ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΜΗΧΑΝΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΤΩΝ ΚΑΤΕΡΓΑΣΙΩΝ

Υποστήριξη διαδικασιών προετοιμασίας εργαλειομηχανών με τη βοήθεια Επαυξημένης Πραγματικότητας

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΤΖΙΜΑΣ ΕΥΑΓΓΕΛΟΣ

Επιβλέπων : ΓΕΩΡΓΙΟΣ-ΧΡΙΣΤΟΦΟΡΟΣ ΒΟΣΝΙΑΚΟΣ

ΚΑΘΗΓΗΤΗΣ

Αθήνα, Μάρτιος 2016

Περιεχόμενα

Ευχαριστίες

Σε αυτό το σημείο θα ήθελα να εκφράσω από καρδιάς τις ευχαριστίες μου στον κ. Γ.-Χ. Βοσνιάκο, καθηγητή του Ε.Μ.Π. για την άμεση και καθοριστική υποστήριξη που παρείχε σε όλα τα στάδια εκπόνησης της παρούσας εργασίας.

Ευχαριστώ επίσης τους ανθρώπους που δουλέψαμε μαζί αυτούς τους μήνες στο εργαστήριο για το φιλόξενο κλίμα που έχουν δημιουργήσει και ιδιαιτέρως τους Ηλία Μάτσα και Γιώργο Παπαζέτη για τις πολύτιμες οδηγίες που μου παρείχαν.

Τέλος, δεν θα μπορούσα να παραλείψω τους γονείς μου, τον αδερφό μου, τη Γωγώ και το Χάρη τους οποίους ευχαριστώ πολύ για όλη την στήριξη που μου παρείχαν καθ' όλη την διάρκεια των σπουδών μου.

Τζίμας Ευάγγελος
Αθήνα, Μάρτιος 2016

Περίληψη

Στο πλαίσιο της παρούσας διπλωματικής εργασίας διερευνώνται μέθοδοι δημιουργίας εφαρμογών που αξιοποιούν τεχνολογία επαυξημένης πραγματικότητας στον κατασκευαστικό κλάδο. Το ζήτημα προσεγγίζεται με την αναλυτική παρουσίαση των σταδίων ανάπτυξης δυο εφαρμογών, που λειτουργούν υποστηρικτικά στη διαδικασία προετοιμασίας εργαλειομηχανών.

Για την υπέρθεση των εικονικών μοντέλων σε κατάλληλα σημεία του φυσικού χώρου, χρησιμοποιήθηκε προκαθορισμένη εικόνα ως στόχος προς αναγνώριση και παρακολούθηση, καθώς η θέση του αποτελεί το σημείο αναφοράς / διασύνδεσης εικονικού και πραγματικού περιβάλλοντος. Ο καθορισμός της αναγνώρισης της εικόνας εκτελέστηκε με χρήση του λογισμικού Vuforia SDK ενώ η ανάπτυξη των εφαρμογών έγινε στο περιβάλλον του λογισμικού Unity3d. Η γλώσσα προγραμματισμού που χρησιμοποιείται για τον καθορισμό της συμπεριφοράς των τρισδιάστατων αντικειμένων είναι η C#. Ως συσκευή απεικόνισης χρησιμοποιείται οθόνη ηλεκτρονικού υπολογιστή ενώ η εικόνα του φυσικού χώρου λαμβάνεται από συμβατική web κάμερα.

Για την αύξηση της διάδρασης της εφαρμογής με το φυσικό περιβάλλον μελετήθηκε η χρήση πρόσθετων αισθητήρων χαμηλού κόστους. Για αυτό το σκοπό δημιουργήθηκαν διατάξεις που φέρουν τα απαραίτητα ηλεκτρονικά στοιχεία για την επεξεργασία του σήματος από τον αισθητήρα και το οδηγούν σε πλακέτα ArduinoUno. Η πλακέτα arduino μετατρέπει το σήμα του αισθητήρα σε ψηφιακή πληροφορία, η οποία τροφοδοτείται σε ηλεκτρονικό υπολογιστή μέσω σειριακής θύρας για να εισαχθεί στην εφαρμογή και να αξιοποιηθεί από αυτήν.

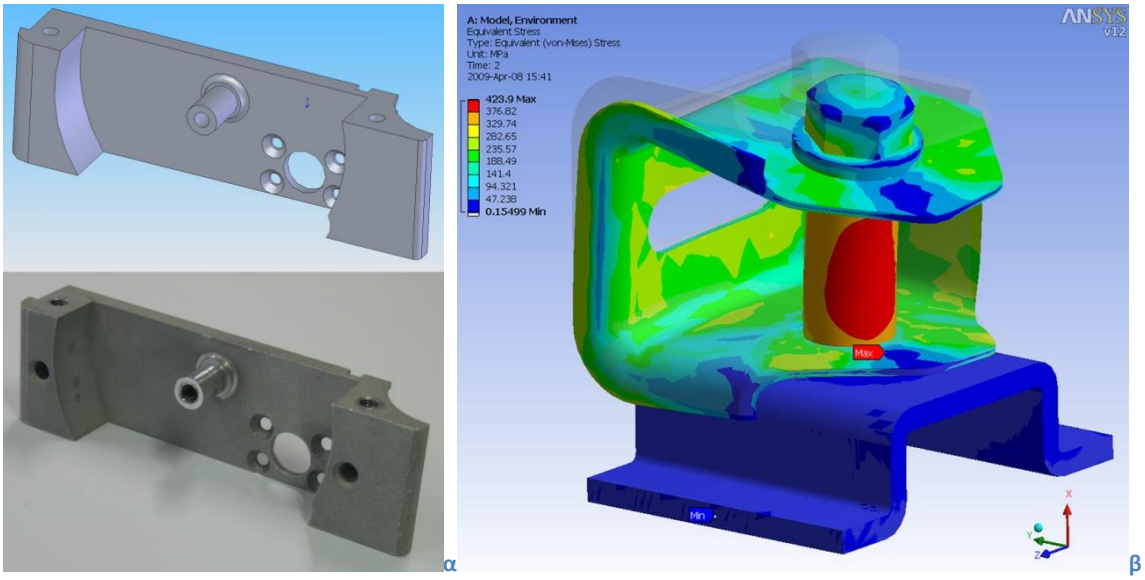
Πίνακας Εικόνων

1. Εισαγωγή

1.1 Πρόλογος

Η ραγδαία ανάπτυξη των υπολογιστών τις τελευταίες δεκαετίες, τους έχει καταστήσει αναπόσπαστο εργαλείο όλων των εφαρμοσμένων επιστημών. Η μηχανολογία φυσικά δεν αποτελεί εξαίρεση καθώς οι Η/Υ έχουν εισχωρήσει στο πεδίο της, απλοποιώντας σημαντικά επίπονες και χρονοβόρες διαδικασίες.

Χαρακτηριστικό παράδειγμα αποτελούν πακέτα λογισμικού CAD (Computer – aided Design), CAE (Computer – aided Engineering) και CAM (Computer – aided Manufacturing) που έχουν κυριαρχήσει στο σύγχρονο κατασκευαστικό τομέα. Αυτά σε συνδυασμό με την εκτεταμένη χρήση CNC μηχανών μας δείχνουν το βαθμό στον οποίο οι δυνατότητες που παρέχει η επιστήμη των υπολογιστών αξιοποιούνται σε τομείς που άπτονται του αντικειμένου του μηχανολόγου μηχανικού (βλ. Εικόνα 1.1) .



Εικόνα 1.1 – (α) CAD μοντέλο και τελικό τεμάχιο (β)Ανάλυση παραμορφώσεων στο ANSYS

Ωστόσο, το ανταγωνιστικό περιβάλλον επιβάλει συνεχή εξέλιξη και έτσι τα τελευταία χρόνια στον κατασκευαστικό και σχεδιαστικό κλάδο των μηχανικών έχει δοθεί έμφαση, ερευνητικά κυρίως, στην αξιοποίηση κάποιων νέων τεχνολογιών, όπως η εικονική και η επαυξημένη πραγματικότητα, η οποία χρησιμοποιείται στην παρούσα διπλωματική εργασία.

1.2 Επαυξημένη Πραγματικότητα

1.2.1 Ορισμός

Με τον όρο επαυξημένη πραγματικότητα (Augmented Reality) ονομάζουμε την αποτύπωση σε πραγματικό χρόνο του φυσικού χώρου, τα στοιχεία του οποίου '*επαυξάνονται'* (υπερτίθενται) με εποπτικά μέσα που δημιουργούνται μέσω υπολογιστή, όπως ήχος, βίντεο, γραφικά και άλλα.

Σαν αποτέλεσμα, η τεχνολογία αυτή λειτουργεί με το να βελτιώνει την αντίληψη του χρήστη για την πραγματικότητα, αντίθετα με την εικονική πραγματικότητα, η οποία αντικαθιστά πλήρως το φυσικό περιβάλλον με ένα εικονικό (αποτέλεσμα προσομοίωσης).

Ένα πολύ απλό παράδειγμα επαυξημένης πραγματικότητας είναι η ένδειξη του σκορ στην τηλεόραση σε μια αθλητική εκδήλωση. Ωστόσο με χρήση τεχνικών Computer Vision για αναγνώριση αντικειμένων η πληροφορία που δέχεται ο χρήστης για τον περιβάλλοντα χώρο γίνεται διαδραστική και ψηφιακά χειραγωγήσιμη.[1]

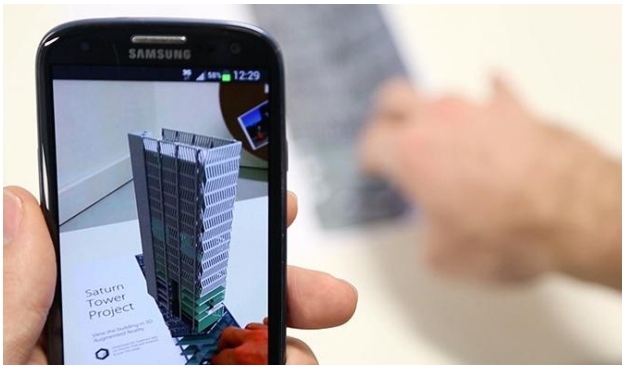
1.2.2 Εφαρμογές

Η τεχνολογία επαυξημένης πραγματικότητας έχει εισχωρήσει δυναμικά σε πολλούς επαγγελματικούς κλάδους, αποτελώντας ένα πολύ χρήσιμο εργαλείο. Χαρακτηριστικά παραδείγματα εφαρμογών ακολουθούν και δείχνουν την κατεύθυνση στην οποία κινείται η εκμετάλλευση των νέων δυνατοτήτων για το κάθε πεδίο.

- ***Στην αρχαιολογία*** : Εφαρμογή AR χρησιμοποιείται δίνοντας τη δυνατότητα στο χρήστη να επαναφέρει εικονικά κτήρια (βλ. Εικόνα 1.2), τοπία ακόμα και χαρακτήρες όπως υπήρξαν. [2][3]

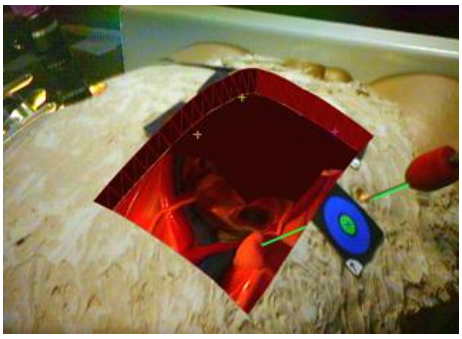
Εικόνα 1.2 – AR app για επίσκεψη αρχαιολογικού χώρου

- ***Στην αρχιτεκτονική*** : Η επαυξημένη πραγματικότητα μπορεί να βοηθήσει στην καλύτερη οπτικοποίηση αρχιτεκτονικών project. Η εικονιζόμενη εφαρμογή για smartphone δίνει τη δυνατότητα 3Δ απεικόνισης ενός κτηρίου που εμφανίζεται δυναμικά όταν ο χρήστης περάσει το χέρι του πάνω από τη σελίδα του περιοδικού που το αποτυπώνει (βλ. Εικόνα 1.3). [1],[4]



Εικόνα 1.3 – 3Δ απεικόνιση 2Δ σχεδίου

- ***Στην ιατρική:*** Αναπτύσσονται πολλές εφαρμογές που στοχεύουν στη βελτίωση της εκπαίδευσης του ιατρικού προσωπικού ή ακόμα και στο να βοηθούν στην εκτέλεση ιατρικών διαδικασιών. Τυπικό παράδειγμα AR εφαρμογής στην ιατρική είναι η υποφαινόμενη στην επόμενη εικόνα, που επιτρέπει στο γιατρό να βλέπει οπτικοποιημένα σε 3Δ μορφή τα αποτελέσματα του υπερηχογραφήματος πάνω στο σώμα του υπό εξέταση ασθενούς.[5]



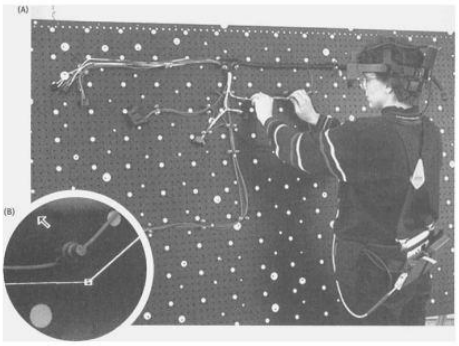
Εικόνα 1.4 – UNC Laparoscopic Visualization Research

Αντίστοιχες εφαρμογές χρησιμοποιούνται για πλοήγηση οχημάτων, για εκπαιδευτικούς σκοπούς, για διαφημιστικές εταιρίες, στη βιομηχανία βιντεοπαιχνιδιών και σε πληθώρα άλλων τομέων.

Παρά τη μεγάλη επιτυχία ωστόσο της εν λόγω τεχνολογίας στους κλάδους που αναφέραμε, η χρήση της σε βιομηχανικό επίπεδο δεν έχει εδραιωθεί ακόμα. Το γεγονός αυτό οφείλεται σε μια σειρά τεχνικών δυσκολιών και περιορισμών που θα αναλυθούν σε επόμενη παράγραφο, οι όποιες όμως αίρονται σταδιακά με την εξέλιξη των Η/Υ και των περιφερειακών τους. Γι αυτόν ακριβώς το σκοπό η αξιοποίηση τεχνολογίας επαυξημένης πραγματικότητας σε βιομηχανικό και ημι-βιομηχανικό περιβάλλον αποτελεί ένα σύγχρονο και σημαντικό αντικείμενο έρευνας πολλών ιδρυμάτων και πεδίο στο οποίο επενδύουν αρκετές επιχειρήσεις.

Πιο συγκεκριμένα η συμβολή εφαρμογών AR μπορεί να είναι πολύ σημαντική σε πολλές φάσεις λειτουργίας ενός βιομηχανικού περιβάλλοντος. Για παράδειγμα στη σχεδίαση ή στην τροποποίηση προϊόντων παρέχεται η δυνατότητα στο σχεδιαστή να εποπτεύσει το σχεδιασμό και τη λειτουργικότητα ενός υπό ανάπτυξη προϊόντας πριν την ολοκλήρωσή του. Επιπρόσθετα μπορεί να αποτελέσει βασικό βοήθημα για το ανθρώπινο δυναμικό που εκτελεί εργασία συντήρησης ή συναρμολόγησης εξοπλισμού. Αξίζει λοιπόν να εστιάσουμε την προσοχή μας σε χαρακτηριστικές εφαρμογές που δίνουν νέες προοπτικές στον κατασκευαστικό και σχεδιαστικό κλάδο της βιομηχανίας.

Πρωτοπόρος σε χρήση εφαρμογής AR είναι η *Boeing*, η οποία σε συνεργασία με τη CMU, τη Honeywell και τη Virtual Vision τη δεκαετία του 90 ανέπτυξε λογισμικό σαν βοήθημα για το έργο της συναρμολόγησης των πλεγμάτων καλωδίων των αεροσκαφών (βλ. Εικόνα 1.5), εργασία δύσκολη και διαφοροποιημένη για κάθε τύπο αεροσκάφους. Κάθε σύστημα που χρησιμοποιούσε την εν λόγω εφαρμογή διέθετε ένα φορητό υπολογιστή, μια κάσκα στερεοσκοπίας (Head Mounted Display -HMD), και ένα σύστημα αναγνώρισης εικόνας. Μελέτες ωστόσο που έγιναν συγκρίνοντας την αρχική μέθοδο συναρμολόγησης με την νέα δεν έδειξαν γι αυτή την πιλοτική εφαρμογή σημαντικό χρονικό πλεονέκτημα στην εκτέλεση του καθήκοντος. [6]



Εικόνα 1.5 – Boeing Wire Bundle application

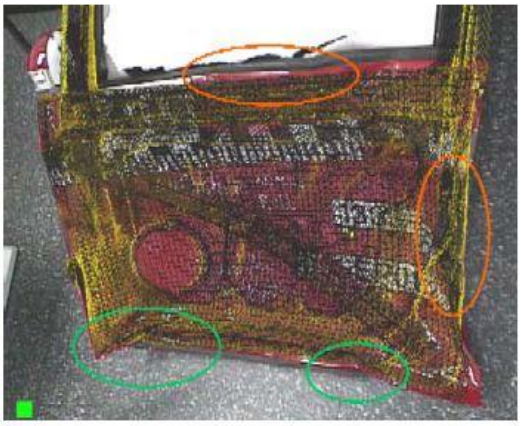
Σε αυτό το πνεύμα κινούνται και μεγάλες σύγχρονες αυτοκινητοβιομηχανίες, όπως η Audi η οποία χρησιμοποιεί την εφαρμογή *Window to the World* αναπτυγμένη από την *ART* προκειμένου να βελτιώσει τις συνθήκες ανάπτυξης νέων μοντέλων. Στη φάση ανάπτυξης νέων οχημάτων τα πρωτότυπα συναρμολογούνται χειροκίνητα από χειριστές. Δυο από τις μεγαλύτερες προκλήσεις για τους εργαζομένους αποτελούν η σωστή και ακριβής τοποθέτηση εύκαμπτων τμημάτων και η τελική επιβεβαίωση ότι όλα τα εξαρτήματα είναι τοποθετημένα στη σωστή θέση. Παραδοσιακά οι εργαζόμενοι εκτελούσαν αυτά τα στάδια συμβουλευόμενοι τα CAD μοντέλα του εκάστοτε οχήματος, διαδικασία αρκετά χρονοβόρα. Με τη χρήση του εν λόγω λογισμικού παρέχεται η δυνατότητα υπέρθεσης του CAD μοντέλου πάνω στο υπό συναρμολόγηση όχημα με αποτέλεσμα ο εργαζόμενος να μπορεί να λάβει άμεσα την επιθυμητή πληροφορία. [7]

Εικόνα 1.6 – Window to the World application

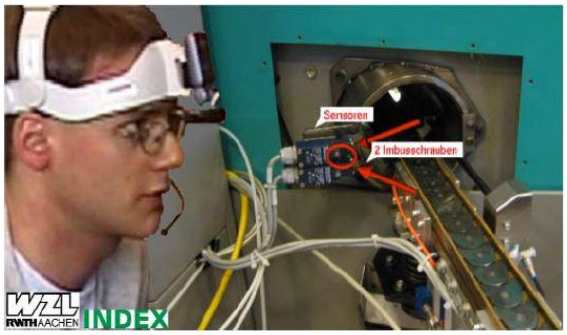
Η *ARVIKA* είναι ένας από τους πλέον σημαντικούς οργανισμούς που εργάζεται στην ανάπτυξη εφαρμογών AR εστιασμένων στην κάλυψη αναγκών της αυτοκινητο-βιομηχανίας και της αεροπορικής βιομηχανίας σε πολλαπλά επίπεδα. Στο κομμάτι του σχεδιασμού ενός οχήματος και κατά την διάρκεια των crash test, τεχνολογία AR επιτρέπει την άμεση σύγκριση των αριθμητικά υπολογισμένων παραμορφώσεων με τις πραγματικές παραμορφώσεις του δοκιμαζόμενου οχήματος. Αυτό επιτυγχάνεται με την υπέρθεση του υπολογισμένου με τη μέθοδο των πεπερασμένων διαφορών πλέγματος πάνω στο παραμορφωμένο πλέον όχημα. Έτσι αμέσως γίνονται ορατές τυχούσες αποκλίσεις.

Εφαρμογές της ARVIKA στοχεύουν στην υποβοήθηση του τομέα της συντήρησης εκτελώντας διάγνωση βλαβών και αποτυπώνοντας εικονικά σχετικό βοηθητικό υλικό

που ανασύρεται από βάση δεδομένων. Ταυτόχρονα δίνει τη δυνατότητα απομακρυσμένης βοήθειας από εξειδικευμένο προσωπικό. [8]



α



β

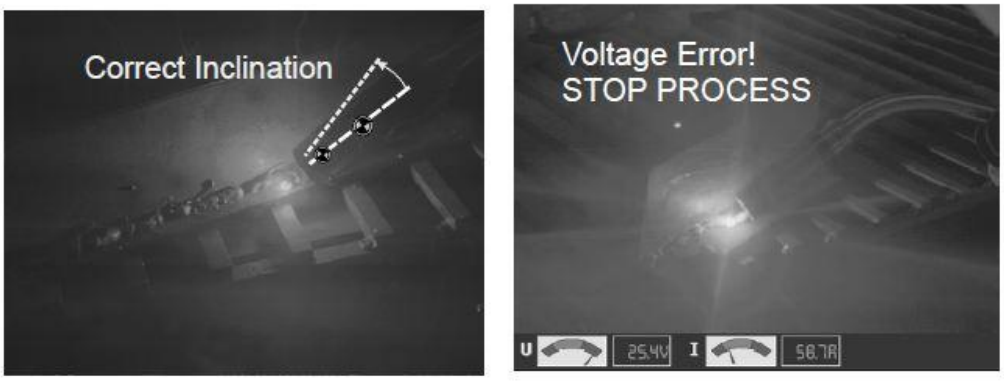
Εικόνα 1.7 –(α),(β) Εφαρμογές της ARVIKA

Και άλλες εφαρμογές αναπτύσσονται για την υποβοήθηση του έργου συντήρησης όπως αυτή που προτείνεται από το πανεπιστήμιο της Σιγκαπούρης για δημιουργία ενός δικτύου που να συλλέγει πληροφορίες από όλα τα τμήματα ενός μηχανουργείου σε πραγματικό χρόνο και να παρέχει απαραίτητες πληροφορίες διαδραστικά μέσω επαυξημένης πραγματικότητας στο προσωπικό (βλ. Εικόνα 1.8). [9]

Εικόνα 1.8 – Manufacturing Grid application

Μια ενδιαφέρουσα εφαρμογή για τη βελτίωση των συνθηκών συγκόλλησης έχει αναπτυχθεί από τους *Aitenau, Hillers* και *Graser* (*Institute of Automation, University of Bremen*). Ο συγκολλητής φοράει ένα νέο κράνος, μέσα από το οποίο 'επαυξάνεται' η οπτική πληροφορία πριν και κατά τη διάρκεια της συγκόλλησης.

Η εικόνα βελτιώνεται παρέχοντας καλύτερη εποπτεία του χώρου εργασίας ενώ ταυτόχρονα ένας online οδηγός είναι διαθέσιμος κατά τη διάρκεια της συγκόλλησης προτείνοντας διορθώσεις στη θέση του πιστολιού συγκόλλησης (βλ. Εικόνα 1.9) , οι οποίες αποτελούν αποτέλεσμα ανάλυσης των ηλεκτρικών παραμέτρων. [10]



Εικόνα 1.9 – Εφαρμογή για Συγκολλήσεις

Η τεχνολογία επαυξημένης πραγματικότητας είναι ένα ισχυρό εργαλείο επίσης, για την εκπαίδευση προσωπικού σε σύνθετες εργασίες που περιλαμβάνουν πρόσθετο μηχανολογικό εξοπλισμό, καθώς σε πολλές περιπτώσεις τα εγχειρίδια μπορεί να είναι δυσνόητα. Είναι προφανές ότι η προβολή στο χώρο του εργαζομένου μιας σειράς οδηγιών με χρήση γραφικών έχει σημαντικό πλεονέκτημα καθώς αφήνει λιγότερο χώρο παρερμηνειών από το αντίστοιχο έντυπο σενάριο. Επιπρόσθετα μπορεί να επιταχύνει το χρόνο εκπαίδευσης ενός ειδικευόμενου εργάτη και να μειώσει τις ώρες που δαπανά ένας έμπειρος στην εκπαίδευσή του.

1.2.3 Απαιτήσεις σε υλικό και λογισμικό

Γενικά, για τη χρήση μιας εφαρμογής επαυξημένης πραγματικότητας είναι απαραίτητη η χρήση hardware που να δίνει σαν έξοδο οπτικοακουστικό υλικό, αισθητήρων που να ελέγχουν τις επιθυμητές παραμέτρους του φυσικού περιβάλλοντος και λογισμικό που να δέχεται την είσοδο από τους αισθητήρες να την επεξεργάζεται κατάλληλα και να την αποδίδει σε επιθυμητή μορφή. Σε αυτό το σημείο θα αναλύσουμε σύντομα τις πλέον χρησιμοποιούμενες συσκευές. [11]

Σίγουρα οι πιο διαδεδομένες συσκευές που πληρούν τις παραπάνω προϋποθέσεις είναι τα *smartphones* και τα *tablets* καθώς περιλαμβάνουν στην πλειοψηφία τους κάμερα, αισθητήρες (γυροσκόπιο, επιταχυνσιόμετρο, GPS και άλλα) και διαθέτουν λειτουργικό σύστημα που επιτρέπει την εύκολη προσθήκη διαφόρων εφαρμογών. Σε συνδυασμό με την ενεργειακή αυτονομία και την εύκολη μεταφορά τους καθίστανται εξαιρετικά δημοφιλείς. [1]

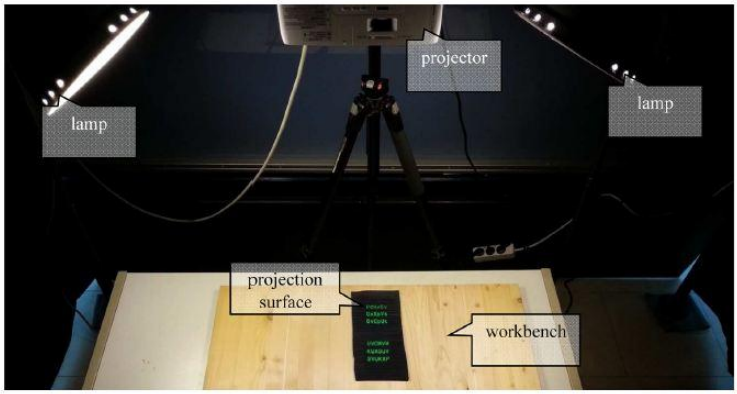
Πολύ συνηθισμένη είναι η χρήση συσκευών Head Mounted Display – *HMD* για την αποτύπωση της εικόνας ως περιφερειακό κάποιου ηλεκτρονικού υπολογιστή. Αυτή η εναλλακτική έχει το πλεονέκτημα ότι ο χρήστης έχει πλέον τα χέρια του ελεύθερα, καθώς η συσκευή προβολής HMD τοποθετείται στο κεφάλι του χρήστη. Οι συσκευές αυτές διακρίνονται στις αδιαφανείς (opaque), στις οποίες ο χρήστης βλέπει μέσω βίντεο το μείγμα του φυσικού περιβάλλοντος (το οποίο λαμβάνεται από ενσωματωμένη κάμερα στην συσκευή HMD) με τα εικονικά μοντέλα και στις ημιδιαφανείς (optical see – through) . Στη δεύτερη περίπτωση, ημιδιαφανείς καθρέφτες τοποθετούνται μπροστά στα μάτια του χρήστη επιτρέποντάς του αφενός να βλέπει τον φυσικό χώρο που τον περιβάλλει, αφετέρου αντικατοπτρίζουν το εικονικό μοντέλο που παράγεται από τον υπολογιστή. [12]

Από την άλλη πλευρά, και στις δυο περιπτώσεις προκύπτουν κάποια προβλήματα. Το πρώτο από αυτά αφορά στο πεδίο όρασης του χρήστη το οποίο είναι περιορισμένο. Ειδικά για τις διαφανείς HMD ανακύπτει και η πρόσθετη δυσκολία της ρύθμισης της αντίθεσης της εικόνας προκειμένου τα εικονικά μοντέλα να είναι ευκρινή, πρόβλημα που αντιμετωπίζεται δύσκολα ιδίως σε εξωτερικό χώρο. Για τις αδιαφανείς HMD είναι ευκολότερη η επίλυση του προβλήματος της αντίθεσης της εικόνας καθώς το βίντεο είναι ευκολότερο να υποστεί επεξεργασία, αλλά η ποιότητα της εικόνας του περιβάλλοντος είναι χειρότερη με αποτέλεσμα να προκαλείται κόπωση στο χρήστη από εκτεταμένη χρήση τέτοιας συσκευής. [13]

Εικόνα 1.10 – Optical see-through HMD, opaque HMD εικόνες από Wikipedia.org

Ωστόσο, σαν συσκευή εξόδου εικόνας σε αρκετές περιπτώσεις χρησιμοποιείται και η οθόνη ενός ηλεκτρονικού υπολογιστή. Σε αυτή την περίπτωση η εικόνα του φυσικού χώρου προέρχεται από χρήση κάμερας, αλλά υπάρχει το μειονέκτημα της δυσκολίας στη μεταφορά της οθόνης, οπότε αυτή η εναλλακτική προτιμάται σε ειδικές περιπτώσεις.

Ένας από τους βασικούς λόγους που εφαρμογές επαυξημένης πραγματικότητας δεν είναι αρκετά διαδομένες στη βιομηχανία είναι η πολύ συνηθισμένη απαίτηση για χρήση κάποιας συσκευής προβολής την οποία ο χρήστης καλείται να μεταφέρει, γεγονός που από εργονομικής σκοπιάς είναι μακριά από το βέλτιστο καθώς θέτει περιορισμούς στην κίνηση και στην άνεση του χρήστη. Σε αναζήτηση απάντησης αυτού του προβλήματος γίνονται έρευνες προκειμένου να χρησιμοποιηθούν προβολείς ή άλλες συσκευές που θα δίνουν στο χρήστη την πληροφορία που επιθυμεί χωρίς να απαιτείται ο ίδιος να περιορίζεται από επιπλέον εξοπλισμό και το προβαλλόμενο ερέθισμα δεν θα επηρεάζεται σημαντικά από τις συνθήκες του περιβάλλοντος χώρου (φωτισμός, υφή επιφάνειας προβολής). [11]



Εικόνα 1.11 – Εφαρμογή με χρήση προβολέα ως συσκευή εξόδου εικόνας

Εκτός από το hardware που αναφέρθηκε, οι εφαρμογές AR διαθέτουν απαραίτητα λογισμικό το οποίο με διάφορες τεχνικές computer vision εκτελεί αναγνώριση εικόνας. Συνήθως πρώτα αναγνωρίζονται κάποια σημεία ενδιαφέροντος ή στόχοι που έχουν τοποθετηθεί εντός του χώρου που βλέπει η κάμερα, αυτά χρησιμεύουν ως σημείο αναφοράς για την εκτέλεση της εφαρμογές και παρακολουθούνται με χρήση κατάλληλων αλγορίθμων.

Οι εφαρμογές ανάλογα και με το πλήθος και την ποιότητα των αισθητήρων που χρησιμοποιούν, με βάση την ακρίβεια με την οποία επιτελούν το έργο τους (τοποθέτηση στη σωστή θέση και με ακριβή προσανατολισμό των εικονικών

αντικειμένων),την ταχύτητα με την οποία λειτουργούν οι αλγόριθμοι αναγνώρισης και παρακολούθησης της εικόνας και τον ρεαλισμό του αποτελέσματος. Περισσότερα για την αναγνώριση εικόνας και την παρακολούθηση στόχων ακολουθούν σε επόμενο κεφάλαιο. [1], [15]

1.3 Στόχος και διάρθρωση διπλωματικής εργασίας

Η παρούσα εργασία έχει σκοπό την ανάπτυξη εφαρμογών επαυξημένης πραγματικότητας για την υποστήριξη του σχεδιασμού και της λειτουργίας συστημάτων κατεργασιών.

Αρχικό στάδιο, είναι η διερεύνηση των εναλλακτικών επιλογών που έχουμε στη διάθεση μας για την ανάπτυξη εφαρμογών AR και η αξιολόγησή τους με κριτήριο οι εφαρμογές να έχουν την επιθυμητή αξιοπιστία, κατά το δυνατόν ελάχιστο κόστος κατασκευής και να υποστηρίζουν εύκολα μελλοντική επέκτασή τους.

Για κάθε εφαρμογή αρχικά ορίζουμε το πρόβλημα του οποίου την επίλυση θα κληθεί να βοηθήσει και κατόπιν παρουσιάζουμε την μέθοδο υλοποίησης που προτείνουμε καθώς και τα μέσα που χρησιμοποιούμε για να την επιτύχουμε.

Συνοπτικά σε αυτό το σημείο αναφέρεται το software και το hardware που χρησιμοποιούμε για την ανάπτυξη των εφαρμογών.

- Unity 3D
- Vuforia SDK for Unity
- Arduino IDE
- Μια συμβατική Web Camera
- Πλακέτα Arduino Uno
- SolidWorks
- 3d's Max

Η γλώσσα προγραμματισμού που χρησιμοποιείται στο Unity 3D είναι η C# , ενώ για τον προγραμματισμό του μικροελεγκτή του Arduino χρησιμοποιούμε C.

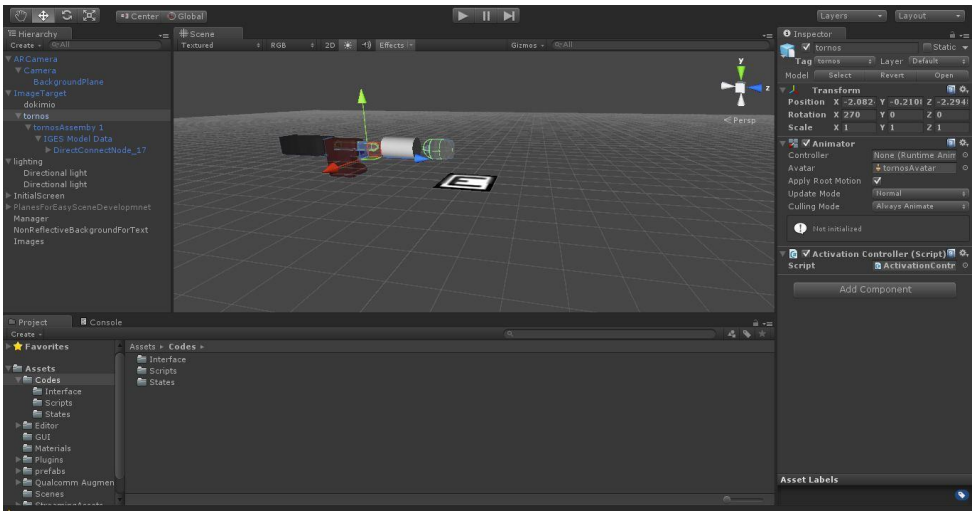
2. Unity 3D

2.1 Περιγραφή

Η ανάπτυξη εφαρμογής που να χρησιμοποιεί τρισδιάστατα εικονικά μοντέλα που να αλληλεπιδρούν με το χρήστη, είναι μια διαδικασία που απαιτεί γενικά πολλαπλά frameworks. Το πρόβλημα αυτό παρακάμπτεται με τη χρήση του λογισμικού Unity 3D, το οποίο είναι μια πλατφόρμα ανάπτυξης βιντεοπαιχνιδιών.

Δημιουργήθηκε 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 15 πλατφόρμες, καθιστώντας το ένα από τα πλέον δημοφιλή περιβάλλοντα του είδους. [16]

Το Unity υποστηρίζεται άρτια από εκτενή τεκμηρίωση και διαθέτει μεγάλη και ενεργή κοινότητα προγραμματιστών, που διευκολύνει την επίλυση τεχνικών ζητημάτων. Επιπρόσθετα μπορεί να δεχθεί εικονικά μοντέλα που έχουν κατασκευαστεί με χρήση άλλου λογισμικού σε μορφή fbx και obj, και υποστηρίζει plug-ins όπως το Vuforia SDK για ταχεία ανάπτυξη εφαρμογών επαυξημένης πραγματικότητας και άλλων για προσθήκη και προγραμματισμό περιφερειακών συσκευών.



Εικόνα 2.1 – Περιβάλλον Unity

Υποστηρίζει για την ανάπτυξη εφαρμογών τις ακόλουθες γλώσσες προγραμματισμού:

- JavaScript
- C#
- Boo (βασισμένη στην Python)
- Unity Script

Επιτρέπει τη χρήση διαφορετικών εξωτερικών IDE (Integrated Development Environment), με προεπιλεγμένο το MonoDevelop το οποίο και χρησιμοποιούμε στην παρούσα εργασία σε συνδυασμό με τη γλώσσα C# για τον προγραμματισμό της συμπεριφοράς των εικονικών αντικειμένων.

2.2 C#

Η επιλογή μιας αντικειμενοστραφούς γλώσσας προγραμματισμού αποτελεί μονόδρομο για την ανάπτυξη εφαρμογών στο Unity. Η C# είναι μία ολοκληρωμένη αντικειμενοστραφής γλώσσα προγραμματισμού σχεδιασμένη για τη δημιουργία λογισμικού σε .Net Framework. Το βασικό πλεονέκτημα είναι η άμεση πρόσβαση που παρέχει σε μεγάλη ποικιλία βιβλιοθηκών και η πλήρης υποστήριξή της τόσο από το Unity όσο και από τη Microsoft.

Ο κώδικας που γράφεται σε C# οργανώνεται σε Namespaces. Ένα Namespace μπορεί να περιέχει έναν από τους εξής τύπους:

- Namespaces
- Classes
- Delegates
- Enums
- Structs
- Interfaces

Ένας βασικός λόγος που χρησιμοποιούμε αντικειμενοστραφείς γλώσσες προγραμματισμού είναι ότι επιτρέπουν την αξιοποίηση εννοιών, όπως της κληρονομικότητας (inheritance) και της αφαιρετικής γραφής κώδικα (abstraction), δυνατότητες που έχουν πολλαπλά οφέλη όπως η εύκολη ανάγνωση και τροποποίηση των προγραμμάτων και η επαναχρησιμοποίηση κλάσεων.

Ακολουθεί ένα παράδειγμα που εισάγει αυτές τις έννοιες, και δείχνει τον τρόπο που δομείται ένα απλό πρόγραμμα.

```
using System;

namespace Inheritance
{
    class Vehicle
    {
        // Member variables
        // Εδώ μπορούμε να δηλώσουμε τις μεταβλητές
        // που χαρακτηρίζουν το αντικείμενο της κλάσης
        private int cost;

        // Class constructor section
        // Κάθε φορά που δημιουργείται ένα αντικείμενο αυτής
        // της κλάσης καλείται ο Constructor
        public Vehicle(int cost)
        {
            this.cost = cost;
        }

        // public Method of the class
        // Αυτή είναι η μέθοδος που κληθεί όταν δημιουργηθεί
        // αντικείμενο της κλάσης
        public int GetCost()
        {
            return cost;
        }
    }
}
```

Οι κλάσεις που κληρονομούν από την κλάση Vehicle θα έχουν κληρονομήσει και τις public μεθόδους της, δηλαδή την GetCost() και θα μπορούν να έχουν πρόσβαση στον constructor της κλάσης.

Θα δημιουργήσουμε λοιπόν δυο νέες κλάσεις παιδιά της Vehicle.

```
using System;

namespace Inheritance
{
    // Αυτή η κλάση είναι παιδί της κλάσης Vehicle
    class Car:Vehicle
    {
        public Car(int carCost):base(carCost)
        {
            Console.WriteLine("This car costs; "+GetCost());
        }
    }
}
```

```
using System;

namespace Inheritance
{
    // Αυτή η κλάση είναι παιδί της κλάσης Vehicle
    class Plane:Vehicle
    {
        public Plane(int planeCost):base(planeCost)
        {
            Console.WriteLine("This Plane costs; "+GetCost());
        }
    }
}
```

Κατόπιν μπορούμε να εκτελέσουμε τον κώδικα που ακολουθεί και να πάρουμε τα αποτελέσματα που φαίνονται.

```
using System;

namespace Inheritance
{
    public class Program
    {
        public static void Main(string[] args)
        {
            Plane onePlane = new Plane(10000000);
            Console.WriteLine();
            Car typicalCar = new Car(8000);
            Console.WriteLine("Press any key to terminate");
            Console.ReadKey();
        }
    }
}
```

```
This Plane costs; 10000000

This car costs; 8000
Press any key to terminate
```

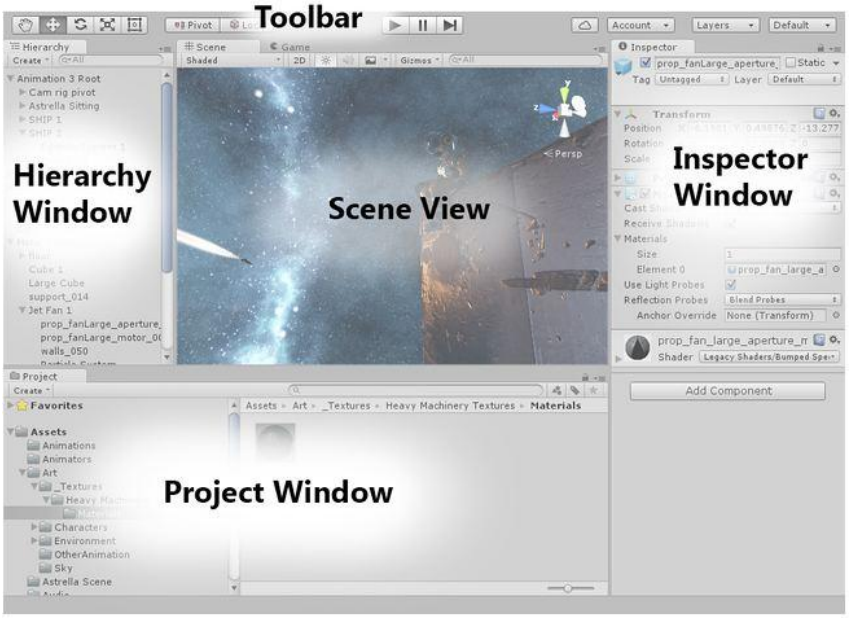
Σε αυτό το σημείο δε θα αναλύσουμε περαιτέρω τη γλώσσα και τα χαρακτηριστικά της, καθώς αυτό δεν αποτελεί στόχο της διπλωματικής εργασίας και σε κάθε περίπτωση δεν μπορεί να περιοριστεί σε ένα κεφάλαιο. Απαραίτητα στοιχεία όμως, θα περιγράφονται κατά περίπτωση στο σημείο όπου τα χρησιμοποιούμε για να υλοποιήσουμε συγκεκριμένο έργο.

2.3 Λειτουργία του λογισμικού

Το Unity παρέχει στο χρήστη τη δυνατότητα να δημιουργεί *Projects*, κάθε ένα εκ των οποίων συντίθεται από έναν αριθμό "σκηνών" (*scene*). Η κάθε σκηνή απαρτίζεται από έναν αριθμό *Game Objects* τα οποία με τη σειρά τους αποτελούν μια ομάδα *Components.*

Ξεκινώντας αντίστροφα τα *components* είναι αντικείμενα κάποιας κλάσης και καθένα συνοδεύεται από δικές του μεθόδους και ιδιότητες. Υπάρχει μια σημαντική ποικιλία έτοιμων *components* που παρέχει το Unity και υπάρχει και η επιλογή να δημιουργηθούν νέες κλάσεις από τον προγραμματιστή, τα αντικείμενα των οποίων μπορούν να είναι *components* . Η αξιοποίηση αυτών των κλάσεων γίνεται με την προσάρτησή τους σε κάποιο *game object* . [17]

Ουσιαστικά τα game objects είναι κλάσεις containers που αποκτούν τη συμπεριφορά από τα *components* που έχουν προσαρτημένα πάνω τους. Αφού όμως είναι αντικείμενα κλάσεων μπορούν να έχουν και σχέση κληρονομικότητας μεταξύ τους. Το περιβάλλον του Unity είναι δομημένο κατά τρόπο τέτοιο που να απεικονίζει άμεσα τις σχέσεις των αντικειμένων που απαρτίζουν την κάθε σκηνή και δίνει επίσης σημαντικό πλήθος *game objects* με προσαρτημένα τα κατάλληλα *components* επιταχύνοντας με αυτό τον τρόπο την ανάπτυξη των εφαρμογών. [18] Αυτό γίνεται σαφές στην εικόνα που έπεται.



Εικόνα 2.2 – Διαμόρφωση GUI του Unity από http://docs.unity3d.com/Manual/LearningtheInterface.html

Στο αριστερό παράθυρο (hierarchy window) βρίσκονται σε μορφή λίστας όλα τα game objects της σκηνής. Τα game objects απεικονίζονται δενδρικά, υποδεικνύοντας με αυτόν τον τρόπο τη σχέση μεταξύ των αντικειμένων. Τα game objects που εμφανίζονται στην κορυφή κάθε δένδρου είναι αντικείμενα γονείς των υπολοίπων. Στο κεντρικό παράθυρο (scene view) δίνεται η δυνατότητα πλοήγησης στην εικονική σκηνή όπου είναι ευδιάκριτη η χωροθέτηση και ο προσανατολισμός των αντικειμένων στο χώρο και σε συνδυασμό με τα κουμπιά που βρίσκονται στη γραμμή εργαλείων που φαίνεται στο επάνω μέρος της εικόνας μπορούμε να τα τροποποιήσουμε. Στο παράθυρο στα δεξιά της εικόνας (inspector window) φαίνονται για το επιλεγμένο game object τα components που το απαρτίζουν, κάθε ένα εκ των οποίων έχει ένα πλήθος παραμέτρων και επιλογών διαθέσιμο προς τροποποίηση από το χρήστη. Ειδικά για την περίπτωση των script components, στο inspector window εμφανίζονται οι public member μεταβλητές του κάθε script. Στο κάτω μέρος της εικόνας φαίνεται το project window, όπου τοποθετούνται όλα τα αρχεία του project οργανωμένα σε φακέλους και υποφακέλους.

2.3.1 Βασικά Game Objects και Components

Όπως ήδη αναφέραμε τα game objects στο Unity είναι τα δομικά στοιχεία της σκηνής και υπάρχει μια ποικιλία έτοιμων τέτοιων στοιχείων που παρέχει το λογισμικό. Σε αυτή την ενότητα παρουσιάζονται τα σημαντικότερα για την παρούσα εργασία έτοιμα game objects και τρόποι που χρησιμοποιούνται.

