγωγής είναι μέτρο του πόσο εύκολα μπορεί ένα σύστημα κατεργασίας να ξεπερνάει τις δυσκολίες που προκαλούνται εκ των έσω. Η ικανότητα κατασκευής εξαρτημάτων και συναρμολογήσεων με διαφορετικούς τρόπους αποτελεί απαραίτητη προϋπόθεση (διευκολύνει τον προγραμματισμό, ελαχιστοποιεί τις επιδράσεις από τις βλάβες του συστήματος).

Έτσι, σύμφωνα με τον ορισμό μία μονάδα του FMS αποτελείται από μια αριθμητικά ελεγχόμενη εργαλειομηχανή με ενσωματωμένο μικροϋπολογιστή. Περιλαμβάνει επίσης συσκευή αυτόματης αλλαγής εργαλείων, αποθήκη εργαλείων και ενδεχομένως αυτόματο σύστημα φόρτωσης / εκφόρτωσης και αποθήκευσης των προς κατεργασία κομματιών (μεταφορική ταινία, ρομποτικό βραχίονα, κλπ).

Το βασικό δεδομένο για το σχεδιασμό ενός FMS, όπως και ενός κυττάρου παραγωγής, είναι η οικογένεια των προϊόντων που πρέπει να παράγει, καθώς και ο αντίστοιχος ρυθμός (όγκος) παραγωγής. Η οικογένεια των προϊόντων καθορίζει σε μεγάλο βαθμό τον τύπο του βασικού εξοπλισμού και τα χαρακτηριστικά του (ακρίβεια), καθώς και την χωροταξική διάταξη. Ο όγκος παραγωγής καθορίζει τον αριθμό των μηχανών αλλά και τα χαρακτηριστικά του συστήματος χειρισμού και μεταφοράς των υλικών [36].

Η χωροταξική διάταξη συνδέεται κυρίως με την ποικιλία των διαδρομών ροών που παρατηρείται. Εάν ένα σύστημα έχει μικρό ποσοστό ανάστροφης ροής τότε προτιμάται η γραμμική διάταξη. Αντίθετα, εάν υπάρχει μεγάλη ποικιλία διαδικασιών συνιστάται μια κυκλικού τύπου διάταξη. Οι διατάξεις τύπου σκάλας ή ανοικτού τύπου ενδείκνυνται για σχετικά μεγάλη ποικιλία προϊόντων και μεγάλα ποσοστά ανάστροφης ροής και παρακάμψεων.

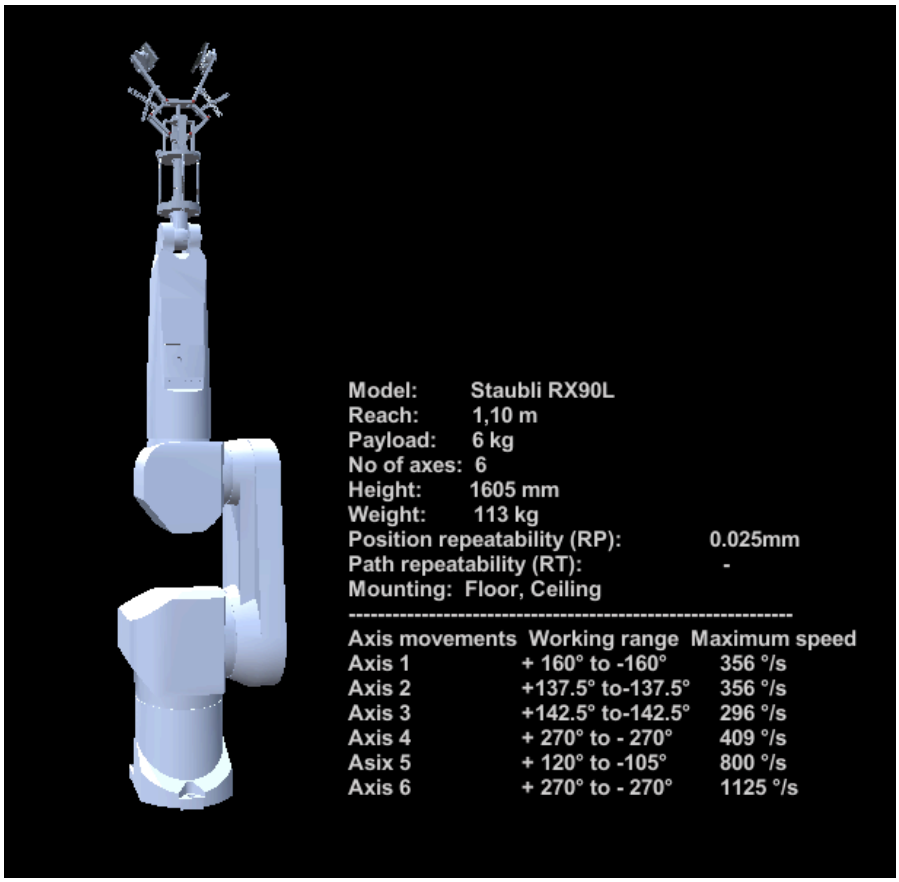
Σημαντική σχεδιαστική παράμετρος αποτελεί και ο απαιτούμενος βαθμός εκμετάλλευσης (ποσοστό χρησιμοποίησης) του εξοπλισμού, ο οποίος είναι συνήθως αντιστρόφως ανάλογος του αριθμού των μονάδων υπό επεξεργασία. Εάν οι μονάδες υπό επεξεργασία είναι σχετικά λίγες, οι μηχανές παραμένουν για σχετικά μεγάλα χρονικά διαστήματα ανενεργές, καθώς δεν έχουν υλικό προς επεξεργασία. Μεγάλος βαθμός εκμετάλλευσης του εξοπλισμού, αντίθετα, είναι δυνατός μέσω υψηλού αριθμού μονάδων υπό επεξεργασία που εξασφαλίζει σταθερή τροφοδοσία των μηχανών. Από την άλλη πλευρά, μεγάλος αριθμός μονάδων υπό επεξεργασία αυξάνει το δεσμευμένο σε υλικά και εργασία κεφάλαιο, καθώς και τον χρόνο παράδοσης/εκτέλεσης μιας παραγγελίας. Βέλτιστο, συνεπώς θεωρείται το επίπεδο που εξασφαλίζει υψηλό βαθμό εκμετάλλευσης χωρίς να επιβαρύνει υπέρογκα το κόστος και να αυξάνει σημαντικά τους χρόνους παράδοσης.

Στα πλαίσια της παρούσας εργασίας επιλέχθηκε η δημιουργία μιας ευέλικτης μονάδας κατεργασίας σε γραμμική διάταξη, καρδιά της οποίας θα αποτελέσει το κέντρο τόρνευσης EMCO COMPACT 5 (με δυνατότητα κατεργασίας κομματιών αξονοσυμμετρικής μορφής) καθώς και ο ρομποτικός βραχίονας 6 βαθμών ελευθερίας Staubli RX90L του εργαστηρίου, ενώ θα επιχειρηθεί η υπέρθεση των στοιχείων που λείπουν από την μονάδα, με την βοήθεια επαυξημένης πραγματικότητας, προκειμένου αρχικά να γίνει η αποτύπωση των πιθανών χωροταξικών εναλλακτικών καθώς και ο καθορισμός της βέλτιστης τελικής χωροθέτησης.

4. Το Εργαστήριο Τεχνολογίας των Κατεργασιών του ΕΜΠ

4.1. Το Ρομπότ Staubli RX90L

Το ρομπότ του εργαστηρίου Κατεργασιών των Υλικών του ΕΜΠ είναι το Staubli RX90L Paint Robot. Η χρήση του ρομποτικού βραχίονα Staubli RX90L είναι ευρεία στις βιομηχανικές εφαρμογές. Εργασίες όπως κατασκευή ποδηλατών, αγροτικού εξοπλισμού, αεροναυπηγική, οπτικά κ.α. είναι από τις πιο χαρακτηριστικές για την χρήση του συγκεκριμένου ρομπότ [38]. Ο βραχίονας αποτελείται από 6 μέλη τα οποία συνδέονται ανά δύο με 6 περιστροφικές αρθρώσεις.



Εικόνα 10: Βιομηχανικό ρομπότ Staubli RX90L και τα τεχνικά του χαρακτηριστικά (SetupState/Scene0)

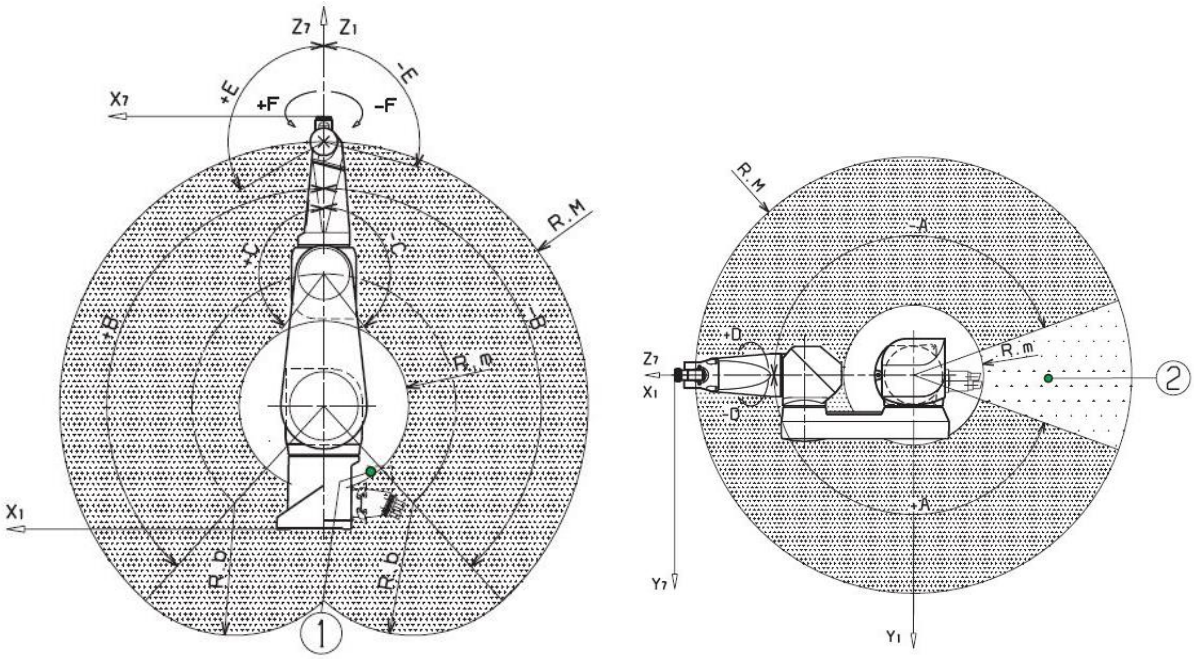
Τα μέλη ονομάζονται αλλιώς σύνδεσμοι και στην παρακάτω εικόνα συμβολίζονται με κεφαλαία γράμματα. Επειδή το ρομπότ ανήκει στην κατηγορία των ανθρωπομορφικών βραχιόνων, οι σύνδεσμοι έχουν ονόματα εμπνευσμένα από τα μέρη του ανθρώπινου χεριού:

Axis 1	A	Base	0
Axis 2	B	Shoulder	-90
Axis 3	C	Arm	90
Axis 4	D	Elbow	0
Axis 5	E	Forearm	0
Axis 6	F	Wrist	0

Πίνακας: Συμβολισμός, όνομα και περιστροφή κάθε άρθρωσης σε θέση DoReady

Στην Εικόνα 10 φαίνεται ο ρομποτικός βραχίονας Staubli RX90L με την πνευματική αρπάγη του εργαστηρίου κατεργασιών προσαρμοσμένη στην τελευταία του άρθρωση. Επίσης αναγράφονται όλα τα τεχνικά και λειτουργικά χαρακτηριστικά του όπως ύψος, εμβέλεια εργασίας και συνήθη σημεία τοποθέτησης ενώ διακρίνονται και τα εύρη γωνιών και ταχυτήτων για την κάθε άρθρωση ξεχωριστά. Το ρομπότ στην παραπάνω εικόνα βρίσκεται στην θέση Do Ready δηλαδή στην θέση στην οποία η εταιρία παραγωγής προτείνει να βρίσκεται ο βραχίονας όταν είναι σε αδράνεια. Οι γωνίες των αρθρώσεων σε αυτή την περίπτωση φαίνονται στον παραπάνω πίνακα.

Τα όρια των αρθρώσεων καθορίζουν τον χώρο εργασίας του βραχίονα. Ως χώρος εργασίας ορίζεται ο χώρος στον οποίο μπορεί να βρεθεί το τελικό σημείο δράσης με οποιονδήποτε προσανατολισμό



Εικόνα 11: Χώρος εργασίας του ρομπότ Staubli RX90L

Ο προγραμματισμός του ρομπότ μπορεί να γίνει με δυο τρόπους:

OnLine προγραμματισμός όπου ο χρήστης χρησιμοποιεί το χειριστηρίο διδασκαλίας (teach pendant) προκειμένου να κατευθύνει το τελικό σημείο δράσης του ρομπότ ακριβώς στο σημείο του χώρου

εργασίας που επιθυμεί ελέγχοντας την περιστροφή κάθε μιας άρθρωσης ξεχωριστά ή για να εισάγει την επιθυμητή γωνία κάθε άρθρωσης στην αντίστοιχη μεταβλητή.

OffLine προγραμματισμός όπου ο χρήστης προγραμματίζει μέσω Η/Υ χρησιμοποιώντας την γλώσσα προγραμματισμού V+ της Adept Technology την οποία χρησιμοποιεί το συγκεκριμένο ρομπότ. Στον OffLine προγραμματισμό ο χρήστης είναι σε θέση να καθορίσει όλα τα σημεία από τα οποία επιθυμεί να περάσει το ρομπότ προκειμένου να ακολουθήσει πλήρως την επιθυμητή τροχιά.

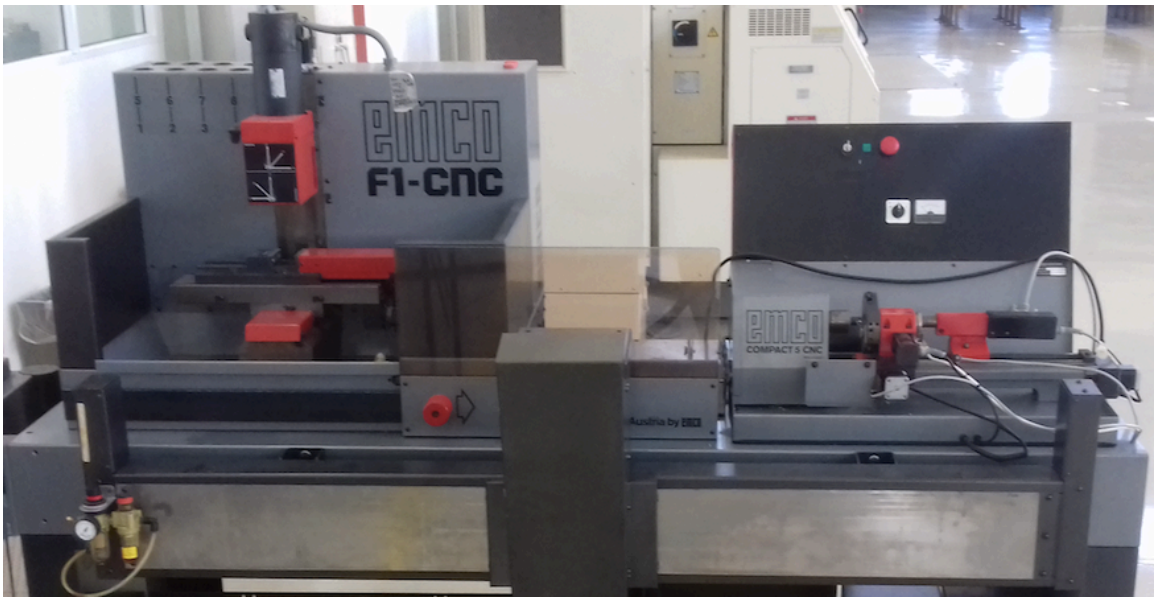
Επειδή ο OffLine προγραμματισμός δεν παρέχει την δυνατότητα εποπτείας της ορθότητας των σημείων που εισάγει ο χειριστής, είναι προτιμότερος ένας συνδυασμός OnLine και OffLine προγραμματισμού. Με αυτόν τον τρόπο ο χρήστης μπορεί να καταγράψει τις γωνίες των αρθρώσεων σε όλα τα επιθυμητά σημεία με OnLine προγραμματισμό και έπειτα να εισάγει αυτά τα δεδομένα στο πρόγραμμα που θα συντάξει κατά τον OffLine προγραμματισμό. Έτσι επιτυγχάνεται ακριβής πρόβλεψη της τροχιά που θα ακολουθήσει το ρομπότ ενώ ταυτόχρονα μηδενίζεται η πιθανότητα σύγκρουσης του ρομπότ σε περίπτωση λάθους.

Ο προγραμματισμός του ρομπότ για τις ανάγκες της παρούσας εργασίας ακολούθησε την παραπάνω υβριδική λογική και το πρόγραμμα το οποίο προέκυψε παρατίθεται στο Παράρτημα Β.

4.2. Η Εργαλειομηχανή EMCO

Στο Εργαστήριο Τεχνολογίας των Κατεργασιών υπάρχουν επίσης οι εργαλειομηχανές EMCO COMPACT 5 CNC και EMCO F1-CNC. Η EMCO MAIER & CO είναι η εταιρία παραγωγής των παραπάνω δυο μηχανών CNC με εμπειρία χρόνων η οποία δραστηριοποιείται και στη τεχνική εκπαίδευση παγκοσμίως. Μετά την κατασκευή της EMCO COMPACT 5 CNC η οποία χρησιμοποιήθηκε παγκοσμίως σε μεγάλο βαθμό συνέχισε στην κατασκευή της EMCO F1-CNC. Η πρώτη αποτελεί τόρνο δύο αξόνων ενώ η δεύτερη κέντρο κατεργασιών τριών αξόνων. Στην παρούσα εφαρμογή χρησιμοποιούμε μόνο το κέντρο τόρνευσης EMCO COMPACT 5 CNC το οποίο θα αποτελέσει την καρδιά του κυττάρου κατεργασιών που σχεδιάζουμε.

Το κέντρο τόρνευσης EMCO COMPACT 5 CNC αποτελεί ένα κέντρο τόρνευσης, δύο αξόνων, Z και X. Η εν λόγω εργαλειομηχανή προγραμματίζεται και εκτελεί τις λειτουργίες της με χρήση λογισμικού linux CNC, που είναι εγκατεστημένο σε ηλεκτρονικό υπολογιστή που τη συνοδεύει. Για την κίνηση των αξόνων χρησιμοποιούνται βηματικοί κινητήρες και γρανάζια μετάδοσης κίνησης. Για την κοπή διαφόρων τεμαχίων, η εργαλειομηχανή διαθέτει τρία διαφορετικά εργαλεία τα οποία συγκρατούνται σε ένα εργαλειοφορείο ενώ για την συγκράτηση του υπό κατεργασία τεμαχίου η εργαλειομηχανή διαθέτει τις κατάλληλες διατάξεις συγκράτησης (τσόκ –κεντροφορέας)[39].



Εικόνα 12: Οι Εργαλειομηχανές EMCO F1-CNC (αριστερά) και EMCO COMPACT 5 CNC (δεξιά).

5. Unity 3D

5.1. Περιγραφή

Για την ανάπτυξη εφαρμογών Εικονικής Πραγματικότητας (VR) και πολύ περισσότερο Επαυξημένης Πραγματικότητας (AR) το λογισμικό το οποίο χρησιμοποιείται ολοένα και περισσότερο έναντι του ανταγωνισμού του, είναι το λογισμικό Unity 3D, το οποίο αρχικά δημιουργήθηκε ως πλατφόρμα πλατφόρμα ανάπτυξης βιντεοπαιχνιδιών.

Αναπτύχθηκε από την εταιρία Unity Technologies, με σκοπό την ανάπτυξη παιχνιδιών για ηλεκτρονικούς υπολογιστές, κονσόλες, κινητές συσκευές καθώς και διαδικτυακές εφαρμογές. Η πρώτη έκδοση του Unity πραγματοποιήθηκε στις 8 Ιουνίου του 2005 και υποστήριζε αποκλειστικά Mac OS X. Έκτοτε, η λειτουργικότητά του επεκτάθηκε και πλέον η χρήση του στοχεύει σε παραπάνω από 27 πλατφόρμες [40].

Το Unity graphical editor δίνει την δυνατότητα στον χρήστη να σχεδιάσει με ευκολία, να οργανώσει τα project του, να εισάγει εικονικά μοντέλα κατασκευασμένα σε άλλο λογισμικό (υπό μορφή fbx και obj), να γράψει κώδικα, να προσθέσει εφέ ήχου ή φωτισμού και πολλά άλλα. Όλα τα παραπάνω χαρακτηριστικά και η ευκολία με την οποία αυτά υλοποιούνται στο περιβάλλον του Unity, μειώνουν κατά πολύ τον χρόνο ανάπτυξης παιχνιδιών και γενικότερα εφαρμογών εικονικού περιεχομένου. Επίσης, διατίθεται ένα πολύ λεπτομερές documentation και μια μεγάλη κοινότητα προγραμματιστών που αλληλεπιδρά για την επίλυση προβλημάτων των μελών της. Το Unity 3D φαίνεται να κερδίζει έδαφος έναντι του ανταγωνισμού και από στρατηγικής άποψης καθώς φαίνεται να κυριαρχεί στον τομέα των εφαρμογών επαυξημένης πραγματικότητας υποστηρίζοντας plug-ins όπως το ARKit της Apple και το Vuforia SDK.

5.2. Λειτουργία

Το User Interface του Unity απαρτίζεται από πέντε πολύ βασικά windows ή tabs η χωροθέτηση των οποίων μπορεί να επιλεγεί από τον χρήστη βάσει των προσωπικών του προτιμήσεων. Έτσι, το Unity αποτελείται από τα:

- Toolbar
- Project Window
- Hierarchy Window
- Inspector Window
- SceneView

Στο Toolbar, που στην default διάταξη βρίσκεται στην κορυφή του UI του Unity, ο χρήστης μπορεί να επιλέξει τον τρόπο με τον οποίο θα διαχειριστεί τα αντικείμενα στην σκηνή (SceneView) μετατοπίζοντας, περιστρέφοντας, προσεγγίζοντας τα κλπ, το σύστημα συντεταγμένων βάσει του

οποίου εκφράζονται τα Game Objects στην σκηνή αλλά και να ελέγξει της έναρξη ή τη λήξη του παιχνιδιού το οποίο αναπτύσσει.

Το Project Window είναι το σημείο στο οποίο όλα τα αντικείμενα της εφαρμογής ή του παιχνιδιού μας αποθηκεύονται πριν τα χρησιμοποιήσουμε στην σκηνή μας (SceneView). 3D μοντέλα, αρχεία ήχου, κώδικες (scripts), plugins, εικόνες και πολλά άλλα αποθηκεύονται Project Window και γίνονται με αυτόν τον τρόπο κομμάτια του project μας.

Το Hierarchy Window είναι ο χώρος που κάθε αντικείμενο του Project, το οποίο θέλουμε να χρησιμοποιηθεί στην σκηνή, αποκτά οντότητα. Εκεί υπάρχει μια λίστα με όλα τα ενεργά ή και όχι Game Objects της σκηνής. Κάθε Game Object το οποίο έχει στην λίστα του ένα άλλο υπό-αντικείμενο ονομάζεται γονέας και το υπό-αντικείμενο ονομάζεται παιδί. Αυτή η κάθετη ιεράρχηση είναι πολύ σημαντική καθώς μπορούμε βάσει της κληρονομικότητας που προκύπτει μεταξύ γονέα και τέκνου να μεταφέρουμε όλες τις ιδιότητες (πχ ένα script) των γονέων στα τέκνα.

Όλα αυτά όμως εκτυλίσσονται σε μια εικονική σκηνή την SceneView η οποία αποδίδει υψηλής ποιότητας γραφικά στα Game Objects τα οποία βρίσκονται σε αυτήν και στην οποία ο χρήστης μπορεί να περιηγηθεί με ευκολία χρησιμοποιώντας τα διαθέσιμα Buttons του Toolbar που αναφέραμε παραπάνω.

Τέλος, στο Inspector Window ο χρήστης είναι σε θέση να προσθέσει να αφαιρέσει ή να απενεργοποιήσει ιδιότητες στο κάθε επιλεγμένο αντικείμενο της σκηνής καθώς και να ρυθμίσει κάποιες δημόσιες παραμέτρους των ιδιοτήτων αυτών.

5.3. C#

Όσον αφορά την σύνταξη κώδικα στο Unity, υπάρχουν δυο διαθέσιμες επιλογές, C# και Java. Επιλέχθηκε στα πλαίσια της συγκεκριμένης εργασίας η χρήση της C# η οποία είναι μια πιο εξελιγμένη, επίσης αντικειμενοστραφής γλώσσα προγραμματισμού αλλά με μεγαλύτερη κοινότητα χρηστών της σε περιβάλλον Unity. Η καμπύλη εκμάθησης της δείχνει ότι είναι δομημένη με σχετικά απλό τρόπο και ότι είναι πιο κοντά στον χρήστη παρότι συνδυάζει χαρακτηριστικά “παλαιότερων” γλωσσών όπως οι C, C++ και Java.

Η C# παρέχει πλήρη πρόσβαση στη βασική βιβλιοθήκη του .NET Framework αλλά και εύκολη πρόσβαση στις βιβλιοθήκες του Unity.

5.4. Εγκατάσταση Unity 3D

Κατά την εγκατάσταση του Unity, έκδοση: 2017.2.0f3, επιλέχθηκαν από την Unity component selection list, εκτός των προεπιλεγμένων components, το “iOS Built Support” component το οποίο μας δίνει την δυνατότητα ανάπτυξης εφαρμογών για iPhone και iTab (επίσης δίνεται η δυνατότητα εγκατάστασης του Vuforia, απευθείας στο περιβάλλον του Unity επιλέγοντας “Vuforia Augmented Reality Support” component το οποίο στην παρούσα εργασία εγκαταστάθηκε προεραιτικά για την περίπτωση χρήσης κάποιου image target προκειμένου να γίνει ακριβέστερος προσδιορισμός του σημείου στο οποίο θα αρχικοποιήσουμε τα εικονικά μας μοντέλα μέσα στην πραγματική μας σκηνή-εν τέλει αυτο πραγματοποιήθηκε προγραμματιστικά χρησιμοποιόντας μόνο το ARKit).

Αφού ολοκληρωθεί η εγκατάσταση του Unity, ακολουθώντας τη διαδρομή Files/Build Settings, επιλέγουμε iOS και μετά Switch Platform προκειμένου να γίνει deployment για iOS.

6. Apple ARKit

6.1. Περιγραφή

Η Διεπαφή προγραμματισμού εφαρμογών (API), "ARKit" επιτρέπει σε προγραμματιστές να αναπτύξουν εφαρμογές επαυξημένης πραγματικότητας, χρησιμοποιώντας τις κάμερες, τον επεξεργαστή, τον επεξεργαστή γραφικών και τους αισθητήρες κίνησης της συσκευής. Το ARKit είναι διαθέσιμο μόνο σε συσκευές με επεξεργαστή τουλάχιστον Apple A9 (iPhone 6S και μεταγενέστερες συσκευές). Αυτός ο περιορισμός οφείλεται στο ότι οι συγκεκριμένοι επεξεργαστές προσφέρουν επιδόσεις οι οποίες επιτρέπουν γρήγορη αναγνώριση σκηνών και επιτρέπουν στον χρήστη να κατασκευάσει λεπτομερές εικονικό υλικό πάνω σε αντικείμενα της πραγματικότητας [41].

6.2. Προ-απαιτούμενα

Η ανάπτυξη εφαρμογών επαυξημένης πραγματικότητας με το ARKit είναι εφικτή μόνο για τις εκδόσεις των λογισμικών που αναγράφονται παρακάτω ή μεταγενέστερων αυτών και για συσκευές Apple που φέρουν επεξεργαστή κλάσης τουλάχιστον A9. Έτσι απαιτούνται τουλάχιστον:

IOS 11 (ή νεότερη έκδοση) - Λειτουργικό Σύστημα iPhone

Xcode 9 (ή νεότερη έκδοση) -Πλατφόρμα Ανάπτυξης Εφαρμογών για iOS

Unity 3D (τρέχουσα έκδοση) -Λογισμικό Ανάπτυξης Βιντεοπαιχνιδιών

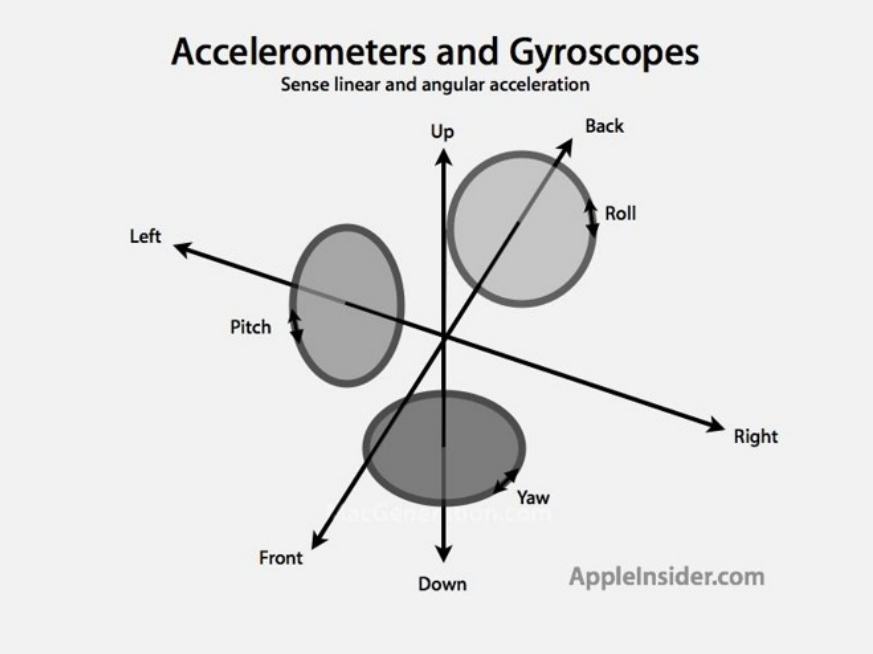
Unity ArKit Plugin-Πρόσθετο ArKit για Unity

6.3. Visual Inertial Odometry (VIO)

Το ARKit χρησιμοποιεί Οπτική Αδρανειακή Οδομετρία (Visual Inertial Odometry-VIO) για την ακριβή παρακολούθηση του κόσμου γύρω του. Το VIO συγχρονίζει τα δεδομένα της κάμερας με τα δεδομένα των αισθητήρων κίνησης που διαθέτει (επιταχυνσιόμετρο και γυροσκόπιο). Αυτές οι δύο είσοδοι επιτρέπουν στη συσκευή να αντιλαμβάνεται πώς κινείται μέσα σε ένα δωμάτιο με υψηλό βαθμό ακρίβειας και χωρίς επιπλέον βαθμονόμηση.

Το VIO αναλύει τα δεδομένα της κάμερας ("οπτικά") για να εντοπίσει τα ορόσημα που μπορεί να χρησιμοποιήσει για να μετρήσει ("οδομετρία") τον τρόπο με τον οποίο η συσκευή κινείται στο χώρο σε σχέση με τα ορόσημα που βλέπει. Τα δεδομένα των αισθητήρων κίνησης ("αδρανειακά") χρησιμοποιούνται για να συμπληρώσουν τα κενά παρέχοντας συμπληρωματικές πληροφορίες που η συσκευή μπορεί να συγκρίνει με αυτό που βλέπει η κάμερα της, για να καταλάβει καλύτερα πώς κινείται στο χώρο.

Ουσιαστικά, το VIO επιτρέπει στο σύστημα να δημιουργεί κινούμενα 3D γραφικά που μπορούν να απεικονίσουν ζωντανά με "6 βαθμούς ελευθερίας", ακολουθώντας τις σύνθετες κινήσεις της συσκευής στους 6 άξονες: πάνω / κάτω, πίσω / εμπρός και στις αντίστοιχες περιστροφές.



Εικόνα 13: Επιταχυνσιόμετρο και γυροσκόπιο είναι οι αισθητήρες στους οποίους βασίζεται η τεχνολογία Visual Inertial Odometry για την ανίχνευση ευθείας και γωνιακής επιτάχυνσης

Σε συνδυασμό με τη κατανόηση της σκηνής, η κάμερα μπορεί να αναγνωρίσει οπτικά ορόσημα του πραγματικού κόσμου για να τα ορίσει ως οριζόντιες επιφάνειες τις οποίες θα χρησιμοποιήσει ως βάση για την υπέρθεση των 3D εικονικών μοντέλων πάνω σε αυτές. Επιπλέον, η κάμερα χρησιμοποιείται επίσης για την εκτίμηση του επιπέδου φωτισμού. Με αυτόν τον τρόπο επιτυγχάνεται η προσαρμογή του φωτισμού και των σκιών. Το αποτέλεσμα είναι ένα φωτορεαλιστικό 3D μοντέλο, που αποδίδεται ως υπέρθεση πάνω από το βίντεο του πραγματικού κόσμου που καταγράφεται από την κάμερα, το οποίο μπορεί να προβληθεί ελεύθερα από οποιαδήποτε γωνία, απλά με κλίση και περιστροφή της συσκευής.

Το VIO επιτρέπει στις εφαρμογές ARKit να εντοπίζουν επιφάνειες όπως τοίχους, δάπεδα και τραπέζια και στη συνέχεια να τοποθετούν εικονικά αντικείμενα σε αυτές, επιτρέποντας στον χρήστη να βλέπει τον πραγματικό κόσμο επαυξημένο με γραφικά που είναι ‘’χτισμένα’’ σε αυτές τις επιφάνειες και στη συνέχεια να τα εξερευνά με τον ίδιο τρόπο που θα εξέταζε αντικείμενα στον πραγματικό κόσμο [42].

7. Ανάπτυξη της Εφαρμογής ARobot_Layout

7.1. Στόχος της Εφαρμογής

Η πολυπλοκότητα του προβλήματος εύρεσης της βέλτιστης χωροταξικής διάταξης του εξοπλισμού ενός εργοστασίου επιτάσσει την αναζήτηση μεθόδων οι οποίες να λαμβάνουν υπόψιν τα ιδιαίτερα χαρακτηριστικά κάθε περίπτωσης, τους παράγοντες άνεσης και ασφάλειας αλλά και τις συνήθειες και την εμπειρία του προσωπικού.

Στην παρούσα εργασία επιχειρείται η ανάπτυξη μιας iOS εφαρμογής επαυξημένης πραγματικότητας με στόχο:

-Τον σχεδιασμό ενός κυττάρου κατεργασιών στο Εργαστήριο Τεχνολογίας των Κατεργασιών του ΕΜΠ

-Την αξιολόγηση της διάταξης που επιλέχθηκε, χρησιμοποιώντας εργαλεία επαυξημένης πραγματικότητας

-Την προσομοίωση της παραγωγικής διαδικασίας

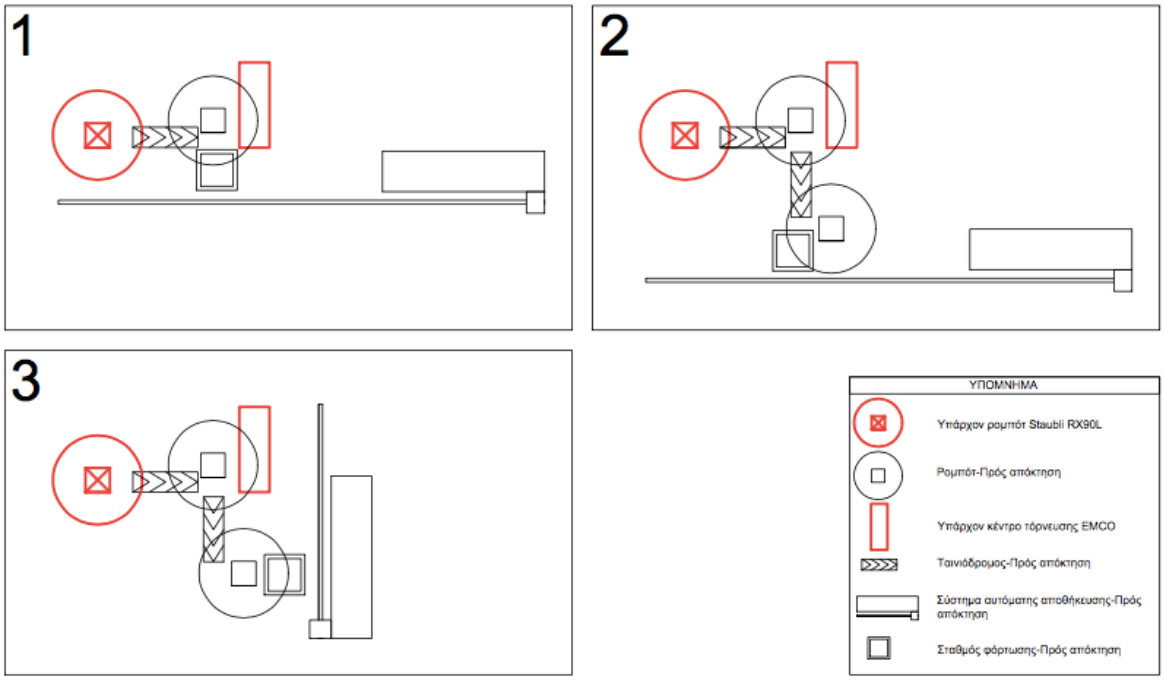
-Την συνεργασία εικονικού και πραγματικού εξοπλισμού στα πλαίσια της προσομοίωσης

7.2. Σχεδιασμός Ευέλικτου Συστήματος Κατεργασιών

Στα πλαίσια της παρούσας εργασίας αποφασίστηκε η εξέταση κάποιων εναλλακτικών χωροθετήσεων του εξοπλισμού που υπάρχει αλλά και που θα πρέπει να εγκατασταθεί στο Εργαστήριο Τεχνολογίας των Κατεργασιών προκειμένου να σχηματισθεί μια ευέλικτη μονάδα (κύτταρο) κατεργασιών ικανή να μεταποιεί αξονοσυμμετρικά τεμάχια ερταλόν.

