ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΔΥΤΙΚΗΣ ΕΛΛΑΔΑΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

"Ανάπτυξη Εφαρμογής Επαυξημένης Πραγματικότητας"

Σπουδαστής:
Δούρος Γεώργιος
Εισηγητής:
Χριστοδούλου Σωτήριος

ΑΝΤΙΡΡΙΟ 2018

ΠΕΡΙΕΧΟΜΕΝΑ

Περίληψη

Στην πτυχιακή εργασία αυτή περιγράφονται σημερινές τεχνολογίες επαύξησης πραγματικότητας καθώς επίσης και η ανάπτυξη μίας εφαρμογής με τη χρήση του Unity και της βιβλιοθήκης Vuforia.

Εισαγωγή

Η πτυχιακή αυτή εργασία αποτελεί το τελικό στάδιο των σπουδών μου στο Τ.Ε.Ι. Δυτικής Ελλάδας, στο Τμήμα Μηχανικών Πληροφορικής και αποτελεί μέρος έρευνας τεχνολογιών Επαυξημένης Πραγματικότητας, αλλά και υλοποίηση μιας εφαρμογής με τη χρήση αυτών.

Στο πρώτο μισό της εργασίας αυτής, θα αναλύσουμε τον ορισμό της επαυξημένης πραγματικότητας, με ποιους τρόπους επιτυγχάνεται και ποια είναι τα αποτελέσματά της στην καθημερινότητα μας. Ακόμα θα δούμε σε ποιους τομείς μπορεί να χρησιμοποιηθεί και τι μπορούμε να αποκομίσουμε από αυτήν.

Στο δεύτερο μισό της εργασίας, περιγράφεται βήμα- βήμα, η ανάπτυξη μιας τέτοιου είδους εφαρμογής. Θα αναφερθούν επίσης κάποιες σημερινές τεχνολογίες και εργαλεία υλοποίησης. Θα δούμε επίσης διάφορα προβλήματα που παρουσιάζονται κατά την ανάπτυξη και πως μπορούμε να τα αντιμετωπίσουμε.

Τέλος, θα ήθελα να ευχαριστήσω τον κ. Χριστοδούλου για την βοήθειά και τις συμβουλές που μου παρείχε για να πραγματοποιηθεί αυτή η εργασία.

Επαυξημένη Και Εικονική Πραγματικότητα

Αρχικά θα πρέπει να ξεκαθαρίσουμε την σύγχυση που υπάρχει ανάμεσα στους όρους *Επαυξημένη Πραγματικότητα* και *Εικονική Πραγματικότητα*. Ένα σημαντικό λάθος που παρατηρείται είναι ότι ο κόσμος θεωρεί πως η Επαυξημένη Πραγματικότητα υπάγεται στην Εικονική ή πολλές φορές δεν υπάρχει διαχωρισμός για τους δύο αυτούς όρους παρ' όλο που ο καθένας εμπεριέχει διαφορετική σημασία.

Θα προσπαθήσουμε να εξαλείψουμε αυτή σύγχυση λοιπόν δίνοντας δυο απλοϊκούς ορισμούς για τις δύο έννοιες. Αρχικά ας δούμε τον ορισμό της "Επαυξημένης Πραγματικότητας":

Επαυξημένη Πραγματικότητα είναι η απεικόνιση εικόνων που έχουν παραχθεί από υπολογιστή (Computer Generated Images) πάνω στον πραγματικό κόσμο, δημιουργώντας ένα ενιαίο συνθετικό περιβάλλον.

Παρατηρούμε από τον ορισμό, ότι στην Επαυξημένη Πραγματικότητα είμαστε θεατές του **πραγματικού κόσμου** ο οποίος έχει **επαυξηθεί** με τη χρήση εικόνων παραγόμενων από έναν υπολογιστή.

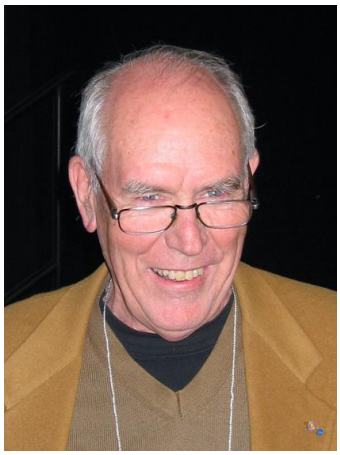
Ας συνεχίσουμε με έναν απλουστευμένο ορισμό της "Εικονικής Πραγματικότητας", που δόθηκε από τον *Jaron Lanier,* πρωτοπόρο στις τεχνολογίες Εικονικής Πραγματικότητας το 1989:

Εικονική Πραγματικότητα είναι η προσομοίωση ενός φανταστικού περιβάλλοντος που έχει παραχθεί από έναν υπολογιστή και ο χρήστης μπορεί να "βυθιστεί" σε αυτό.

Βλέπουμε λοιπόν πως οι δύο έννοιες αναφέρονται σε δύο **διαφορετικά** περιβάλλοντα. Ενώ η Επαυξημένη Πραγματικότητα μιλάει για ένα υβριδικό περιβάλλον με στοιχεία του πραγματικού κόσμου και εικονικά στοιχεία παραγόμενα από υπολογιστή, η Εικονική Πραγματικότητα αναφέρεται σε ένα περιβάλλον εξολοκλήρου παραγόμενο από υπολογιστή.
Φυσικά και στις δυο περιπτώσεις ο χρήστης μπορεί να αλληλεπιδράσει με το εκάστοτε περιβάλλον με τη χρήση κάποιων μηχανισμών, π.χ. αισθητήρων.

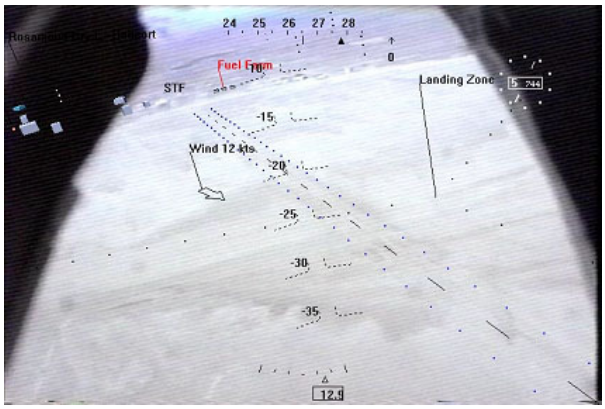
Η εργασία αυτή εστιάζει στην πρώτη έννοια, αυτή της Επαυξημένης Πραγματικότητας και εφόσον έχουμε πλέον διαχωρίσει την Επαυξημένη από την Εικονική πραγματικότητα ας κάνουμε μία ιστορική αναδρομή για αυτήν.