Αρχικά μπορούν να χρησιμοποιηθούν βασικές γεωμετρικές μορφές, να παραμετροποιηθούν και να συνδυαστούν ανάλογα με τις ανάγκες της κάθε εφαρμογής. Τα στοιχεία αυτά είναι:

- Κύβος
- Σφαίρα
- Κάψουλα
- Κύλινδρος
- Επίπεδο

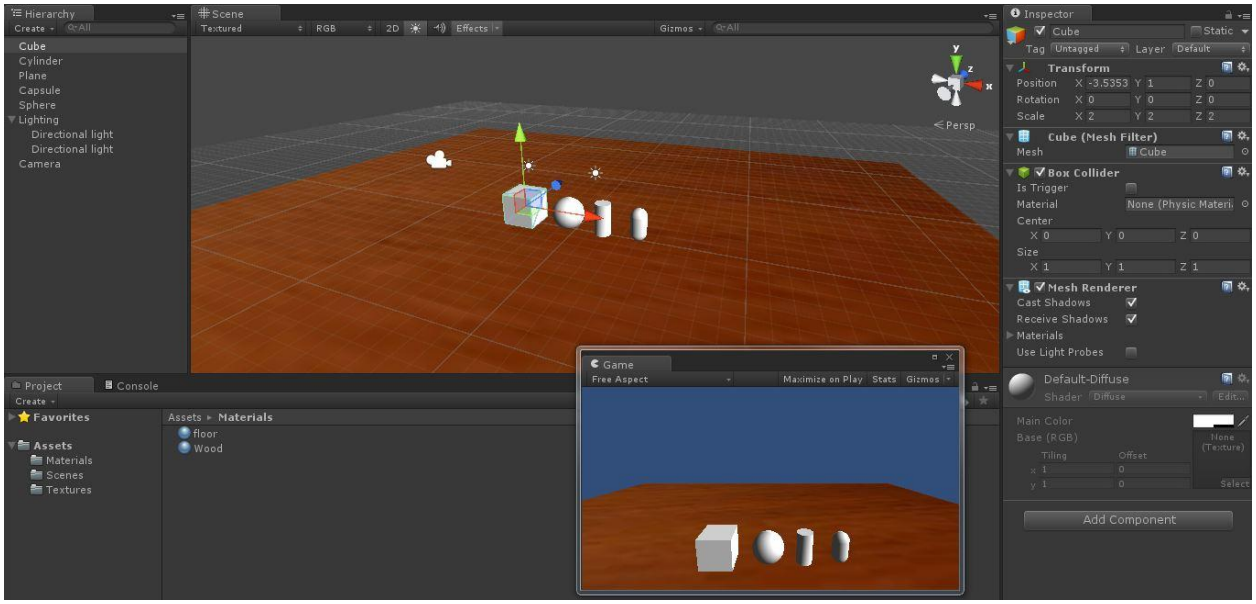
Διατίθενται επίσης game objects υπεύθυνα για το φωτισμό της σκηνής:

- Directional Light
- Point Light
- Spot Light
- Area Light

Τέλος ένα ιδιαίτερα σημαντικό game object είναι η ***κάμερα***, αντικείμενο υπεύθυνο για το οπτικό πεδίο του χρήστη εντός σκηνής κατά την εκτέλεση της εφαρμογής.

Κάθε ένα από αυτά τα αντικείμενα διαθέτει τα κατάλληλα components προκειμένου να είναι λειτουργικό και εύκολα τροποποιήσιμο. Διατίθενται επίσης και game objects κενά , δηλαδή χωρίς components που τους αποδίδουν συγκεκριμένη συμπεριφορά, τα οποία αξιοποιούνται πρωτίστως για να ομαδοποιούν game objects των οποίων ο ρόλος είναι συμπληρωματικός και με αυτόν τον τρόπο δημιουργούνται συνθετότερα αντικείμενα στη σκηνή με σχέσεις κληρονομικότητας να τα συνδέουν.

Οι έννοιες που εισήχθησαν γίνονται πιο σαφείς αν παρατηρήσουμε την εικόνα που ακολουθεί όπου παρουσιάζεται μια σκηνή με τα βασικά game objects που αναφέραμε.



Εικόνα 2.3 – Παράδειγμα Unity

Παρατηρούμε λοιπόν πως συντίθεται μια απλή σκηνή. Έχοντας επιλέξει το game object 'Cube' μπορούμε να δούμε στο δεξί μέρος της εικόνας πως συνοδεύεται από τέσσερα πολύ βασικά components, των οποίων τη λειτουργία θα εξηγήσουμε καθώς χρησιμοποιούνται κατ επανάληψη. Αυτά είναι τα ακόλουθα:

- Transform
- Mesh Filter
- Box Collider
- Mesh Renderer

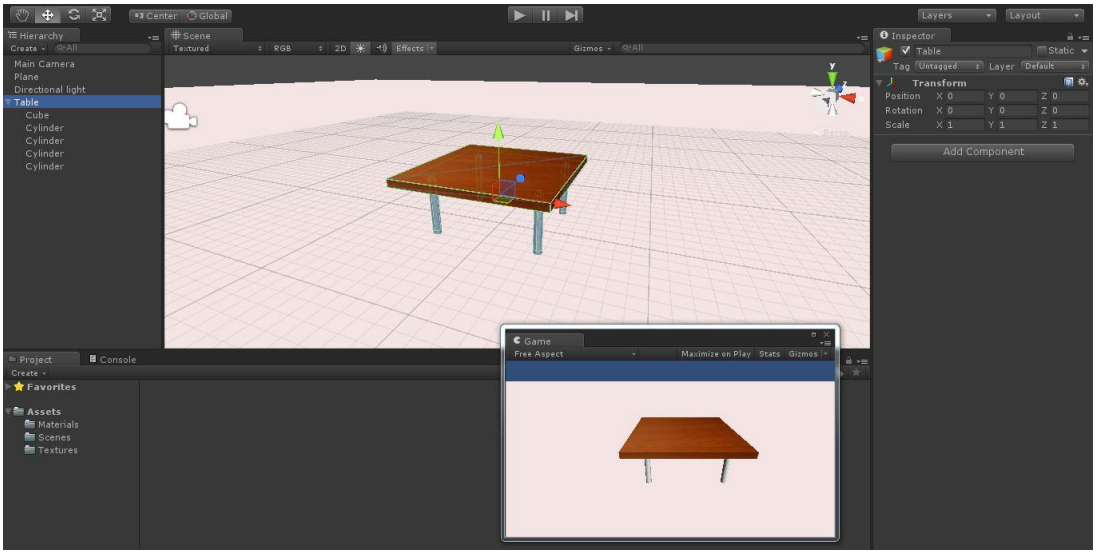
Το ***Transform component*** συναντάται σχεδόν σε όλα τα game objects και καθορίζει τη θέση, τον προσανατολισμό και τις διαστάσεις του αντικειμένου.

Το ***Mesh Filter component*** συνδέει το αντικείμενο με ένα ορισμένο πλέγμα προκειμένου να μπορεί να απεικονιστεί η μορφή του και οι επιφάνειές του με τον επιθυμητό τρόπο.

Το ***Box Collider component*** είναι ένα επίσης βασικό στοιχείο, καθώς σε έναν όγκο ελέγχου ορθογωνίου παραλληλεπιπέδου που περιβάλει τον κύβο, μπορεί να εντοπίσει άλλα αντικείμενα που εισέρχονται, εξέρχονται ή βρίσκονται σε αυτόν.

Το ***Mesh Renderer*** component είναι υπεύθυνο για τον τρόπο με τον οποίο θα αποτυπώνονται στην οθόνη κατά την εκτέλεση τις εφαρμογής οι μορφές που είναι συνδυασμένες με το κάθε game object. Υπάρχει η δυνατότητα αυτό το component να συνδυαστεί με διάφορα υλικά κάνοντας την απεικόνιση αρκετά πιο ρεαλιστική.

Παρατηρούμε επίσης ότι εκτός από τα προεπιλεγμένα παραμετροποιήσιμα components που απαρτίζουν το game object, δίνεται η δυνατότητα να προστεθούν και άλλα βελτιώνοντας τις δυνατότητες του κάθε game object. Για παράδειγμα προσθέτοντας το ***Rigid Body*** component σε ένα game object, δίνεται η δυνατότητα να προστεθούν σε αυτό φυσικές ιδιότητες όπως μάζα, να ασκηθούν επάνω του δυνάμεις και να υπολογισθεί η επιτάχυνση που οφείλεται στη συνισταμένη τους. Μένει να δείξουμε το πώς δημιουργούνται πιο σύνθετα αντικείμενα, οπότε τροποποιούμε τη σκηνή όπως φαίνεται την εικόνα στην εικόνα που ακολουθεί.



Εικόνα 2.4 – Unity Παράδειγμα 2

Βλέπουμε στη νέα σκηνή ότι το αρχικά άδειο game object με όνομα table λειτουργεί ως γονέας πέντε βασικών αντικειμένων δηλαδή ενός παραμετροποιημένου κύβου και τεσσάρων τροποποιημένων κυλίνδρων. Πλέον αν τροποποιηθεί το Transform component του αντικείμενου – γονέα θα ενημερωθούν αυτόματα τα αντίστοιχα components των αντικειμένων – παιδιών και έτσι το τραπέζι που δημιουργήθηκε στη σκηνή μπορεί να παραμετροποιηθεί χωρίς να χρειάζεται κάποια παρέμβαση στα επιμέρους στοιχεία που το απαρτίζουν.

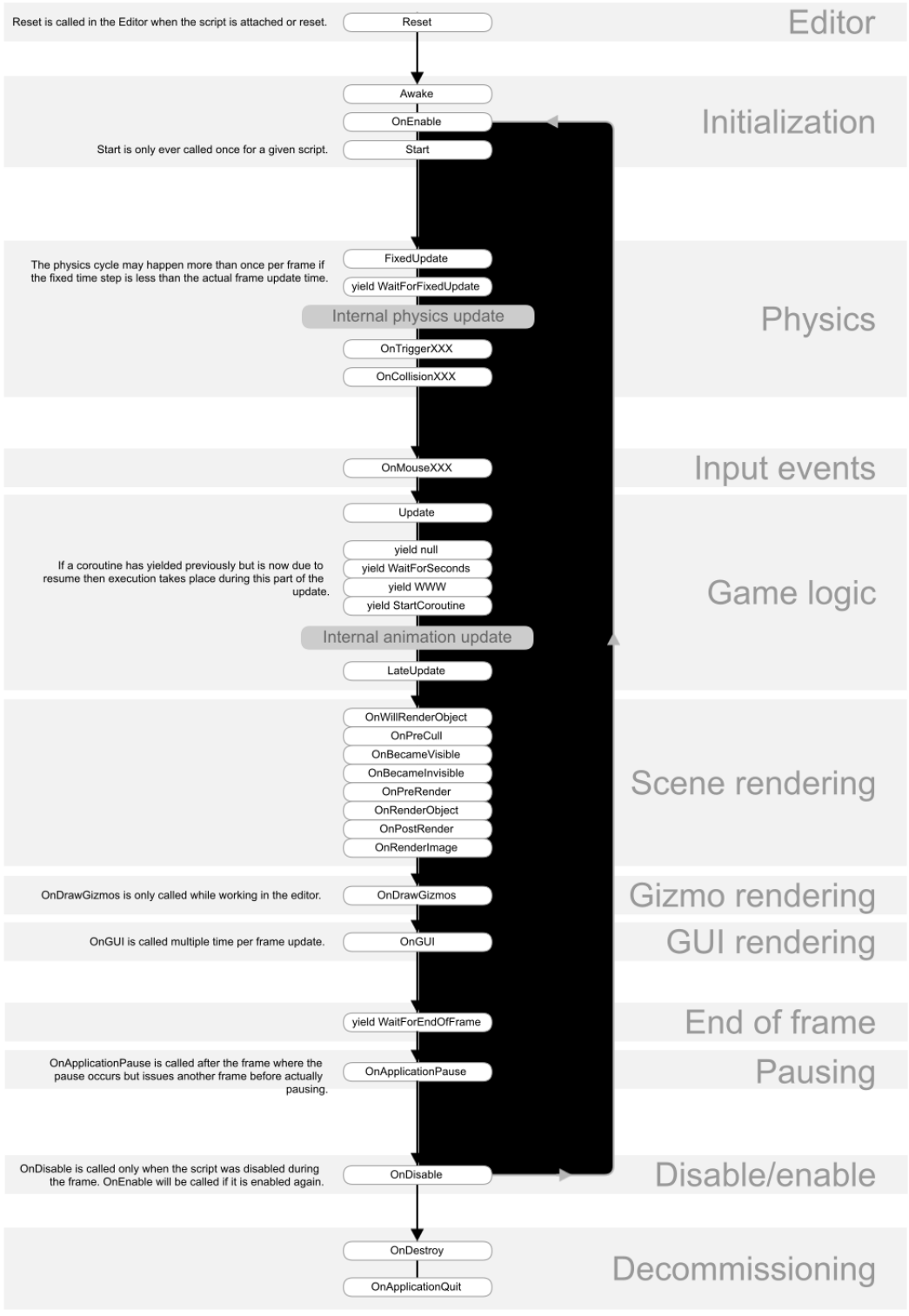
2.3.2 Scripts – Behaviour Components

Προκειμένου η σκηνή και τα αντικείμενα που την απαρτίζουν να αποκτήσουν διαδραστική συμπεριφορά είναι επιβεβλημένη η δημιουργία script components που να λαμβάνουν κάποια είσοδο και να παράγουν κάποια έξοδο στη σκηνή κατά τη διάρκεια της εκτέλεσης του προγράμματος. Η δυνατότητα αυτή που παρέχεται στο Unity είναι και ο κύριος λόγος που επιλέξαμε το συγκεκριμένο λογισμικό.

Κάθε component που βρίσκεται στη σκηνή συνοδεύεται από μεθόδους που δίνουν πρόσβαση στις δημόσια προσβάσιμες μεταβλητές του (public variables) και με αυτό τον τρόπο η παραμετροποίησή του μπορεί να συμβεί δυναμικά κατά την εκτέλεση της εφαρμογής μέσω ενός ή περισσότερων script components. Να σημειωθεί ότι οι δημόσια προσβάσιμες μεταβλητές είναι αυτές που εμφανίζονται στην καρτέλα inspector.

Τα script components δεν είναι τίποτα περισσότερο από κλάσεις παιδιά της MonoBehaviour κλάσης του Unity. Αυτό σημαίνει ότι κληρονομούνται οι μέθοδοι Start( ) , Awake( ) που χρησιμοποιούνται στην αρχικοποίηση παραμέτρων του component καθώς και οι μέθοδοι Update( ), FixedUpdate( ), που εκτελούνται κάθε frame και η OnGUI( ). Επίσης δίνεται η δυνατότητα δημιουργίας και εκτέλεσης Coroutines, μέθοδοι που μπορούν να διακόπτουν την εκτέλεσή τους πριν αυτή ολοκληρωθεί και να την επανεκκινούν από το σημείο που σταμάτησε. Απαραίτητη είναι επίσης η χρήση του namespace UnityEngine προκειμένου να είναι ορατές οι κλάσεις του Unity με τις μεθόδους που τις συνοδεύουν. Μια ειδική κατηγορία μεθόδων είναι οι event functions, οι οποίες περιμένουν κάποιου είδους γεγονός για να εκτελέσουν το τμήμα του κώδικα που είναι γραμμένος σε αυτές, τέτοιες είναι η OnMouseDown( ), On TriggerEnter( ), OnCollisionEnter( ) και άλλες.

Το διάγραμμα ροής που ακολουθεί δείχνει την αλληλουχία με την οποία εκτελούνται οι μέθοδοι της κλάσης MonoBehaviour.



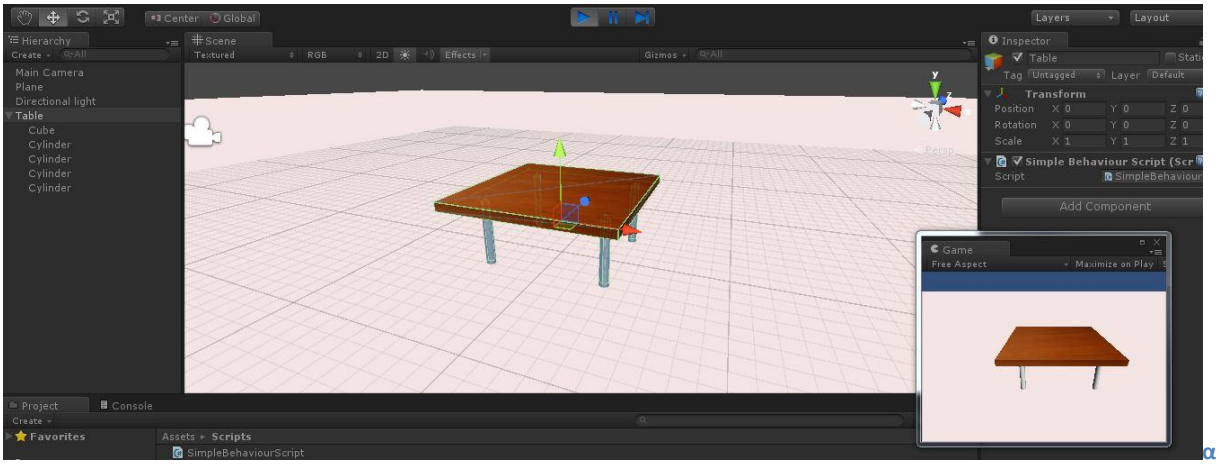
Εικόνα 2.5 – Διάγραμμα Ροής MonoBehaviour από http://docs.unity3d.com/uploads/Main/monobehaviour_flowchart.svg

Για να γίνει σαφές το νόημα χρήσης script components θα γυρίσουμε στο παράδειγμα με το τραπέζι. Εάν επιθυμούμε το τραπέζι να αλλάζει προσανατολισμό μετά από προτροπή του χρήστη της εφαρμογής, πρέπει να δημιουργηθεί ένα script component που θα προστεθεί στο game object Table. Αρχικά δημιουργούμε το script στο MonoDevelop.

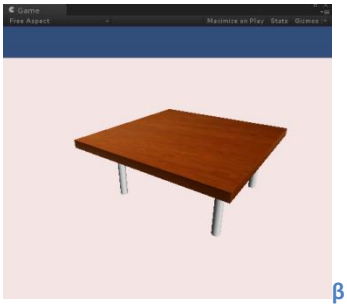
```
1 using UnityEngine;
2 using System.Collections;
3
4 // Δημιουργια Script που θα προστεθει ως Component στο
5 // Game Object: Table
6 public class SimpleBehaviourScript : MonoBehaviour {
7
8     // Δημιουργια μιας μεταβλητης τυπου Transform και μιας τυπου float
9     private Transform tableRotator;
10    private float speed;
11    // Η μεθοδος Start χρησιμοποιειται για
12    // αρχικοποιηση των μεταβλητων
13    void Start ()
14    {
15        // Με τη μεθοδο GetComponent<Tranform> αποκτουμε
16        // προσβαση στο Transform Component που ειναι
17        // προσαρτημενο στο ιδιο Game Object με το
18        // SimpleBehaviourScript Component
19        tableRotator = GetComponent<Transform> ();
20        speed = 20.0f;
21    }
22
23
24    // Η μεθοδος Update καλειται μια φορα ανα frame
25    void Update ()
26    {
27        // Η συνθηκη ελεγχει αν πατιεται το κουμπι space στο πληκτρολογιο
28        if(Input.GetKey(KeyCode.Space))
29        {
30            // Η επομενη γραμμη ειναι υπευθυνη για την φορα και
31            // την ταχυτητα περιστροφης του τραπεζιου
32            tableRotator.transform.Rotate(new Vector3(0,speed*Time.deltaTime,0) );
33        }
34    }
35
36 }
```

Κατόπιν το αποθηκεύουμε, επιλέγουμε το Table Game Object και από την καρτέλα Inspector επιλέγουμε Add Component ⇨ Scripts ⇨ SimpleBehaviourScript και εκτελούμε την εφαρμογή πατώντας το κουμπί Play από τη γραμμή εργαλείων.

Πλέον το τραπέζι έχει αποκτήσει τη συμπεριφορά να περιστρέφεται όταν πατιέται το Space (βλ. Εικόνα 2.6).



α

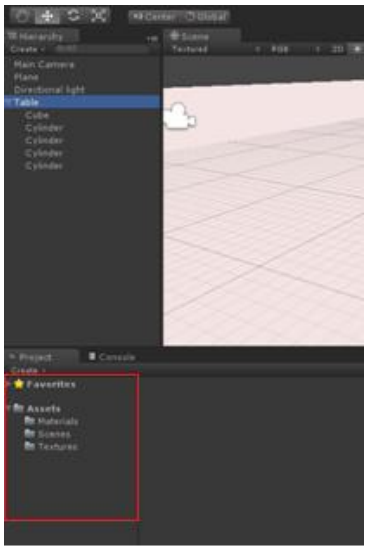


β

Εικόνα 2.6 – Unity Παράδειγμα 3

2.3.3 Assets

Τα στοιχεία που χρησιμοποιούνται για τη δημιουργία ενός project στο Unity3D (σκηνές, υλικά, εικόνες, script components και άλλα) ονομάζονται Assets και είναι οργανωμένα σε ένα σύστημα φακέλων που εμφανίζεται κάτω αριστερά στην οθόνη του Unity, όπως φαίνεται στην επόμενη εικόνα.



Εικόνα 2.7 – Καρτέλα Assets στο Unity3D

Υπάρχει η δυνατότητα εισαγωγής Assets από εξωτερική πηγή, ή από κάποιο άλλο project του Unity γεγονός που επιταχύνει την ανάπτυξη εφαρμογών καθώς υπάρχει πληθώρα στοιχείων διαθέσιμων που διαφορετικά η δημιουργία τους θα ήταν εξαιρετικά χρονοβόρα.

Επίσης εκμεταλλευόμαστε αυτό το χαρακτηριστικό προκειμένου να δημιουργήσουμε πιο σύνθετα τρισδιάστατα αντικείμενα από τα προκαθορισμένα με χρήση άλλου λογισμικού. Η σχεδίαση των μοντέλων σε αυτή την εργασία έγινε σε SolidWorks και 3Ds Max και κατόπιν εισήχθησαν στο Unity σε μορφή .fbx ή .obj .

3. Προσθήκη Εξωτερικών Συσκευών και Αισθητήρων

3.1 Εισαγωγή

Σε πολλές εφαρμογές επαυξημένης πραγματικότητας μπορεί να είναι απαραίτητη διάδραση με το περιβάλλον που δεν καλύπτεται από την αναγνώριση εικόνας. Εάν, για παράδειγμα, μια εφαρμογή που εκτελείται σε μια φορητή συσκευή έχει τη δυνατότητα να αναγνωρίζει τις εργαλειομηχανές που βρίσκονται σε ένα μηχανουργείο με στόχο να υπερθέτει πάνω στην κάθε μια από αυτές την τρέχουσα θερμοκρασία της, απαιτείται η προσθήκη κάποιων θερμοστοιχείων προκειμένου να εισαχθεί η νέα πληροφορία. Επιπρόσθετα, εάν για κάποια εφαρμογή ήταν απαραίτητο να έρχεται κάποια ειδοποίηση στην περίπτωση που τεμάχιο τοποθετήθηκε σε μια εργαλειομηχανή ακόμα και εάν η μηχανή είναι εκτός του πεδίου ορατότητας της συσκευής, τότε πάλι είναι επιβεβλημένη η χρήση πρόσθετου αισθητήρα.

Είναι εμφανής, λοιπόν, η ανάγκη χρήσης εκτός της κάμερας και πρόσθετου εξοπλισμού για τη βελτίωση της αποτελεσματικότητας μιας εφαρμογής. Τα smartphones και τα tablets έχουν ενσωματωμένο πλήθος αισθητήρων που τα καθιστούν ιδανικές συσκευές για σημαντική ποικιλία εφαρμογών επαυξημένης πραγματικότητας, ωστόσο σε περίπτωση που απαιτείται υψηλότερη ακρίβεια ή κάποια επιπρόσθετη πληροφορία, είναι αναπόφευκτη η χρήση εξωτερικών συσκευών.

Κατά την εκπόνηση της παρούσας εργασίας το ζήτημα αυτό προσεγγίστηκε με χρήση αισθητήρων χαμηλού κόστους. Τα σήματα των αισθητήρων οδηγούνται σε μικροελεγκτή, όπου γίνεται και η πρώτη επεξεργασία τους και κατόπιν οδηγούνται σε ηλεκτρονικό υπολογιστή μέσω θύρας usb ώστε να αξιοποιούνται στο Unity3d για την ανάπτυξη και την εκτέλεση των εφαρμογών που αναπτύσσονται σε αυτό.

Στο παρόν κεφάλαιο αναλύουμε τα επιμέρους στοιχεία που χρησιμοποιούνται και τον τρόπο που αυτά επικοινωνούν μεταξύ τους.

3.2 Πλακέτα Arduino UNO

3.2.1 Λειτουργία και Χαρακτηριστικά

Ο arduino Uno είναι μια πλακέτα ανοικτού κώδικα με μικροελεγκτή και τα απαραίτητα περιφερειακά ηλεκτρονικά. Χρησιμοποιείται για την ανάπτυξη ψηφιακών συσκευών και διαδραστικών αντικειμένων που μπορούν να παίρνουν πληροφορίες και να ελέγχουν αντικείμενα στο φυσικό χώρο.[19]

Εικόνα 3.3.1 – ArduinoUno

Η πλακέτα συνοδεύεται από λογισμικό για τον προγραμματισμό του μικροελεγκτή. Η γλώσσα προγραμματισμού του Arduinoείναι βασισμένη στις γλώσσες C και C++ και παρέχεται η δυνατότητα χρήσης των βιβλιοθηκών της C++ για επέκταση των δυνατοτήτων της.[20]

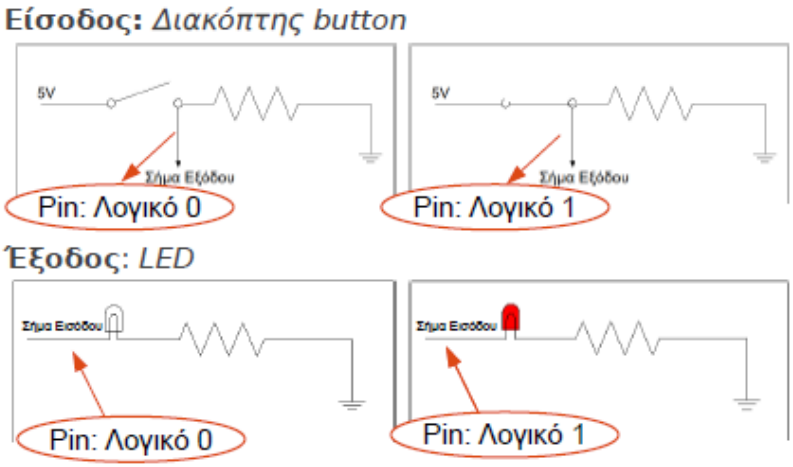
Το πλεονέκτημα χρήσης της εν λόγω πλατφόρμας είναι το χαμηλό κόστος απόκτησης, η δυνατότητα που παρέχει για ταχεία ανάπτυξη πρωτότυπων διατάξεων, η άρτια υποστήριξη με την οποία συνοδεύεται καθώς και πλήθος βιβλιοθηκών για την υποστήριξη μεγάλης ποικιλίας πρόσθετων συσκευών εισόδου – εξόδου (αισθητήρες, κινητήρες, οθόνες και άλλα).

Ακολουθούν τα τεχνικά χαρακτηριστικά της πλακέτας.

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6

Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
Length	68.6 mm
Width	53.4 mm
Weight	25 g

Ο arduino μπορεί να δέχεται ηλεκτρικά σήματα (τάσεις) εντός των αποδεκτών ορίων και να παρέχει τις αντίστοιχες εξόδους που προκύπτουν από το πρόγραμμα που εκτελεί. Οι έξοδοι μπορούν να είναι τάσεις 0 ή 5 voltπου παρέχονται από τους ψηφιακούς I/O ακροδέκτες (pins 0 – 13 για arduino Uno). Οι ακροδέκτες του μικροελεγκτή έχουν χαρακτήρα "on – off". [21], βλ. Εικόνα 3.3.2.



Εικόνα 3.3.2 – Ακροδέκτες με χαρακτήρα on/off

Είναι προφανές από τα προηγηθέντα πως δυνατότητα παραγωγής αναλογικού σήματος εξόδου δεν υπάρχει, αφού δεν μπορεί να παραχθεί σαν έξοδος τάση ενδιάμεση των 0 και 5 volt. Για την προσέγγιση, ωστόσο, τέτοιων τάσεων υποστηρίζεται από κάποιους ακροδέκτες η τεχνική PWM (Pulse Width Modulation).

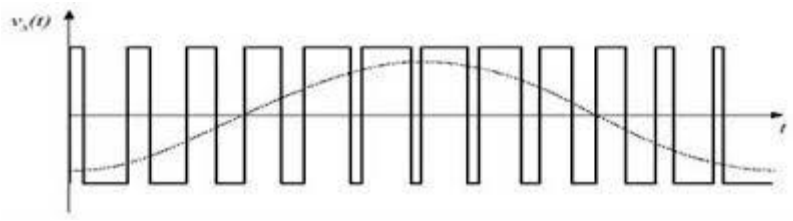
Η τεχνική αυτή βασίζεται στην παραγωγή ορθογωνικών παλμών μεταβλητού εύρους σε υψηλή συχνότητα, με αποτέλεσμα να μεταβάλλεται η μέση συχνότητα της

κυματομορφής. Εάν θεωρήσουμε ένα παλμό $f(t)$ με περίοδο T, ελάχιστη τιμή $y_{\min}$, μέγιστη τιμή $y_{\max}$ με κύκλο λειτουργίας D, τότε η μέση τιμή της κυματομορφής δίνεται από τη σχέση:

$$\bar{y} = \frac{1}{T}\int_0^T f(t)\,dt.$$

Καθώς η $f(t)$ περιγράφει παλμό, η μέγιστη τιμή της θα προκύπτει για 0 <t<D · T και η ελάχιστη για D · T<t< T, έτσι η σχέση που προηγήθηκε δίνεται από την ακόλουθη σχέση, βλ. και Εικόνα 3.3.3:

$$\begin{aligned}\bar{y} &= \frac{1}{T}\left(\int_0^{DT} y_{\max}\,dt + \int_{DT}^{T} y_{\min}\,dt\right)\\ &= \frac{1}{T}\left(D\cdot T\cdot y_{\max} + T\left(1-D\right)y_{\min}\right)\\ &= D\cdot y_{\max} + \left(1-D\right)y_{\min}.\end{aligned} \quad [22]$$

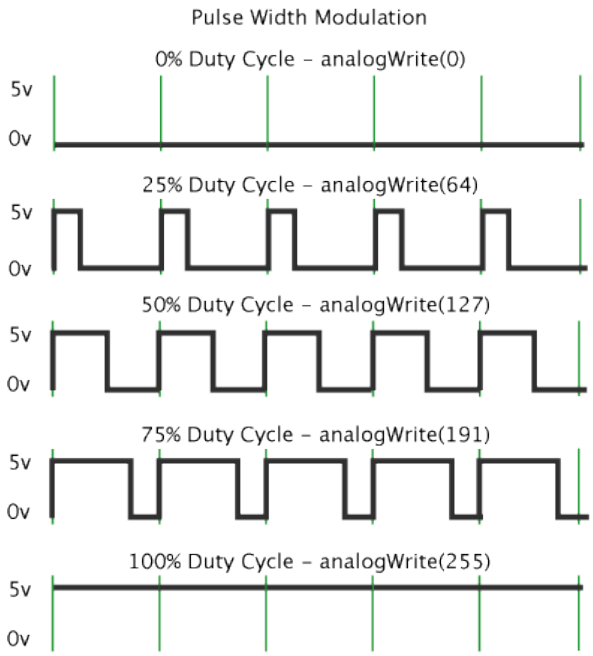


Εικόνα 3.3.3 – PWM Αποτέλεσμα

Στην περίπτωση των ακροδεκτώνPWM του arduino η μέγιστη τάση που παράγεται είναι 5 volt και η ελάχιστη 0 συνεπώς η τελική σχέση που περιγράφει την ισοδύναμη τάση εξόδου θα είναι:

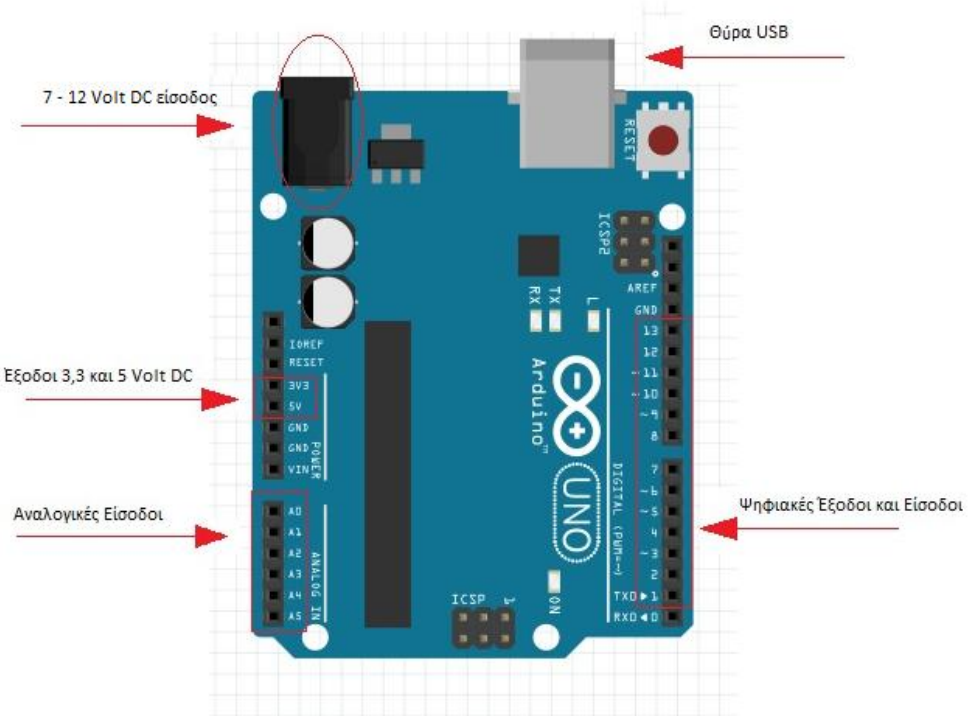
$$V_{PWM} = 5 \cdot D$$

όπου D ο κύκλος λειτουργίας (Duty Cycle), ο οποίος περιγράφει το ποσοστό του χρόνου που ο παλμός έχει την υψηλή τιμή του ανά περίοδο, βλ. Εικόνα 3.3.4.



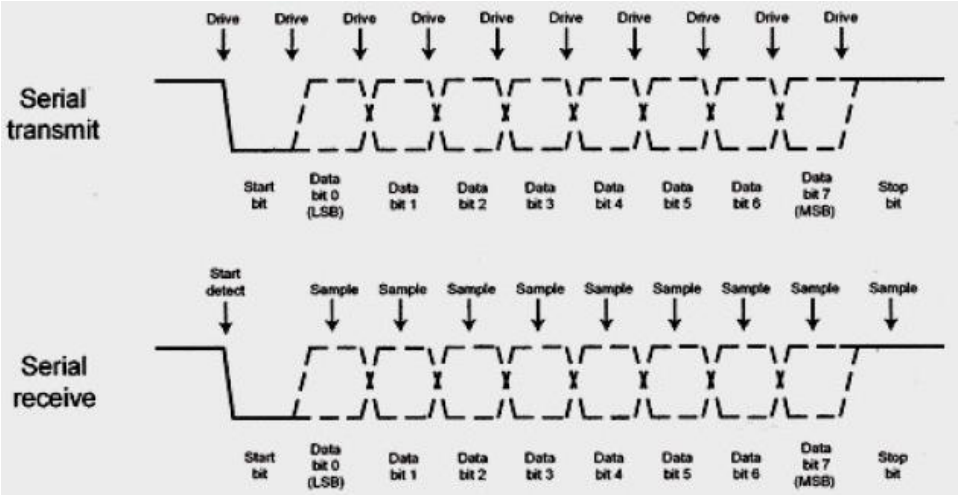
Εικόνα 3.3.4 – PWM Arduino

Οι έξοδοι που περιγράφονται μέχρι στιγμής παρέχονται από τους ακροδέκτες της πλακέτας, όπως φαίνεται και στην Εικόνα 3.5.



Εικόνα 3.5 – Arduino Βασικοί ακροδέκτες και υποδοχές

Εναλλακτικά, μπορεί να στέλνονται bytes σε ηλεκτρονικό υπολογιστή μέσω της θύρας usb. Αυτό συμβαίνει καθώς όλες οι πλακέτες arduino διαθέτουν τουλάχιστον μια σειριακή θύρα η οποία επικοινωνεί εκτός από τη θύρα usb και με τους ψηφιακούς ακροδέκτες 0 (RX) και 1 (TX).[23]Το πρωτόκολλο επικοινωνίας που χρησιμοποιείται για την μεταφορά των δεδομένων από τη μια συσκευή στην άλλη είναι το UART, βλ. Εικόνα 3.6. [21]

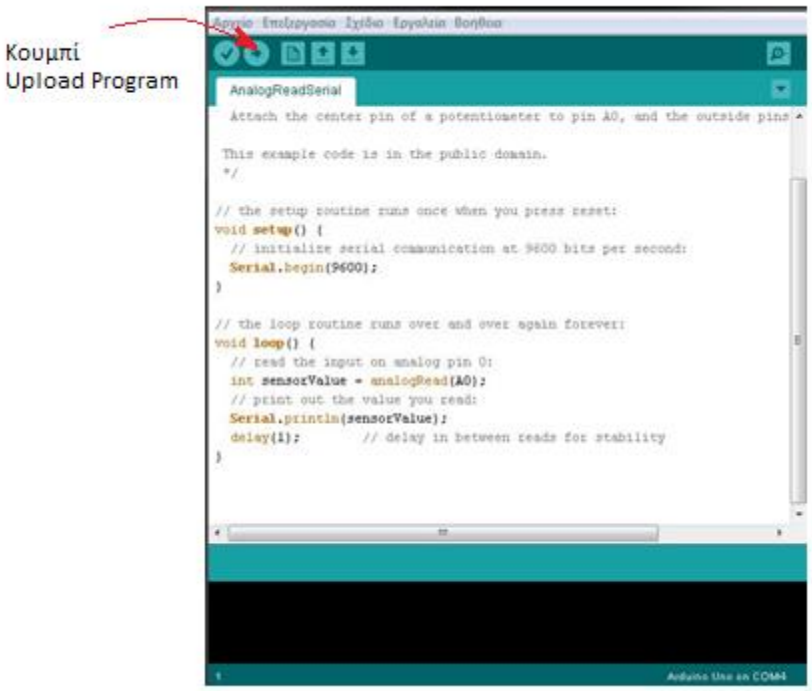


Εικόνα 3.6 – Πρωτόκολλο UART

Εκτός από την αποστολή της ψηφιακής πληροφορίας μέσω usb υπάρχουν και πρόσθετες επιλογές, που όμως απαιτούν την προσθήκη επιπλέον hardware στην πλακέτα όπως wi-fi shields, Ethernet shields και άλλων, τα οποία δεν κρίνεται σκόπιμο να αναλυθούν στην παρούσα εργασία. Η επικοινωνία με τον υπολογιστή φυσικά είναι αμφίδρομη, δηλαδή υπάρχει επιλογή τόσο αποστολής όσο και λήψης bytes μεταξύ των δυο συσκευών.

Οι ψηφιακοί ακροδέκτες μπορούν να χρησιμοποιηθούν επίσης ως είσοδοι, αλλά μπορούν να αντιληφθούν μόνο δύο τιμές τάσης, δηλ. 'υψηλή' και 'χαμηλή'. Άλλη είσοδος είναι οι αναλογικοί ακροδέκτες που φαίνονται κάτω αριστερά στηνΕικόνα 3.5.

Το πρόγραμμα που θα κληθεί να εκτελέσει ο μικροελεγκτής συντάσσεται σε ηλεκτρονικό υπολογιστή, χρησιμοποιώντας το IDE (Integrated Development Environment) του arduino.



Εικόνα 3.7 – Arduino IDE

Στο πρόγραμμα αυτό, εκτός των άλλων, καθορίζονται ποιοί ψηφιακοί ακροδέκτες θα χρησιμοποιηθούν και με ποιόν τρόπο (είσοδοι, απλές έξοδοι ή PWM έξοδοι), ποιες αναλογικές θύρες θα είναι ενεργές και ποιος θα είναι ο ρυθμός αποστολής δεδομένων μέσω usb στον ηλεκτρονικό υπολογιστή. Οι βασικές εντολές που χρησιμοποιούνται παρατίθενται στην επόμενη ενότητα.

Μετά τη σύνταξη του προγράμματος και αφού έχει συνδεθεί ο arduino στον υπολογιστή, πατώντας το κουμπί 'Upload' που φαίνεται στην Εικόνα 3.7εκτελείται η φόρτωση του προγράμματος στη μνήμη του επεξεργαστή. Το πρόγραμμα διατηρείται στην μνήμη του επεξεργαστή ακόμα και αν διακοπεί η παροχή ισχύος. Όταν αυτή επανέλθει το πρόγραμμα θα επανεκκινήσει. Εναλλακτικά, επανεκκίνηση του προγράμματος του μικροεπεξεργαστή επιτυγχάνεται πατώντας το κόκκινο κουμπί που φαίνεται πάνω δεξιά στην Εικόνα 3.5.

Να αναφέρουμε ότι ο arduino όταν είναι συνδεδεμένος σε υπολογιστή η θύρα usb αναλαμβάνει τον πρόσθετο ρόλο της πηγής ισχύος. Διαφορετικά μπορεί να γίνει χρήση της υποδοχής εισόδου συνεχούς ρεύματος (DC jack) για παροχή ισχύος στην πλακέτα από κάποια εξωτερική συσκευή.

3.2.2 Προγραμματισμός μικρό – επεξεργαστή

Η δομή ενός προγράμματος arduino ακολουθεί τη λογική και τη σύνταξη της γλώσσας C, ωστόσο ο χρήστης της συσκευής καλείται να συμπληρώσει τις συναρτήσεις δομής ***void setup( ) { … }***και ***void loop( ) { … }*** . Οι εντολές της void setup() εκτελούνται μια φορά κατά την εκκίνηση του προγράμματος, συνεπώς το τμήμα αυτό του κώδικα χρησιμοποιείται για αρχικοποίηση. Σε αυτό το σημείο είναι που ορίζεται ποιοί ακροδέκτες θα χρησιμοποιηθούν και με ποιον τρόπο, καθώς και που αρχικοποιείται η σειριακή επικοινωνία (ρυθμός αποστολής δεδομένων) και οποιαδήποτε άλλη παράμετρος είναι απαραίτητη για την εκτέλεση του υπόλοιπου κώδικα.

Κατόπιν εκτελούνται οι εντολές της void loop( ) που αποτελούν και το κύριο τμήμα του προγράμματος καθώς θα εκτελούνται μέχρι να σταματήσει η παροχή ρεύματος στην πλακέτα.

Ακολουθούν βασικές συναρτήσεις:

- *pinMode (pinNumber, mode)*: Ορίζει αν ο ψηφιακός ακροδέκτης που αντιστοιχίζεται στον ακέραιο αριθμό του πρώτου ορίσματος, θα χρησιμοποιηθεί ως είσοδος ή έξοδος χρησιμοποιώντας σα δεύτερο όρισμα τις παραμέτρους *INPUT ή OUTPUT*.

- *Serial.begin(baudRate)*: Ορίζει την έναρξη της σειριακής επικοινωνίας (μέσω αυτής αποστέλλονται δεδομένα στον υπολογιστή), με παράμετρο το ρυθμό baud (bits ανά δευτερόλεπτο). Συνιστάται για συγχρονισμό με Η/Υ η χρήση ενός από τους ακόλουθους ρυθμούςbaud:300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600 ή 115200. Συνηθισμένη για σημαντικό πλήθος εφαρμογών είναι η χρήση της τιμής 9600.Σε αυτό το σημείο πρέπει να αναφέρουμε ότι εάν χρησιμοποιηθεί οποιαδήποτε συνάρτηση της βιβλιοθήκης Serial, δεν μπορούν να χρησιμοποιηθούν ως είσοδοι ή έξοδοι οι ακροδέκτες 0 και 1.

  Οι συναρτήσεις pinMode( ) και Serial.begin( ) χρησιμοποιούνται κυρίως εντός της setup( ) για την αρχικοποίηση του προγράμματος.

- *digitalWrite (pinNumber, state)* : Ορίζει αν ο ψηφιακός ακροδέκτης που αντιστοιχίζεται στον ακέραιο αριθμό του πρώτου ορίσματος, θα έχει τιμή 5 volt ή 0 volt χρησιμοποιώντας σα δεύτερο όρισμα τις παραμέτρους HIGH ή LOW. Ισοδύναμα, μπορούν να χρησιμοποιηθούν και σα δεύτερο όρισμα οι αριθμοί 1 και 0.

- *digitalRead (pinNumber):* Επιστρέφει τιμήHIGH ή LOW, που αντιστοιχεί στον ακροδέκτη που υποδεικνύει το όρισμα της συνάρτησης.

- *analogRead (pinNumber)* : Επιστρέφει την τιμή από τον αναλογικό ακροδέκτη (κάτω αριστερά στην εικόνα 4.4) που υποδεικνύει το όρισμα της συνάρτησης, χρησιμοποιώντας έναν αναλογικό σε ψηφιακό μετατροπέα 10 bit. Αυτό σημαίνει ότι τάσεις από 0 έως 5 volt αντιστοιχίζονται σε έναν ακέραιο αριθμό από 0 έως 1023.

- *analogWrite (pinNumber, value)* : Δημιουργεί με την τεχνική PWM μια τάση ως έξοδο στον ψηφιακό ακροδέκτη του πρώτου ορίσματος. Το δεύτερο όρισμα θέτει τον κύκλο λειτουργίας του κύματος PWM και λαμβάνει ακέραιες τιμές από 0 έως 255 που αντιστοιχούν σε κύκλο λειτουργίας από 0% έως 100%.

- *millis( ) / micros( )* : Οι δυο συναρτήσεις επιστρέφουν τον αριθμό των millisecond και microsecond που έχουν παρέλθει από τη έναρξη εκτέλεσης του τρέχοντος προγράμματος. Για την συνάρτηση millis( ) ο αριθμός που επιστρέφει υπερχειλίζει (επιστρέφει στην τιμή 0) μετά από περίπου 50 ημέρες ενώ για την micros( ) μετά από 70 λεπτά.

- *delay (param) / delayMicroseconds (param):* Σταματούν την εκτέλεση του προγράμματος για τον αριθμό millisecondκαι microsecond αντίστοιχα, που δίνεται στο όρισμα της συνάρτησης.

- *Serial.print(var) :*Επιστρέφει ψηφιακά την τιμή της μεταβλητής που χρησιμοποιείται στο όρισμα της συνάρτησης.

- *Serial.available(var) :* Επιστρέφει τον αριθμό των bytes που είναι διαθέσιμα προς ανάγνωση στη σειριακή θύρα. Τα δεδομένα αυτά είναι αποθηκευμένα σε ένα buffer χωρητικότητας 64 bytes.

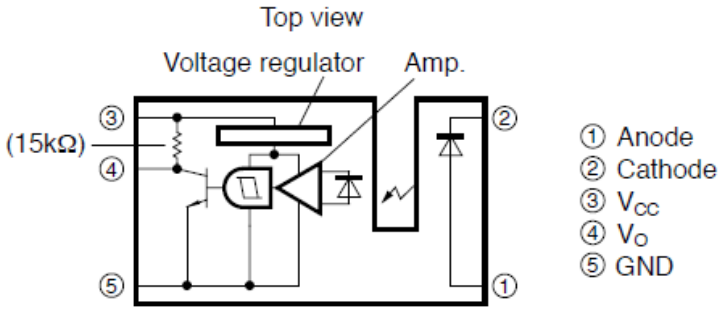
- *Serial.read( ) :* Διαβάζει εισερχόμενα δεδομένα μέσω σειριακής επικοινωνίας.[24]

Υπάρχει μεγάλο πλήθος συναρτήσεων διαθέσιμων προς χρήση, ωστόσο κρίνεται ότι οι προηγηθείσες αρκούν για την πρώτη προσέγγιση του προγραμματισμού του μικρο-επεξεργαστή.