Κριτήριο της επιλογής και χωροθέτησης του εξοπλισμού ήταν η ταχύτητα κατεργασίας και αποθήκευσης των τεμαχίων οπότε οι πιθανές εναλλακτικές δεν έλαβαν υπόψιν το κόστος του εξοπλισμού. Η “καρδιά” της υπό σχεδιασμό μονάδας κατεργασίας είναι το κέντρο τόρνευσης EMCO COMPACT 5 με το ρομπότ Staubli RX90L του εργαστηρίου να έχει υποστηρικτικό ρόλο. Αξίζει να σημειωθεί ότι οι δυο παραπάνω διατάξεις(εργαλειομηχανή EMCO και ρομπότ) ανήκουν στον υπάρχοντα εξοπλισμό του εργαστηρίου και οι θέσεις τους θα παραμείνουν αμετάβλητες στην ευέλικτη μονάδα κατεργασιών που θα προκύψει. Οι προσεγγίσεις που επιχειρήθηκαν έπρεπε να λάβουν υπόψιν και εξωγενείς παράγοντες όπως η άνετη λειτουργία του λοιπού εξοπλισμού του εργαστηρίου, η εύκολη μετακίνηση φορτωτικών μηχανημάτων, η άνεση των εργαζομένων αλλά και η ευκολία επίβλεψης και επέκτασης του συστήματος. Η μαθηματική μοντελοποίηση και εύρεση της βέλτιστης χωροθετικά λύσης στην συγκεκριμένη περίπτωση δεν επιχειρήθηκε καθώς οι περιορισμοί που θέσαμε στο πρόβλημα μας όπως επίσης το γεγονός ότι το σύστημα κατεργασιών μας αποτελείται από μια μόνο εργαλειομηχανή δεν μας έδωσαν την δυνατότητα πολλών εναλλακτικών χωροθετήσεων.

Έτσι προέκυψαν τρείς εναλλακτικές χωροθετήσεις όπως φαίνεται στο σκαρίφημα που ακολουθεί:



Εικόνα 14: Επικρατέστερες χωροταξικές διατάξεις της υπό σχεδιασμό μονάδας κατεργασιών

Οι παραπάνω διατάξεις αποτυπώθηκαν με τις πραγματικές τους διαστάσεις και μέσα στον διαθέσιμο για την δημιουργία της μονάδας κατεργασίας χώρο.

Από τις επικρατέστερες χωροταξικές διατάξεις που φαίνονται παραπάνω επιλέχθηκε σε πρώτη φάση η διάταξη No2. Η επιλογή αυτή προέκυψε κυρίως βάσει περιορισμών (δυσκολία κίνησης στον χώρο, επικάλυψη χώρου εργασίας διπλανών εργαλειομηχανών κλπ) και όχι τόσο σαν αποτέλεσμα της Συστηματικής Χωροταξικής Διάταξης.

7.3. Επαυξημένη Πραγματικότητα στον Σχεδιασμό Διάταξης Εξοπλισμού

Όπως αναφέρθηκε παραπάνω, είτε η βέλτιστη χωροταξική διάταξη προκύψει με υπολογιστή μέσω κάποιου solver είτε χωρίς, η τελική επιλογή θα πρέπει να αξιολογηθεί ως προς τους μη μετρήσιμους παράγοντες της άνεσης της ασφάλειας αλλά και βάσει της εμπειρίας. Αυτό μέχρι τώρα επιτυγχανόταν με την εκτίμηση είτε μέσα από κατόψεις είτε μέσα από 3D εικονικά περιβάλλοντα (VR). Πλέον όμως έχουν ωριμάσει όλες οι συνθήκες που θα καθιστούσαν την επαυξημένη πραγματικότητα βασικό εργαλείο αξιολόγησης του προτεινόμενου χωροταξικού σχεδιασμού καθώς ενσωματώνει την λεπτομέρεια των 3D εικονικών μοντέλων μέσα στον υπό εξέταση χώρο. Αυτό αποτελεί ένα εργαλείο live εξέτασης του σεναρίου, των πιθανών εναλλακτικών του αλλά και προβλημάτων ή περιορισμών οι οποίοι δεν είχαν ληφθεί υπόψιν.

7.4. Το "Σενάριο" Παραγωγής

Το σενάριο στο οποίο βασίστηκε η παρούσα εργασία, αφορά την επεξεργασία ενός κυλινδρικού τεμαχίου. Η συνεργασία εικονικών και πραγματικών μοντέλων ξεκινά με την παραλαβή από το πραγματικό βιομηχανικό ρομπότ (Staubli RX90L) του εργαστηρίου κατεργασιών ενός πραγματικού κυλινδρικού τεμαχίου και την απόθεση του στον εικονικό ταινιόδρομο για να οδηγηθεί προς επεξεργασία. Εδώ αξίζει να σημειωθεί ότι ακριβώς στο σημείο απόθεσης του πραγματικού μοντέλου, το οποίο με τεχνητό τρόπο ουσιαστικά πέφτει στο έδαφος, αρχικοποιείται ένα εικονικό αντίγραφο του τεμαχίου, το οποίο και συνεχίζει την διαδρομή πάνω στον εικονικό διάδρομο.

Στόχος είναι ο χρήστης να εμβυθιστεί κατά το δυνατό περισσότερο στο περιβάλλον επαυξημένης πραγματικότητας έτσι ώστε να του δίνεται η ψευδαίσθηση ότι όντως εικονικά και πραγματικά μοντέλα μπορούν να συνεργαστούν άριστα. Το εικονικό πλέον τεμάχιο αφού φτάσει στο τέλος του πρώτου ταινιοδρόμου θα παραληφθεί από το δεύτερο ρομπότ το οποίο είναι εικονικό. Αυτό θα αναλάβει να τοποθετήσει το εικονικό τεμάχιο τον πραγματικό τόρνο οπού θα ακολουθήσει η κατεργασία του.

Σε αυτό το σημείο θα καταστραφεί το αρχικό εικονικό τεμάχιο και θα δημιουργηθεί ένα τελικό "κατεργασμένο" τεμάχιο για να δώσει πιο πιστευτή αίσθηση στον χρήστη. Το ίδιο ρομπότ και πάλι θα αναλάβει να παραλάβει το κατεργασμένο πλέον εικονικό τεμάχιο από τον πραγματικό τόρνο για να το μεταφορτώσει στον δεύτερο εικονικό ταινιόδρομο ο οποίος θα το μεταφέρει στο τελευταίο στάδιο του συστήματος.

Εκεί ένα εικονικό ρομπότ θα αναλάβει την παραλαβή του αλλά και την τοποθέτηση-παλετοποίηση του σε ειδικά εικονικά τεμάχια αποθήκευσης (δίσκοι). Η τακτοποίηση των τεμαχίων στους δίσκους γίνεται με 1 κατεργασμένο τεμάχιο ανά δίσκο. Για κάθε τεμάχιο όπου και ο δίσκος θεωρείται πλήρης, το σύστημα αυτόματης αποθήκευσης και ανάκτησης (ASRS) αναλαμβάνει την παραλαβή του δίσκου αλλά και την τοποθέτηση του σε κάποια "ελεύθερη" θέση στον αποθηκευτικό χώρο.

Παρακάτω φαίνεται ο εξοπλισμός που υπάρχει αλλά και που θα τοποθετηθεί στο Εργαστήριο Τεχνολογίας των Κατεργασιών καθώς και οι εργασίες που κάθε διάταξη εκτελεί.

Θέση	Σύμβολο	Στοιχείο FMS	Φύση Στοιχείου	Τύπος Εργασίας
1	R1	1ο Ρομπότ (6R)	Πραγματικό	Επιλογή πραγματικου τεμαχίου/Απόθεση του στον 1ο ταινιόδρομο
2	C1	1ος Ταινιόδρομος	Εικονικό	Δημιουργία και Μεταφορά εικονικού τεμαχίου
3	R2	2ο Ρομπότ (6R)	Εικονικό	Παραλαβή τεμαχίου/Φόρτωση τόρνου
4	L	Τόρνος-Φρέζα	Πραγματικό	Τόρνευση
5	R2	2ο Ρομπότ (6R)	Εικονικό	Παραλαβή τεμαχίου/Απόθεση του στον 2ο ταινιόδρομο
6	C2	2ος Ταινιόδρομος	Εικονικό	Μεταφορά τεμαχίου
7	R3	3ο Ρομπότ (6R)	Εικονικό	Παραλαβή τεμαχίου/Παλετοποίηση/Τοποθέτηση δίσκου
8	ASRS	ASRS	Εικονικό	Παραλαβή δίσκου με κατεργασμένο τεμάχιο και αποθήκευση του

Πίνακας: Ενδεικτικός εξοπλισμός της υπό σχεδιασμό μονάδας κατεργασίας

Έτσι λοιπόν σε έναν πλήρη κύκλο κατεργασίας, το σύστημα μας επεξεργάζεται 1 τεμάχιο και αποθηκεύει μία παλέτα-δίσκο ενώ ο χρόνος που απαιτείται, εξαρτάται κυρίως από τον χρόνο κατεργασίας στην εργαλειομηχανή η οποία και θα δώσει τον ρυθμό.

7.5. Συνεργασία Πραγματικού και Επαυξημένου Εξοπλισμού και Προσομοίωση Παραγωγικής Διαδικασίας

Η πρόκληση του παραπάνω σεναρίου είναι να δίνεται η εντύπωση στον χρήστη πως ο υφιστάμενος εξοπλισμός του εργαστηρίου όντως μπορεί να συνεργαστεί με εικονικά μοντέλα. Αυτό μπορεί να πραγματοποιηθεί με διάφορους τρόπους όπως η τοποθέτηση εικόνων στόχων στα πραγματικά αντικείμενα ή χρήση κάποιων αισθητήρων οι οποίοι πχ θα ενημερώνουν την εφαρμογή πότε το πραγματικό ρομπότ είναι έτοιμο να "ξεφορτώσει" το πραγματικό τεμάχιο ώστε την ίδια στιγμή να αρχικοποιεί το εικονικό.

Στην περίπτωση μας, για την ανάπτυξη της εφαρμογής ARobot_Layout χρησιμοποιήθηκε ο χρονικός προγραμματισμός. Όλες οι κινήσεις είτε πραγματικού είτε εικονικού εξοπλισμού πραγματοποιούνται ανά συγκεκριμένα χρονικά διαστήματα και με μία συγκεκριμένη σειρά ώστε να συνθέτουν όλες μαζί μια σωστή παραγωγική διαδικασία.

Οι εφαρμογές επαυξημένης πραγματικότητας προσθέτουν εικονικό περιεχόμενο ανάμεσα στην κάμερα του χρήστη και τα πραγματικά αντικείμενα (On-Top). Αυτό σημαίνει ότι ένα πραγματικό αντικείμενο δεν μπορεί να παρεμβληθεί ανάμεσα στην κάμερα του χρήστη και τα εικονικά αντικείμενα χωρίς το κατάλληλο λογισμικό ή και τους κατάλληλους αισθητήρες (πχ Microsoft Kinect) [43].

Η διαδικασία κατά την οποία ένα πραγματικό αντικείμενο παρεμβάλλεται μεταξύ της κάμερας του χρήστη και του εικονικού περιεχομένου ονομάζεται Έμφραξη (Occlusion) και είναι επιθυμητή ώστε να γίνεται πιο ρεαλιστική η εκάστοτε εφαρμογή και διαδραστική με τα πραγματικά αντικείμενα.

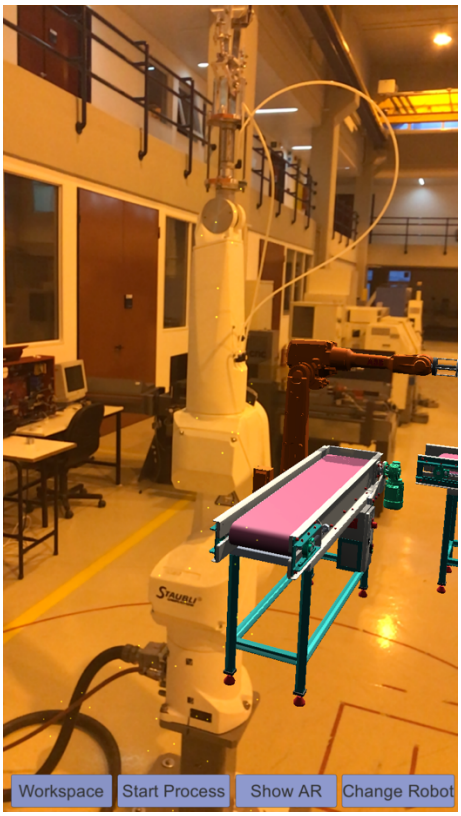
α) *β)* *γ)*

Εικόνα 15: Επαυξημένη πραγματικότητα α) στον χώρο χωρίς να παρεμβάλλεται πραγματικό αντικείμενο ανάμεσα στην κάμερα και το εικονικό αντικείμενο β) χωρίς έμφραξη γ) με έμφραξη *[44]*

Στην παρούσα εφαρμογή η έμφραξη δεν αποτέλεσε στόχο υλοποίησης καθώς η χωροταξική διάταξη που επιλέχθηκε και οι κινήσεις του πραγματικού ρομπότ του εργαστηρίου γίνονται με τρόπο που από

τις περισσότερες γωνίες λήψης το εικονικό περιεχόμενο είναι On-Top δηλαδή ανάμεσα στον χρήστη και τα πραγματικά αντικείμενα.

Έτσι, ο χώρος επισκόπησης της μονάδας κατεργασίας επαυξημένης πραγματικότητας που προκύπτει, μας δίνει ένα σωστό αποτέλεσμα μόνο όταν ο πραγματικός εξοπλισμός είναι τοποθετημένος περιμετρικά του σημείου λήψης της εικόνας, προκειμένου να είναι ανεξάρτητος της έμφραξης. Με την χρήση κατάλληλων αισθητήρων ή με την χρήση νέων συσκευών που θα υποστηρίζουν εφαρμογές Μεικτής Πραγματικότητας (MR) όπως το Magic Leap και το Hololens οι οποίες θα δίνουν την δυνατότητα έμφραξης σκανάροντας αυτόματα το περιβάλλον στο οποίο βρίσκεται ο χρήστης, το πρόβλημα αυτό είναι εύκολα αντιμετωπίσιμο.

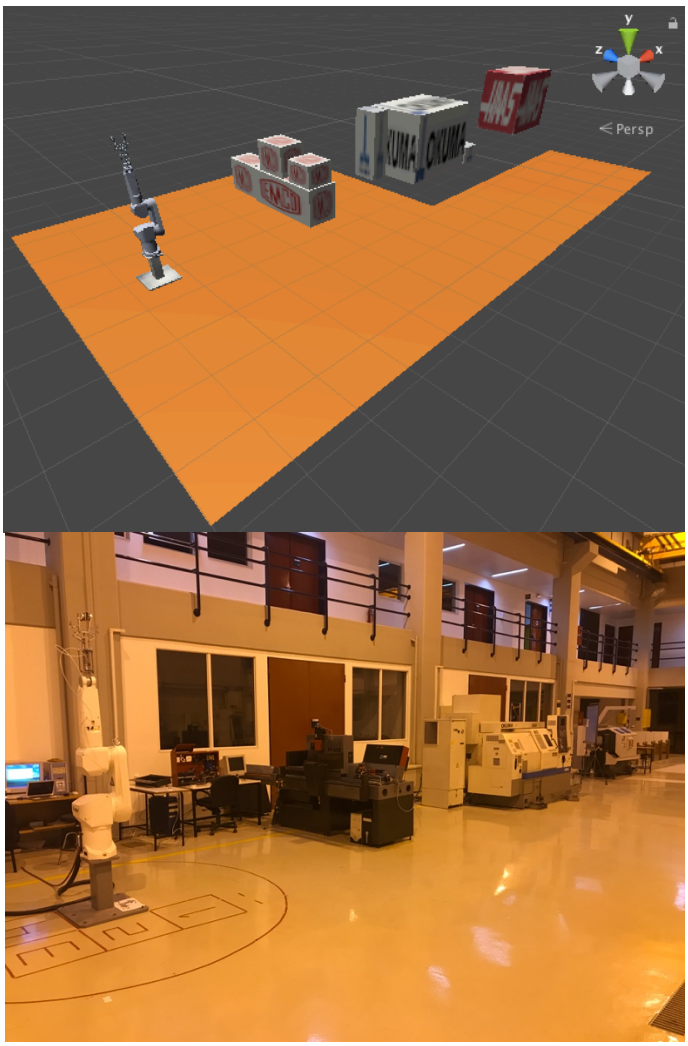


Εικόνα 16: Εικονικός εξοπλισμός ανάμεσα στον πραγματικό εξοπλισμό και τον χρήστη (On-Top). Απουσία Έμφραξης

7.6. Εικονικά Μοντέλα

7.6.1. Εισαγωγή

Στο περιβάλλον του Unity, αρχικά έγινε η μεταφορά της υπάρχουσας κατάστασης του εργαστηρίου κατεργασιών χρησιμοποιώντας κυρίως primitives με διαστάσεις αντίστοιχες του πραγματικού εξοπλισμού προκειμένου να υπάρχει καλύτερη εποπτεία του διαθέσιμου χώρου.



Εικόνα 16: Υφιστάμενος εξοπλισμός εργαστηρίου κατεργασιών και διαθέσιμος χώρος για την δημιουργία της ευέλικτης μονάδας κατεργασιών

Στη συνέχεια προστέθηκαν τα 3D μοντέλα όλου του εξοπλισμού που πιθανόν να τοποθετηθεί. Η εισαγωγή των μοντέλων αυτών έγινε με την παρακάτω διαδικασία:

7.6.2. Asset Importing

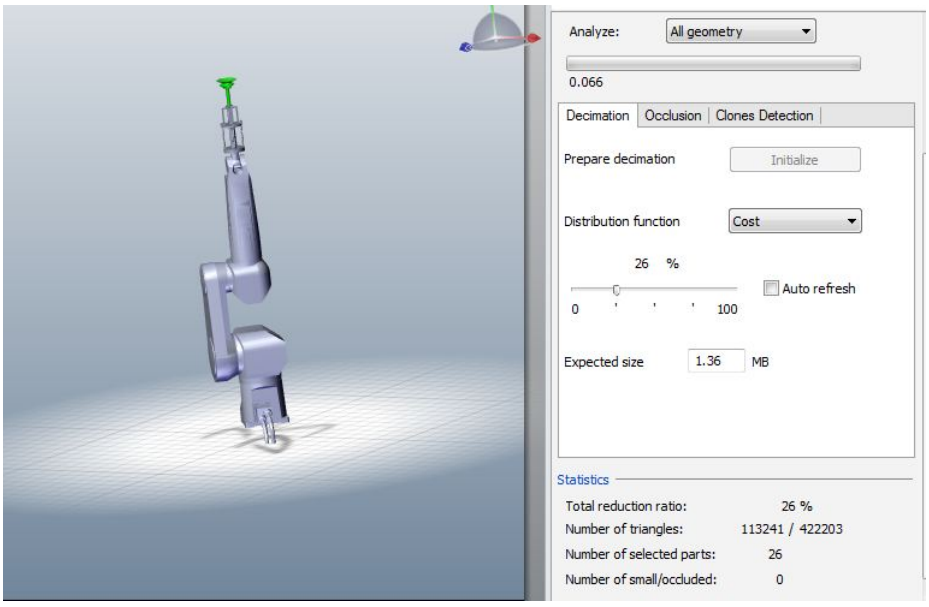
Στον φάκελο Assets της καρτέλας Project του Unity μπορεί να γίνει εισαγωγή των συμβατών 2D,3D, Audio, Animation, Texture και άλλων Assets [45].

Ο υπολογιστής (MacBook Air) που χρησιμοποιήθηκε κατά την διάρκεια της εργασίας έφερε επεξεργαστή Intel Core i5 1.6GHz dual-core με 4GB μνήμη RAM και κάρτα γραφικών Intel HD Graphics

6000. Κατά την διάρκεια των δοκιμών παρουσιάστηκαν σημαντικά προβλήματα απόδοσης στα γραφικά (κυρίως κατά την κίνηση των εικονικών μοντέλων) παρότι η κάρτα γραφικών είχε αρκετά καλά χαρακτηριστικά για τα δεδομένα της παρούσας εργασίας. Αυτό κυρίως ήταν αποτέλεσμα της ύπαρξης ταυτόχρονα στην σκηνή πολλών High-Poly 3D μοντέλων, δηλαδή μοντέλων τα οποία είχαν πολύ μεγάλο αριθμό τρίγωνων επομένως ήταν πάρα πολύ λεπτομερή. Προς την επίλυση αυτού του προβλήματος επιλέχθηκε η απλοποίηση κάποιων εκ των μοντέλων προκειμένου να είναι πιο ομαλή η διαχείριση τους από την κάρτα γραφικών. Η "απλοποίηση" ή αλλιώς decimation ή simplification των εικονικών μοντέλων είναι μια πολύ διαδεδομένη διαδικασία ιδιαίτερα στο game development όπου η ταχύτητα και η βέλτιστη απόδοση είναι ζητούμενα. Η επέμβαση έγινε με την λογική της απόρριψης των περιττών για τα δεδομένα της εργασίας λεπτομερειών και πάντα με κριτήριο να μην

υποβαθμιστεί το τελικό αποτέλεσμα. Αυτό επιτεύχθηκε με το λογισμικό SolidWorks Composer. Το SolidWorks Composer είναι ένα λογισμικό που χρησιμοποιείται για την καλύτερη απεικόνιση σχεδίων, τρισδιάστατων μοντέλων, και aniamation όπως επίσης για την συγγραφή ποιοτικότερων τεχνικών εγχειριδίων.

Έχοντας εισάγει τα μοντέλα μας ένα προς ένα στο περιβάλλον του SolidWorks Composer επιλέξαμε την εντολή Simplification από την καρτέλα Workshops. Έπειτα επιλέξαμε All geometry στην κατηγορία Analyze και την επιλογή Cost στην κατηγορία Distribution function προκειμένου να ρυθμίσουμε το ποσοστό μείωσης των τριγώνων σε σχέση με το αρχικό μοντέλο.



Εικόνα 17: Decimation του 3D ρομπότ Staubli RX90L

Στη παραπάνω εικόνα φαίνεται το τελικό αποτέλεσμα το οποίο είναι σχεδόν ίδιο σε σχέση με το αρχικό μοντέλο. Στην πραγματικότητα η παραπάνω εικόνα είναι περίπου 75% λιγότερο λεπτομερής (113 241 τρίγωνα σε σχέση με τα 422 203 που είχε το αρχικό μοντέλο), με το τελικό αισθητικό

αποτέλεσμα να είναι ανεπαίσθητα υποδεέστερο εξασφαλίζοντας όμως ταυτόχρονα πολύ καλύτερη γραφική απόδοση στη σκηνή μας, ιδιαίτερα κατά την διάρκεια κινήσεων των 3D μοντέλων.

Κατά την παραπάνω διαδικασία χάνονται όλες οι εξαρτήσεις (πχ κληρονομικότητα) των Assemblies οπότε θα πρέπει να δημιουργηθούν εκ νέου μέσα από το Unity, δηλαδή ομάδες Game Object τα οποία θέλουμε να έχουν την ίδια συμπεριφορά κατά την διάρκεια μιας συγκεκριμένης κίνησης.

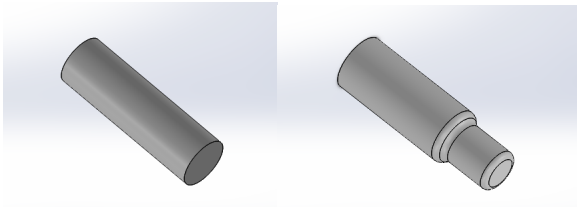
Κάθε μοντέλο αποθηκεύθηκε από το SolidWorks Composer σε μορφή 3D Studio Model για να μπορεί να εισαχθεί μέσω της εντολής import στο λογισμικό 3ds Max προκειμένου από εκεί να το μετατρέψουμε σε μορφή FBX η οποία είναι συμβατή με το Unity.

Τέλος, αξίζει να σημειωθεί ότι με την παραπάνω διαδικασία, τα 3D μοντέλα χάνουνε το αρχικό τους scaling με αποτέλεσμα κατά την εισαγωγή τους στο περιβάλλον του Unity να έχουν μεγαλύτερες διαστάσεις από τις πραγματικές. Για τον λόγο αυτό κάθε φορά που γινόταν εισαγωγή ενός Decimated 3D μοντέλου ως Asset στο Unity ρυθμίζαμε την παράμετρο Scale Factor ώστε οι διαστάσεις του Decimated 3D μοντέλου να συμπίπτουν με τις διαστάσεις του αρχικού μοντέλου (3D μοντέλο πρίν υποστεί απλοποίηση με το SolidWorks Composer) το οποίο διατηρούσαμε στην σκηνή του Unity μόνο γι’ αυτό τον λόγο. Έτσι κατά την εισαγωγή decimated μοντέλων δηλαδή μοντέλων που υπέστησαν την παραπάνω διαδικασία απλοποίησης των γραφικών τους το scale factor τους πήρε την τιμή 0,0392. Επίσης σε αυτό το σημείο θα πρέπει να αναφερθεί ότι το Unity υποθέτει ότι 1 unity unit = 1 m.

7.6.3. Ακατέργαστο και Κατεργασμένο Τεμάχιο

7.6.4. Περιγραφή

Η μονάδα κατεργασίας επαυξημένης πραγματικότητας που θα προκύψει, θα έχει ως στόχο να κατεργάζεται αξονοσυμμετρικά τεμάχια ερταλόν με διαστάσεις 40mmX130mm και να τα μεταποιεί σε τεμάχια διαστάσεων 40mmX90mm+20mmX40mm όπως φαίνεται και την εικόνα που ακολουθεί.



Εικόνα 18*: Ακατέργαστο και κατεργασμένο τεμάχιο ερταλόν*

7.6.5. Κίνηση των Τεμαχίων στην Σκηνή του Unity

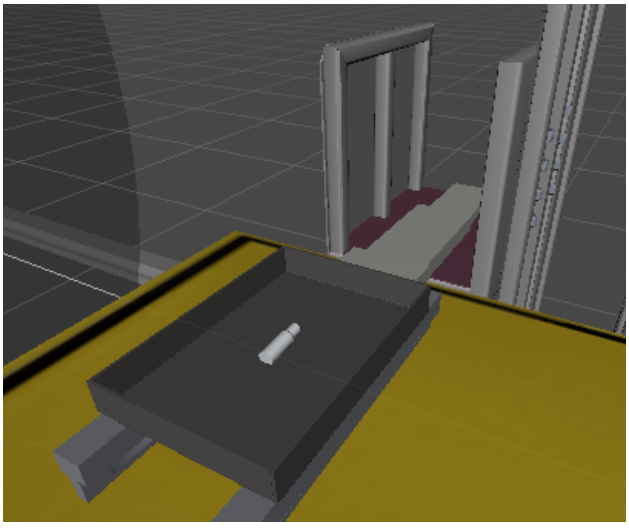
Τα ακατέργαστα και κατεργασμένα προϊόντα είναι το επίκεντρο μίας παραγωγικής διαδικασίας. Έτσι και στην δική μας περίπτωση τα ακατέργαστα και κατεργασμένα τεμάχια είναι το επίκεντρο όλων των κινήσεων στην σκηνή του Unity. Όλα τα Game Objects στη σκηνή προγραμματίστηκαν με τέτοιο τρόπο ώστε να ικανοποιούν τις ανάγκες μεταφοράς, κατεργασίας και αποθήκευσης των εικονικών τεμαχίων.

Κάθε εικονικό τεμάχιο λοιπόν εκτελεί, στην ακατέργαστη μορφή του, μια γραμμική κίνηση επι του πρώτου ταινιοδρόμου ώσπου να γίνει child του end effector του ρομπότ το οποίο το μεταφέρει στην εργαλειομηχανή για επεξεργασία. Η επεξεργασία αυτή οδηγεί στην παραγωγή του τελικού τεμαχίου (γίνεται Destroy του rawItem και Instantiate του readyItem στο ίδιο ακριβώς σημείο).

Στη συνέχεια ακολουθεί ξανά γραμμική κίνηση επι του δεύτερου ταινιοδρόμου για να γίνει child του end effector του τελευταίου ρομπότ το οποίο το εναποθέτει στον δίσκο μεταφοράς. Από εκεί και πέρα, και αφού πλέον το κατεργασμένο τεμάχιο γίνει παιδί του μεταφορικού δίσκου, με την μέθοδο ItemUnLoading() του ReadyItemMotionManager όπως φαίνεται και παρακάτω, αναλαμβάνει δράση το σύστημα Αυτόματης Αποθήκευσης και Ανάκτησης για να το αποθηκεύσει σε κενή θέση της αποθήκης.

```
//This method assigns the ready item on the active tray in order to be ready for storage.
void ItemUnLoading()
{
    desiredParentGameObject = GameObject.FindWithTag ("ActiveTray");
    targetGameObject.transform.parent = desiredParentGameObject.transform;
}
```

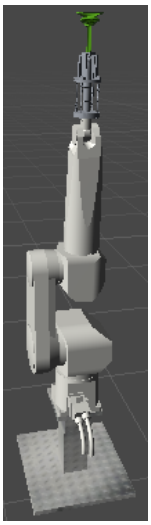
Εικόνα 19*: Μέθοδος ItemUnLoading() με την οποία το κατεργασμένο τεμάχιο γίνεται παιδί του ενεργού* δίσκου μεταφοράς



Εικόνα 20: Κατεργασμένο τεμάχιο ως παιδί του ενεργού δίσκου κατά την διαδικασία της παραλαβής του δίσκου από το αυτόματο σύστημα μεταφοράς

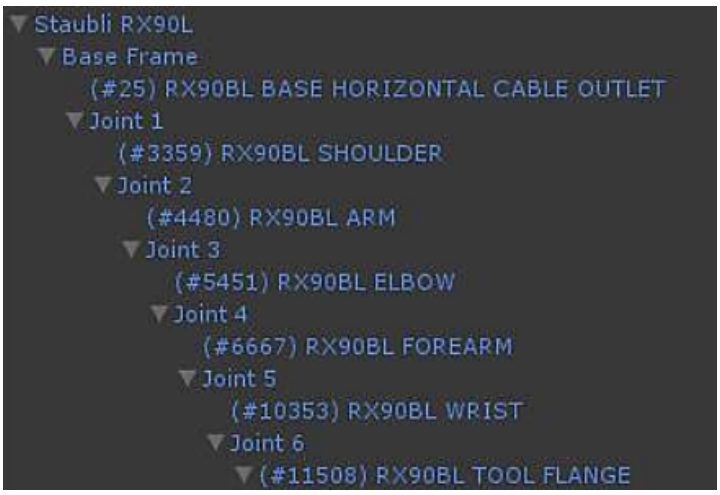
7.6.6. Staubli RX90L

7.6.7. Περιγραφή



Το εικονικό ρομπότ Staubli RX90L έχει ακριβώς τα ίδια χαρακτηριστικά και διαστάσεις με το πραγματικό ρομπότ του εργαστηρίου κατεργασιών που περιεγράφηκε παραπάνω. Τα εύρη των γωνιών των αρθρώσεων ακολούθησαν μέσω κώδικα τα πραγματικά εύρη ενώ στην τελευταία του άρθρωση έχει προσαρμοστεί ειδική 'αρπάγη' κενού με αναρρόφηση (robotic vacuum gripper)

Το ρομπότ αποτελείται από ξεχωριστά parts τα οποία είναι τελείως ανεξάρτητα κατά την εισαγωγή στου εικονικού μοντέλο στο περιβάλλον του Unity. Σε ένα βιομηχανικό ρομπότ κάθε άρθρωση κινείται όχι μόνο γύρω από τον άξονα περιστροφής της αλλά και γύρω από τους άξονες περιστροφής των προηγούμενων αρθρώσεων αν και εφόσον αυτές κινούνται. Έτσι θα δημιουργήσαμε μια σχέση κληρονομικότητας μεταξύ των αρθρώσεων για να το πετύχουμε αυτό όπως φαίνεται παρακάτω.



Εικόνα 21: Σχέση κληρονομικότητας μεταξύ των αρθρώσεων του ρομπότ στο Unity

Κάθε άρθρωση επιθυμούμε να περιστρέφεται γύρω από τον τοπικό άξονα Z προκειμένου να είμαστε συμβατοί με την προσέγγιση Denavit Hartenberg. Γι' αυτό τον λόγο δημιουργούμε τα κενά Game Objects με όνομα Joint 1-6 τα οποία τοποθετούμε με τέτοιον τρόπο ώστε να περιστρέφονται πάντα γύρω από τον αντίστοιχο Z άξονα. Η άρθρωση που είναι παιδί του αντίστοιχου Joint θα περιστραφεί και αυτή με τον ίδιο τρόπο λόγω κληρονομικότητας.

7.6.8. Κινηματική Ανάλυση του Ρομπότ

Κατά την ανάλυση της κίνησης ενός ρομπότ και τον προσδιορισμό της ακριβούς θέσης του τελικού σημείου δράσης, υπάρχουν δύο δυνατές προσεγγίσεις.

Η πρώτη είναι το ευθύ κινηματικό πρόβλημα και λαμβάνει ως δεδομένες τις τιμές των γωνιών των αρθρώσεων και μεσώ των γεωμετρικών χαρακτηριστικών του ρομπότ προσδιορίζει την θέση και τον

προσανατολισμό του τελικού σημείου δράσης ως προς το σύστημα συντεταγμένων της βάσης του ρομπότ.

Η δεύτερη προσέγγιση είναι το αντίστροφο κινηματικό πρόβλημα και λαμβάνει ως δεδομένη τη θέση και τη περιστροφή του τελικού σημείου δράσης, προκειμένου να υπολογίσει τις γωνίες των αρθρώσεων που το οδηγούν σε αυτή τη θέση. Η λύση σε αυτό το πρόβλημα δεν υπάρχει πάντα και εφόσον υπάρχει, οι λύσεις μπορεί να είναι πολλαπλές.

Το πρόβλημα της κίνησης του ρομπότ στη παρούσα εργασία επιλύθηκε με την πρώτη προσέγγιση δηλαδή χρησιμοποιώντας το ευθύ κινηματικό μοντέλο [46].

Η ευθεία κινηματική ανάλυση βασίζεται στην μέθοδο Denavit – Hartenberg (DH) η οποία αναπτύχθηκε για την περιγραφή της κινηματικής σύνθετων μηχανισμών από έναν ελάχιστο αριθμό παραμέτρων [47].

Η μέθοδος αυτή έχει στόχο την παραγωγή των ομογενών μετασχηματισμών που συνδέουν τη θέση και τον προσανατολισμό ενός συνδέσμου ως προς τον προηγούμενο. Οι ομογενείς μετασχηματισμοί είναι πίνακες 4x4 και περιέχουν και την θέση και τον προσανατολισμό [48].

Πολλαπλασιασμός των επιμέρους πινάκων διαδοχικά από τη βάση μέχρι το εργαλείο δίνει τον ομογενή μετασχηματισμό του εργαλείου ως προς τη βάση του ρομπότ.