Επαυξημένη Πραγματικότητα: Ιστορική Αναδρομή



Όσο και αν φαίνεται περίεργο, η Επαυξημένη Πραγματικότητα πρωτοεμφανίστηκε, με μια πιο απλή μορφή, το **1968**. Η μορφή της ήταν μία "πρωτόγονη" σε άποψη ρεαλισμού και διεπαφής χρήστη, **επικεφαλής οθόνη (Head-Mounted Display)** η οποία παρουσίαζε στον χρήστη ένα εικονικό δωμάτιο, όπου ο ίδιος μπορούσε να σχεδιάσει απλά σκίτσα **Wireframe** στο χώρο. Ο χρήστης έβλεπε το δωμάτιο στερεοσκοπικά ανάλογα με την κλίση του κεφαλιού του και ήταν δυνατή η σχεδίαση μόνο με μαύρο χρώμα. Η διεπαφή αυτή απαιτούσε την ανίχνευση της κλίσης και του προσανατολισμού του κεφαλιού του χρήστη και καθιστούσε αναγκαία την προσαρμογή της οθόνης σε έναν μηχανικό βραχίωνα ο οποίος εκτεινόταν από την οροφή του εργαστηρίου. Η περίεργη αυτή εμφάνιση του μηχανήματος ενέπνευσε και το όνομά του το οποίο ήταν: το **Σπαθί Του Δαμοκλή (The Sword Of Damocle)** και δημιουργός του ήταν ο **Ivan Sutherland** (εικόνα).

Ο όρος *Επαυξημένη Πραγματικότητα* ακούστηκε για πρώτη φορά το **1990**, από τον **Tom Caudell,** έναν ερευνητή της εταιρείας **Boeing**, γνωστή ακόμα και σήμερα για την κατασκευή επιβατικών αεροπλάνων, ελικοπτέρων, πυραύλων και δορυφόρων. Ο Tom Caudell κλήθηκε να βρει μία εναλλακτική λύση για τα δαπανηρά διαγράμματα και τις συσκευές σήμανσης όπου χρησιμοποιούνταν για την καθοδήγηση των εργαζομένων στη συναρμολόγηση τμημάτων αεροσκαφών. Η εναλλακτική που πρότεινε ήταν η αντικατάσταση μεγάλων κόντρα πλακέ πινάκων, οι οποίοι εμπεριείχαν σχέδια για τη καλωδίωση κάποιον μηχανισμών των αεροπλάνων, με ιδικά γυαλιά τα οποία θα απεικόνιζαν τις οδηγίες στους εργαζόμενους για την σωστή καλωδίωση.



Το **2000**, η **NASA** εκτοξεύει το διαστημόπλοιο **NASA X-38** το οποίο χρησιμοποιούσε ένα **Σύστημα Συνθετικής Όρασης (Synthetic Vision System)** για την πλοήγηση του (εικόνα δεξιά). Το σύστημα αυτό εκμεταλλευόταν την Επαυξημένη Πραγματικότητα για την απεικόνιση πληροφοριών της πτήσης του σκάφους σε πραγματικό χρόνο. Η εφαρμογή αυτή της ίσως είναι η πρώτη εφαρμογή Επαυξημένης Πραγματικότητας η οποία είναι πιο κοντά στις εφαρμογές όπου χρησιμοποιούνται σήμερα.
Στο σημείο αυτό αξίζει να σημειωθεί ότι επίσης το 2000 δημιουργείται η πρώτη βιβλιοθήκη ανοιχτού κώδικα για την ανάπτυξη εφαρμογών Επαυξημένης Πραγματικότητας από τον

Hirokazu Kato, με το όνομα **ARToolKit**. Η βιβλιοθήκη έγινε προσβάσιμη μέσο του Πανεπιστημίου της Ουάσιγκτον το 2001 και αρχικά έτρεχε σε λειτουργικά συστήματα **Symbian**. Αργότερα αναπτύχθηκε και για άλλες πλατφόρμες.

Βασισμένες στις μεθόδους λειτουργίας της ARToolKit, άρχισαν να εμφανίζονται νέες βιβλιοθήκες για την ανάπτυξη εφαρμογών. Σήμερα υπάρχουν πάρα πολλές βιβλιοθήκες και εργαλεία όπου μπορούμε να επιλέξουμε για την ανάπτυξη μίας εφαρμογής Επαυξημένης Πραγματικότητας, μερικές ονομαστικά είναι οι: **Vuforia**, **Wikitude**, **Kudan** και **ARCore**, η οποία είναι βιβλιοθήκη της Google.

Όλες οι παραπάνω βιβλιοθήκες έχουν ως κοινό χαρακτηριστικό ότι χρησιμοποιούν διάφορους μηχανισμούς ανάλυσης εικόνας για να αναγνωρίζουν τον φυσικό κόσμο και να απεικονίζουν την επαύξησή του με τη χρήση ενός μέσου, λόγου χάρη μίας οθόνης. Αυτό σημαίνει πως είναι ιδανικές για την ανάπτυξη εφαρμογών σε **φορητές** αλλά και **φορετές (Wearable)** συσκευές, όπως "έξυπνα" τηλέφωνα, "έξυπνα" ρολόγια και ψηφιακά γυαλιά.

Η πτυχιακή αυτή εστιάζει σε τέτοιου είδους εφαρμογές και θα αναλυθεί περισσότερο το πως λειτουργούν και πως μπορούμε να εκμεταλλευτούμε τις βιβλιοθήκες αυτές για να κατασκευάσουμε Επαυξημένης Πραγματικότητας εφαρμογές. Ωστόσο, θα ήταν λογικό να δούμε και άλλες μεθόδους όπου μπορούμε να χρησιμοποιήσουμε για να πετύχουμε εμπειρίες Επαυξημένης Πραγματικότητας.

Επαυξημένη Πραγματικότητα: Σήμερα

Στην εποχή μας και με το πέρασμα των ετών, η επαυξημένη πραγματικότητα ακμάζει ραγδαία. Πλέον όλο και περισσότερες εφαρμογές όπου χρησιμοποιούν τεχνολογίες επαύξησης της πραγματικότητας έρχονται στο προσκήνιο και σε ποικίλους τομείς και ενδέχεται και ακόμα μεγαλύτερη άνθηση τέτοιων εφαρμογών τα επόμενα χρόνια.

Από τουριστικές εφαρμογές, εφαρμογές υγείας, μέχρι και εκπαιδευτικές εφαρμογές αλλά και βιντεοπαιχνίδια, πολλές από αυτές εκμεταλλεύονται την Επαυξημένη Πραγματικότητα για την καλύτερη διεπαφή με τον χρήστη ή για την αναπαράσταση πληροφοριών ή και πιο απλά για μία πιο διασκεδαστική εμπειρία με την εφαρμογή.