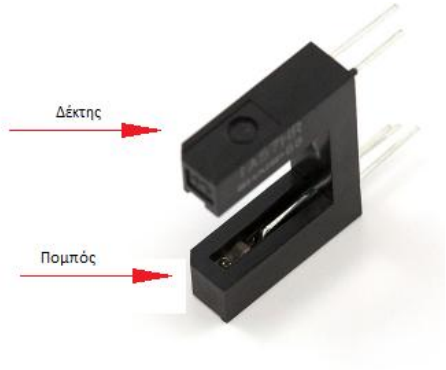
3.3 Δοκιμαστική εφαρμογή με επικοινωνία Arduino – Unity3d

Για την εκπόνηση της εργασίας έγιναν δοκιμές προκειμένου να επιτευχθεί συνεργασία εφαρμογής αναπτυγμένης στο Unity3d με διάφορους αισθητήρες με χρήση πλακέτας arduino uno. Σε αυτή την παράγραφο θα περιγράψουμε την διαδικασία αυτή σε μια απλή εφαρμογή που αναπτύχθηκε για δοκιμή.

Στόχος της εφαρμογής είναι η εμφάνιση μηνύματος σε πραγματικό χρόνο όταν αντικείμενο βρεθεί σε μια προκαθορισμένη θέση στο χώρο. Για τον σκοπό αυτό χρησιμοποιείται ένας φωτοδιακόπτης GP1A57HR της εταιρίας SHARP.

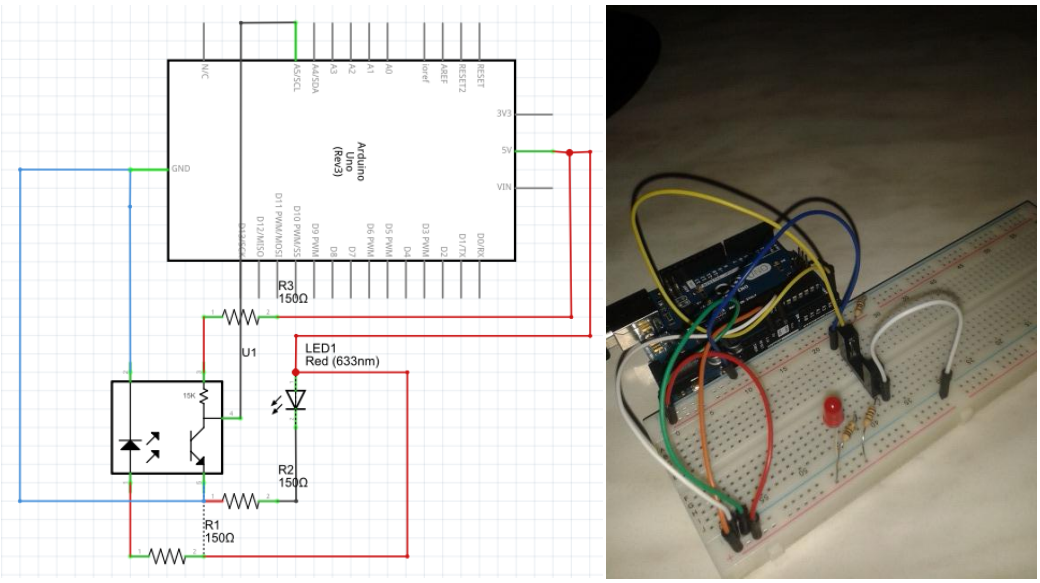


Εικόνα 3.8 – Φωτοδιακόπτης GP1A57HR κυκλωματικό διάγραμμα



Εικόνα 3.9 – Φωτοδιακόπτης

Ο αισθητήρας αυτός απαρτίζεται από έναν πομπό και έναν δέκτη υπέρυθρης ακτινοβολίας τοποθετημένους αντιδιαμετρικά όπως φαίνεται στην Εικόνα 3.9. [25] Όταν παρεμβληθεί εμπόδιο μεταξύ πομπού και δέκτη, αυτό διακόπτει την υπέρυθρη δέσμη και πέφτει η τάση στο συλλέκτη ενός διπολικού npn τρανζίστορ (ακροδέκτης 4) που διεγείρεται από τον δέκτη υπέρυθρης ακτινοβολίας μέσω ενισχυτή όπως φαίνεται στην Εικόνα 3.8. Ακολουθεί στην Εικόνα 3.10 σχηματικό διάγραμμα που δείχνει τον τρόπο σύνδεσης του κυκλώματος (ο σχεδιασμός έγινε στο λογισμικό fritzing).



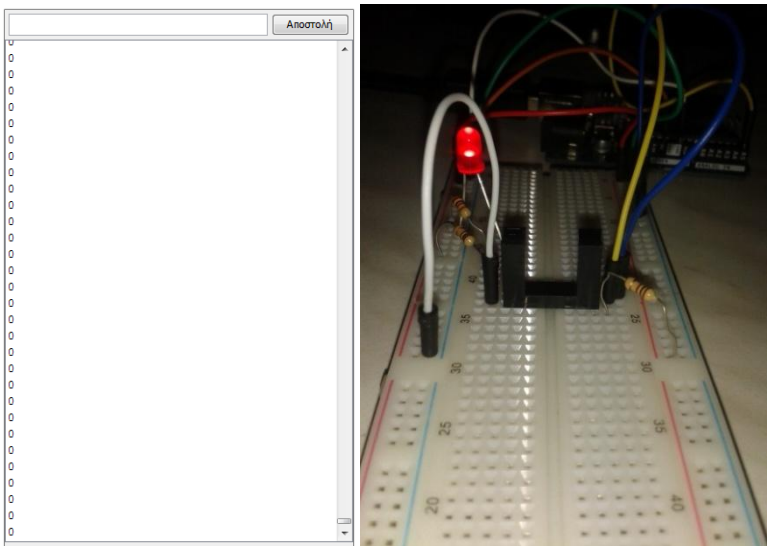
Εικόνα 3.10 – Σχηματικό διάγραμμα κυκλώματος και πραγματικό κύκλωμα

Η πρόσθετη φωτοδίοδος έχει τοποθετηθεί ως ένδειξη για το πότε ο αισθητήρας τροφοδοτείται με ισχύ και δεν έχει περαιτέρω λειτουργικό ρόλο. Το επόμενο στάδιο είναι η σύνταξη προγράμματος στο οποίο θα δηλώνεται ο χρησιμοποιούμενος αναλογικός ακροδέκτης που λαμβάνει το σήμα από το φωτοδιακόπτη και θα καθορίζεται η σειριακή επικοινωνία. Ακολουθεί ο σχετικός κώδικας:

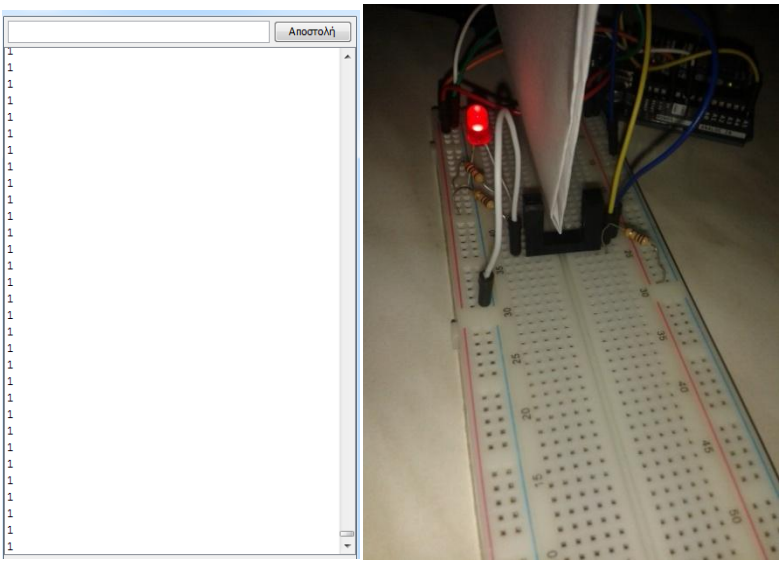
```
intanalogInputPin;
intval;
booleanobstaclePresent;

voidsetup()
{ // Αρχικοποίηση προγράμματος
        analogInputPin = 5;
        obstaclePresent = false;
        Serial.begin(9600); // baud rate 9600
}

void loop()
{
        val=analogRead(analogInputPin);
        if(val>500)
          {
```

```
    obstaclePresent=false;
   }
  else
   {
    obstaclePresent=true;
  }
  Serial.println(obstaclePresent);
  delay(100);
}
```

Εκτελώντας το πρόγραμμα και ανοίγοντας τη σειριακή οθόνη που παρέχει το IDE του arduino, επιβεβαιώνουμε τη σωστή λειτουργία του αισθητήρα, βλ. Εικόνα 3.11 και Εικόνα 3.12.



Εικόνα 3.11 – Περίπτωση 1η Αισθητήρας δεν εντοπίζει εμπόδιο / αποτελέσματα



Εικόνα 3.12 – Περίπτωση 2η Αισθητήρας εντοπίζει εμπόδιο / αποτελέσματα

Απομένει η ανάπτυξη εφαρμογής στο Unity3d που επικοινωνεί μέσω της σειριακής θύρας με τον arduino λαμβάνει την πληροφορία, την επεξεργάζεται και κατόπιν εμφανίζει κατάλληλο μήνυμα στην οθόνη.

Αρχικά από τη γραμμή εργαλείων επιλέγουμε File ->Build Settings ->Player Settings και στο πεδίο Api Compatibility Level αλλάζουμε την προεπιλεγμένη επιλογή σε .NET 2.0 (βλ. Εικόνα 3.13). Αυτό γίνεται καθώς, προκειμένου να δημιουργήσουμε αντικείμενο της κλάσης SerialPort, πρέπει να χρησιμοποιήσουμε το namespace System.IO.Ports, το οποίο δεν περιλαμβάνεται στην επιλογή .NET 2.0 Subset.

Εικόνα 3.13 – BuildSettings

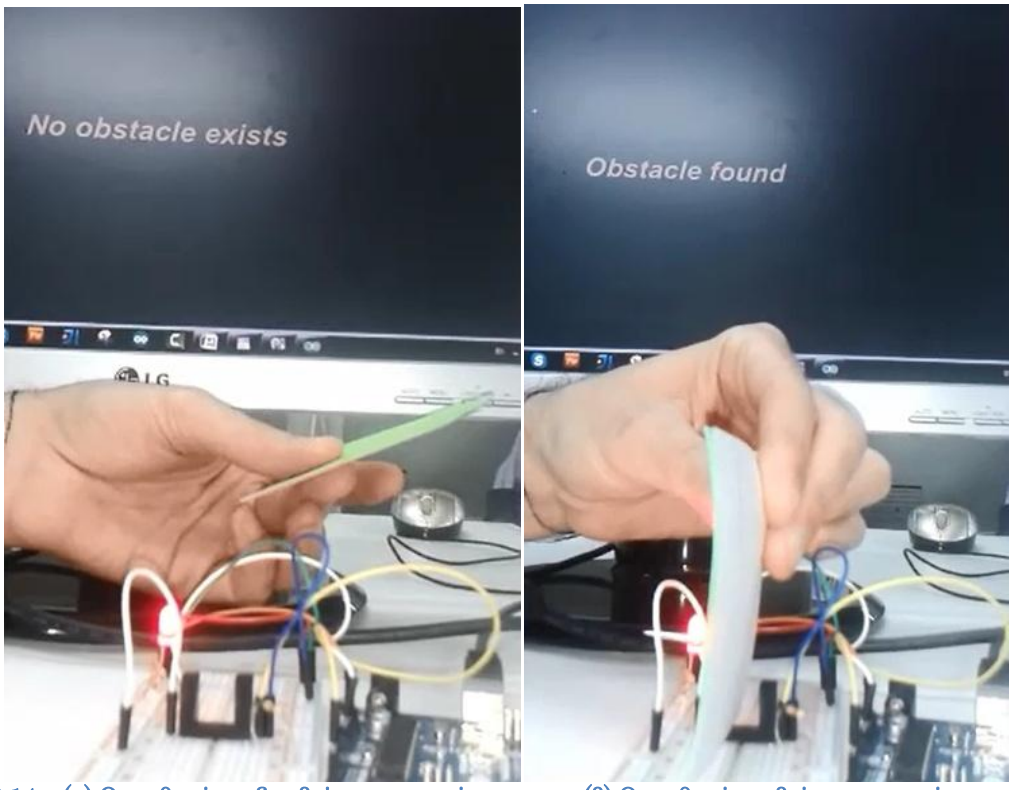
Κατόπιν δημιουργούμε ένα component script ως εξής:

```
usingUnityEngine;
usingSystem.Collections;
using System;

usingSystem.IO.Ports;

namespace Code
{
  public class SerialPortListener : MonoBehaviour
  { // Δημιουργία αντικειμένου της κλάσης SerialPort
    // οι παράμετροι πρέπει να είναι ίδιες με τις αντίστοιχες
    // που φαίνονται στο IDE του arduino
    SerialPort serialPort = new SerialPort("COM3",9600);
    string input;
    public GUISkin myGui;
    private string text;

    // Use this for initialization
    void Start ()
```

```
{
  serialPort.Open ();
  serialPort.ReadTimeout = 1;
}
voidOnGUI ()
{
  GUI.skin = myGui;
  if (serialPort.IsOpen)
  {
  // Χρησιμοποιούμε try catch προκειμένου να αποφύγουμε σφάλματα
  //στην περίπτωση που η συγκεκριμένη σειριακή θύρα δεν
  // συμπεριφέρεται με τον αναμενόμενο τρόπο
    try
    {
      // Λήψη δεδομένων
      //Να σημειωθεί ότι τα δεδομένα που λαμβάνουμε από
                                          //τη σειριακή θύρα είναι τύπου string
      input =serialPort.ReadLine();
      Debug.Log(input);
    // Επεξεργασία δεδομένων
      if(String.Equals(input,"1",StringComparison.Ordinal))
      {
        text = "Obstacle found";
      }
              else   if(String.Equals(input,"0",StringComparison.Ordinal))
      {
        text = "No obstacle exists";
      }
    }
    catch(System.Exception)
    {

    }
  }
  // Εμφάνιση ανάλογου μηνύματος
  GUI.Box (new Rect ((int)(Screen.width / 2) - 300, (int)(Screen.height / 2), 600, 200), text);
 }
 }
}
```

Να σημειωθεί ότι το κυρίως μέρος της παραπάνω κλάσης βρίσκεται εντός της μεθόδου OnGUI( ) η οποία, όπως έχουμε δει σε προηγούμενο κεφάλαιο, εκτελείται πολλές φορές ανά frame. Ωστόσο, εάν στον arduino δεν επιβληθεί μια καθυστέρηση στο ρυθμό που θα δίνει την νέα ένδειξη του αισθητήρα (μεγαλύτερη των 50 millisecond ανά ένδειξη), η πληροφορία δεν μπορεί να αξιοποιηθεί από την εφαρμογή. Αντίστοιχος περιορισμός υπάρχει και για την μέθοδο Update( ) η οποία εκτελείται μια φορά κάθε frame.

Το component Script μπορεί να προσαρτηθεί στην κάμερα, αλλά στην περίπτωσή μας για μεγαλύτερη ευκρίνεια δημιουργούμε ένα κενό game object το οποίο ονομάζουμε Arduino Listener στο οποίο και προσαρτούμε και το αντίστοιχο component για να αποκτήσει την συμπεριφορά που επιθυμούμε. Σημειώνεται ότι κατά τη διάρκεια εκτέλεσης της εφαρμογής δεν υπάρχει δυνατότητα άλλο πρόγραμμα στον υπολογιστή να χρησιμοποιήσει τη θύρα COM 3. Ακολουθούν τα αποτελέσματα στην Εικόνα 3.14

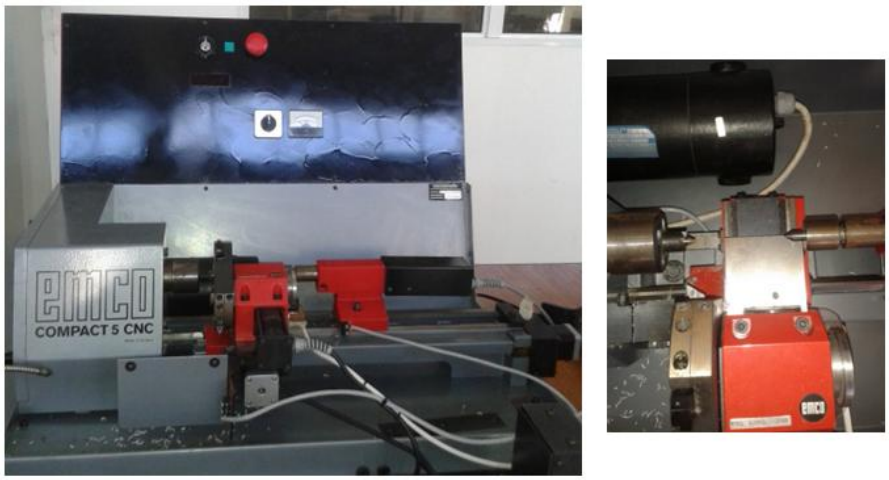


Εικόνα 3.14 – (α) Ο αισθητήρας δεν βρίσκει αντικείμενο (β) Ο αισθητήρας βρίσκει αντικείμενο

4. Παρουσίαση Πρώτης Εφαρμογής

4.1 Σκοπός και προσέγγιση

Ο σκοπός της πρώτης εφαρμογής είναι να αποτελέσει έναν εύχρηστο εικονικό οδηγό για την κατάδειξη του τρόπου στήριξης κυλινδρικού τεμαχίου στον CNC τόρνο EMCO, που βρίσκεται στο κτήριο Ξ της σχολής μηχανολόγων μηχανικών του Ε.Μ.Π (βλ. Εικόνα 4.4.1). Η εν λόγω εργαλειομηχανή προγραμματίζεται και εκτελεί τις λειτουργίες της με χρήση λογισμικού linux CNC, που είναι εγκατεστημένο σε ηλεκτρονικό υπολογιστή που τη συνοδεύει.



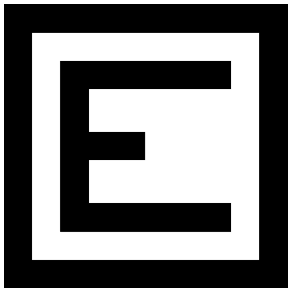
Εικόνα 4.4.1 – EMCO CNC τόρνος

Οι βασικοί στόχοι της εφαρμογής είναι:

- Οι οδηγίες που παρέχει στο χρήστη να είναι εύληπτες και να μην αφήνουν περιθώρια παρερμηνειών.
- Η εύκολη επέκταση και τροποποίηση του λειτουργικού χαρακτήρα της.
- Η αξιολόγηση των εργαλείων (πακέτα λογισμικού, κάμερα, αισθητήρες, συσκευή απεικόνισης) που χρησιμοποιούνται για την ανάπτυξη και την λειτουργία της.
- Η ανάδειξη προβλημάτων που προκύπτουν κατά την ανάπτυξή της, προκειμένου να ακολουθηθούν διαφορετικές πρακτικές στην ανάπτυξη επόμενης εφαρμογής.

Η ανάπτυξη της εφαρμογής και η εκτέλεσή της πραγματοποιείται σε ηλεκτρονικό υπολογιστή, ενώ η λήψη της εικόνας του περιβάλλοντος χώρου γίνεται από συμβατική web κάμερα (*Logitech HD webcam C525*). Παράλληλα χρησιμοποιείται πλακέτα Arduino Uno για εισαγωγή σήματος από εξωτερικό αισθητήρα στην εφαρμογή.

Για την σωστή λειτουργία της εφαρμογής έχει εισαχθεί προκαθορισμένης μορφής γεωμετρία (βλ. Εικόνα 4.2) ως στόχος προς αναγνώριση και παρακολούθηση. Ο ρόλος του στόχου είναι να αποτελέσει σημείο αναφοράς στον προγραμματικό χώρο, τόσο για το σημείο τοποθέτησής του, όσο και για τις διαστάσεις του, καθώς αυτά είναι τα στοιχεία που θα χρησιμοποιηθούν για την χωροθέτηση και τη διαστασιολόγηση των εικονικών μοντέλων.

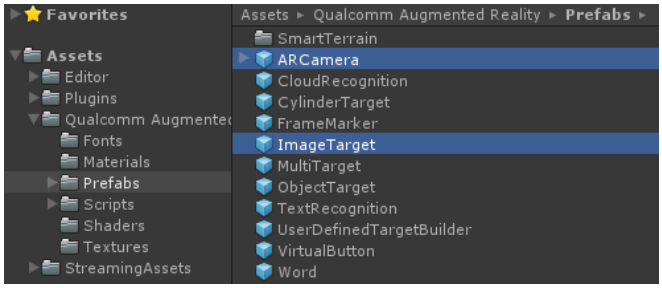


Εικόνα 4.2 – Στόχος προς αναγνώριση

4.2 Αναγνώριση στόχου

Η αναγνώριση και η παρακολούθηση του στόχου, όπως έχει ήδη αναφερθεί, πραγματοποιείται με χρήση του **Vuforia SDK** για το Unity. Το πρώτο στάδιο είναι η εισαγωγή, από τη γραμμή εργαλείων του Unity 3d (*Assets -> Import Package -> Custom Package*), του πακέτου:
vuforia-unity-mobile-android-ios-4-0-103.unitypackage, το οποίο ελήφθη από τον ιστότοπο https://developer.vuforia.com/downloads/sdk

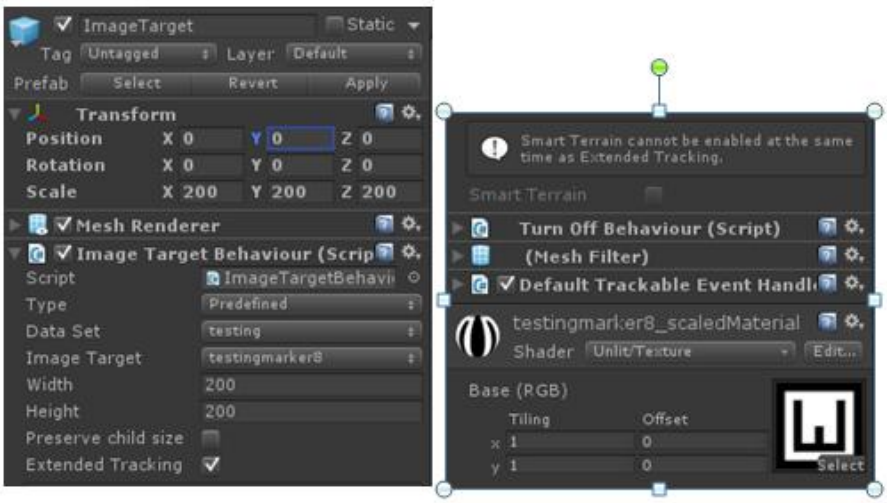
Μετά, από αυτή την επιλογή μπορούμε να παρατηρήρουμε στο Project Window (καρτέλα κάτω αριστερά), τους νέους φακέλους και τα στοιχεία που έχουν προστεθεί στο φάκελο Assets (βλ. Εικόνα 4.3)



Εικόνα 4.3 – Vuforia package

Στην σκηνή τοποθετούνται τα prefabs που παρέχει το Vuforia '*ARCamera'* και '*ImageTarget'*, και αφαιρείται το game object '*MainCamera'* που υπάρχει από προεπιλογή.

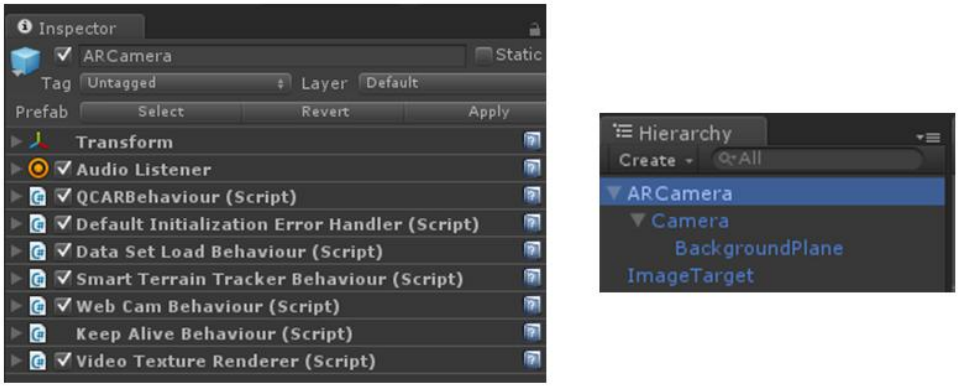
- *ImageTarget* : Τα components του αντικειμένου παρουσιάζονται στην Εικόνα 4.4. Προκειμένου να αποκτήσει λειτουργική συμπεριφορά πρέπει να παραμετροποιηθεί το Component 'ImageTargetBehaviour'.
  Αρχικά δημιουργήθηκε λογαριασμός στο: https://developer.vuforia.com/targetmanager/licenseManager/licenseListing, από όπου και ελήφθησαν τα στοιχεία Data Set και License Key, απαραίτητα για ανάπτυξη εφαρμογής AR με χρήση του Vuforia SDK. Το Data Set λοιπόν που ελήφθη από τον παραπάνω ιστότοπο αντιστοιχίστηκε στο ομώνυμο πεδίο του Component 'ImageTargetBehaviour' ούτως ώστε η γεωμετρία αυτή να αναζητείται σε κάθε frame που παρέχει η web camera κατά την εκτέλεση της εφαρμογής.



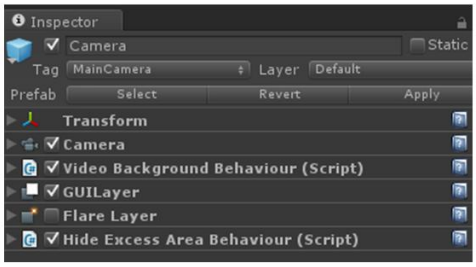
Εικόνα 4.4 – ImageTarget Components

Τα game objects που υπερτίθενται στην εικόνα του φυσικού χώρου ορίζονται ως παιδιά του game object 'ImageTarget' προκειμένου η χωροθέτηση, η κάτεύθυνση κίνησης και η διαστασιολόγησή τους να εκτελούνται σε σχέση με τα αντίστοιχα στοιχεία (πεδία του Transform Component) του 'ImageTarget'.

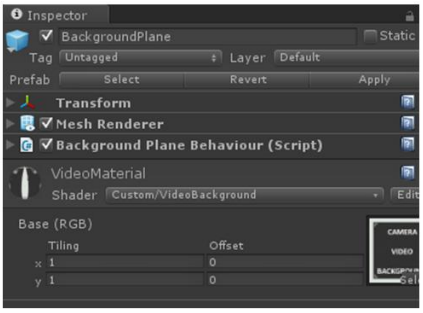
- *ARCamera* : Είναι γονέας του αντικειμένου 'Camera', το οποίο έχει παιδί το αντικείμενο 'BackgroundPlane'. Η ιεραρχική δομή και τα components που δημιουργούν τη συμπεριφορά του εν λόγω αντικειμένου φαίνονται στην Εικόνα 4.5, ενώ τα components των αντικειμένων παιδιών φαίνονται στην Εικόνα 4.6(α),(β).



Εικόνα 4.5 – ARCamera prefab



Εικόνα 4.6 (α) – Camera prefab Components



(β) – BackgroundPlane prefab Components

Το αντικείμενο 'BackgroundPlane' λειτουργεί ως επιφάνεια προβολής της εικόνας που λαμβάνεται από τη web camera. Το αντικείμενο 'Camera', έχει όλα τα components που συνοδεύουν το game object 'Camera' που παρέχει το Unity, συνεπώς αυτό είναι τελικά υπεύθυνο για την εικόνα που λαμβάνει ο χρήστης κατά την εκτέλεση της εφαρμογής. Είναι τοποθετημένο (συντεταγμένες του Transform Component) έτσι ώστε το αντικείμενο 'BackgroundPlane' να βρίσκεται εντός του πεδίου ορατότητάς του. Στα δυο αυτά αντικείμενα οι ρυθμίσεις των components έχουν αφεθεί στις προεπιλεγμένες.

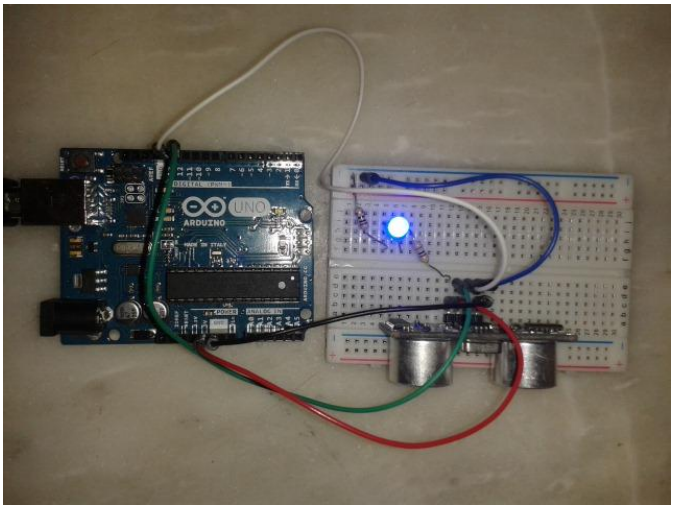
Στο γονέα των δυο αντικειμένων, δηλαδή στο αντικείμενο ‘ARCamera’, αντιστοιχίζεται το License Key στο ομώνυμο πεδίο του component QCARBaheviour. Ρόλος του αντικειμένου αυτού είναι να εκτελεί αναγνώριση και παρακολούθηση της εικόνας στόχου και να μεταβάλλει τις συντεταγμένες του κατά τη διάρκεια της εκτέλεσης της εφαρμογής, έτσι ώστε η σχετική θέση της web camera και της πραγματικής εικόνας στόχου στο φυσικό χώρο, να μεταφέρεται κατά ανάλογο τρόπο στη σκηνή μεταξύ του αντικειμένου ‘ImageTarget’ και ‘Camera’. Αυτό μπορεί να επιτευχθεί λόγω της ιεραρχικής σχέσης των αντικειμένων ‘ARCamera’, ‘Camera’ και ‘BackgroundPlane’ που επιβάλλει τη μεταφορά της μεταβολής των παραμέτρων του Transform component του αντικειμένου ‘ARCamera’ στα δυο αντικείμενα παιδιά, ενώ ταυτόχρονα διατηρεί τις σχετικές αποστάσεις των τριών αντικειμένων αμετάβλητες.

Εκτενής βιβλιογραφία για τις κλάσεις που παρέχει το Vuforia SDK μπορεί να ευρεθεί στον ιστότοπο:

https://developer.vuforia.com/library/resources/api/unity/annotated

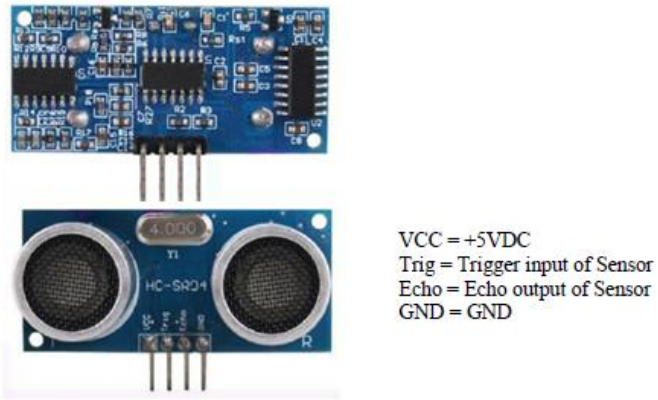
4.3 Προσθήκη αισθητήρα μέτρησης μήκους

Για την κατάδειξη της στήριξης του κυλινδρικού δοκιμίου πρέπει να εισαχθεί στην εφαρμογή σαν πληροφορία το ύψος του. Έχει δημιουργηθεί αντίστοιχο πεδίο, που εμφανίζεται κατά την εκτέλεση της εφαρμογής, στο οποίο ο χρήστης πληκτρολογεί την πληροφορία, ή εναλλακτικά μπορεί να χρησιμοποιήσει διάταξη που φέρει αισθητήρα μέτρησης μήκους και επικοινωνεί με την εφαρμογή μέσω πλακέτας arduino Uno και θύρας usb (βλ. Εικόνα 4.7).

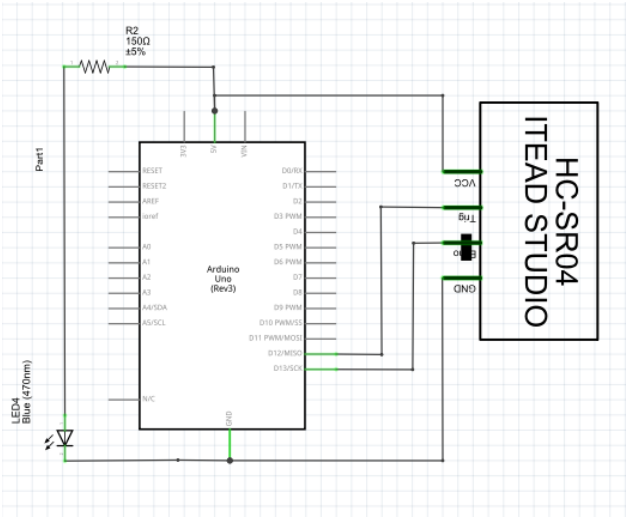


Εικόνα 4.7 – Διάταξη μέτρησης μήκους

Σε πρώτο στάδιο έχει χρησιμοποιηθεί αισθητήρας υπερήχων HC – SR04 (βλ. Εικόνα 4.8). Λειτουργεί ως ραντάρ, καθώς διαθέτει έναν πομπό και ένα δέκτη υπερήχων και υπολογίζει την απόσταση που παρεμβάλλεται μεταξύ πομπού και εμποδίου, ενώ το δραστικό βεληνεκές είναι από 2 – 400 εκατοστά με ακρίβεια 0,3 εκατοστά. Η διαμόρφωση του σήματος εξόδου πραγματοποιείται στο κύκλωμα ελέγχου, αφού πρώτα συνδεθούν κατάλληλα οι τέσσερις ακροδέκτες Vcc, GND, Trig και Echo. Οι πρώτοι δύο συνδέονται με το αντίστοιχο τμήμα τροφοδοσίας DC που διαθέτει η πλακέτα arduino (στις εξόδους 5V και GND αντιστοίχως) και παρέχουν την απαιτούμενη ηλεκτρική ισχύ στον αισθητήρα. Οι ακροδέκτες Trig και Echo συνδέονται στους ψηφιακούς ακροδέκτες 12 και 13 της πλακέτας arduino (το σχηματικό διάγραμμα του κυκλώματος φαίνεται στην Εικόνα 4.9, ενώ έχει τοποθετηθεί παράλληλα με τους ακροδέκτες Vcc και GND φωτοεκπομπός δίοδος ως ένδειξη για το πότε ο αισθητήρας είναι ενεργός).



Εικόνα 4.8 – Αισθητήρας HC-SR04

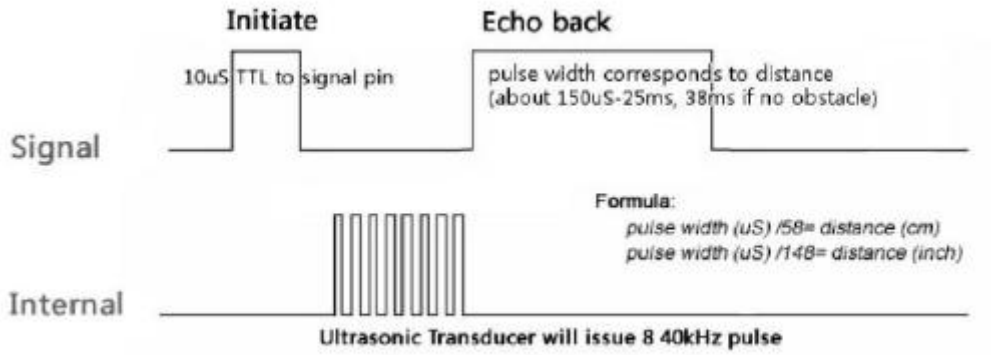


Εικόνα 4.9 – Σχηματικό διάγραμμα διάταξης

Για την έναρξη της μέτρησης ο ακροδέκτης Trig τροφοδοτείται με σήμα 5V για τουλάχιστον 10μs. Αυτό ενεργοποιεί την διαδικασία αποστολής 8 υπερηχητικών παλμών με συχνότητα 40kHz από τον υπερηχητικό επενεργητή και την αναμονή του ανακλώμενου κύματος. Όταν ο δέκτης λάβει τον υπέρηχο που επιστρέφει, ο ακροδέκτης Echo αποστέλλει παλμό 5V με πλάτος ανάλογο της απόστασης μεταξύ εμποδίου και αισθητήρα (βλ. Εικόνα 4.10) [21]. Ο υπολογισμός της απόστασης γίνεται με την ακόλουθη σχέση:

$$d\ (cm) = \frac{Time}{58},$$

όπου d η απόσταση και Time το πλάτος του παλμού του ακροδέκτη Echo σε μs .



Εικόνα 4.10 – Σήματα ελέγχου HC-SR04

Η διαδικασία που περιγράφηκε στις προηγούμενες παραγράφους προγραμματίζεται σε γλώσσα arduino, προκειμένου το σήμα του αισθητήρα να οδηγηθεί στην σειριακή θύρα του ηλεκτρονικού υπολογιστή και να αξιοποιηθεί κατάλληλα από την εφαρμογή. Ο κώδικας που έχει φορτωθεί στον μικρο – ελεγκτή του arduino ακολουθεί:

```
#include <NewPing.h>     // Φόρτωση βιβλιοθήκης αισθητήρα

#define TRIGGER_PIN 12   // Αντιστοίχιση arduino pin και trigger pin στον
                         // αισθητήρα
#define ECHO_PIN 13      // Αντιστοίχιση arduino pin και echo pin στον
                         // αισθητήρα

#define MAX_DISTANCE 400 // Ορισμός μέγιστης απόστασης (cm)
```

```
NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE); // Αρχικοποίηση μεταβλητής
                                                    // sonar

void setup()
{
  Serial.begin(9600);    // Αρχικοποίηση σειριακής επικοινωνίας
}

void loop()
{
  delay(200);     //Καθυστέρηση για την αποστολή επόμενου υπερήχου
  unsigned int uS = sonar.ping();    // Χρόνος μεταξύ αποστολής - λήψης υπερήχου
 Serial.println(uS / US_ROUNDTRIP_CM); // Αποστολή απόστασης (cm) στη σειριακή
                                       // θύρα
}
```

Κατόπιν δημιουργήθηκε το component 'ArduinoSensor' στο unity, υπεύθυνο για να λαμβάνει την, ψηφιακή πλέον, πληροφορία. Το προαναφερθέν component που είναι προσαρτημένο στο αντικείμενο 'BackgroundPlane', ορίζει την σειριακή θύρα (COM3) που παρακολουθείται από την εφαρμογή και το συγχρονισμό του ρυθμού baud με τον αντίστοιχο που ορίζεται στον κώδικα του arduino (9600). Κατόπιν, με χρήση της εντολής ReadLine( ), λαμβάνει την πληροφορία που αποστέλλει η πλακέτα arduino σε μορφή αλφαριθμητικού (string), το οποίο και μετατρέπει σε ακέραιο. Έτσι, παρέχει τη δημόσια προσβάσιμη μεταβλητή *int height* που αντιστοιχεί στο ύψος του κυλινδρικού τεμαχίου σε εκατοστά.

Η λογική που ακολουθεί η δόμηση του component 'ArduinoSensor' είναι παρεμφερής με εκείνη που παρουσιάστηκε στο τέλος του Κεφαλαίου 3.3 για το component SerialPortListener, γι' αυτό το λόγο δεν κρίνεται σκόπιμο να παρουσιαστεί σε αυτό το σημείο κώδικας. Για όλες τις κλάσεις της εφαρμογής ο κώδικας παρατίθεται στο Παράρτημα Α.

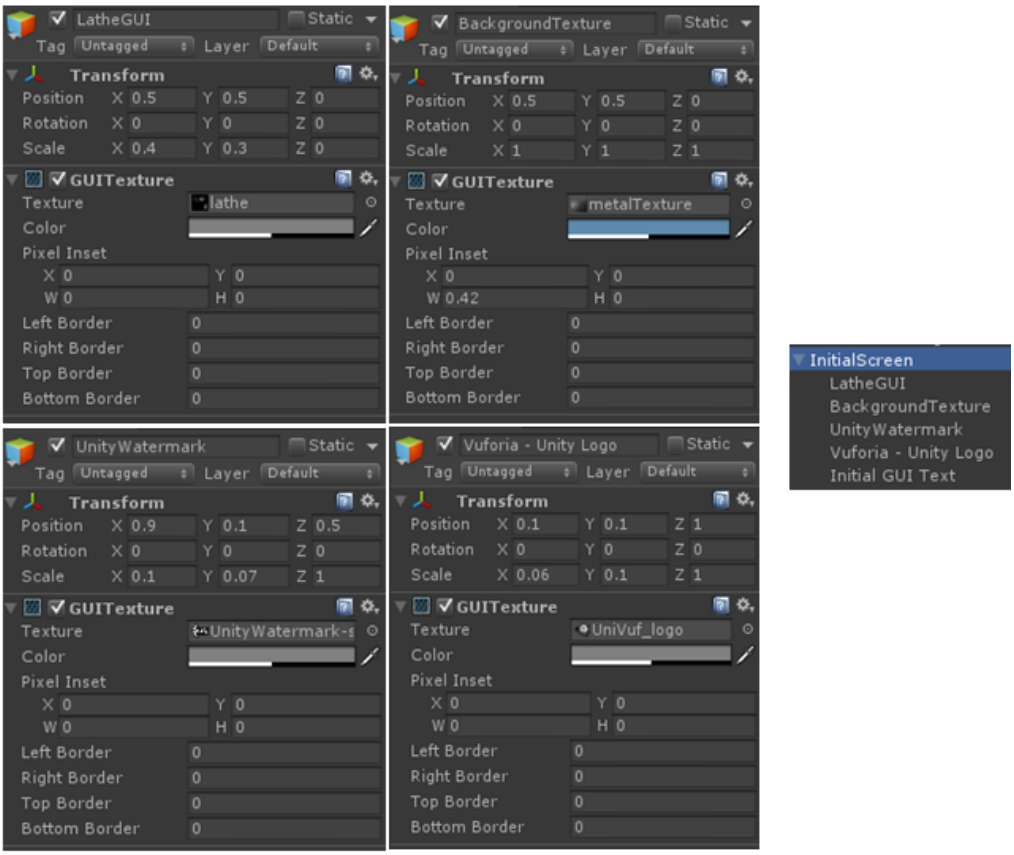
4.4 Εικονικά Μοντέλα – Game Objects

Σε αυτή την ενότητα παρουσιάζονται τα game objects που συμμετέχουν στην εκτέλεση της εφαρμογής σε συνδυασμό με τις ιεραρχικές σχέσεις που τα συνδέουν και τα components που ορίζουν τη συμπεριφορά και τις δυνατότητές τους (εξαιρούνται τα αντικείμενα 'ARCamera' και 'ImageTarget' καθώς έχουν ήδη παρουσιαστεί στο κεφάλαιο 4.2).

Πρώτα πρέπει να αναφερθεί ότι για αυτή την εφαρμογή όλα τα game objects που συμμετέχουν στην εκτέλεσή της είναι εξ' αρχής τοποθετημένα στη σκηνή. Η συμπεριφορά ορισμένων εξ' αυτών μεταβάλλεται δυναμικά ενεργοποιώντας/ απενεργοποιώντας components προσαρτημένα σε αυτά. Ακολουθεί η περιγραφή τους.

- *Lighting*: Είναι ένα κενό αντικείμενο γονέας δυο αντικειμένων *Directional Light* (game object που παρέχει το Unity). Αυτά είναι και τα στοιχεία που επιτελούν τη λειτουργία του φωτισμού της σκηνής, αφού πρώτα έχουν παραμετροποιηθεί κατάλληλα ο προσανατολισμός τους (πεδία Rotation του Transform Component) η φωτεινότητα και το χρώμα του φωτός που εκπέμπουν (πεδία intensity και Color του Light Component). Να σημειωθεί ότι το αποτέλεσμα που επιφέρουν είναι ανεξάρτητο του σημείου στο οποίο τοποθετούνται στη σκηνή.

- InitialScreen : Περιλαμβάνει την ετικέτα 'guitexture' και είναι γονέας των αντικειμένων 'LatheGUI', 'BackgroundTexture', 'UnityWatermark', 'Vuforia-Unity logo' και 'InitialGUIText'. Το καθένα από τα πρώτα τέσσερα έχει προσαρτημένο ένα component 'GUITexture' με το πεδίο Texture να αντιστοιχίζεται σε κάποια εικόνα (βλ. Εικόνα 4.11). Το αντικείμενο 'InitialGUIText' έχει προσαρτημένο το component 'GUIText' και διαθέτει την ετικέτα 'guiText'. Ο συνδυασμός των πέντε αντικειμένων παιδιών έχει σαν αποτέλεσμα τη δημιουργία της αρχικής εικόνας που βλέπει ο χρήστης μόλις η εφαρμογή εκκινεί (βλ. Εικόνα 4.12).



Εικόνα 4.11 – InitialScreen / Αντικείμενα παιδιά και Components

Εικόνα 4.12 – Αρχική εικόνα εφαρμογής

Το game object 'InitialScreen' ωστόσο, έχει προσαρτημένο το script component 'GUIManager', το οποίο είναι υπεύθυνο για την συμπεριφορά του αντικειμένου. Το component 'GUIManager' παρέχει την δημόσια μεταβλητή *bool targetRecognized* στην οποία αποκτά πρόσβαση το component script 'DefaultTrackableEventHandler' με την εντολή:

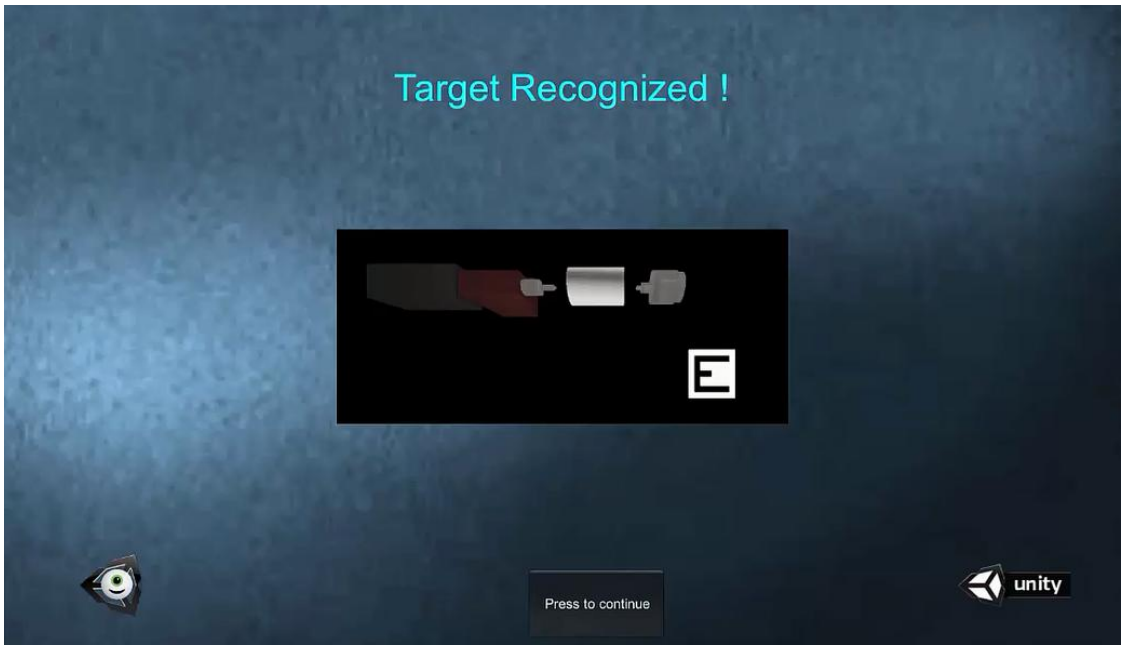
```
GameObject.FindWithTag("guitexture").GetComponent<GUIManager>().targetRecognized =… ;
```

Η εντολή αυτή έχει προστεθεί προκειμένου η μεταβλητή 'targetRecognized' να αποκτά την τιμή 'true' όταν έχει αναγνωριστεί η εικόνα στόχος.