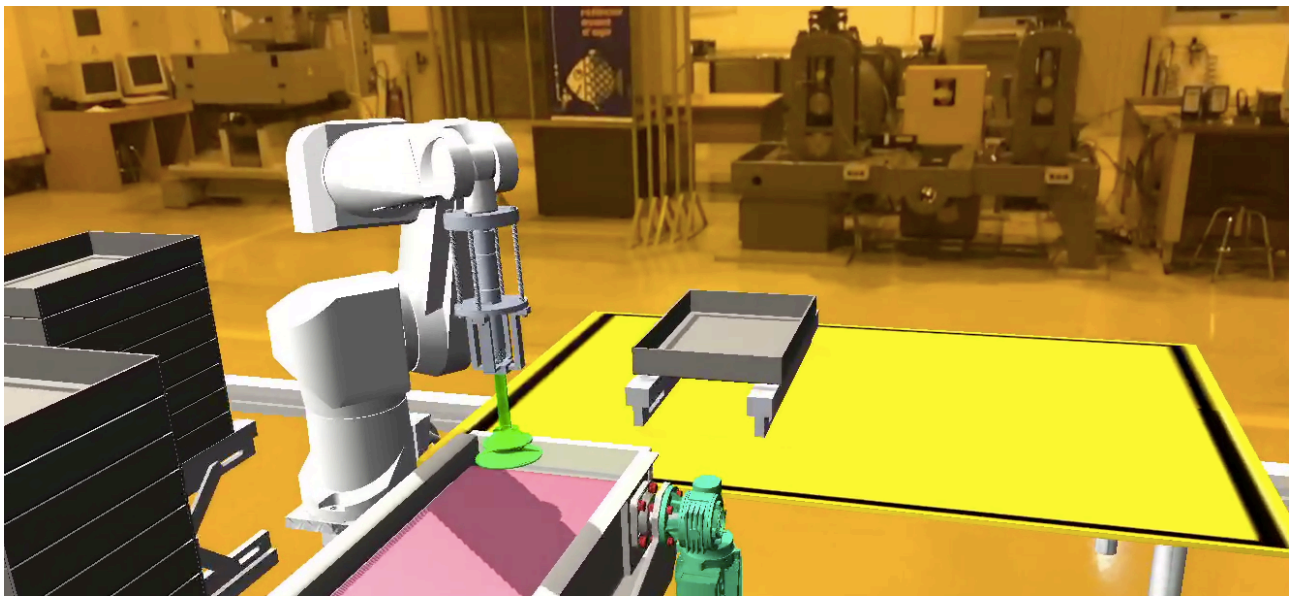
Γενικά η μέθοδος χωρίζεται σε δύο βήματα:

α) Προσάρτηση συστημάτων συντεταγμένων (ΣΣ) στους συνδέσμους

β) Υπολογισμός παραμέτρων Denavit – Hartenberg

Η προσάρτηση των συστημάτων συντεταγμένων στο εικονικό μας ρομπότ έγινε σύμφωνα με τις συμβάσεις DH όπως αναλύεται και παρακάτω και οι περιστροφή κάθε άρθρωσης γίνονται πάντα γύρω από τον άξονα Z.

Ο υπολογισμός των παραμέτρων DH στην δική μας περίπτωση δεν ήταν απαραίτητος καθώς η θέση και ο προσανατολισμός των αντικειμένων τέκνων είναι άρρηκτα συνδεδεμένος με την θέση και τον προσανατολισμό των αντικειμένων γονέων. Έτσι, και με τον τρόπο που έχουμε ιεραρχημένο κάθε joint ως τέκνο του προηγούμενου μπορούμε να προσδιορίσουμε την θέση και τον προσανατολισμό του τελικού σημείου δράσης ως προς το σύστημα συντεταγμένων της βάσης του ρομπότ χωρίς τους ομογενείς μετασχηματισμούς.



Εικόνα 22: Εικονικό βιομηχανικό ρομπότ Staubli, σε αναμονή φόρτωσης του επεξεργασμένου τεμαχίου, στο Εργαστήριο Τεχνολογίας των Κατεργασιών

Κάθε ρομπότ στη σκηνή του Unity έχει προσαρτημένο πάνω του το component Script FKMoveri, όπου i=1-3, αφού κάθε ρομπότ επιθυμούμε να εκτελεί διαφορετικές κινήσεις. Προκειμένου να προσδιορίσουμε ακριβώς την θέση και τον προσανατολισμό των τελικών σημείων δράσης, χρησιμοποιήσαμε για κάθε ρομπότ το component Script FKMover με το οποίο σε λειτουργία Play του Unity μπορέσαμε να αλλάξουμε τις γωνίες κάθε άρθρωσης ξεχωριστά προκειμένου να οδηγήσουμε το τελικό σημείο δράσης στο επιθυμητό σημείο. Μετά από αυτό το Script FKMover δεν μας χρειάζεται οπότε το απενεργοποιούμε από κάθε ρομπότ.

7.6.9. Τρόπος Προγραμματισμού της Τροχιάς

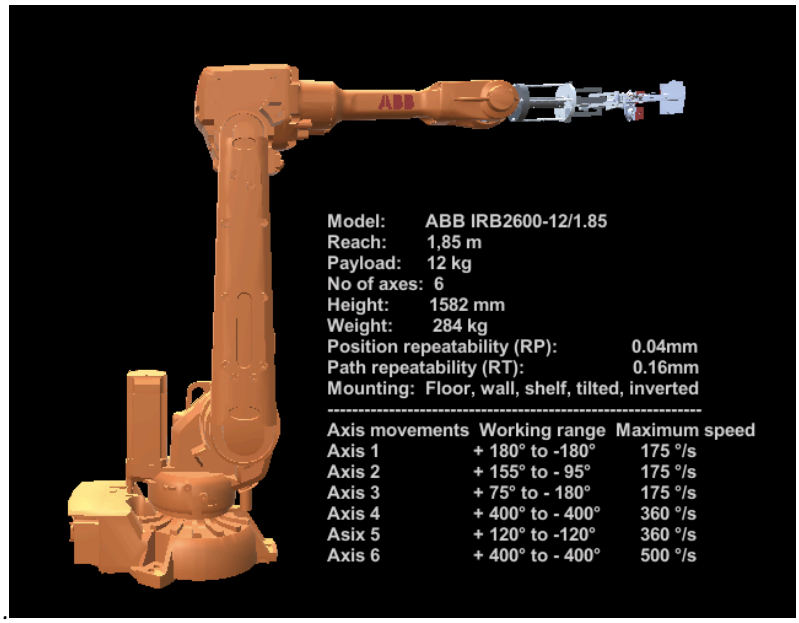
Για την κίνηση του ρομπότ από ένα σημείο σε ένα άλλο χρησιμοποιήσαμε “σφαιρική” παρεμβολή μέσω της συνάρτησης SLerp της coroutine GoToTarget(). Η συγκεκριμένη coroutine λαμβάνει ως ορίσματα εισόδου τις αρχικές rotationZiFrom και τελικές rotationZiTo γωνίες όλων των αρθρώσεων (i=1-6) και τον χρόνο μέσα στον οποίο επιθυμούμε να ολοκληρωθεί αυτή η κίνηση. Όλες οι υπόλοιπες coroutines των Script FKMoveri χρησιμοποιούνται για να δηλώσουν τις αρχικές και τελικές γωνίες των αρθρώσεων σε μια κίνηση.

7.6.10. ABB IRB2600

7.6.11. Περιγραφή

Το ρομπότ ABB IRB2600 είναι επίσης ένα 6R βιομηχανικό ρομπότ το οποίο χρησιμοποιήσαμε ως εναλλακτική κατά την διερεύνηση του τύπου ρομπότ που θα συμπληρώσει το σύστημα κατεργασιών. Παρακάτω φαίνονται τα τεχνικά και λειτουργικά χαρακτηριστικά του ενώ αξίζει να σημειωθεί ότι είναι ένα ρομπότ με σχεδόν διπλάσιο payload από το Staubli. Η παρούσα εφαρμογή είναι επεκτάσιμη και δίνει την δυνατότητα εξέτασης πολλών εναλλακτικών ρομπότ η γενικότερα συμπληρωματικών

μηχανημάτων αλλά. Η κινηματική ανάλυση έγινε μόνο για το ρομπότ Staubli RX90L στα πλαίσια της παρούσας εφαρμογής.



Εικόνα 23*: Βιομηχανικό ρομπότ ABB IRB2600 και τα τεχνικά του χαρακτηριστικά (SetupState/Scene0)*

7.6.12. Αυτόματο Σύστημα Αποθήκευσης και Ανάκτησης (ASRS)

7.6.13. Περιγραφή

Ένα αυτόματο σύστημα αποθήκευσης και ανάκτησης (ASRS ή AS / RS) αποτελείται από μια ποικιλία ελεγχόμενων από υπολογιστή συστημάτων για αυτόματη τοποθέτηση και ανάκτηση φορτίων από καθορισμένες θέσεις αποθήκευσης [49].

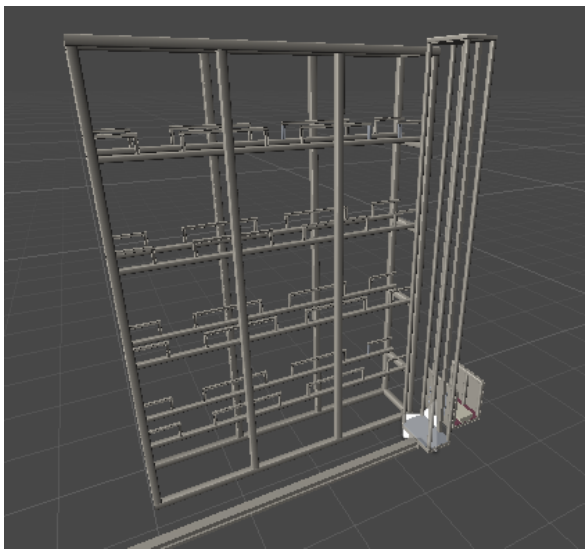
Τα αυτόματα συστήματα αποθήκευσης και ανάκτησης (ASRS) χρησιμοποιούνται συνήθως σε εφαρμογές όπου:

-Υπάρχει ένας πολύ μεγάλος όγκος φορτίων που μεταφέρονται μέσα και έξω από την αποθήκη.

-Η πυκνότητα αποθήκευσης είναι σημαντική λόγω των περιορισμών χώρου.

-Δεν υπάρχει κάποια κατεργασία σε αυτήν τη διαδικασία (μόνο αποθήκευση και μεταφορά).

-Η ακρίβεια είναι κρίσιμη λόγω των πιθανών δαπανηρών βλαβών στο φορτίο αλλά κι την εγκατάσταση [50].



Εικόνα 24: Αυτόματο Σύστημα Αποθήκευσης και Ανάκτησης (ASRS)

Το σύστημα αυτόματης αποθήκευσης και ανάκτησης που φαίνεται παραπάνω και που χρησιμοποιήσαμε στα πλαίσια της εργασίας διαθέτει 24 θέσεις αναμονής. Το "βαγόνι" κίνησης κινείται πάνω σε μια ράγα στον άξονα X, ενώ οι κατακόρυφη διάταξη που το συνοδεύει είναι υπεύθυνη για την στήριξη και κίνηση του βαγονιού κατά τον άξονα Y. Για την αποθήκευση και ανάκτηση τεμαχίων ή παλετών το βαγόνι διαθέτει μηχανισμό ο οποίος κινείται επάνω στον άξονα Z για να φορτώσει και να αποθηκεύσει το φορτίο. Η κίνηση κατά τους άξονες X και Y συνοδεύεται από προειδοποιητικό αλάρμ για λόγους ασφαλείας το οποίο εισήχθηκε στο project ως audio source asset και η αναπαραγωγή του ρυθμίστηκε με τις εντολές asrsAlarm.Play (); και asrsAlarm.Pause ();. Η κίνηση στον άγονα Z είναι προγραμματισμένη να γίνεται μόνο όταν δεν υπάρχει κίνηση στους άλλους άξονες για την αποφυγή ζημιών στον εξοπλισμό.

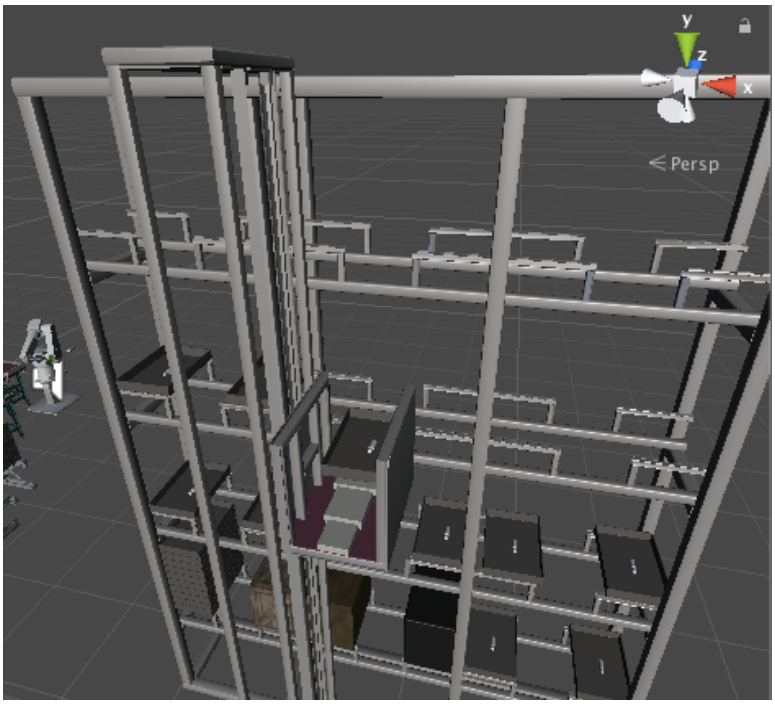
7.6.14. Προγραμματισμός Κίνησης και Αποθήκευσης Τεμαχίων

Το script το οποίο είναι υπεύθυνο για την κίνηση του συστήματος ASRS είναι το asrsMotion. Για την μοντελοποίηση της κίνησης του αυτόματου συστήματος αποθήκευσης και ανάκτησης (ASRS), ο αποθηκευτικός χώρος αντιμετωπίστηκε σαν ένας πίνακας με τέσσερις γραμμές και τρείς στήλες. Επιπλέον, οι κινήσεις του χωρίστηκαν με βάση του σε ποιόν άξονα γίνονται και δημιουργήθηκαν Game Objects στα σημεία που αποτελούν στόχο.

Για την κίνηση του βαγονιού κατά τους τρείς αυτούς άξονες χρησιμοποιήσαμε "γραμμική" παρεμβολή μέσω της συνάρτησης Lerp της coroutine MoveObject() για την κίνηση στους άξονες X και Y και της coroutine MoveCarrierOnAxisZ() για την κίνηση στον άξονα Z. Οι παραπάνω coroutines λαμβάνουν ως ορίσματα εισόδου τις αρχικές και τελικές θέσεις για τον αντίστοιχο άξονα (πχ xStartPos-xEndPos για την κίνηση στον άξονα X) όπως επίσης και τον χρόνο μέσα στον οποίο επιθυμούμε να ολοκληρωθεί αυτή η κίνηση. Όλες οι υπόλοιπες coroutines του Script asrsMotion χρησιμοποιούνται για να δηλώσουν τις αρχικές και τελικές θέσεις σε μια κίνηση. Η κάθε επιθυμητή θέση δεν δόθηκε υπό την

μορφή απόλυτων ή σχετικών συντεταγμένων από το περιβάλλον του Unity. Κάθε επιθυμητό σημείο αναπαρίσταται με ένα κενό Game Object στη σκηνή του Unity. Έτσι ομαδοποιήσαμε σε τρείς λίστες, μία για κάθε άξονα, όλα τα σημεία στα οποία θέλουμε να βρεθεί το σύστημα μεταφοράς του ASRS.

Εκτός από τις coroutines που εξετάσαμε, το Script asrsMotion φέρει κάποιες μεθόδους όπως είναι οι BoxLoading(), BoxUnloading() κλπ οι οποίες είναι υπεύθυνες για την παραλαβή του έτοιμου τεμαχίου πάνω στον δίσκο αλλά και την αποθήκευση του στον αποθηκευτικό χώρο. Αυτό επιτεύχθηκε χρησιμοποιώντας σχέσεις γονέα-παιδιού μεταξύ των Game Objects του Unity. Κάθε έτοιμος προς φόρτωση δίσκος (τοποθετημένος στην θέση παραλαβής αλλά και με το τεμάχιο τοποθετημένο πάνω του) παίρνει το Tag ActiveTray. Με το που γίνει η φόρτωση του, η μέθοδος BoxLoading() είναι υπεύθυνη μέσω της εντολής targetGameObject.tag = "Untagged"; να κάνει Untagged τον συγκεκριμένο δίσκο προκειμένου να πάρει ο νέος προς παραλαβή δίσκος το Tag ActiveTray.

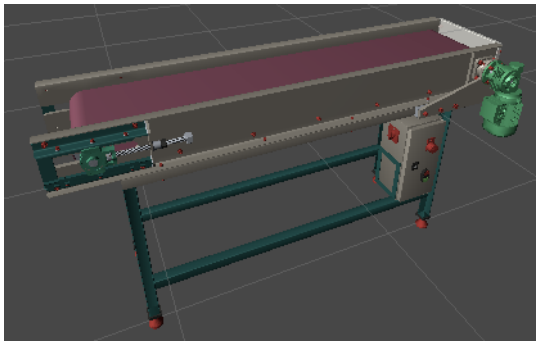


Εικόνα 25: ASRS κατά την διαδικασία εκφόρτωσης-αποθήκευσης

7.6.15. Ταινιόδρομος Μεταφοράς

7.6.16. Περιγραφή

Ένα σύστημα ταινιοδρόμου αποτελείται από τουλάχιστον δύο τροχαλίες (ή ράουλα ή τύμπανα), με έναν ατέρμονα βρόχο μεταφοράς μέσου, τον μεταφορικό ιμάντα (συνήθως από καουτσούκ) - που περιστρέφεται γύρω από αυτές.



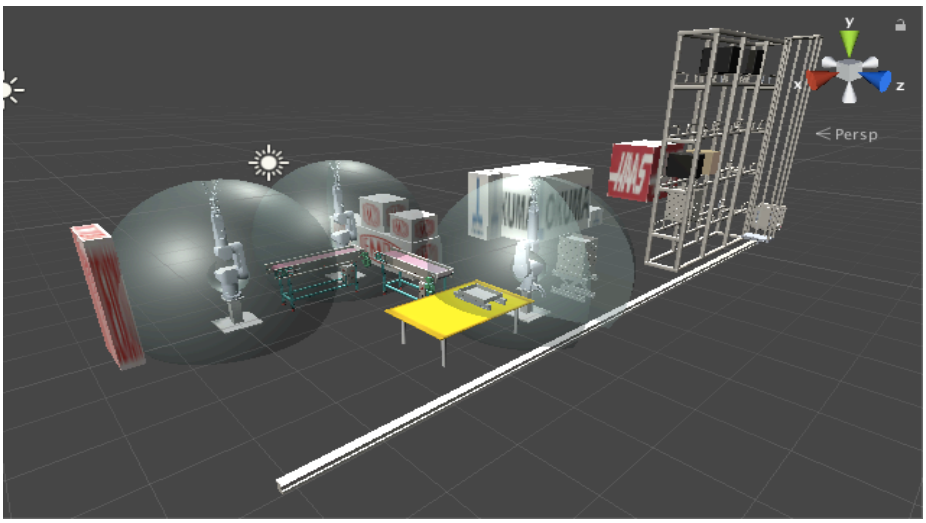
Εικόνα 26: Ταινιόδορμος μεταφοράς τεμαχίων

Τουλάχιστον μία από τις τροχαλίες περιστρέφονται, μετακινώντας την ταινία και το υλικό πάνω σε αυτήν προς τα εμπρός. Η περιστρεφόμενη τροχαλία ονομάζεται τροχαλία κίνησης ενώ η τροχαλία χωρίς κινητήρα ονομάζεται τροχαλία αδράνειας. Υπάρχουν δύο κύριες βιομηχανικές κατηγορίες ταινιοδρόμων. Οι ταινιόδρομοι υλικού, όπως εκείνοι που κινούν κιβώτια/τεμάχια/παλέτες στο εσωτερικό ενός εργοστασίου και οι ταινιόδρομοι υλικών χύδην, όπως εκείνοι που χρησιμοποιούνται για τη μεταφορά μεγάλων ποσοτήτων πόρων και γεωργικών αγαθών, όπως σιτηρά, αλάτι, άνθρακας, μεταλλεύματα κ.α.

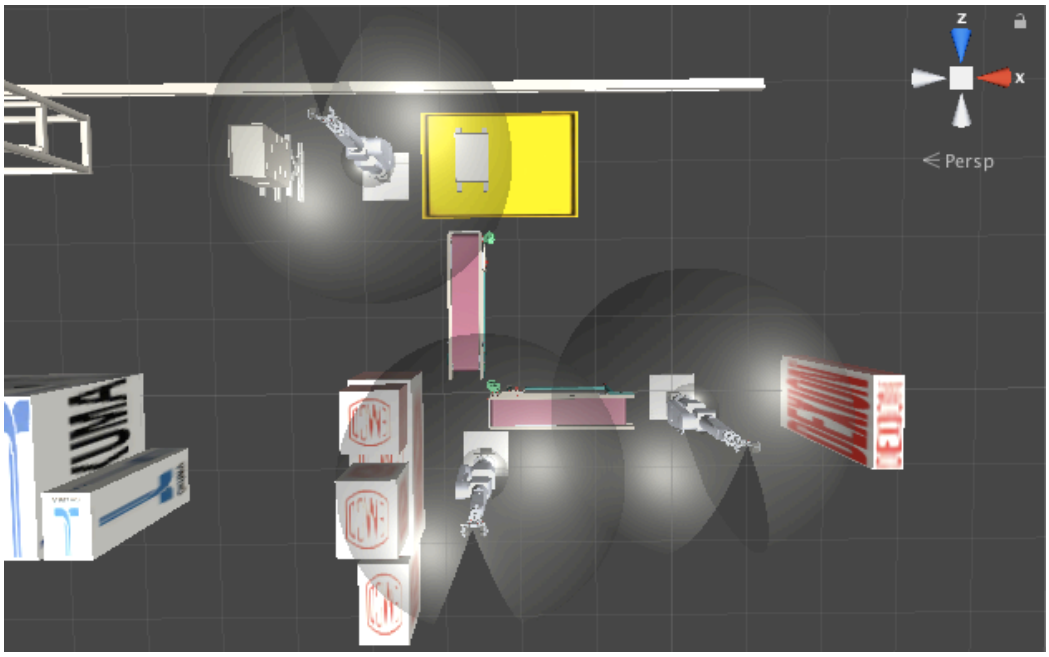
7.6.17. Προγραμματισμός Κίνησης Τεμαχίων στους Ταινιοδρόμους

Η κίνηση των εικονικών τεμαχίων πάνω στους ταινιοδρόμους γίνεται με γραμμική παρεμβολή μεταξύ του αρχικού και του τελικού σημείου σε προκαθορισμένο χρόνο μέσα απο το την coroutine MoveItem η οποία βρίσκεται στα script components RawItemMotion και ReadyItemMotion για την κίνηση των ακατέργαστων και έτοιμων τεμαχίων αντίστοιχα αφού πρώτα γίνει instantiate μέσω της μεθόδου CreateItemPrefab().Με αυτόν τον τρόπο δίνεται η αίσθηση στον χρήστη ότι κινείται ο ιμάντας του ταινιοδρόμου. Αισθητικά το αποτέλεσμα είναι ακριβώς το ίδιο.

Έτσι βάσει της χωροταξικής διάταξης, αλλά και του εξοπλισμού που επιλέξαμε παραπάνω η μονάδα κατεργασίας που προκύπτει φαίνεται στην επόμενη εικόνα:



Εικόνα 27:Υπό σχεδιασμό Μονάδα Κατεργασίας εργαστηρίου Κατεργασιών των Υλικών

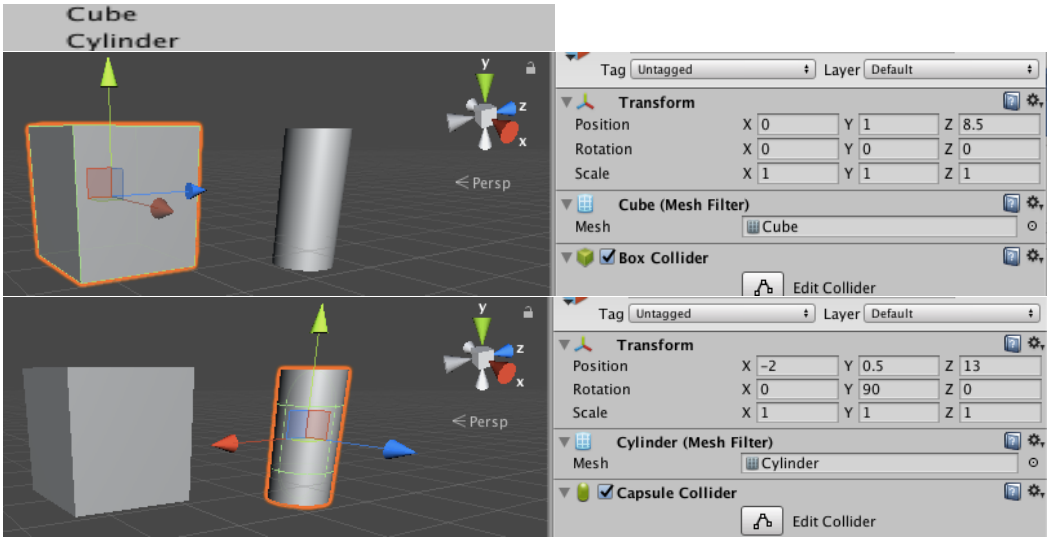


Εικόνα 28: Έλεγχος της διάταξης των ρομπότ με βάση τον χώρο εργασίας τους

7.7. Συστήματα συντεταγμένων

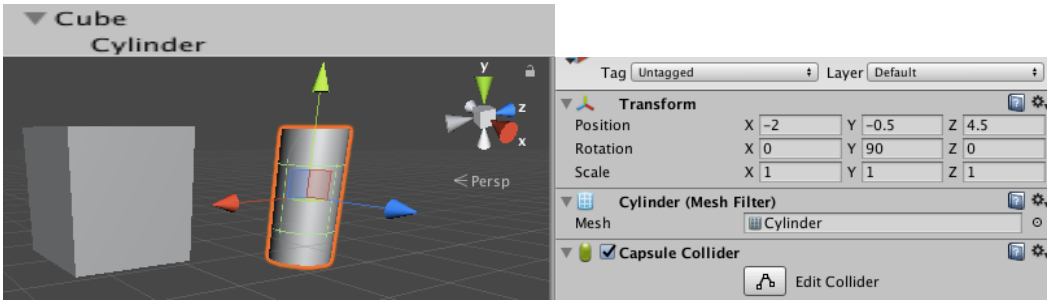
Υπάρχουν δυο συστήματα συντεταγμένων στο Unity. Το αδρανειακό (Global) και το τοπικό (Local) τα οποία είναι αριστερόστροφα, δηλαδή ακολουθούν τον κανόνα του αριστερού χεριού. Και τα δυο ορίζουν την θέση και τον προσανατολισμό των Game Objects στην σκηνή του Unity.

Το τοπικό σύστημα συντεταγμένων είναι προσαρτημένο πάνω σε κάθε Game Object ενώ οι τιμές που αναγράφονται στο transform component του Game Object αφορούν την μετατόπιση από το σημείο αναφοράς του αδρανειακού συστήματος (0,0,0) και την περιστροφή του σε σχέση με αυτό.



Εικόνα 29: Μετατόπιση και περιστροφή των Game Objects ως προς το αδρανειακό σύστημα συντεταγμένων

Στην περίπτωση που το Game Object το οποίο εξετάζουμε είναι child ενός άλλου Game Object, τότε αυτό κληρονομεί την θέση και τον προσανατολισμό του αντικειμένου γονέα και οι όποιες τιμές στο transform component του αντικειμένου παιδιού, αφορούν την περιστροφή κι την μετατόπιση του ως προς το σύστημα συντεταγμένων του αντικειμένου γονέα.

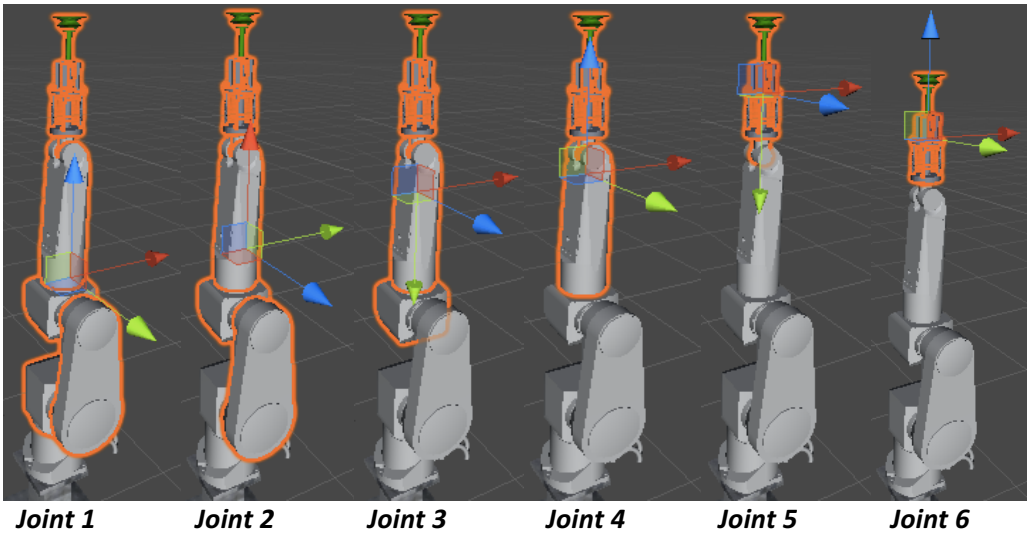


Εικόνα 30: Μετατόπιση και περιστροφή του κυλίνδρου ως προς το αντικειμένου γονέα του(κύβος)

Στην εικόνα 29 παρατηρούμε την μετατόπιση και την περιστροφή των δυο Game Objects της σκηνής ως προς το αδρανειακό σύστημα. Στην εικόνα 30 βλέπουμε ότι από τη στιγμή που ο κύλινδρος γίνεται παιδί του κύβου, το transform component του κυλίνδρου εκφράζεται ως συνάρτηση του transform του κύβου και όχι του αδρανειακού συστήματος. Αυτή η σχέση κληρονομικότητας μεταξύ των Game Objects θα μας φανεί ιδιαίτερα χρήσιμη κατά την ανάλυση της κινηματικής αλυσίδας του αρθρωτού ρομπότ, καθώς η μετατόπιση και η περιστροφή κάθε άρθρωσης θα πρέπει να εξαρτάται από τις αντίστοιχες παραμέτρους της προηγούμενης άρθρωσης.

Τα τοπικά συστήματα συντεταγμένων των αρθρώσεων του ρομπότ τοποθετήθηκαν έτσι ώστε όλες οι αρθρώσεις να περιστρέφονται πάντα γύρω από τον τοπικό τους άξονα Z. Κάθε ρομπότ στην σκηνή UnityARKitScene έχει προσαρτημένο το script component FKMover το οποίο καθορίζει την κίνηση του στον χώρο. Το Script FKMover λαμβάνει σαν είσοδο τις γωνίες περιστροφής κάθε άρθρωσης γύρω από τον τοπικό της άξονα Z. Η μετατόπιση του Τελικού Σημείου Δράσης στο σημείο στόχο, γίνεται με

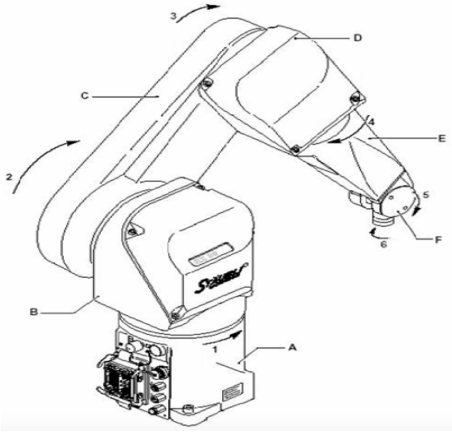
“σφαιρική” παρεμβολή, από τις γωνίες περιστροφής της κάθε άρθρωσης στην τρέχουσα θέση, έως τις γωνίες περιστροφής της κάθε άρθρωσης στην επόμενη θέση του ρομπότ. Η παρεμβολή μεταξύ των δυο σημείων του χώρου επιτυγχάνεται χρησιμοποιώντας την συνάρτηση Quaternion.Slerp.



Εικόνα 31: Τοπικά συστήματα συντεταγμένων των αρθρώσεων του ρομπότ Staubli RX90L

Η φορά περιστροφής των αρθρώσεων ήταν μια βασική παράμετρος προκειμένου το ρομπότ να κατευθυνθεί σωστά και με τον επιθυμητό τρόπο στο σημείο που του ορίσαμε ως στόχο.

Για αυτόν το λόγο οι κινήσεις των αρθρώσεων χωρίστηκαν σε δύο κατηγορίες: Αριστερόστροφες και Δεξιόστροφες. Στην πρώτη κατηγορία ανήκουν όλες οι κινήσεις κατά τις οποίες η άρθρωση περιστρέφεται γύρω από τον τοπικό άξονα Z με φορά από τον τοπικό άξονα Y (πράσινος άξονας) προς τον τοπικό άξονα X (κόκκινος άξονας) και τις θεωρούμε κατά σύμβαση θετικές, ενώ δεξιόστροφές είναι οι κινήσεις κατά τις οποίες η άρθρωση περιστρέφεται γύρω από τον τοπικό άξονα Z με φορά από τον τοπικό άξονα X προς τον τοπικό άξονα Y και τις θεωρούμε κατά σύμβαση αρνητικές. Στην παρακάτω εικόνα αναγράφεται με βέλος η θετική φορά κίνησης κάθε άρθρωσης (αριστερόστροφη κίνηση) στις οποίες βασιστήκαμε για τον σωστό υπολογισμό των περιστροφών τους.



Εικόνα 32: Θετική φορά κίνησης των αρθρώσεων του ρομπότ Staubli RX90L

Προκειμένου οι περιστροφές να γίνονται σύμφωνα με το σύστημα του Unity και με τον επιθυμητό τρόπο, και πάντα ως αποτέλεσμα ευθείας και όχι αντίστροφης κινηματικής ανάλυσης, αφού δίνουμε σαν είσοδο τις γωνίες των αρθρώσεων και όχι την θέση και τον προσανατολισμό του Τελικού Σημείου Δράσης, ο συνδυασμός γωνιών των αρθρώσεων στην επόμενη θέση υπολογίζεται ως εξής:

`rotationZiFrom−(Zi−rotationZiFrom)` : Για θετική περιστροφή της άρθρωσης
`−rotationZiFrom−(Zi+rotationZiFrom)` : Για αρνητική περιστροφή της άρθρωσης

όπου `rotationZiFrom` είναι η αρχική περιστροφή γύρω από τον τοπικό άξονα Z της άρθρωσης i, ενώ Zi είναι η περιστροφή γύρω από τον τοπικό άξονα Z της άρθρωσης i στην τελική θέση της άρθρωσης για i=1-6 για βιομηχανικό ρομπότ 6R το οποίο εξετάζεται στην παρούσα εργασία.

7.8. Διάταξη Κώδικα Κίνησης Εικονικών Μοντέλων

Σε αυτή την ενότητα αναλύονται τα script components τα οποία είναι υπεύθυνα για την κίνηση όλων των Game Objects στη σκηνή.

Τα διαθέσιμα Motion-scripts είναι τα εξής:

AnimationController: Το script αυτό είναι ο διαχειριστής της εφαρμογής σε επίπεδο user interface. Οι μέθοδοι του Workspace(), startProcess(), ShowAR() και SelectRobot() είναι υπεύθυνες για το τί θα γίνει στη σκηνή όταν ο χρήστης επιλέξει το αντίστοιχο UI Button.

Εικόνα 33: user interface της εφαρμογής για τον έλεγχο των Game Objects στη σκηνή

Στην update function και με την εντολή robotID=gameDataref.selectedRobotId; το script AnimationController γνωρίζει ανά πάσα στιγμή ποιο ρομπότ επέλεξε ο χρήστης της εφαρμογής στην πρώτη σκηνή (Scene0) αλλά και το αν αυτή η επιλογή του, άλλαξε στην παρούσα σκηνή με το Change Robot Button.

Το Show AR Button είναι υπεύθυνο για το αν θα εμφανίζεται το εικονικό περιεχόμενο στη σκηνή μας ή όχι. Αυτό έγινε κυρίως για λόγους εποπτείας του διαθέσιμου χώρου κατά την εκτέλεση της εφαρμογής.

Με το Start Process Button η βασισμένη στην επαυξημένη πραγματικότητα προσομοίωση λειτουργίας της μονάδας κατεργασίας ξεκινά. Πλέον το Change Robot Button και το Show AR Button καθίστανται ανενεργά προκειμένου να μην διακόπτεται η προσομοίωση της κατεργασίας. Πολύ σηματικές για την λειτουργία της εφαρμογής είναι οι μεταβλητές τύπου bool που παίρνουν τιμές true/false. Μέσα στην startProcess() μέθοδο παίρνει τιμή η bool μεταβλητή startTheProcessFlag η οποία δίνει εντολή στα script RawItemMotionManager και ReadyItemMotionManager να ξεκινήσουν.