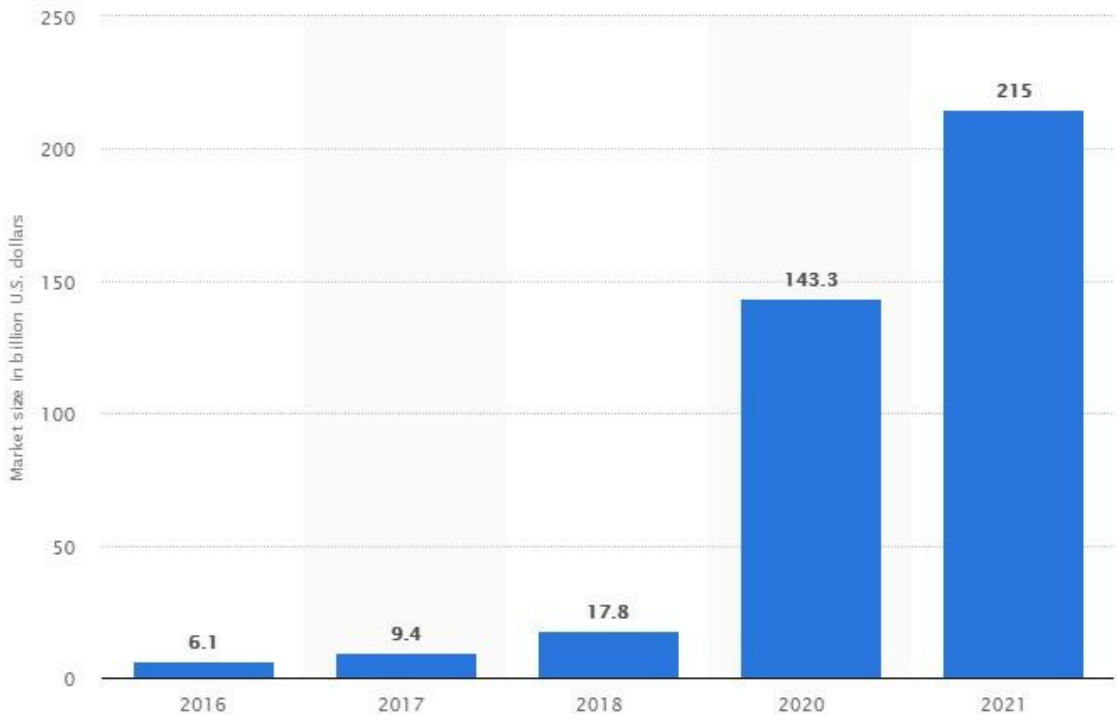
Συγκεκριμένα, ερευνώντας στατιστικές και μελέτες από ερευνητικά sites μπορούμε να παρατηρήσουμε πως η Επαυξημένη Πραγματικότητα αναμένεται να έχει τουλάχιστον 1 δισεκατομμύριο χρήστες μέχρι το 2020 σε όλον τον κόσμο.

Επίσης μέχρι τον Ιανουάριο του 2018, είχαν καταγραφεί πάνω από 543 νεοσύστατες εταιρείες οι οποίες κατατάσσονται σε εταιρίες που χρησιμοποιούν τεχνολογίες Επαυξημένης Πραγματικότητας.

Το μέγεθος της αγοράς στις εφαρμογές Επαυξημένης Πραγματικότητας ανέρχεται στα 61 δισεκατομμύρια δολάρια ΗΠΑ μέχρι το 2023. Το 2016 εκτιμήθηκε στα 2.39 δισ., επομένως ο Σύνθετος Ετήσιος Ρυθμός Ανάπτυξης (CARG) της αγοράς αναμένεται να αυξηθεί κατά 55% κατά την περίοδο της πρόβλεψης.

Μέρος της αύξησης επίσης, θα είναι και η παγκόσμια μεταφορά έξυπνων γυαλιών Augmented Reality που προβλέπεται να φτάσει περίπου 5,4 εκατομμύρια μονάδες μέχρι το 2020.

Επιπλέον, σύμφωνα με την **Infoholic Research**, η βιομηχανία των παιχνιδιών Επαυξημένης Πραγματικότητας αναμένεται να φθάσει τα 285 δισεκατομμύρια δολάρια μέχρι το 2023, αυξάνοντας σε CAGR κατά 152% κατά την περίοδο πρόβλεψης 2017-2023.



Προβλεπόμενη Αύξηση του μεγέθους της Αγοράς Εφαρμογών Επαυξημένης Πραγματικότητας βάσει του ***statista.com*** για την περίοδο 2018-2021.

Με βάση τα παραπάνω στατιστικά παρατηρούμε πως η Επαυξημένη Πραγματικότητα, βρίσκεται και θα βρίσκεται στο προσκήνιο των αγορών και των καινοτόμων τεχνολογιών για τα επόμενα χρόνια.

Κατηγορίες Επαυξημένης Πραγματικότητας

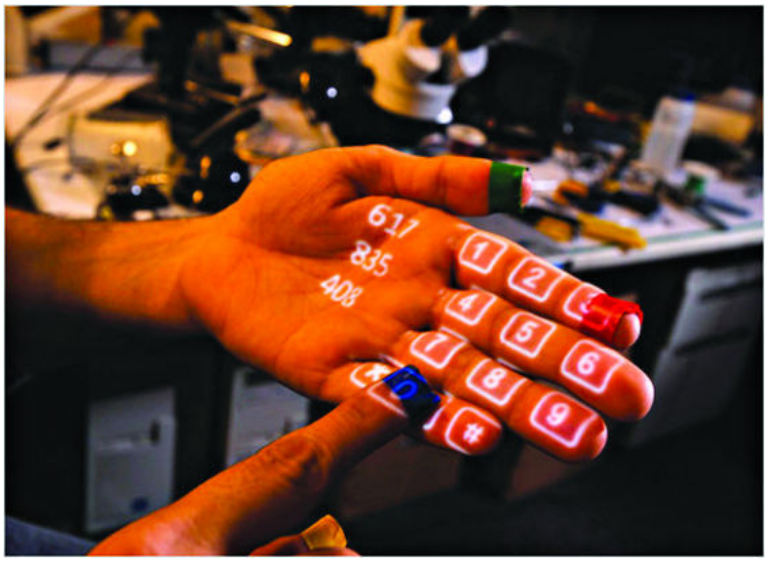
Οι δυνατότητες της επαυξημένης πραγματικότητας είναι πραγματικά απεριόριστες και η ίδια μπορεί να χρησιμοποιηθεί με αμέτρητους τρόπους και για διαφορετικές περιπτώσεις χρήσης. Λόγο λοιπόν των απεριόριστων δυνατοτήτων είναι δύσκολο να κάνουμε απόλυτη κατηγοριοποίηση των εφαρμογών. Παρ' όλα αυτά, μπορούμε, κατά προσέγγιση να διαιρέσουμε τον τρόπο επαύξησης πραγματικότητας σε 3 κατηγορίες.

Η επαυξημένη πραγματικότητα δεν τελειώνει στα smartphones. Σίγουρα τα έξυπνα τηλέφωνα είναι το πρώτο μέσον που μας έρχεται στο μυαλό όταν μιλάμε για επαύξηση της πραγματικότητας, όμως υπάρχουν και άλλα μέσα που παρέχουν διαφορετικές δυνατότητες. Έτσι έχουμε λοιπόν την πρώτη κατηγορία, την **Επαύξηση με βάση την προβολή**.