Όσο η μεταβλητή έχει την τιμή 'false' εκτελείται η Coroutine 'waitText( )' (ιδιωτική μέθοδος του GUIManager) η οποία αποκτά πρόσβαση στο πεδίο Text του αντικειμένου 'InitialGUIText' και μεταβάλλει το αναγραφόμενο μήνυμα (βλ. Εικόνα 4.13). Οι coroutines είναι ειδικές συναρτήσεις του Unity, που επιτρέπουν να σταματά η εκτέλεση του κώδικά τους για ένα χρονικό διάστημα που ορίζεται με την εντολή `yield return new WaitForSeconds(float seconds)` και έπειτα συνεχίζουν την εκτέλεση από το σημείο που είχε μείνει.

Εικόνα 4.13 – Αποτέλεσμα εκτέλεσης Coroutine waitText

Όταν αναγνωριστεί η εικόνα στόχος, τότε η μεταβλητή targetRecognized αποκτά την τιμή true και το script component 'GUIManager', φέρνει την αρχική οθόνη στην μορφή που φαίνεται στην Εικόνα 4.14.



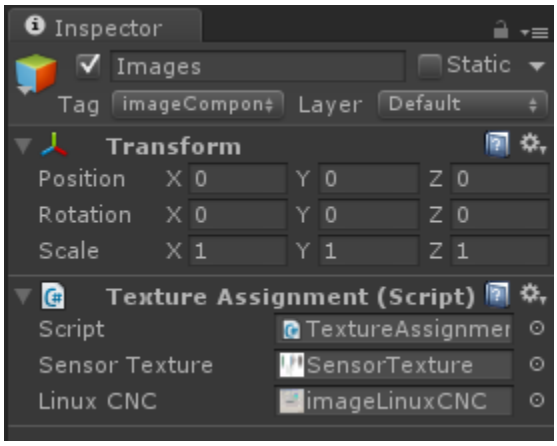
Εικόνα 4.14 – Αποτέλεσμα εκτέλεσης μεθόδου OnGUI ( ) του GUIManager

Έχει δημιουργηθεί πλέον ένα κουμπί στην οθόνη που μπορεί να πατήσει ο χρήστης της εφαρμογής για να προχωρήσει στο κύριο μέρος της. Αυτό έχει επιτευχθεί με χρήση της μεθόδου OnGUI( ){ … } που όπως έχει αναφερθεί και στο Κεφάλαιο 2 κληρονομείται από την κλάση MonoBehaviour. Ο κώδικας που συμπληρώνεται στο σώμα της μεθόδου (εντός των άγκιστρων) εκτελείται πολλαπλές φορές ανά frame και δίνει τη δυνατότητα χρήσης των στατικών μεθόδων της κλάσης GUI (οι πιο συνηθισμένες είναι: GUI.Box(…), GUI.Button(…), GUI.Label(…)) προκειμένου να δημιουργούνται πεδία διάδρασης με το χρήστη.

Εάν ο χρήστης επιλέξει το κουμπί 'Press to continue', τότε το GUIManager component απενεργοποιεί το game object 'InitialScreen' (κατ' επέκταση και τα παιδιά του, και συνεπώς αποκρύπτει την αρχική εικόνα) και επιτρέπει πλέον την εμφάνιση των υπόλοιπων αντικειμένων που βρίσκονται ενεργά εντός σκηνής. Παράλληλα ενεργοποιεί το component 'GUIText' και το script component 'MainManager' που είναι προσαρτημένα στο αντικείμενο 'Manager'.

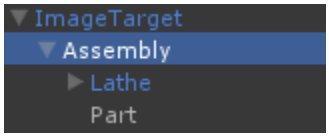
- Images : Περιλαμβάνει το script component 'TextureAssignment', το οποίο παρέχει τις δημόσιες μεταβλητές *SensorTexture* και *linuxCNC* που είναι τύπου Texture2D. Στις μεταβλητές αυτές αντιστοιχίζονται από την καρτέλα Inspector

εικόνες. Σκοπός λοιπόν, του game object 'Images' είναι να κρατά, στις δημόσιες μεταβλητές του, τις εικόνες αυτές προκειμένου να μπορεί να αποκτηθεί πρόσβαση σε αυτές από άλλα αντικείμενα που τις χρησιμοποιούν κατά την εκτέλεση της εφαρμογής (βλ. Εικόνα 4.15). Παράλληλα έχει προσαρτηθεί σε αυτό η ετικέτα 'imageComponent'.



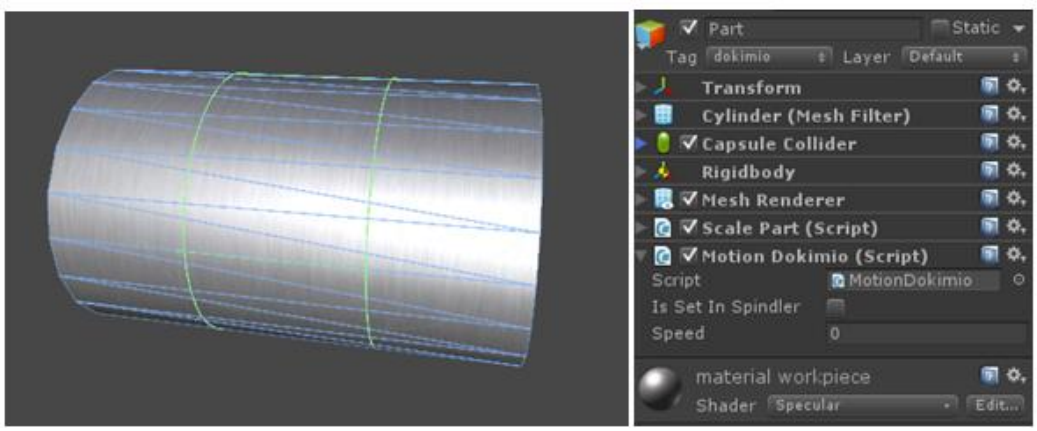
Εικόνα 4.15 – Game object 'Images'

- Assembly : Είναι γονέας των αντικειμένων 'Part' και 'Lathe', ενώ δεν διαθέτει πρόσθετα components εκτός του Transform. Το αντικείμενο 'Assembly' έχει τεθεί παιδί του αντικειμένου 'ImageTarget' (βλ. Εικόνα 4.16), έτσι αυτό είναι ενεργό όταν το αντικείμενο 'ImageTarget' είναι ενεργό. Σκοπός του, είναι να επιτρέπει αλλαγές στην χωροθέτηση των αντικειμένων παιδιών εντός σκηνής (μεταβάλλοντας τις παραμέτρους του Transform component), διατηρώντας όμως τις σχετικές αποστάσεις που αυτά έχουν μεταξύ τους.



Εικόνα 4.16 – Αντικείμενο Assembly/Σχέσεις Ιεραρχίας

Το αντικείμενο 'Part' είναι το εικονικό κυλινδρικό δοκίμιο το οποίο προσομειώνει το αντίστοιχο πραγματικό, προκειμένου να καταδείξει τη στήριξή του στον τόρνο EMCO. Βάση της δημιουργίας του έχει αποτελέσει ένα 'Cylinder' game object, που παρέχει το Unity, ενώ έχουν προστεθεί πάνω σε αυτό τα components 'Rigidbody' (με ενεργή την επιλογή isKinematic) και 'CapsuleCollider' (με ενεργή την επιλογή isTrigger) και επιπρόσθετα τα script components 'ScalePart' και 'MotionDokimio' (βλ. Εικόνα 4.17).



Εικόνα 4.17 – Αντικείμενο 'Part' και components

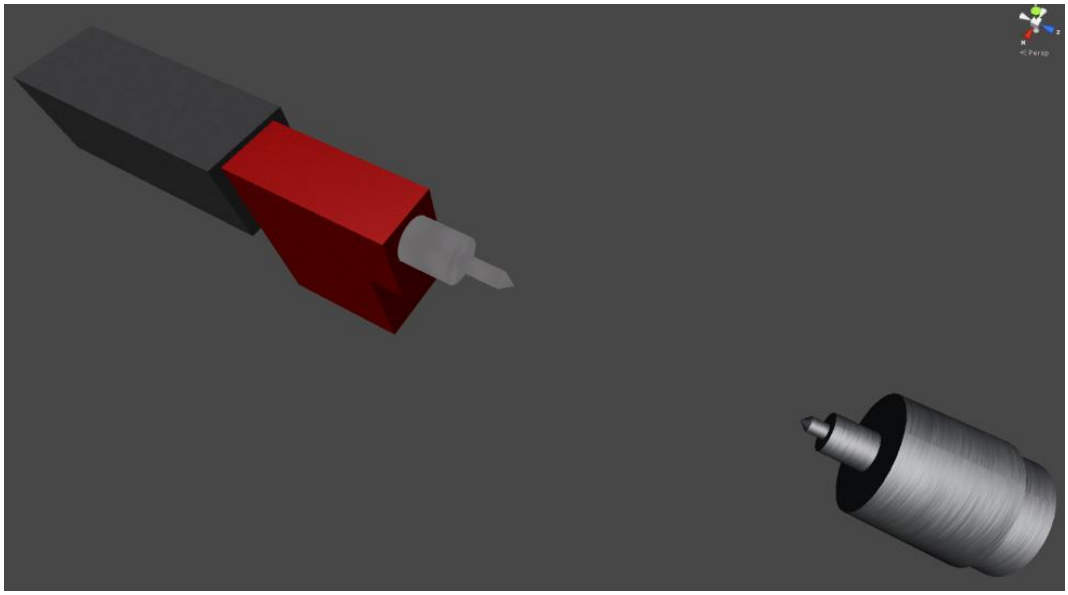
Το 'ScalePart' component παρέχει τις δημόσιες μεθόδους *void GetSensorValue( )* και *void MannuallySetHeight(int height)*. Οι δυο αυτές μέθοδοι όταν κληθούν αλλάζουν το ύψος του κυλίνδρου (συνιστώσα y του διανύσματος localScale του αντικειμένου). Η διαφορά τους είναι ότι η πρώτη μέθοδος λαμβάνει την τιμή του ύψους από την ένδειξη που παρέχει η πλακέτα arduino, ενώ η δεύτερη τη λαμβάνει από το όρισμά της (η διάσταση δίνεται σε εκατοστά και τροποποιείται σε μονάδες Unity εντός του script component). Να σημειωθεί ότι όταν μεταβάλλεται το ύψος του κυλίνδρου ενημερώνεται και η αντίστοιχη διάσταση του Collider που φέρει.

Το 'MotionDokimio' component παρέχει τη δημόσια μέθοδο *void MoveFromAnotherScript( )*, η οποία όταν κληθεί εκκινεί την Coroutine *MoveToSpindle( )*. Το αποτέλεσμα είναι το δοκίμιο να μετακινείται με σταθερή ταχύτητα μέχρι να συναντήσει τον Collider του αντικειμένου με την ετικέτα 'spindle'. Αυτό επιτυγχάνεται με τον κώδικα:

```
IEnumerator MoveToSpindle()// Δημιουργία Coroutine
 {
  yield return new WaitForSeconds (0.5f);
  while(!isSetInSpindler)
  {
   MoveRight ();
   yield return null;
  }
 }

 // Ανίχνευση Σύγκρουσης
 void OnTriggerEnter(Collider other)
 {
  if (other.gameObject.CompareTag ("spindle"))
```

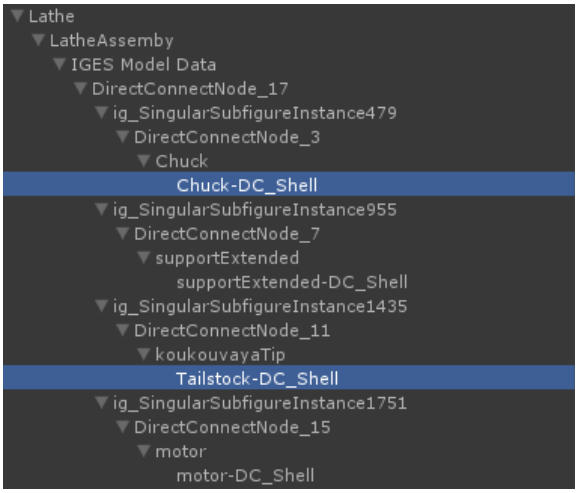
```
    {
     Debug.Log ("Collision Detected");
     isSetInSpindler = true;
    }
  }
```

Ο τόρνος προσομειώνεται με το αντίστοιχο εικονικό αντικείμενο 'Lathe'. Η γεωμετρία του σχεδιάστηκε αρχικά με χρήση του λογισμικού SolidWorks και το μοντέλο που προέκυψε αποθηκεύθηκε σε μορφή .IGS. Κατόπιν εισήχθη στο 3DsMax όπου παραμετροποιήθηκε (ορίστηκε η αρχή των αξόνων του μοντέλου) και αποθηκεύθηκε σε μορφή .fbx, προκειμένου το πλέγμα του μοντέλου να μπορέσει να αξιοποιηθεί εντός του Unity. Τέλος στο περιβάλλον του Unity έγινε περαιτέρω επεξεργασία, προκειμένου να προστεθούν υλικά που προσδίδουν πιο ρεαλιστική όψη στο μοντέλο (βλ. Εικόνα 4.18).



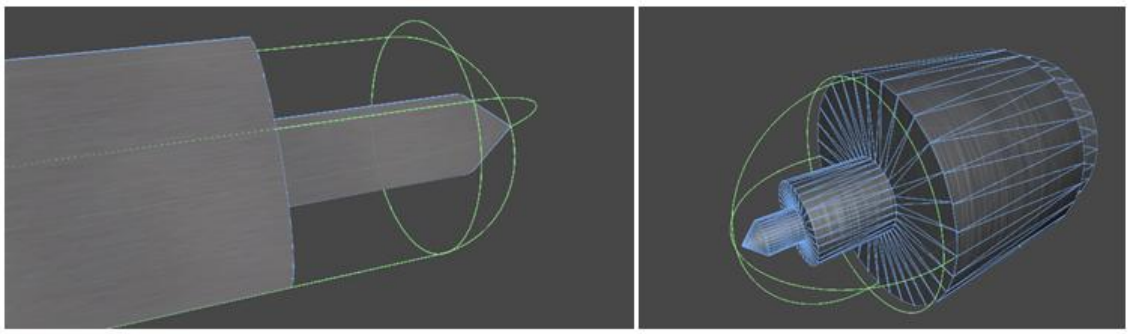
Εικόνα 4.18 – Αντικείμενο Lathe/ Τελική μορφή

Το game object 'Lathe' λοιπόν, λόγω της διαδικασίας που ακολουθήθηκε για τη δημιουργία του, συντίθεται από ένα πλήθος αντικειμένων παιδιών, όπως φαίνεται στην Εικόνα 4.19. Έχει προσαρτημένη επίσης την ετικέτα 'tornos', καθώς και το script component 'ActivationController' το οποίο παρέχει τη δημόσια μέθοδο *void ChangeVisibilityOfLathe( ),* προκειμένου να αλλάζει την κατάσταση του MeshRenderer component των αντικειμένων παιδιών καθιστώντας το μοντέλο του τόρνου από ορατό σε αόρατο και αντίστροφα.



Εικόνα 4.19 – Αντικείμενο 'Lathe' / Σχέσεις Ιεραρχίας

Από τα αντικείμενα παιδιά, ωστόσο, δύο έχουν ιδιαίτερη σημασία για την εκτέλεση της εφαρμογής, καθώς αποτελούν τα τμήματα του τόρνου που χρησιμοποιούνται για την συγκράτηση του δοκιμίου. Αυτά είναι τα 'Chuck-DC_Shell' και 'Tailstock-DC_Shell' (βλ. Εικόνα 4.20) στα οποία έχουν προστεθεί οι ετικέτες 'spindle' και 'tailstock' αντίστοιχα. Στα αντικείμενα αυτά έχει προσαρτηθεί component CapsuleCollider (με ενεργή την επιλογή IsTrigger), προκειμένου να ανιχνεύεται μέσω της μεθόδου *OnTriggerEnter* η επαφή με το αντικείμενο 'Part'. Επιπρόσθετα το 'Tailstock-DC_Shell' περιλαμβάνει το component 'Rigidbody' (με ενεργή την επιλογή IsKinematic), καθώς και το script component 'TailstockMotion'. Το component 'TailstockMotion' παρέχει τη δημόσια μέθοδο *void MoveTailstockFromExternalScript( )* η οποία όταν κληθεί κινεί το αντικείμενο 'Tailstock-DC_Shell' με σταθερή ταχύτητα προς το αντικείμενο 'Part' και σταματά μόλις συναντήσει τον Collider του δεύτερου (αντίστοιχη λειτουργία με τη μέθοδο *void MoveFromAnotherScript( )*).



Εικόνα 4.20 – (α) Tailstock-DC_Shell (β) Chuck-DC_Shell

- Manager : Το αντικείμενο 'Manager' έχει ειδική σημασία καθώς, στο κυρίως μέρος της εκτέλεσης της εφαρμογής, αυτό είναι υπεύθυνο για την εμφάνιση και τις επιλογές της γραφικής διεπιφάνειας χρήστη (Graphical User Interface), και για την μεταχείρηση των αντικειμένων που συμμετέχουν στη σκηνή, με κλήση των δημόσιων μεθόδων των components που διαθέτουν.
  Αυτή του η λειτουργία επιτυγχάνεται λόγω του script component 'MainManager', το οποίο ενεργοποιείται από το component 'GUIManager' του αντικειμένου InitialScreen μετά από την αναγνώριση της εικόνας στόχου.

Το script component 'MainManager' υλοποιεί τις ακόλουθες λειτουργίες:

- Δέχεται αντικείμενο κλάσης που εφαρμόζει το interface 'IState'. Κάθε τέτοια κλάση λοιπόν θα έχει υλοποιημένες και δημόσια προσβάσιμες τις μεθόδους *void Action( ), void Reset( )* και void GUIImage( ). Για συντομία, όλες τις κλάσεις που υλοποιούν το εν λόγω interface τις ονομάζουμε States. Να σημειωθεί ότι τα States δεν έχουν σχέση κληρονομικότητας με την κλάση MonoBehaviour και συνεπώς δεν είναι script components (δεν μπορούν να χρησιμοποιήσουν τις μεθόδους Start(), Update() κλπ).

  Για να δέχεται η κλάση MainManager αντικείμενα των κλάσεων που αναφέραμε έχει δηλωθεί το namespace του IState με την εντολή:

  ```
  using Assets.Codes.Interface
  ```

  Κατόπιν ορίζεται member variable τύπου IState με την προσθήκη της εντολής:

  ```
  private IState activeState;
  ```

- Μεταθέτει μέρος της εκτέλεσης των μεθόδων Update( ) και OnGUI( ) στις μεθόδους των αντικειμένων στα οποία δείχνει η μεταβλητή activeState. Αυτό επιτυγχάνεται με τον κώδικα που ακολουθεί:

```
void Update ()
{
 activeState.Action (); // Εκτελούνται η εντολές που διαθέτει η μέθοδος
                        // Action()
}
void OnGUI()
{
 activeState.GUIImage();// Εκτελούνται η εντολές που διαθέτει η μέθοδος
                        // GUIImage( )
  ...  }
```

Είναι σαφές ότι η λειτουργία της εφαρμογής αλλάζει δυναμικά και εξαρτάται από τις μεθόδους του State που είναι ενεργό στο component 'MainManager'.

- Αρχικοποιεί τα αντικείμενα των States που μπορούν να χρησιμοποιηθούν κατά την εκτέλεση της εφαρμογής και τα αποθηκεύει σε ένα array τύπου IState. Καθορίζει παράλληλα την σειρά με την οποία αυτά μπορούν να εναλλάσονται. Αυτά επιτυγχάνονται με τον ακόλουθο τρόπο:

```
IState [] everyState; // Δημιουργία array τύπου IState

 void Start ()
 {
  everyState = new IState[] {new InitialState(),new ScaleState(),new
ScaleStateTwo(),new SetInSpindleState(),new SetInTailstockState(),new
CorrectionState()}; // Δημιουργία όλων των State που μπορούν να συμμετέχουν
                    // στην εκτέλεση της εφαρμογής
  stateCounter = 0;
  activeState = everyState [stateCounter];  // Αρχικοποίηση της activeState
 }

void OnGUI()
 {
   activeState.GUIImage()
// Δημιουργία πεδίων για πλοήγηση του χρήστη
  if (GUI.Button (new Rect (100, Screen.height - 100, 150, 50), "Previous"))
  {
   activeState.Reset();
   activeState = everyState[stateCounter-1]; // Εναλλαγή ενεργού State με
                             // βάση τη θέση που κατέχει στο array everyState
   stateCounter--;
  }
  if (GUI.Button (new Rect (Screen.width - 200, Screen.height - 100, 150,
50), "Next"))
  {
   activeState = everyState[stateCounter +1]; // Εναλλαγή ενεργού State με
                             // βάση τη θέση που κατέχει στο array everyState

   stateCounter++;
  }
 }
```

Το πλεονέκτημα αυτής της προσέγγισης είναι ότι προσθέτοντας ένα State ή εναλλάσσοντας τη σειρά που κατέχει στο array everyState, άμεσα αλλάζει και ο τρόπος λειτουργίας της εφαρμογής, πράγμα που βοηθά στη μελλοντική επέκτασή της. Στην επόμενη ενότητα παρουσιάζονται τα States και τα αποτελέσματα που επιφέρουν κατά την εκτέλεση της εφαρμογής.

4.3 Αποτελέσματα των State / Λειτουργία της εφαρμογής

Αρχικά πρέπει να αναφέρουμε ότι όλες οι κλάσεις State στη δήλωση των namespaces περιλαμβάνουν τη γραμμή:

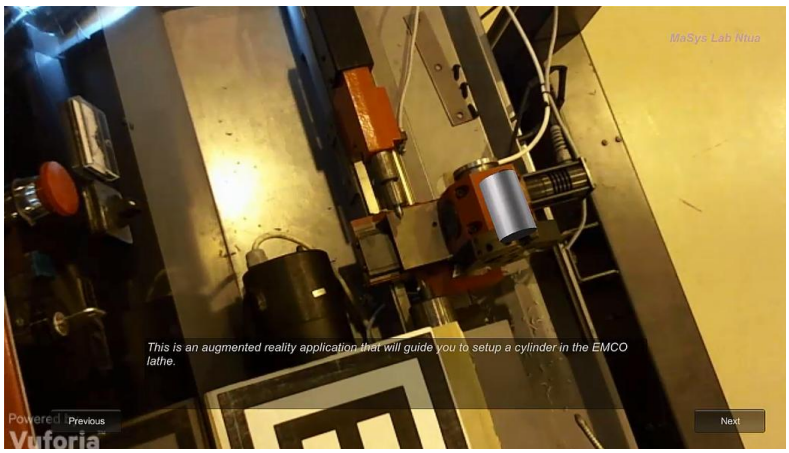
```
using Assets.Codes.Interface;
```

Ενώ η δήλωση της κληρονομικότητας, δηλαδή η εφαρμογή του interface IState γίνεται με τον ακόλουθο τρόπο:

```
public class ClassName :IState
```

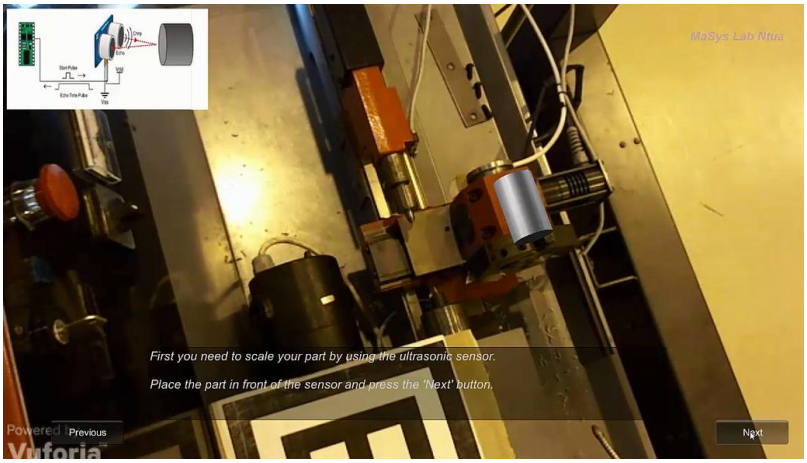
Στο κύριο μέρος των κλάσεων αυτών υλοποιούνται και ορίζονται δημόσιες οι μέθοδοι που επιβάλλει το IState (void Action, void GUIImage, void Reset( )). Μετά την αναγνώριση της εικόνας στόχου απενεργοποιείται το αντικείμενο InitialScreen και η λειτουργία της εφαρμογής ανατίθεται στα αντικείμενα των State( ) που αρχικοποιούνται εντός του script component 'MainManager'. Η σειρά με την οποία καλούνται παρουσιάζεται στη συνέχεια, ενώ η εναλλαγή τους καθορίζεται από το χρήστη με το πάτημα των κουμπιών πλοήγησης 'Previous' και 'Next' που εμφανίζονται στην οθόνη.

- InitialState: Είναι υπεύθυνο για την εμφάνιση και τη χωροθέτηση του εισαγωγικού μηνύματος που περιγράφει το σκοπό της εφαρμογής αυτής (βλ. Εικόνα 4.21). Να σημειωθεί ότι κατά την εκτέλεση της εφαρμογής χρησιμοποιούνται μέθοδοι και αντικείμενα της κλάσης GUIStyle τα οποία διαμορφώνουν την μορφή των πεδίων της γραφικής διεπιφάνειας χρήστη.



Εικόνα 4.21 – Αποτέλεσμα InitialState

- ScaleState: Εμφανίζει κατάλληλη εικόνα στο επάνω μέρος της οθόνης που υποδεικνύει στο χρήστη τη λειτουργία του αισθητήρα. Επίσης δημιουργεί αντίστοιχο μήνυμα στο κάτω μέρος της οθόνης (βλ. Εικόνα 4.22). Η εικόνα που εμφανίζεται είναι αντιστοιχισμένη στη μεταβλητή SensorTexture του game object 'Images' και το ScaleState αποκτά πρόσβαση σε αυτήν με την εντολή:
```
GameObject.FindWithTag("imageComponent").GetComponent<TextureAssignment>().SensorTexture;
```



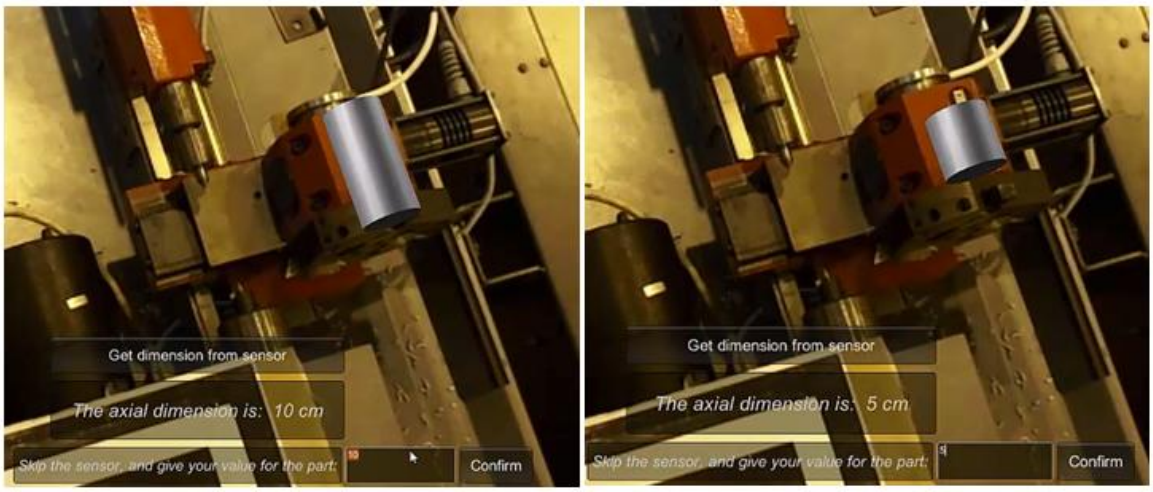
Εικόνα 4.22 – Αποτέλεσμα ScaleState

- ScaleStateTwo: Είναι υπεύθυνο για την εμφάνιση πεδίου στο οποίο ο χρήστης μπορεί να εισάγει από το πληκτρολόγιο το ύψος του κυλίνδρου τον οποίο επιθυμεί να στηρίξει στον τόρνο. Προκειμένου να ελεγχθεί εάν η πληροφορία που εισάγεται από το πληκτρολόγιο είναι έγκυρη έχει δημιουργηθεί η κλάση InputSecurity, η οποία παρέχει τις μεθόδους *public static bool CheckInt(string input)* και *public static bool CheckDecimal(string input)* που επιστρέφουν την τιμή true εάν σαν παράμετρος τους δοθεί ακέραιος και δεκαδικός αντίστοιχα. Οι μέθοδοι που αναφέρθηκαν είναι στατικές, που σημαίνει ότι μπορούν να χρησιμοποιηθούν χωρίς να έχει πρώτα δημιουργηθεί αντικείμενο της κλάσης στην οποία έχουν οριστεί. Έτσι, εάν κατά τη διάρκεια που είναι ενεργό αντικείμενο της κλάσης ScaleStateTwo εισαχθεί από το χρήστη εσφαλμένη πληροφορία, εμφανίζεται το μήνυμα που φαίνεται στην Εικόνα 4.23.

Εικόνα 4.23 – Έλεγχος εγκυρότητας εισόδου

Παράλληλα εμφανίζεται κουμπί που παρέχει τη δυνατότητα η πληροφορία για το ύψος να ληφθεί από την ένδειξη του αισθητήρα (βλ. Εικόνα 4.24). Όταν εισαχθεί έγκυρη πληροφορία για το ύψος του δοκιμίου ενημερώνεται η αντίστοιχη διάσταση του εικονικού δοκιμίου, όπως φαίνεται στην Εικόνα 4.25.

Εικόνα 4.24 – Πεδία που παρέχονται από το ScaleStateTwo



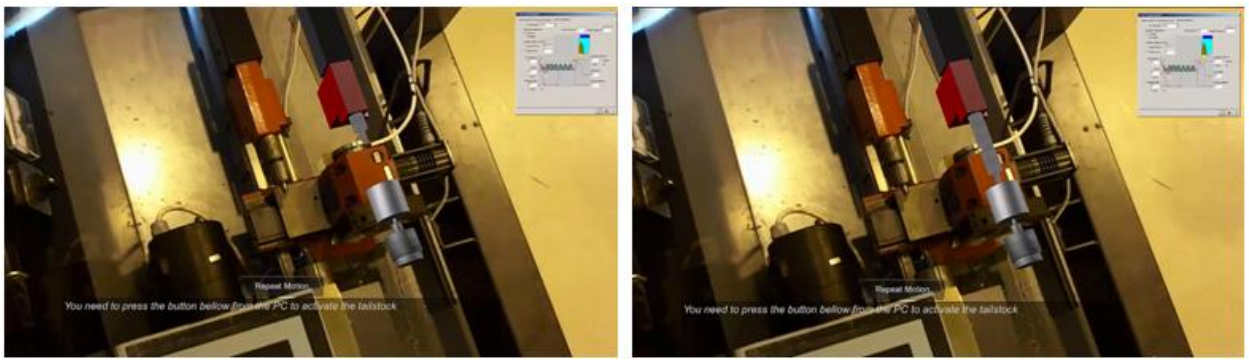
Εικόνα 4.25 – Αποτελέσματα ScaleStateTwo

- SetInSpindleState: Καθιστά αρχικά τον τόρνο ορατό με την εκτέλεση της εντολής: `GameObject.FindWithTag("tornos").GetComponent<ActivationController> ().ChangeVisibilityOfLathe ();`
- Στη συνέχεια μετακινεί το εικονικό βοήθημα στην κατεύθυνση του game object 'Chuck' αποκτώντας πρόσβαση στη μέθοδο *MoveFromAnotherScript( ),* που έχει περιγραφεί, εμφανίζοντας παράλληλα αντίστοιχο μήνυμα στην οθόνη. Το

αποτέλεσμα φαίνεται στην Εικόνα 4.26. Επίσης, δημιουργεί κουμπί στην οθόνη που εκτελεί επανάληψη της κίνησης.

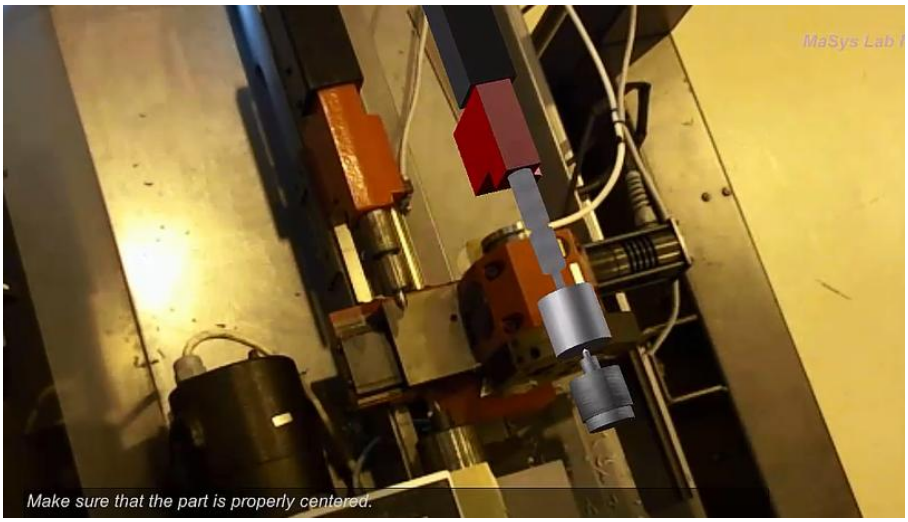
Εικόνα 4.26 – Αποτελέσματα SetInSpindleState

- SetInTailstockState : Μετακινεί το game object 'Tailstock-DC_Shell' καλώντας τη μέθοδο *MoveTailstockFromExternalScript( )* , ολοκληρώνοντας έτσι την επίδειξη στήριξης του δοκιμίου. Παράλληλα εμφανίζει γραπτό μήνυμα και εικόνα στην οθόνη που υποδεικνύουν στο χρήστη τη χρήση του λογισμικού linuxCNC για την ενεργοποίηση της κουκουβάγιας του πραγματικού τόρνου. Το αποτέλεσμα φαίνεται στην Εικόνα 4.27.



Εικόνα 4.27 – Αποτέλεσμα SetInTailstockState

- CorrectionState: Εμφανίζει γραπτό μήνυμα στην οθόνη προτρέποντας το χρήστη να επαληθεύσει πως το πραγματικό δοκίμιο που έχει τοποθετήσει στον τόρνο είναι σωστά ευθυγραμμισμένο (βλ. Εικόνα 4.28).



Εικόνα 4.28 – Αποτέλεσμα CorrectionState

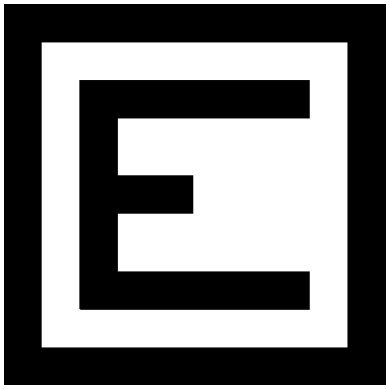
Ο πλήρης κώδικας της πρώτης εφαρμογής παρατίθεται στο παράρτημα Α.

5. Παρουσίαση Δεύτερης Εφαρμογής

5.1 Σκοπός και προσέγγιση

Στόχος της δεύτερης εφαρμογής είναι η δημιουργία ενός εικονικού οδηγού – βοηθήματος για τη στήριξη συγκεκριμένων τεμαχίων στην τράπεζα κέντρου κατεργασιών προκειμένου να ακολουθήσει κατεργασία τους. Η ανάπτυξη μιας τέτοιας εφαρμογής επιχειρείται ώστε αυτή να αποτελέσει μέσο εκπαίδευσης νέου προσωπικού ή φοιτητών, αλλά και να αντικαταστήσει σκαριφήματα σε χαρτί ή προφορικές οδηγίες, που συνήθως χρησιμοποιούνται στη βιομηχανία. Παράλληλα θα εξεταστεί κατά πόσον το τελικό οπτικό αποτέλεσμα είναι ικανοποιητικό με τα μέσα που χρησιμοποιούνται για την αναγνώριση της εικόνας και την υπέρθεση των εικονικών μοντέλων προκειμένου να γίνουν μελλοντικές βελτιώσεις.

Αρχικά θα χρησιμοποιηθεί η ίδια εικόνα με εκείνη της προηγούμενης εφαρμογής ως στόχος προς αναγνώριση (βλ.Εικόνα 5.1). Ο ρόλος του στόχου είναι να αποτελέσει πρώτον το σημείο αναφοράς σε σχέση με το οποίο παραμετρικά θα εμφανίζονται τα εικονικά μοντέλα στη συσκευή απεικόνισης (χρησιμοποιείται τόσο η θέση στην οποία εντοπίζεται ο στόχος, όσο και ο προσανατολισμός του). Δεύτερον αποτελεί το μέσο με βάση το οποίο καθορίζεται το μέγεθος των εικονικών μοντέλων προκειμένου να προσαρμόζονται οι διαστάσεις τους προς αυτές των αντίστοιχων πραγματικών αντικειμένων στο φυσικό χώρο.Έτσι, ο πραγματικός στόχος που χρησιμοποιείται έχει διαστάσεις 20cmx 20cm.



Εικόνα 5.1 – Στόχος για αναγνώριση

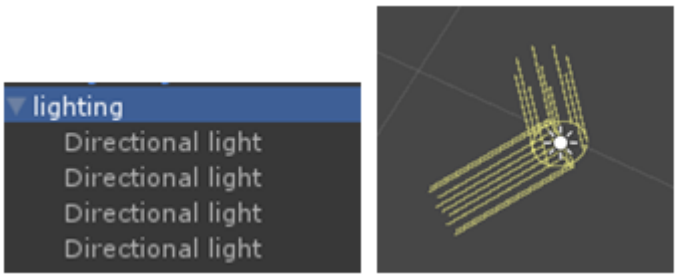
Η ανάπτυξη της εφαρμογής και η εκτέλεσή της πραγματοποιείται σε ηλεκτρονικό υπολογιστή και χρησιμοποιείται webcamera. Κατά την εκτέλεση της εφαρμογής, η κάμερα έχει στο πεδίο ορατότητας της το στόχο ο οποίος είναι τοποθετημένος στην τράπεζα κέντρου κατεργασιών, προκειμένου εκεί να βρίσκεται το

επίπεδο αναφοράς για την δημιουργία των εικονικών εξαρτημάτων. Τα εικονικά μοντέλα δημιουργούνται στο λογισμικό SolidWorksαποθηκεύονται σε μορφή .IGS και ακολουθεί τροποποίησή τους μέσω του λογισμικού 3DsMax. Κατόπιν αποθηκεύονται σε τελική μορφή .fbx προκειμένου να μπορούν να εισαχθούν στο Unity3d.

Έπειτα, δημιουργούνται τα scriptcomponentsπου περιγράφουν τη συμπεριφορά των εικονικών αντικειμένων.Να σημειωθεί ότι χρησιμοποιείται η ρύθμιση .NET 2.0 στο πεδίο ApiCompatibilityLevel. Ακολουθεί η δημιουργία των υπόλοιπων κλάσεων, με στόχο την επίτευξη της σωστής λειτουργίας της εφαρμογής, καθώς και τη δυνατότητα εύκολης τροποποίησης και μελλοντικής επέκτασής της.Τα στάδια αυτά αναλύονται στις επόμενες παραγράφους αυτού του κεφαλαίου, ενώ ο πλήρης κώδικας παρατίθεται στο παράρτημα Β.

5.2 Εικονικά Μοντέλα – Game Objects

Για αυτή την εφαρμογή η σκηνή αρχικά περιέχει τα game objects 'ImageTarget', 'ARCamera' , 'lighting' και 'Manager'. Όσον αφορά τα πρώτα δύο, αυτάχρησιμοποιούνται με τον ίδιο τρόπο που παρουσιάστηκε στην πρώτη εφαρμογή (βλ.Κεφάλαιο 4). Το game object 'lighting' είναι ένα κενό game object γονέας τεσσάρων 'Directional Light' που επιτελούν το ρόλο του φωτισμού της σκηνής (βλ.Εικόνα 5.2).



Εικόνα 5.2 – Gameobject 'lighting' καισχέσεις ιεραρχίας

Το game object 'Directional Light' είναι δημιουργημένο από το Unity και σε σχέση με τους υπόλοιπους τύπους φωτισμού η διαφορά του είναι ότι φωτίζει ομοιόμορφα (παράλληλες ακτίνες φωτός μιας κατεύθυνσης) όλη τη σκηνή ανεξαρτήτως σημείου τοποθέτησης. Οι κύριες παράμετροι που τροποποιούνται για τη χρήση του εν λόγω αντικειμένου είναι ο προσανατολισμός του, το χρώμα και η ένταση φωτισμού.

Το game object '**Manager**' διαθέτει το script component '**MainManager**', το οποίο δημιουργήθηκε για να επιτυγχάνονται οι ακόλουθες λειτουργίες:

- Μπορεί να δέχεται αντικείμενο κλάσης που εφαρμόζει το interface 'IState'. Κάθε τέτοια κλάση λοιπόν θα έχει υλοποιημένες και δημόσια προσβάσιμες τις μεθόδους *void Action( )* και void GUIImage( ) (βλ. Εικόνα 5.3).

```
1 using System.Collections;
2
3 namespace Assets.Codes.Interface
4 {
5
6     public interface IState
7     {
8         void Action();
9         void GUIImage();
10    }
11 }
```

Εικόνα 5.3 – IState interface

Για να δέχεται η κλάση MainManager τα αντικείμενα που αναφέραμε έχει δηλωθεί το namespace του IState με την εντολή:

```
using Assets.Codes.Interface
```

Κατόπιν ορίζεται member variable τύπου IState με την προσθήκη της εντολής:

```
private IState activeState;
```

Αυτή η πρακτική δίνει τη δυνατότητα εντός του MainManagerνα κληθούν οι μέθοδοι *void Action( )* και *void GUIImage( )* χωρίς να είναι γνωστό εκ των προτέρων από ποιά κλάση θα ληφθεί η υλοποίηση αυτών των μεθόδων. Έτσι λοιπόν γίνεται μετάθεση της υλοποίησης των μεθόδων *void Update( )* και*void OnGUI( )* (μέθοδοι που κληρονομούνται στο MainManager από την κλάση MonoBehaviour) σε αντικείμενο άλλης κλάσης. Αυτό γίνεται με τον κώδικα:

```
void Update ()
 {
 activeState.Action ();
 }

void OnGUI()
 {
 activeState.GUIImage();
 }
```

Αυτή η πρακτική ακολουθείται έτσι ώστε η συμπεριφορά του gameobject 'Manager' να μεταβάλλεται δυναμικά κατά την εκτέλεση της εφαρμογής. Η λογική αυτή ομοιάζει με εκείνη που ακολουθήθηκε στην πρώτη εφαρμογή, με την

διαφορά ότι στην περίπτωση αυτή το component MainManager δεν είναι υπεύθυνογια το αντικείμενο το οποίο θα δείχνει η μεταβλητή activeState κατά τη διάρκεια της εκτέλεσης της εφαρμογής, με την εξαίρεση αντικειμένου της κλάσης InitialScreenState, που αρχικοποιείται στη voidStart( ) μέθοδο του MainManagerμε τον κώδικα:

```
void Start ()
{
activeState = newInitialScreenState (this);
}
```

Στο κεφάλαιο αυτό τα αντικείμενα που υλοποιούν το interface IState θα αναφέρονται απλά ως States για να μην υπάρξει σύγχυσή τους με άλλα αντικείμενα, ενώ οι κλάσεις που τα ορίζουν θα παρουσιαστούν σε επόμενη ενότητα (ο πλήρης κώδικας παρατίθεται στο παράρτημα Β).

- Παρέχει τη δημόσια προσβάσιμη μέθοδο:

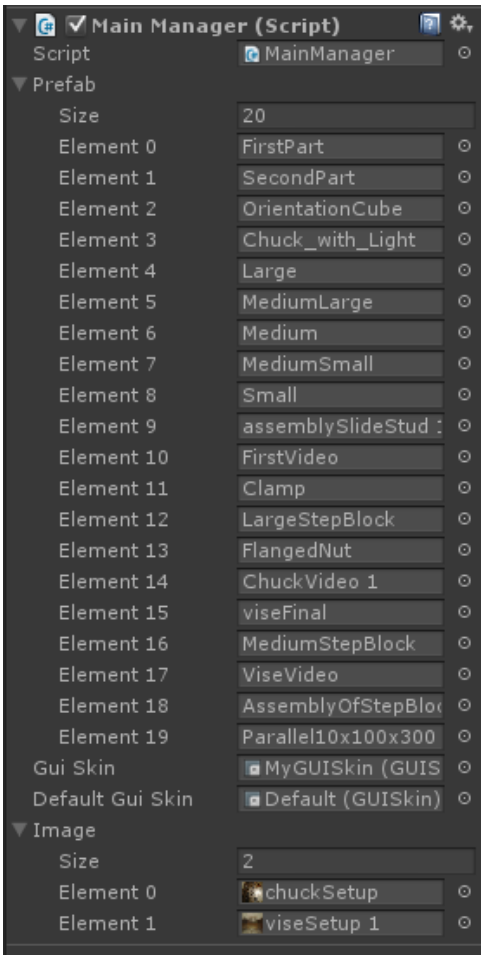
```
public void SwitchState(IState nextState)
{
activeState = nextState;
}
```

Όταν κληθεί η μέθοδος αυτή από άλλο αντικείμενο, του επιτρέπει να αλλάξει το State στο οποίο δείχνει η μεταβλητή activeStatetου MainManager και αυτομάτως αλλάζει η υλοποίησητων *void Update( )* και *void OnGUI( ).*

- Παρέχει τις δημόσια προσβάσιμες μεταβλητές prefab, image, guiSkin, defaultGuiSkin. Όλες οι μεταβλητές που αναφέρθηκαν αρχικοποιούνται από τον Inspector του Unity. Οι guiSkin και defaultGuiSkin είναι τύπου GUISkin και είναι υπεύθυνες για την εμφάνιση των κουμπιών, των γραμματοσειρών και των παραθύρων διαλόγου που εμφανίζονται κατά την εκτέλεση της εφαρμογής. Η μεταβλητή image είναι arrayτύπου Texture2D και στις θέσεις μνήμης που δεσμεύει αντιστοιχίζονται εικόνες που θα χρειαστεί να εμφανιστούν.

  Η μεταβλητή prefab είναι και η πλέον σημαντική από τις προαναφερθείσες. Είναι array τύπου GameObject και σε αυτό αντιστοιχίζονται από τον φάκελο Prefabs όλα εκείνα τα αντικείμενα που θα δημιουργηθούν κατά την εκτέλεση της εφαρμογής(βλ. Εικόνα 5.4).