Τέλος το Workspace Button εμφανίζει ή εξαφανίζει τον χώρο εργασίας του ρομπότ ο οποίος είναι ενα Game Object κατάλληλης γεωμετρίας και μεγέθους με invisible texture το οποίο μπορεί να μας δώσει εποπτεία κάθε πιθανού σημείου εμβέλειας του τελικού σημείου δράσης του ρομπότ.

Τα παραπάνω Buttons εισήχθησαν στο Canvas Game Object ακολουθώντας την διαδρομή GameObject/UI/Canvas.

Το **RawItemMotionManager**, το **ReadyItemMotionManager** και το **ASRSmotion** script είναι υπεύθυνα για τον συντονισμό όλων των κινήσεων στη σκηνή. Η λογική με την οποία κάθε Game Object κινείται βασίζεται κυρίως σε Coroutines (Παράρτημα Α) και αυτή περιγράφεται στα παραπάνω scripts. Οι λόγοι για τους οποίους η σύνταξη αυτής της λογικής έγινε σε 3 διαφορετικά scripts είναι δυο. Αφενός η ομαδοποίηση σε τρείς διακριτές ομάδες προσέφερε καλύτερο έλεγχο της εφαρμογής κατά την ανάπτυξη της, αφετέρου η φύση των Coroutines επιτάσσει την ολοκλήρωση ενός Coroutine προκειμένου να ξεκινήσει το επόμενο. Η δημιουργία ενός μόνο αντί τριών scripts θα επέφερε μια πολύ διακριτή ακολουθία κινήσεων εξαιτίας αυτής της δυσκολίας επικάλυψης των κινήσεων με Coroutines ή πιο απλά δεν θα μπορούσαμε να έχουμε κίνηση δυο Game Objects την ίδια χρονική στιγμή.

RawItemMotionManager: Είναι το script που διαχειρίζεται την κίνηση του ακατέργαστου τεμαχίου, του πρώτου εικονικού ρομπότ (κατά την ροή της κατεργασίας) και του κατεργασμένου τεμαχίου μέχρι αυτό να τοποθετηθεί στον δεύτερο ταινιόδρομο. Η εκκίνηση του συγκεκριμένου script γίνεται όταν ο χρήστης επιλέξει το Start Process Button. Έτσι το RawItemMotionManager ενημερώνεται με την εντολή startTheProcessFlag = animationController.startTheProcessFlag; για την τιμή της μεταβλητής startTheProcessFlag οπότε και ξεκινά μόλις πάρει την τιμή true η παρακάτω συνθήκη:

```
yield return new WaitUntil(() => startTheProcessFlag == true);
```

Επιπλέον στο script RawItemMotionManager βρίσκονται οι μέθοδοι ItemLoading() και ItemUnLoading() με τις οποίες το τεμάχιο γίνεται παιδί του τελικού σημείου δράσης του ρομπότ και της εργαλειομηχανής αντίστοιχα.

ReadyItemMotionManager: Είναι το script που διαχειρίζεται την κίνηση του κατεργασμένου τεμαχίου, του δεύτερου εικονικού ρομπότ (κατά την ροή της κατεργασίας), και της φόρτωσης κενών δίσκων και των κατεργασμένων τεμαχίων πάνω σε αυτούς, στο σημείο παραλαβής (Table Game Object). Η εκκίνηση του συγκεκριμένου script γίνεται επίσης όταν ο χρήστης επιλέξει το Start Process Button όπως στην περίπτωση του RawItemMotionManager.

Όμως προκειμένου να συγχρονίσουμε τις κινήσεις αυτού του script με το RawItemMotionManager, εισάγουμε την bool μεταβλητή firstItemIsReadyAndPositioned ή οποία θα πάρει τιμή true μόλις το κατεργασμένο τεμάχιο μεταφερθεί από το πρώτο εικονικό ρομπότ στον δεύτερο ταινιόδρομο. Με αυτό τον τρόπο υπάρχει μία φυσιολογική ροή των εργασιών και των κινήσεων στο σύστημα.

Οι εντολές οι οποίες είναι υπεύθυνες για την καθυστέρηση εκτέλεσης των Coroutines μέχρι να ικανοποιηθούν οι αντίστοιχες συνθήκες είναι:

```
yield return new WaitUntil(() => startTheProcessFlag == true);
yield return new WaitUntil(() => firstItemIsReadyAndPositioned == true);
```

Επιπλέον στο script ReadyItemMotionManager βρίσκονται οι μέθοδοι ItemLoading(), ItemUnLoading(), BringTray(), ReleaseTray(), CreateNewTray(). Με τις δυο πρώτες πετυχαίνουμε το τεμάχιο να φορτωθεί (ως παιδί) στο τελικό σημείο δράσης του δεύτερου εικονικού ρομπότ και του 'ενεργού' δίσκου αντίστοιχα. Ο δίσκος είναι κενός και σε αναμονή στην θέση παραλαβής. Οι επόμενες δυο μέθοδοι έχουν ανάλογη λειτουργία αλλά αντί του τεμαχίου μεταφέρουν τον επόμενο κενό δίσκο στην θέση παραλαβής αφού πρώτα γίνει instantiate με την μέθοδο CreateNewTray().

Τα script **RawItemMotion**, **ReadyItemMotion**, **FKMover2** και **FKMover3** είναι τα script στα οποία βρίσκονται όλες οι Coroutines των δύο παραπάνω στα οποία γίνεται απλά η κλήση αυτών των Coroutines. Στόχος ήταν τα ReadyItemMotionManager και RawItemMotionManager να περιέχουν μόνο οδηγίες του πώς θα προγραμματιστεί η μονάδα κατεργασίας και όχι με ποιόν τρόπο η με ποια προγραμματιστικά εργαλεία αυτό θα υλοποιηθεί. Γι΄ αυτόν το λόγο προέκυψαν τα τέσσερα αυτά scripts που περιγράφονται παρακάτω.

RawItemMotion: Εδώ βρίσκονται οι Coroutines οι οποίες είναι υπεύθυνες για την δημιουργία και την κίνηση του ακατέργαστου τεμαχίου στην σκηνή και συγκεκριμένα πάνω στον πρώτο εικονικό ταινιόδρομο.

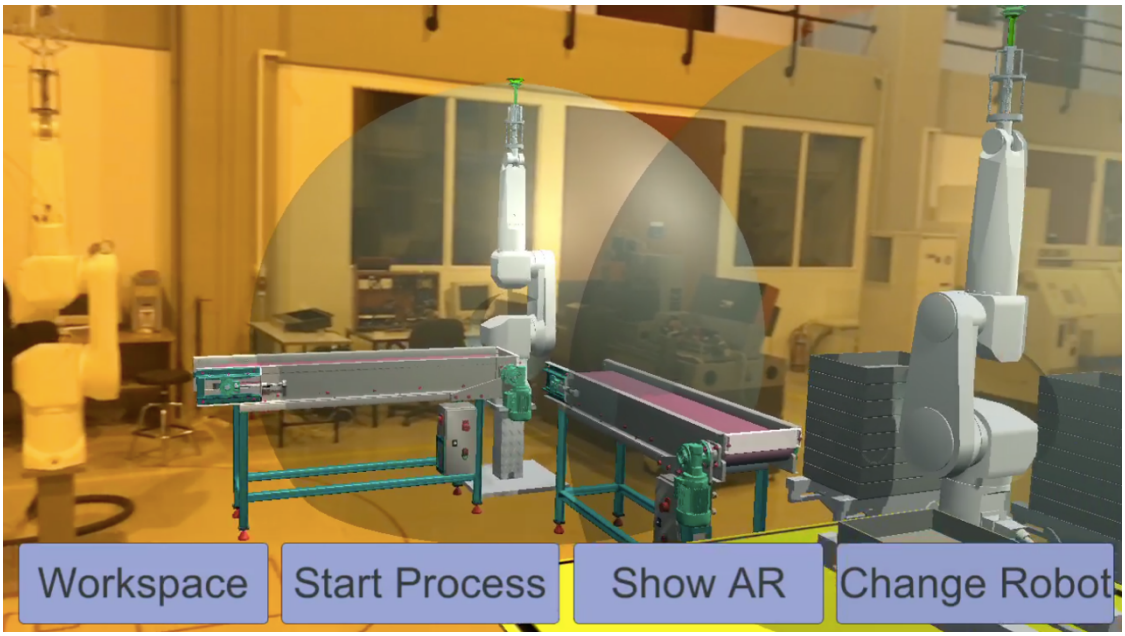
ReadyItemMotion: Στο script ReadyItemMotion βρίσκονται οι Coroutines με τις οποίες δημιουργείται και κινείται το κατεργασμένο τεμάχιο στην σκηνή. Αξίζει να σημειωθεί ότι η Coroutine CreateItemPrefab η οποία είναι υπεύθυνη για την δημιουργία του κατεργασμένου τεμαχίου στη σκηνή, έχει γίνει overload προκειμένου όταν γίνει η κλήση της από το script RawItemMotionManager να μπορεί να δημιουργήσει το κατεργασμένο τεμάχιο ακριβώς στην ίδια θέση και προσανατολισμό με το ακατέργαστο τεμάχιο. Αυτό γίνεται προκειμένου στο τέλος της προσομοίωσης της κατεργασίας να έχουμε μέσα στην εργαλειομηχανή το εικονικό κατεργασμένο τελικό τεμάχιο και όχι το ακατέργαστο το οποίο αρχικά τοποθετήσαμε.

FKMover2 και **FKMover3**: Όπως αναφέρθηκε και πιο πάνω, η κίνηση των ρομπότ απο ένα σημείο σε ενα άλλο βασίστηκε στη "σφαιρική" παρεμβολή μέσω της συνάρτησης SLerp της coroutine GoToTarget. Έτσι τα scripts FKMover2 και FKMover3 υλοποιούν αποκλειστικά τις κινήσεις μεταξύ αρχικής και τελικής θέσης κάθε άρθρωσης βάσει της coroutine GoToTarget.

ASRSmotion: Σε αντίθεση με τα παραπάνω, ο έλεγχος της κίνησης του ASRS καθώς και οι Coroutines που την υλοποιούν βρίσκονται στο script ASRSmotion.

Οι βασικές coroutines του script ASRSmotion ειναι οι MoveObject() και MoveCarrierOnAxisZ() όπως αναλύθηκε πιο πάνω, με τις υπόλοιπες να έχουν συμπληρωματικό ρόλο καθορίζοντας μόνο τα σημεία έναρξης και τερματισμού μίας κίνησης καθώς και την διάρκεια της.

Και στην περίπτωση του ASRSmotion script, προκειμένου να υπάρχει επικοινωνία μεταξύ των τριών script "διαχειριστών" χρησιμοποιείται η τιμή της bool μεταβλητής startTheProcessFlag την οποία ρυθμίζει το RawItemMotionManager script ακριβώς όπως και παραπάνω.



Εικόνα 34: Εικονικός και πραγματικός εξοπλισμός εργαστηρίου κατεργασιών βάσει της χωροταξικής διάταξης που επιλέχθηκε.

8. Υλοποίηση

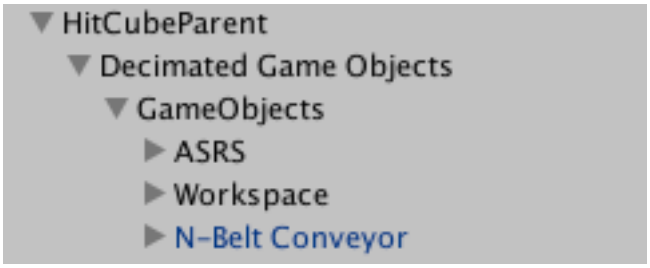
8.1. Χρήση ARKit στο Περιβάλλον του Unity

Για την χρήση του ARKit SDK, έγινε λήψη του ARKit Plugin από το Asset Store του Unity [51].

Στο Project panel του Unity, στον φάκελο Assets και ακολουθώντας τη διαδρομή UnityARKitPlugin/Examples/UnityARKitScene είναι διαθέσιμη η σκηνή UnityARKitScene πάνω στην οποία είναι δυνατή η δημιουργία εφαρμογών επαυξημένης πραγματικότητας βασισμένων στο ARKit.

Από τα διαθέσιμα Game Objecs του Hierarchy Panel μετά την εισαγωγή του ARKit, 2 είναι κυρίως αυτά που μας χρειάζονται. Αυτά είναι το HitCubeParent και το ARCameraManager Game Objects ενώ τα Game Objects Directional light και CameraParent ενεργοποιούνται από το StateManager script ταυτόχρονά με την ενεργοποίηση της σκηνής UnityARKitScene προκειμένου να είναι πιο εύκολη η διαχείριση των διαφορετικών καμερών και φωτισμού σε κάθε σκηνή.

Τα 3D μοντέλα της βασικής σκηνής του ARKit δηλαδή της σκηνής UnityARKitScene είναι ομαδοποιημένα σε ένα κενό Game Object γονέα με όνομα Decimated Game Objects στον υποφάκελο Game Objects.



Εικόνα 35*: Κληρονομικότητα αντικειμένων μετά την εισαγωγή του ARKit Plugin στο Unity*

Αυτό το Game Object γονέας το κάναμε τέκνο του HitCubeParent Game Object αφού πρώτα διαγράψαμε τον κύβο που είχε by default ως τέκνο του. Σε αυτό το Game Object προσθέσαμε το component script με όνομα UnityARHitTestExample ενώ στην public μεταβλητή Hit Transform του script δώσαμε την τιμή του HitCubeParent Game Object κάνοντας Drag and Drop το Game Object στο αντίστοιχο slot του inspector panel του Unity.

Το UnityARHitTestExample script είναι υπεύθυνο για την τοποθέτηση των εικονικών μας μοντέλων, πάνω σε οριζόντιες επιφάνειες τις οποίες θα εντοπίζει η κάμερα του κινητού μας με την βοήθεια του ARKit.

Η UnityARKitScene θα μετακινήσει το περιεχόμενο του Decimated Game Objects σε οποιοδήποτε σημείο (ARAnchor) ακουμπήσει το δάκτυλο του χρήστη στην οθόνη του κινητού. Αυτό περιλαμβάνει τόσο τις επίπεδες επιφάνειες όσο και τα τυχαία σημεία πράγμα το οποίο σημαίνει ότι τα μοντέλα θα τοποθετηθούν στον αέρα. Για να είναι πάντα το εικονικό περιεχόμενο της σκηνής τοποθετημένο σε

ένα ανιχνευμένο επίπεδο και ποτέ στον αέρα, τροποποιήσαμε το UnityARHitTestExample script προκειμένου να δηλώσουμε ότι μας ενδιαφέρουν μόνο τα αγγίγματα στις εντοπισμένες από το GeneratePlanes game object επιφάνειες.

Για να επιτευχθεί αυτό, στη μέθοδο Update () του UnityARHitTestExample script , μετατράπηκαν σε σχόλια όλες οι δηλώσεις εκτός της ARHitTestResultTypeExistingPlaneUsingExtent όπως φαίνεται στην παρακάτω εικόνα:

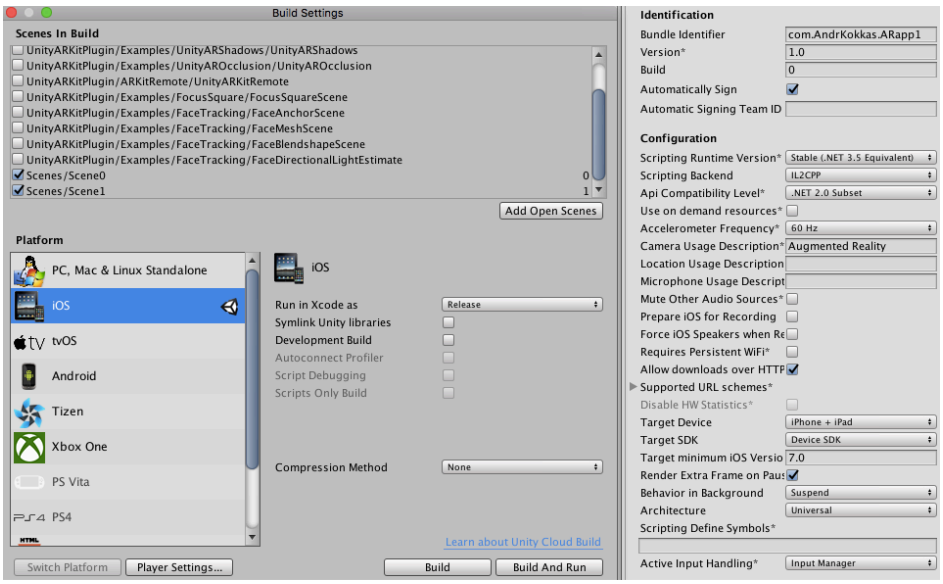
```
ARHitTestResultType[] resultTypes = {
    ARHitTestResultType.ARHitTestResultTypeExistingPlaneUsingExtent,
    // if you want to use infinite planes use this:
    //ARHitTestResultType.ARHitTestResultTypeExistingPlane,
    //ARHitTestResultType.ARHitTestResultTypeHorizontalPlane,
    //ARHitTestResultType.ARHitTestResultTypeFeaturePoint
};
```

Εικόνα 36: Επιλογή μόνο των εντοπισμένων οριζόντιων επιφανειών για υπέρθεση των εικονικών μοντέλων.

8.2. Ανάπτυξη Εφαρμογής για iOS

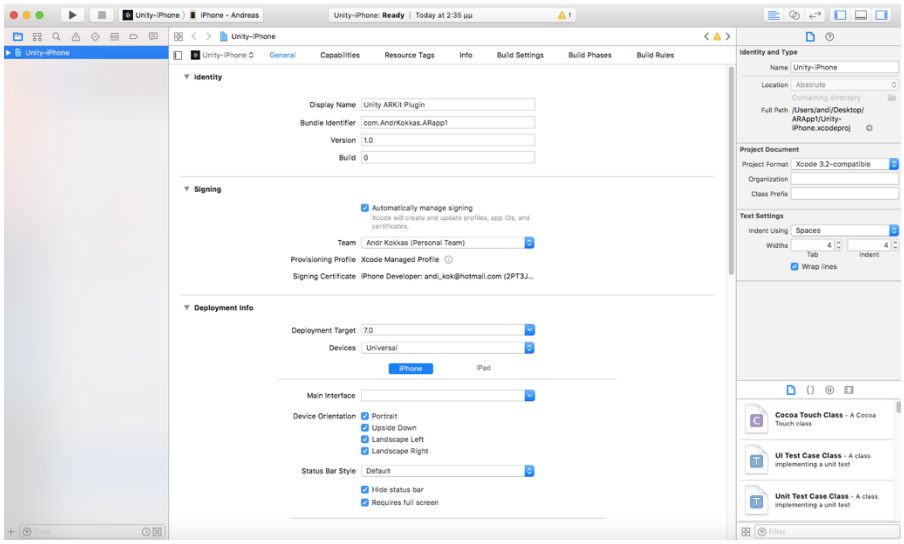
Στο Hierarchy panel του Unity υπάρχει το HitCubeParent Game Object το οποίο είναι αντικείμενο γονέας όλων των μοντέλων που εμφανίζονται στην σκηνή UnityARKitScene. Το script UnityARHitTestExample είναι component script του HitCubeParent οπότε όλα τα Game Objects τα οποία είναι τέκνα του, λόγω κληρονομικότητας, έχουν επίσης όλες τις ιδιότητες που προσδίδει το UnityARHitTestExample script.

Αφού προσθέσαμε στη σκηνή UnityARKitScene τα 3D μοντέλα, πηγαίνουμε στην καρτέλα Build Settings και επιλέγουμε Player settings/ Other settings προκειμένου να συμπληρώσουμε το Bundle Identifier στην κατηγορία Identification - com.yourName.yourARapp , και το Camera Usage Description στην κατηγορία Camera Usage Description -Augmented Reality. Έτσι και αφού πρώτα δώσουμε ένα όνομα στο Project μας μπορούμε να κάνουμε Build. Προσοχή πρέπει να δοθεί επίσης και στην σειρά με την οποία εμφανίζονται οι σκηνές σε αυτό το παράθυρο γιατί είναι ακριβώς η σειρά με την οποία θα εμφανίζονται και στην εφαρμογή.



Εικόνα 37: Build Project για iOS

Προκειμένου να κάνουμε ‘’Deploy’’ την εφαρμογή (ARobot Layout) για iOS, μέσα από το περιβάλλον του Xcode, ανοίξαμε το αρχείο που μόλις δημιουργήσαμε και ακολουθώντας τη διαδρομή Identity/ Bundle Identifier συμπληρώσαμε --- com.yourName.yourARapp και ακολουθώντας τη διαδρομή Signing/ Add account επιλέξαμε ομάδα --- Your Name (Personal Team). Έτσι είμαστε έτοιμοι και αφού συνδέσουμε μια συμβατή με το ARKit συσκευή την επιλέγουμε στην καρτέλα Unity-iPhone προκειμένου να εγκατασταθεί η εφαρμογή σε αυτήν.



Εικόνα 38: Ανάπτυξη της εφαρμογής iOS μέσα από Xcode

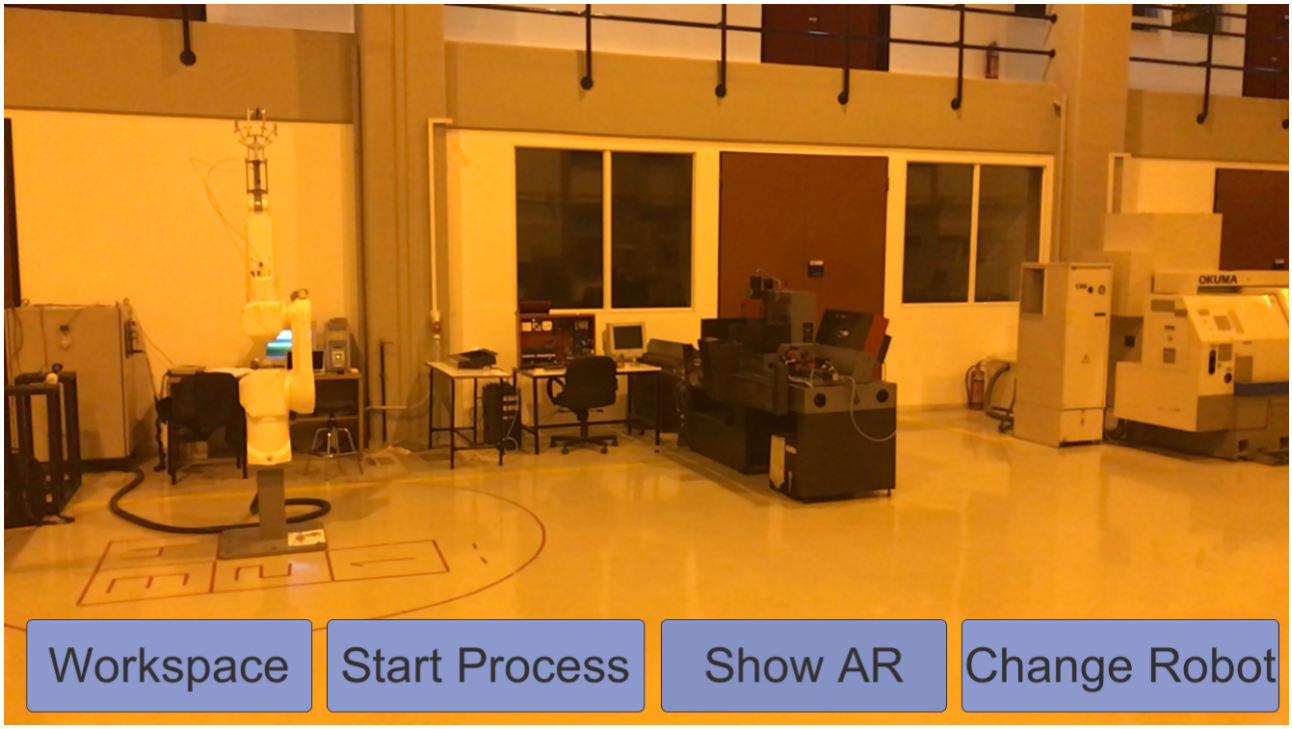
8.3. Παρουσίαση της Εφαρμογής ARobot_Layout

8.3.1. Περιγραφή

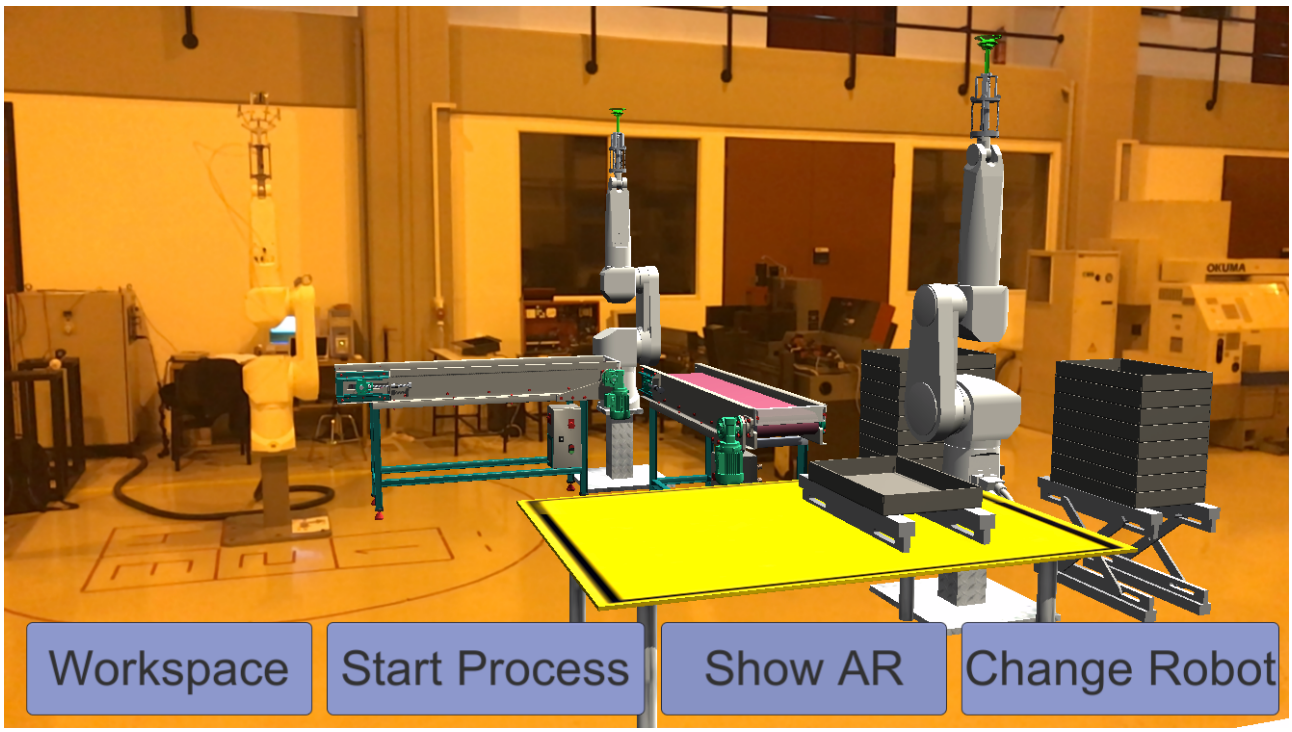
Το ARobot_Layout είναι μια iOS εφαρμογή επαυξημένης πραγματικότητας που μπορεί να χρησιμοποιηθεί για την επιλογή της βέλτιστης χωροταξικής διάταξης (FLP) μηχανουργικού εξοπλισμού αλλά και την προσομοίωση της παραγωγικής διαδικασίας.

Το πλεονέκτημα της είναι ότι χρησιμοποιεί Visual Inertial Odometry μέσω του ARKit Plugin της Apple οπότε είναι ουσιαστικά απαλλαγμένη από τον εντοπισμό εικόνων-στόχων ενώ ταυτόχρονα έχει στηθεί με τέτοιο τρόπο ώστε να δίνεται η ψευδαίσθηση στον χρήστη ότι ο υφιστάμενος εξοπλισμός του εργαστηρίου, συνεργάζεται με τον εικονικό εξοπλισμό που προσθέτουμε μέσω της εφαρμογής.

Ο χρήστης της εφαρμογής ARobot_Layout είναι σε θέση να επιλέξει εκτός από χωροταξική διάταξη, και κάποιο από τον βασικό εξοπλισμό και τέλος είναι σε θέση να διενεργήσει προσομοίωση της παραγωγικής διαδικασίας.



Εικόνα 39: Εργαστήριο Τεχνολογίας των Κατεργασιών (διακρίνονται τόρνος EMCO και ρομπότ Staubli)



Εικόνα 40: Υπέρθεση εικονικού εξοπλισμού στο Εργαστήριο Τεχνολογίας των Κατεργασιών

8.3.2. Χρονικά Χαρακτηριστικά του Συστήματος που Προέκυψε

Στον πίνακα που ακολουθεί φαίνονται οι βασικότερες λειτουργίες του συστήματος κατεργασίας επαυξημένης πραγματικότητας, καθώς και η διάρκεια τους. Επίσης φαίνεται και η συνολική διάρκεια κάθε κύκλου κατεργασίας (χρόνος από την παραλαβή του ακατέργαστου τεμαχίου μέχρι την αποθήκευση του κατεργασμένου στο σύστημα ASRS). Το ρομπότ του εργαστηρίου κινείται με το 30% της ονομαστικής του ταχύτητας.

Εκκίνηση με	Θέση	Σύμβολο	Στοιχείο FMS	Φύση Στοιχείου	Κίνηση	Χρόνος(sec)	Διάρκεια κύκλου(sec)
	1	R1	1ο Ρομπότ (6R)	Πραγματικό	Από θέση "Catch" έως τη θέση "Release"	14	
					Από θέση "Release" έως την επόμενη φορά που θα είναι στην ίδια θέση	42	56
	2	C1	1ος Ταινιόδρομος	Εικονικό	Μεταφορά ακατέργαστου τεμαχίου	10	
	3	R2	2ο Ρομπότ (6R)	Εικονικό	Φόρτωση τεμαχίου	1	
					Μεταφορά τεμαχίου στην εργαλειομηχανή(1ο τμήμα)	2	
					Μεταφορά τεμαχίου στην εργαλειομηχανή(2ο τμήμα)	3	
					Απομάκρυνση ρομπότ για επεξεργασία τεμαχίου	2-δεν λαμβάνεται υπόψιν	
					Προσέγγιση εργαλειομηχανής για φόρτωση τεμαχίου	2-δεν λαμβάνεται υπόψιν	
					Αναμονή για φόρτωση	1	
					Μεταφορά τεμαχίου στον 2ο ταινιόδρομο(1ο τμήμα)	2	
					Επιστροφή στην θέση αναμονής	5	
Flag					Μεταφορά τεμαχίου στον 2ο ταινιόδρομο(2ο τμήμα)	3	
	4	L	Τόρνος-Φρέζα	Πραγματικό	Επεξεργασία τεμαχίου	29	56
	5	C2	2ος Ταινιόδρομος	Εικονικό	Μεταφορά έτοιμου τεμαχίου	10	
Flag	6	R2	3ο Ρομπότ (6R)	Εικονικό	Φόρτωση τεμαχίου	1	
					Μεταφορά τεμαχίου στο κιβώτιο αποθήκευσης(1ο τμήμα)	2	
					Μεταφορά τεμαχίου στο κιβώτιο αποθήκευσης(2ο τμήμα)	3	16
Flag	7	ASRS	Σύστημα Αποθήκευσης	Εικονικό	Απο αρχική θέση έως το σημείο φόρτωσης (Z άξονας)	2	
					Απο σημείο φόρτωσης (Z άξονας) έως αρχική θέση	2	
					Απο το σημείο φόρτωσης (X&Y άξονας) έως την αρχική θέση	10	14
						Διάρκεια Κύκλου:	**86**

Πίνακας: Λειτουργίες του AR συστήματος κατεργασίας, η διάρκεια τους και η διάρκεια κάθε κύκλου κατεργασίας.

Στον παραπάνω πίνακα φαίνεται η διάρκεια της συνολικής διαχείρισης κάθε τεμαχίου από το εν λόγω σύστημα κατεργασίας μέχρι την τελική του αποθήκευση. Έτσι παρατηρούμε ότι κάθε κύκλος χρειάζεται περίπου 86 δευτερόλεπτά προκειμένου να ολοκληρωθεί. Ο συγχρονισμός εικονικών και πραγματικών μοντέλων επιλέχθηκε να γίνει κατά την στιγμή παραλαβής του τεμαχίου από το

πραγματικό ρομπότ του εργαστηρίου. Έτσι, με το πάτημα του Start Process UI button μέσα την εφαρμογή, την χρονική στιγμή που η αρπάγη του ρομπότ πιάνει το τεμάχιο, πετυχαίνουμε συγχρονισμό των κινήσεων των εικονικών και πραγματικών μοντέλων. Μέχρι τότε όλα τα εικονικά μοντέλα είναι σε αναμονή. Το ρομπότ απαιτεί περίπου 14 δευτερόλεπτα για να μεταφέρει το τεμάχιο με ταχύτητα 30% της ονομαστικής του, ενώ αφήνει ένα τεμάχιο στον εικονικό ταινιόδρομο ανά 56 δευτερόλεπτα.

8.3.3. Οι Σκηνές της Εφαρμογής και η Λειτουργία της

Ο έλεγχος των σκηνών της εφαρμογής γίνεται μέσα από το script StateManager. Σε κάθε frame το StateManager script ελέγχει ποια state είναι ενεργή και της αναθέτει τον έλεγχο προκειμένου αυτή να εμφανίσει την αντίστοιχη σκηνή και ότι είναι προγραμματισμένο σε αυτήν. Επίσης το StateManager script δίνει τον έλεγχο στις μεθόδους StateUpdate(), ShowIt() της ενεργής state. Η εφαρμογή αποτελείται απο δυο σκηνές και τρείς states:

States	Σκηνές/State	Χρήση
BeginState	Scene0	Εισαγωγική state-Έναρξη της εφαρμογής
SetupState	Scene0	State επιλογής εξοπλισμού
PlayStateFirstPosition	UnityARKitScene	State εκτέλεσης της εφαρμογής ARobot_Layout

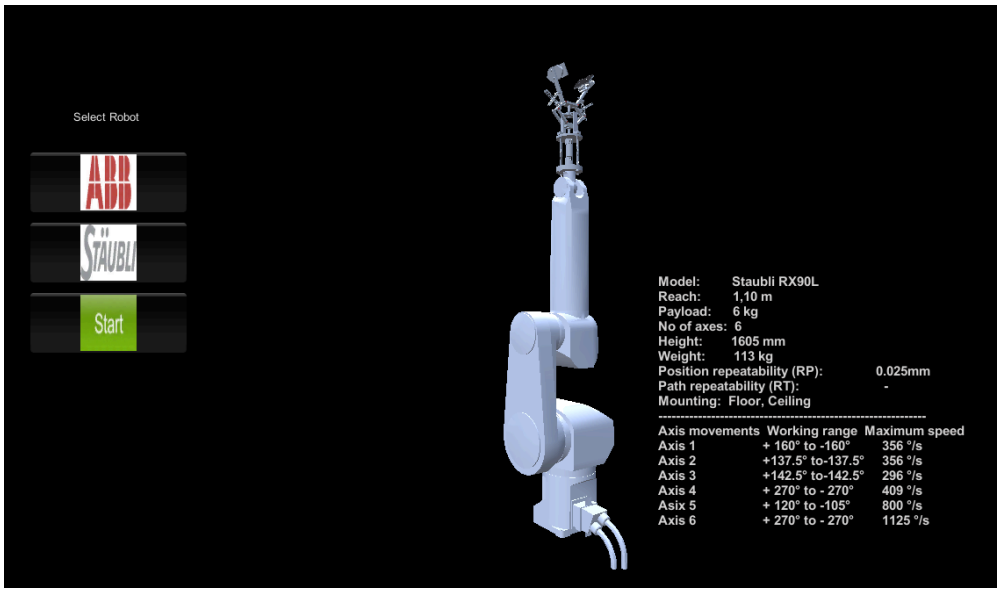
Πίνακας: Διάταξη Σκηνών και State της εφαρμογής ARobot_Layout

Κάθε state αποφασίζει ποια σκηνή θα εμφανίσει σε κάθε frame μέσα από τις εντολές:

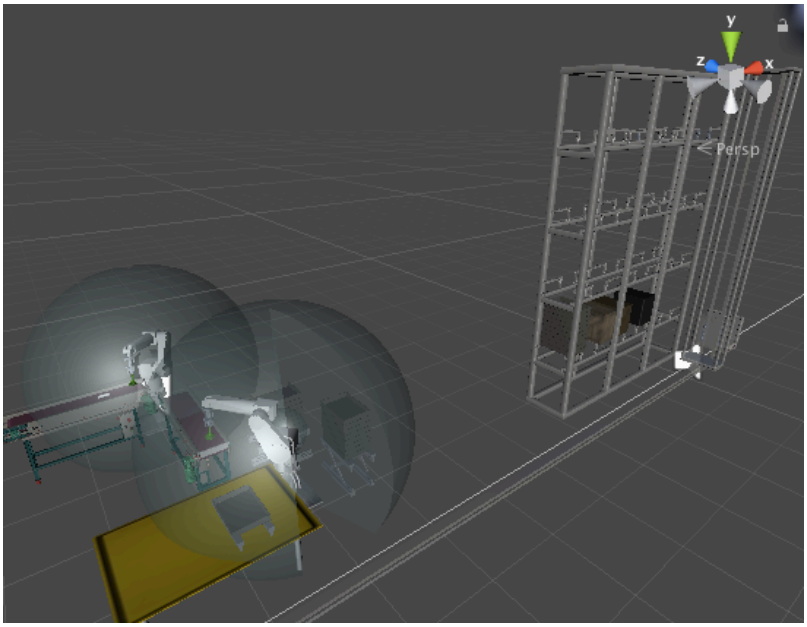
```
if (Application.loadedLevelName != "SceneName")
Application.LoadLevel ("SceneName");
```

Επίσης με αντίστοιχο τρόπο η κάθε state θα αποφασίσει για το ποια camera θα είναι ενεργή στην αντίστοιχη σκηνή, επιλέγοντας την μέσα από μια λίστα η οποία είναι αποθηκευμένη στο StateManager script.