Επαυξημένη Πραγματικότητα βάσει Προβολής

Επαυξημένη Πραγματικότητα βάσει Προβολής (Projection-based AR) είναι η επαύξηση της πραγματικότητας χρησιμοποιώντας προβολές φωτός πάνω σε αντικείμενα. Οι εφαρμογές τέτοιου τύπου είναι πιο ελκυστικές εν αντιθέσει με εφαρμογές AR που μπορούν να εγκατασταθούν στο κινητό μας. Αυτό όμως που καθιστά ενδιαφέρουσα αυτή την κατηγορία είναι η μεγάλη ποικιλία δυνατοτήτων της.

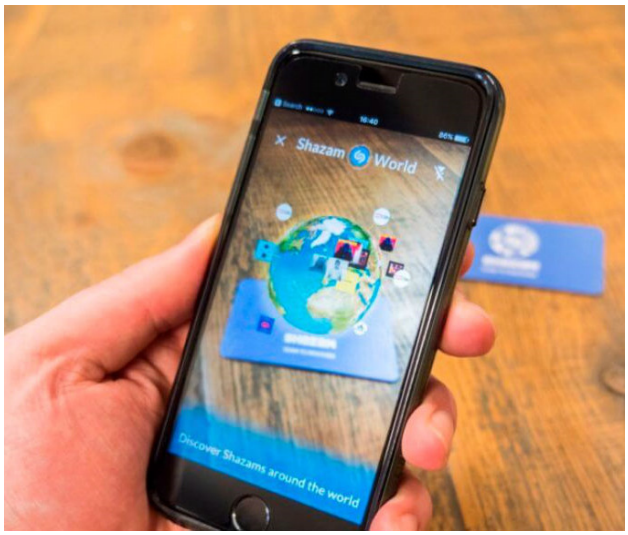
Ένας απλός τρόπος υλοποίησης τέτοιας επαύξησης είναι με τη χρήση προβολής φωτός σε επιφάνεια. Η παρακάτω εικόνα μας δίνει ένα παράδειγμα χρήσης αυτής της κατηγορίας. Όπως μπορούμε να διακρίνουμε στην εικόνα, δέσμες φωτός εκτοξεύονται στο χέρι μας και η αλληλεπίδραση γίνεται με την χρήση της προβαλλόμενης εικόνας στη παλάμη του χεριού μας.



Μία από τις ευρύτερες χρήσεις των τεχνικών AR που βασίζονται στην προβολή είναι μη διαδραστική. Η ριπή φωτός σε επιφάνειες μπορεί να χρησιμοποιηθεί για να εξαπατήσει τον θεατή σχετικά με τη θέση, τον προσανατολισμό και το βάθος ενός αντικειμένου το οποίο

θέλουμε να αναπαραστήσουμε στον πραγματικό κόσμο. Έτσι ως αποτέλεσμα μπορούμε να έχουν απεικόνιση τρισδιάστατων εικόνων στον χώρο.

Επαυξημένη Πραγματικότητα βάσει Αναγνώρισης



Η **Επαυξημένη Πραγματικότητα Βάσει Αναγνώρισης (Recognition-based AR)** επικεντρώνεται στην αναγνώριση αντικειμένων και στη συνέχεια μας παρέχει περισσότερες πληροφορίες σχετικά με το αντικείμενο οι οποίες απεικονίζονται με τη χρήση ενός μέσου (π.χ. smartphone) στον πραγματικό κόσμο.

Η αλήθεια είναι πως όλες οι κατηγορίες επαύξησης, εκτός από την *Επαύξηση βάσει Τοποθεσίας* που θα δούμε αργότερα, χρησιμοποιούν κάποιο τύπο συστήματος αναγνώρισης για να ανιχνεύσουν τον τύπο του αντικειμένου πάνω από το οποίο θα γίνει η απεικόνιση της της πληροφορίας ή για την αλληλεπίδραση του χρήστη με το επαυξημένο αντικείμενο.

Η βιβλιοθήκη Vuforia, που αναφέρθηκε προηγουμένως και η οποία χρησιμοποιείται στο επόμενο κεφάλαιο για την υλοποίηση AR εφαρμογής, καθώς και η ίδια η εφαρμογή, κατατάσσεται σε αυτή την κατηγορία επαυξημένης πραγματικότητας.

Επαυξημένη Πραγματικότητα βάσει Τοποθεσίας

Η **Επαυξημένη Πραγματικότητα βάσει Τοποθεσίας (Location-based AR)** είναι μία από τις πιο ευρέως εφαρμοζόμενες τεχνικές επαύξησης της πραγματικότητας. Κινητήρια δύναμη πίσω από αυτό, είναι η εύκολη διαθεσιμότητα των smartphones και τις δυνατότητες που παρέχουν για την ανίχνευση της γεωλογικής θέσης τους (Geo-Location). Υπηρεσίες GPS αλλά και αισθητήρες όπως το Γυροσκόπιο για την ανίχνευση της κατεύθυνσης του χρήστη εξυπηρετούν στην απεικόνιση επαυξημένων αντικειμένων στο χώρο. Ένα παράδειγμα χρήσης αυτής της κατηγορίας είναι εφαρμογές που λειτουργούν ως “τουριστικοί ξεναγοί” για τους χρήστες, παρέχοντας πληροφορίες για την τοποθεσία που βρίσκονται.

Επαύξηση Πραγματικότητας βάσει Αναγνώρισης

Computer Vision

Η εργασία αυτή στοχεύει στην ανάπτυξη εφαρμογής επαυξημένης πραγματικότητας βάσει αναγνώρισης. Όπως αναφέρθηκε και νωρίτερα, οι εφαρμογές αυτές λειτουργούν ανιχνεύοντας μοτίβα στον πραγματικό κόσμο και αναπαριστούν επάνω σε αυτόν, εικονικά αντικείμενα.

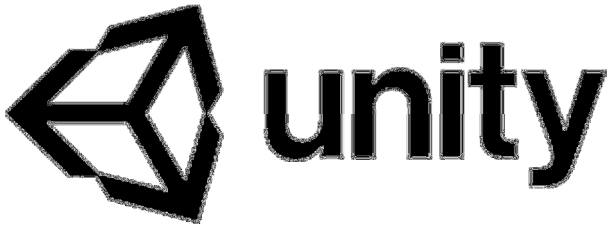
Προτού λοιπόν αναφέρουμε και αναλύσουμε τα εργαλεία και τις βιβλιοθήκες που χρησιμοποιήθηκαν για την υλοποίηση της εφαρμογής μας ας δούμε πως μπορούν οι βιβλιοθήκες αυτές να αναλύσουν και ανιχνεύσουν τα μοτίβα αυτά έτσι ώστε να καταφέρνουν την απεικόνιση εικονικών αντικειμένων.

Οι τεχνολογίες που εκμεταλλεύονται οι βιβλιοθήκες αυτές είναι οι τεχνολογίες **Computer Vision**. Computer Vision είναι ένα πεδίο της Πληροφορικής που ασχολείται με τον τρόπου όπου μπορούν οι υπολογιστές να αποκτήσουν υψηλού επιπέδου κατανόηση ψηφιακών εικόνων και βίντεο και να αντλούν πολύπλοκα δεδομένα από αυτά. Με πιο απλά λόγια, το πεδίο αυτό της Πληροφορικής προσπαθεί να "υλοποιήσει" διαδικασίες που εκτελεί το ανθρώπινο μάτι.