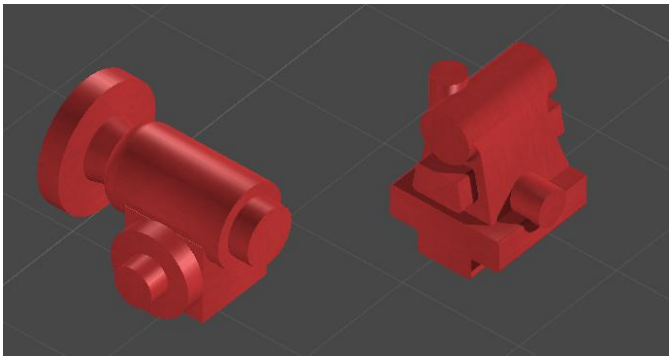
- Παρέχει τις δημόσια προσβάσιμες μεθόδους *GameObject* ***CreatePrefab****(int arrayIndex)* και*void* ***DestroyPrefab*** (GameObject name). Η πρώτη μέθοδος δέχεται σαν όρισμα έναν ακέραιο και επιστρέφει το GameObject που είναι αποθηκευμένο στη θέση που υποδεικνύει το όρισμα στο array prefab που αναφέρθηκε στην προηγούμενη παράγραφο, παράλληλα το καθιστά παιδί του gameobject 'ImageTarget' που βρίσκεται ήδη εντός σκηνής. Η μέθοδος αυτή έχει γίνει overloadedμε πέντε υλοποιήσεις, προκειμένου να μπορούν να οριστούν προαιρετικά επιπρόσθετες παράμετροι του αντικειμένου που δημιουργεί. Η δεύτερη μέθοδος καταστρέφει το gameobject που δίνεται στο όρισμά της. Αυτές οι μέθοδοι χρησιμοποιούνται σε διάφορα States (τα States δεν κληρονομούν από το MonoBehaviour και συνεπώς δεν μπορούν να καλέσουν τις μεθόδους του), έτσι τα αντικείμενα που απαρτίζουν τη σκηνή αλλάζουν δυναμικά.



Εικόνα 5.4 – MainManager και δημόσιες μεταβλητές

Τα αντικείμενα που δεν βρίσκονται εξ' αρχής στη σκηνή τοποθετούνται στο φάκελο Prefabs, όπου και παραμετροποιούνται πλήρως, προκειμένου η μορφή και η συμπεριφορά τους να είναι η επιθυμητή, όταν κατά την εκτέλεση της εφαρμογής με κάποια κλήση της CreatePrefab μεθόδου δημιουργηθούν. Ακολουθεί η περιγραφή τους.

Πρώτα σχεδιάστηκαν στο SolidWorks τα τεμάχια που φαίνονται στην Εικόνα 5.5 και έπειτα στο 3d'sMax τροποποιήθηκε το μέγεθος, ο προσανατολισμός και η χωροθέτησή τους σε σχέση με την αρχή των αξόνων. Κατόπιν τα μοντέλα εισήχθησαν στο Unity ως prefabs (βλ. Εικόνα 5.6). Εντός του Unity στα μοντέλα προστέθηκαν υλικά προκειμένου να φαίνονται πιο ρεαλιστικά, ενώ η ίδια μέθοδος σχεδίασης έχει ακολουθηθεί και για τα υπόλοιπα prefabs.



Εικόνα 5.5 – Εικονικά μοντέλα τεμαχίων

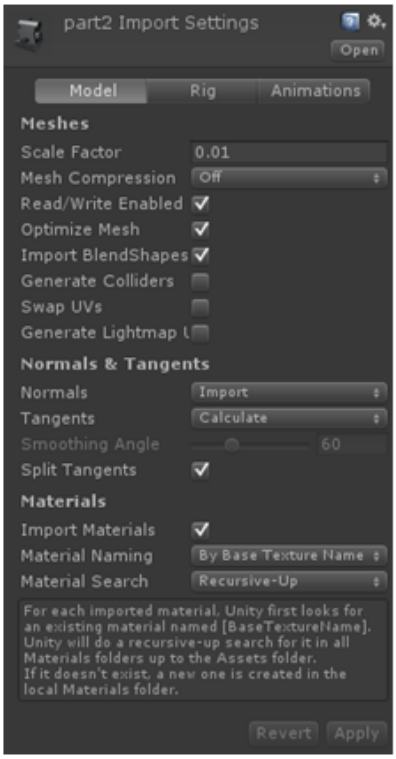
Ο ρόλος τους είναι να εμφανίζονται στη σκηνή αφού αναγνωριστεί ο στόχος από την κάμερα και με κατάλληλες ενέργειες να υποδεικνύεται ο τρόπος στήριξής τους πάνω στην τράπεζα (για το κάθε τεμάχιο υπάρχει δυνατότητα επιλογής μεταξύ έξι διαφορετικών προσανατολισμών). Για να αποκτήσουν την τελική λειτουργική συμπεριφορά τους, στο κάθε ένα από αυτά προστίθενται τα ακόλουθα components(βλ.Εικόνα 5.6 (β)):

- ***Rigidbody***,με ενεργοποιημένη την επιλογή IsKinematic, προκειμένου η μετακίνηση του αντικειμένου να υπολογίζεται με τροποποίηση του Transformcomponentκαι όχι με χρήση του εργαλείου Physics Engine. Να σημειωθεί ότι την ίδια επιλογή χρησιμοποιούμε στη συγκεκριμένη εφαρμογή για όλα τα game objects εκτός από την ARCamera.
- ***BoxCollider***, με ενεργοποιημένη την επιλογή IsTrigger. Ο ρόλος αυτού του Componenteίναι να οριοθετήσει έναν όγκο αναφοράς που περιέχει το αντικείμενο. Σε αυτό τον όγκο ανιχνεύονται συμβάντα (εισχώρηση άλλου όγκου αναφοράς, επιλογή με το mouseαπό το χρήστη και άλλα) που μπορούν να παρακολουθούνται και από άλλα components και να ενεργοποιούνται ενέργειες για την διαχείρισή τους.
- ***MovingPart***: Είναι component script που δημιουργήθηκε προκειμένου να προσδίδει συμπεριφορά κίνησης στο αντικείμενο. Διαθέτει τις επόμενες δημόσια προσβάσιμες μεθόδους:
  RotateToPlane(int planeIndex): Χρήσητηςμεθόδου επιτρέπει την ακαριαία αλλαγή του προσανατολισμού του αντικειμένου. Σαν όρισμα χρησιμοποιείται ακέραιος αριθμός από το 1 έως το 6, που αντιστοιχεί σε προκαθορισμένο προσανατολισμό.
  DemoRotate(): Περιστρέφει το αντικείμενο με σταθερή ταχύτητα.

*MoveX*Direction( ) / MoveOppositeXDirection( ) / *MoveY*Direction( ) /MoveOppositeYDirection( ) /*MoveZ*Direction( ) /MoveOppositeZDirection( ): Μετακινεί το αντικείμενο στην κατεύθυνση που υποδηλώνει το όνομα της μεθόδου με σταθερή ταχύτητα.

RotateCwYDirection( ) / RotateACwYDirection( ): Περιστρέφει δεξιόστροφα /αριστερόστροφα το αντικείμενο με σταθερή ταχύτητα.

- ***MouseListener:*** Διαθέτει δυο δημόσια προσβάσιμες μεταβλητές:
  bool mouseToggleSwitch: αλλάζει κατάσταση κάθε φορά που το αντικείμενο επιλεγεί από το mouse.
  int numberOfClicks: μετρητής των clicks.
  Οι δύο αυτές μεταβλητές παίρνουν την τιμή τους από τη μέθοδο OnMouseDown, που διαθέτει το component MouseListener. Αυτή η μέθοδος καλείται σε Colliders που έχουν την επιλογή IsTrigger ενεργή, καθώς και την Physics.queriesHitTriggers.

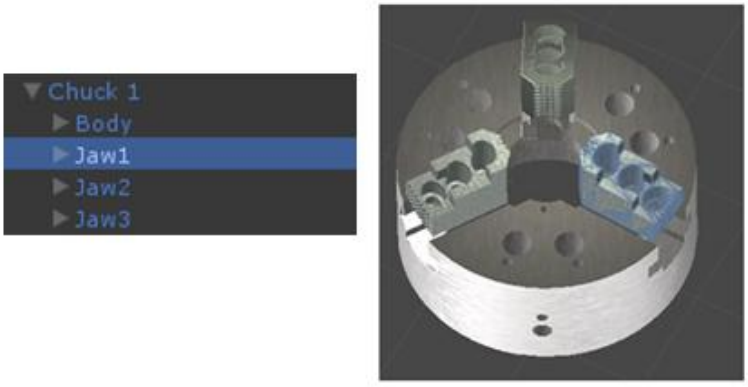


Εικόνα 5.6 – Παράμετροι Unity για εισαγωγή fbx/obj

Τα Components που αναφέραμε δημιουργούν την επιθυμητή συμπεριφορά για τα τεμάχια των οποίων θα καταδειχθεί η συγκράτηση. Δυο βασικές συσκευές

συγκράτησης που χρησιμοποιούνται είναι ένα τσωκ και μια μέγγενη, γι αυτό το λόγο δημιουργούνται και παραμετροποιούνται τα αντίστοιχα prefabs 'Chuck' και 'Vise'.

Αρχικά παρατίθεται το prefab'Chuck', το οποίο συντίθεται από τέσσερα game objects παιδιά (βλ.Εικόνα 5.7).



Εικόνα 5.7 – GameObject 'Chuck' καισχέσεις ιεραρχίας

Το αντικείμενο γονέας περιλαμβάνει το script component '***ChuckMover***' το οποίο παρέχει τις δημόσια προσβάσιμες μεθόδους:

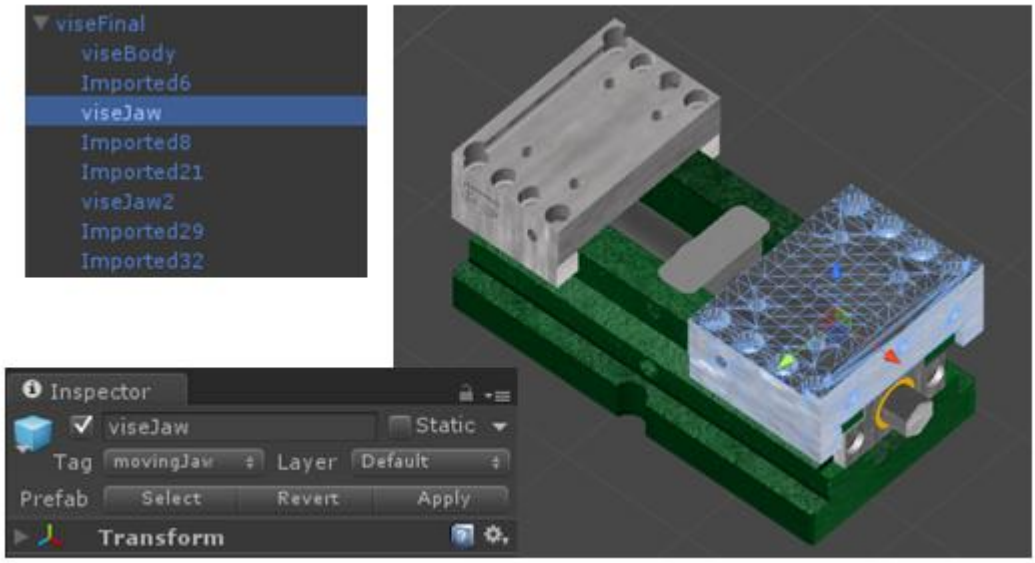
- *voidCloseJaws( )* : Κλήση της, μετακινεί ταυτόχρονα τα gameobjects 'Jaw1', 'Jaw2', 'Jaw3' με σταθερή ταχύτητα στη διεύθυνση του άξονα συμμετρίας τους και κατεύθυνση προς το εσωτερικό του gameobject 'Chuck'. Για να λειτουργήσει αυτή η μέθοδος έχουν προστεθεί οι ετικέτες 'jaw1', 'jaw2', 'jaw3' στα gameobjects 'Jaw1', 'Jaw2', 'Jaw3', καθώς η μέθοδος αποκτά πρόσβαση στο Transform component του καθενός με την εντολή *GameObject.FindWithTag(tagName).GetComponent<Transform>( ).*
- *void OpenJaws( )*: Έχει την ίδια λειτουργία με την *void CloseJaws( )* με αντίθετη φορά κίνησης.
- *void SetPositionOfJaws(float xvalue)*: Δέχεται ως όρισμα δεκαδικό αριθμό και αλλάζει την τιμή x των τοπικών συντεταγμένων των gameobjects 'Jaw1', 'Jaw2', 'Jaw3' σε αυτή. Το αποτέλεσμα είναι η ακαριαία μετακίνησή τους στην κατεύθυνση x των τοπικών συντεταγμένων τους.

Το prefab'***Vise***' απαρτίζεται από οκτώ παιδιά gameobjects (βλ.Εικόνα 5.8). Ο τρόπος που έχει δομηθεί η συμπεριφορά του ομοιάζει με αυτή του prefab 'Chuck'. Ο γονέας περιλαμβάνει το componentscript'***ViseMover***', που παρέχει τις δημόσιες μεθόδους:

- *voidOpenViseJaw( )*: Κλήση της μεθόδου εκτελεί κλείσιμο της μέγγενης με σταθερή ταχύτητα. Αυτό επιτυγχάνεται με τροποποίηση του Transform

component του gameobject 'viseJaw', αφού πρώτα αποκτηθεί πρόσβαση σε αυτό με χρήση της ετικέτας 'movingJaw'.

- *void CloseViseJaw( )*: Εκτελεί άνοιγμα της μέγγενης. Λειτουργεί όπως και η *OpenViseJaw( ).*
- *void MoveForward( )*: Μετακινεί την μέγγενη προς τα εμπρός (στην κατεύθυνση x των τοπικών συντεταγμένων του gameobject γονέα).
- *void MoveBack( )*: Μετακινεί την μέγγενη προς τα πίσω.



Εικόνα 5.8 – Vise prefab,σχέσεις ιεραρχίας και προσθήκη ετικέτας

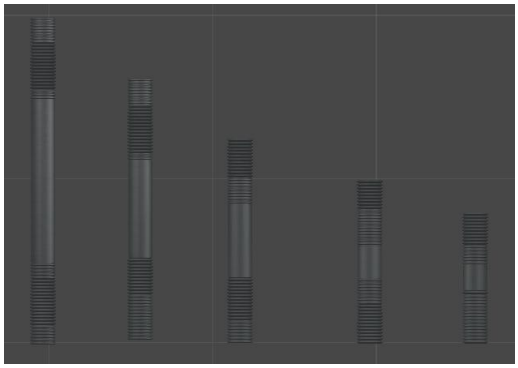
Όταν εντός της σκηνής κληθεί συσκευή (το τσακ ή η μέγγενη) για να προσδεθεί κάποιο τεμάχιο πάνω σε αυτή πρέπει πρώτα να καταδειχθεί ο τρόπος που η ίδια στηρίζεται σταθερά πάνω στην τράπεζα του κέντρου κατεργασιών. Γι αυτό το σκοπό έχουν δημιουργηθεί prefabs που έχουν τη μορφή και τη λειτουργικότητα των πραγματικών εξαρτημάτων που χρησιμοποιούνται για τη στήριξη των συσκευών συγκράτησης. Αυτά είναι:

- ❖ Φυτευτοί κοχλίες (threaded stud prefab)

Χρησιμοποιούνται πέντε κοχλίες τυποποιημένων διαστάσεων με σπείρωμα M16 και μήκος στελέχους 80,100,125,160, 200mm(βλ. Εικόνα 5.9) τα CADαρχεία των οποίων ελήφθησαν από:

http://www.fixtureworks.net/store/pc/viewcategories.asp?idCategory=1500&idproduct=130372

Κάθε κοχλίας διαθέτει το script component '***StudMover***', που παρέχει τις δημόσιες μεθόδους *void MoveRight( )*, *void MoveLeft( )*. Κλήση των μεθόδων προκαλεί μετατόπιση του κοχλία προς τα δεξιά και τα αριστερά με σταθερή ταχύτητα. Όταν οι κοχλίες εμφανιστούν στη σκηνή (με τη μέθοδο *CreatePrefab( )*)και κληθεί κάποια από τις προαναφερθείσες μεθόδους, φαίνεται ότι κινούνται κατά μήκος των σχισμών που φέρει η τράπεζα της φρέζας.



Εικόνα 5.9 – Stud prefabs

- Ολισθητήρας με σπείρωμα και διατομή T (*t-slot slide prefab* – βλ. Εικόνα 5.10(α))

Απαρτίζεται από τα βασικά components '*Transform*', '*MeshFilter*' και '*MeshRenderer*'. Δεν διαθέτει script component.

- Περικόχλιο με ενσωματωμένο δακτύλιο συσφίξεως (*flangednut prefab* – Εικόνα 5.10(β))

Πέραν των βασικών components διαθέτει το script component '*NutMover*', το οποίο παρέχει τις δημόσιες μεθόδους:

MoveDown( ): Μετακινεί το περικόχλιο προς τα κάτω με σταθερή ταχύτητα (κατεύθυνση z των τοπικών συντεταγμένων του αντικειμένου και φορά προς τα αρνητικά).

Screw( ): Προσομοιώνει την κίνηση του βιδώματος.

- Στοιχείο συσφίξεως και ασφαλίσεως (*clamp prefab*– βλ. Εικόνα 5.10(γ))

Διαθέτει το component script '*ClampMover*', το οποίο παρέχει τις δημόσιες μεθόδους:

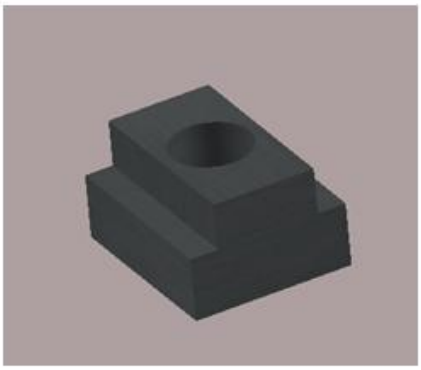
MoveDown( )/MoveUp( ): Μετακινούν το στοιχείο συσφίξεως προς τα κάτω και πάνω αντίστοιχα, με σταθερή ταχύτητα (κατεύθυνση z των τοπικών συντεταγμένων του αντικειμένου).

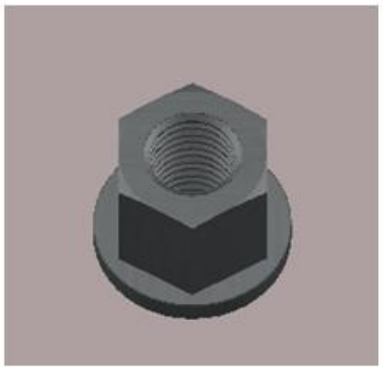
RotateCW( )/RotateACW( ): Περιστρέφουν το στοιχείο συσφίξεως δεξιόστροφα και αριστερόστροφα αντίστοιχα, με σταθερή ταχύτητα (γύρω από τον άξονα z των τοπικών συντεταγμένων του αντικειμένου).

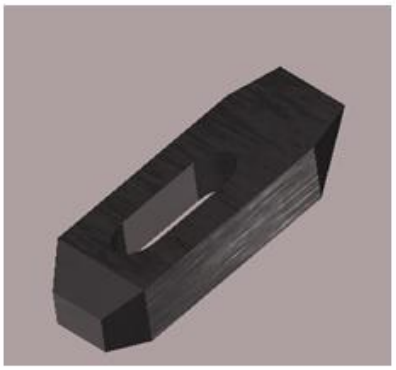
- Σφήνα με βαθμίδες *(StepBlock prefab* - βλ. Εικόνα 5.10 (δ))

Διαθέτει το component script '*StepBlockMover*', το οποίο παρέχει τις δημόσιες μεθόδους:

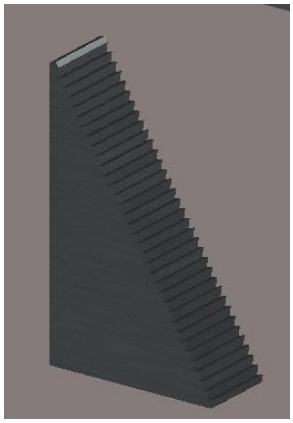
MoveForward( )/MoveBack( ): Μετακινούν την σφήνα προς τα εμπρός και πίσω αντίστοιχα, με σταθερή ταχύτητα (κατεύθυνση z των τοπικών συντεταγμένων του αντικειμένου).







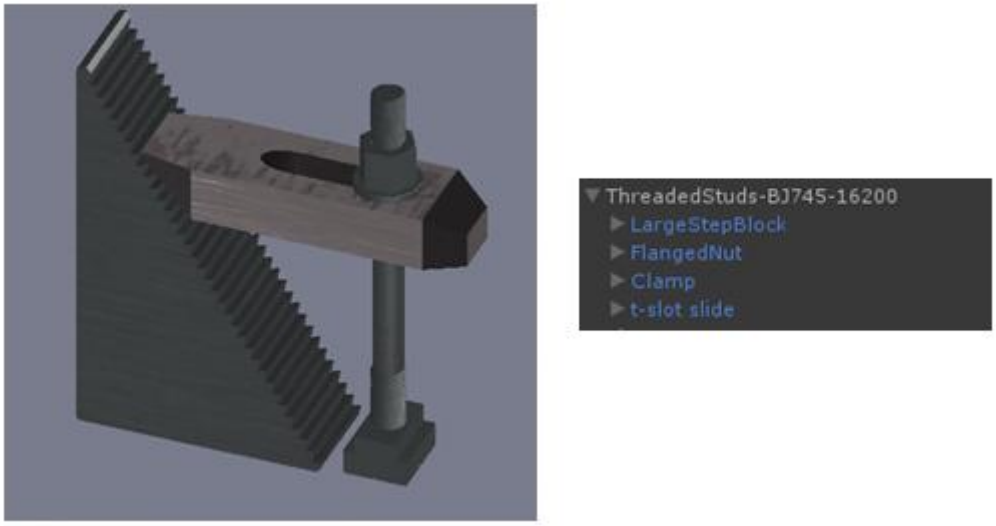
Εικόνα 5.10 – (α) Ολισθητήρας (β) Περικόχλιο (γ) Στοιχείο Συσφίξεως



(δ) Σφήνα με βαθμίδες

Τα εικονιζόμενα αντικείμενα όταν καλούνται στη σκηνή πάντα συνδυάζονται μεταξύ τους καθώς και με κάποιον κοχλία, οπότε ορίζονται και αντίστοιχες σχέσεις ιεραρχίας – κληρονομικότητας για να διασφαλίζεται η σωστή συνεργασία τους. Συγκεκριμένα ως γονέας ορίζεται ο κοχλίας προκειμένου ο άξονας συμμετρίας του και η χωροθέτησή του να αποτελέσουν σημείο αναφοράς για τα αντικείμενα παιδιά(βλ.Εικόνα 5.11).

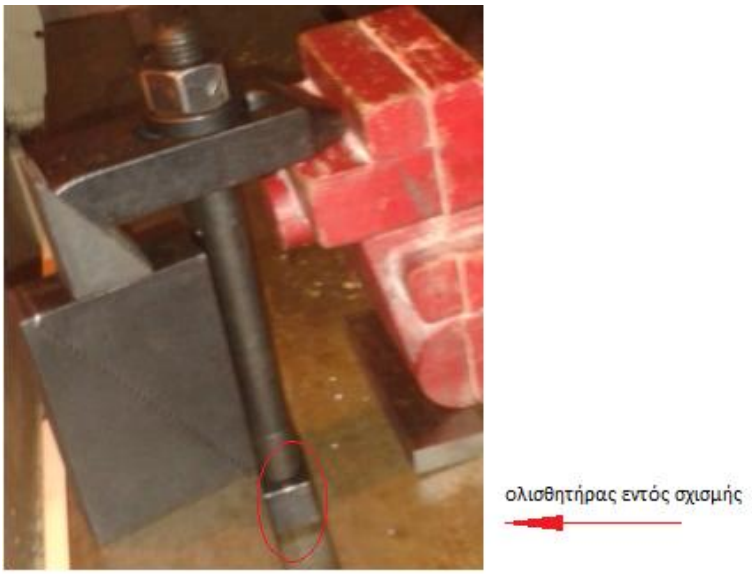
Ο ρόλος τους είναι να προσομοιώνουν τις κινήσειςπου θα πρέπει να κάνει ο χειριστής της φρέζας με τα πραγματικά εξαρτήματα στο φυσικό χώρο.



Εικόνα 5.11 – Συνεργασία αντικειμένων/ Σχέσεις ιεραρχίας

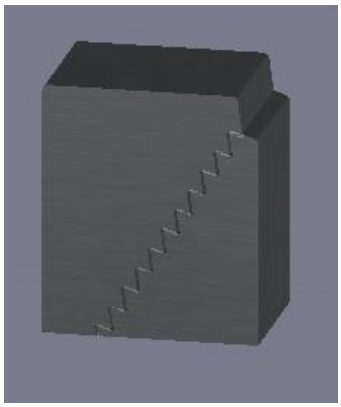
Με βάση αυτά τα εξαρτήματα, έχει δημιουργηθεί το βοηθητικό prefab '*AssemblySlideStud*' (συνδυασμός ολισθητήρα – κοχλία) και η κατηγορία 'CutThreadedStuds' που περιλαμβάνει τα prefabs 'Large', 'MediumLarge', 'Medium', 'MediumSmall', 'Small' (κοχλίας μεγέθους που υποδηλώνεται από το όνομα, του οποίου έχει αφαιρεθεί στο ένα άκρο του μήκος σπειρώματος ίσο με αυτό που φέρει ο ολισθητήρας). Το πρώτο prefab χρησιμοποιείται για να καταδείξει την πρόσδεση του κοχλία στον ολισθητήρα. Αυτό επιτυγχάνεται καθώς στο αντικείμενο παιδί '*Stud*' διαθέτει το componentscript '*BoltMover*',το οποίο το υποχρεώνει τον εικονικό κοχλία να κινηθεί προστην κατεύθυνση του εικονικού ολισθητήρακαι μόλις τον συναντά να εκτελέσει περιστροφική κίνηση (προσομοίωση βιδώματος). Το αντικείμενο γονέας περιλαμβάνει το script component '*DemoStatus*' με δημόσια προσβάσιμη τη μεταβλητή *bool demoFinished* η οποία ειδοποιείται όταν έχει ολοκληρωθεί η κίνηση προσομοίωσης, παίρνοντας την τιμή true.

Όμως, όπως έχουμε ήδη αναφέρει το επίπεδο αναφοράς για τα αντικείμενα της εφαρμογής θεωρείται η επιφάνεια της τράπεζας του κέντρου κατεργασιών. Ως εκ τούτου ο ολισθητήρας και το αντίστοιχο τμήμα του κοχλία που κινούνται εντός της σχισμής ολισθήσεως της τράπεζας (βλ.Εικόνα 5.12) δεν είναι επιθυμητό να είναι ορατά κατά την εκτέλεση της εφαρμογής και αντικαθίστανται από του αντίστοιχου μεγέθους prefabτης κατηγορίας 'CutThreadedStuds'.



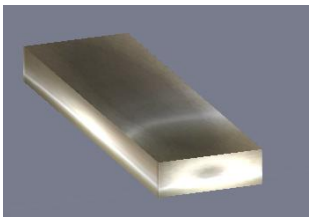
Εικόνα 5.12 – Συνδυασμός πραγματικών εξαρτημάτων και σημεία που σημεία του χώρου που δεν μπορούν να αξιοποιηθούν από την εφαρμογή.

Επίσης δημιουργήθηκε και το βοηθητικό prefab '*AssemblyOfStepBlocks*' το οποίο απαρτίζεται από δυο σφήνες τοποθετημένες με τέτοιο τρόπο, ούτως ώστε η βαθμίδες της μιας να συνεργάζονται με τις βαθμίδες της άλλης και τελικά να σχηματίζουν μια βάση σχήματος ορθογωνίου παραλληλεπιπέδου προς υπερύψωση άλλου αντικειμένου (συνήθως τρίτης σφήνας) (βλ.Εικόνα 5.13).



Εικόνα 5.13 – AssemblyOfStepBlocks prefab

Τέλοςέχειδημιουργηθείτοprefab '*Parallel 10x100x300* ' (βλ.Εικόνα 5.14), το οποίο χρησιμοποιείται ως πρότυπο πλακίδιο για υπερύψωση εξαρτήματος. Δεν διαθέτει κάποιο scriptcomponent.



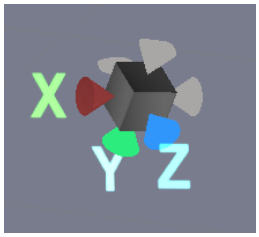
Εικόνα 5.14 – Parallel 10x100x30 prefab

Στη συνέχεια κατασκευάστηκαν prefabs για να κάνουν πιο εύληπτη στο χρήστη την ενέργεια που καλείται να εκτελέσει στο χώρο. Αυτά είναι τα:

- FirstVideo prefab
- ChuckVideo prefab
- ViseVideo prefab

Αυτά τα prefabs έχουν τη γεωμετρία επιπέδου (Plane gameObject) και όταν δημιουργηθούν στη σκηνή ορίζονται παιδιά του gameObject 'background Plane', προκειμένου να φαίνονται σε σταθερή θέση της οθόνης κατά την εκτέλεση της εφαρμογής. Διαθέτουν το scriptcomponent 'VideoHolder' το οποίο διαθέτει τις δημόσιες μεταβλητές *MovieTexture firstVideo* και *bool finished* . Η μεταβλητή firstVideo αρχικοποιείται από τον inspector με αντιστοίχιση κάποιου βίντεο που έχει εισαχθεί στο Unity ως MovieTexture. Έτσι το επίπεδο λειτουργεί πλέον ως επιφάνεια προβολής του βίντεο.Η μεταβλητή finished παίρνει την τιμή true όταν τελειώσει η αναπαραγωγή του βίντεο. Στις μεθόδους Start( ) και Update( ) ορίζεται η έναρξη της αναπαραγωγής του βίντεο.

Διαθέτουν επίσης το component 'VideoObserver' το οποίο παρακολουθεί την κατάσταση της μεταβλητής videoFinished του component 'VideoHolder' και ενημερώνει αντικείμενα άλλων κλάσεων.



Εικόνα 5.15 – OrientationCubeprefab

Το τελευταίο prefab που δημιουργήθηκε έχει ονομαστεί *'OrientationCube'* και σκοπός του είναι όταν δημιουργηθεί στη σκηνή να υποδεικνύει τους άξονες προσανατολισμού του τεμαχίου (είτε του FirstPart είτε του SecondPart). Αποτελείται από έξι αντικείμενα παιδιά (βλ.Εικόνα 5.15) και διαθέτει το script component 'MovingPart' το οποίο έχει ήδη παρουσιαστεί καθώς χρησιμοποιείται και από τα gameobjects FirtsPart και SecondPart.

5.3 Συνδυασμός και τρόποι χρήσης εικονικών αντικειμένων

Μέχρι στιγμής παρουσιάστηκαν όλα τα gameobjects που βρίσκονται εξ' αρχής στη σκηνή και τα prefabs που θα εισέλθουν στη σκηνή κατά τη διάρκεια της εκτέλεσης σε συνδυασμό με τα componentscripts που τα συνοδεύουν για να ορίζουν τη συμπεριφορά και τις δυνατότητές τους. Η κλήση των prefabs στη σκηνή, ο συνδυασμός και ο χειρισμός τους γίνεται από τα States που εναλλάσσονται με επιλογή του χρήστη.

Να υπενθυμίσουμε ότι με τον όρο 'State' στο κεφάλαιο αυτό, αναφερόμαστε σε αντικείμενα κλάσεων που υλοποιούν το Interface '*IState*', το οποίο παρουσιάστηκε στην παράγραφο 5.2. Συνεπώς, υλοποιούν τις μεθόδους *Action( )* και *GUIImage( ).* Το gameobject '*Manager*' που διαθέτει το script component '*MainManager*' αναλαμβάνει να αντιστοιχίσει κάποιο State (εναλλάσσεται κατά την εκτέλεση της εφαρμογής) στην μεταβλητή του (member variable) activeState, και έτσι αποκτά πρόσβαση στις μεθόδους του. Ο κώδικας που περιλαμβάνει η μέθοδος Action( ) εκτελείται εντός της μεθόδου Update( ) του script component 'MainManager', ενώ ο κώδικας της GUIImage( ) εκτελείται εντός της μεθόδου OnGUI( ) αντίστοιχα. Περισσότερες λεπτομέρειες για τον τρόπο λειτουργίας του script component 'MainManager' βρίσκονται στην παράγραφο 5.2 ενώ ο πλήρης κώδικας έχει προσαρτηθεί στο παράρτημα Β.

Για τον χειρισμό των gameobjects έχουν δημιουργηθεί τα ακόλουθα States τοποθετημένα με αλφαβητική σειρά:

- ChoosePartState
- ChuckSetupFinal
- ChuckSetupFour
- ChuckSetupThree
- ClampsSetupFinal
- InitialScreenState

- OrientationState
- SetupChuckState
- SetupChuckTwo
- SetupState
- ViseSetup
- ViseSetupFinal
- ViseSetupFour
- ViseSetupThree

5.3.1 Δόμηση ενός State

Όλες οι κλάσεις που παράγουν States έχουν τα ακόλουθα δομικά στοιχεία.

- Ορισμό namespace.

`Namespace Assets. Codes. States` Όλες οι κλάσεις ανήκουν στο ίδιο namespace

- Δήλωση κληρονομικότητας

`public class` **ClassName**`: IState`

- Δομητή (constructor) για αρχικοποίηση μεταβλητών

`Public` **Class Name** `(MainManager manager){ … }` Ο δομητής καλείται κάθε φορά που δημιουργείται αντικείμενο της κλάσης με χρήση της λέξης κλειδιού *new* συνοδευόμενης από το όνομα της κλάσης. Σαν παράμετρος του δομητή χρησιμοποιείται το script component 'MainManager' που είναι προσαρτημένο στο gameobject 'Manager', και έτσι δημιουργείται η σύνδεση μεταξύ των δυο αντικειμένων. Έτσι, σε μια μεταβλητή (private member variable) της κλάσης που δημιουργεί το State περνά το αντικείμενο 'MainManager' δίνοντας πρόσβαση στις μεθόδους του. Το όνομα της μεταβλητής αυτής είναι '*_manager*' οι μέθοδοι που καλούνται σε κάποιο σημείο κάθε κλάσης είναι:

  - _manager.CreatePrefab(…)
  - _manager.DestroyPrefab(…)
  - _manager.SwitchState(*new State*)

Με αυτές τις μεθόδους κάθε State επιβάλει τα αντικείμενα που θα συμπληρώνουν τη σκηνή και αντίστοιχα καταστρέφει όσα θεωρούνται περιττά, ενώ έχει τη δυνατότητα

να αλλάξει εντός του script component 'MainManager' το αντικείμενο που δείχνει η μεταβλητή activeState με χρήση της SwitchState μεθόδου.

- Υλοποιημένητημέθοδο Action( ){…}

Εντός της Action τοποθετείται ο κώδικας που επιβάλλει τις κινήσεις που θα εκτελέσουν τα εικονικά αντικείμενα που βρίσκονται στη σκηνή. Κάθε State για τα αντικείμενα που χρησιμοποιεί ορίζει μια ακολουθία κινήσεων (ολοκλήρωση μιας κίνησης ή μιας κατηγορίας κινήσεων ενεργοποιεί την επόμενη). Το πέρας μιας κίνησης ορίζεται συνήθως από μια λογική συνθήκη (σύγκριση κάποιας συνιστώσας του διανύσματος transform.localPosition/ localEulerAngles του εξεταζόμενου αντικειμένου με μια οριακή τιμή). Ο λόγος που χρησιμοποιείται λογική συνθήκη έναντι των Colliders, που χρησιμοποιούνται στην προηγούμενη εφαρμογή ως σημείο αναφοράς ολοκλήρωσης κίνησης, είναι ο καλύτερος έλεγχος του τελικού σημείου τοποθέτησης του εικονικού αντικειμένου, καθώς συμμετέχουν περισσότερα gameobjectsστη σκηνή κανένα εκ των οποίων δεν μεταβάλλει κατά την εκτέλεση τη γεωμετρία του.

Για την επιβολή των κινήσεων χρησιμοποιούνται είτε οι μέθοδοι των script components των εικονικών αντικειμένων, είτε αποκτάται πρόσβαση στο Transform component του αντικειμένου και εκτελείται κάποια από τις επόμενες μεθόδους:

```
transform.Translate(new Vector3( … ));

transform.localPosition+=new Vector3( … );
```

Για τις περισσότερες περιπτώσεις της εφαρμογής αυτής αποτελεσματικότερη κρίνεται η δεύτερη πρακτική.

Κατά την εκτέλεση της εφαρμογής διαπιστώθηκε ότι περιστροφή του αντικειμένου με μεταβολή των τιμών του διανύσματος localEulerAngles δεν επιφέρει αλλαγή στη διεύθυνση των τοπικών αξόνων του, με αποτέλεσμα να προκύπτει μη αναμενόμενη συμπεριφορά του μοντέλου σε περιπτώσεις που η περιστροφική κίνηση συνδυάζεται με μεταφορική. Η τεχνική που χρησιμοποιήθηκε για την επίλυση του προαναφερθέντος ζητήματος είναι η περιστροφή ενός άδειου gameobject, που επιβάλεται ως γονέας στην καρτέλα ιεραρχίας του εικονικού αντικειμένου, γεγονός που έχει σαν αποτέλεσμα την περιστροφή των αξόνων του αντικειμένου παιδιού. Κατόπιν εκτελείται μεταφορική κίνηση στους ενημερωμένους πλέον άξονες.

- Υλοποιημένητημέθοδο GUIImage( ){…}

Εντός της GUIImage τοποθετούνται οι εντολές που δημιουργούν πλαίσια παραμετρικά ορισμένα σε θέσεις της οθόνης, τα οποία χρησιμοποιούνται για προβολή μηνυμάτων/εικόνων προς το χρήστη της εφαρμογής ή επιλογών που καθοδηγούν την εξέλιξη της εφαρμογής.

Τυπικό παράδειγμα είναι η χρήση της εντολης:

```
if (GUI.Button (new Rect (Screen.width - 200, Screen.height - 100, 150, 50), "Proceed"))
        {
                _manager.SwitchState(new ChuckSetupFour(_manager));// αλλαγήState

        }
```

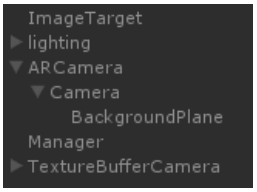
Σε αυτή την εφαρμογή έχει προτιμηθεί η χρήση παραμετροποιημένου στοιχείου GUISkin για τη βελτίωση της απεικόνισης των πεδίων που δημιουργεί η μέθοδος GUIImage, που ενεργοποιείται κατά περίπτωση από το κάθε State.

5.3.2 Αποτελέσματα των State/ Λειτουργία εφαρμογής

Όπως έχει γίνει σαφές η λειτουργία της εφαρμογής καθορίζεται από τις εντολές των διαφόρων State, συνεπώς προσθήκη περισσότερων τέτοιων αντικειμένων ή τροποποίηση των υπαρχόντων μπορεί να επεκτείνει με γρήγορο τρόπο τις δυνατότητες της εφαρμογής (ο τρόπος αυτός ανάπτυξης της εφαρμογής βασίζεται στην πηγή [22].

Σε αυτή την ενότητα παρουσιάζονται τα αποτελέσματα που επιφέρει κάθε Stateστη σκηνή με τη σειρά που εναλλάσσονται κατά τη διάρκεια της εκτέλεσης.

- Με την εκκίνηση της εφαρμογής δημιουργείται από το script component 'MainManager' το αντικείμενο **InitialScreenState**, το οποίο αναλαμβάνει να δημιουργήσει μήνυμα αναμονής όσο εκκρεμεί η αναγνώριση του στόχου. Κατόπιν ορίζει ως νέο activeState το ChoosePartState.



Εικόνα5.16 – InitialScreenState - GameObjectsστηνκαρτέλαιεραρχίας

- **ChoosePartState**. Δημιουργεί δυο τεμάχια και προτρέπει το χρήστη να διαλέξει κάποιο από αυτά. Το τεμάχιο που επιλέγεται με κλικ του ποντικιού εκτελεί περιστροφική κίνηση, προκειμένου να επιθεωρηθεί η γεωμετρία του, ενώ το δεύτερο αντικείμενο εγκαταλείπει την σκηνή. Ταυτόχρονα δημιουργούνται κουμπιά στην οθόνη για ακύρωση επιλογής ή επιβεβαίωση επιλογής.

Εικόνα 5.17 – ChoosePartState - GameObjectsστηνκαρτέλαιεραρχίας

Εικόνα 5.18 – ChoosePartStateΕπιβεβαίωση επιλογής

Συγκρίνοντας τις εικόνες 5.16 και 5.17 γίνεται αντιληπτός ο δυναμικός τρόπος με τον οποίο τα εικονικά αντικείμενα εισέρχονται στη σκηνή (και αντίστοιχα αφαιρούνται από τη σκηνή). Πατώντας το κουμπί confirm, activeState ορίζεται το OrientationState.

- **OrientationState.** Δημιουργεί κατάλληλη λίστα επιλογών προκειμένου ο χρήστης να διαλέξει τον προσανατολισμό για τον οποίο θα καταδειχθεί η στήριξη του επιλεγμένου τεμαχίου. Σαν βοηθητικό στοιχείο για τον χρήστη εισέρχεται και το gameobject 'OrientationCube' στη σκηνή (βλ.Εικόνα 5.19).

Εικόνα 5.19 – OrientationState εναλλακτικές επιλογές προσανατολισμού

Αφού επιλεγεί ο επιθυμητός προσανατολισμός δημιουργείται το κουμπί της επιβεβαίωσης, επιλογή του οποίου ορίζει νέο activeState (βλ. Εικόνα 5.20).

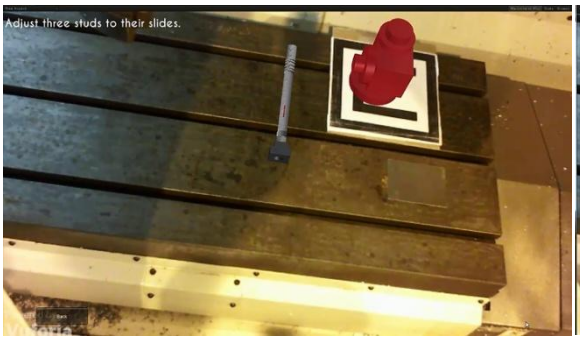
Εικόνα 5.20 – OrientationState επιβεβαίωση επιλογής

Το επόμενο activeState εξαρτάται από το συνδυασμό επιλογών τεμαχίου/ προσανατολισμού που έχει εισάγει ο χρήστης. Για απλούστευση εφόσον είναι δύο τα τεμάχια και έξι οι δυνατές επιλογές προσανατολισμού του καθενός έχει δημιουργηθεί η μεταβλητή scenario (public static int scenario) στο OrientationState που παίρνει τιμές 1 – 12 και χρησιμοποιείται από το βοηθητικό Stateμε όνομα SetupState (δεν επιφέρει καμιά μεταβολή στη σκηνή) το οποίο και ορίζει τελικά το επόμενο State στο οποίο θα δείχνει η μεταβλητή activeState. Τα εναλλακτικά States είναι:

- **SetupChuckState** Η μεταβλητή scenario έχει μια από τις τιμές 7, 10, 11 (προτείνεται χρήση τσωκ για συγκράτηση τεμαχίου)
- **SetupChuckStateTwo** Η μεταβλητή scenario έχει μια από τις τιμές 2, 8 (προτείνεται χρήση στοιχείων συσφίξεως για συγκράτηση του τεμαχίου)
- **ViseSetup** Η μεταβλητή scenario έχει μια από τις τιμές 1,3,4,5,6,9,12 (προτείνεται χρήση μέγγενης για συγκράτηση του τεμαχίου)

Για το πρώτο από τα τρία το αποτέλεσμα που επιφέρει στη σκηνή φαίνεται στην Εικόνα 5.21. Ως επόμενο activeState το διαδέχεται το SetupChuckStateTwo το οποίο δημιουργεί ένα κοχλία και έναν ολισθητήρα και υποδεικνύει τον τρόπο πρόσδεσης του πρώτου στο δεύτερο (βλ. Εικόνα 5.22). Το ίδιο State ενεργοποιείται και όταν δεν υπάρχει λόγος χρήσης τσωκ ή μέγγενης παρακάμπτοντας τη σειρά του SetupChuckState, με μόνη διαφοροποίηση το μήνυμα που εμφανίζει στο πάνω μέρος της οθόνης. Το State 'ViseSetup' έχει αντίστοιχη συμπεριφορά με εκείνη του 'SetupChuckState'και φαίνεται στην Εικόνα 5.23.

Εικόνα 5.21 – SetupChuckState



Εικόνα 5.22 – SetupChuckState

Εικόνα 5.23 – ViseSetupState

Τα τρια States που μόλις περιγράφησαν αποτελούν τα πρώτα σε μια ακολουθία States που ολοκληρώνουν την κατάδειξη της στήριξης του τεμαχίου ανάλογα με τις συσκευές που χρειάζονται (τσωκ, μέγγενη, ιδιοσυσκευή συσφίξεως).