Εικόνα 41: BeginState/Scene0 - Έναρξη εφαρμογής ARobot_Layout

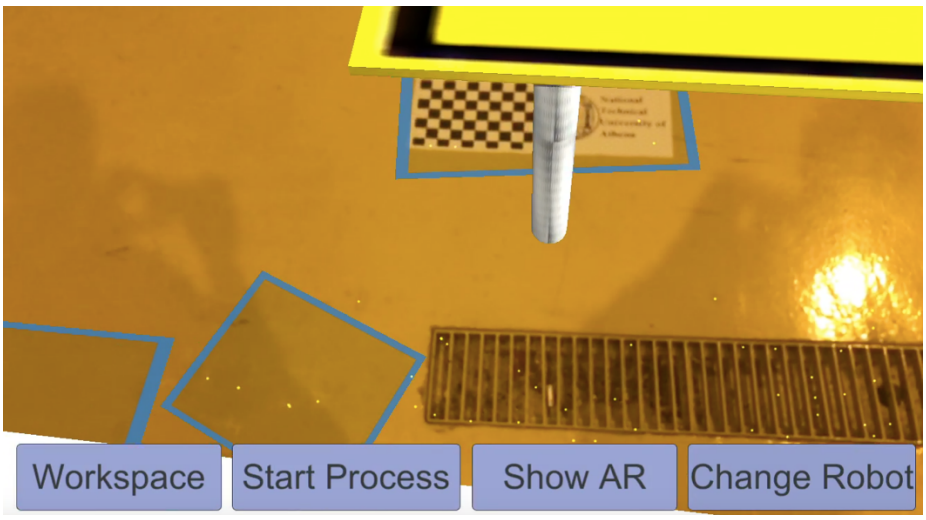


Εικόνα 42: PlayStateFirstPosition/UnityARKitScene



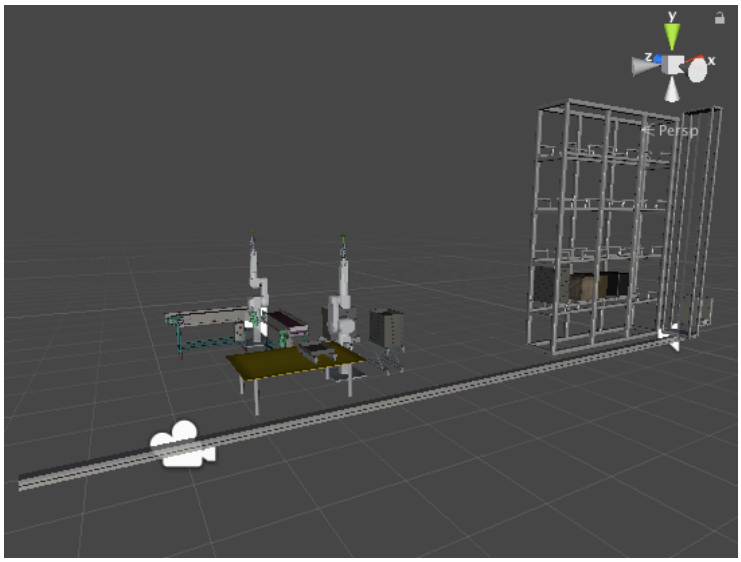
Εικόνα 43: PlayStateFirstPosition/UnityARKitScene

Τέλος αξίζει να σημειωθεί ότι όταν ξεκινήσει η UnityARKitScene θα πρέπει να επιλεγεί πάνω σε ποιο plane θα εμφανίσουμε το εικονικό περιεχόμενο. Δεδομένου ότι αρχικά αναζητούμε ένα plane το οποίο εφάπτεται του δαπέδου, το ARKit μας παρέχει κάποιες επιλογές όπως φαίνεται και παρακάτω.



Εικόνα 44: Εντοπισμένα ARKit Planes στο δάπεδο του εργαστηρίου κατεργασιών

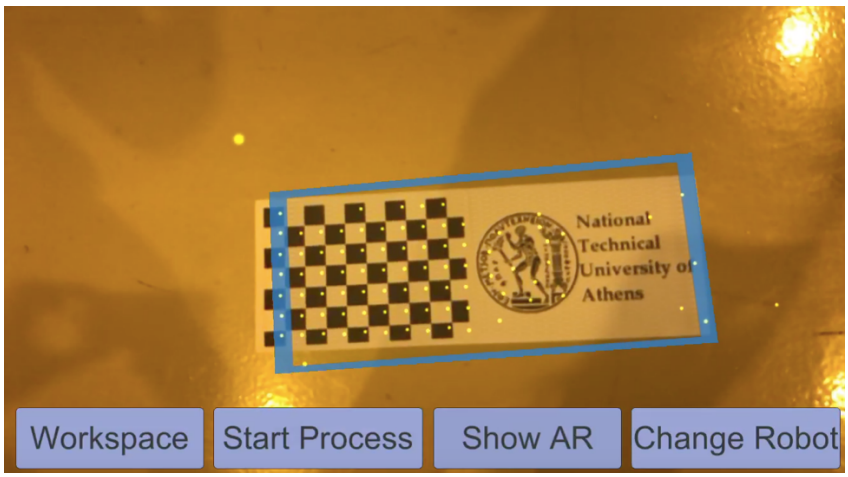
Επειδή η ARCamera της σκηνής UnityARKitScene τοποθετήθηκε στο σημείο 0 (X,Y,Z=0) το σημείο που θα επιλέξουμε πάνω σε κάποιο από τα planes που το ARKit εντοπίζει, θα αντιστοιχεί στο σημείο 0. Γι αυτόν το λόγο θα πρέπει να επιλεγεί ένα συγκεκριμένο σημείο, και πόσο μάλλον σε ένα συγκεκριμένο plane προκειμένου να γίνει σωστή αρχικοποίηση και τα εικονικά μοντέλα να έχουν τη σωστή τοποθέτηση και τις σωστές αποστάσεις σε σχέση με τον πραγματικό εξοπλισμό.



Εικόνα 45: Κάτω αριστερά φαίνεται η ARCamera, που συμπίπτει με το σημείο 0 της σκηνής

Επίσης αν το plane για παράδειγμα έχει περιστραφεί κατά 45 μοίρες, τότε όλο το εικονικό περιεχόμενο της σκηνής θα έχει περιστραφεί μαζί του, χαλώντας τις σχέσεις που επιθυμούμε να δημιουργήσουμε μεταξύ εικονικού και πραγματικού εξοπλισμού και παίρνοντας μια διάταξη μακριά από την επιθυμητή.

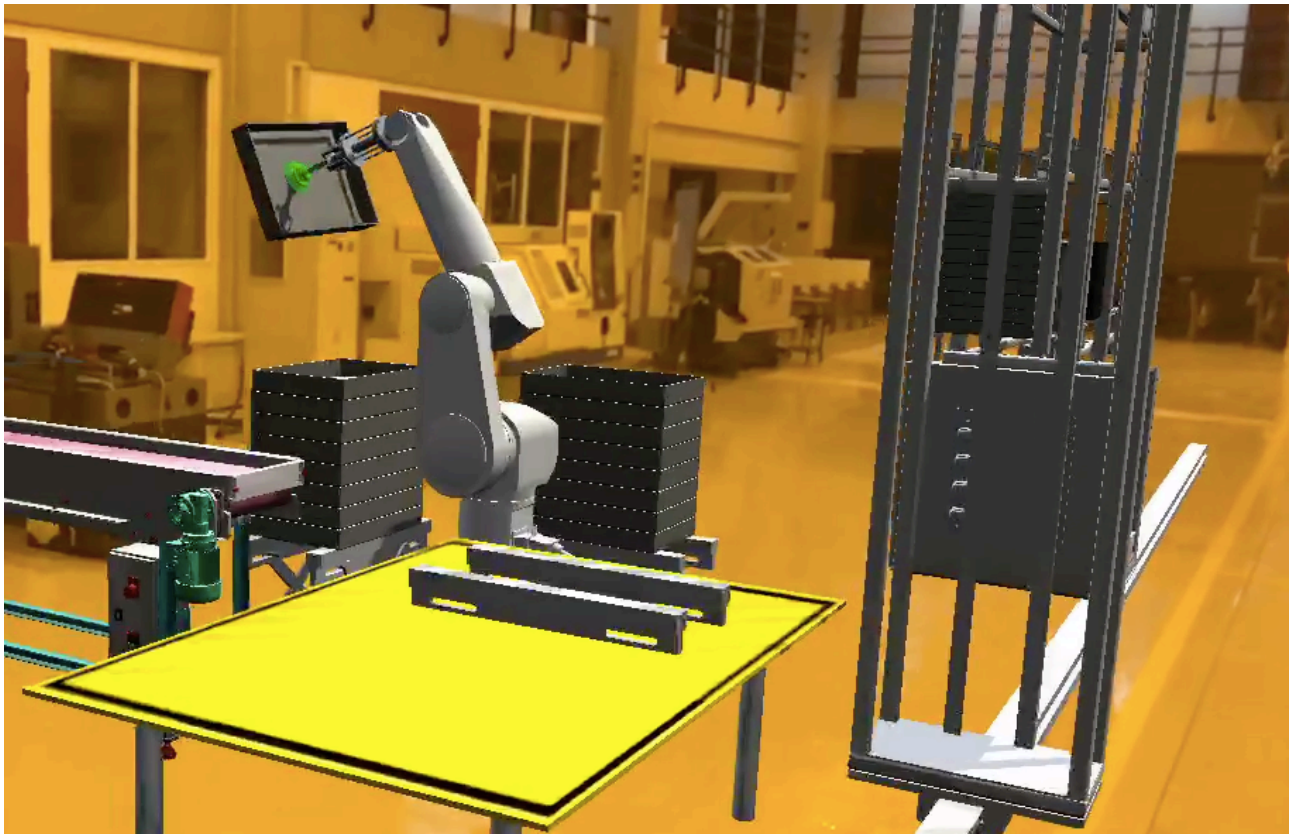
Αυτό αποτέλεσε ένα κομμάτι της παρούσας εργασίας το οποίο λύθηκε με σύμβαση. Προσπαθήσαμε δηλαδή με κάποιες εικόνες με έντονες αντιθέσεις και παραλληλόγραμμο layout να προσδιορίσουμε και τον προσανατολισμό και την θέση του plane όπως φαίνεται στην ακόλουθη εικόνα.



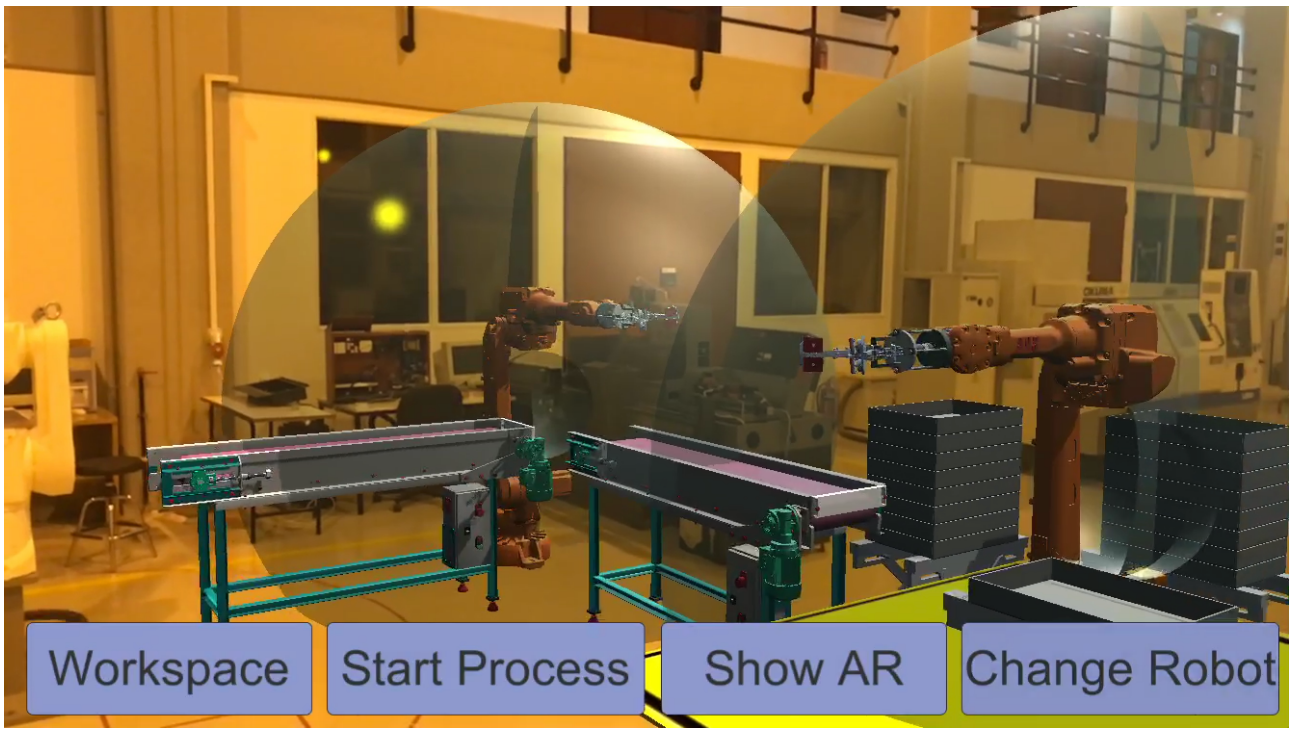
Εικόνα 46: Επιθυμητή χωροθέτηση του ARKit Plane στην σκηνή

Έτσι το σημείο του plane το οποίο θα ακουμπήσουμε με ένα άγγιγμα επι της οθόνης της συσκευής μας θα είναι και το σημείο 0 της σκηνής επαυξημένης πραγματικότητας.

Παρακάτω φαίνονται κάποια χαρακτηριστικά στιγμιότυπα από την λειτουργία της μονάδας κατεργασιών (FMS) επαυξημένης πραγματικότητας, που προέκυψε στο Εργαστήριο Τεχνολογίας των Κατεργασιών του ΕΜΠ.

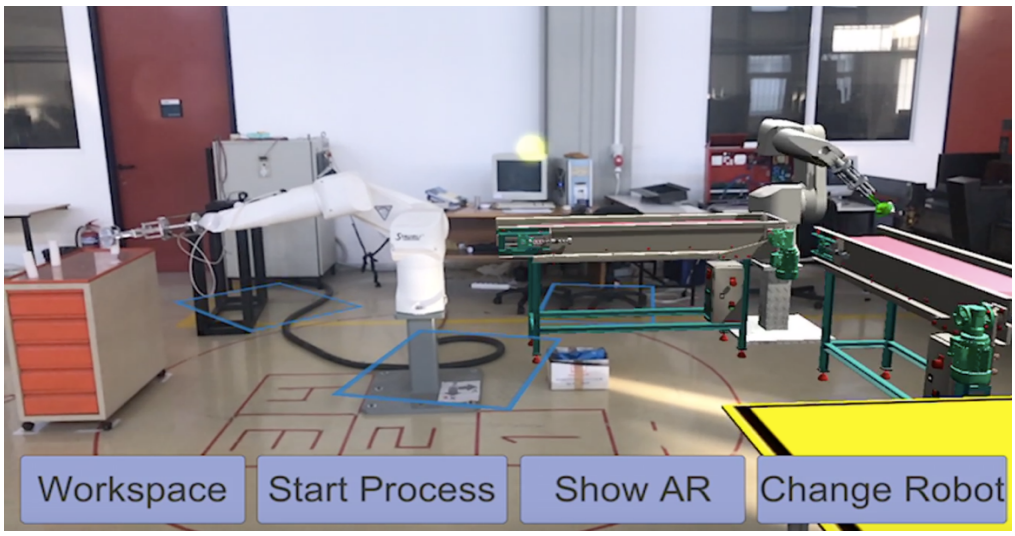


Εικόνα 47: Φόρτωση της θέσης παραλαβής με έναν κενό δίσκο

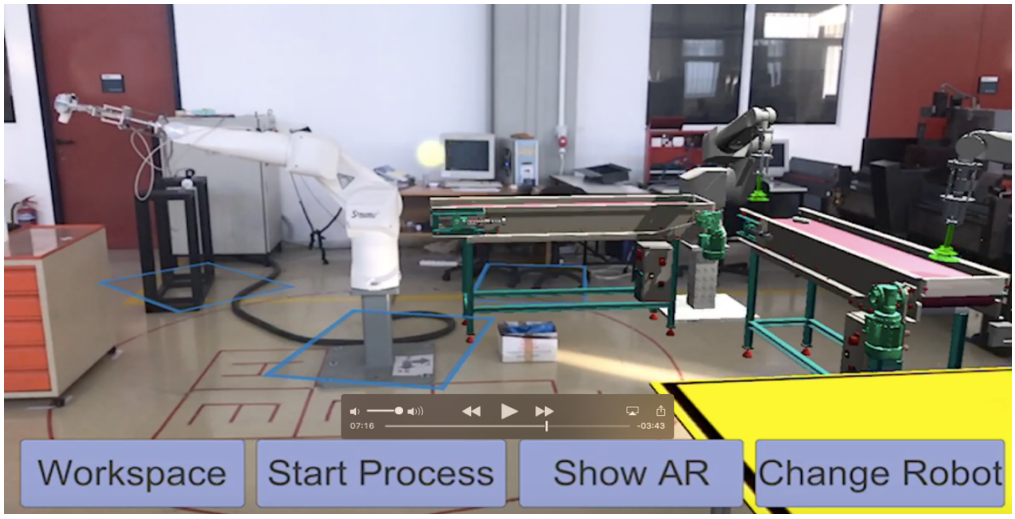


Εικόνα 48: ABB IRB 2600 στην μονάδα κατεργασίας του εργαστηρίου Κατεργασιών

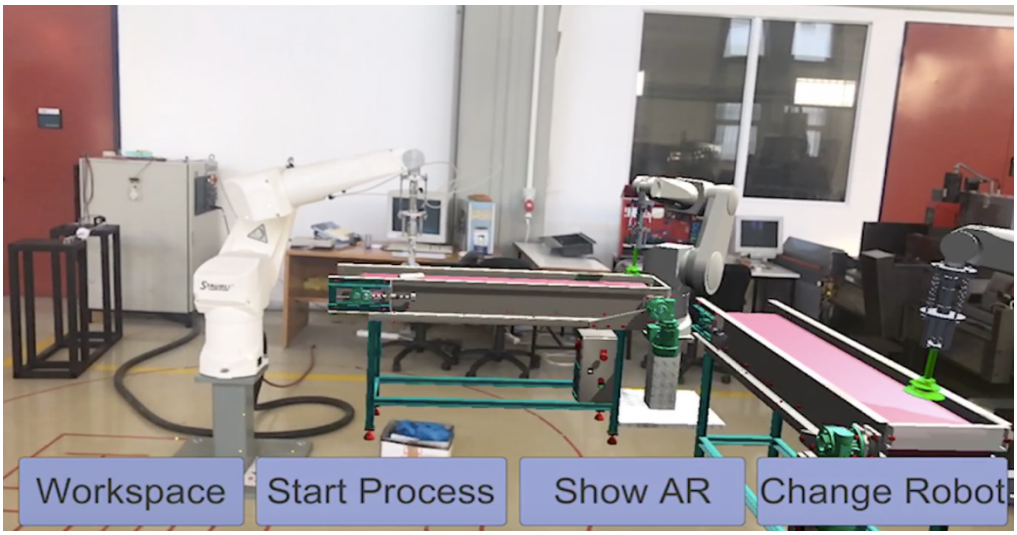
Εικόνα 49: Σύστημα Αυτόματης Αποθήκευσης και Ανάκτησης (ASRS) στο Εργαστήριο Τεχνολογίας των Κατεργασιών



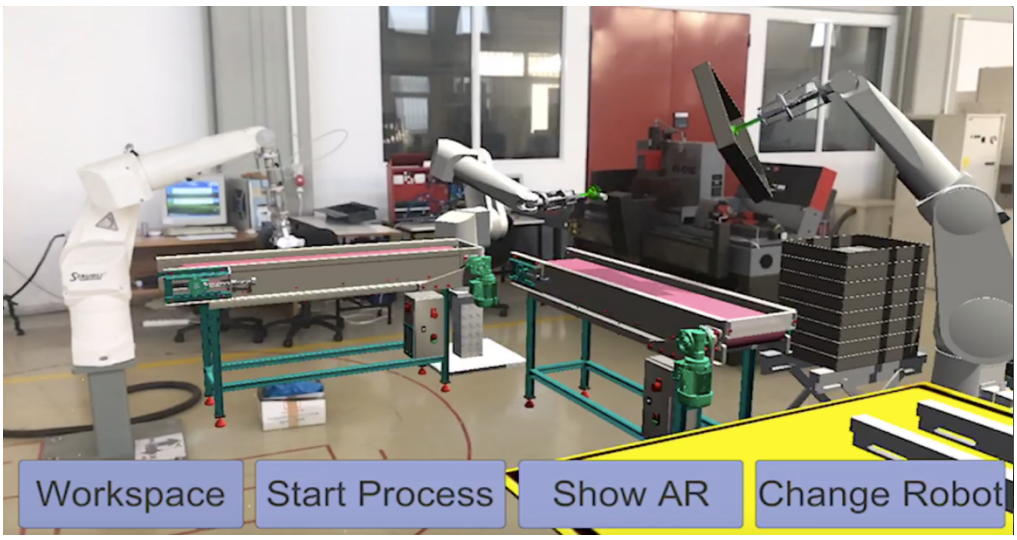
Εικόνα 50: Παραλαβή αξονοσυμμετρικού τεμαχίου



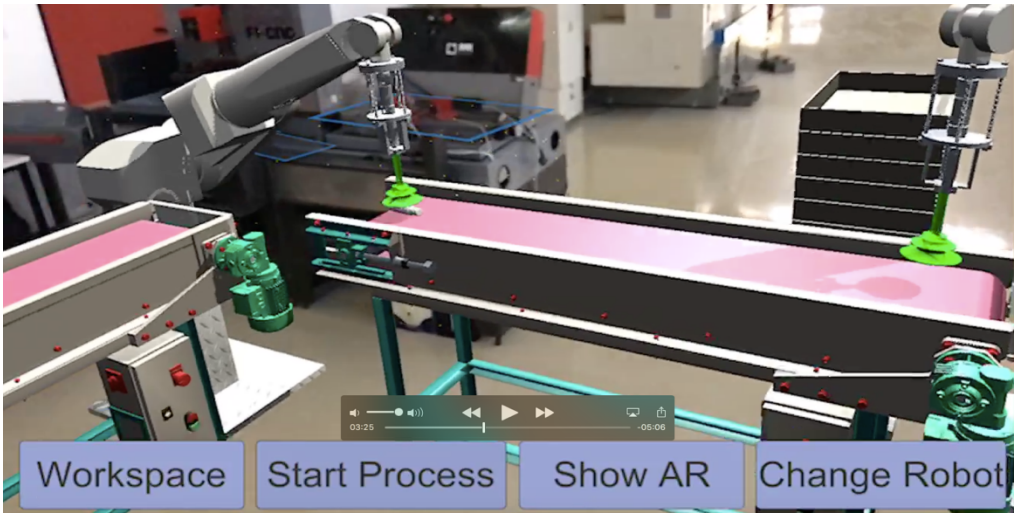
Εικόνα 51:Μεταφορά τεμαχίου και προετοιμασία του δεύτερου ρομπότ



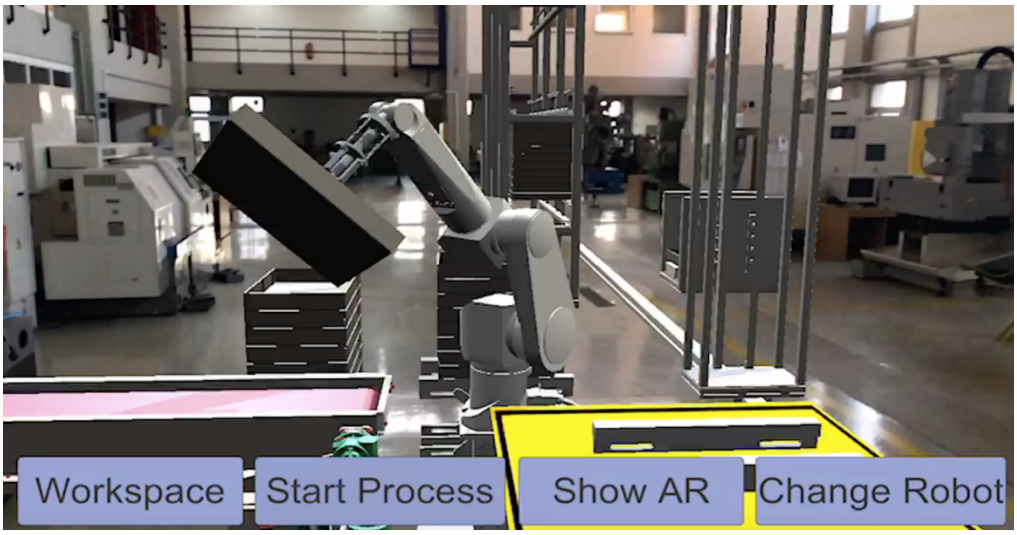
Εικόνα 52: Release του πραγματικού τεμαχίου από το ρομπότ Staubli RX90L του Εργαστηρίου Τεχνολογίας των Κατεργασιών πάνω στον εικονικό ταινιόδρομο του FMS



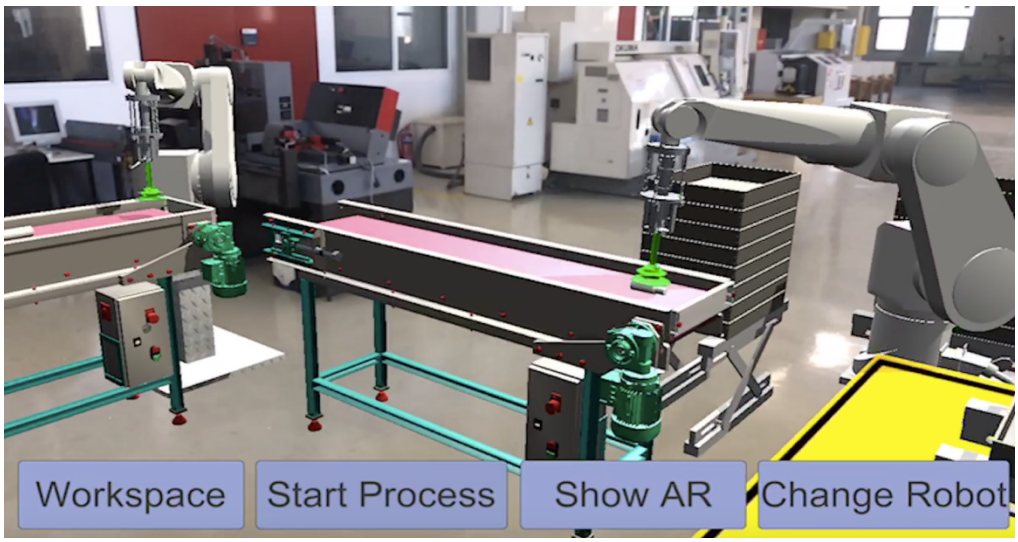
Εικόνα 53: 1^{ο} ρομπότ-Απόθεση τεμαχίου στον ταινιόδρομο, 2^{ο} ρομπότ-μεταφορά έτοιμου τεμαχίου από την εργαλειομηχανή προς τον δεύτερο ταινιόδρομο, 3^{ο} ρομπότ- φόρτωση της θέσης παραλαβής με νέο δίσκο μεταφοράς



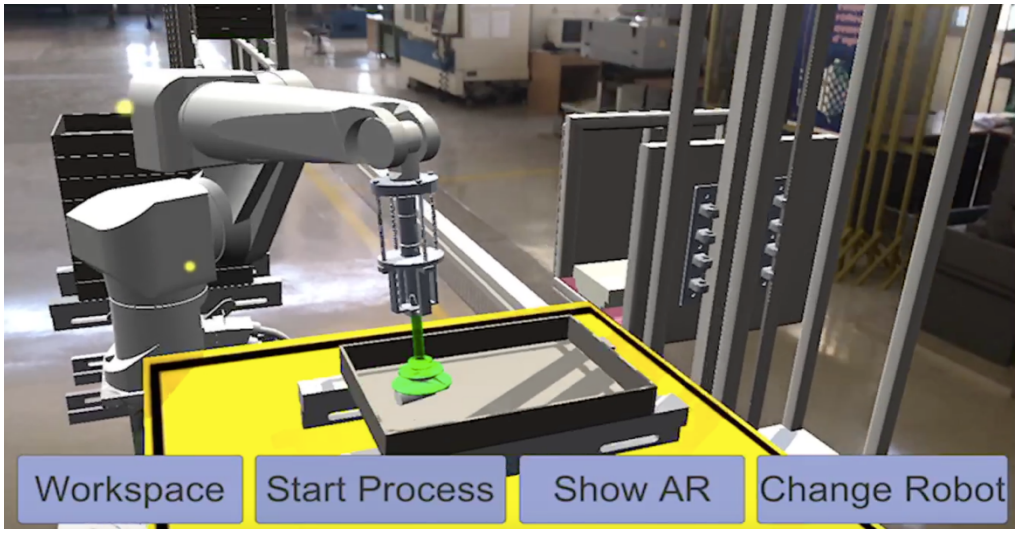
Εικόνα 54: Απόθεση και μεταφορά κατεργασμένου τεμαχίου



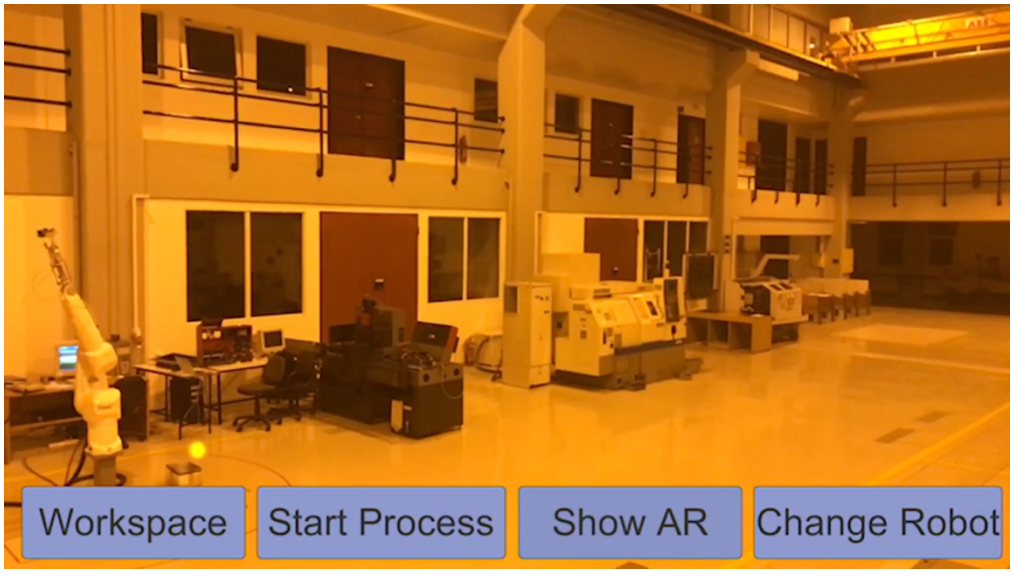
Εικόνα 55: Φόρτωση θέσης παραλαβής με νέο δίσκο



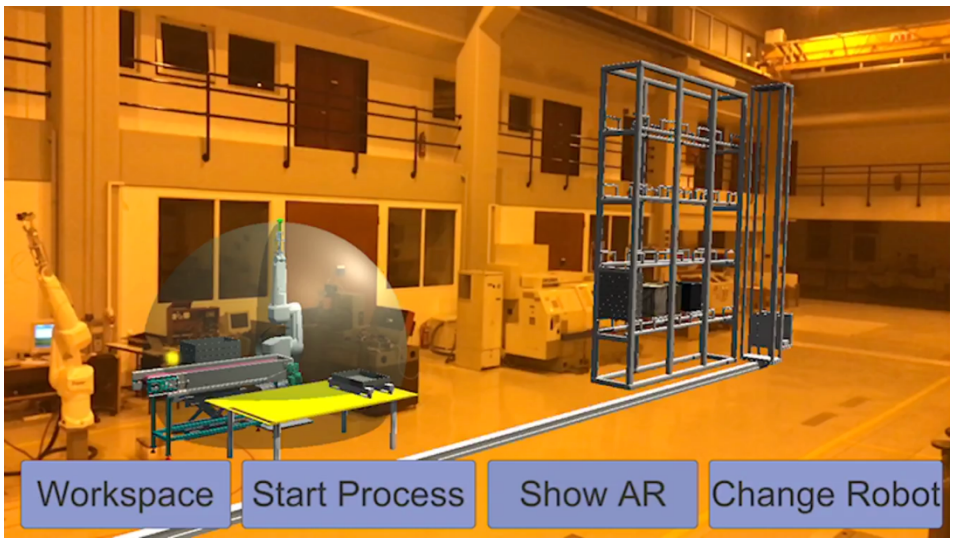
Εικόνα 56: Μεταφορά κατεργασμένου τεμαχίου προς στον κενό δίσκο



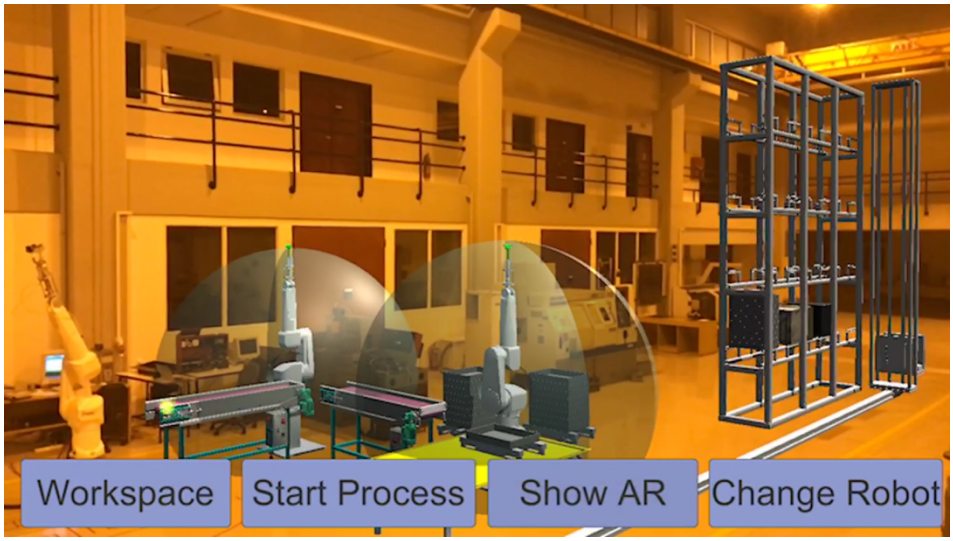
Εικόνα 57: Απόθεση κατεργασμένου τεμαχίου στον κενό δίσκο και προετοιμασία φόρτωσης από το σύστημα ASRS



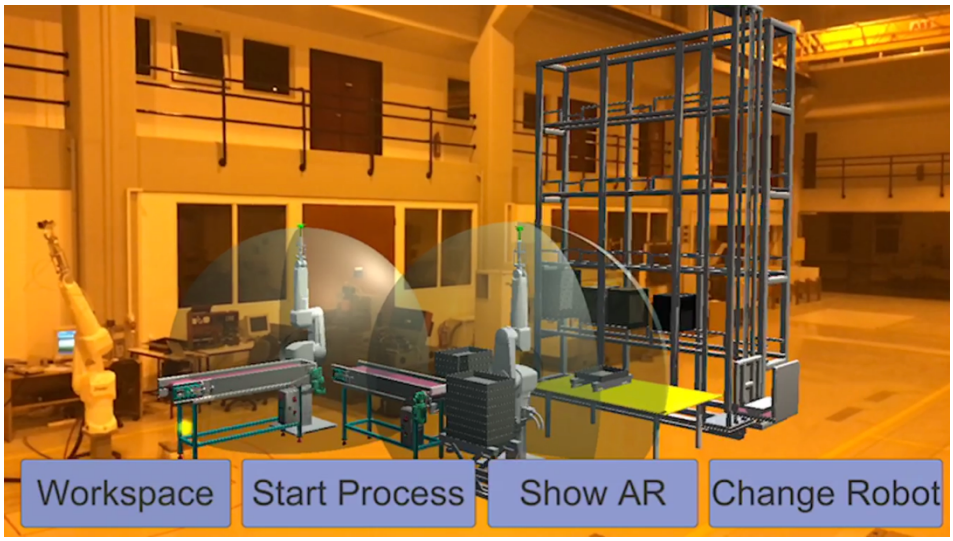
Εικόνα 58: Διαθέσιμος χώρος Εργαστηρίου Τεχνολογίας των Κατεργασιών



Εικόνα 59: Χωροταξική Διάταξη Νο1



Εικόνα 60: Χωροταξική Διάταξη Νο2



Εικόνα 61: Χωροταξική Διάταξη Νο3

9. Συμπεράσματα

9.1. Συνεισφορά

Η συνεισφορά της εφαρμογής ARobot_Layout είναι ουσιώδης στην κατεύθυνση της ανάπτυξης AR εργαλείων κατά τον σχεδιασμό μηχανουργικών διατάξεων. Τα βασικά στοιχεία στα οποία η εφαρμογή ARobot_Layout συνείσφερε στην υπάρχουσα κατάσταση είναι:

- Ότι ο χρήστης μπορεί να αξιολογεί διαισθητικά και εμπειρικά τις διαθέσιμες εναλλακτικές χωροταξικές διατάξεις του εξοπλισμού ενός εργοστασίου χρησιμοποιώντας εργαλεία επαυξημένης πραγματικότητας έχοντας ταυτόχρονα την δυνατότητα περιήγησης μεταξύ εικονικού και πραγματικού εξοπλισμού.