Οι τεχνολογίες αυτές άρχισαν να εμφανίζονται στα τέλη του 1960, σε πανεπιστήμια τα οποία ήταν πρωτοπόροι στο πεδίο της Τεχνητής Νοημοσύνης (Artificial Intelligence) και είχαν ως σκοπό να μιμούνται λειτουργίες του ανθρώπινου ματιού, έτσι ώστε να μπορούν ρομπότ να ενισχύονται με "έξυπνη συμπεριφορά".

Οι διεργασίες που εκτελούν οι Computer Vision τεχνολογίες περιλαμβάνουν μεθόδους **απόκτησης**, **επεξεργασίας**, **ανάλυσης** και **κατανόησης** ψηφιακών εικόνων με αποτέλεσμα την παραγωγή αριθμητικών ή συμβολικών πληροφοριών υπό τη μορφή π.χ. **αποφάσεων**

Unity



Η **Unity**, είναι μια μηχανή ανάπτυξης παιχνιδιών (game engine) η οποία δημιουργήθηκε από την εταιρεία, Unity Technologies το 2005. Η μηχανή χρησιμοποιείται κυρίως για την ανάπτυξη τρισδιάστατων (3D) και δισδιάστατων (2D) παιχνιδιών, αλλά και προσομοιώσεων (simulation) για υπολογιστές, κονσόλες και κινητές συσκευές.

Αρχικά η Unity, προοριζόταν στο να υλοποιούνται παιχνίδια και εφαρμογές μόνο για λειτουργικό σύστημα OS X, αλλά σήμερα έχει επεκταθεί σε 27 πλατφόρμες, μεταξύ άλλων και των λειτουργικών Windows και Linux.

Vuforia

Η βιβλιοθήκη που θα χρησιμοποιηθεί για την υλοποίηση της εφαρμογής, είναι η βιβλιοθήκη **Vuforia**.

Η **Vuforia**, είναι μια Βιβλιοθήκη Ανάπτυξης Λογισμικού (SDK) για κινητές συσκευές που επιτρέπει τη δημιουργία εφαρμογών Επαυξημένης Πραγματικότητας.Η βιβλιοθήκη χρησιμοποιεί τεχνολογίες **Computer Vision** για να αναγνωρίζει και να παρακολουθεί επίπεδες εικόνες (Image Targets) και απλά τρισδιάστατα σχήματα (Object Targets) σε πραγματικό χρόνο. Η ανάλυση και η παρακολούθηση αυτή μας δίνει τη δυνατότητα να τοποθετούμε και να προσανατολίζουμε εικονικά αντικείμενα σε σχέση με εικόνες πραγματικού κόσμου, όταν αυτές προβάλλονται με τη χρήση ενός μέσου (π.χ. Κάμερα Κινητού Τηλεφώνου).

Το εικονικό αντικείμενο που αναπαριστάται με τη σειρά του, παρακολουθεί την εικόνα του πραγματικού κόσμου που προβάλλεται από το οπτικό μέσο, σε πραγματικό χρόνο και μεταποιεί τις διαστάσεις του και το προσανατολισμό του έτσι ώστε να φαίνεται ότι το εικονικό αντικείμενο είναι μέρος του πραγματικού κόσμου.

ΑΝΑΠΤΥΞΗ ΣΚΑΚΙΣΤΙΚΗΣ ΕΦΑΡΜΟΓΗΣ

Περιγραφή

Στην ενότητα αυτή θα αναπτύξουμε μία σκακιστική εφαρμογή επαυξημένης πραγματικότητας με την χρήση των εργαλείων Unity και Vuforia που αναφέραμε πιο πάνω. Επίσης θα μελετήσουμε τα διάφορα προβλήματα και περιορισμούς που παρουσιάστηκαν και θα δούμε και με ποιους τρόπους τα αντιμετωπίσαμε.

Η λειτουργία της εφαρμογής θα είναι να παρουσιάζει στον χρήστη επαυξημένα, χρήσιμες πληροφορίες για μία παρτίδα σκάκι όπως ποια είναι η καλύτερη κίνηση για τα λευκά και για τα μαύρα πιόνια, έναν συντελεστή για κάθε μία από αυτές τις κινήσεις, ο οποίος αναπαριστά πόσο καλή είναι η κίνηση και θα παρουσιάζει επίσης την τελευταία κίνηση που έγινε στον προηγούμενο γύρο.

Αξίζει να σημειωθεί πως στην εργασία αυτή θα αναλυθεί μόνο το πεδίο της επαύξησης και της παρουσίασης των πληροφοριών στο χρήστη. Οι μέθοδοι υπολογισμού καλύτερης κίνησης καθώς και το πως θα εισέρχονται σαν είσοδο στην εφαρμογή αποτελούν διαφορετικό αντικείμενο απασχόλησης και δεν θα ασχοληθούμε με αυτό. Ωστόσο, θα θεωρούμε πως λαμβάνουμε έτοιμα τα δεδομένα τα οποία θέλουμε να αναπαραστήσουμε μέσο της εφαρμογής μας.

Στήσιμο των απαραίτητων προαπαιτούμενων

Όπως είπαμε η εφαρμογή μας κατατάσσεται στις εφαρμογές **Επαυξημένης Πραγματικότητας βάση Αναγνώρισης**, επομένως θα χρειαστούμε κάτι το οποίο θα μπορεί να αναγνωρίζει η εφαρμογή μας για να κατανοεί ότι "αυτό είναι ένα ταμπλό του σκάκι" και να υλοποιεί στην συνέχεια την απεικόνιση των πληροφοριών.