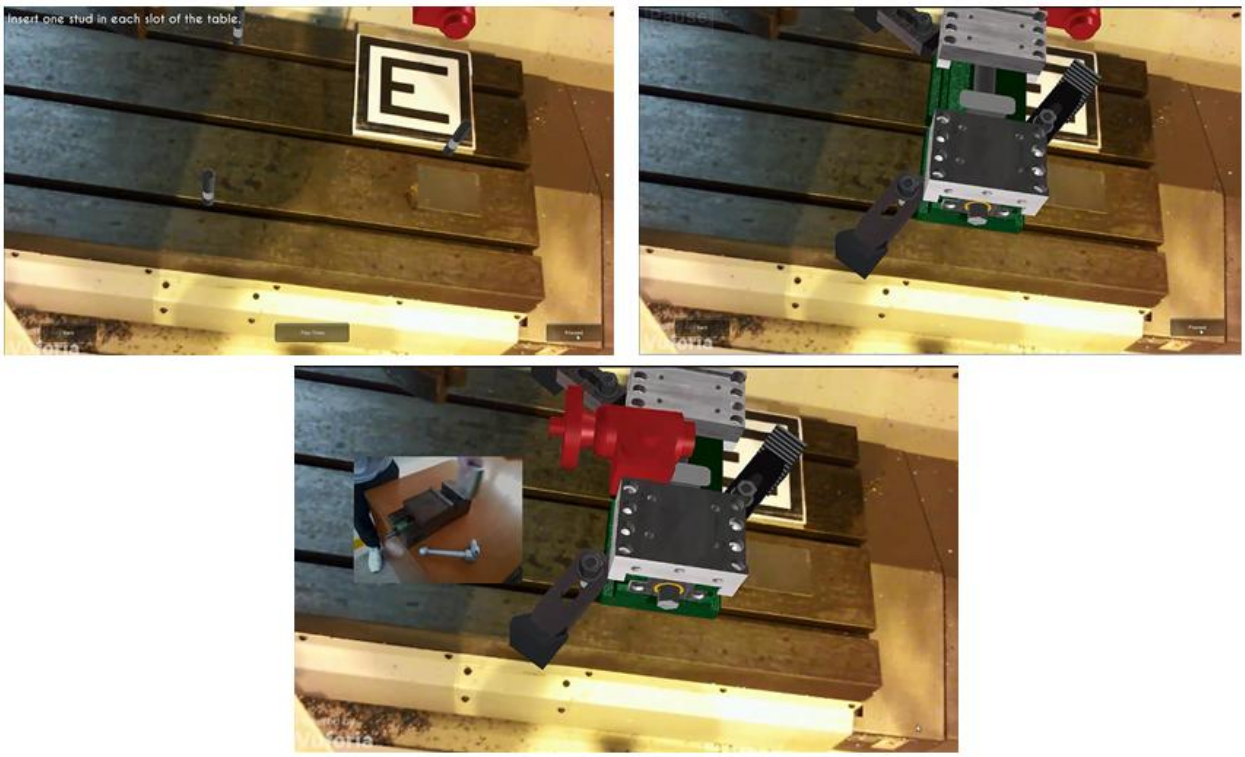
Έτσι σε συνέχεια του State '**SetupChuckState**' ακολουθούν κατά σειρά τα : **SetupChuckStateTwo, ChuckSetupThree, ChuckSetupFour , ChuckSetupFinal** (βλ.Εικόνα 5.24)

Σε συνέχεια του State '**SetupChuckStateTwo**' ακολουθούν κατά σειρά τα : **ChuckSetupThree, ClampsSetupFinal** (βλ.Εικόνα 5.25)

Σε συνέχεια του State '**ViseSetup**' ακολουθούν κατά σειρά τα : **SetupChuckStateTwo, ViseSetupThree, ViseSetupFour, ViseSetupFinal** (βλ.Εικόνα 5.26)

Εικόνα 5.24 – Κατάδειξη στήριξης με χρήση τσωκ

Εικόνα 5.25 – Κατάδειξη συγκράτησης με χρήση ιδιοσυσκευών συσφίξεως



Εικόνα 5.26 - Κατάδειξη συγκράτησης με χρήση μέγγενης

6. Συμπεράσματα – Προτάσεις για Βελτίωση

Η αξιοποίηση τεχνολογίας επαυξημένης πραγματικότητας σε περιβάλλον που άπτεται της ειδικότητας του μηχανολόγου μηχανικού είναι πολλά υποσχόμενη καθώς συνεχώς αίρονται περιορισμοί της, μέσω της συνεχούς ανάπτυξης νέων περιφερειακών συσκευών, αλλά και της βελτίωσης των ίδιων των περιβαλλόντων ανάπτυξης και εκτέλεσης των εφαρμογών,

Η συμβολή της παρούσας διπλωματικής εργασίας είναι:

Πρώτον, η δημιουργία δυο λειτουργικών εφαρμογών επαυξημένης πραγματικότητας που μπορούν να χρησιμοποιηθούν ως εργαλείο εκπαίδευσης φοιτητών και ανειδίκευτου προσωπικού για την προετοιμασία εργαλειομηχανών που βρίσκονται στο εργαστήριο του τομέα τεχνολογίας των κατεργασιών του ΕΜΠ, αλλά και ως βοήθημα τεκμηρίωσης της προετοιμασίας της εργαλειομηχανής στο τμήμα που αφορά τη στήριξη των τεμαχίων, σε αντικατάσταση σκαριφημάτων, σχεδίων ή και απλής λεκτικής καθοδήγησης από εργοδηγούς, που συμβαίνει συχνά στην πράξη.

Δεύτερον, η παράθεση των εργαλείων και των προγραμματιστικών πρακτικών που χρησιμοποιήθηκαν για την ανάπτυξή τους, μπορούν να αποτελέσουν ένα χρήσιμο οδηγό για τη δημιουργία αντίστοιχων εφαρμογών. Επιπλέον η δημιουργία States προκειμένου να αποδοθεί ο λειτουργικός χαρακτήρας και των δυο εφαρμογών επιτρέπει την εύκολη τροποποίηση και επέκτασή τους.

Τρίτον, η διασύνδεση των εφαρμογών με χρήση εξωτερικών αισθητήρων χαμηλού κόστους με τις μεθόδους που παρουσιάστηκαν στα κεφάλαια 3 και 4 βοηθά στη δημιουργία εφαρμογών αυξημένης διάδρασης με το περιβάλλον.

Μέσα από τη διαδικασία της ανάπτυξης και κατόπιν της χρήσης των εφαρμογών, αναδείχθηκαν κάποια προβλήματα και προτάσεις για βελτιωτικές παρεμβάσεις που αξίζει να διερευνηθούν.

Ένα από τα ζητήματα που προέκυψαν, είναι ότι οι συμβατικές κάμερες (web camera, κάμερα κινητού τηλεφώνου) έχουν περιορισμένο πεδίο – εύρος ορατότητας, γεγονός που καθιστά αναγκαία την αύξηση της απόστασης της κάμερας από το προβαλλόμενο αντικείμενο όσο μεγαλύτερος είναι ο όγκος του δεύτερου (ο όγκος ή η επιφάνεια που είναι επιθυμητό να προβληθεί στην συσκευή απεικόνισης). Οι εργαλειομηχανές είναι γενικά ογκώδεις διατάξεις και οι εφαρμογές που

αναπτύχθηκαν έχουν σκοπό ο χρήστης να μπορεί να δει σημαντικό τμήμα τους μέσω της συσκευής απεικόνισης ενώ ταυτόχρονα εργάζεται πάνω σε αυτές. Γι αυτό το λόγο έχει γίνει χρήση σταθερής κάμερας τοποθετημένης σε υπερυψωμένο σημείο και σαν συσκευή απεικόνισης έχει χρησιμοποιηθεί οθόνη υπολογιστή. Η χρήση των συγκεκριμένων συσκευών έχει σαν πλεονέκτημα ότι ο χρήστης είναι ελεύθερος να περιηγηθεί χωρίς να κρατά κανενός είδους εξοπλισμό ενώ το μεγάλο μέγεθος της οθόνης συγκριτικά με εκείνη ενός tablet ή ενός smartphone καθιστά την εικόνα ευδιάκριτη και από μεγαλύτερη απόσταση. Το αρνητικό είναι ότι η πλοήγηση στην εφαρμογή γίνεται με χρήση ποντικιού, αναγκάζοντας το χρήστη να εγκαταλείψει την εργασία του στην εργαλειομηχανή, προκειμένου να πάρει οδηγίες από την εφαρμογή. Μια βελτίωση των εφαρμογών λοιπόν, θα ήταν η χρήση αισθητήρα ή άλλης συσκευής που να επιτρέπει στο χρήστη να πλοηγηθεί στην εφαρμογή χωρίς να χρειαστεί να μετακινηθεί, ούτε και να χρησιμοποιήσει πληκρολόγιο ή ποντίκι.

Ένα άλλο ζήτημα που προέκυψε, που όμως αναμενόταν, είναι η σημαντική επίδραση των συνθηκών φωτισμού στην αναγνώριση του στόχου. Σε πολύ φωτεινό ή σκοτεινό χώρο η αναγνώριση του στόχου καθίσταται δύσκολη και σε κάποιες περιπτώσεις αδύνατη. Το πρόβλημα αυτό είναι δύσκολο να αντιμετωπιστεί με χρήση συμβατικής κάμερας και μόνο.

Η χρήση προκαθορισμένης εικόνας στόχου προς αναγνώριση, δημιουργεί κάποια εξάρτηση των εφαρμογέών από αυτή, περιορίζοντας τη γενικότητα του χαρακτήρα τους. Εάν, για παράδειγμα στη δεύτερη εφαρμογή εκτελούνταν αναγνώριση της γεωμετρίας της τράπεζας του κέντρου κατεργασιών, αντί της γεωμετρίας του στόχου, τότε η ίδια εφαρμογή θα μπορούσε να χρησιμοποιηθεί σε οποιοδήποτε κέντρο κατεργασιών που φέρει τράπεζα αντίστοιχης μορφής. Κατ' αναλογία, για την πρώτη εφαρμογή θα μπορούσαν σαν γεωμετρίες προς αναγνώριση να χρησιμοποιηθούν εξαρτήματα που αναμένεται να έχουν αντίστοιχη μορφή σε μεγάλη ποικιλία τόρνων, όπως για παράδειγμα τσωκ, εργαλειοδέτης, εργαλειοφορείο και άλλα.

Για την αξιοποίηση της δεύτερης εφαρμογής προτείνεται, η χρήση της από έναν αριθμό υποκειμένων, προκειμένου να αξιολογηθεί κατά πόσον οι πληροφορίες που παρέχει επαρκούν και είναι εύληπτες ούτως ώστε ο χρήστης να είναι σε θέση να ολοκληρώσει τις εργασίες που περιγράφονται από την εφαρμογή. Επιπρόσθετα, χρήσιμο θα ήταν να εκτιμηθεί κατά πόσον χρήση αυτής της εφαρμογής συνεισφέρει στην ταχύτερη εκπαίδευση ανειδίκευτου προσωπικού, ή αντίστοιχα μειώνει το χρόνο που δαπανά ειδικευμένο προσωπικό στην εκπαίδευση νέων χειριστών.

Βιβλιογραφία

[1] wikipedia. [Online]. https://en.wikipedia.org/wiki/Augmented_reality

[2] Patrick Dähne - John N.Karigiannis, "Archeoguide: System Architecture of a Mobile Outdoor," in *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR'02)*.

[3] Carl Engelking, "Archaeologists See and Smell the Past With Augmented Reality," *Discover Magazine*, Mar. 2015.

[4] augmentdev. [Online]. http://www.augmentedev.com/augmented-reality-architecture/

[5] Jeremy Ackerman. (2000, June) Department of Computer Science University of North Carolina. [Online]. http://cs.unc.edu/Research/us/

[6] Alan B. Craig William R. Sherman, "Boeing Wire Bundles, An Augmented Reality Application," in *Understanding Virtual Reality*.: MORGAN KAUFMANN PUBLISHERS, ch. APPENDIX C, pp. 512-520.

[7] Advanced Realtime Tracking. [Online]. http://www.ar-tracking.com/company/references/automotive-industry/audi-ingolstadt-germany/

[8] ARVIKA,Wolfgang Friedrich, "Augmented Reality for Development, Production and Service," pp. 3-6.

[9] S.K. Ong, A.Y.C Nee A.W.W Yew, "Towards a griddable distributed manufacturing system with augmented reality interfaces," *RoboticsandComputer-IntegratedManufacturing*, no. 39, pp. 43-55, 2016.

[10] D., Hillers, B., Gräser A Aiteanu, "A Step Forward in Manual Welding:," in *Proceedings of the Second IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR '03)*, 2003.

[11] Kevin M. Baird, "Augmented Reality," in *EVALUATING THE EFFECTIVENESS OF AUGMENTED REALITY AND WEARABLE COMPUTING FOR A MANUFACTURING ASSEMBLY TASK*.: Virginia Polytechnic Institute and State University, 1999, ch. 2.2, pp. 9-10.

[12] Henry Fuchs Jannick P. Rolland, "Optical Versus Video See-Through Head-Mounted Displays in Medical Visualization," *Presence*, vol. 9, no. 3, pp. 287-309, June 2000.

[13] D.W.F. van Krevelen and R. Poelman, "A Survey of Augmented Reality Technologies, Applications and Limitations," *The International Journal of Virtual Reality*, no. 2, pp. 1-20, Sep. 2010.

[14] Michele Fiorentino, Antonio E. Uva, Michele Gattullo ,Giuseppe Monno Michele Di Donato, "Text legibility for projected Augmented Reality on industrial workbenches," *Computers in Industry*, pp. 3-

4, Mar. 2015.

[15] Yohan Baillot,Reinhold Behringer,Steven Feiner,Simon Julier,Blair MacIntyre Ronald Azuma, "Recent Advances in Augmented Reality," *Computers & Graphics*, November 2001.

[16] (2016, Jan.) Unity (game engine). [Online]. https://en.wikipedia.org/wiki/Unity_%28game_engine%29

[17] (2016, Jan.) unity|DOCUMENTATION. [Online]. http://docs.unity3d.com/Manual/UsingComponents.html

[18] (2016, Jan.) unity|DOCUMENTATION. [Online]. http://docs.unity3d.com/Manual/GameObjects.html

[19] ΕΡΓΑΣΤΗΡΙΟ ΑΥΤΟΜΑΤΟΥ ΕΛΕΓΧΟΥ - ΕΜΠ. (2016, Feb.) Συστήματα Ελέγχου με μικρο-Υπολογιστές. [Online]. http://courseware.mech.ntua.gr/ml23259/index.php?page=lessons

[20] (2016, Feb.) Pulse-width modulation Wikipedia. [Online]. https://en.wikipedia.org/wiki/Pulse-width_modulation

[21] Cytron Technologies Sdn. Bhd., "Product User's Manual – HC-SR04 Ultrasonic Sensor," 2013.

[22] Terry Norton, *Learning C# by Developing Games with Unity 3D*. BIRMINGHAM, MUMBAI: PACKT, 2013.

Παράρτημα Α – Κωδικας Πρώτης Εφαρμογής

Component scripts - Classes

- ActivationController

```
using UnityEngine;
using System.Collections;

// This class changes the visibility status of
the lathe and its parts
// The method used for this purpose is:
// ChangeVisibilityOfLathe()
public class ActivationController : MonoBehaviour
{
  // This script is a component of tornos

  private MeshRenderer spindleActivator;
  private MeshRenderer [] tailstockActivator;
  // Use this for initialization
  void Start () {
    spindleActivator = GameObject.FindWithTag
("spindle").GetComponent<MeshRenderer> ();
    tailstockActivator = new MeshRenderer[] {
GameObject.FindWithTag
("tailstock").GetComponent<MeshRenderer> (),
      GameObject.FindWithTag
("supportExtended").GetComponent<MeshRenderer>
(),
      GameObject.FindWithTag
("motor").GetComponent<MeshRenderer> ()};
}

  public void ChangeVisibilityOfLathe(){
    foreach(MeshRenderer tailstock in
tailstockActivator)
    {
      tailstock.enabled =!tailstock.enabled;

    }
    spindleActivator.enabled
=!spindleActivator.enabled;
  }
}
```

- arduinoSensor

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;
using System.IO.Ports;
using System;
using Assets.Codes.Scripts;

namespace diplwmatiki   {
public class arduinoSensor : MonoBehaviour {

  SerialPort sp = new SerialPort("COM3",9600);
  string input;
  public int height;



  // Use this for initialization
  void Start () {
    sp.Open ();
    sp.ReadTimeout = 1;

  }


  void Update () {
    if (sp.IsOpen)
    {
      try
      {
        input =sp.ReadLine();
          if (InputSecurity.CheckInt(input))
height = Int32.Parse(input);
      }
      catch(System.Exception)
      {
      }
    }
  }
}
```

- GUIManager

```
using UnityEngine;
using System.Collections;

// This class is a component of Initial Screen
public class GUIManager : MonoBehaviour
{
  public bool targetRecognized;
  GUIText textComponentToBlink;
  string initialText;
  GameObject guiTransform;
  int counter;
  ActivationController hideLathe;

  void Start(){
    textComponentToBlink = GameObject.FindWithTag
("guiText").GetComponent<GUIText> ();
    hideLathe = GameObject.FindWithTag
("tornos").GetComponent<ActivationController> ();
    guiTransform =GameObject.FindWithTag
("guiText");
    initialText = textComponentToBlink.text;
    targetRecognized = false;
    StartCoroutine (waitText());
    counter = 1;
    }
  // This method is executed every frame
  void OnGUI()
  {
    if (targetRecognized)
    {
      StopCoroutine(waitText());
    /*
      Once the target is recognized the text and
its attributes change,
      the lathe becomes invisible
       */
      if (counter == 1 )
      {
      Vector3 initialPosition =
guiTransform.transform.localPosition;
      guiTransform.transform.localPosition=new
Vector3(initialPosition.x+0.23f,initialPosition.y
,initialPosition.z);
        counter++;
        hideLathe.ChangeVisibilityOfLathe();
      }
      textComponentToBlink.color =Color.cyan;
      textComponentToBlink.anchor =
TextAnchor.UpperCenter;
      textComponentToBlink.text="Target
Recognized !";
      GUIStyle style = GUI.skin.GetStyle
("button");

      //Set the style font size to increase and
decrease over time
      style.fontSize = 20;
        if (GUI.Button (new Rect ((Screen.width -
100) / 2, Screen.height - 120, 200, 100), "Press
to continue"))
        {
          print ("You clicked the button!");
          GameObject.FindWithTag
("guitexture").SetActive (false);

  GameObject.FindWithTag("manager").GetComponent<
GUIText>().enabled = true;

  GameObject.FindWithTag("manager").GetComponent<
MainManager>().enabled = true;
        }
      }
    }
  /* Coroutine that makes the initial text do a
dot blinking effect
   */
```

```
            IEnumerator waitText()
            {
              while(!targetRecognized)
              {
                string temp =initialText;
                for (int i=1;i<= 5;i++)
                {
                  if(!targetRecognized)
                  {
                    textComponentToBlink.text=temp;
                    temp = temp +" .";
                  }
                  yield return new WaitForSeconds(.5f);
                }
              }
            }
          }
```

- InputSecurity

```
using UnityEngine;
using System.Collections;

namespace Assets.Codes.Scripts
{
public class InputSecurity
  {
    // This mathod checks if user input is an int
    public static bool CheckInt(string input)
    {
      char [] myArray = input.ToCharArray();
      foreach( char character in myArray)
      {
        if (character<'0'||character>'9') return false;
      }
      return true;
    }

    // This method checks if user input is a decimal number
    public static bool CheckDecimal(string input)
    {
      int commaCounter = 0;
      char[] myArray = input.ToCharArray();
      foreach (char character in myArray)
      {
        if (character < '0' || character > '9')
        {
          if ((character != '.')||(commaCounter>=1))
          {
            return false;
          }
          else
          {
            commaCounter++;
          }
        }
      }
      return true;
    }
  }
}
```

- MainManager

```
using UnityEngine;
using System.Collections;
using Assets.Codes.Interface;

public class MainManager : MonoBehaviour
{
  int stateCounter; // the state counter will be used to
shift between states
  IState activeState;
  IState [] everyState;
  // Use this for initialization
  void Start ()
  {

    everyState = new IState[] {new InitialState(),new
ScaleState(),new ScaleStateTwo(),
      new SetInSpindleState(),new SetInTailstockState(),new
CorrectionState()};
    stateCounter = 0;
    activeState = everyState [stateCounter];
  }

  // Update is called once per frame
  void Update ()
  {
    activeState.Action ();
  }
  void OnGUI()
  {
    activeState.GUIImage();

    // Main manager GUI controls and positioning
    if (GUI.Button (new Rect (100, Screen.height - 100,
150, 50), "Previous"))
    {
      activeState.Reset();
      activeState = everyState[stateCounter-1];
      stateCounter--;
    }
    if (GUI.Button (new Rect (Screen.width - 200,
Screen.height - 100, 150, 50), "Next"))
    {
      activeState = everyState[stateCounter +1];
      stateCounter++;
    }
  }
}
```

- MotionDokimio

```
using UnityEngine;
using System.Collections;

public class MotionDokimio : MonoBehaviour {

  public bool isSetInSpindler;
  public float speed ;

  void Start()
  {
    isSetInSpindler = false;
    speed = 40.0f;
    }
  // This method will be called by the SetInSpindleState
  public void MoveFromAnotherScript()
  {
    StartCoroutine ("MoveToSpindle");
    }

  // Basic Motion Methods of the testing part
  void MoveLeft(){

    transform.Translate (Vector3.down *
Time.deltaTime*speed);
  }
  void MoveRight(){
    transform.Translate (Vector3.up *
Time.deltaTime*speed);
  }

  void MoveDown()
  {
    transform.Translate (Vector3.right * Time.deltaTime *
3.0f * speed);
  }
  void MoveUp()
  {
    transform.Translate (Vector3.left * Time.deltaTime *
3.0f * speed);
  }
  /* This Coroutine moves the testing part in the Spindler
   *
   *
   */
  IEnumerator MoveToSpindle()
  {
    yield return new WaitForSeconds (0.5f);
    while(!isSetInSpindler)
    {

            MoveRight ();
      yield return null;
        }
    }

  // Collision detection part
  void OnTriggerEnter(Collider other)
  {

    if (other.gameObject.CompareTag ("spindle"))
    {
      Debug.Log ("Collision Detected");
      isSetInSpindler = true;
    }
  }
}
```

- ScalePart

```
using UnityEngine;
using System.Collections;
using diplwmatiki;

public class ScalePart : MonoBehaviour {

  arduinoSensor dimension;
  // Use this for initialization

  void Start () {
    dimension = GameObject.FindWithTag
("height").GetComponent<arduinoSensor> ();
  }

  public void GetSensorValue()
```

```
  {
    float h = (float) dimension.height;
    h = h/40.0f;
    Debug.Log(h);
    Vector3 temp = transform.localScale;
    temp.y = h;
    transform.localScale =temp;

  }
  public void ManuallySetHeight(int height)
  {
    float h = (float) height;
    h = h/40.0f;
    Vector3 temp = transform.localScale;
    temp.y = h;
    transform.localScale =temp;
  }
}
```

- TailstockMotion

```
using UnityEngine;
using System.Collections;

public class TailstockMotion : MonoBehaviour {
  public bool dokimioInPlace;
  public float speed ;

  void Start()
  {
    dokimioInPlace = false;
    speed = 40.0f;
  }

  public void MoveTailstockFromExternalScript()
  {
    StartCoroutine("MoveToDokimio");
    }
  // Basic Motion Methods of the testing part
  private void MoveLeft(){
    transform.Translate (Vector3.down *
Time.deltaTime*speed);
  }
  private void MoveRight(){
    transform.Translate (Vector3.up *
Time.deltaTime*speed);
  }
  /* This Coroutine moves the testing part in the Spindler
   */
  IEnumerator MoveToDokimio()
  {
    yield return new WaitForSeconds (1.0f);
    while(!dokimioInPlace)
    {

      MoveRight ();
      yield return null;
    }
  }

  // Collision detection part
  void OnTriggerEnter(Collider other)
  {

    if (other.gameObject.CompareTag ("dokimio"))
    {
      Debug.Log ("Collision Detected");
      dokimioInPlace = true;
    }
  }
}
```

- TextureAssignment

```
using UnityEngine;
using System.Collections;

public class TextureAssignment : MonoBehaviour
{
  public Texture2D SensorTexture;
  public Texture2D linuxCNC;
}
```

States

- CorrectionState

```
using UnityEngine;
using System.Collections;
using Assets.Codes.Interface;

public class CorrectionState : IState
{
  public void Action(){}
  private string ShowText()
  {
    string displayText = "Make sure that the part is
properly centered.\r\n" +
      "\r\nThe next step is to setup the origin of the
axes.";
    return displayText;
  }
  public void GUIImage()
  {
    Rect textHolder = new Rect(300,Screen.height-
250,Screen.width-600,150);
    GUIStyle boxStyle = "box";
    boxStyle.fontSize = 25;
    boxStyle.alignment = TextAnchor.UpperLeft;
    boxStyle.wordWrap = true;
    GUI.Box(textHolder,ShowText());
  }
  public void Reset(){}
}
```

- InitialState

```
using UnityEngine;
using System.Collections;
using Assets.Codes.Interface;

public class InitialState :IState
{
  public void Action(){}
  private string ShowText()
  {
    string displayText = "This is an augmented reality
application that will guide you to setup a cylinder in the
EMCO lathe.";
    return displayText;
  }
  public void GUIImage()
  {
    Rect textHolder = new Rect(300,Screen.height-
250,Screen.width-600,150);
    GUIStyle boxStyle = "box";
    boxStyle.fontSize = 25;
    boxStyle.alignment = TextAnchor.UpperLeft;
    boxStyle.wordWrap = true;
    GUI.Box(textHolder,ShowText());
  }
  public void Reset(){}
}
```

- ScaleState

```
using UnityEngine;
using System.Collections;
using Assets.Codes.Interface;

public class ScaleState : IState {
  private Texture2D sensor;

  public ScaleState()
  {
    sensor = GameObject.FindWithTag
("imageComponent").GetComponent<TextureAssignment>
().SensorTexture;
    }
  public void Action(){}
  public string ShowText()
  {
    string displayText = "First you need to scale your part
by using the ultrasonic sensor.\r\n" +
      "\r\nPlace the part in front of the sensor and press
the 'Next' button.";
    return displayText;
  }
  public void GUIImage()
  {
    //GUI.Box(new Rect(10, 10, 400, 200), sensor);
    GUI.Label(new Rect(10, 10, sensor.width,
(int)(sensor.height/1.2f)), sensor);
    Rect textHolder = new Rect(300,Screen.height-
250,Screen.width-600,150);
    GUIStyle boxStyle = "box";
    boxStyle.fontSize = 25;
    boxStyle.alignment = TextAnchor.UpperLeft;
    boxStyle.wordWrap = true;
    GUI.Box(textHolder,ShowText());
  }
  public void Reset(){}
}
```

- ScaleStateTwo

```
using UnityEngine;
using System.Collections;
using Assets.Codes.Interface;
using Assets.Codes.Scripts;
using diplwmatiki;
using System;

public class ScaleStateTwo : IState {

  int val;
  string message;
  string userInput;
```

```
  public ScaleStateTwo()
  {
    val = 0;
    message = "";
    userInput = "";
    /*GUIStyle TextFieldStyle = new GUIStyle ();
    TextFieldStyle.fontSize = 20;
    TextFieldStyle.normal.background =
GameObject.FindWithTag
("imageComponent").GetComponent<TextureAssignment>
().textTexture;
    */
  }
  public void Action(){}

  public void Reset(){}

  public void GUIImage()
  { // GUI.Button (new Rect ((int)(Screen.width / 2) - 200,
Screen.height - 250, 400, 50), "Get dimension from sensor"
    if (GUI.Button (new Rect ((int)(Screen.width / 2) -
200, Screen.height - 300, 400, 50), "Get dimension from
sensor"))
    {


  val=GameObject.FindWithTag("height").GetComponent<arduino
Sensor>().height;
      if(val>=20)
      {
        message="\r\nToo large part for this lathe!";
      }
      else
      {

  GameObject.FindWithTag("dokimio").GetComponent<ScalePart>
().GetSensorValue();
        message ="";
      }
    }
    // Text position and style for the axial dimension box
    Rect textHolder = new Rect ((int)(Screen.width / 2)-
200, Screen.height - 240, 400, 80);
    GUIStyle boxStyle = "box";
    boxStyle.fontSize = 25;
    boxStyle.alignment = TextAnchor.MiddleCenter;
    boxStyle.wordWrap = true;
    GUI.Box(textHolder,"The axial dimension is:  "+val+"
cm"+message);

    // Text position and style for the manual prompt
message
    boxStyle.fontSize = 20;
    boxStyle.fontStyle = FontStyle.Italic;
    Rect textHolderManual = new Rect ((int)(Screen.width /
2) - 250, Screen.height - 150, (int)(Screen.width / 4) +30,
50);
    GUI.Box (textHolderManual,"Skip the sensor, and give
your value for the part: ");

    // Get user input as a string of five characters
    /*boxStyle.fontSize = 20;
    boxStyle.normal.background = GameObject.FindWithTag
("imageComponent").GetComponent<TextureAssignment>
().textTexture;
    boxStyle.alignment = TextAnchor.MiddleCenter;
    boxStyle.wordWrap = true;*/
    userInput = GUI.TextField(new Rect
((int)(Screen.width/2)-
250+(int)(Screen.width/4)+35,Screen.height-
150,150,50),userInput,5);
    if (GUI.Button (new Rect ((int)(Screen.width / 2) - 250
+ (int)(Screen.width / 4) + 190, Screen.height - 150, 100,
50), "Confirm"))
    {
      // Parsing the string
      // We could use the InputSecurity class to also
handle floating point input
      int temp ;
      bool result = Int32.TryParse(userInput, out temp);
      if (result)
      {
        val = temp;
        if(val>=20)
        {
          message="\r\nToo large part for this lathe!";
        }
        else
        {

  GameObject.FindWithTag("dokimio").GetComponent<ScalePart>
().ManuallySetHeight(temp);
          message ="";
        }
      }
      else
      {
        val = 0;
        message = "\r\nInvalid Input";
      }
    }
  }
}
```

- SetInSpindleState

```
using UnityEngine;
using System.Collections;
using Assets.Codes.Interface;

public class SetInSpindleState :IState
{
  private bool showLathe;
  private Vector3 initialPosition;
  private Transform dokimio;
  private int count;

  public SetInSpindleState()
  {
    showLathe = false;
    count = 1;
    initialPosition = GameObject.FindWithTag
("dokimio").GetComponent<Transform>
().transform.localPosition;
  }
  public void Action()
  {
    if (!showLathe)
    {
      GameObject.FindWithTag
("tornos").GetComponent<ActivationController>
().ChangeVisibilityOfLathe ();
      showLathe=true;
    }
    if (count ==1)
    {
      GameObject.FindWithTag
("dokimio").GetComponent<MotionDokimio>
().MoveFromAnotherScript ();
      count++;
    }
  }
  private string ShowText()
  {
    string displayText = "Place the part in the center of
the spindle.";
    return displayText;
  }
  public void GUIImage()
  {
    Rect textHolder = new Rect(300,Screen.height-
250,Screen.width-600,150);
    GUIStyle boxStyle = "box";
    boxStyle.fontSize = 25;
    boxStyle.alignment = TextAnchor.UpperLeft;
    boxStyle.wordWrap = true;
    GUI.Box(textHolder,ShowText());

    if (GUI.Button (new Rect ((int)(Screen.width / 2) -
100, Screen.height - 305, 200, 50), "Repeat Motion"))
          RepeatMotion ();

  }
  private void ResetPosition()
  {
    GameObject.FindWithTag
("dokimio").GetComponent<Transform>
().transform.localPosition = initialPosition;
  }
  private void RepeatMotion()
  {
    ResetPosition ();
    GameObject.FindWithTag
("dokimio").GetComponent<MotionDokimio> ().isSetInSpindler
= false;
    GameObject.FindWithTag
("dokimio").GetComponent<MotionDokimio> ().speed = 35.0f;
    GameObject.FindWithTag
("dokimio").GetComponent<MotionDokimio>
().MoveFromAnotherScript ();
  }
  public void Reset()
  {
    ResetPosition ();
    GameObject.FindWithTag
("dokimio").GetComponent<MotionDokimio> ().isSetInSpindler
= false;
    GameObject.FindWithTag
("tornos").GetComponent<ActivationController>
().ChangeVisibilityOfLathe ();
    showLathe = false;
```

```
    count = 1;
  }
}
```

- SetInTailstockState

```
using UnityEngine;
using System.Collections;
using Assets.Codes.Interface;

public class SetInTailstockState : IState
{
  private Texture2D imageToShow;
  private Vector3 initialPosition;
  private Transform tailstock;
  private int count;

  public SetInTailstockState()
  {
    count = 1;
    initialPosition = GameObject.FindWithTag
("tailstock").GetComponent<Transform>
().transform.localPosition;
    imageToShow = GameObject.FindWithTag
("imageComponent").GetComponent<TextureAssignment>
().linuxCNC;
    tailstock = GameObject.FindWithTag
("tailstock").GetComponent<Transform> ();
  }
  public void Action()
  {
    if (count ==1)
    {
      GameObject.FindWithTag
("tailstock").GetComponent<TailstockMotion>().MoveTailstock
FromExternalScript();
      count++;
    }
  }

  private void ResetPosition()
  {
    tailstock.transform.localPosition = initialPosition;
  }

  public void Reset()
  {
    ResetPosition ();
    GameObject.FindWithTag
("tailstock").GetComponent<TailstockMotion>
().dokimioInPlace = false;
    count = 1;
  }

  private void RepeatMotion()
  {
    ResetPosition ();
    GameObject.FindWithTag
("tailstock").GetComponent<TailstockMotion>
().dokimioInPlace = false;
    GameObject.FindWithTag
("tailstock").GetComponent<TailstockMotion> ().speed =
35.0f;
    GameObject.FindWithTag
("tailstock").GetComponent<TailstockMotion>().MoveTailstock
FromExternalScript();
  }
  private string ShowText()
  {
    string displayText = "You need to press the button
bellow from the PC to activate the tailstock";
    return displayText;
  }
  public void GUIImage(){
    //Image that will be shown and its positioning
    GUI.Label(new Rect(Screen.width-
(int)(imageToShow.width/2.0f)-10, 10,
(int)(imageToShow.width/2.0f),
(int)(imageToShow.height/2.0f)), imageToShow);

    // Text that will be shown , its positioning and style
    Rect textHolder = new Rect(300,Screen.height-
250,Screen.width-600,150);
    GUIStyle boxStyle = "box";
    boxStyle.fontSize = 25;
    boxStyle.alignment = TextAnchor.UpperLeft;
    boxStyle.wordWrap = true;
    GUI.Box(textHolder,ShowText());

    if (GUI.Button (new Rect ((int)(Screen.width / 2) -
100, Screen.height - 305, 200, 50), "Repeat Motion"))
      RepeatMotion ();
  }
}
```

Παραρτημα Β – Κώδικας Δεύτερης Εφαρμογής

Component scripts

- MainManager

```
using UnityEngine;
using System.Collections;
using Assets.Codes.Interface;
using Assets.Codes.States;

namespace Assets.Codes.ComponentScripts
{
  public class MainManager : MonoBehaviour
  {
    public GameObject [] prefab;
    public GUISkin guiSkin;
    public GUISkin defaultGuiSkin;
    public Texture2D[] image;

    private IState activeState;

    void Start ()
    {
      activeState = new InitialScreenState (this);
    }

  // Update is called once per frame
    void Update ()
    {
      activeState.Action ();
    }
    void OnGUI()
    {
    activeState.GUIImage();
    }
    public void SwitchState(IState nextState)
    {
      activeState = nextState;
    }
    public GameObject CreatePrefab(int param)
    {
      GameObject aPrefab;
      aPrefab =
Instantiate(prefab[param],transform.position,Quaternion.ide
ntity) as GameObject;
      aPrefab.transform.parent = GameObject.FindWithTag
("imageTarget").transform;
      return aPrefab;
    }
    // overloading the Create Prefab method
    public GameObject CreatePrefab(int param, Vector3
positionParam)
    {
      GameObject aPrefab;
      aPrefab =
Instantiate(prefab[param],positionParam,Quaternion.identity
) as GameObject;
      aPrefab.transform.parent = GameObject.FindWithTag
("imageTarget").transform;
      return aPrefab;
    }
    public GameObject CreatePrefab(int param, Vector3
positionParam, Vector3 orientationParam)
    {
      GameObject aPrefab;
      aPrefab =
Instantiate(prefab[param],positionParam,Quaternion.identity
) as GameObject;
      aPrefab.transform.localEulerAngles =
orientationParam;
      aPrefab.transform.parent = GameObject.FindWithTag
("imageTarget").transform;
      return aPrefab;
    }
    public GameObject CreatePrefab(int arrayParam, Vector3
positionParam, string parentObjectName)
    {
      GameObject aPrefab;
      aPrefab =
Instantiate(prefab[arrayParam],transform.position,Quaternio
n.identity) as GameObject;
      aPrefab.transform.parent = GameObject.Find
(parentObjectName).transform;
      aPrefab.transform.localPosition = positionParam;
      return aPrefab;
    }
    public GameObject CreatePrefab(int arrayParam, Vector3
positionParam,Vector3 orientationParam, string
parentObjectName)
    {
      GameObject aPrefab;
      aPrefab =
Instantiate(prefab[arrayParam],transform.position,Quaternio
n.identity) as GameObject;
      aPrefab.transform.localEulerAngles =
orientationParam;
      aPrefab.transform.parent = GameObject.Find
(parentObjectName).transform;
      aPrefab.transform.localPosition = positionParam;
      return aPrefab;
    }
    public void DestroyPrefab(GameObject prefabName)
    {
      UnityEngine.GameObject.Destroy (prefabName);
    }
  }
}
```

- Assembly Step Block Mover

```
using UnityEngine;
using System.Collections;
using Assets.Codes.Interface;
using Assets.Codes.States;

namespace Assets.Codes.ComponentScripts
{
  public class MainManager : MonoBehaviour
  {
    public GameObject [] prefab;
    public GUISkin guiSkin;
    public GUISkin defaultGuiSkin;
    public Texture2D[] image;

    private IState activeState;

    void Start ()
    {
      activeState = new InitialScreenState (this);
    }

  // Update is called once per frame
    void Update ()
    {
      activeState.Action ();
    }
    void OnGUI()
    {
    activeState.GUIImage();
    }
    public void SwitchState(IState nextState)
    {
      activeState = nextState;
    }
    public GameObject CreatePrefab(int param)
    {
      GameObject aPrefab;
      aPrefab =
Instantiate(prefab[param],transform.position,Quaternion.ide
ntity) as GameObject;
```

```
      aPrefab.transform.parent = GameObject.FindWithTag
("imageTarget").transform;
      return aPrefab;
    }
    // overloading the Create Prefab method
    public GameObject CreatePrefab(int param, Vector3
positionParam)
    {
      GameObject aPrefab;
      aPrefab =
Instantiate(prefab[param],positionParam,Quaternion.identity
) as GameObject;
      aPrefab.transform.parent = GameObject.FindWithTag
("imageTarget").transform;
      return aPrefab;
    }
    public GameObject CreatePrefab(int param, Vector3
positionParam, Vector3 orientationParam)
    {
      GameObject aPrefab;
      aPrefab =
Instantiate(prefab[param],positionParam,Quaternion.identity
) as GameObject;
      aPrefab.transform.localEulerAngles =
orientationParam;
      aPrefab.transform.parent = GameObject.FindWithTag
("imageTarget").transform;
      return aPrefab;
    }
    public GameObject CreatePrefab(int arrayParam, Vector3
positionParam, string parentObjectName)
    {
      GameObject aPrefab;
      aPrefab =
Instantiate(prefab[arrayParam],transform.position,Quaternio
n.identity) as GameObject;
      aPrefab.transform.parent = GameObject.Find
(parentObjectName).transform;
      aPrefab.transform.localPosition = positionParam;
      return aPrefab;
    }
    public GameObject CreatePrefab(int arrayParam, Vector3
positionParam,Vector3 orientationParam, string
parentObjectName)
    {
      GameObject aPrefab;
      aPrefab =
Instantiate(prefab[arrayParam],transform.position,Quaternio
n.identity) as GameObject;
      aPrefab.transform.localEulerAngles =
orientationParam;
      aPrefab.transform.parent = GameObject.Find
(parentObjectName).transform;
      aPrefab.transform.localPosition = positionParam;
      return aPrefab;
    }
    public void DestroyPrefab(GameObject prefabName)
    {
      UnityEngine.GameObject.Destroy (prefabName);
    }
  }
}
```

• Bolt Mover

```
using UnityEngine;
using System.Collections;

public class BoltMover : MonoBehaviour {

  public float speed;
  // Use this for initialization
  void Start () {
    speed = 50.0f;
  }

  // Update is called once per frame
  private void Update () {
    if (transform.localPosition.z >= (-33.0f))
    {
      speed = 100.0f;
      MoveDown();
    }
    else if (transform.localPosition.z >= (-55.0f))
    {
      speed = 50.0f;
      Screw ();
    }
    else
    {

  GameObject.FindWithTag("DemoSlideStud").GetComponent<Demo
Status>().demoFinished = true;
    }
  }

  public void MoveDown()
  {
    transform.localPosition -=new
Vector3(transform.localPosition.x
,transform.localPosition.y,(speed * Time.deltaTime)/8.0f);
  }
  public void Screw()
  {
    MoveDown ();
    transform.eulerAngles +=new Vector3(0.0f ,0.0f,speed *
Time.deltaTime*20.0f);
  }
}
```

• Chuck Mover

```
using UnityEngine;
using System.Collections;

public class ChuckMover : MonoBehaviour {

  public float speed;
  public bool manualOperation;
  // Use this for initialization
  void Start ()
  {
    speed = 10.0f;
    manualOperation = false;
  }

  // Update is called once per frame
  void Update ()
  {
    if(manualOperation)
    {
      if (Input.GetKey (KeyCode.RightArrow))
      {
        OpenJaws();
      }
      if (Input.GetKey (KeyCode.LeftArrow))
      {
        CloseJaws();
      }
    }
  }
  public void CloseJaws()
  {
    GameObject.FindWithTag ("jaw1").transform.Translate
(new Vector3 (-speed*Time.deltaTime, 0, 0));
    GameObject.FindWithTag ("jaw2").transform.Translate
(new Vector3 (-speed*Time.deltaTime, 0, 0));
    GameObject.FindWithTag ("jaw3").transform.Translate
(new Vector3 (-speed*Time.deltaTime, 0, 0));
  }
  public void OpenJaws()
  {
    GameObject.FindWithTag ("jaw1").transform.Translate
(new Vector3 (speed*Time.deltaTime, 0, 0));
    GameObject.FindWithTag ("jaw2").transform.Translate
(new Vector3 (speed*Time.deltaTime, 0, 0));
    GameObject.FindWithTag ("jaw3").transform.Translate
(new Vector3 (speed*Time.deltaTime, 0, 0));
  }
  public void SetPositionOfJaws(float xvalue)
  {
    GameObject localObject;
    for (int i=1; i<3; i++)
    {
      localObject =GameObject.FindWithTag ("jaw" +i);
      localObject.transform.localPosition = new
Vector3(xvalue,localObject.transform.localPosition.y,localO
bject.transform.localPosition.z);
    }
  }
}
```

• Chuck Video Observer

```
using UnityEngine;
using System.Collections;
using Assets.Codes.States;

public class chuckVideoObserver : MonoBehaviour {

  private bool videoFinished;
  // Use this for initialization
  void Start ()
  {
    videoFinished = false;
  }

  // Update is called once per frame
  void Update ()
  {
    videoFinished = GetComponent<VideoHolder>().finished;
    ChuckSetupFinal.videoFinished = videoFinished;
  }
```

```
}
```

- ClampMover

```
using UnityEngine;
using System.Collections;

public class ClampMover : MonoBehaviour {

  private float speed;
  // Use this for initialization
  void Start () {
    speed = 40.0f;
  }
  // Update is called once per frame
  void Update () {}

  public void MoveDown()
  {
    transform.localPosition-=new
Vector3(0,0,speed*Time.deltaTime);
  }
  public void MoveUp()
  {
    transform.localPosition+=new
Vector3(0,0,speed*Time.deltaTime);
  }
  public void RotateCW()
  {
    transform.localEulerAngles += new Vector3 (0, 0, 5.0f *
speed * Time.deltaTime);
  }
  public void RotateACW()
  {
    transform.localEulerAngles -= new Vector3 (0, 0, 5.0f *
speed * Time.deltaTime);
  }
}
```

- Demo Status

```
using UnityEngine;
using System.Collections;

public class DemoStatus : MonoBehaviour {

  public bool demoFinished;
  // Use this for initialization
  void Start () {
    demoFinished = false;
  }

}
```

- Image Target Watcher

```
using UnityEngine;
using System.Collections;

public class ImageTargetWatcher : MonoBehaviour
{
  public bool targetFound;
  // Use this for initialization
  void Start () {
    targetFound = false;
  }
}
```

- Mouse Listener

```
using UnityEngine;
using System.Collections;

public class MouseListener : MonoBehaviour {

  public bool mouseToggleSwitch;
  public int numberOfClicks;
  // Use this for initialization
  void Start ()
  {
    mouseToggleSwitch = false;
    numberOfClicks = 0;
  }

  // Update is called once per frame
  void Update () {

  }
  void OnMouseDown()
  {
    mouseToggleSwitch = !mouseToggleSwitch;
    numberOfClicks++;
    Debug.Log ("It has been clicked: "+numberOfClicks+"
times");
  }
}
```

- Moving Kaloupi

```
using UnityEngine;
using System.Collections;

public class MovingKaloupi : MonoBehaviour {

  private Vector3 initialPosition;
  private Vector3 initialOrientation;
  private Transform objectTransform;
  private Vector3 [] planeChoice =new Vector3[6];
  public bool manualOrientationSet;
  private float speed;

  // Use this for initialization
  void Start ()
  {
    speed = .8f;
    objectTransform = GetComponent<Transform> ();
    initialPosition =
objectTransform.transform.localPosition;
    initialOrientation =
objectTransform.transform.localEulerAngles;
    planeChoice [0] = initialOrientation;
    planeChoice [1] = new Vector3 (90f, 0, 0);
    planeChoice [2] = new Vector3 (180f, 0, 0);
    planeChoice [3] = new Vector3 (-90f, 0, 0);
    planeChoice [4] = new Vector3 (0, 0, 90f);
    planeChoice [5] = new Vector3(0,0,-90f);
    manualOrientationSet = false;
  }

  // Update is called once per frame
  void Update () {

    if (manualOrientationSet)
    {
      if (Input.GetKeyDown (KeyCode.Space)) {
          RotateToPlane (1);
      }
      if (Input.GetKeyDown (KeyCode.LeftArrow)) {
          RotateToPlane (2);
      }
      if (Input.GetKeyDown (KeyCode.RightArrow)) {
          RotateToPlane (3);
      }
      if (Input.GetKeyDown (KeyCode.UpArrow)) {
          RotateToPlane (4);
      }
      if (Input.GetKeyDown (KeyCode.DownArrow)) {
          RotateToPlane (5);
      }
      if (Input.GetKeyDown (KeyCode.Return)) {
          RotateToPlane (6);
      }
    }
  }

  // This method is used to orient the gameObject as the
user wants
  public void RotateToPlane(int caseSwitch)
  {
    switch (caseSwitch)
    {
    case 6:
      objectTransform.transform.localEulerAngles =
planeChoice[0];
      break;
    case 3:
      objectTransform.transform.localEulerAngles =
planeChoice[1];
      break;
    case 2:
      objectTransform.transform.localEulerAngles =
planeChoice[2];
      break;
    case 5:
      objectTransform.transform.localEulerAngles =
planeChoice[3];
      break;
    case 4:
      objectTransform.transform.localEulerAngles =
planeChoice[4];
      break;
    case 1:
      objectTransform.transform.localEulerAngles =
planeChoice[5];
      break;
    /*case 6:
      Vector3 temp5 =
  objectTransform.transform.localEulerAngles;
      temp5.z = -90.0f;
```

```
      objectTransform.transform.localEulerAngles = temp5;
      break;*/
    default:
      Debug.Log("Error");
      break;

    }
  }
  // This method should be used inside an update method
  // It rotates the part in order be inspected
  public void DemoRotate()
  {
    objectTransform.transform.Rotate (new Vector3 (0, 50.0f
* Time.deltaTime, 0));
  }
  // This method sets the gameObject in its initial
position and orientation
  public void ResetPosition()
  {
    objectTransform.transform.localEulerAngles =
initialOrientation;
    objectTransform.transform.localPosition =
initialPosition;
  }
  // Overloading the ResetPosition method
  // in order to set the part in a user defined position
and orientation
  public void ResetPosition(Vector3 paramOrientation,
Vector3 paramPosition)
  {
    objectTransform.transform.localEulerAngles =
paramOrientation;
    objectTransform.transform.localPosition =
paramPosition;
  }
  public void MoveXDirection()
  {
    transform.localPosition += new Vector3 (speed *
Time.deltaTime, 0.0f, 0.0f);
  }
  public void MoveOppositeXDirection()
  {
    transform.localPosition -= new Vector3 (speed *
Time.deltaTime, 0.0f, 0.0f);
  }
  public void MoveYDirection()
  {
    transform.localPosition += new Vector3 (0.0f, speed *
Time.deltaTime, 0.0f);
  }
  public void MoveOppositeYDirection()
  {
    transform.localPosition -= new Vector3 (0.0f, speed *
Time.deltaTime, 0.0f);
  }
  public void MoveZDirection()
  {
    transform.localPosition += new Vector3 (0.0f, 0.0f,
speed * Time.deltaTime);
  }
  public void MoveOppositeZDirection()
  {
    transform.localPosition -= new Vector3 (0.0f, 0.0f,
speed * Time.deltaTime);
  }
  public void RotateCwYDirection()
  {
    transform.localEulerAngles += new Vector3 (0, 80.0f *
speed * Time.deltaTime, 0);
  }
  public void RotateACwYDirection()
  {
    transform.localEulerAngles -= new Vector3 (0, 80.0f *
speed * Time.deltaTime, 0);
  }
}
```

• Nut Mover