- Μπορεί να πετύχει προσομοίωση της παραγωγικής διαδικασίας χρησιμοποιώντας υφιστάμενο και πρός απόκτηση εξοπλισμό

- Μπορεί να εξασφαλίσει ακρίβεια στον σχεδιασμό καθώς του δίνεται η δυνατότητα εποπτείας της συνεργασίας εικονικού και πραγματικού εξοπλισμού, στα πλαίσια της προσομοίωσης, από την οποία μπορεί να εξάγει χρήσιμα συμπεράσματα.

9.2. Προτάσεις

- Η ένταξη στο Project περισσότερων ARσκηνών ώστε ο χρήστης να εξετάζει live όλες τις δυνατές χωροταξικές εναλλακτικές μέσω της εφαρμογής ARobot_Layout
- Ο τρόπος με τον οποίο οι ρομποτικοί βραχίονες κινούνται στην σκηνή μεταξύ δυο σημείων που επιθυμούμε να βρεθεί το Τελικό Σημείο Δράσης τους, έγινε με παρεμβολή μεταξύ αρχικής και τελικής θέσης της κάθε άρθρωσης ξεχωριστά. Αυτό, με δεδομένο ότι είχαμε καταλήξει εκ των προτέρων στην βέλτιστη χωροταξική διάταξη, δεν αποτέλεσε πρόβλημα για την ανάπτυξη της κινηματικής προσομοίωσης των εικονικών μοντέλων μας γιατί οι κινήσεις τους ήταν προκαθορισμένες και πολύ συγκεκριμένες. Κατά την εξέταση όμως μεγαλύτερου αριθμού εναλλακτικών χωροθετήσεων ή εργασιών, το πρόβλημα της κίνησης των ρομπότ γίνεται αρκετά μεγαλύτερο και χρονοβώρο με αυτόν το τρόπο. Έτσι προτείνεται ο έλεγχος της κίνησης των ρομπότ να γίνει με αντίστροφη κινηματική ανάλυση. Με αυτόν τον τρόπο θα εισάγεται στην εφαρμογή μόνο το σημείο στο οποίο επιθυμούμε να βρεθεί το τελικό σημείο δράσης και το αντίστοιχο script της αντίστροφης κινηματικής ανάλυσης θα υπολογίζει την βέλτιστη λύση περιστροφής των αρθρώσεων χωρίς να απαιτείται ο προγραμματισμός κάθε άρθρωσης ξεχωριστά. [46]
- Η επέκταση της εφαρμογής σε επίπεδο Μεικτής Πραγματικότητας (Mixed Reality-MR) η οποία θα λαμβάνει υπόψη την έμφραξη (Occlusion) των γεωμετριών του πραγματικού κόσμου και πολύ περισσότερο του υφιστάμενου εξοπλισμού. Στο προσεχές μέλλον θα γίνουν εμπορικά διαθέσιμες

συσκευές όπως το Magic Leap και το Hololens οι οποίες θα δίνουν αυτήν την δυνατότητα σκανάροντας αυτόματα το περιβάλλον στο οποίο βρίσκεται ο χρήστης (Mapping βασισμένο στην τεχνολογία SLAM). Παρόλα αυτά, αυτό είναι δυνατό να επιτευχθεί και μέσω του ARKit προγραμματιστικά, σκανάροντας ο χρήστης κάθε επιφάνεια (πολλά planes) προκειμένου να εισάγει την πραγματική γεωμετρία του χώρου στην εφαρμογή.

- Η χρήση αισθητήρων προκειμένου η συνεργασία εικονικών και πραγματικών μοντέλων να είναι ανεξάρτητη ενός χρονικού σεναρίου και πολύ πιο ευέλικτη και διαδραστική. Για παράδειγμα θα μπορούσε να διακόπτεται η λειτουργία πραγματικών και εικονικών μηχανημάτων όταν ένας άνθρωπος έμπαινε στον χώρο εργασίας ενός ρομπότ ή η εφαρμογή να ενημερώνεται πότε το πραγματικό ρομπότ είναι έτοιμο να “ξεφορτώσει“ το πραγματικό τεμάχιο προκειμένου να κάνει instantiate το εικονικό τεμάχιο.

Βιβλιογραφία

[1] G. Chryssolouris, D. Mavrikios, N. Papakostas, D. Mourtzis, G. Michalos, and K. Georgoulias, "Digital manufacturing: History, perspectives, and outlook" *Proc. Inst. Mech. Eng. Part B J. Eng. Manuf.*, vol. 223, no. 5, pp. 451–462, 2009.

[2] K. Khan, W. A., Raouf, A., & Cheng, *Virtual Manufacturing for Discrete Manufacturing Systems. In Virtual Manufacturing*. Springer, 2011.

[3] R. Schraft, J. Neugebauer, K. Grefen, "Factory and Logistics Planning with Virtual Reality" in *7th International Conference on Flexible Automation and Intelligent Manufacturing, Middlesbrough, UK, 25-27 June 1997, pp. 958-968*.

[4] N. Shariatzadeh, G. Sivard, D. Chen, "Software Evaluation Criteria for Rapid Factory Layout Planning" in *Procedia CIRP 3 (2012) 299–304*.

[5] B. Korves, M. Loftus, "Designing an immersive virtual reality interface for layout planning" *Mater. Process. Technol.*, p. 107, 2000.

[6] K. Pentenrieder, C. Bade, F. Doil, P. Meier, "Augmented Reality-based factory planning - an application tailored to industrial needs" in *6th IEEE and ACM International Symposium on Mixed and Augmented Reality, Nara, Japan, 13 – 16 November 2007, pp. 31-42.*

[7] J. Lee, S. Han, J. Yang, "Construction of a computer-simulated mixed reality environment for virtual factory layout planning" in *Computers in Industry*, 2011, p. 62.

[8] W. Zhai, X., Fan, J. Yan, P. Zhu, "An Integrated Simulation Method to Support Virtual Factory Engineering" *Int. J. CAD/CAM*, vol. 2/1, pp. 39–44, 2002.

[9] M. Weyrich, P. Drews, "An interactive environment for virtual manufacturing: the virtual workbench," in *Computers in Industry*, 1999, p. 38.

[10] P.R. Moore, J. Pu, H.C. Ng, C.B. Wong, S.K. Chong, X. Chen, J. Adolfsson, P. Olofsgård, J.O. Lundgren, "Virtual engineering: an integrated approach to agile manufacturing machinery design and controlNo Title" in *Mechatronics 13*, 2003, pp. 1105–1121.

[11] H. Hibino, T. Inukai, Y. Fukuda"Efficient manufacturing system implementation based on combination between real and virtual factory" *Int. J. Prod. Res.*, vol. 44/18-19, pp. 3897–3915, 2006.

[12] B.-K. Min, Z. Huang, Z.J. Pasek, D. Yip-Hoi, F. Husted, S. Marker, "Integration of Real-Time Control Simulation to a Virtual Manufacturing Environment" *Adv. Manuf. Syst.*, vol. 1/1, pp. 67–87, 2002.

[13] M. H. M. Bal, "Virtual factory approach for implementation of holonic control in industrial applications: A case study in die-casting industry" *Robot. Comput. Integr. Manuf.*, vol. 25, pp. 570–581, 2009.

[14] A. Abdul Kadir, X. Xu, E. Haemmerle, "Virtual machine tools and virtual machining—A technological review," *Robot. Comput. Integr. Manuf.*, vol. 27, pp. 494–508, 2011.

[15] P. Pedrazzoli, M. Sacco, A. Jonsson, C.R. Boer, "Virtual Factory Framework: Key Enabler For Future Manufacturing," in *P.G. Maropoulos (eds.) Digital Enterprise Technology: perspectives and future challenges*, Springer, 2007, pp. 83–90.

[16] A. K. J. Cecil, "Virtual engineering approaches in product and process design," *Int. J. Adv. Manuf. Technol.*, vol. 31/9-10, pp. 846–856, 2007.

[17] D. Ostermayer, J. C. Aurich, C. Wagenknecht, “Improvement of Manufacturing Processes with Virtual Reality based CIP-Workshops,” *Int. J. Prod. Res.*, vol. 47/19, pp. 5297–5309, 2009.

[18] Gupta, O. K., & Jarvis, R. A. ,“Using a Virtual World to Design a Simulation Platform for Vision and Robotic Systems,” in *Advances in Visual Computing*, Springer, 2009, pp. 233–242.

[19] W. Reinhart, G., Munzert, U., & Vogl, “A programming system for robot-based remote-laser-welding with conventional optics,” *CIRP Ann. Technol.*, vol. 57(1), pp. 37–40, 2008.

[20] L. Akan, B., Ameri, A., Curuklu, B., & Asplund, “Intuitive industrial robot programming through incremental multimodal language and augmented reality,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 3934–3939.

[21] K. Chong, J., Ong, S., Nee, A., & Youcef-Youmi, “Robot programming using augmented reality: An interactive method for planning collision-free paths,” *Robot. Comput. Integr. Manuf.*, vol. 25(3), pp. 689–701, 2009.

[22] A. Fang, H., Ong, S., & Nee, “Interactive robot trajectory planning and simulation using Augmented Reality,” *Robot. Comput. Integr. Manuf.*, vol. 28(2), pp. 227–237, 2012.

[23] “Learning ARKit and Unity3d.” [Online]. Available: http://talesfromtherift.com/learning-arkit-and-unity3d/.

[24] V. Paelke, “Augmented reality in the smart factory: Supporting workers in an industry 4.0. environment,” *19th IEEE Int. Conf. Emerg. Technol. Fact. Autom. ETFA 2014*, 2014.

[25] F. Almada-Lobo, “The Industry 4.0 revolution and the future of Manufacturing Execution Systems (MES),” *J. Innov. Manag.*, vol. 3, no. 4, p. 17, 2016.

[26] N. Menck, Yang X., Weidig C., Winkes P., Lauer C., Hagen H., Hamann B. and Aurich J.C, “Collaborative Factory Planning in Virtual Reality,” *Procedia CIRP*, vol. 3, pp. 317–322, 2012.

[27] Y.-B. Luo, “A virtual layout system integrated with polar coordinates-based genetic algorithm,” *Int. J. Comput. Appl. Technol.*, vol. 35, pp. 122 – 127, 2009.

[28] A. Y. C. Nee, S. K. Ong, G. Chryssolouris, and D. Mourtzis, “Augmented reality applications in design and manufacturing,” *CIRP Ann. - Manuf. Technol.*, vol. 61, no. 2, pp. 657–679, 2012.

[29] X. V. Gogouvitis and G. C. Vosniakos, “Construction of a virtual reality environment for robotic manufacturing cells,” *Int. J. Comput. Appl. Technol.*, vol. 51, no. 3, p. 173, 2015.

[30] G. C. Vosniakos, E. Levedianos, and X. V. Gogouvitis, “Streamlining virtual manufacturing cell modelling by behaviour modules,” *Int. J. Manuf. Res.*, vol. 10, no. 1, p. 17, 2015.

[31] J. Lee, S. Han, and J. Yang, “Construction of a computer-simulated mixed reality environment for virtual factory layout planning,” *Comput. Ind.*, vol. 62, no. 1, pp. 86–98, 2011.

[32] S. Jiang and A. Y. C. Nee, “A novel facility layout planning and optimization methodology,” *CIRP Ann. - Manuf. Technol.*, vol. 62, no. 1, pp. 483–486, 2013.

[33] “Χωροταξικός Σχεδιασμός.” [Online]. Available: http://academics.epu.ntua.gr/LinkClick.aspx?fileticket=oZHasbEbopQ%3D&tabid=380&mid=838.

[34] N. Xie, W, Sahinidis, “A Branch-and-bound Algorithm for the Continuous Facility Layout Problem,” *Comput. Chem. Eng.*, vol. 32(4), pp. 1016–1028, 2008.

[35] R. A. Inman, “Ευέλικτα Συστήματα Παραγωγής” [Online].Available: https://eclass.teiwm.gr/modules/document/file.php/TUCMECH116/2-FMS.pdf.

[36] “ΕΥΕΛΙΚΤΑ ΣΥΣΤΗΜΑΤΑ ΠΑΡΑΓΩΓΗΣ & ΤΕΧΝΟΛΟΓΙΑ ΟΜΑΔΩΝ” [Online]. Available:https://repository.kallipos.gr/bitstream/11419/4525/1/02_chapter_4.pdf.

[37] Α. Αποστολίδου, “Εξέλιξη των Ειδικών Εργαλειομηχανών και Χρήση τους στη Βιομηχανία” 2006.

[38] “Stäubli Robotics: Painting Brochure.” [Online]. Available: https://www.staubli.com/en/robotics/robot-solution-application/painting-robot/.

[39] Ν. Κοντογιάννης, “ΕΛΕΓΧΟΣ ΚΕΝΤΡΟΥ ΤΟΡΝΕΥΣΗΣ ΜΕ ΒΑΣΗ ΤΟ LINUXCNC,” 2015.

[40] “Unity (game engine).” [Online]. Available: https://en.wikipedia.org/wiki/Unity_(game_engine) .

[41] “iOS 11.” [Online]. Available: https://el.wikipedia.org/wiki/IOS_11.

[42] D. E. Dilger, “Inside Apple’s ARKit and Visual Inertial Odometry, new in iOS 11.” [Online]. Available: http://appleinsider.com/articles/17/10/12/inside-apples-arkit-and-visual-inertial-odometry-new-in-ios-11.

[43] “Kinect.” [Online]. Available: https://en.wikipedia.org/wiki/Kinect.

[44] “ARKit - Mixed Reality - Real world geometry occlusion - Unity 3D.” [Online]. Available: https://www.youtube.com/watch?v=CQFkTvEcfpk.

[45] “Importing Assets.” [Online]. Available: https://docs.unity3d.com/Manual/ImportingAssets.html.

[46] Σ. ΜΙΧΑΣ, “Προγραμματισμός Τροχιάς και Τηλεπαρακολούθηση Λειτουργίας Ρομποτικού Βραχίονα σε Περιβάλλον Εικονικής Πραγματικότητας με χρήση Αισθητήρων Ηλεκτρονικών Συσκευών Ευρείας Κατανάλωσης.”

[47] Ε. Παπαδόπουλος, “Κινηματική,” in *Σημειώσεις Ρομποτικής*, .

[48] Ε. Λεβεδιανός, “Διερεύνηση Μεθοδολογίας και Εργαλείων Κατασκευής Μοντελων Εικονικής Πραγματικότητας για Ρομποτικά Κύτταρα Κατεργασιών,” 2013.

[49] “Material Handling Industry of America - Glossary,” in *Retrieved*.

[50] “Automated storage and retrieval system.” [Online]. Available: https://en.wikipedia.org/wiki/Automated_storage_and_retrieval_system.

[51] M. Hallberg, “Augmented Reality Tutorial: APPLE ARkit is AMAZING!!!,” 2017. [Online]. Available: https://www.youtube.com/watch?v=S7kKQZuOdlk.

[52] “Coroutines.” [Online]. Available: https://docs.unity3d.com/Manual/Coroutines.html.

ΠΑΡΑΡΤΗΜΑ Α

Coroutines [52]

Όταν καλούμε μια μέθοδο, αυτήν εκτελείται άμεσα πριν την επιστροφή. Αυτό σημαίνει ουσιαστικά ότι οποιαδήποτε ενέργεια που λαμβάνει χώρα σε μια λειτουργία πρέπει να συμβεί μέσα σε ένα frame.

Μια μέθοδος δεν μπορεί να χρησιμοποιηθεί για να περιέχει μια σταδιακή κίνηση ή μια ακολουθία συμβάντων κατά την πάροδο του χρόνου. Για παράδειγμα, αν επιθυμούμε την σταδιακή μείωση της τιμής alpha (αδιαφάνειας) ενός αντικειμένου μέχρι να γίνει εντελώς αόρατο έχουμε:

```
void Fade()
{
    for (float f = 1f; f >= 0; f -= 0.1f)
    {
        Color c = renderer.material.color;
        c.a = f;
        renderer.material.color = c;
    }
}
```

Ως έχει, η μέθοδος Fade δεν θα έχει το αναμενόμενο αποτέλεσμα. Προκειμένου να είναι ορατό το “ξεθώριασμα”, η τιμή alpha πρέπει να μειωθεί κατά την διάρκεια μιας ακολουθίας frame ώστε να εμφανιστούν όλες οι ενδιάμεσες τιμές πριν+ γίνει εντελώς αόρατο. Ωστόσο, η μέθοδος θα εκτελεστεί στο σύνολό της εντός ενός μόνο frame. Οι ενδιάμεσες τιμές δεν θα εμφανιστούν ποτέ και το αντικείμενο θα εξαφανιστεί αμέσως.

Είναι δυνατό να διαχειριστούμε καταστάσεις όπως αυτή, προσθέτοντας κώδικα στην μέθοδο Update προκειμένου να εκτελεσθεί η εξασθένιση της αδιαφάνειας καρέ-καρέ.

Ωστόσο, είναι συχνά πιο βολικό να χρησιμοποιείται μια Coroutine για αυτού του είδους τις λειτουργίες.

Μια Coroutine είναι σαν μια μέθοδος που έχει τη δυνατότητα να διακόπτει την εκτέλεση της και να επιστρέφει τον έλεγχο στην Unity, αλλά στη συνέχεια να συνεχίζει εκεί όπου σταμάτησε αλλά στο επόμενο frame. Στην C #, μια coroutine δηλώνεται ως εξής:

```
IEnumerator Fade()
 {
   for (float f = 1f; f >= 0; f -= 0.1f)
      {
          Color c = renderer.material.color;
          c.a = f;
          renderer.material.color = c;
          yield return null;
      }
}
```

Είναι βασικό μια Coroutine η οποία δηλώνεται με έναν τύπο επιστροφής IEnumerator να φέρει επίσης την

yield δήλωση επιστροφής. Η δήλωση επιστροφής yield είναι το σημείο στο οποίο η εκτέλεση θα διακοπεί η Coroutine και θα συνεχιστεί το επόμενο frame απο τον βασικό κώδικα. Προκειμένου να κληθεί μια Coroutine, πρέπει να χρησιμοποιηθεί η μέθοδος StartCoroutine ώς ακολούθως:

```
void Update()
{
   if (Input.GetKeyDown("f"))
       {
          StartCoroutine("Fade");
        }
}
```

Στην UnityScript, τα πράγματα είναι λίγο πιο απλά. Κάθε μέθοδος η οποία περιέχει την yield δήλωση επιστροφής θεωρείται ότι είναι μια Coroutine και ο τύπος επιστροφής IEnumerator δεν χρειάζεται απαραιτήτως να δηλωθεί:

```
function Fade()
{
   for (var f = 1.0; f >= 0; f -= 0.1)
      {
         var c = renderer.material.color;
         c.a = f;
         renderer.material.color = c;
         yield;
       }
}
```

Επιπλέον, μια Coroutine μπορεί να ξεκινήσει στην UnityScript απλά καλώντας την σαν να ήταν μια απλή μέθοδος:

```
function Update()
 {
   if (Input.GetKeyDown("f"))
       {
         Fade();
       }
 }
```

Από προεπιλογή, μια coroutine συνεχίζεται ανεξάρτητα από το σε ποιο frame βρίσκεται η εκτέλεση. Είναι επίσης όμως δυνατό να εισαχθεί μια χρονική καθυστέρηση χρησιμοποιώντας την μέθοδο WaitForSeconds.

Σε κάθε περίπτωση η χρήση μιας Coroutine είναι πολύ βολική όταν επιθυμούμε κάποιες διαδικασίες, όπως κινήσεις ενός αντικειμένου, να γίνονται στην σκηνή ανεξάρτητα από το frame στο οποίο βρισκόμαστε.

ΠΑΡΑΡΤΗΜΑ Β

-Script σε V+

```
.PROGRAM andreas.pg()
SET #grab= #PPOINT (90.82, -151.96, 48.82, 10.54, 4.5, -189.54)
SET #endiameso1= #PPOINT (91.64, -93.28, 71.87, 10.54, 4.5, -189.54)
SET #endiameso2= #PPOINT (91.64, -83.09, 158.51, 10.54, 4.5, -189.54)
SET #release= #PPOINT (92.02, -70.29, 143.34, 0, 106.95, -187.49)
SPEED 30 ALWAYS
FOR i= 0 TO 19
OPENI
MOVE #endiameso1
BREAK
MOVE #grab
BREAK
CLOSEI
MOVE #endiameso1
BREAK
MOVE #endiameso2
BREAK
MOVE #release
BREAK
OPENI
BREAK
MOVE # endiameso2
MOVE # endiameso1
BREAK
DELAY18
END
```

ΠΑΡΑΡΤΗΜΑ Γ

-Scripts

- **StateManager**

```
using UnityEngine;
using System.Collections;
using Assets.Code.States;
using Assets.Code.Interfaces;
public class StateManager : MonoBehaviour
{
  private IStateBase activeState;
  private static StateManager instanceRef;
  private GameObject activeRobot;
  public GameObject[] robotPrefabs;

  [HideInInspector]
  public GameData gameDataRef;

  void Awake()
  {
    if (instanceRef == null)
    {
      instanceRef = this;
      DontDestroyOnLoad (gameObject);
    }
    else
    {
      DestroyImmediate (gameObject);
    }
  }


  void Start ()
  {
     // in order to take an instance of BeginState
        activeState =new BeginState(this);
        gameDataRef=GetCompo
  nent<GameData>();
    }

  // It calls the StateUpdate method of active
  state
    void Update ()
    {
      if (activeState != null)
        activeState.StateUpdate ();
    }

  // It calls the ShowIt method of active state
    void OnGUI()
    {
      if (activeState!=null)
        activeState.ShowIt();
    }
      pub-
  lic void SwitchState(IStateBase newState)
    {
      activeState = newState;
    }

  }
```

- **GameData**

```
using UnityEngine;
using System.Collections.Generic;

public class GameData : MonoBehaviour
{
  public Texture2D beginStateSplash;
  public Texture2D AbbButton;
  public Texture2D StaubliButton;
  public Texture2D playButton;
  public Texture2D poiSplash;
  public Texture2D MasysLogo;
  public Texture2D NTUALogo;
  public Texture2D ARKitLogo;
  public Texture2D UnityLogo;
  public List <GameObject> cameras;
  public int selectedRobotId=0;

  public void setRobotID(int rbt)
  {
  selectedRobotId = rbt; // store the robot ID
  }
}
```

-Interfaces

- **IStateBase**

```
namespace Assets.Code.Interfaces
{
  public interface IStateBase
  {
    void StateUpdate ();
    void ShowIt();
  }
}
```

-States

- **BeginState**

```
using UnityEngine;
using Assets.Code.Interfaces;
namespace Assets.Code.States
{
   public class BeginState:IStateBase
   {
      private StateManager manager;

      pub-
lic BeginState (StateManager managerRef)        //Con
structor
      {
         manager = managerRef;
         if (Application.loadedLevelName != "Scene0")
            Application.LoadLevel ("Scene0");
      }

      public void ShowIt()
      {

// BeginState Textures
         GUI.DrawTexture (new Rect (0, 0, Screen.width,
 Screen.height), manager.gameDataRef.beginStateSpla
sh, ScaleMode.StretchToFill);

 //Screen width and height:
float screenWidth= Screen.width;
float screenHeight= Screen.height;

//Half screen width and height:
float halfScreenWidth= Screen.width/2;
float halfScreenHeight= Screen.height/2;

         GUI.DrawTexture (new Rect (screenWidth -3*
screenWidth/4-
240, 230, 200, 200), manager.gameDataRef.MasysLog
o);
         GUI.DrawTexture (new Rect (screenWidth -2*
screenWidth/4-
240, 230, 200, 200), manager.gameDataRef.NTUALogo
);
         GUI.DrawTexture (new Rect (screenWidth -
screenWidth/4-
240, 230, 200, 200), manager.gameDataRef.ARKitLogo
);
         GUI.DrawTexture (new Rect (screenWidth-
240, 240, 200, 160), manager.gameDataRef.UnityLogo
);

// Create style for a button
GUIStyle myPressHereButtonStyle = new GUIStyle(GUI
.skin.button);
myPressHereButtonStyle.fontSize = 33;

// Create style for Labels
GUIStyle myLabelStyle = new GUIStyle(GUI.skin.label);
myLabelStyle.fontSize = 15;

// Create style for Title Labels
GUIStyle myTitleStyle = new GUIStyle(GUI.skin.label);
myTitleStyle.fontSize = 50;

// Load and set the Button Font from Resources file. ttf
/otf extension
Font myButtonFont = (Font)Resources.Load("sansatio
n/Sansation-Bold", typeof(Font));
myPressHereButtonStyle.font = myButtonFont;

// Load and set the Label Font from Resources file. ttf/
otf extension
Font myLabelFont = (Font)Resources.Load("sansation/
Sansation-Bold", typeof(Font));
myTitleStyle.font = myLabelFont;

// Load and set the Label Font from Resources file. ttf/
otf extension
Font myTitleFont = (Font)Resources.Load("kaushan-
script-Regular/KaushanScript-Regular", typeof(Font));
myLabelStyle.font = myTitleFont;

// Set color for selected and unselected buttons
myPressHereBut-
tonStyle.normal.textColor = Color.red;
myPressHereBut-
tonStyle.hover.textColor = Color.white;


//BeginState Label
GUI.Label (new Rect (halfScreenWidth-
200, 90, 400, 120), "  ARobot_Layout",myTitleStyle);


if (GUI.Button(new Rect (screenWidth-
390,screenHeight-
120,370,110), "Press HERE to Continue",myPressHere
ButtonStyle) || Input.anyKeyDown)
    {
   manager.SwitchState(new SetupState(manager));
    }
   }
  }
}
```

- **SetupState**

```
using UnityEngine;
using Assets.Code.Interfaces;
using System.Collections;
namespace Assets.Code.States
```

```
{
   public class SetupState:IStateBase
   {
      private StateManager manager;
      private GameObject robotAbb;
      private GameObject robotStaubli;
      private GameObject selectAbb;
      private GameObject selectStaubli;
      private string abbPlayer;
      private string staubliPlayer;
      public GameObject robot;
      private Transform  pivot ;
      private Transform  pivotAbb ;
      private Transform  pivotStaubli ;
      private GameObject abbSpec;
      private GameObject staubliSpec;
private GameObject gameManager;
private SelectARobot robotChoice;
private GameObject setupCamera;
private SelectARobot robotSelected;

public SetupState (StateManager manager
Ref)         // Constructor
      {
manager = managerRef; //StateManager instance
// This ensures that in any case  the SetupState will
load the Scene0
if (Application.loadedLevelName != "Scene0")
Application.LoadLevel ("Scene0");

// This saves the robots game objects into the follow-
ing variables
robotAbb = GameObject.Find ("ABB2600_End-
Effector_Setup");
robotStaubli = GameObject.Find ("Staubli_End-
Effector_Setup");

//This is an auxiliary but empty game object in order to
achieve the rotation of the robot around the desirable
axis
pivot-
Abb = robotAbb.transform.Find("PivotPointAbb");

pivot-
Staubli = robotStaubli.transform.Find("PivotPointStau
bli");

//Initial values
robot=robotStaubli;
pivot = pivotStaubli;
robotAbb.active = false;

// Game Objects for the text
abbSpec = GameObject.Find ("AbbSpecificationsGO");

staubliSpec = GameObject.Find ("StaubliSpecifications
GO");
abbSpec.active = false;

//In this way I can select from cameras list in
GameData script the corresponding camera for this
state
fo-
reach(var camera in manager.gameDataRef.cameras)

         {
            if(camera.name != "Setup Camera")
               camera.SetActive(false);
            else
               camera.SetActive(true);
         }
         setupCam-
era=GameObject.Find("Setup Camera");
      }

      public void StateUpdate ()
      {
//Stops the active robots rotation
if (!Input.GetButton ("Jump"))
ro-
bot.transform.RotateAround (pivot.position, Vector3.
up, 2);
      }

      public void ShowIt()
      {
//SetupState Label
GUI.Label (new Rect (90, 130, 150, 50), "Select Robot")
;

//ABB Button-We loaded the corresponding texture
from the Hierarchy panel to the AbbButton variable
if (GUI.Button (new Rect (40, 180, 180, 70), manager.g
ame-
Da-
taRef.AbbButton) || Input.GetKeyDown (KeyCode.A))
{
robotStaubli.active = false;
robotAbb.active = true;
robot = robotAbb;
pivot = pivotAbb;

abbSpec.active = true;
staubliSpec.active = false;
//This stores the the id of the active robot in a variable
in GameData script Staubli=0, Abb=1
manager.gameDataRef.setRobotID (1);
                              }

//Staubli Button
if (GUI.Button (new Rect (40, 260, 180, 70), manager.g
ame-
Da-
taRef.StaubliButton) || Input.GetKeyDown (KeyCode.S
)) {
robotStaubli.active = true;
robot = robotStaubli;
pivot = pivotStaubli;
```

```
robotAbb.active = false;
abbSpec.active = false;
staubliSpec.active = true;

manager.gameDataRef.setRobotID (0);
                                }

//Play Button
if (GUI.Button(new Rect (40,340,180,70), manager.ga
meDa-
taRef.playButton)|| Input.GetKeyDown(KeyCode.N))
        {
setupCamera.SetActive(false);
// In this way we invoke the next state
manag-
er.SwitchState(new PlayStateFirstPosition(manager));
        }
    }
  }
}
```

- **PlayStateFirstPosition**

```
using UnityEngine;
using Assets.Code.Interfaces;
namespace Assets.Code.States

{
  public class PlayStateFirstPosition:IStateBase
  {
    private StateManager manager;
    public GameObject robot;
    private GameObject gameManager;
    public GameObject robotAbb;
    public GameObject robotStaubli;
    public GameObject activeRobot;
    private GameObject parentOb;
    private GameObject childOb;
    private GameObject instance;
    private GameObject parent;
    public Rigidbody aRobot;
    private GameData gameDataRef;
    public int robotID;

    pub-
lic PlayStateFirstPosition (StateManager managerRef)
          //Constructor
    {
manager = managerRef;
// We load the UnityARKitScene
if (Application.loadedLevelName != "UnityARKitScene"
)
Application.LoadLevel ("UnityARKitScene");

 //Camera selection
fo-
reach (var camera in manager.gameDataRef.cameras)
    {
      if (camera.name != "CameraParent")
        camera.SetActive (false);
      else
        camera.SetActive (true);
    }

    if (manager.gameDataRef.selectedRobotId == 0)

    {
      robotID = 0;  //Staubli
    }
    else
    {
      robotID =1;  //Abb
    }
  }
 }
}
```

-Motion Script Components

- **AnimationController**

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class AnimationController : MonoBehaviour {

  public GameData gameDataref;

  public GameObject staubli2;
  public GameObject staubli3;
  public GameObject abb2;
  public GameObject abb3;

  public GameObject staubli22;
  public GameObject abb22;

  public GameObject staubli23;
  public GameObject staubli33;
  public GameObject abb23;
  public GameObject abb33;

  public GameObject arkitSceneGameObjects;
  public GameObject arkitSceneGameObjects2;
  public GameObject arkitSceneGameObjects3;
  public GameObject arkitSceneGameObjectsEmpty;

  public GameObject staubli2Workspace;
  public GameObject staubli3Workspace;
  public GameObject abb2Workspace;
  public GameObject abb3Workspace;
```

```
    public GameObject staubli22Workspace;
    public GameObject abb22Workspace;

    public GameObject staubli23Workspace;
    public GameObject staubli33Workspace;
    public GameObject abb23Workspace;
    public GameObject abb33Workspace;

    private int robotID;
    private int layoutButtonClicks=0;

    public bool startTheProcessFlag=false;
    public bool isPlaying=false;

    void Awake ()
    {
        gameDataref=GameObject.Find ("GameManager"
).GetComponent<GameData> ();

        //----1st Layout Robots----//
        staubli2=GameObject.Find ("N-
Staubli_Vacuum_Base2");
        staubli3=GameObject.Find ("N-
Staubli_Vacuum_Base3");
        abb2=GameObject.Find ("N-ABB IRB2600-
End_Effector2");
        abb3=GameObject.Find ("N-ABB IRB2600-
End_Effector3");

        //----2nd Layout Robots----//
        staubli22=GameObject.Find ("N-
Staubli_Vacuum_Base22");
        abb22=GameObject.Find ("N-ABB IRB2600-
End_Effector22");

        //----3rd Layout Robots----//
        staubli23=GameObject.Find ("N-
Staubli_Vacuum_Base23");
        staubli33=GameObject.Find ("N-
Staubli_Vacuum_Base33");
        abb23=GameObject.Find ("N-ABB IRB2600-
End_Effector23");
        abb33=GameObject.Find ("N-ABB IRB2600-
End_Effector33");

        //----Layouts----//
        arkitSceneGameOb-
jects = GameObject.Find ("GameObjects");
        arkitSceneGameOb-
jects2 = GameObject.Find ("GameObjects2");
        arkitSceneGameOb-
jects3 = GameObject.Find ("GameObjects3");
        arkitSceneGameObjectsEmp-
ty = GameObject.Find ("GameObjectsEmpty");
    }

    void Start ()
    {
        //----1st Layout Robot Workspaces----//
        staubli2Workspace=GameObject.Find ("Workspac
e_Staubli2");
        staubli3Workspace=GameObject.Find ("Workspac
e_Staubli3");
        abb2Workspace=GameObject.Find ("Workspace_
Abb2");
        abb3Workspace=GameObject.Find ("Workspace_
Abb3 ");

        //----2nd Layout Robot Workspaces----//
        staubli22Workspace=GameObject.Find ("Workspa
ce_Staubli22");
        abb22Workspace=GameObject.Find ("Workspace
_Abb22");

        //----3rd Layout Robot Workspaces----//
        staubli23Workspace=GameObject.Find ("Workspa
ce_Staubli23");
        staubli33Workspace=GameObject.Find ("Workspa
ce_Staubli33");
        abb23Workspace=GameObject.Find ("Workspace
_Abb23");
        abb33Workspace=GameObject.Find ("Workspace
_Abb33");

        //----
Initial Values of Robot Game Objects (Inactive)----//
        abb2.active = false;
        abb3.active = false;
        staubli2.active = false;
        staubli3.active = false;

        abb22.active = false;
        staubli22.active = false;

        abb23.active = false;
        abb33.active = false;
        staubli23.active = false;
        staubli33.active = false;

        InitialRobotChoice();

        //----Initial Values of Layouts (Inactive)----//
        arkitSceneGameObjects2.active = false;
        arkitSceneGameObjects3.active = false;
        arkitSceneGameObjectsEmpty.active = false;
    }

    void Update ()
    {
        robotID=gameDataref.selectedRobotId;
    }

    public void InitialRobotChoice()
    {
        if (gameDataref.selectedRobotId == 0)
        {
```