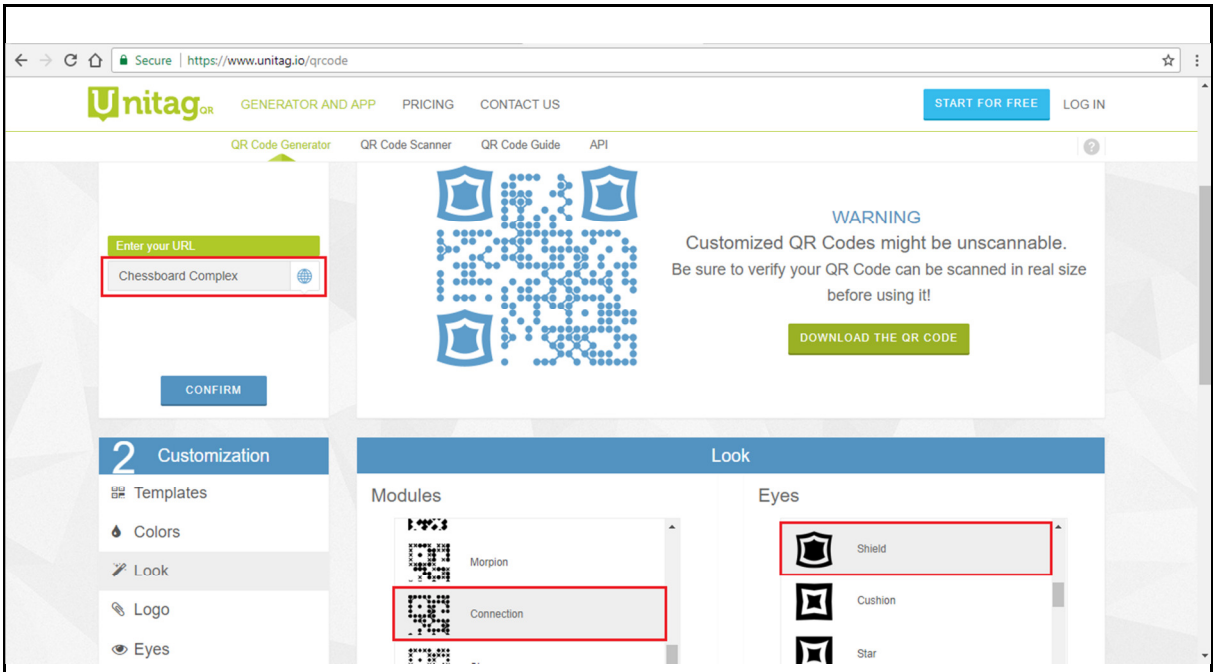
Είναι λογικό να αναρωτηθεί κάποιος "*Γιατί να μην χρησιμοποιήσουμε την ίδια την σκακιέρα για την αναγνώριση;*" και η απάντηση είναι απλή:

Οι τεχνολογίες Computer Vision όπου αποτελούν την κύρια βιβλιοθήκη που θα χρησιμοποιήσουμε λειτουργούν με μεγαλύτερη ακρίβεια όταν τα μοτίβα που αναγνωρίζονται είναι πολύπλοκα και όχι επαναλαμβανόμενα, επομένως η τετραγωνισμένη σκακιέρα δεν διευκολύνει στην αναγνώριση. Έτσι λοιπόν μία εναλλακτική λύση είναι να χρησιμοποιήσουμε ένα QR Code ως αντικείμενο αναγνώρισης.

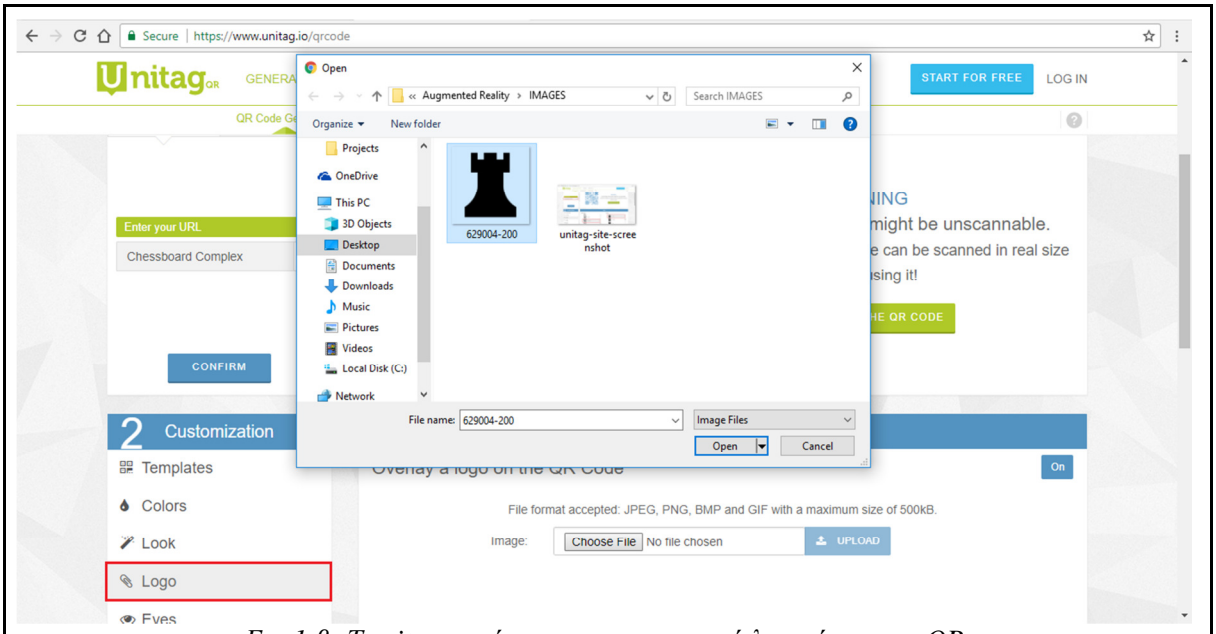
Το πιο πάνω QR, δημιουργήθηκε με τη χρήση μίας online γεννήτριας QR Code η οποία φιλοξενείται στην διεύθυνση: https://www.unitag.io/qrcode

Η συγκεκριμένη γεννήτρια μας εξυπηρετεί ιδιαιτέρως, διότι έχει πολλές επιλογές διακόσμησης του QR, πράγμα που μας διευκολύνει στο να γίνει πιο περίπλοκο και άρα πιο εύκολο από την εφαρμογής μας να αναγνωριστεί. Όλα τα διακοσμητικά όπως το χρώμα, τα σχήματα αλλά και το λογότυπο με τον πύργο στο κέντρο, υλοποιήθηκαν μέσα από τη γεννήτρια. Η μοναδική παρέμβαση στην παραγόμενη εικόνα ήταν η τοποθέτηση του περιγράμματος, όπου χρησιμοποιήθηκε το πρόγραμμα MS Paint.

Ενδεικτικά πιο κάτω απεικονίζονται μερικές επιλογές που χρησιμοποιήθηκαν για τη διακόσμηση του QR.