```
using UnityEngine;
using System.Collections;

public class NutMover : MonoBehaviour {

  public float speed;
  // Use this for initialization
  void Start () {
    speed = 100.0f;
  }

  // Update is called once per frame
  private void Update () {}

  public void MoveDown()
  {
    transform.localPosition -=new
Vector3(transform.localPosition.x
,transform.localPosition.y,(speed * Time.deltaTime)/4.0f);
  }
  public void Screw()
  {
    MoveDown ();
    transform.eulerAngles +=new Vector3(0.0f ,0.0f,speed *
Time.deltaTime*20.0f);
  }
}
```

• Step Block Mover

```
using UnityEngine;
using System.Collections;

public class StepBlockMover : MonoBehaviour
{
  public float zPosition;
  public float speed;
  // Use this for initialization
  void Start ()
  {
    speed = 40.0f;
  }

  // Update is called once per frame
  void Update ()
  {
    zPosition = transform.localPosition.z;
  }

  public void MoveForward()
  {
    transform.localPosition += new Vector3 (0, 0, speed *
Time.deltaTime);
  }
  public void MoveBack()
  {
    transform.localPosition -= new Vector3 (0, 0, speed *
Time.deltaTime);
  }
}
```

• Stud Mover

```
using UnityEngine;
using System.Collections;

public class StudMover : MonoBehaviour {

  public float speed;
  // Use this for initialization
  void Start () {
    speed = 50.0f;
  }

  // Update is called once per frame
  void Update () {}

  public void MoveRight()
  {
    transform.Translate (new Vector3 (0, -speed *
Time.deltaTime,0));
  }
  public void MoveLeft()
  {
    transform.Translate (new Vector3 (0, speed *
Time.deltaTime,0));
  }
}
```

• Stud Mover

```
using UnityEngine;
using System.Collections;

public class StudMover : MonoBehaviour {

  public float speed;
  // Use this for initialization
  void Start () {
    speed = 50.0f;
  }

  // Update is called once per frame
  void Update () {}

  public void MoveRight()
  {
    transform.Translate (new Vector3 (0, -speed *
Time.deltaTime,0));
  }
  public void MoveLeft()
```

```
  {
    transform.Translate (new Vector3 (0, speed *
Time.deltaTime,0));
  }
}
```

- VideoHolder

```
using UnityEngine;
using System.Collections;

public class VideoHolder : MonoBehaviour {

  public MovieTexture firstVideo;
  public bool finished;

  AudioSource audio;
  private float endTime;

  // Use this for initialization
  void Start ()
  {
    finished = false;
    renderer.material.mainTexture = firstVideo;
    audio = GetComponent<AudioSource> ();
    firstVideo.Play ();
    firstVideo.loop = false;
    audio.Play ();
    endTime = Time.realtimeSinceStartup +
firstVideo.duration;
  }

  // Update is called once per frame
  void Update ()
  {
    if (Time.realtimeSinceStartup >= endTime)
    {
      finished = true;
      firstVideo.Stop();
    }
  }
}
```

- VideoObserver

```
using UnityEngine;
using System.Collections;
using Assets.Codes.States;

// This script is created to notify other scripts
// that the video has finished
public class videoObserver : MonoBehaviour {

  private bool videoFinished;
  // Use this for initialization
  void Start ()
  {
    videoFinished = false;
  }

  // Update is called once per frame
  void Update ()
  {
    videoFinished = GetComponent<VideoHolder>().finished;
    ChuckSetupThree.videoFinished = videoFinished;
    ViseSetupThree.videoFinished = videoFinished;
  }
}
```

- ViseMover

```
using UnityEngine;
using System.Collections;

public class ViseMover : MonoBehaviour {

  public float speedVise;
  public bool manualControl;
  // Use this for initialization
  void Start ()
  {
    speedVise = 10.0f;
    manualControl = false;
  }

  // Update is called once per frame
  void Update ()
  {
    if(manualControl)
    {
      if (Input.GetKey (KeyCode.LeftArrow))
      {
        CloseViseJaw();
      }
      if (Input.GetKey (KeyCode.RightArrow))
      {
        OpenViseJaw();
      }
    }

  }
  public void OpenViseJaw()
  {
    GameObject.FindWithTag
("movingJaw").transform.Translate (new Vector3 (speedVise *
Time.deltaTime, 0, 0));
  }
  public void CloseViseJaw()
  {
    GameObject.FindWithTag
("movingJaw").transform.Translate (new Vector3 (-speedVise
* Time.deltaTime, 0, 0));
  }

  public void MoveForward()
  {
    transform.localPosition -= new Vector3 (
Time.deltaTime, 0, 0);
  }
  public void MoveBack()
  {
    transform.localPosition += new Vector3 (
Time.deltaTime, 0, 0);
  }
}
```

- ViseVideoObserver

```
using UnityEngine;
using System.Collections;
using Assets.Codes.States;

public class ViseVideoObserver : MonoBehaviour {

  private bool videoFinished;
  // Use this for initialization
  void Start ()
  {
    videoFinished = false;
  }

  // Update is called once per frame
  void Update ()
  {
    videoFinished = GetComponent<VideoHolder>().finished;
    ViseSetupFinal.videoFinished = videoFinished;
  }
}
```

Interface – States

- IState

```
using UnityEngine;
using System.Collections;

namespace Assets.Codes.Interface
{
  public interface IState
  {
    void Action();
    void GUIImage();
    void Reset();
  }
}
```

- Choose Part State

```
using UnityEngine;
using System.Collections;
using Assets.Codes.Interface;
using Assets.Codes.ComponentScripts;

namespace Assets.Codes.States
{
  public class ChoosePartState : IState
  {
    private bool choisePromptActive;
    private GameObject FirstKaloupi;
    private GameObject SecondKaloupi;
    private MainManager _manager;
    private string name1;
    private string name2;
    private int counter;
    private Vector3[] initialPosition= new Vector3[2];
    private Vector3 finalPosition;

    public static string finalName;
```

```
    public static int scenarioNumber;

    // Constructor
    public ChoosePartState(MainManager manager)
    {
      Debug.Log("Choose Part State is active");
      finalName = "Part";
      initialPosition [0] = new Vector3 (100.0f, 0, 0);
      initialPosition [1] = new Vector3 (-100.0f, 0, 0);
      finalPosition = new Vector3 (0, 0, 0);
      name1 = "FirstKaloupi";
      name2 = "SecondKaloupi";
      _manager = manager;
      choisePromptActive = false;
      Reset ();
    }

    public void Reset()
    {
      if (FirstKaloupi != null)
      {
        _manager.DestroyPrefab(FirstKaloupi);
      }
      if (SecondKaloupi != null)
      {
        _manager.DestroyPrefab(SecondKaloupi);

      }
      choisePromptActive = false;
      counter = 1;

  FirstKaloupi=_manager.CreatePrefab(0,initialPosition[0]);
      FirstKaloupi.name = name1 ;
      SecondKaloupi=_manager.CreatePrefab
(1,initialPosition[1]);
      SecondKaloupi.name = name2;
      scenarioNumber = 0;
    }
    public void Action()
    {
      if (choisePromptActive) {
        GameObject.Find(finalName).transform.localPosition
= finalPosition;
        GameObject.Find
(finalName).GetComponent<MovingKaloupi>().DemoRotate();
      }
      if (counter==GameObject.Find
(name1).GetComponent<MouseListener> ().numberOfClicks)
      {
        _manager.DestroyPrefab(SecondKaloupi);
        choisePromptActive = true;
        FirstKaloupi.name =finalName;
        scenarioNumber = 1;
        counter++;
      }
      else if(counter==GameObject.Find
(name2).GetComponent<MouseListener> ().numberOfClicks)
      {
        _manager.DestroyPrefab(FirstKaloupi);
        choisePromptActive = true;
        SecondKaloupi.name = finalName;
        scenarioNumber = 2;
        counter++;
      }
      Debug.Log (choisePromptActive);
    }
    public void GUIImage()
    {
      // Text position and style for the axial dimension
box
      GUI.skin = _manager.guiSkin;

      Rect textHolder = new Rect ((int)(Screen.width / 2)-
200, Screen.height-(int)(Screen.height/7), 400,
(int)(Screen.height/8));
      /*
      GUIStyle boxStyle = "box";
      boxStyle.fontSize = 25;
      boxStyle.alignment = TextAnchor.MiddleCenter;
      boxStyle.wordWrap = true;*/
      if (!choisePromptActive)
      {
          GUI.Box (textHolder, "Choose the part you want to
setup");
      }
      else
      {
        GUI.Box (textHolder, "Proceed with the setup of
this part?");
        if (GUI.Button (new Rect (100, Screen.height - 100,
150, 50), "Cancel"))
        {
          Reset();
        }
        if (GUI.Button (new Rect (Screen.width - 200,
Screen.height - 100, 150, 50), "Confirm"))
        {
          GameObject.Find
(finalName).GetComponent<MouseListener>
().mouseToggleSwitch = false;
          GameObject.Find
(finalName).GetComponent<MovingKaloupi>().ResetPosition();

  GameObject.Find(finalName).transform.localPosition =
finalPosition;
          _manager.SwitchState(new
OrientationState(_manager));
        }
      }
    }
  }
}
```

- Chuck Setup Final

```
using UnityEngine;
using System.Collections;
using Assets.Codes.Interface;
using Assets.Codes.ComponentScripts;
using Assets.Codes.States;

namespace Assets.Codes.States
{
  public class ChuckSetupFinal : IState
  {

    private MainManager _manager;
    private GameObject chuck;
    private GameObject part;
    private GameObject[] jaw = new GameObject[3];
    private GameObject videoObject;
    private string textToShow;
    private int scenario;
    private bool[] completedMotion = new bool[6];
    private int helpingCounter;

    private Vector3 partInitialPosition;
    private Vector3 videoInitialPosition;
    private Vector3 videoInitialRotation;
    private Vector3 videoInitialScale;

    public static bool videoFinished;

    public ChuckSetupFinal(MainManager manager)
    {
      _manager = manager;
      _manager.StartCoroutine (showText ());
      chuck = GameObject.Find (ChuckSetupFour.chuckName);
      part = GameObject.Find (ChoosePartState.finalName);
      partInitialPosition = part.transform.localPosition;
      scenario = OrientationState.scenario;
      helpingCounter = 0;
      for(int i=0; i<completedMotion.Length;i++)
      {
        completedMotion[i] =false;
      }
      for (int i=0; i<jaw.Length;i++)
      {
        int j = i+1;
        jaw[i] = GameObject.FindWithTag("jaw"+j);
      }
      videoInitialPosition = new Vector3 (-0.56f,-
0.3012f,0.1605f);
      videoInitialRotation = new Vector3 (1.4283f,180,-
3.37f);
      videoInitialScale = new Vector3
(0.05f,0.0025f,0.05f);
      videoFinished = false;
    }
    public void Action()
    {
      if(videoFinished)
      {
        _manager.DestroyPrefab(videoObject);
      }
      if(scenario == 7)
      {

  if((part.transform.localPosition.y<=1.4f)&&(!completedMot
ion[0]))
        {

  part.GetComponent<MovingKaloupi>().MoveYDirection();
        }
        else if(helpingCounter==0)
        {
          part.transform.localPosition = new
Vector3(part.transform.localPosition.x,1.4f,part.transform.
localPosition.z);
```

```
          completedMotion[0] = true;
        }
        if((part.transform.localPosition.z>=-
0.594f)&&(!completedMotion[1])&&(completedMotion[0]))
        {

  part.GetComponent<MovingKaloupi>().MoveOppositeZDirection
();
          helpingCounter = 1;
        }
        else if(helpingCounter == 1&&(!completedMotion[1]))
        {
          part.transform.localPosition = new
Vector3(part.transform.localPosition.x,part.transform.local
Position.y,-0.594f);
          completedMotion[1] = true;
        }

  if((part.transform.localPosition.x<=0.525f)&&(!completedM
otion[2])&&(completedMotion[1]))
        {
          helpingCounter = 2;

  part.GetComponent<MovingKaloupi>().MoveXDirection();
        }
        else if(helpingCounter == 2&&(!completedMotion[2]))
        {
          part.transform.localPosition = new
Vector3(0.525f,part.transform.localPosition.y,part.transfor
m.localPosition.z);
          completedMotion[2] = true;
        }

  if((part.transform.localPosition.y>=1.039f)&&(!completedM
otion[3])&&(completedMotion[2]))
        {
          helpingCounter = 3;

  part.GetComponent<MovingKaloupi>().MoveOppositeYDirection
();
        }
        else if(helpingCounter == 3&&(!completedMotion[3]))
        {
          completedMotion[3] = true;
          CreateVideoObject();
          part.transform.localPosition = new
Vector3(part.transform.localPosition.x,1.039f,part.transfor
m.localPosition.z);
        }
        if (((jaw[0].transform.localPosition.x>-
16.0f)||(jaw[1].transform.localPosition.x>-
16.0f)||(jaw[2].transform.localPosition.x>-
16.0f))&&(!completedMotion[4])&&(completedMotion[3]))
        {
          helpingCounter = 4;
          chuck.GetComponent<ChuckMover>().CloseJaws();
        }
        else if(helpingCounter == 4&&(!completedMotion[4]))
        {
          completedMotion[4] = true;

  chuck.GetComponent<ChuckMover>().SetPositionOfJaws(-
16.0f);
        }
      }
      if(scenario == 10)
      {

  if((part.transform.localPosition.y<=1.5f)&&(!completedMot
ion[0]))
        {

  part.GetComponent<MovingKaloupi>().MoveYDirection();
        }
        else if(helpingCounter==0)
        {
          part.transform.localPosition = new
Vector3(part.transform.localPosition.x,1.5f,part.transform.
localPosition.z);
          completedMotion[0] = true;
        }
        if((part.transform.localPosition.z>=-
0.605f)&&(!completedMotion[1])&&(completedMotion[0]))
        {

  part.GetComponent<MovingKaloupi>().MoveOppositeZDirection
();
          helpingCounter = 1;
        }
        else if(helpingCounter == 1&&(!completedMotion[1]))
        {
          part.transform.localPosition = new
Vector3(part.transform.localPosition.x,part.transform.local
Position.y,-0.605f);
          completedMotion[1] = true;
        }

  if((part.transform.localPosition.x<=1.232f)&&(!completedM
otion[2])&&(completedMotion[1]))
        {
          helpingCounter = 2;

  part.GetComponent<MovingKaloupi>().MoveXDirection();
        }
        else if(helpingCounter == 2&&(!completedMotion[2]))
        {
          part.transform.localPosition = new
Vector3(1.232f,part.transform.localPosition.y,part.transfor
m.localPosition.z);
          completedMotion[2] = true;
        }
        if
(((jaw[0].transform.localPosition.x<17.0f)||(jaw[1].transfo
rm.localPosition.x<17.0f)||(jaw[2].transform.localPosition.
x<17.0f))&&(!completedMotion[3])&&(completedMotion[2]))
        {
          helpingCounter = 3;
          chuck.GetComponent<ChuckMover>().OpenJaws();
        }
        else if(helpingCounter == 3&&(!completedMotion[3]))
        {
          completedMotion[3] = true;
        }

  if((part.transform.localPosition.y>=1.14f)&&(!completedMo
tion[4])&&(completedMotion[3]))
        {
          helpingCounter = 4;

  part.GetComponent<MovingKaloupi>().MoveOppositeYDirection
();
        }
        else if(helpingCounter == 4&&(!completedMotion[4]))
        {
          part.transform.localPosition = new
Vector3(part.transform.localPosition.x,1.14f,part.transform
.localPosition.z);
          completedMotion[4] = true;
          CreateVideoObject();
        }
        if
(((jaw[0].transform.localPosition.x>10.5f)||(jaw[1].transfo
rm.localPosition.x>10.5f)||(jaw[2].transform.localPosition.
x>10.5f))&&(!completedMotion[5])&&(completedMotion[4]))
        {
          helpingCounter = 5;
          chuck.GetComponent<ChuckMover>().CloseJaws();
        }
        else if(helpingCounter == 5&&(!completedMotion[5]))
        {
          completedMotion[5] = true;

  chuck.GetComponent<ChuckMover>().SetPositionOfJaws(10.5f)
;
        }
      }
      if(scenario == 11)
      {

  if((part.transform.localPosition.y<=1.4f)&&(!completedMot
ion[0]))
        {

  part.GetComponent<MovingKaloupi>().MoveYDirection();
        }
        else if(helpingCounter==0)
        {
          part.transform.localPosition = new
Vector3(part.transform.localPosition.x,1.4f,part.transform.
localPosition.z);
          completedMotion[0] = true;
        }
        if((part.transform.localPosition.z>=-
0.437f)&&(!completedMotion[1])&&(completedMotion[0]))
        {

  part.GetComponent<MovingKaloupi>().MoveOppositeZDirection
();
          helpingCounter = 1;
        }
        else if(helpingCounter == 1&&(!completedMotion[1]))
        {
          part.transform.localPosition = new
Vector3(part.transform.localPosition.x,part.transform.local
Position.y,-0.437f);
          completedMotion[1] = true;
        }

  if((part.transform.localPosition.x<=0.85f)&&(!completedMo
tion[2])&&(completedMotion[1]))
```

```
        {
          helpingCounter = 2;

  part.GetComponent<MovingKaloupi>().MoveXDirection();
        }
        else if(helpingCounter == 2&&(!completedMotion[2]))
        {
          part.transform.localPosition = new
Vector3(0.85f,part.transform.localPosition.y,part.transform
.localPosition.z);
          completedMotion[2] = true;
        }

  if((part.transform.localPosition.y>=1.03754f)&&(!complete
dMotion[3])&&(completedMotion[2]))
        {
          helpingCounter = 3;

  part.GetComponent<MovingKaloupi>().MoveOppositeYDirection
();
        }
        else if(helpingCounter == 3&&(!completedMotion[3]))
        {
          part.transform.localPosition = new
Vector3(part.transform.localPosition.x,1.03754f,part.transf
orm.localPosition.z);
          completedMotion[3] = true;
          CreateVideoObject();
        }
        if (((jaw[0].transform.localPosition.x>-
23.5f)||(jaw[1].transform.localPosition.x>-
23.5f)||(jaw[2].transform.localPosition.x>-
23.5f))&&(!completedMotion[4])&&(completedMotion[3]))
        {
          helpingCounter = 4;
          chuck.GetComponent<ChuckMover>().CloseJaws();
        }
        else if(helpingCounter == 4&&(!completedMotion[4]))
        {
          completedMotion[4] = true;

  chuck.GetComponent<ChuckMover>().SetPositionOfJaws(-
23.5f);
        }
      }
    }

    public void GUIImage()
    {
      GUI.skin = _manager.guiSkin;
      GUI.Label(new Rect(10, 10, Screen.width-20,
(int)(Screen.height/7)), textToShow);
      if(videoFinished)
      {
        if (GUI.Button (new Rect (100, Screen.height - 100,
250, 50), "Choose different Setup"))
        {
          Reset();
          _manager.SwitchState(new
ChoosePartState(_manager));
        }
        if (GUI.Button (new Rect (Screen.width - 200,
Screen.height - 100, 150, 50), "Play Video"))
        {
          CreateVideoObject();
        }
      }
    }

    public void Reset()
    {
      _manager.DestroyPrefab (part);
      _manager.DestroyPrefab (chuck);
    }

    private void CreateVideoObject()
    {
      videoFinished = false;
      videoObject = _manager.CreatePrefab
(14,videoInitialPosition,videoInitialRotation,"BackgroundPl
ane");
      videoObject.transform.localPosition =
videoInitialPosition;
      videoObject.transform.localEulerAngles =
videoInitialRotation;
      videoObject.transform.localScale = videoInitialScale;

    }

    IEnumerator showText()
    {
      string str = "Set the part on the chuck and hold
it.";
      char [] temp = str.ToCharArray();
      int counter = 0;
      while (counter < temp.Length)
      {
        textToShow += temp[counter];
        counter++;
        yield return new WaitForSeconds(0.01f);
      }

      yield return new WaitForSeconds (5.0f);
      textToShow = "";
    }
  }
}
```

- ChuckSetupFour

```
using UnityEngine;
using System.Collections;
using Assets.Codes.Interface;
using Assets.Codes.ComponentScripts;
using Assets.Codes.States;

namespace Assets.Codes.States
{
  public class ChuckSetupFour : IState
  {

    private MainManager _manager;

    private GameObject part;
    private GameObject chuck;
    private GameObject[] stud = new GameObject[3];
    private GameObject[] clamp = new GameObject[3];
    private GameObject[] stepBlock = new GameObject[3];
    private GameObject[] nut = new GameObject[3];

    private bool nutsCreated;
    private bool[] nutPlaced= new bool[3];
    private bool[] studPlaced = new bool[3];
    private bool[] clampPlaced = new bool[3];
    private bool chuckPlaced;
    private bool clampsCreated;
    private bool stepBlocksCreated;
    private bool[] stepBlockPlaced = new bool[3];

    private Vector3 initilaChuckPosition;
    private Vector3 initialChuckOrientation;
    private Vector3 initialClapmPosition;
    private Vector3 [] initialClampOrientation = new
Vector3[3];
    private Vector3[] initialStepBlockPosition = new
Vector3[3];
    private Vector3[] initialStepBlockOrientation = new
Vector3[3];
    private Vector3 initialNutPosition;
    private Vector3 initialNutOrientation;
    private float[] finalZPositionStepBlock = new float[3];
    private float nutScrewStartPoint;
    private float nutScrewEndPoint;

    private string textToShow;

    public static string chuckName;

    public ChuckSetupFour(MainManager manager)
    {
      _manager = manager;
      initilaChuckPosition = new Vector3 (800f, 182.0f, -
68.0f);
      initialChuckOrientation = new Vector3 (1.17f, 215.5f,
180.0f);

      initialClapmPosition = new Vector3 (0, 0, 190.0f);
      initialClampOrientation [0] = new Vector3 (0, 0, 30);
      initialClampOrientation [1] = new Vector3 (0, 0,
260);
      initialClampOrientation [2] = new Vector3 (0, 0,
300);

      initialStepBlockPosition [0] = new Vector3 (465, 0, -
332);
      initialStepBlockPosition [1] = new Vector3 (46, 0,
247);
      initialStepBlockPosition [2] = new Vector3 (343, 0,
203);
      initialStepBlockOrientation [0] = new Vector3
(0,300,0);
      initialStepBlockOrientation [1] = new Vector3
(0,170,0);
      initialStepBlockOrientation [2] = new Vector3
(0,210,0);
      finalZPositionStepBlock [0] = 83.0f;
      finalZPositionStepBlock [1] = 96.0f;
      finalZPositionStepBlock [2] = 73.0f;
      initialNutPosition = new Vector3 (0, 0, 260);
```

```
      nutScrewStartPoint = 180.0f;
      nutScrewEndPoint = 124.0f;

      chuckPlaced = false;
      clampsCreated = false;
      stepBlocksCreated = false;
      nutsCreated = false;
      chuckName = "Chuck";

      part = GameObject.Find (ChoosePartState.finalName);
      stud [0] = GameObject.Find
(ChuckSetupThree.chuckStudFirstSlotFar);
      stud [1] = GameObject.Find
(ChuckSetupThree.chuckStudSecondSlot);
      stud[2] = GameObject.Find
(ChuckSetupThree.chuckStudFirstSlotNear);
      chuck = _manager.CreatePrefab (3,
initilaChuckPosition, initialChuckOrientation);
      chuck.name = chuckName;

      _manager.StartCoroutine (showText ());
    }
    public void Action()
    {
      if (chuck.transform.localPosition.x > 2.9f)
      {
        chuck.transform.localPosition-= new Vector3 (
1.0f*Time.deltaTime, 0, 0);
      }
      else if(!chuckPlaced)
      {
        chuck.transform.localPosition = new
Vector3(2.9f,chuck.transform.localPosition.y,chuck.transfor
m.localPosition.z);
        chuckPlaced = true;
      }
      if (chuckPlaced)
      {
        if (stud [0].transform.localPosition.z <= -1.2f)
        {
          stud [0].GetComponent<StudMover> ().MoveRight ();
        }
        else
        {
          studPlaced[0] = true;
        }
        if (stud [1].transform.localPosition.z >= 0.09f)
        {
          stud [1].GetComponent<StudMover> ().MoveLeft ();
        }
        else
        {
          studPlaced[1] = true;
        }
        if (stud [2].transform.localPosition.z >= 0.02f)
        {
          stud [2].GetComponent<StudMover> ().MoveLeft ();
        }
        else
        {
          studPlaced[2] = true;
        }
      }

  if((studPlaced[0])&&(studPlaced[1])&&(studPlaced[2])&&(!c
lampsCreated))
      {
        CreateClamps();
      }
      if(clampsCreated)
      {
        for(int i = 0; i< clamp.Length;i++)
        {
          if (clamp [i].transform.localPosition.z >= 97)
          {
            clamp
[i].GetComponent<ClampMover>().MoveDown();
          }
          else
          {
            clampPlaced[i] = true;
          }
        }
      }
      if
((clampPlaced[0])&&(clampPlaced[1])&&(clampPlaced[2])&&(!st
epBlocksCreated))
      {
        CreateStepBlocks();
      }
      if(stepBlocksCreated)
      {
        for(int i = 0; i< stepBlock.Length;i++)
        {
          if (stepBlock
[i].GetComponentInChildren<StepBlockMover>().zPosition <=
finalZPositionStepBlock[i])
          {
            stepBlock
[i].GetComponentInChildren<StepBlockMover>().MoveForward();
          }
          else
          {
            stepBlockPlaced[i] = true;
          }
        }
      }
      if
((stepBlockPlaced[0])&&(stepBlockPlaced[1])&&(stepBlockPlac
ed[2])&&(!nutsCreated))
      {
        CreateNuts();
      }
      if(nutsCreated)
      {
        for(int i = 0; i< nut.Length;i++)
        {
          if (nut [i].transform.localPosition.z >=
nutScrewStartPoint)
          {
            nut[i].GetComponent<NutMover>().MoveDown();
          }
          else if(nut [i].transform.localPosition.z >=
nutScrewEndPoint)
          {
            nut[i].GetComponent<NutMover>().Screw();
          }
          else
          {
            nutPlaced[i] = true;
          }
        }
      }
    }

    public void GUIImage()
    {
      GUI.skin = _manager.guiSkin;
      GUI.Label(new Rect(10, 10, Screen.width-20,
(int)(Screen.height/7)), textToShow);
      if(nutPlaced[2])
      {
        if (GUI.Button (new Rect (100, Screen.height - 100,
150, 50), "Back"))
        {
          MakeEveryGameObjectChildOfChuck();
          _manager.DestroyPrefab(chuck);
          _manager.SwitchState(new
ChuckSetupThree(_manager));
        }
        if (GUI.Button (new Rect (Screen.width - 200,
Screen.height - 100, 150, 50), "Proceed"))
        {
          MakeEveryGameObjectChildOfChuck();
          _manager.SwitchState(new
ChuckSetupFinal(_manager));
        }
      }
    }

    public void Reset()
    {}

    private void MakeEveryGameObjectChildOfChuck()
    {
      for(int i=0; i<stud.Length;i++)
      {
        stud[i].transform.parent = chuck.transform;
      }
    }

    private void CreateClamps()
    {
      for(int i=0; i<clamp.Length;i++)
      {

  clamp[i]=_manager.CreatePrefab(11,initialClapmPosition,in
itialClampOrientation[i],stud[i].name);
        clamp[i].transform.localPosition =
initialClapmPosition;
        clamp[i].transform.localEulerAngles =
initialClampOrientation[i];
        clamp[i].name = "clamp"+i;
      }
      clampsCreated = true;
    }

    private void CreateStepBlocks()
    {
```

```
      for(int i=0; i<stepBlock.Length;i++)
      {

  stepBlock[i]=_manager.CreatePrefab(12,initialStepBlockPos
ition[i],initialStepBlockOrientation[i]);
        stepBlock[i].name = "stepBlock"+i;
        stepBlock[i].transform.parent = stud[i].transform;
      }
      stepBlocksCreated = true;
    }

    private void CreateNuts()
    {
      for(int i=0; i<nut.Length;i++)
      {

  nut[i]=_manager.CreatePrefab(13,initialNutPosition,initia
lNutOrientation,stud[i].name);
        nut[i].transform.localPosition =
initialNutPosition;
        nut[i].transform.localEulerAngles =
initialNutOrientation;
        nut[i].name = "nut"+i;
      }
      nutsCreated = true;
    }

    IEnumerator showText()
    {
      string str = "Place the chuck on the table and use
the clamps to hold it.";
      char [] temp = str.ToCharArray();
      int counter = 0;
      while (counter < temp.Length)
      {
        textToShow += temp[counter];
        counter++;
        yield return new WaitForSeconds(0.01f);
      }
      yield return new WaitForSeconds (5.0f);
      textToShow = "";
    }
  }
}
```

- Chuck Setup Three

```
using UnityEngine;
using System.Collections;
using Assets.Codes.Interface;
using Assets.Codes.ComponentScripts;
using Assets.Codes.States;

namespace Assets.Codes.States
{
  public class ChuckSetupThree : IState
  {

    private MainManager _manager;
    private GameObject videoObject;
    private GameObject part;
    private Vector3 initialPosition;
    private Vector3 initialRotaion;
    private Vector3 initialScale;
    private Vector3 firstSlotPosition;
    private Vector3 secondSlotPosition;
    private Vector3 studOrientation;
    private string textToShow;
    private bool terminateExecution;
    private GameObject[] stud = new GameObject[3];

    public static bool videoFinished;
    public static string chuckStudFirstSlotFar;
    public static string chuckStudFirstSlotNear;
    public static string chuckStudSecondSlot;
    public static string studFirstSlot;
    public static string studSecondSlot;
    public int scenario;

    public ChuckSetupThree(MainManager manager)
    {
      _manager = manager;
      scenario = OrientationState.scenario;
      videoFinished = false;
      terminateExecution = false;

      part = GameObject.Find (ChoosePartState.finalName);
      part.transform.localPosition = new Vector3 (-0.5f,
part.transform.localPosition.y,
part.transform.localPosition.z);
      initialPosition = new Vector3 (-0.56f,-
0.3012f,0.1605f);
      initialRotaion = new Vector3 (1.4283f,180,-3.37f);
      initialScale = new Vector3 (0.05f,0.0025f,0.05f);
      chuckStudFirstSlotFar = "studOne";
      chuckStudSecondSlot = "studTwo";
      chuckStudFirstSlotNear ="studThree";
      studFirstSlot = chuckStudFirstSlotFar;
      studSecondSlot = chuckStudSecondSlot;
      firstSlotPosition = new Vector3 (258.0f,0.0f,90.0f);
      secondSlotPosition = new Vector3 (114.0f,0.0f,90.0f);

      studOrientation = new Vector3 (270.0f,0,0);


      stud [0] = _manager.CreatePrefab
(4,firstSlotPosition,studOrientation);
      stud [1] = _manager.CreatePrefab
(4,secondSlotPosition,studOrientation);
      if(!(scenario==2||scenario==8))
      {
        stud[2] = _manager.CreatePrefab
(4,firstSlotPosition,studOrientation);
        stud[2].GetComponentInChildren<MeshRenderer>
().enabled = false;
        stud[2].name = chuckStudFirstSlotNear;
        CreateVideoObject ();
      }
      stud[0].name = chuckStudFirstSlotFar;
      stud[1].name = chuckStudSecondSlot;
      _manager.StartCoroutine (showText ());
    }
    public void Action()
    {
      if(videoFinished)
      {
        _manager.DestroyPrefab(videoObject);
      }
      if (!terminateExecution)
      {
        if(!(scenario==2||scenario==8))
        {
          if (stud [1].transform.localPosition.z >= 0.25f)
          {
            stud [1].GetComponent<StudMover> ().MoveLeft
();
          }
          else if (stud [2].transform.localPosition.z >=
0.25f)
          {
            stud [2].GetComponentInChildren<MeshRenderer>
().enabled = true;
            stud [2].GetComponent<StudMover> ().MoveLeft
();
          }
          if (stud [0].transform.localPosition.z >= -1.5f)
          {
            stud [0].GetComponent<StudMover> ().MoveLeft
();
          }
        }
        else
        {
          if (stud [0].transform.localPosition.z >= -1.0f)
          {
            stud [0].GetComponent<StudMover> ().MoveLeft
();
          }
          if (stud [1].transform.localPosition.z >= -1.0f)
          {
            stud [1].GetComponent<StudMover> ().MoveLeft
();
          }
          else
          {
            videoFinished=true;
          }
        }

      }
    }

    public void GUIImage()
    {
      GUI.skin = _manager.guiSkin;
      GUI.Label(new Rect(10, 10, Screen.width-20,
(int)(Screen.height/7)), textToShow);
      if(videoFinished)
      {
        if (GUI.Button (new Rect (100, Screen.height - 100,
150, 50), "Back"))
        {
          terminateExecution = true;
          Reset();
          _manager.SwitchState(new
OrientationState(_manager));
        }
        if (GUI.Button (new Rect (Screen.width - 200,
Screen.height - 100, 150, 50), "Proceed"))
        {
```

```
          if(!(scenario==2||scenario==8))
          {
            _manager.SwitchState(new
ChuckSetupFour(_manager));
          }
          else
          {
            _manager.SwitchState(new
ClampsSetupFinal(_manager));
          }
        }
        if(!(scenario==2||scenario==8))
        {
          if (GUI.Button (new Rect ((int)(Screen.width/2) -
100, Screen.height - 100, 200, 50), "Play Video"))
          {
            videoFinished = false;
            CreateVideoObject();
          }
        }
      }
    }

    public void Reset()
    {
      part.transform.localPosition = new Vector3 (0,0,0);
      for (int i = 0; i<stud.Length; i++)
      {
        _manager.DestroyPrefab(stud[i]);
      }
    }

    private void CreateVideoObject()
    {
      videoObject = _manager.CreatePrefab
(10,initialPosition,initialRotaion,"BackgroundPlane");
      videoObject.transform.localPosition =
initialPosition;
      videoObject.transform.localEulerAngles =
initialRotaion;
      videoObject.transform.localScale = initialScale;

    }
    IEnumerator showText()
    {
      string str;
      if(!(scenario==2||scenario==8))
      {
        str = "Insert two studs in the first slot and the
third stud in the second slot of the table.";
      }
      else
      {
        str = "Insert one stud in the first slot and one in
the second.";
      }
      char [] temp = str.ToCharArray();
      int counter = 0;
      while (counter < temp.Length)
      {
        textToShow += temp[counter];
        counter++;
        yield return new WaitForSeconds(0.01f);
      }
      yield return new WaitForSeconds (1.0f);
    }
  }
}
```

• Clamp Setup Final

```
using UnityEngine;
using System.Collections;
using Assets.Codes.Interface;
using Assets.Codes.ComponentScripts;
using Assets.Codes.States;

namespace Assets.Codes.States
{
  public class ClampsSetupFinal : IState
  {

    private MainManager _manager;
    private GameObject part;
    private GameObject[] stud= new GameObject[2];
    private GameObject[] parallel= new GameObject[2];
    private GameObject[] assemblyStepBlock = new
GameObject[2];
    private GameObject[] clamp = new GameObject[2];
    private GameObject[] stepBlock = new GameObject[2];
    private GameObject[] nut = new GameObject[2];

    private Vector3 parallelinitialposition;
    private Vector3[] assemblyInitialPosition = new
Vector3[2];
    private Vector3[] assemblyInitialRotation = new
Vector3[2];
    private Vector3[] initialClapmPosition = new
Vector3[2];
    private Vector3[] initialClampOrientation = new
Vector3[2];
    private Vector3[] initialStepBlockPosition = new
Vector3[2];
    private Vector3[] initialStepBlockOrientation = new
Vector3[2];
    private Vector3 initialNutPosition;
    private Vector3 initialNutOrientation;

    private bool[] parallelCreated= new bool[2];
    private bool[] clampsPlaced = new bool[2];
    private bool[] stepBlockPlaced = new bool[2];
    private bool[] nutPlaced = new bool[2];
    private bool completedParallelMotion;
    public static bool assemblyPlaced;
    private bool clampsCreated;
    private bool stepBlocksCreated;
    private bool nutsCreated;
    private float nutScrewStartPoint;
    private float nutScrewEndPoint;

    private float parallelFinalYPosition;
    private string textToShow;
    private int scenario;
    private bool[] completedMotion = new bool[6];
    private int helpingCounter;

    public ClampsSetupFinal(MainManager manager)
    {
      _manager = manager;
      scenario = OrientationState.scenario;

      part = GameObject.Find (ChoosePartState.finalName);
      stud [0] = GameObject.Find
(ChuckSetupThree.studFirstSlot);
      stud [1] = GameObject.Find
(ChuckSetupThree.studSecondSlot);
      parallelinitialposition = new Vector3 (200, 100, -
200);
      assemblyInitialPosition[0] = new Vector3(-33,25,0);
      assemblyInitialPosition[1] = new Vector3(31,-25,0);
      assemblyInitialRotation [0] = new Vector3 (0, 90,
90);
      assemblyInitialRotation [1] = new Vector3 (0, -90, -
90);
      initialClapmPosition [0] = new Vector3 (0, 0, 300);
      initialClapmPosition [1] = new Vector3 (8.31f, 0,
300);
      initialClampOrientation [0] = new Vector3 (0, 0, 0);
      initialClampOrientation [1] = new Vector3 (0,0,180);
      initialStepBlockPosition [0] = new Vector3 (250, -25,
117);
      initialStepBlockPosition [1] = new Vector3 (-250, 25,
117);
      initialStepBlockOrientation [0] = new Vector3 (0, -
90, -90);
      initialStepBlockOrientation [1] = new Vector3 (0, 90,
90);
      initialNutPosition = new Vector3 (0,0,230);
      initialNutOrientation = new Vector3 (0, 0, 0);
      nutScrewStartPoint = 181.0f;
      nutScrewEndPoint = 151.0f;

      parallelCreated[0] = false;
      parallelCreated [1] = false;
      assemblyPlaced = false;
      clampsCreated = false;
      stepBlocksCreated = false;
      nutsCreated = false;
      completedParallelMotion = false;
      helpingCounter = 0;
      for(int i=0; i<completedMotion.Length;i++)
      {
        completedMotion[i] =false;
      }
      for(int i=0; i<clampsPlaced.Length;i++)
      {
        clampsPlaced[i] =false;
      }
      for(int i=0; i<stepBlockPlaced.Length;i++)
      {
        stepBlockPlaced[i] =false;
        nutPlaced[i] =false;
      }
      _manager.StartCoroutine (showText ());
    }
    public void Action()
    {
      if(scenario == 2)
      {
        if(!parallelCreated[0])
```

```
        {

  parallel[0]=_manager.CreatePrefab(19,parallelinitialposit
ion);
          parallelCreated[0]=true;
        }

  if(parallel[0].transform.localPosition.y>=0.025f&&(!compl
etedMotion[0]))
        {
          parallel[0].transform.localPosition -= new
Vector3(0,1.0f*Time.deltaTime,0);

        }
        else if(helpingCounter ==0 &&(!completedMotion[0]))
        {
          parallel[0].transform.localPosition = new
Vector3(parallel[0].transform.localPosition.x,0.025f,parall
el[0].transform.localPosition.z);
          completedMotion[0]=true;
          parallel[0].transform.parent = stud[0].transform;
          CreateAssemblyStepBlocks();
          helpingCounter = 1;
        }

  if((part.transform.localPosition.y<=1.7f)&&(!completedMot
ion[1])&&assemblyPlaced)
        {
          helpingCounter =2;

  part.GetComponent<MovingKaloupi>().MoveYDirection();
        }
        else if(helpingCounter == 2&&(!completedMotion[1]))
        {
          completedMotion[1] = true;
        }
        if((part.transform.localPosition.z>=-
0.995f)&&(!completedMotion[2])&&(completedMotion[1]))
        {
          helpingCounter=3;

  part.GetComponent<MovingKaloupi>().MoveOppositeZDirection
();
        }
        else if(helpingCounter == 3&&(!completedMotion[2]))
        {
          completedMotion[2] = true;
          part.transform.localPosition = new
Vector3(part.transform.localPosition.x,part.transform.local
Position.y,-0.995f);
        }

  if((part.transform.localPosition.x<0.955f)&&(!completedMo
tion[3])&&(completedMotion[2]))
        {
          helpingCounter = 4;

  part.GetComponent<MovingKaloupi>().MoveXDirection();
        }
        else if(helpingCounter == 4&&(!completedMotion[3]))
        {
          part.transform.localPosition = new
Vector3(0.955f,part.transform.localPosition.y,part.transfor
m.localPosition.z);
          completedMotion[3] = true;
        }


  if((part.transform.localPosition.y>=0.71f)&&(!completedMo
tion[4])&&(completedMotion[3]))
        {
          helpingCounter = 5;

  part.GetComponent<MovingKaloupi>().MoveOppositeYDirection
();
        }
        else if(helpingCounter == 5&&(!completedMotion[4]))
        {
          part.transform.localPosition = new
Vector3(part.transform.localPosition.x,0.71f,part.transform
.localPosition.z);
          completedMotion[4] = true;
        }
        if(completedMotion[4]&&!clampsCreated)
        {
          CreateClamps();
        }

  if(clampsCreated&&(!(clampsPlaced[0]||clampsPlaced[1])))
        {
          for(int i = 0; i< clamp.Length;i++)
          {
            if (clamp [i].transform.localPosition.z >=
123.74f)
            {
              clamp
[i].GetComponent<ClampMover>().MoveDown();
            }
            else
            {
              clampsPlaced[i] = true;
              clamp[i].transform.localPosition = new
Vector3(clamp[i].transform.localPosition.x,clamp[i].transfo
rm.localPosition.y,123.74f);
            }
          }
        }

  if(clampsPlaced[0]&&clampsPlaced[1]&&(!stepBlocksCreated)
)
        {
          CreateStepBlocks();
        }

  if(stepBlocksCreated&&!(stepBlockPlaced[0]||stepBlockPlac
ed[1]))
        {
          if(stepBlock[0].transform.localPosition.x>=179)
          {
            stepBlock[0].transform.localPosition -=new
Vector3(30.0f*Time.deltaTime,0,0);
          }
          else if(!stepBlockPlaced[0])
          {
            stepBlock[0].transform.localPosition =new
Vector3(179,stepBlock[0].transform.localPosition.y,stepBloc
k[0].transform.localPosition.z);
            stepBlockPlaced[0] = true;
          }
          if(stepBlock[1].transform.localPosition.x<=-179)
          {
            stepBlock[1].transform.localPosition +=new
Vector3(30.0f*Time.deltaTime,0,0);
          }
          else if(!stepBlockPlaced[1])
          {
            stepBlock[1].transform.localPosition =new
Vector3(-
179,stepBlock[1].transform.localPosition.y,stepBlock[1].tra
nsform.localPosition.z);
            stepBlockPlaced[1] = true;
          }
        }
        if
((stepBlockPlaced[0])&&(stepBlockPlaced[1])&&(!nutsCreated)
)
        {
          CreateNuts();
        }
        if(nutsCreated&&!(nutPlaced[0]||nutPlaced[1]))
        {
          for(int i = 0; i< nut.Length;i++)
          {
            if (nut [i].transform.localPosition.z >=
nutScrewStartPoint)
            {
              nut[i].GetComponent<NutMover>().MoveDown();
            }
            else if(nut [i].transform.localPosition.z >=
nutScrewEndPoint)
            {
              nut[i].GetComponent<NutMover>().Screw();
            }
            else
            {
              nutPlaced[i] = true;
            }
          }
        }
      }

      if(scenario == 8)
      {
        if(!parallelCreated[0])
        {

  parallel[0]=_manager.CreatePrefab(19,parallelinitialposit
ion);
          parallelCreated[0]=true;
        }

  if(parallel[0].transform.localPosition.y>=0.025f&&(!compl
etedMotion[0]))
        {
          parallel[0].transform.localPosition -= new
Vector3(0,0.5f*Time.deltaTime,0);
        }
        else if(helpingCounter ==0 &&(!completedMotion[0]))
        {
```

```
          parallel[0].transform.localPosition = new
Vector3(parallel[0].transform.localPosition.x,0.025f,parall
el[0].transform.localPosition.z);
          completedMotion[0]=true;
          parallel[0].transform.parent = stud[0].transform;

  parallel[1]=_manager.CreatePrefab(19,parallelinitialposit
ion);
        }

  if(parallel[1].transform.localPosition.y>=0.075f&&(!compl
etedParallelMotion))
        {
          parallel[1].transform.localPosition -= new
Vector3(0,0.5f*Time.deltaTime,0);
        }
        else if(!completedParallelMotion)
        {
          parallel[1].transform.localPosition = new
Vector3(parallel[1].transform.localPosition.x,0.075f,parall
el[1].transform.localPosition.z);
          completedParallelMotion = true;
          parallel[1].transform.parent = stud[0].transform;
          CreateAssemblyStepBlocks();
          assemblyStepBlock[0].transform.localPosition =
new Vector3(-
20,assemblyStepBlock[0].transform.localPosition.y,assemblyS
tepBlock[0].transform.localPosition.z);
          assemblyStepBlock[1].transform.localPosition =
new
Vector3(29,assemblyStepBlock[1].transform.localPosition.y,a
ssemblyStepBlock[1].transform.localPosition.z);
          helpingCounter = 1;
        }

  if((part.transform.localPosition.y<=1.7f)&&(!completedMot
ion[1])&&assemblyPlaced)
        {
          helpingCounter =2;

  part.GetComponent<MovingKaloupi>().MoveYDirection();
        }
        else if(helpingCounter == 2&&(!completedMotion[1]))
        {
          completedMotion[1] = true;
        }
        if((part.transform.localPosition.z>=-
0.995f)&&(!completedMotion[2])&&(completedMotion[1]))
        {
          helpingCounter=3;

  part.GetComponent<MovingKaloupi>().MoveOppositeZDirection
();
        }
        else if(helpingCounter == 3&&(!completedMotion[2]))
        {
          completedMotion[2] = true;
          part.transform.localPosition = new
Vector3(part.transform.localPosition.x,part.transform.local
Position.y,-0.995f);
        }

  if((part.transform.localPosition.x<0.955f)&&(!completedMo
tion[3])&&(completedMotion[2]))
        {
          helpingCounter = 4;

  part.GetComponent<MovingKaloupi>().MoveXDirection();
        }
        else if(helpingCounter == 4&&(!completedMotion[3]))
        {
          part.transform.localPosition = new
Vector3(0.955f,part.transform.localPosition.y,part.transfor
m.localPosition.z);
          completedMotion[3] = true;
        }

  if((part.transform.localEulerAngles.y>90)&&(!completedMot
ion[4])&&(completedMotion[3]))
        {
          helpingCounter = 5;

  part.GetComponent<MovingKaloupi>().RotateACwYDirection();
        }
        else if(helpingCounter == 5&&(!completedMotion[4]))
        {
          part.transform.localEulerAngles = new
Vector3(part.transform.localEulerAngles.x,90,part.transform
.localEulerAngles.z);
          completedMotion[4] = true;
        }


  if((part.transform.localPosition.y>=0.62f)&&(!completedMo
tion[5])&&(completedMotion[4]))
        {
          helpingCounter = 6;

  part.GetComponent<MovingKaloupi>().MoveOppositeYDirection
();
        }
        else if(helpingCounter == 6&&(!completedMotion[5]))
        {
          part.transform.localPosition = new
Vector3(part.transform.localPosition.x,0.62f,part.transform
.localPosition.z);
          completedMotion[5] = true;
        }
        if(completedMotion[5]&&!clampsCreated)
        {
          CreateClamps();
          clamp[0].transform.localPosition = new Vector3(-
12,clamp[0].transform.localPosition.y,clamp[0].transform.lo
calPosition.z);
          clamp[1].transform.localPosition = new
Vector3(15,clamp[1].transform.localPosition.y,clamp[1].tran
sform.localPosition.z);
        }

  if(clampsCreated&&(!(clampsPlaced[0]||clampsPlaced[1])))
        {
          for(int i = 0; i< clamp.Length;i++)
          {
            if (clamp [i].transform.localPosition.z >=
124.0f)
            {
              clamp
[i].GetComponent<ClampMover>().MoveDown();
            }
            else
            {
              clampsPlaced[i] = true;
              clamp[i].transform.localPosition = new
Vector3(clamp[i].transform.localPosition.x,clamp[i].transfo
rm.localPosition.y,124.0f);
            }
          }
        }

  if(clampsPlaced[0]&&clampsPlaced[1]&&(!stepBlocksCreated)
)
        {
          CreateStepBlocks();
        }
        if(stepBlocksCreated&&!(stepBlockPlaced[0]&&
stepBlockPlaced[1]))
        {

  if(stepBlock[0].transform.localPosition.x>=173.9f)
          {
            stepBlock[0].transform.localPosition -=new
Vector3(30.0f*Time.deltaTime,0,0);
          }
          else if(!stepBlockPlaced[0])
          {
            stepBlock[0].transform.localPosition =new
Vector3(173.9f,stepBlock[0].transform.localPosition.y,stepB
lock[0].transform.localPosition.z);
            stepBlockPlaced[0] = true;
          }
          if(stepBlock[1].transform.localPosition.x<=-
168.9)
          {
            stepBlock[1].transform.localPosition +=new
Vector3(30.0f*Time.deltaTime,0,0);
          }
          else if(!stepBlockPlaced[1])
          {
            stepBlock[1].transform.localPosition =new
Vector3(-
168.9f,stepBlock[1].transform.localPosition.y,stepBlock[1].
transform.localPosition.z);
            stepBlockPlaced[1] = true;
          }
        }
        if
((stepBlockPlaced[0])&&(stepBlockPlaced[1])&&(!nutsCreated)
)
        {
          Debug.Log("Finally here");
          CreateNuts();
        }
        if(nutsCreated&&!(nutPlaced[0]||nutPlaced[1]))
        {
          for(int i = 0; i< nut.Length;i++)
          {
            if (nut [i].transform.localPosition.z >=
nutScrewStartPoint)
            {
              nut[i].GetComponent<NutMover>().MoveDown();
```