```
            staubli2.active = true;
            staubli3.active = true;

            staubli22.active = true;

            staubli23.active = true;
            staubli33.active = true;
        }
        else if (gameDataref.selectedRobotId == 1)
        {
            abb2.active = true;
            abb3.active = true;

            abb22.active = true;

            abb23.active = true;
            abb33.active = true;
        }
    }

    //Button "Start Process" on the ARKitScene
    public void startProcess()
    {
        if (robotID == 0 && arkitSceneGameObjects.active
InHierarchy==true)
        {
            startTheProcessFlag = true;
            isPlaying = true;
        }
    }

    //Button "Change Robot" on the ARKitScene
    public void SelectRobot()
    {
        if (isPlaying == false && robotID == 0)
        {
            //----1st Layout Robots Activation----//
            abb2.active = !abb2.active;
            abb3.active = !abb3.active;
            staubli2.active = !staubli2.active;
            staubli3.active = !staubli3.active;

            //----2nd Layout Robots Activation----//
            abb22.active = !abb22.active;
            staubli22.active = !staubli22.active;

            //----3rd Layout Robots Activation----//
            abb23.active = !abb23.active;
            abb33.active = !abb33.active;
            staubli23.active = !staubli23.active;
            staubli33.active = !staubli33.active;

            gameDataref.setRobotID (1);

        }
        else if (isPlaying == false && robotID == 1)
        {
            //----1st Layout Robots Activation----//
            abb2.active = !abb2.active;
            abb3.active = !abb3.active;
            staubli2.active = !staubli2.active;
            staubli3.active = !staubli3.active;

            //----2nd Layout Robots Activation----//
            abb22.active = !abb22.active;
            staubli22.active = !staubli22.active;

            //----3rd Layout Robots Activation----//
            abb23.active = !abb23.active;
            abb33.active = !abb33.active;
            staubli23.active = !staubli23.active;
            staubli33.active = !staubli33.active;

            gameDataref.setRobotID (0);
        }
    }


    //Button "Workspace" on the ARKitScene
    public void Workspace()
    {
        if (robotID == 0)
        {
            staubli2Workspace.active = !staubli2Workspace
.active;
            staubli3Workspace.active = !staubli3Workspace
.active;

            staubli22Workspace.active = !staubli22Workspa
ce.active;

            staubli23Workspace.active = !staubli23Workspa
ce.active;
            staubli33Workspace.active = !staubli33Workspa
ce.active;
        }

        else if (robotID == 1)
        {
            abb2Workspace.active = !abb2Workspace.activ
e;
            abb3Workspace.active = !abb3Workspace.activ
e;

            abb22Workspace.active = !abb22Workspace.ac
tive;

            abb23Workspace.active = !abb23Workspace.ac
tive;
            abb33Workspace.active = !abb33Workspace.ac
tive;
        }
    }

    //Button "Show AR" on the ARKitScene
    public void ShowAR()
    {
        if (isPlaying == false && arkitSceneGameObjects.a
```

```
ctiveInHierarchy==true)
        {
            arkitSceneGameOb-
jects.active = !arkitSceneGameObjects.active;
            arkitSceneGameOb-
jects2.active = !arkitSceneGameObjects2.active;
        }
    else if (isPlaying == false && arkitSceneGameObje
cts2.activeInHierarchy==true)
        {
            arkitSceneGameOb-
jects2.active = !arkitSceneGameObjects2.active;
            arkitSceneGameOb-
jects3.active= !arkitSceneGameObjects3.active;
        }

    else if (isPlaying == false && arkitSceneGameObje
cts3.activeInHierarchy==true)
        {
            arkitSceneGameOb-
jects3.active = !arkitSceneGameObjects3.active;
            arkitSceneGameObjectsEmp-
ty.active  = !arkitSceneGameObjectsEmpty.active;
        }
    else if (isPlaying == false && arkitSceneGameObje
ctsEmpty.activeInHierarchy==true)
        {
            arkitSceneGameOb-
jects.active = !arkitSceneGameObjects.active;
            arkitSceneGameObjectsEmp-
ty.active = !arkitSceneGameObjectsEmpty.active;
        }
   }

}
```

- **RawItemMotionManager**

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
pub-
lic class RawItemMotionManager : MonoBehaviour {

private FKMover2 robot2;
private RawItemMotion rawItem;
private ReadyItemMotion readyItem;
public AnimationController animationController;
GameObject itemClone;
private GameObject readyItemClone;
public GameObject desiredParentGameObject;
public GameObject targetGameObject;
Vector3 itemPositionBeforeDestroy;
Vector3 itemRotationBeforeDestroy;
public bool firstItemIsReadyAndPositioned = false;
public bool startTheProcessFlag;
public AudioSource latheSound;

void Awake()
{
//Instances of Item and Robot script components
ro-
bot2 = GameObject.Find ("Staubli RX90L2").GetCompo
nent<FKMover2> ();
ra-
wItem=GameObject.Find ("Decimated Game Objects "
).GetComponent<RawItemMotion> ();
ready-
Item=GameObject.Find ("Decimated Game Objects ").
GetComponent<ReadyItemMotion> ();
lathe-
Sound=GameObject.Find ("Staubli RX90L2").GetComp
onent<AudioSource> ();
animationControl-
ler=GameObject.Find ("HitCubeParent").GetCompone
nt<AnimationController> ();
  }

  void Update()
  {
startTheProcess-
Flag = animationController.startTheProcessFlag;
  }

IEnumerator Start()
  {

  //---------RawItemMotion & Robot2Motion----------

  //The process starts when the corresponding button
is pressed
yield return new WaitUntil(() => startTheProcessFlag =
= true);

// Move the robot from DOREADY position to Rest posi
tion in order to wait for the new item to arrive
yield return StartCoroutine (robot2.MoveFromInitialTo
ConveyorRestPosition (5.0f));

// 8sec delay in order the first robot to bring on the co
nveyor the first raw item the same moment with the
rawitem instantiate
  yield return new WaitForSecondsRealtime(8);

//This instantiates the new item
itemClone = rawItem.CreateItemPrefab ();

//Linear interpolation of the new item between lerpSta
rtingPosition and lerpEndingPosition positions
yield return StartCoroutine(rawItem.MoveItem (itemC
lone.transform, 10.0f));

// The total production cycle consists of the production
 of 20 items (20 items in order the storage to fill up).
for (int i = 1; i < 20; i++)
 {
yield return StartCoroutine (robot2.ConveyorRestToLo
```

```
adPositionRotation (1.0f));
ItemLoading(itemClone);
yield return StartCoroutine (robot2.LoadToIntermedia
tePositionRotation (2.0f));
yield return StartCoroutine (robot2.IntermediateToFirs
tPositionRotation (3.0f));
ItemUnLoading ();

//Lathe sound effect
latheSound.Play ();
//In these variables we store the position and rotation
of the rawitem in order to instantiatein the same positi
on the readyitem
itemPositionBefore-
Destroy= itemClone.transform.position;
itemRotationBefore-
Destroy= itemClone.transform.eulerAngles;

yield return StartCoroutine (robot2.FirstToLatheRestP
ositionRotation (2.0f));
De-
stroy(itemClone, 20.0f); //20.0f is the amount of time t
o wait before destroy the game object itemClone (raw
item)

//Instantiate new item. This will return us the ReadyIte
m in the same position with the "destroyed" raw item
readyItem-
Clone = readyItem.CreateItemPrefab (itemPositionBef
oreDestroy,itemRotationBeforeDestroy);

// During the following 29 seconds the item is being pr
ocessed
yield return new WaitForSecondsRealtime(29);
latheSound.Pause ();
yield return StartCoroutine (robot2.LatheRestToFirstP
ositionRotation (2.0f));
yield return new WaitForSecondsRealtime(1);
ItemLoading(readyItemClone);

yield return StartCoroutine (robot2.FirstToIntermediat
ePositionRotation (2.0f));
yield return StartCoroutine (robot2.IntermediateToSec
ondPositionRotation (3.0f));
ItemUnLoading ();
Destroy(readyItemClone, 0.01f);

//This flag informs the second conveyor and the 3rd ro
bot when ro start
firstItemIsReadyAndPositioned = true;

yield return StartCoroutine (robot2.SecondToConveyo
rRestPositionRot (5.0f));
firstItemIsReadyAndPosi-
tioned = false; // We turn that to false to be ready for t
he next iteration

// The first cycle began after the unloading of the first
real item.
itemClone = rawItem.CreateItemPrefab ();
yield return StartCoroutine( rawItem.MoveItem(itemC
lone.transform, 10.0f));
  }
 }

// This method is responsible for the ready item to bec
ome a child of the end effector of the robot.
void ItemLoading(GameObject itemClone)
  {
desiredParentGameOb-
ject = GameObject.Find("Robot2EndEffector");
targetGameObject = itemClone;
targetGameOb-
ject.transform.parent = desiredParentGameObject.tra
nsform;
  }

//This method assigns the raw item on the Lathe in ord
er to be processed.
void ItemUnLoading()
  {
desiredParentGameOb-
ject = GameObject.Find("Storage");
//Actually we dont care about the parent object at this
 moment
targetGameOb-
ject.transform.parent = desiredParentGameObject.tra
nsform;
  }
}
```

- **ReadyItemMotionManager**

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
pub-
lic class ReadyItemMotionManager : MonoBehaviour {

private FKMover3 robot3;
private RawItemMotion rawItem;
private ReadyItemMotion readyItem;
private RawItemMotionManager rawItemManager;
public AnimationController animationController;
GameObject itemClone;
GameObject newTray;
public GameObject desiredParentGameObject;
public GameObject targetGameObject;
public GameObject Tray;
public bool firstItemIsReadyAndPositioned;
public bool firstItemIsReadyForStorage= false;
public bool startTheProcessFlag;
private Transform leftTrayTransform;
private Transform rightTrayTransform;

void Awake()
```

```
    {
//Instances of Item and Robot script components
ro-
bot3 = GameObject.Find ("Staubli RX90L3").GetCompo
nent<FKMover3> ();
ready-
Item=GameObject.Find ("Decimated Game Objects ").
GetComponent<ReadyItemMotion> ();
ra-
wItem=GameObject.Find ("Decimated Game Objects "
).GetComponent<RawItemMotion> ();
rawItemManag-
er=GameObject.Find ("Decimated Game Objects ").Ge
tComponent<RawItemMotionManager> ();
leftTrayTrans-
form = GameObject.Find ("UpperTrayLeft").GetCompo
nent<Transform> ();
rightTrayTrans-
form = GameObject.Find ("UpperTrayRight").GetComp
onent<Transform> ();
animationControl-
ler=GameObject.Find ("HitCubeParent").GetCompone
nt<AnimationController> ();
    }

    //----------ReadyItemMotion & Robot3Motion---------

void Update()
    {
firstItemIsReadyAndPosi-
tioned = rawItemManager.firstItemIsReadyAndPositio
ned;
startTheProcess-
Flag = animationController.startTheProcessFlag;
    }
IEnumerator Start()
    {
//The process starts when the corresponding button is
pressed
yield return new WaitUntil(() => startTheProcessFlag =
= true);

//These out-of-the-
loop commands was done because the first tray is alre
ady in place
// The first cycle begins after the unloading of the first
real item.

//Move From Initial Position To the Rest Position Just A
bove the Conveyor
yield return StartCoroutine (robot3.MoveFromInitialTo
ConveyorRestPosition (5.0f));

//Unity will wait for the following condition to get a tr
ue value to continue execution.
yield return new WaitUntil(() => firstItemIsReadyAndP
ositioned == true);

//Instantiates and moves a new Item on the second co
nveyor
itemClone = readyItem.CreateItemPrefab ();
//Linear interpolation of the new item between lerpSta
rtingPosition and lerpEndingPosition positions
yield return StartCoroutine (readyItem.MoveItem (ite
mClone.transform, 10.0f));


//Load item
yield return StartCoroutine (robot3.RestToLoadPosRot
ation (1.0f));
ItemLoading(itemClone);
yield return StartCoroutine (robot3.LoadingIntermedia
tePosRotation (2.0f));
yield return StartCoroutine (robot3.IntermediateToUnl
oadPosRotation (3.0f));
ItemUnLoading ();

//Bring Tray2
yield return StartCoroutine (robot3.TrayLoadingInterm
ediatePosRotation (2.0f));
yield return StartCoroutine (robot3.IntermediateToGr
abTray2PosRotation (2.0f));
newTray = CreateNewTray (leftTrayTransform.position
,leftTrayTransform.eulerAngles);
BringTray (newTray);
yield return StartCoroutine (robot3.TrayLoadingInterm
ediatePosRotation (2.0f));
yield return StartCoroutine (robot3.IntermediateToRel
easeTrayPosRotation (3.0f));
ReleaseTray ();
yield return StartCoroutine (robot3.LoadingIntermedia
tePosRotation (2.0f));
yield return StartCoroutine (robot3.IntermediateToRes
tPosRotation (5.0f));

// From now on, each iteration for the ready item, will
take place over a certain period of time we adjust that
with coroutines

for (int I = 1; I < 20; I++) {

// Is odd, because something divided by two with a re
mainder of 1 is not even, i.e. 5/2 = 2, remainder 1
if (I % 2 == 1)
        {
yield return new WaitUntil(() => firstItemIsReadyAndP
ositioned == true);
//MoveItem
itemClone = readyItem.CreateItemPrefab ();
yield return StartCoroutine (readyItem.MoveItem (ite
mClone.transform, 10.0f));

//Load-Unload  item
yield return StartCoroutine (robot3.RestToLoadPosRot
ation (1.0f));
ItemLoading(itemClone);
yield return StartCoroutine (robot3.LoadingIntermedia
tePosRotation (2.0f));
```

```
yield return StartCoroutine (robot3.IntermediateToUnl
oadPosRotation (3.0f));
ItemUnLoading ();

//Bring Tray1
yield return StartCoroutine (robot3.TrayLoadingInterm
ediatePosRotation (2.0f));
yield return StartCoroutine (robot3.IntermediateToGr
abTray1PosRotation (2.0f));
newTray = CreateNewTray (rightTrayTransform.positio
n,rightTrayTransform.eulerAngles);
BringTray (newTray);
yield return StartCoroutine (robot3.TrayLoadingInterm
ediatePosRotation (2.0f));
yield return StartCoroutine (robot3.IntermediateToRel
easeTrayPosRotation (3.0f));
ReleaseTray ();
yield return StartCoroutine (robot3.LoadingIntermedia
tePosRotation (2.0f));
yield return StartCoroutine (robot3.IntermediateToRes
tPosRotation (5.0f));

        }

// Is even, because something divided by two without r
emainder is even, i.e 4/2 = 2, remainder 0
if (I % 2 == 0)
        {
yield return new WaitUntil(() => firstItemIsReadyAndP
ositioned == true);
//MoveItem
itemClone = readyItem.CreateItemPrefab ();
yield return StartCoroutine (readyItem.MoveItem (ite
mClone.transform, 10.0f));

//Load-Unload Item
yield return StartCoroutine (robot3.RestToLoadPosRot
ation (1.0f));
ItemLoading(itemClone);
yield return StartCoroutine (robot3.LoadingIntermedia
tePosRotation (2.0f));
yield return StartCoroutine (robot3.IntermediateToUnl
oadPosRotation (3.0f));
ItemUnLoading ();

//Bring Tray2
yield return StartCoroutine (robot3.TrayLoadingInterm
ediatePosRotation (2.0f));
yield return StartCoroutine (robot3.IntermediateToGr
abTray2PosRotation (2.0f));
newTray = CreateNewTray (leftTrayTransform.position
,leftTrayTransform.eulerAngles);
BringTray (newTray);
yield return StartCoroutine (robot3.TrayLoadingInterm
ediatePosRotation (2.0f));
yield return StartCoroutine (robot3.IntermediateToRel
easeTrayPosRotation (3.0f));
ReleaseTray ();
yield return StartCoroutine (robot3.LoadingIntermedia
tePosRotation (2.0f));
yield return StartCoroutine (robot3.IntermediateToRes
tPosRotation (5.0f));
        }
      }
  }

// This method is responsible for the ready item to bec
ome a child of the end effector of the robot.
void ItemLoading(GameObject itemClone)
  {
desiredParentGameOb-
ject = GameObject.Find("Robot3EndEffector");
targetGameObject = itemClone;
targetGameOb-
ject.transform.parent = desiredParentGameObject.tra
nsform;
  }

//This method assigns the ready item on the active tra
y in order to be ready for storage.
void ItemUnLoading()
  {
desiredParentGameOb-
ject = GameObject.FindWithTag ("ActiveTray");
targetGameOb-
ject.transform.parent = desiredParentGameObject.tra
nsform;
  }

// This method is responsible for the active tray, to bec
ome a child of the end effector of the robot.
void BringTray(GameObject trayClone)
  {
desiredParentGameOb-
ject = GameObject.Find("Robot3EndEffector");
targetGameObject = trayClone;
targetGameOb-
ject.transform.parent = desiredParentGameObject.tra
nsform;
  }

//This method releases the active tray on the waiting p
osition.
void ReleaseTray()
  {
desiredParentGameOb-
ject = GameObject.Find ("Table");
targetGameOb-
ject.transform.parent = desiredParentGameObject.tra
nsform;
  }

//This method instantiates a new tray and gives it an A
ctiveTray tag.
pub-
lic GameObject CreateNewTray (Vector3 position, Vect
or3 rotation)
  {
```

```
GameObject trayClone;
tray-
Clone = Instantiate (Tray, position, Quaternion.identity
 ) as GameObject;
trayClone.transform.localEulerAngles = rotation;
tray-
Clone.transform.localScale += new Vector3(0.23F, 0.23
F, 0.23F); // Scale the Tray at the desired dimension
trayClone.tag = "ActiveTray";
return trayClone;
   }
}
```

- **RawItemMotion**

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class RawItemMotion : MonoBehaviour {


public GameObject RawItemPrefab;
private Transform lerpStartingPosition;
private Transform lerpEndingPosition;
GameObject itemClone;
private Vector3 initialPosition;
private Vector3 initialRotation;

void Awake()
   {
//The following two variables give us the starting and
end-
ing positions, for the linear interpolation of the item on
 the conveyor
lerpStartingPosi-
tion =GameObject.FindWithTag ("Conveyor1Position1
").GetComponent<Transform>();
lerpEndingPosi-
tion =GameObject.FindWithTag ("Conveyor1Position2
").GetComponent<Transform>();

//These are specific position and rotation of the item w
e want to instantiate
initialPosition = lerpStartingPosition.position;
initialRotation = new Vector3 (0.0f, 0.0f, 90.0f);
   }

pub-
lic IEnumerator MoveItem(Transform Transform, float
time)
   {
      var i= 0.0f;
      var rate= 1.0f/time;
      while (i < 1.0f)
   {
i += Time.deltaTime * rate;
Trans-
form.localPosition = Vector3.Lerp(lerpStartingPosition.
posi-
tion, lerpEndingPosition.position, i); //Vector 3 Variabl
es
yield return null;
      }
   }

public GameObject CreateItemPrefab ()
   {
GameObject itemClone;
item-
Clone = Instantiate (RawItemPrefab, initialPosition, Qu
aternion.identity ) as GameObject;
item-
Clone.transform.localEulerAngles = initialRotation;
return itemClone;
   }
}
```

- **ReadyItemMotion**

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class ReadyItemMotion : MonoBehaviour {

public GameObject ReadyItemPrefab;
public Transform lerpStartingPosition;
public Transform lerpEndingPosition;
GameObject itemClone;
public Vector3 initialPosition;
public Vector3 initialRotation;



void Awake()
   {
//The following two variables give us the starting and
end-
ing positions, for the linear interpolation of the item on
 the conveyor
lerpStartingPosi-
tion =GameObject.FindWithTag ("Conveyor2Position1
").GetComponent<Transform>();
lerpEndingPosi-
tion =GameObject.FindWithTag ("Conveyor2Position2
").GetComponent<Transform>();

//These are specific position and rotation of the item w
e want to instantiate
initialPosi-
tion = lerpStartingPosition.position; //new Vector3 (2.
1439f,0.789f,2.076f);
initialRotation = new Vector3 (0.0f, -90.0f, 90.0f);
   }

pub-
```

```
lic IEnumerator MoveItem(Transform Transform, float
time)
   {
      var i= 0.0f;
      var rate= 1.0f/time;
      while (i < 1.0f)
      {
i += Time.deltaTime * rate;
Trans-
form.localPosition = Vector3.Lerp(lerpStartingPosition.
position, lerpEndingPosition.position, i);
yield return null;
      }
   }

public GameObject CreateItemPrefab ()
   {
GameObject readyItem_Clone;
ready-
Item_Clone = Instantiate (ReadyItemPrefab, initialPosi
tion, Quaternion.identity ) as GameObject;
ready-
Item_Clone.transform.localEulerAngles = initialRotatio
n;
return readyItem_Clone;
   }


//Overload CreateItemPrefab method
pub-
lic GameObject CreateItemPrefab (Vector3 position, V
ector3 rotation)
   {
GameObject readyItemClone;
readyItem-
Clone = Instantiate (ReadyItemPrefab, position, Quate
rnion.identity ) as GameObject;
readyItemClone.transform.localEulerAngles = rotation;
return readyItemClone;
   }
}
```

- **FKMover2**

```
using UnityEngine;
using System.Collections;

public class FKMover2 : MonoBehaviour
{
   [Range (-160f,160f)] public float rotZ1 = 0;
   [Range (-227.5f,47.5f)] public float rotZ2 = -90;
   [Range (-52.5f,232.5f)] public float rotZ3 = 90;
   [Range (-270f,270f)] public float rotZ4 = 0;
   [Range (-105f,120f)] public float rotZ5 = 0;
   [Range (-270f,270f)] public float rotZ6 = 0;

   // DO READY angles. We ve changed only the Z axis
angles  The other angles are needed from the Quater-
nion of Unity.
   private float rotX1 = -90f;
   private float rotX2 = -90f;
   private float rotX3 = 0f;
   private float rotX4 = 90f;
   private float rotX5 = -90f;
   private float rotX6 = 90f;

   private float rotYAll = 0f;

   private float initRotZ1 = 0f;
   private float initRotZ2 = -90f;
   private float initRotZ3 = 90f;
   private float initRotZ4 = 0f;
   private float initRotZ5 = 0f;
   private float initRotZ6 = 0f;

   private Transform joint1;
   private Transform joint2;
   private Transform joint3;
   private Transform joint4;
   private Transform joint5;
   private Transform joint6;

   // Starting Rotations of Z axis of every joint
   private float rotationZ1From;
   private float rotationZ2From;
   private float rotationZ3From;
   private float rotationZ4From;
   private float rotationZ5From;
   private float rotationZ6From;

   // Target Rotations of Z axis of every joint
   private float rotationZ1To;
   private float rotationZ2To;
   private float rotationZ3To;
   private float rotationZ4To;
   private float rotationZ5To;
   private float rotationZ6To;

   private Quaternion j1RotFrom;
   private Quaternion j2RotFrom;
   private Quaternion j3RotFrom;
   private Quaternion j4RotFrom;
   private Quaternion j5RotFrom;
   private Quaternion j6RotFrom;

   private Quaternion j1RotTo;
   private Quaternion j2RotTo;
   private Quaternion j3RotTo;
   private Quaternion j4RotTo;
   private Quaternion j5RotTo;
   private Quaternion j6RotTo;
```

```
    private float Z1;
    private float Z2;
    private float Z3;
    private float Z4;
    private float Z5;
    private float Z6;

    private float Z1PrevLocalRot;
    private float Z2PrevLocalRot;
    private float Z3PrevLocalRot;
    private float Z4PrevLocalRot;
    private float Z5PrevLocalRot;
    private float Z6PrevLocalRot;


    void Awake ()
    {
        joint1 = GameObject.Find ("Joint 2.1").GetCompo
nent<Transform> ();
        joint2 = GameObject.Find ("Joint 2.2").GetCompo
nent<Transform> ();
        joint3 = GameObject.Find ("Joint 2.3").GetCompo
nent<Transform> ();
        joint4 = GameObject.Find ("Joint 2.4").GetCompo
nent<Transform> ();
        joint5 = GameObject.Find ("Joint 2.5").GetCompo
nent<Transform> ();
        joint6 = GameObject.Find ("Joint 2.6").GetCompo
nent<Transform> ();

    }

pub-
lic IEnumerator MoveFromInitialToConveyorRestPositi
on(float time)
    {
        rotationZ1From= 0f;
        rotationZ2From= -90f;
        rotationZ3From= 90f;
        rotationZ4From= 0f;
        rotationZ5From= 0f;
        rotationZ6From= 0f;

        Z1 = -38f;
        Z2=-115f;
        Z3=198f;
        Z4=0f;
        Z5=90f;
        Z6=0f;


        rotationZ1To=-rotationZ1From-
(Z1+rotationZ1From);   //Negative rotation
        rotationZ2To=-rotationZ2From-
(Z2+rotationZ2From);   //Negative rotation
        rotationZ3To= rotationZ3From-(Z3-
rotationZ3From);   //Positive rotation
        rotationZ4To=Z4;                        //=0
        rotationZ5To= rotationZ5From-(Z5-
rotationZ5From);   //Positive rotation
        rotationZ6To=Z6;                        //=0

        Z1PrevLocalRot = Z1;
        Z2PrevLocalRot= Z2;
        Z3PrevLocalRot = Z3;
        Z4PrevLocalRot = Z4;
        Z5PrevLocalRot = Z5;
        Z6PrevLocalRot = Z6;


        yield return StartCoroutine (GoToTarget(rotationZ
1From, rotationZ2From, rotationZ3From, rotationZ4Fr
om, rotationZ5From, rotationZ6From,
            time,
            rota-
tionZ1To, rotationZ2To, rotationZ3To, rotationZ4To, ro
tationZ5To, rotationZ6To));

    }

    pub-
lic IEnumerator ConveyorRestToLoadPositionRotation(
float time)
    {
        rotationZ1From=rotationZ1To;
        rotationZ2From=rotationZ2To;
        rotationZ3From=rotationZ3To;
        rotationZ4From=rotationZ4To;
        rotationZ5From=rotationZ5To;
        rotationZ6From=rotationZ6To;

        Z1=-38f;
        Z2=-115f;
        Z3=201f;
        Z4=0f;
        Z5=90f;
        Z6=0f;

        rotationZ1To = rotationZ1From-(Z1-
Z1PrevLocalRot);   //Positive rotation
        rotationZ2To =-rotationZ2From-
(Z2+Z2PrevLocalRot);   //Negative rotation
        rotationZ3To = rotationZ3From-(Z3-
Z3PrevLocalRot);   //Positive rotation
        rotationZ4To = rotationZ4From-(Z4-
Z4PrevLocalRot);   //Positive rotation
        rotationZ5To = rotationZ5From-(Z5-
Z5PrevLocalRot);   //Positive rotation
        rotationZ6To = rotationZ6From-(Z6-
Z6PrevLocalRot);   //Positive rotation

        Z1PrevLocalRot = Z1;
        Z2PrevLocalRot = Z2;
        Z3PrevLocalRot = Z3;
        Z4PrevLocalRot = Z4;
        Z5PrevLocalRot = Z5;
        Z6PrevLocalRot = Z6;
```

```
        yield return StartCoroutine (GoToTarget(rotationZ
1From, rotationZ2From, rotationZ3From, rotationZ4Fr
om, rotationZ5From, rotationZ6From,
            time,
            rota-
tionZ1To, rotationZ2To, rotationZ3To, rotationZ4To, ro
tationZ5To, rotationZ6To));
    }

    pub-
lic IEnumerator LoadToIntermediatePositionRotation(f
loat time)
    {

        //This will take the values from the previous move
ment so as the new one to begin exactly at the same p
oint.
        rotationZ1From = rotationZ1To;
        rotationZ2From=rotationZ2To;
        rotationZ3From=rotationZ3To;
        rotationZ4From=rotationZ4To;
        rotationZ5From=rotationZ5To;
        rotationZ6From=rotationZ6To;

        //Rotation values
        Z1=-38f;
        Z2=-115f;
        Z3=185f;
        Z4=0f;
        Z5=90f;
        Z6=0f;


        // Rotation target for each joint
        rotationZ1To = rotationZ1From-(Z1-
Z1PrevLocalRot);   //Positive rotation
        rotationZ2To =-rotationZ2From-
(Z2+Z2PrevLocalRot);   //Negative rotation
        rotationZ3To = rotationZ3From-(Z3-
Z3PrevLocalRot);   //Positive rotation
        rotationZ4To =-rotationZ4From-
(Z4+Z4PrevLocalRot);   //Positive rotation
        rotationZ5To = rotationZ5From-(Z5-
Z5PrevLocalRot);   //Positive rotation
        rotationZ6To = rotationZ6From-(Z6-
Z6PrevLocalRot);   //Positive rotation

        // Here we store the rotation values which will be
useful for the next movement
        Z1PrevLocalRot = Z1;
        Z2PrevLocalRot = Z2;
        Z3PrevLocalRot = Z3;
        Z4PrevLocalRot = Z4;
        Z5PrevLocalRot = Z5;
        Z6PrevLocalRot = Z6;


        yield return StartCoroutine (GoToTarget(rotationZ
1From, rotationZ2From, rotationZ3From, rotationZ4Fr
om, rotationZ5From, rotationZ6From,
            time,
            rota-
tionZ1To, rotationZ2To, rotationZ3To, rotationZ4To, ro
tationZ5To, rotationZ6To));
    }


    pub-
lic IEnumerator IntermediateToFirstPositionRotation(fl
oat time)
    {
        rotationZ1From=rotationZ1To;
        rotationZ2From=rotationZ2To;
        rotationZ3From=rotationZ3To;
        rotationZ4From=rotationZ4To;
        rotationZ5From=rotationZ5To;
        rotationZ6From=rotationZ6To;

        Z1=50f;
        Z2=-102f;
        Z3=203f;
        Z4=104f;
        Z5=53f;
        Z6=-68f;

        rotationZ1To = rotationZ1From-(Z1-
Z1PrevLocalRot);   //Positive rotation
        rotationZ2To =-rotationZ2From-
(Z2+Z2PrevLocalRot);   //Negative rotation
        rotationZ3To = rotationZ3From-(Z3-
Z3PrevLocalRot);   //Positive rotation
        rotationZ4To = -rotationZ4From-
(Z4+Z4PrevLocalRot);   //Positive rotation
        rotationZ5To =rotationZ5From-(Z5-
Z5PrevLocalRot);   //Positive rotation
        rotationZ6To = rotationZ6From-(Z6-
Z6PrevLocalRot);   //Positive rotation

        Z1PrevLocalRot = Z1;
        Z2PrevLocalRot = Z2;
        Z3PrevLocalRot = Z3;
        Z4PrevLocalRot = Z4;
        Z5PrevLocalRot = Z5;
        Z6PrevLocalRot = Z6;


        yield return StartCoroutine (GoToTarget(rotationZ
1From, rotationZ2From, rotationZ3From, rotationZ4Fr
om, rotationZ5From, rotationZ6From,
            time,
            rota-
tionZ1To, rotationZ2To, rotationZ3To, rotationZ4To, ro
tationZ5To, rotationZ6To));
    }


    pub-
```

```
lic IEnumerator FirstToLatheRestPositionRotation(float
 time)
   {
      rotationZ1From=rotationZ1To;
      rotationZ2From=rotationZ2To;
      rotationZ3From=rotationZ3To;
      rotationZ4From=rotationZ4To;
      rotationZ5From=rotationZ5To;
      rotationZ6From=rotationZ6To;

      Z1=25f;
      Z2=-102f;
      Z3=213f;
      Z4=104f;
      Z5=73f;
      Z6=10f;

      rotationZ1To = rotationZ1From-(Z1-
Z1PrevLocalRot);   //Positive rotation
      rotationZ2To =-rotationZ2From-
(Z2+Z2PrevLocalRot);   //Negative rotation
      rotationZ3To = rotationZ3From-(Z3-
Z3PrevLocalRot);   //Positive rotation
      rotationZ4To = rotationZ4From-(Z4-
Z4PrevLocalRot);   //Positive rotation
      rotationZ5To = rotationZ5From-(Z5-
Z5PrevLocalRot);   //Positive rotation
      rotationZ6To = rotationZ6From-(Z6-
Z6PrevLocalRot);   //Positive rotation

      Z1PrevLocalRot = Z1;
      Z2PrevLocalRot = Z2;
      Z3PrevLocalRot = Z3;
      Z4PrevLocalRot = Z4;
      Z5PrevLocalRot = Z5;
      Z6PrevLocalRot = Z6;

      yield return StartCoroutine (GoToTarget(rotationZ
1From, rotationZ2From, rotationZ3From, rotationZ4Fr
om, rotationZ5From, rotationZ6From,
         time,
         rota-
tionZ1To, rotationZ2To, rotationZ3To, rotationZ4To, ro
tationZ5To, rotationZ6To));
   }

   pub-
lic IEnumerator LatheRestToFirstPositionRotation(float
 time)
   {
      rotationZ1From=rotationZ1To;
      rotationZ2From=rotationZ2To;
      rotationZ3From=rotationZ3To;
      rotationZ4From=rotationZ4To;
      rotationZ5From=rotationZ5To;
      rotationZ6From=rotationZ6To;

      Z1=50f;
      Z2=-102f;
      Z3=203f;
      Z4=104f;
      Z5=53f;
      Z6=-68f;

      rotationZ1To = rotationZ1From-(Z1-
Z1PrevLocalRot);   //Positive rotation
      rotationZ2To =-rotationZ2From-
(Z2+Z2PrevLocalRot);   //Negative rotation
      rotationZ3To = rotationZ3From-(Z3-
Z3PrevLocalRot);   //Positive rotation
      rotationZ4To = rotationZ4From-(Z4-
Z4PrevLocalRot);   //Positive rotation
      rotationZ5To = rotationZ5From-(Z5-
Z5PrevLocalRot);   //Positive rotation
      rotationZ6To = rotationZ6From-(Z6-
Z6PrevLocalRot);   //Positive rotation

      Z1PrevLocalRot = Z1;
      Z2PrevLocalRot = Z2;
      Z3PrevLocalRot = Z3;
      Z4PrevLocalRot = Z4;
      Z5PrevLocalRot = Z5;
      Z6PrevLocalRot = Z6;

      yield return StartCoroutine (GoToTarget(rotationZ
1From, rotationZ2From, rotationZ3From, rotationZ4Fr
om, rotationZ5From, rotationZ6From,
         time,
         rota-
tionZ1To, rotationZ2To, rotationZ3To, rotationZ4To, ro
tationZ5To, rotationZ6To));
   }


   pub-
lic IEnumerator FirstToIntermediatePositionRotation(fl
oat time)
   {
      rotationZ1From=rotationZ1To;
      rotationZ2From=rotationZ2To;
      rotationZ3From=rotationZ3To;
      rotationZ4From=rotationZ4To;
      rotationZ5From=rotationZ5To;
      rotationZ6From=rotationZ6To;

      Z1=6.3f;
      Z2=-102f;
      Z3=203f;
      Z4=52f;
      Z5=80f;
      Z6=5f;

      rotationZ1To = rotationZ1From-(Z1-
Z1PrevLocalRot);   //Positive rotation
      rotationZ2To =-rotationZ2From-
(Z2+Z2PrevLocalRot);   //Negative rotation
      rotationZ3To = rotationZ3From-(Z3-
Z3PrevLocalRot);   //Positive rotation
```

```
        rotationZ4To = rotationZ4From-(Z4-
Z4PrevLocalRot);   //Positive rotation
        rotationZ5To = rotationZ5From-(Z5-
Z5PrevLocalRot);   //Positive rotation
        rotationZ6To = rotationZ6From-(Z6-
Z6PrevLocalRot);   //Positive rotation

        Z1PrevLocalRot = Z1;
        Z2PrevLocalRot = Z2;
        Z3PrevLocalRot = Z3;
        Z4PrevLocalRot = Z4;
        Z5PrevLocalRot = Z5;
        Z6PrevLocalRot = Z6;

        yield return StartCoroutine (GoToTarget(rotationZ
1From, rotationZ2From, rotationZ3From, rotationZ4Fr
om, rotationZ5From, rotationZ6From,
            time,
            rota-
tionZ1To, rotationZ2To, rotationZ3To, rotationZ4To, ro
tationZ5To, rotationZ6To));
    }


    pub-
lic IEnumerator IntermediateToSecondPositionRotatio
n(float time)
    {
        rotationZ1From=rotationZ1To;
        rotationZ2From=rotationZ2To;
        rotationZ3From=rotationZ3To;
        rotationZ4From=rotationZ4To;
        rotationZ5From=rotationZ5To;
        rotationZ6From=rotationZ6To;

        Z1=6.3f;
        Z2=-28f;
        Z3=96f;
        Z4=0f;
        Z5=111f;
        Z6=-44f;

        rotationZ1To = rotationZ1From-(Z1-
Z1PrevLocalRot);   //Positive rotation
        rotationZ2To =-rotationZ2From-
(Z2+Z2PrevLocalRot);   //Negative rotation
        rotationZ3To = rotationZ3From-(Z3-
Z3PrevLocalRot);   //Positive rotation
        rotationZ4To = rotationZ4From-(Z4-
Z4PrevLocalRot);   //Positive rotation
        rotationZ5To = rotationZ5From-(Z5-
Z5PrevLocalRot);   //Positive rotation
        rotationZ6To = rotationZ6From-(Z6-
Z6PrevLocalRot);   //Positive rotation

        Z1PrevLocalRot = Z1;
        Z2PrevLocalRot = Z2;
        Z3PrevLocalRot = Z3;
        Z4PrevLocalRot = Z4;
        Z5PrevLocalRot = Z5;
        Z6PrevLocalRot = Z6;

        yield return StartCoroutine (GoToTarget(rotationZ
1From, rotationZ2From, rotationZ3From, rotationZ4Fr
om, rotationZ5From, rotationZ6From,
            time,
            rota-
tionZ1To, rotationZ2To, rotationZ3To, rotationZ4To, ro
tationZ5To, rotationZ6To));
    }


    pub-
lic IEnumerator SecondToConveyorRestPositionRot(flo
at time)
    {
        rotationZ1From=rotationZ1To;
        rotationZ2From=rotationZ2To;
        rotationZ3From=rotationZ3To;
        rotationZ4From=rotationZ4To;
        rotationZ5From=rotationZ5To;
        rotationZ6From=rotationZ6To;

        Z1=-38f;
        Z2=-115f;
        Z3=198f;
        Z4=0f;
        Z5=90f;
        Z6=0f;

        rotationZ1To = rotationZ1From-(Z1-
Z1PrevLocalRot);   //Positive rotation
        rotationZ2To =-rotationZ2From-
(Z2+Z2PrevLocalRot);   //Negative rotation
        rotationZ3To = rotationZ3From-(Z3-
Z3PrevLocalRot);   //Positive rotation
        rotationZ4To = rotationZ4From-(Z4-
Z4PrevLocalRot);   //Positive rotation
        rotationZ5To = rotationZ5From-(Z5-
Z5PrevLocalRot);   //Positive rotation
        rotationZ6To = rotationZ6From-(Z6-
Z6PrevLocalRot);   //Positive rotation

        Z1PrevLocalRot = Z1;
        Z2PrevLocalRot = Z2;
        Z3PrevLocalRot = Z3;
        Z4PrevLocalRot = Z4;
        Z5PrevLocalRot = Z5;
        Z6PrevLocalRot = Z6;

        yield return StartCoroutine (GoToTarget(rotationZ
1From, rotationZ2From, rotationZ3From, rotationZ4Fr
om, rotationZ5From, rotationZ6From,
            time,
            rota-
tionZ1To, rotationZ2To, rotationZ3To, rotationZ4To, ro
tationZ5To, rotationZ6To));
    }
```

```
    pub-
lic IEnumerator GoToTarget(float initRotZ1, float initR
otZ2, float initRotZ3, float initRotZ4, float initRotZ5, flo
at initRotZ6,
        float time,
        float finalRotZ1, float finalRotZ2,float finalRotZ3,fl
oat finalRotZ4,float finalRotZ5,float finalRotZ6)
    {
        j1RotFrom = Quaternion.Euler (rotX1, rotYAll, init
RotZ1);
        j2RotFrom = Quaternion.Euler (rotX2, rotYAll, init
RotZ2);
        j3RotFrom = Quaternion.Euler (rotX3, rotYAll, init
RotZ3);
        j4RotFrom = Quaternion.Euler (rotX4, rotYAll, init
RotZ4);
        j5RotFrom = Quaternion.Euler (rotX5, rotYAll, init
RotZ5);
        j6RotFrom = Quaternion.Euler (rotX6, rotYAll, init
RotZ6);

        j1RotTo = Quaternion.Euler (rotX1, rotYAll, finalRo
tZ1) ;
        j2RotTo = Quaternion.Euler (rotX2, rotYAll, finalRo
tZ2) ;
        j3RotTo = Quaternion.Euler (rotX3, rotYAll, finalRo
tZ3) ;
        j4RotTo = Quaternion.Euler (rotX4, rotYAll, finalRo
tZ4) ;
        j5RotTo = Quaternion.Euler (rotX5, rotYAll, finalRo
tZ5) ;
        j6RotTo = Quaternion.Euler (rotX6, rotYAll, finalRo
tZ6) ;


        var i= 0.0f;
        var rate= 1.0f/time;
        while (i < 1.0f)


        {
            i += Time.deltaTime * rate;
            joint1.localRotation = Quaternion.Slerp (j1RotFr
om, j1RotTo,  i);
            joint2.localRotation = Quaternion.Slerp (j2RotFr
om, j2RotTo,  i);
            joint3.localRotation = Quaternion.Slerp (j3RotFr
om, j3RotTo,  i);
            joint4.localRotation = Quaternion.Slerp (j4RotFr
om, j4RotTo,  i);
            joint5.localRotation = Quaternion.Slerp (j5RotFr
om, j5RotTo,  i);
            joint6.localRotation = Quaternion.Slerp (j6RotFr
om, j6RotTo,  i);

            yield return null;
        }
        De-
bug.Log (" End of MoveToTarget Coroutine at time:"+T
ime.time);

    }
}
```

- **FKMover3**

```
using UnityEngine;
using System.Collections;

public class FKMover3 : MonoBehaviour
{

    // Joints angle range expressed in the robot coordina
te system
    // Afterwards we have to give them in a way that co
rresponds to the unity coordinate system.