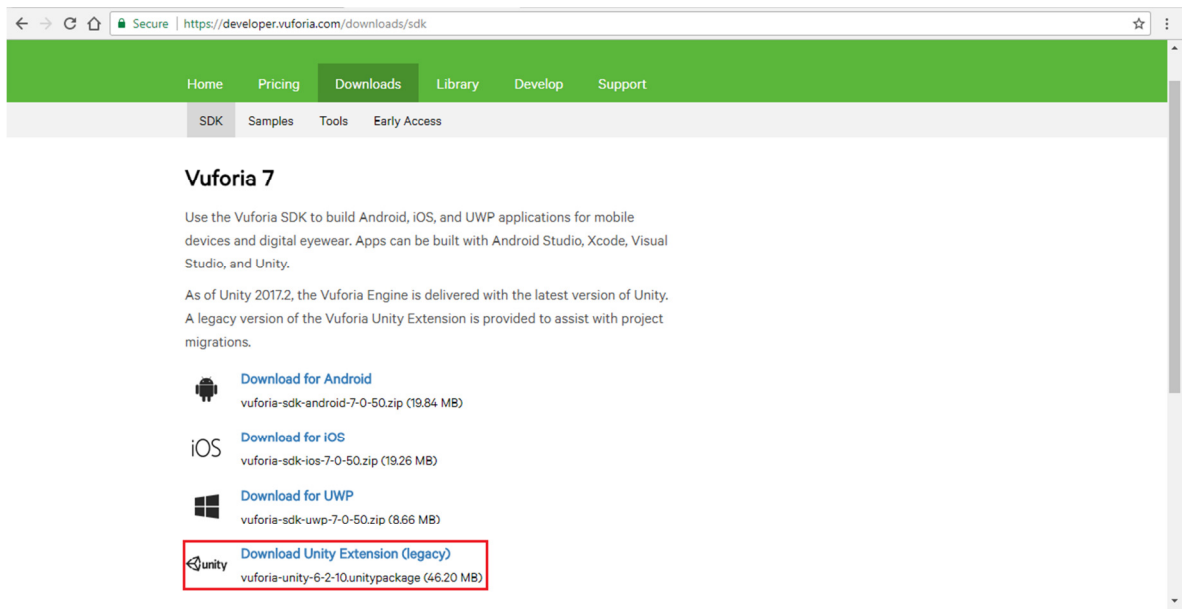


Εικ.1 α: Το site που δημιουργήθηκε το QR Code.



Εικ.1 β: Το site επιτρέπει και την εισαγωγή λογοτύπου στο QR.

Αφού λοιπόν έχουμε έτοιμο το QR σειρά έχει η βιβλιοθήκη της Vuforia. Η βιβλιοθήκη είναι διαθέσιμη στην ιστοσελίδα: https://developer.vuforia.com/downloads/sdk

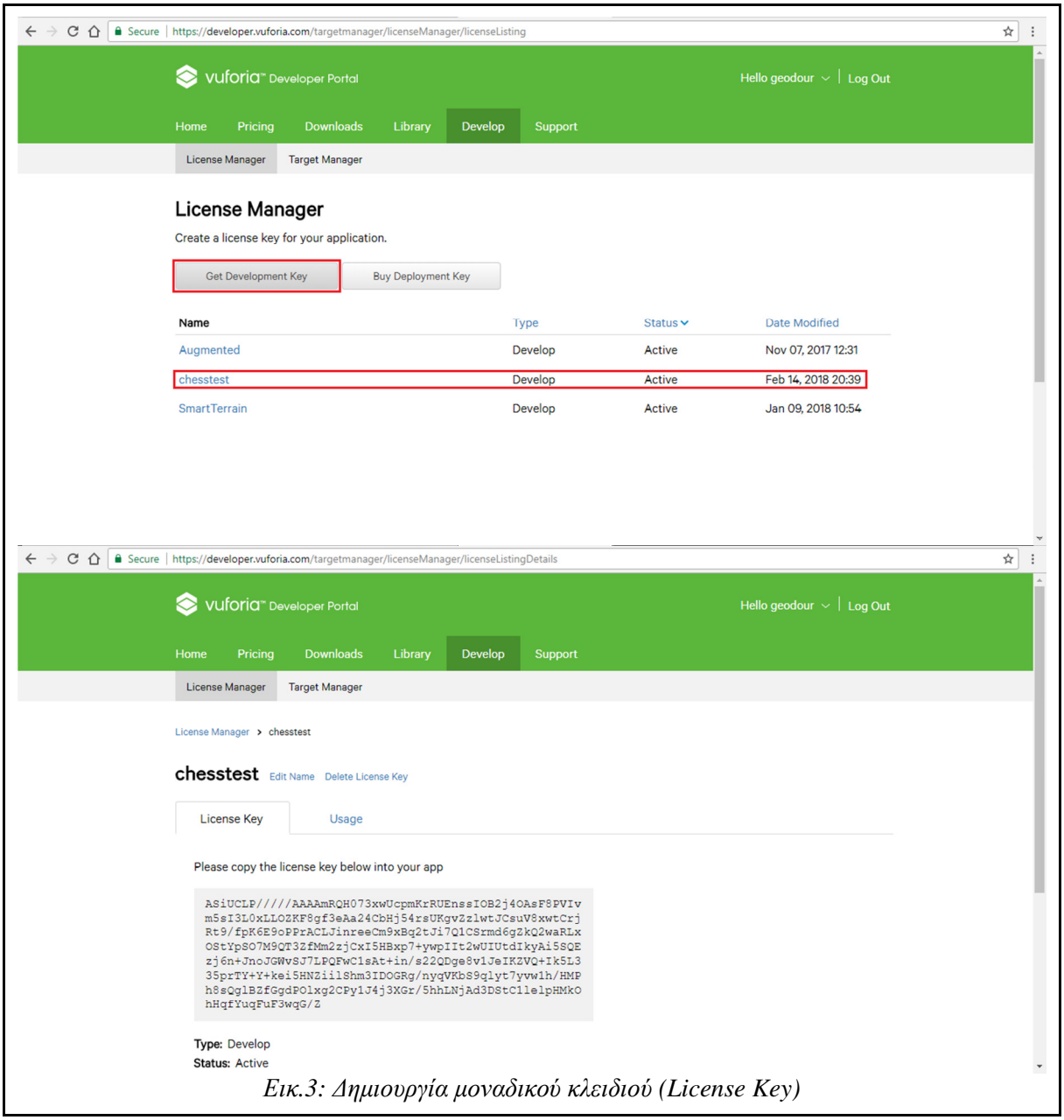


Εικ.2: Επιλέγουμε την επέκταση της βιβλιοθήκης για το Unity.

Η βιβλιοθήκη είναι πλέον ενσωματωμένη στο Unity. Για λόγους τυπικότητας όμως διότι η εφαρμογή δημιουργήθηκε πριν από το γεγονός αυτό της ενσωμάτωσης, θα δουλέψουμε με την Legacy έκδοση της βιβλιοθήκης την οποία πρέπει να την κατεβάσουμε ξεχωριστά και να την εισάγουμε στο Unity.