```
            }
            else if(nut [i].transform.localPosition.z >=
nutScrewEndPoint)
            {
              nut[i].GetComponent<NutMover>().Screw();
            }
            else
            {
              nutPlaced[i] = true;
            }
          }
        }
      }
    }

    public void GUIImage()
    {
      GUI.skin = _manager.guiSkin;
      GUI.Label(new Rect(10, 10, Screen.width-20,
(int)(Screen.height/7)), textToShow);
      if(scenario==2 && completedMotion[0])
      {
        ShowButton();
      }
      else if(scenario == 8 && completedParallelMotion)
      {
        ShowButton();
      }

    }

    private void ShowButton()
    {
      if (GUI.Button (new Rect (100, Screen.height - 100,
250, 50), "Choose different Setup"))
      {
        Reset();
        _manager.SwitchState(new
ChoosePartState(_manager));
      }
    }

    public void Reset()
    {
      _manager.DestroyPrefab (part);
      _manager.DestroyPrefab (stud [0]);
      _manager.DestroyPrefab (stud [1]);
    }

    private void CreateAssemblyStepBlocks()
    {
      for(int i=0;i<assemblyStepBlock.Length;i++)
      {

  assemblyStepBlock[i]=_manager.CreatePrefab(18,assemblyIni
tialPosition[i],assemblyInitialRotation[i],stud[i].name);
        assemblyStepBlock[i].transform.localPosition =
assemblyInitialPosition[i];

  assemblyStepBlock[i].transform.localEulerAngles=assemblyI
nitialRotation[i];
      }
    }
    private void CreateClamps()
    {
      for(int i=0; i<clamp.Length;i++)
      {

  clamp[i]=_manager.CreatePrefab(11,initialClapmPosition[i]
,initialClampOrientation[i],stud[i].name);
        clamp[i].transform.localPosition =
initialClapmPosition[i];
        clamp[i].transform.localEulerAngles =
initialClampOrientation[i];
        clamp[i].name = "clamp"+i;
      }
      clampsCreated = true;
    }
    private void CreateStepBlocks()
    {
      for(int i=0; i<stepBlock.Length;i++)
      {

  stepBlock[i]=_manager.CreatePrefab(16,initialStepBlockPos
ition[i],initialStepBlockOrientation[i],stud[i].name);
        stepBlock[i].name = "stepBlock"+i;
        stepBlock[i].transform.localPosition =
initialStepBlockPosition[i];
        stepBlock[i].transform.localEulerAngles =
initialStepBlockOrientation[i];
      }
      stepBlocksCreated = true;
    }

    private void CreateNuts()
    {
      for(int i=0; i<nut.Length;i++)
      {

  nut[i]=_manager.CreatePrefab(13,initialNutPosition,initia
lNutOrientation,stud[i].name);
        nut[i].transform.localPosition =
initialNutPosition;
        nut[i].transform.localEulerAngles =
initialNutOrientation;
        nut[i].name = "nut"+i;
      }
      nutsCreated = true;
    }


    IEnumerator showText()
    {
      string str;
      if (scenario == 2)
      {
        str = "Use a gauge block(10x100x300) to support the
part and hold it with the clamps.";
      }
      else
      {
        str = "Use two gauge blocks(10x100x300) to support
the part and hold it with the clamps.";
      }
      char [] temp = str.ToCharArray();
      int counter = 0;

      while (counter < temp.Length)
      {
        textToShow += temp[counter];
        counter++;
        yield return new WaitForSeconds(0.01f);
      }

      yield return new WaitForSeconds (5.0f);
      textToShow = "";
    }
  }
}
```

• Initial Screen State

```
using UnityEngine;
using System.Collections;
using Assets.Codes.Interface;
using Assets.Codes.ComponentScripts;

namespace Assets.Codes.States
{
  public class InitialScreenState : IState
  {
    private MainManager _manager;
    private GameObject ImageTargetWatcher;
    public string textToShow;
    private int counter;
    private int countdown;

    public InitialScreenState(MainManager manager)
    {
      counter = 1;
      ImageTargetWatcher =
GameObject.FindWithTag("imageTarget");
      _manager = manager;
      Debug.Log("Initial State now active");
      countdown = 5;
    }
    public void Action()
    {}

    public void GUIImage()
    {

  if((!ImageTargetWatcher.GetComponent<ImageTargetWatcher>(
).targetFound) &&(counter==1))
      {
        counter++;
        _manager.StartCoroutine(waitText());
      }
      else
if(ImageTargetWatcher.GetComponent<ImageTargetWatcher>().ta
rgetFound&&(counter ==2))
      {
        _manager.guiSkin.label.normal.textColor =
Color.cyan;
        counter=1;
        _manager.StartCoroutine(wait());
      }
      GUI.skin = _manager.guiSkin;
      GUI.Label(new Rect((Screen.width/2)-200, 10, 600,
80), textToShow);
```

```
    }
    public void Reset(){}

    IEnumerator waitText()
    {
      while
(!ImageTargetWatcher.GetComponent<ImageTargetWatcher>().tar
getFound) {
        string temp = "Waiting For Target Recognition";
        for (int i=1; i<= 5; i++) {
          if
(!ImageTargetWatcher.GetComponent<ImageTargetWatcher>
().targetFound) {
            textToShow = temp;
            temp = temp + " .";
          }
          yield return new WaitForSeconds (.5f);
        }
      }
    }
    IEnumerator wait()
    {
      for (int i = 5; i >= 0; i--)
        {
        textToShow = "Target Recognized, continue in " +
countdown;
        countdown = i;
        yield return new WaitForSeconds (1.0f);
        }
      _manager.guiSkin.label.normal.textColor =
Color.white;
      _manager.SwitchState (new ChoosePartState
(_manager));
    }
  }
}
```

• OrientationState

```
using UnityEngine;
using System.Collections;
using Assets.Codes.Interface;
using Assets.Codes.ComponentScripts;
using Assets.Codes.States;

namespace Assets.Codes.States
{
  public class OrientationState : IState
  {
    private GameObject Part;
    private GameObject OrientationCube;
    private MovingKaloupi PartSetter;
    private MovingKaloupi CubeSetter;
    private Rect HolderBox;
    private int widthSpan;
    private int heightSpan;
    private int xPosition;
    private int yPosition;
    private MainManager _manager;
    private bool proceedQuestion;
    private string [] choises;

    public static int scenario;

    public OrientationState(MainManager manager)
    {
      scenario = 0;
      _manager = manager;
      widthSpan = 150;
      heightSpan = 50;
      xPosition = 10;
      yPosition = xPosition;
      HolderBox = new Rect (xPosition, yPosition,
widthSpan, heightSpan * 6);
      Part = GameObject.Find (ChoosePartState.finalName);
      OrientationCube = _manager.CreatePrefab (2, new
Vector3 (60f, 110f, 0f), ChoosePartState.finalName);
      OrientationCube.transform.parent =
GameObject.FindWithTag ("imageTarget").transform;
      PartSetter = Part.GetComponent<MovingKaloupi> ();
      CubeSetter =
OrientationCube.GetComponent<MovingKaloupi> ();
      proceedQuestion = false;
      choises = new string[]{"X","Y","Z","-X","-
Z","Default"};
    }
    public void Action(){}

    public void GUIImage()
    {
      GUI.Box (HolderBox, "");
      GUI.skin = _manager.guiSkin;
      for (int i = 1; i<=6; i++)
      {
        if(GUI.Button(new Rect(xPosition,(i-
1)*heightSpan+yPosition,widthSpan,heightSpan),"Choice :
"+choises[i-1]))
        {
          scenario = ((ChoosePartState.scenarioNumber-1)*6)
+ i;
          proceedQuestion = true;
          PartSetter.RotateToPlane(i);
          CubeSetter.RotateToPlane(i);
        }
      }
      Rect textHolder = new Rect ((int)(Screen.width / 2)-
200, Screen.height-(int)(Screen.height/7), 400,
(int)(Screen.height/8));
      GUI.Box (textHolder, "Choose part side that will be
facing towards the table. ");
      if (GUI.Button (new Rect ((int)(Screen.width/2)-220,
Screen.height-(int)(Screen.height/7)-
(int)(Screen.height/8)-10, 200, 50), "Choose another part
for setup"))
      {
        _manager.DestroyPrefab(Part);
        _manager.DestroyPrefab(OrientationCube);
        _manager.SwitchState(new
ChoosePartState(_manager));
      }
      if (proceedQuestion) {
        if (GUI.Button (new Rect ((int)(Screen.width/2)+20,
Screen.height-(int)(Screen.height/7)-
(int)(Screen.height/8)-10, 150, 50), "Continue"))
        {
          _manager.DestroyPrefab(OrientationCube);
          _manager.SwitchState(new SetupState(_manager));
        }
      }
    }

    public void Reset(){}
  }
}
```

• SetupChuckState

```
using UnityEngine;
using System.Collections;
using Assets.Codes.Interface;
using Assets.Codes.ComponentScripts;
using Assets.Codes.States;

namespace Assets.Codes.States
{
  public class SetupChuckState : IState
  {

    private MainManager _manager;
    private GameObject holdingDevice;
    private GameObject part;
    private string textToShow;
    private bool showImage;
    private Texture2D chuckImage;

    public SetupChuckState(MainManager manager)
    {
      _manager = manager;
      _manager.StartCoroutine (showText ());
      showImage = false;
      chuckImage = _manager.image [0];
    }
    public void Action(){}

    public void GUIImage()
    {
      GUI.skin = _manager.guiSkin;
      GUI.Label(new Rect(10, 10, Screen.width-20,
(int)(Screen.height/7)), textToShow);
      if (showImage)
      {
        GUI.Label(new Rect((int)((Screen.width-
(int)(chuckImage.width/2))/2),(int)((Screen.height-
(int)(chuckImage.height/2))/2),(int)(chuckImage.width/2),(i
nt)(chuckImage.height/2)),chuckImage);
        if (GUI.Button (new Rect (100, Screen.height - 100,
150, 50), "Back"))
        {
          _manager.SwitchState (new OrientationState
(_manager));
        }
        if (GUI.Button (new Rect (Screen.width - 200,
Screen.height - 100, 150, 50), "Proceed"))
        {
          _manager.SwitchState (new SetupChuckStateTwo
(_manager));
        }
      }
    }
```

```
    public void Reset()
    {}

    IEnumerator showText()
    {
      string str = "First you need to place the chuck on
the table.";
      char [] temp = str.ToCharArray();
      int counter = 0;

      while (counter < temp.Length)
      {
        textToShow += temp[counter];
        counter++;
        yield return new WaitForSeconds(0.01f);
      }

      yield return new WaitForSeconds (1.0f);
      showImage = true;
      }
    }
  }
```

- Setup Chuck Two

```
using UnityEngine;
using System.Collections;
using Assets.Codes.Interface;
using Assets.Codes.ComponentScripts;
using Assets.Codes.States;

namespace Assets.Codes.States
{
  public class SetupChuckStateTwo : IState
  {

    private MainManager _manager;
    private GameObject demoObject;
    private Vector3 initialPosition;
    private Vector3 initialRotaion;
    private string textToShow;
    private bool nextState;
    private int scenario;

    public SetupChuckStateTwo(MainManager manager)
    {
      _manager = manager;
      nextState = false;
      scenario = OrientationState.scenario;
      initialPosition = new Vector3 (196.3181f,45.91039f,-
165.8001f);
      initialRotaion = new Vector3 (0,0,80);
      demoObject = _manager.CreatePrefab
(9,initialPosition,initialRotaion);
      _manager.StartCoroutine (showText ());
    }
    public void Action(){}

    public void GUIImage()
    {
      GUI.skin = _manager.guiSkin;
      GUI.Label(new Rect(10, 10, Screen.width-20,
(int)(Screen.height/7)), textToShow);
      if (GUI.Button (new Rect (100, Screen.height - 100,
150, 50), "Back"))
      {
        _manager.DestroyPrefab(demoObject);
        _manager.SwitchState (new OrientationState
(_manager));
      }
      if
((demoObject.GetComponent<DemoStatus>().demoFinished) &&
(nextState))
      {
        _manager.DestroyPrefab(demoObject);
        if((scenario==7)||(scenario==10)||(scenario==11))
        {
        _manager.SwitchState (new ChuckSetupThree
(_manager));
        }

  if((scenario==1)||(scenario==3)||(scenario==4)||(scenario
==5)||(scenario==6)||(scenario==9)||(scenario==12))
        {
          _manager.SwitchState (new ViseSetupThree
(_manager));
        }
        if (scenario == 2 || scenario == 8)
        {
          _manager.SwitchState(new
ChuckSetupThree(_manager));
        }
      }
    }

    public void Reset(){}

    IEnumerator showText()
    {
      string str;
      if (scenario == 2 || scenario == 8)
      {
        str ="Adjust two studs to their slides.";
      }
      else
      {
        str = "Adjust three studs to their slides.";
      }
      char [] temp = str.ToCharArray();
      int counter = 0;

      while (counter < temp.Length)
      {
        textToShow += temp[counter];
        counter++;
        yield return new WaitForSeconds(0.01f);
      }

      yield return new WaitForSeconds (6.0f);
      nextState = true;
    }

  }
}
```

- Setup State

```
using UnityEngine;
using System.Collections;
using Assets.Codes.Interface;
using Assets.Codes.ComponentScripts;
using Assets.Codes.States;

namespace Assets.Codes.States
{
  public class SetupState : IState
  {

    private MainManager _manager;
    private int scenario;

    public SetupState(MainManager manager)
    {
      _manager = manager;
      scenario = OrientationState.scenario;
    }
    public void Action()
    {
      if ((scenario == 7) || (scenario == 10) || (scenario
== 11))
      {
          _manager.SwitchState(new
SetupChuckState(_manager));
      }
      else if ((scenario == 1) || (scenario == 3)||
(scenario == 4)|| (scenario == 5)|| (scenario == 6)||
(scenario == 9)|| (scenario == 12))
      {
        _manager.SwitchState(new ViseSetup(_manager));
      }

      else if(scenario==2||scenario==8)
      {
        _manager.SwitchState(new
SetupChuckStateTwo(_manager));
      }
    }

    public void GUIImage()
    {}

    public void Reset()
    {}
  }
}
```

- Vise Setup

```
using UnityEngine;
using System.Collections;
using Assets.Codes.Interface;
using Assets.Codes.ComponentScripts;
using Assets.Codes.States;

namespace Assets.Codes.States
{
  public class ViseSetup : IState
  {

    private MainManager _manager;
    private string textToShow;
```

```
    private bool showImage;
    private Texture2D chuckImage;

    public ViseSetup(MainManager manager)
    {
      _manager = manager;
      _manager.StartCoroutine (showText ());
      showImage = false;
      chuckImage = _manager.image [1];
    }
    public void Action(){}

    public void GUIImage()
    {
      GUI.skin = _manager.guiSkin;
      GUI.Label(new Rect(10, 10, Screen.width-20,
(int)(Screen.height/7)), textToShow);
      if (showImage)
      {
        GUI.Label(new Rect((int)((Screen.width-
(int)(chuckImage.width/2))/2),(int)((Screen.height-
(int)(chuckImage.height/2))/2),(int)(chuckImage.width/2),(i
nt)(chuckImage.height/2)),chuckImage);
        if (GUI.Button (new Rect (100, Screen.height - 100,
150, 50), "Back"))
        {
          _manager.SwitchState (new OrientationState
(_manager));
        }
        if (GUI.Button (new Rect (Screen.width - 200,
Screen.height - 100, 150, 50), "Proceed"))
        {
          _manager.SwitchState (new SetupChuckStateTwo
(_manager));
        }
      }
    }

    public void Reset()
    {}

    IEnumerator showText()
    {
      string str = "First you need to place the vise on the
table.";
      char [] temp = str.ToCharArray();
      int counter = 0;

      while (counter < temp.Length)
      {
        textToShow += temp[counter];
        counter++;
        yield return new WaitForSeconds(0.01f);
      }

      yield return new WaitForSeconds (1.0f);
      showImage = true;
    }
  }
}
```

- ViseSetupFinal

```
using UnityEngine;
using System.Collections;
using Assets.Codes.Interface;
using Assets.Codes.ComponentScripts;
using Assets.Codes.States;

namespace Assets.Codes.States
{
  public class ViseSetupFinal : IState
  {

    private MainManager _manager;
    private GameObject vise;
    private GameObject part;
    private GameObject jaw;
    private GameObject videoObject;
    private string textToShow;
    private int scenario;
    private bool[] completedMotion = new bool[6];
    private int helpingCounter;

    private Vector3 videoInitialPosition;
    private Vector3 videoInitialRotation;
    private Vector3 videoInitialScale;

    public static bool videoFinished;

    public ViseSetupFinal(MainManager manager)
    {
      _manager = manager;
      _manager.StartCoroutine (showText ());
      vise = GameObject.Find (ViseSetupFour.viseName);
      part = GameObject.Find (ChoosePartState.finalName);
      scenario = OrientationState.scenario;
      helpingCounter = 0;
      for(int i=0; i<completedMotion.Length;i++)
      {
        completedMotion[i] =false;
      }
      jaw = GameObject.FindWithTag("movingJaw");
      videoInitialPosition = new Vector3 (-0.56f,-
0.3012f,0.1605f);
      videoInitialRotation = new Vector3 (1.4283f,180,-
3.37f);
      videoInitialScale = new Vector3
(0.05f,0.0025f,0.05f);
      videoFinished = false;
    }
    public void Action()
    {
      if(videoFinished)
      {
        _manager.DestroyPrefab(videoObject);
      }
      if(scenario == 1)
      {

  if((part.transform.localPosition.y<=1.1f)&&(!completedMot
ion[0]))
        {

  part.GetComponent<MovingKaloupi>().MoveYDirection();
        }
        else
        {
          completedMotion[0] = true;
        }
        if((part.transform.localPosition.z>=-
0.7f)&&(!completedMotion[1])&&(completedMotion[0]))
        {

  part.GetComponent<MovingKaloupi>().MoveOppositeZDirection
();
          helpingCounter = 1;
        }
        else if(helpingCounter == 1&&(!completedMotion[1]))
        {
          completedMotion[1] = true;
        }

  if((part.transform.localPosition.x<0.549f)&&(!completedMo
tion[2])&&(completedMotion[1]))
        {
          helpingCounter = 2;

  part.GetComponent<MovingKaloupi>().MoveXDirection();
        }
        else if(helpingCounter == 2&&(!completedMotion[2]))
        {
          part.transform.localPosition = new
Vector3(0.549f,part.transform.localPosition.y,part.transfor
m.localPosition.z);
          completedMotion[2] = true;
        }

  if((part.transform.localEulerAngles.y<=90.0f)&&(!complete
dMotion[3])&&(completedMotion[2]))
        {
          helpingCounter = 3;

  part.GetComponent<MovingKaloupi>().RotateCwYDirection();
        }
        else if(helpingCounter == 3&&(!completedMotion[3]))
        {
          completedMotion[3] = true;
        }

  if((part.transform.localPosition.y>=0.81f)&&(!completedMo
tion[4])&&(completedMotion[3]))
        {
          helpingCounter = 4;

  part.GetComponent<MovingKaloupi>().MoveOppositeYDirection
();
        }
        else if(helpingCounter == 4&&(!completedMotion[4]))
        {
          completedMotion[4] = true;
          CreateVideoObject();
        }
        if ((jaw.transform.localPosition.x>=-
5.7f)&&(!completedMotion[5])&&(completedMotion[4]))
        {
          helpingCounter = 5;
          vise.GetComponent<ViseMover>().CloseViseJaw();
        }
        else if(helpingCounter == 5&&(!completedMotion[5]))
```

```
        {
          completedMotion[5] = true;
        }
      }
      if(scenario==4)
      {

  if((part.transform.localPosition.y<=1.1f)&&(!completedMot
ion[0]))
        {

  part.GetComponent<MovingKaloupi>().MoveYDirection();
        }
        else
        {
          completedMotion[0] = true;
        }
        if((part.transform.localPosition.z>=-
0.7f)&&(!completedMotion[1])&&(completedMotion[0]))
        {

  part.GetComponent<MovingKaloupi>().MoveOppositeZDirection
();
          helpingCounter = 1;
        }
        else if(helpingCounter == 1&&(!completedMotion[1]))
        {
          completedMotion[1] = true;
        }

  if((part.transform.localPosition.x<0.54f)&&(!completedMot
ion[2])&&(completedMotion[1]))
        {
          helpingCounter = 2;

  part.GetComponent<MovingKaloupi>().MoveXDirection();
        }
        else if(helpingCounter == 2&&(!completedMotion[2]))
        {
          part.transform.localPosition = new
Vector3(0.54f,part.transform.localPosition.y,part.transform
.localPosition.z);
          completedMotion[2] = true;
        }

  if((part.transform.localEulerAngles.y<=90.0f)&&(!complete
dMotion[3])&&(completedMotion[2]))
        {
          helpingCounter = 3;

  part.GetComponent<MovingKaloupi>().RotateCwYDirection();
        }
        else if(helpingCounter == 3&&(!completedMotion[3]))
        {
          completedMotion[3] = true;
        }

  if((part.transform.localPosition.y>=0.81f)&&(!completedMo
tion[4])&&(completedMotion[3]))
        {
          helpingCounter = 4;

  part.GetComponent<MovingKaloupi>().MoveOppositeYDirection
();
        }
        else if(helpingCounter == 4&&(!completedMotion[4]))
        {
          completedMotion[4] = true;
          CreateVideoObject();
        }
        if ((jaw.transform.localPosition.x>=-
5.7f)&&(!completedMotion[5])&&(completedMotion[4]))
        {
          helpingCounter = 5;
          vise.GetComponent<ViseMover>().CloseViseJaw();
        }
        else if(helpingCounter == 5&&(!completedMotion[5]))
        {
          completedMotion[5] = true;
        }
      }
      if(scenario == 3||scenario==5)
      {

  if((part.transform.localPosition.y<=1.1f)&&(!completedMot
ion[0]))
        {

  part.GetComponent<MovingKaloupi>().MoveYDirection();
        }
        else
        {
          completedMotion[0] = true;
        }
        if((part.transform.localPosition.z>=-
0.53f)&&(!completedMotion[1])&&(completedMotion[0]))
        {

  part.GetComponent<MovingKaloupi>().MoveOppositeZDirection
();
          helpingCounter = 1;
        }
        else if(helpingCounter == 1&&(!completedMotion[1]))
        {
          completedMotion[1] = true;
        }

  if((part.transform.localPosition.x<0.555f)&&(!completedMo
tion[2])&&(completedMotion[1]))
        {
          helpingCounter = 2;

  part.GetComponent<MovingKaloupi>().MoveXDirection();
        }
        else if(helpingCounter == 2&&(!completedMotion[2]))
        {
          part.transform.localPosition = new
Vector3(0.555f,part.transform.localPosition.y,part.transfor
m.localPosition.z);
          completedMotion[2] = true;
        }


  if((part.transform.localPosition.y>=0.79f)&&(!completedMo
tion[3])&&(completedMotion[2]))
        {
          helpingCounter = 3;

  part.GetComponent<MovingKaloupi>().MoveOppositeYDirection
();
        }
        else if(helpingCounter == 3&&(!completedMotion[3]))
        {
          completedMotion[3] = true;
          CreateVideoObject();
        }
        if ((jaw.transform.localPosition.x>=-
1.21f)&&(!completedMotion[4])&&(completedMotion[3]))
        {
          helpingCounter = 4;
          vise.GetComponent<ViseMover>().CloseViseJaw();
        }
        else if(helpingCounter == 4&&(!completedMotion[4]))
        {
          completedMotion[4] = true;
        }
      }
      if(scenario == 6)
      {

  if((part.transform.localPosition.y<=1.1f)&&(!completedMot
ion[0]))
        {

  part.GetComponent<MovingKaloupi>().MoveYDirection();
        }
        else
        {
          completedMotion[0] = true;
        }
        if((part.transform.localPosition.z>=-
0.53f)&&(!completedMotion[1])&&(completedMotion[0]))
        {

  part.GetComponent<MovingKaloupi>().MoveOppositeZDirection
();
          helpingCounter = 1;
        }
        else if(helpingCounter == 1&&(!completedMotion[1]))
        {
          completedMotion[1] = true;
        }

  if((part.transform.localPosition.x<0.435f)&&(!completedMo
tion[2])&&(completedMotion[1]))
        {
          helpingCounter = 2;

  part.GetComponent<MovingKaloupi>().MoveXDirection();
        }
        else if(helpingCounter == 2&&(!completedMotion[2]))
        {
          part.transform.localPosition = new
Vector3(0.435f,part.transform.localPosition.y,part.transfor
m.localPosition.z);
          completedMotion[2] = true;
        }
```

```
  if((part.transform.localPosition.y>=0.571f)&&(!completedM
otion[3])&&(completedMotion[2]))
        {
          helpingCounter = 3;

  part.GetComponent<MovingKaloupi>().MoveOppositeYDirection
();
        }
        else if(helpingCounter == 3&&(!completedMotion[3]))
        {
          completedMotion[3] = true;
          CreateVideoObject();
        }
        if ((jaw.transform.localPosition.x>=-
50.0f)&&(!completedMotion[4])&&(completedMotion[3]))
        {
          helpingCounter = 4;
          vise.GetComponent<ViseMover>().CloseViseJaw();
        }
        else if(helpingCounter == 4&&(!completedMotion[4]))
        {
          completedMotion[4] = true;
        }
      }
      if(scenario == 9)
      {

  if((part.transform.localPosition.y<=1.1f)&&(!completedMot
ion[0]))
        {

  part.GetComponent<MovingKaloupi>().MoveYDirection();
        }
        else
        {
          completedMotion[0] = true;
        }
        if((part.transform.localPosition.z>=-
0.8f)&&(!completedMotion[1])&&(completedMotion[0]))
        {

  part.GetComponent<MovingKaloupi>().MoveOppositeZDirection
();
          helpingCounter = 1;
        }
        else if(helpingCounter == 1&&(!completedMotion[1]))
        {
          completedMotion[1] = true;
        }

  if((part.transform.localPosition.x<0.834f)&&(!completedMo
tion[2])&&(completedMotion[1]))
        {
          helpingCounter = 2;

  part.GetComponent<MovingKaloupi>().MoveXDirection();
        }
        else if(helpingCounter == 2&&(!completedMotion[2]))
        {
          part.transform.localPosition = new
Vector3(0.834f,part.transform.localPosition.y,part.transfor
m.localPosition.z);
          completedMotion[2] = true;
        }
        if(((360.0f-
part.transform.localEulerAngles.y)<=90.0f||(int)part.transf
orm.localEulerAngles.y==0)&&(!completedMotion[3])&&(complet
edMotion[2]))
        {
          helpingCounter = 3;

  part.GetComponent<MovingKaloupi>().RotateACwYDirection();
        }
        else if(helpingCounter == 3&&(!completedMotion[3]))
        {
          completedMotion[3] = true;
        }

  if((part.transform.localPosition.y>=0.66f)&&(!completedMo
tion[4])&&(completedMotion[3]))
        {
          helpingCounter = 4;

  part.GetComponent<MovingKaloupi>().MoveOppositeYDirection
();
        }
        else if(helpingCounter == 4&&(!completedMotion[4]))
        {
          completedMotion[4] = true;
          CreateVideoObject();
        }
        if
((jaw.transform.localPosition.x>7.0f)&&(!completedMotion[5]
)&&(completedMotion[4]))
        {
          helpingCounter = 5;
          vise.GetComponent<ViseMover>().CloseViseJaw();
        }
        else if(helpingCounter == 5&&(!completedMotion[5]))
        {
          completedMotion[5] = true;
        }
      }
      if(scenario == 12)
      {

  if((part.transform.localPosition.y<=1.1f)&&(!completedMot
ion[0]))
        {

  part.GetComponent<MovingKaloupi>().MoveYDirection();
        }
        else
        {
          completedMotion[0] = true;
        }
        if((part.transform.localPosition.z>=-
0.53f)&&(!completedMotion[1])&&(completedMotion[0]))
        {

  part.GetComponent<MovingKaloupi>().MoveOppositeZDirection
();
          helpingCounter = 1;
        }
        else if(helpingCounter == 1&&(!completedMotion[1]))
        {
          completedMotion[1] = true;
        }

  if((part.transform.localPosition.x<0.465f)&&(!completedMo
tion[2])&&(completedMotion[1]))
        {
          helpingCounter = 2;

  part.GetComponent<MovingKaloupi>().MoveXDirection();
        }
        else if(helpingCounter == 2&&(!completedMotion[2]))
        {
          part.transform.localPosition = new
Vector3(0.465f,part.transform.localPosition.y,part.transfor
m.localPosition.z);
          completedMotion[2] = true;
        }


  if((part.transform.localPosition.y>=0.57f)&&(!completedMo
tion[3])&&(completedMotion[2]))
        {
          helpingCounter = 3;

  part.GetComponent<MovingKaloupi>().MoveOppositeYDirection
();
        }
        else if(helpingCounter == 3&&(!completedMotion[3]))
        {
          completedMotion[3] = true;
          CreateVideoObject();
        }
        if ((jaw.transform.localPosition.x>=-
29.5f)&&(!completedMotion[4])&&(completedMotion[3]))
        {
          helpingCounter = 4;
          vise.GetComponent<ViseMover>().CloseViseJaw();
        }
        else if(helpingCounter == 4&&(!completedMotion[4]))
        {
          completedMotion[4] = true;
        }
      }
    }

    public void GUIImage()
    {
      GUI.skin = _manager.guiSkin;
      GUI.Label(new Rect(10, 10, Screen.width-20,
(int)(Screen.height/7)), textToShow);
      if(videoFinished)
      {
        if (GUI.Button (new Rect (100, Screen.height - 100,
250, 50), "Choose different Setup"))
        {
          Reset();
          _manager.SwitchState(new
ChoosePartState(_manager));
        }
        if (GUI.Button (new Rect (Screen.width - 200,
Screen.height - 100, 150, 50), "Play Video"))
        {
          CreateVideoObject();
```

```
        }
      }
    }

    public void Reset()
    {
      _manager.DestroyPrefab (part);
      _manager.DestroyPrefab (vise);
    }

    private void CreateVideoObject()
    {
      videoFinished = false;
      videoObject = _manager.CreatePrefab
(17,videoInitialPosition,videoInitialRotation,"BackgroundPl
ane");
      videoObject.transform.localPosition =
videoInitialPosition;
      videoObject.transform.localEulerAngles =
videoInitialRotation;
      videoObject.transform.localScale = videoInitialScale;

    }

    IEnumerator showText()
    {
      string str = "Set the part on the vise and hold it.";
      char [] temp = str.ToCharArray();
      int counter = 0;

      while (counter < temp.Length)
      {
        textToShow += temp[counter];
        counter++;
        yield return new WaitForSeconds(0.01f);
      }

      yield return new WaitForSeconds (5.0f);
      textToShow = "";
    }
  }
}
```

• ViseSetupFour

```
using UnityEngine;
using System.Collections;
using Assets.Codes.Interface;
using Assets.Codes.ComponentScripts;
using Assets.Codes.States;

namespace Assets.Codes.States
{
  public class ViseSetupFour : IState
  {

    private MainManager _manager;

    private GameObject part;
    private GameObject vise;
    private GameObject[] stud = new GameObject[3];
    private GameObject[] clamp = new GameObject[3];
    private GameObject[] stepBlock = new GameObject[3];
    private GameObject[] nut = new GameObject[3];

    private bool nutsCreated;
    private bool[] nutPlaced= new bool[3];
    private bool[] studPlaced = new bool[3];
    private bool[] studPlacedTwo = new bool[2];
    private bool[] clampPlaced = new bool[3];
    private bool[] clampPlacedTwo = new bool[3];
    private bool visePlaced;
    private bool clampsCreated;
    private bool stepBlocksCreated;
    private bool[] stepBlockPlaced = new bool[3];

    private Vector3 initilaVisePosition;
    private Vector3 initialViseOrientation;
    private Vector3 initialClapmPosition;
    private Vector3 [] initialClampOrientation = new
Vector3[3];
    private Vector3[] initialStepBlockPosition = new
Vector3[3];
    private Vector3[] initialStepBlockOrientation = new
Vector3[3];
    private Vector3 initialNutPosition;
    private Vector3 initialNutOrientation;
    private float[] finalZPositionStepBlock = new float[3];
    private float nutScrewStartPoint;
    private float nutScrewEndPoint;

    private string textToShow;

    public static string viseName;

    public ViseSetupFour(MainManager manager)
    {
      _manager = manager;
      initilaVisePosition = new Vector3 (425.0f, 0.00f, -
116.0f);
      initialViseOrientation = new Vector3 (-90.0f,0,0);

      initialClapmPosition = new Vector3 (0, 0, 190.0f);
      initialClampOrientation [0] = new Vector3 (0, 0, 30);
      initialClampOrientation [1] = new Vector3 (0, 0,
212);
      initialClampOrientation [2] = new Vector3 (0, 0,
126);

      initialStepBlockPosition [0] = new Vector3 (421.0f,
0, -279f);
      initialStepBlockPosition [1] = new Vector3 (-67.4f,
0, 71.52f);
      initialStepBlockPosition [2] = new Vector3 (-117.2f,
0, -383.2f);
      initialStepBlockOrientation [0] = new Vector3
(0,300,0);
      initialStepBlockOrientation [1] = new Vector3
(0,125,0);
      initialStepBlockOrientation [2] = new Vector3
(0,36,0);
      finalZPositionStepBlock [0] = 53.0f;
      finalZPositionStepBlock [1] = 81.0f;
      finalZPositionStepBlock [2] = 65.0f;

      initialNutPosition = new Vector3 (0, 0, 120);
      nutScrewStartPoint = 81.0f;
      nutScrewEndPoint = 56.5f;

      visePlaced = false;
      clampsCreated = false;
      stepBlocksCreated = false;
      nutsCreated = false;
      viseName = "vise";

      part = GameObject.Find (ChoosePartState.finalName);
      stud [0] = GameObject.Find
(ViseSetupThree.viseStudFirstSlot);
      stud [1] = GameObject.Find
(ViseSetupThree.viseStudSecondSlot);
      stud[2] = GameObject.Find
(ViseSetupThree.viseStudThirdSlot);
      vise = _manager.CreatePrefab (15,
initilaVisePosition, initialViseOrientation);
      vise.name = viseName;

      for (int i=0; i< studPlaced.Length;i++)
      {
        studPlaced[i] = false;
      }
      for(int i = 0; i< studPlacedTwo.Length;i++)
      {
        studPlacedTwo[i] = false;
      }
      _manager.StartCoroutine (showText ());
    }
    public void Action()
    {
      if (vise.transform.localPosition.x >=
0.625f&&(!visePlaced))
      {
        vise.GetComponent<ViseMover>().MoveForward();
      }
      else if(!visePlaced)
      {
        vise.transform.localPosition = new
Vector3(0.625f,vise.transform.localPosition.y,vise.transfor
m.localPosition.z);
        visePlaced = true;
      }
      if (visePlaced)
      {
        if (stud [0].transform.localPosition.z <= -1.3f)
        {
          stud [0].GetComponent<StudMover> ().MoveRight ();
        }
        else
        {
          studPlaced[0] = true;
        }
        if (stud [1].transform.localPosition.z >= -0.1f)
        {
          stud [1].GetComponent<StudMover> ().MoveLeft ();
        }
        else
        {
          studPlaced[1] = true;
        }
        if (stud [2].transform.localPosition.z <= -1.3f)
        {
```

```
          stud [2].GetComponent<StudMover> ().MoveRight ();
        }
        else
        {
          studPlaced[2] = true;
        }
      }

  if((studPlaced[0])&&(studPlaced[1])&&(studPlaced[2])&&(!c
lampsCreated))
      {
        CreateClamps();
      }
      if(clampsCreated)
      {
        for(int i = 0; i< clamp.Length;i++)
        {
          if (clamp [i].transform.localPosition.z >= 35)
          {
            clamp
[i].GetComponent<ClampMover>().MoveDown();
          }
          else
          {
            clampPlaced[i] = true;
          }
        }
      }

  if((clampPlaced[0])&&(clampPlaced[1])&&(clampPlaced[2]))
      {
        if (stud [0].transform.localPosition.z <= -1.08f)
        {
          stud [0].GetComponent<StudMover> ().MoveRight ();
        }
        else
        {
          studPlacedTwo[0] = true;
        }
        if (stud [2].transform.localPosition.z <= -1.1f)
        {
          stud [2].GetComponent<StudMover> ().MoveRight ();
        }
        else
        {
          studPlacedTwo[1] = true;
        }
      }
      if(studPlacedTwo[0]&&studPlacedTwo[1])
      {
        for(int i = 0; i< clamp.Length;i++)
        {
          if (clamp [i].transform.localPosition.z >= 30)
          {
            clamp
[i].GetComponent<ClampMover>().MoveDown();
          }
          else
          {
            clampPlacedTwo[i] = true;
          }
        }
      }

      if
((clampPlacedTwo[0])&&(clampPlacedTwo[1])&&(clampPlacedTwo[
2])&&(!stepBlocksCreated))
      {
        CreateStepBlocks();
      }
      if(stepBlocksCreated)
      {
        for(int i = 0; i< stepBlock.Length;i++)
        {
          if (stepBlock
[i].GetComponentInChildren<StepBlockMover>().zPosition <=
finalZPositionStepBlock[i])
          {
            stepBlock
[i].GetComponentInChildren<StepBlockMover>().MoveForward();
          }
          else
          {
            stepBlockPlaced[i] = true;
          }
        }
      }
      if
((stepBlockPlaced[0])&&(stepBlockPlaced[1])&&(stepBlockPlac
ed[2])&&(!nutsCreated))
      {
        CreateNuts();
      }
      if(nutsCreated)
      {
        for(int i = 0; i< nut.Length;i++)
        {
          if (nut [i].transform.localPosition.z >=
nutScrewStartPoint)
          {
            nut[i].GetComponent<NutMover>().MoveDown();
          }
          else if(nut [i].transform.localPosition.z >=
nutScrewEndPoint)
          {
            nut[i].GetComponent<NutMover>().Screw();
          }
          else
          {
            nutPlaced[i] = true;
          }
        }
      }
    }

    public void GUIImage()
    {
      GUI.skin = _manager.guiSkin;
      GUI.Label(new Rect(10, 10, Screen.width-20,
(int)(Screen.height/7)), textToShow);
      if(nutPlaced[2])
      {
        if (GUI.Button (new Rect (100, Screen.height - 100,
150, 50), "Back"))
        {
          MakeEveryGameObjectChildOfVise();
          _manager.DestroyPrefab(vise);
          _manager.SwitchState(new
ViseSetupThree(_manager));
        }
        if (GUI.Button (new Rect (Screen.width - 200,
Screen.height - 100, 150, 50), "Proceed"))
        {
          MakeEveryGameObjectChildOfVise();
          _manager.SwitchState(new
ViseSetupFinal(_manager));
        }
      }
    }

    public void Reset()
    {}

    private void MakeEveryGameObjectChildOfVise()
    {
      for(int i=0; i<stud.Length;i++)
      {
        stud[i].transform.parent = vise.transform;
      }
    }

    private void CreateClamps()
    {
      for(int i=0; i<clamp.Length;i++)
      {

  clamp[i]=_manager.CreatePrefab(11,initialClapmPosition,in
itialClampOrientation[i],stud[i].name);
        clamp[i].transform.localPosition =
initialClapmPosition;
        clamp[i].transform.localEulerAngles =
initialClampOrientation[i];
        clamp[i].name = "clamp"+i;
      }
      clampsCreated = true;
    }

    private void CreateStepBlocks()
    {
      for(int i=0; i<stepBlock.Length;i++)
      {

  stepBlock[i]=_manager.CreatePrefab(16,initialStepBlockPos
ition[i],initialStepBlockOrientation[i]);
        stepBlock[i].name = "stepBlock"+i;
        stepBlock[i].transform.parent = stud[i].transform;
      }
      stepBlocksCreated = true;
    }

    private void CreateNuts()
    {
      for(int i=0; i<nut.Length;i++)
      {

  nut[i]=_manager.CreatePrefab(13,initialNutPosition,initia
lNutOrientation,stud[i].name);
        nut[i].transform.localPosition =
initialNutPosition;
```

```
        nut[i].transform.localEulerAngles =
initialNutOrientation;
        nut[i].name = "nut"+i;
      }
      nutsCreated = true;
    }

    IEnumerator showText()
    {
      string str = "Place the vise on the table and use the
clamps to hold it.";
      char [] temp = str.ToCharArray();
      int counter = 0;
      while (counter < temp.Length)
      {
        textToShow += temp[counter];
        counter++;
        yield return new WaitForSeconds(0.01f);
      }
      yield return new WaitForSeconds (5.0f);
      textToShow = "";
    }
  }
}
```

• ViseSetupThree

```
using UnityEngine;
using System.Collections;
using Assets.Codes.Interface;
using Assets.Codes.ComponentScripts;
using Assets.Codes.States;

namespace Assets.Codes.States
{
  public class ViseSetupThree : IState
  {

    private MainManager _manager;
    private GameObject videoObject;
    private GameObject part;
    private Vector3 initialPosition;
    private Vector3 initialRotaion;
    private Vector3 initialScale;
    private Vector3 firstSlotPosition;
    private Vector3 secondSlotPosition;
    private Vector3 thirdSlotPosition;
    private Vector3 studOrientation;
    private string textToShow;
    private bool terminateExecution;
    private GameObject[] stud = new GameObject[3];

    public static bool videoFinished;
    public static string viseStudFirstSlot;
    public static string viseStudThirdSlot;
    public static string viseStudSecondSlot;

    public ViseSetupThree(MainManager manager)
    {
      _manager = manager;
      videoFinished = false;
      terminateExecution = false;

      part = GameObject.Find (ChoosePartState.finalName);
      part.transform.localPosition = new Vector3 (-1.0f,
part.transform.localPosition.y,
part.transform.localPosition.z);
      initialPosition = new Vector3 (-0.56f,-
0.3012f,0.1605f);
      initialRotaion = new Vector3 (1.4283f,180,-3.37f);
      initialScale = new Vector3 (0.05f,0.0025f,0.05f);
      viseStudFirstSlot = "studOne";
      viseStudSecondSlot = "studTwo";
      viseStudThirdSlot ="studThree";

      firstSlotPosition = new Vector3 (258.0f,0.0f,90.0f);
      secondSlotPosition = new Vector3 (118.0f,0.0f,90.0f);
      thirdSlotPosition = new Vector3 (-27.0f, 0.0f,
90.0f);

      studOrientation = new Vector3 (270.0f,0,0);
      CreateVideoObject ();

      stud [0] = _manager.CreatePrefab
(7,firstSlotPosition,studOrientation);
      stud [1] = _manager.CreatePrefab
(7,secondSlotPosition,studOrientation);
      stud[2] = _manager.CreatePrefab
(7,thirdSlotPosition,studOrientation);
      stud[0].name = viseStudFirstSlot;
      stud[1].name = viseStudSecondSlot;
      stud[2].name = viseStudThirdSlot;
      _manager.StartCoroutine (showText ());
    }
    public void Action()
    {
      if(videoFinished)
      {
        _manager.DestroyPrefab(videoObject);
      }
      if (!terminateExecution)
      {
        if (stud [1].transform.localPosition.z >= 0.25f)
        {
          stud [1].GetComponent<StudMover> ().MoveLeft ();
        }
        else if (stud [2].transform.localPosition.z >= -
1.5f)
        {
          stud [2].GetComponent<StudMover> ().MoveLeft ();
        }
        if (stud [0].transform.localPosition.z >= -1.5f)
        {
          stud [0].GetComponent<StudMover> ().MoveLeft ();
        }
      }
    }

    public void GUIImage()
    {
      GUI.skin = _manager.guiSkin;
      GUI.Label(new Rect(10, 10, Screen.width-20,
(int)(Screen.height/7)), textToShow);
      if(videoFinished)
      {
        if (GUI.Button (new Rect (100, Screen.height - 100,
150, 50), "Back"))
        {
          terminateExecution = true;
          Reset();
          _manager.SwitchState(new
OrientationState(_manager));
        }
        if (GUI.Button (new Rect (Screen.width - 200,
Screen.height - 100, 150, 50), "Proceed"))
        {
          _manager.SwitchState(new
ViseSetupFour(_manager));
        }
        if (GUI.Button (new Rect ((int)(Screen.width/2) -
100, Screen.height - 100, 200, 50), "Play Video"))
        {
          videoFinished = false;
          CreateVideoObject();
     }
  }
}

public void Reset()
{
part.transform.localPosition = new Vector3 (0,0,0);
  for (int i = 0; i<stud.Length; i++)
  {
  _manager.DestroyPrefab(stud[i]);
  }
}

private void CreateVideoObject()
{
videoObject = _manager.CreatePrefab
(10,initialPosition,initialRotaion,"BackgroundPlane");

videoObject.transform.localPosition = initialPosition;

videoObject.transform.localEulerAngles = initialRotaion;

videoObject.transform.localScale = initialScale;

}

IEnumerator showText()
    {
string str = "Insert one stud in each slot of the table.";
char [] temp = str.ToCharArray();
int counter = 0;
  while (counter < temp.Length)
  {
  textToShow += temp[counter];
  counter++;
  yield return new WaitForSeconds(0.01f);
  }
    yield return new WaitForSeconds (1.0f);
    }
  }
}
```