    [Range (-160f,160f)] public float rotZ1 = 0;
    [Range (-227.5f,47.5f)] public float rotZ2 = -90;
    [Range (-52.5f,232.5f)] public float rotZ3 = 90;
    [Range (-270f,270f)] public float rotZ4 = 0;
    [Range (-105f,120f)] public float rotZ5 = 0;
    [Range (-270f,270f)] public float rotZ6 = 0;


    // DO READY angles. Changes have been made only
Zangles values as the rest only need to Quaternion fun
ction.
    private float rotX1 = -90f;
    private float rotX2 = -90f;
    private float rotX3 = 0f;
    private float rotX4 = 90f;
    private float rotX5 = -90f;
    private float rotX6 = 90f;

    private float rotYAll = 0f;

    private float initRotZ1 = 0f;
    private float initRotZ2 = -90f;
    private float initRotZ3 = 90f;
    private float initRotZ4 = 0f;
    private float initRotZ5 = 0f;
    private float initRotZ6 = 0f;

    private Transform joint1;
    private Transform joint2;
    private Transform joint3;
    private Transform joint4;
    private Transform joint5;
    private Transform joint6;
```

```
   pri-
vate Transform endEffectorUnityPosision;
   pri-
vate Transform referenceFrameUnityPosition;

   // Starting Rotations of Z axis of every joint
   private float rotationZ1From;
   private float rotationZ2From;
   private float rotationZ3From;
   private float rotationZ4From;
   private float rotationZ5From;
   private float rotationZ6From;

   // Target Rotations of Z axis of every joint
   private float rotationZ1To;
   private float rotationZ2To;
   private float rotationZ3To;
   private float rotationZ4To;
   private float rotationZ5To;
   private float rotationZ6To;

   private Quaternion j1RotFrom;
   private Quaternion j2RotFrom;
   private Quaternion j3RotFrom;
   private Quaternion j4RotFrom;
   private Quaternion j5RotFrom;
   private Quaternion j6RotFrom;

   private Quaternion j1RotTo;
   private Quaternion j2RotTo;
   private Quaternion j3RotTo;
   private Quaternion j4RotTo;
   private Quaternion j5RotTo;
   private Quaternion j6RotTo;

   private float Z1;
   private float Z2;
   private float Z3;
   private float Z4;
   private float Z5;
   private float Z6;

   private float Z1PrevLocalRot;
   private float Z2PrevLocalRot;
   private float Z3PrevLocalRot;
   private float Z4PrevLocalRot;
   private float Z5PrevLocalRot;
   private float Z6PrevLocalRot;


   void Awake ()
   {
      joint1 = GameObject.Find ("Joint 3.1").GetCompo
nent<Transform> ();
      joint2 = GameObject.Find ("Joint 3.2").GetCompo
nent<Transform> ();
      joint3 = GameObject.Find ("Joint 3.3").GetCompo
nent<Transform> ();
      joint4 = GameObject.Find ("Joint 3.4").GetCompo
nent<Transform> ();
      joint5 = GameObject.Find ("Joint 3.5").GetCompo
nent<Transform> ();
      joint6 = GameObject.Find ("Joint 3.6").GetCompo
nent<Transform> ();

   }
   pub-
lic IEnumerator GoToTarget(float initRotZ1, float initR
otZ2, float initRotZ3,
                           float initRotZ4, float initRotZ5, float
 initRotZ6,
                           float time,
                             float finalRotZ1,float finalRotZ2,fl
oat finalRotZ3,
                           float finalRotZ4,float finalRotZ5,floa
t finalRotZ6)
   {


      j1RotFrom = Quaternion.Euler (rotX1, rotYAll, init
RotZ1);
      j2RotFrom = Quaternion.Euler (rotX2, rotYAll, init
RotZ2);
      j3RotFrom = Quaternion.Euler (rotX3, rotYAll, init
RotZ3);
      j4RotFrom = Quaternion.Euler (rotX4, rotYAll, init
RotZ4);
      j5RotFrom = Quaternion.Euler (rotX5, rotYAll, init
RotZ5);
      j6RotFrom = Quaternion.Euler (rotX6, rotYAll, init
RotZ6);

      j1RotTo = Quaternion.Euler (rotX1, rotYAll, finalRo
tZ1) ;
      j2RotTo = Quaternion.Euler (rotX2, rotYAll, finalRo
tZ2) ;
      j3RotTo = Quaternion.Euler (rotX3, rotYAll, finalRo
tZ3) ;
      j4RotTo = Quaternion.Euler (rotX4, rotYAll, finalRo
tZ4) ;
      j5RotTo = Quaternion.Euler (rotX5, rotYAll, finalRo
tZ5) ;
      j6RotTo = Quaternion.Euler (rotX6, rotYAll, finalRo
tZ6) ;


      var i= 0.0f;
      var rate= 1.0f/time;
      while (i < 1.0f)


      {
         i += Time.deltaTime * rate;
         joint1.localRotation = Quaternion.Slerp (j1RotFr
om, j1RotTo,  i);
         joint2.localRotation = Quaternion.Slerp (j2RotFr
om, j2RotTo,  i);
         joint3.localRotation = Quaternion.Slerp (j3RotFr
```

```
om, j3RotTo,  i);
        joint4.localRotation = Quaternion.Slerp (j4RotFr
om, j4RotTo,  i);
        joint5.localRotation = Quaternion.Slerp (j5RotFr
om, j5RotTo,  i);
        joint6.localRotation = Quaternion.Slerp (j6RotFr
om, j6RotTo,  i);

        yield return null;
      }
      De-
bug.Log (" End of MoveToTarget Coroutine at time:"+T
ime.time);

   }

   pub-
lic IEnumerator MoveFromInitialToConveyorRestPositi
on (float time)
   {
      rotationZ1From= 0f;
      rotationZ2From= -90f;
      rotationZ3From= 90f;
      rotationZ4From= 0f;
      rotationZ5From= 0f;
      rotationZ6From= 0f;

      Z1 = 50f;
      Z2=-63f;
      Z3=145f;
      Z4=0f;
      Z5=94f;
      Z6=0f;

      rotationZ1To= rotationZ1From-(Z1-
rotationZ1From);                  //No rotation
      rotationZ2To=-rotationZ2From-
(Z2+rotationZ2From);                  //Negative rotation

      rotationZ3To= rotationZ3From-(Z3-
rotationZ3From);                  //Positive rotation
      rotationZ4To= Z4;      //Positive rotation
      rotationZ5To= rotationZ5From-(Z5-
rotationZ5From);                  //Positive rotation
      rotationZ6To= Z6;     //Positive rotation

      Z1PrevLocalRot = Z1;
      Z2PrevLocalRot=  Z2;
      Z3PrevLocalRot = Z3;
      Z4PrevLocalRot = Z4;
      Z5PrevLocalRot = Z5;
      Z6PrevLocalRot = Z6;


      yield return StartCoroutine (GoToTarget(rotationZ
1From, rotationZ2From, rotationZ3From, rotationZ4Fr
om, rotationZ5From, rotationZ6From,
         time,
         rota-
tionZ1To, rotationZ2To, rotationZ3To, rotationZ4To, ro
tationZ5To, rotationZ6To));

   }

   pub-
lic IEnumerator RestToLoadPosRotation(float time)
   {
      rotationZ1From=rotationZ1To;
      rotationZ2From=rotationZ2To;
      rotationZ3From=rotationZ3To;
      rotationZ4From=rotationZ4To;
      rotationZ5From=rotationZ5To;
      rotationZ6From=rotationZ6To;

      Z1 =50f;
      Z2=-60.22f;
      Z3=145f;
      Z4=0f;
      Z5=92.4f;
      Z6=0f;

      rotationZ1To = rotationZ1From-(Z1-
Z1PrevLocalRot);   //Positive rotation
      rotationZ2To =-rotationZ2From-
(Z2+Z2PrevLocalRot);   //Negative rotation
      rotationZ3To = rotationZ3From-(Z3-
Z3PrevLocalRot);   //Positive rotation
      rotationZ4To = rotationZ4From-(Z4-
Z4PrevLocalRot);   //Positive rotation
      rotationZ5To = rotationZ5From-(Z5-
Z5PrevLocalRot);   //Positive rotation
      rotationZ6To = rotationZ6From-(Z6-
Z6PrevLocalRot);   //Positive rotation

      Z1PrevLocalRot = Z1;
      Z2PrevLocalRot = Z2;
      Z3PrevLocalRot = Z3;
      Z4PrevLocalRot = Z4;
      Z5PrevLocalRot = Z5;
      Z6PrevLocalRot = Z6;

      yield return StartCoroutine (GoToTarget(rotationZ
1From, rotationZ2From, rotationZ3From, rotationZ4Fr
om, rotationZ5From, rotationZ6From,
         time,
         rota-
tionZ1To, rotationZ2To, rotationZ3To, rotationZ4To, ro
tationZ5To, rotationZ6To));
   }

   pub-
lic IEnumerator LoadingIntermediatePosRotation (floa
t time)
   {

      //This will take the values from the previous move
ment so as the new one to begin exactly at the same p
oint.
```

```
        rotationZ1From = rotationZ1To;
        rotationZ2From=rotationZ2To;
        rotationZ3From=rotationZ3To;
        rotationZ4From=rotationZ4To;
        rotationZ5From=rotationZ5To;
        rotationZ6From=rotationZ6To;

        //Rotation values
        Z1=80f;
        Z2=-70f;
        Z3=145f;
        Z4=0f;
        Z5=92.4f;
        Z6=0f;


        // Rotation target for each joint
        rotationZ1To = rotationZ1From-(Z1-
Z1PrevLocalRot);   //Positive rotation
        rotationZ2To =-rotationZ2From-
(Z2+Z2PrevLocalRot);   //Negative rotation
        rotationZ3To = rotationZ3From-(Z3-
Z3PrevLocalRot);   //Positive rotation
        rotationZ4To = rotationZ4From-(Z4-
Z4PrevLocalRot);   //Positive rotation
        rotationZ5To = rotationZ5From-(Z5-
Z5PrevLocalRot);   //Positive rotation
        rotationZ6To = rotationZ6From-(Z6-
Z6PrevLocalRot);   //Positive rotation

        // Here we store the rotation values which will be
useful for the next movement
        Z1PrevLocalRot = Z1;
        Z2PrevLocalRot = Z2;
        Z3PrevLocalRot = Z3;
        Z4PrevLocalRot = Z4;
        Z5PrevLocalRot = Z5;
        Z6PrevLocalRot = Z6;

        yield return StartCoroutine (GoToTarget(rotationZ
1From, rotationZ2From, rotationZ3From, rotationZ4Fr
om, rotationZ5From, rotationZ6From,
            time,
            rota-
tionZ1To, rotationZ2To, rotationZ3To, rotationZ4To, ro
tationZ5To, rotationZ6To));
    }


    pub-
lic IEnumerator IntermediateToUnloadPosRotation (flo
at time)
    {
        rotationZ1From=rotationZ1To;
        rotationZ2From=rotationZ2To;
        rotationZ3From=rotationZ3To;
        rotationZ4From=rotationZ4To;
        rotationZ5From=rotationZ5To;
        rotationZ6From=rotationZ6To;

        Z1=130f;
        Z2=-46f;
        Z3=134.7f;
        Z4=0f;
        Z5=87f;
        Z6=80.0f;

        rotationZ1To = rotationZ1From-(Z1-
Z1PrevLocalRot);   //Positive rotation
        rotationZ2To =-rotationZ2From-
(Z2+Z2PrevLocalRot);   //Negative rotation
        rotationZ3To = rotationZ3From-(Z3-
Z3PrevLocalRot);   //Positive rotation
        rotationZ4To = rotationZ4From-(Z4-
Z4PrevLocalRot);   //Positive rotation
        rotationZ5To = rotationZ5From-(Z5-
Z5PrevLocalRot);   //Positive rotation
        rotationZ6To = rotationZ6From-(Z6-
Z6PrevLocalRot);   //Positive rotation

        Z1PrevLocalRot = Z1;
        Z2PrevLocalRot = Z2;
        Z3PrevLocalRot = Z3;
        Z4PrevLocalRot = Z4;
        Z5PrevLocalRot = Z5;
        Z6PrevLocalRot = Z6;


        yield return StartCoroutine (GoToTarget(rotationZ
1From, rotationZ2From, rotationZ3From, rotationZ4Fr
om, rotationZ5From, rotationZ6From,
            time,
            rota-
tionZ1To, rotationZ2To, rotationZ3To, rotationZ4To, ro
tationZ5To, rotationZ6To));
    }


    pub-
lic IEnumerator TrayLoadingIntermediatePosRotation (
float time)
    {
        rotationZ1From=rotationZ1To;
        rotationZ2From=rotationZ2To;
        rotationZ3From=rotationZ3To;
        rotationZ4From=rotationZ4To;
        rotationZ5From=rotationZ5To;
        rotationZ6From=rotationZ6To;

        Z1=0f;
        Z2=-90f;
        Z3=110f;
        Z4=0f;
        Z5=70f;
        Z6=0f;

        rotationZ1To = -rotationZ1From-
(Z1+Z1PrevLocalRot);   //Positive rotation
```

```
        rotationZ2To =-rotationZ2From-
(Z2+Z2PrevLocalRot);   //Negative rotation
        rotationZ3To = rotationZ3From-(Z3-
Z3PrevLocalRot);   //Positive rotation
        rotationZ4To = rotationZ4From-(Z4-
Z4PrevLocalRot);   //Positive rotation
        rotationZ5To = rotationZ5From-(Z5-
Z5PrevLocalRot);   //Positive rotation
        rotationZ6To = rotationZ6From-(Z6-
Z6PrevLocalRot);   //Positive rotation

        Z1PrevLocalRot = Z1;
        Z2PrevLocalRot = Z2;
        Z3PrevLocalRot = Z3;
        Z4PrevLocalRot = Z4;
        Z5PrevLocalRot = Z5;
        Z6PrevLocalRot = Z6;

        yield return StartCoroutine (GoToTarget(rotationZ
1From, rotationZ2From, rotationZ3From, rotationZ4Fr
om, rotationZ5From, rotationZ6From,
            time,
            rota-
tionZ1To, rotationZ2To, rotationZ3To, rotationZ4To, ro
tationZ5To, rotationZ6To));
    }


    pub-
lic IEnumerator IntermediateToGrabTray1PosRotation
(float time)
    {
        rotationZ1From=rotationZ1To;
        rotationZ2From=rotationZ2To;
        rotationZ3From=rotationZ3To;
        rotationZ4From=rotationZ4To;
        rotationZ5From=rotationZ5To;
        rotationZ6From=rotationZ6To;

        Z1=-90f;
        Z2=-61f;
        Z3=135f;
        Z4=0f;
        Z5=106f;
        Z6=0f;

        rotationZ1To = rotationZ1From-(Z1-
Z1PrevLocalRot);   //Positive rotation
        rotationZ2To =-rotationZ2From-
(Z2+Z2PrevLocalRot);   //Negative rotation
        rotationZ3To = rotationZ3From-(Z3-
Z3PrevLocalRot);   //Positive rotation
        rotationZ4To = rotationZ4From-(Z4-
Z4PrevLocalRot);   //Positive rotation
        rotationZ5To = rotationZ5From-(Z5-
Z5PrevLocalRot);   //Positive rotation
        rotationZ6To = rotationZ6From-(Z6-
Z6PrevLocalRot);   //Positive rotation

        Z1PrevLocalRot = Z1;
        Z2PrevLocalRot = Z2;
        Z3PrevLocalRot = Z3;
        Z4PrevLocalRot = Z4;
        Z5PrevLocalRot = Z5;
        Z6PrevLocalRot = Z6;

        yield return StartCoroutine (GoToTarget(rotationZ
1From, rotationZ2From, rotationZ3From, rotationZ4Fr
om, rotationZ5From, rotationZ6From,
            time,
            rota-
tionZ1To, rotationZ2To, rotationZ3To, rotationZ4To, ro
tationZ5To, rotationZ6To));
    }

    pub-
lic IEnumerator IntermediateToReleaseTrayPosRotatio
n(float time)
    {
        rotationZ1From=rotationZ1To;
        rotationZ2From=rotationZ2To;
        rotationZ3From=rotationZ3To;
        rotationZ4From=rotationZ4To;
        rotationZ5From=rotationZ5To;
        rotationZ6From=rotationZ6To;

        Z1=128.8f;
        Z2=-43f;
        Z3=132.5f;
        Z4=0f;
        Z5=90f;
        Z6=217.6f;

        rotationZ1To = rotationZ1From-(Z1-
Z1PrevLocalRot);   //Positive rotation
        rotationZ2To =-rotationZ2From-
(Z2+Z2PrevLocalRot);   //Negative rotation
        rotationZ3To = rotationZ3From-(Z3-
Z3PrevLocalRot);   //Positive rotation
        rotationZ4To = rotationZ4From-(Z4-
Z4PrevLocalRot);   //Positive rotation
        rotationZ5To = rotationZ5From-(Z5-
Z5PrevLocalRot);   //Positive rotation
        rotationZ6To = rotationZ6From-(Z6-
Z6PrevLocalRot);   //Positive rotation

        Z1PrevLocalRot = Z1;
        Z2PrevLocalRot = Z2;
        Z3PrevLocalRot = Z3;
        Z4PrevLocalRot = Z4;
        Z5PrevLocalRot = Z5;
        Z6PrevLocalRot = Z6;

        yield return StartCoroutine (GoToTarget(rotationZ
1From, rotationZ2From, rotationZ3From, rotationZ4Fr
om, rotationZ5From, rotationZ6From,
            time,
            rota-
```

```
tionZ1To, rotationZ2To, rotationZ3To, rotationZ4To, ro
tationZ5To, rotationZ6To));
   }


   pub-
lic IEnumerator IntermediateToRestPosRotation(float t
ime)
   {
      rotationZ1From=rotationZ1To;
      rotationZ2From=rotationZ2To;
      rotationZ3From=rotationZ3To;
      rotationZ4From=rotationZ4To;
      rotationZ5From=rotationZ5To;
      rotationZ6From=rotationZ6To;

      Z1 = 50f;
      Z2=-63f;
      Z3=145f;
      Z4=0f;
      Z5=94f;
      Z6=0f;

      rotationZ1To = rotationZ1From-(Z1-
Z1PrevLocalRot);   //Positive rotation
      rotationZ2To =-rotationZ2From-
(Z2+Z2PrevLocalRot);   //Negative rotation
      rotationZ3To = rotationZ3From-(Z3-
Z3PrevLocalRot);   //Positive rotation
      rotationZ4To = rotationZ4From-(Z4-
Z4PrevLocalRot);   //Positive rotation
      rotationZ5To = rotationZ5From-(Z5-
Z5PrevLocalRot);   //Positive rotation
      rotationZ6To = rotationZ6From-(Z6-
Z6PrevLocalRot);   //Positive rotation

      Z1PrevLocalRot = Z1;
      Z2PrevLocalRot = Z2;
      Z3PrevLocalRot = Z3;
      Z4PrevLocalRot = Z4;
      Z5PrevLocalRot = Z5;
      Z6PrevLocalRot = Z6;

      yield return StartCoroutine (GoToTarget(rotationZ
1From, rotationZ2From, rotationZ3From, rotationZ4Fr
om, rotationZ5From, rotationZ6From,
         time,
         rota-
tionZ1To, rotationZ2To, rotationZ3To, rotationZ4To, ro
tationZ5To, rotationZ6To));
   }


   pub-
lic IEnumerator IntermediateToGrabTray2PosRotation(
float time)
   {
      rotationZ1From=rotationZ1To;
      rotationZ2From=rotationZ2To;
      rotationZ3From=rotationZ3To;
      rotationZ4From=rotationZ4To;
      rotationZ5From=rotationZ5To;
      rotationZ6From=rotationZ6To;

      Z1=-17f;
      Z2=-61f;
      Z3=135f;
      Z4=0f;
      Z5=106f;
      Z6=-17f;

      rotationZ1To = rotationZ1From-(Z1-
Z1PrevLocalRot);   //Positive rotation
      rotationZ2To =-rotationZ2From-
(Z2+Z2PrevLocalRot);   //Negative rotation
      rotationZ3To = rotationZ3From-(Z3-
Z3PrevLocalRot);   //Positive rotation
      rotationZ4To = rotationZ4From-(Z4-
Z4PrevLocalRot);   //Positive rotation
      rotationZ5To = rotationZ5From-(Z5-
Z5PrevLocalRot);   //Positive rotation
      rotationZ6To = rotationZ6From-(Z6-
Z6PrevLocalRot);   //Positive rotation

      Z1PrevLocalRot = Z1;
      Z2PrevLocalRot = Z2;
      Z3PrevLocalRot = Z3;
      Z4PrevLocalRot = Z4;
      Z5PrevLocalRot = Z5;
      Z6PrevLocalRot = Z6;

      yield return StartCoroutine (GoToTarget(rotationZ
1From, rotationZ2From, rotationZ3From, rotationZ4Fr
om, rotationZ5From, rotationZ6From,
         time,
         rota-
tionZ1To, rotationZ2To, rotationZ3To, rotationZ4To, ro
tationZ5To, rotationZ6To));
   }
}
```

- **ASRSmotion**

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class ASRSmotion : MonoBehaviour {

      public Transform slidingComponent;
      public Transform liftingComponent;
      public Transform loadingComponent1;
      public Transform loadingComponent2;

      public Vector3 xLoadPosition;
      public Vector3 yLoadPosition;
      public Vector3 z1LoadPosition;
      public Vector3 z2LoadPosition;
```

```
    public GameObject[]slidingPositions;
    public GameObject[]liftingPositions;
    public GameObject[]z1LoadingPositions;
    public GameObject[]z2LoadingPositions;

    public GameObject desiredParentGameObject;
    public GameObject targetGameObject;

    public bool firstItemIsReadyForStorage;
  private  bool flagForTheAsrsStart= false;

    pri-
vate ReadyItemMotionManager readyItemManager;
    pri-
vate RawItemMotionManager rawItemManager;

    private AudioSource asrsAlarm;
    int k;
    int l;
    float speed=0.1f;
  void Awake()
  {
    rawItemManag-
er=GameObject.Find ("Decimated Game Objects ").Ge
tComponent<RawItemMotionManager> ();
    asrsAlarm=GameObject.Find ("ASRS").GetCompon
ent<AudioSource> ();
  }

  void Update ()
  {
    flagForTh-
eAsrsStart = rawItemManager.firstItemIsReadyAndPos
itioned;
  }

  IEnumerator Start()
  {
    // The following loop is responsible for the motion
of sliding component of ASRS
    // We assume the storage component as a matrix
with 4 rows and 3 columns.
    // Each storage position has 2 storage slots
    // The first four storage slots are occupied so we d
id the storing coding for the
    // two remaining slots of the first row manually an
d of the other rows within two For loos as follows:


      //First row / Third column / Right unloading


      yield return new WaitUntil(() => flagForTheAsrs
Start == true);
      yield return new WaitForSecondsRealtime(6);
      asrsAlarm.Play ();
      yield return StartCoroutine(    GoToLoadingPosit
ion ());
      asrsAlarm.Pause ();
      yield return StartCoroutine (    CarrierOpen ());


      BoxLoading();
      yield return StartCoroutine (    CarrierClose ());

      asrsAlarm.Play ();
      yield return StartCoroutine(    GoToUnLoadingP
osition (4, 2));
      asrsAlarm.Pause ();
      yield return StartCoroutine (    CarrierOpen ());

    BoxUnLoadingLeft();
      yield return StartCoroutine (    CarrierClose ());

      asrsAlarm.Play ();
      yield return StartCoroutine(    GoToInitialPositio
n (4, 2));
      asrsAlarm.Pause ();



      //First row / Third column / Left unloading
      yield return new WaitUntil(() => flagForTheAsrs
Start == true);
      yield return new WaitForSecondsRealtime(6);
    asrsAlarm.Play ();
      yield return StartCoroutine(    GoToLoadingPosit
ion ());
    asrsAlarm.Pause ();
      yield return StartCoroutine (    CarrierOpen ());


      BoxLoading();
      yield return StartCoroutine (    CarrierClose ());
    asrsAlarm.Play ();
      yield return StartCoroutine(    GoToUnLoadingP
osi-
tion (4, 2));//yield return StartCoroutine(    GoToTheRe
mainingPositionOfFirstRow ());
    asrsAlarm.Pause ();
    yield return StartCoroutine (    CarrierOpen ());

    BoxUnLoadingRight1();
      yield return StartCoroutine (    CarrierClose ());
    asrsAlarm.Play ();
      yield return StartCoroutine(    GoToInitialPositio
n (4, 2)); //yield return StartCoroutine(    GoToInitialPo
sitionAfterTheLastPositionOfFirstRow ());
    asrsAlarm.Pause ();
    //yield return new WaitForSecondsRealtime(24);



      for (int l = 3; l < 6; l++)                    // 10 u
nload-
```

```
ing positions. There are 12 in total but we have two oc
cupied.
        {                                   // Assume that
the first 4 slots (2 positions) in the storage component
are occupied
            for (int k = 2; k < 5; k++)                 // 20 st
or-
age slots. There are 24 in total but we have 4 occupied.
            {

                // Left unloading in the remaining free stor
age slots
                yield return new WaitUntil(() => flagForThe
AsrsStart == true);
                yield return new WaitForSecondsRealtime(
6);
            asrsAlarm.Play ();
                yield return StartCoroutine(    GoToLoading
Position ());
            asrsAlarm.Pause ();
                yield return StartCoroutine (    CarrierOpen
());

                BoxLoading ();
                yield return StartCoroutine (    CarrierClose
());
            asrsAlarm.Play ();
                yield return StartCoroutine(    GoToUnLoadi
ngPosition (k, l));
            asrsAlarm.Pause ();
                yield return StartCoroutine (    CarrierOpen
());

                BoxUnLoadingLeft ();
                yield return StartCoroutine (    CarrierClose
());
            asrsAlarm.Play ();
                yield return StartCoroutine(    GoToInitialPo
sition (k, l));
            asrsAlarm.Pause ();


                // Right unloading in the remaining free sto
rage slots
                yield return new WaitUntil(() => flagForThe
AsrsStart == true);
                yield return new WaitForSecondsRealtime(
6);
            asrsAlarm.Play ();
                yield return StartCoroutine(    GoToLoading
Position ());
            asrsAlarm.Pause ();
                yield return StartCoroutine (    CarrierOpen
());

                BoxLoading ();
                yield return StartCoroutine (    CarrierClose
());
            asrsAlarm.Play ();
                yield return StartCoroutine(    GoToUnLoadi
ngPosition (k, l));
            asrsAlarm.Pause ();
                yield return StartCoroutine (    CarrierOpen
());

                BoxUnLoadingRight ();
                yield return StartCoroutine (    CarrierClose
());
            asrsAlarm.Play ();
                yield return StartCoroutine(    GoToInitialPo
sition (k, l));
            asrsAlarm.Pause();
                //yield return new WaitForSecondsRealtime
(24);

            }
        }
    }


    pub-
lic IEnumerator MoveObject(Transform xTransform, V
ec-
tor3 xStartPos, Vector3 xEndPos, float time, Transform
 yTransform, Vector3 yStartPos, Vector3 yEndPos)
    {
        var i= 0.0f;
        var rate= 1.0f/time;
        while (i < 1.0f)
        {
            //asrsAlarm.Play ();
            i += Time.deltaTime * rate;
            xTrans-
form.localPosition = Vector3.Lerp(xStartPos, xEndPos,
i);
            yTrans-
form.localPosition = Vector3.Lerp(yStartPos, yEndPos,
i);

            yield return null;
//Yield is like return type statement where execution g
ets stopped and go es to the function

        }
 //where it was invoked. After execution, it comes back
 to a point from where it left the execution and starts e
xecuting next lines.

    }


    pub-
lic IEnumerator MoveCarrierOnAxisZ(Transform z1Tra
nsform, Vector3 z1StartPos, Vector3 z1EndPos, float ti
me, Transform z2Transform, Vector3 z2StartPos, Vect
or3 z2EndPos)
    {
```

```
        var i= 0.0f;
        var rate= 1.0f/time;
        while (i < 1.0f)
        {

            i += Time.deltaTime * rate;
            z1Transform.localPosition = Vector3.Lerp(z1Star
tPos, z1EndPos, i);
            z2Transform.localPosition = Vector3.Lerp(z2Star
tPos, z2EndPos, i);
            yield return null;
        }
    }

    pub-
lic IEnumerator GoToLoadingPosition()   //10 secs
    {

        yield return StartCoroutine(
            MoveObject(
                slidingCompo-
nent, slidingPositions[0].transform.localPosition, slidin
gPositions[1].transform.localPosition,
                10.0f,
                liftingCompo-
nent, liftingPositions[0].transform.localPosition, lifting
Positions[1].transform.localPosition));
    }

    pub-
lic IEnumerator GoToUnLoadingPosition(int k, int l)

    {
        yield return StartCoroutine(
            MoveObject(
                slidingCompo-
nent, slidingPositions[1].transform.localPosition, slidin
gPositions[k].transform.localPosition,
                10.0f,
                liftingCompo-
nent, liftingPositions[1].transform.localPosition, lifting
Positions[l].transform.localPosition));
    }

    public IEnumerator GoToInitialPosition (int k, int l)
    {
        yield return StartCoroutine(
            MoveObject(
                slidingCompo-
nent, slidingPositions[k].transform.localPosition, slidin
gPositions[0].transform.localPosition,
                5.0f,
                liftingCompo-
nent, liftingPositions[l].transform.localPosition, liftingP
ositions[0].transform.localPosition));
    }

    void BoxLoading()        // This method is responsibl
e for the active tray to become a child of the carrier co
mponent.
    {
        desiredParentGameOb-
ject = GameObject.Find("z2LoadingComponent");
        targetGameOb-
ject = GameObject.FindWithTag ("ActiveTray");
        targetGameOb-
ject.transform.parent = desiredParentGameObject.tra
nsform;
        targetGameObject.tag = "Untagged";
    }


    void BoxUnLoadingLeft()      //This method assigns t
he active disk on the storage space, to the left.
    {
        desiredParentGameOb-
ject = GameObject.Find("Storage");
        targetGameOb-
ject.transform.parent = desiredParentGameObject.tra
nsform;
        targetGameOb-
ject.transform.Translate (Vector3.back * Time.deltaTi
me * 245.0f * speed);

    }


    void BoxUnLoadingRight()   //This method assigns t
he active disk on the storage space, to the right.
    {
        desiredParentGameOb-
ject = GameObject.Find("Storage");
        targetGameOb-
ject.transform.parent = desiredParentGameObject.tra
nsform;
        targetGameOb-
ject.transform.Translate (Vector3.back * Time.deltaTi
me * 213.0f * speed);
    }

    void BoxUnLoadingRight1()   //This method assigns
the active disk on the storage space, to the right.
    {
        desiredParentGameOb-
ject = GameObject.Find("Storage");
        targetGameOb-
ject.transform.parent = desiredParentGameObject.tra
nsform;
        targetGameOb-
ject.transform.Translate (Vector3.back * Time.deltaTi
me * -240.0f * speed);
    }

    public IEnumerator CarrierOpen ()
    {
        yield return StartCoroutine(
            MoveCarrierOnAxisZ(
                loadingCompo-
```

```
nent1, z1LoadingPositions[0].transform.localPosition,
z1LoadingPositions[1].transform.localPosition,
            2.0f,
            loadingCompo-
nent2, z2LoadingPositions[0].transform.localPosition,
z2LoadingPositions[1].transform.localPosition));
    }


    public IEnumerator CarrierClose ()
    {
        yield return StartCoroutine(
            MoveCarrierOnAxisZ(
                loadingCompo-
nent1, z1LoadingPositions[1].transform.localPosition,
z1LoadingPositions[0].transform.localPosition,
                2.0f,
                loadingCompo-
nent2, z2LoadingPositions[1].transform.localPosition,
z2LoadingPositions[0].transform.localPosition));
    }
}
```