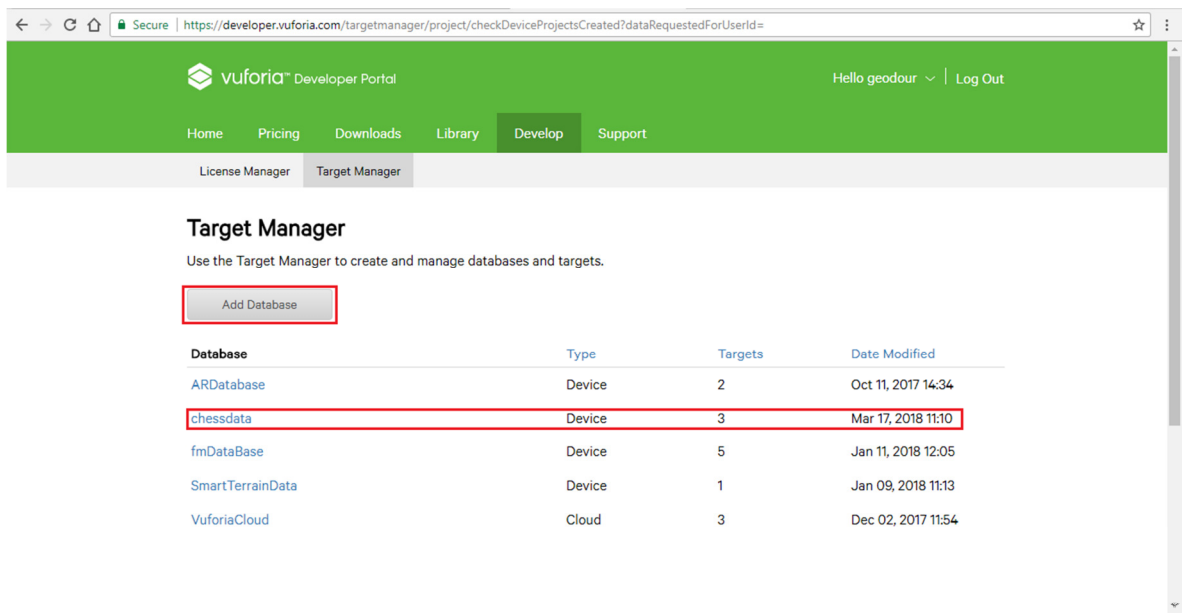
Ακόμα θα χρειαστεί να δημιουργήσουμε έναν λογαριασμό ως προγραμματιστές στην ιστοσελίδα της Vuforia. Έχοντας τον λογαριασμό θα αποκτήσουμε πρόσβαση σε ένα μοναδικό κωδικό "κλειδί" που θα χρειαστεί να τον εισάγουμε στην εφαρμογή μας για να μπορέσουμε να χρησιμοποιήσουμε την βιβλιοθήκη.

Επιπλέον θα πρέπει να "ανεβάσουμε" τις εικόνες που χρησιμοποιούμε για αναγνώριση σε μια βάση δεδομένων που μας παρέχεται από τη Vuforia. Ο λόγος διότι όταν "σηκώνουμε" την εικόνα, ένας αλγόριθμος την αναλύει και επιστρέφει ιδικά **Σημεία (Features)** πάνω στην εικόνα τα οποία θα χρησιμοποιήσει η εφαρμογή μας για να αναγνωρίσει την εικόνα. Τα σημεία αυτά μπορούμε να τα κατεβάσουμε και να τα εισάγουμε στο Unity σε μορφή XML.



Εικ.3: Δημιουργία μοναδικού κλειδιού (License Key)

Στη συνέχεια δημιουργούμε και τη βάση στην οποία θα ανεβάσουμε την εικόνα μας:



Εικ.4: Απο το μενού "Target Manager" δημιουργούμε την βάση "chessdata".

Βελτιστοποίηση ανίχνευσης στόχων και της σταθερότητας παρακολούθησης

Όπως αναφέρθηκε και προηγουμένως, όσο πιο πολύπλοκες είναι οι εικόνες που χρησιμοποιούμε σαν στόχους αναγνώρισης, τόσο πιο σταθερή και ακριβής είναι η επαύξηση. Η ποιότητα όμως της επαύξησης εξαρτάται και από άλλους παράγοντες όπως το μέγεθος και η ανάλυση της εικόνας, καθώς και ο φωτισμός και η γωνία με την οποία προσπαθούμε να την ανιχνεύσουμε. Η εικόνα που χρησιμοποιούμε για αναγνώριση θα πρέπει να είναι **πλούσια σε λεπτομέρειες** και να έχει υψηλές **αντιθέσεις σε χρώματα (Contrast Levels)**.

Χαρακτηριστικά	Παράδειγμα
Πλούσια εικόνα σε λεπτομέρειες	Ένας δρόμος με πολλούς ανθρώπους, η πρόσοψη ενός κτηρίου, κολάζ με πολλά σχήματα και αντικείμενα
Υψηλή αντίθεση	Η εικόνα έχει και σκοτεινές και φωτεινές περιοχές, δεν είναι θαμπή και φωτίζεται καλά
Μη αναλαμβανόμενα μοτίβα	Όπως μια εικόνα από γρασίδι, πρόσοψη κτηρίου με ολόιδια παράθυρα, ταμπλό σκακιέρας κλπ.

Ένας σημαντικός παράγοντας για να έχουμε σταθερή και ποιοτική επαύξηση, είναι ο φωτισμός που υπάρχει στον χώρο την ώρα της αναγνώρισης.

Εάν η εικόνα που θέλουμε να αναγνωρίσουμε φωτίζεται με καλό φωτισμό, τότε γίνεται πιο εύκολη η αναγνώρισή της αλλά είναι και πιο εύκολο για την εφαρμογή μας να "ανιχνεύει" τη εικόνα σε πραγματικό χρόνο για να επιτυγχάνεται έτσι σταθερή επαύξηση των πληροφοριών που θέλουμε να απεικονίσουμε και να έχουμε μια αδιάλειπτη επαύξηση πραγματικότητας, χωρίς λάθη και δυσανάλογους προσανατολισμούς των επαυξημένων αντικειμένων.

Αξίζει να σημειωθεί πως η βιβλιοθήκη της Vuforia λειτουργεί καλύτερα για εφαρμογές κλειστού χώρου, όπου ο φωτισμός είναι πιο σταθερός και διαχειρίσιμος. Λόγω του ότι έχουμε να κάνουμε με αναγνώριση αντικειμένων και εικόνων για να πετύχουμε επαύξηση, έρχεται στην επιφάνεια το εξής πρόβλημα:

"Πως μπορεί να επιτευχθεί Επαύξηση βάση Αναγνώρισης σε σκοτεινό περιβάλλον ή τις νυχτερινές ώρες;"

Η Vuforia δίνει την απλοϊκή λύση το να επιτρέπει στην εφαρμογή μας να χρησιμοποιεί το φλας (εάν υπάρχει) της συσκευής όπου την "τρέχει". Αυτό μπορεί να υλοποιηθεί πολύ εύκολα προγραμματιστικά εφόσον η βιβλιοθήκη μας παρέχει μερικές εντολές για αυτή τη λειτουργία:

```
CameraDevice.Instance.SetFlashTorchMode( true );
```

Μία ακόμα τακτική για βελτίωση της ανίχνευσης και της αναγνώρισης, είναι η χρήση εστίασης της κάμερας ανίχνευσης. Πλέον, σχεδόν όλες οι κάμερες των έξυπνων τηλεφώνων αλλά και άλλων φορητών συσκευών έχουν μηχανισμούς εστίασης εικόνας πράγμα που μας χαρίζει καλύτερη ευκρίνεια. Η βιβλιοθήκη παρέχει διάφορες εντολές για τον χειρισμό της κάμερας.

Λειτουργία εστίασης	Συμπεριφορά
FOCUS_MODE_NORMAL	Χρήση της προεπιλεγμένης λειτουργίας εστίασης, βάσει των οδηγών της κάμερας
FOCUS_MODE_TRIGGERAUTO	Ενεργοποιεί μια μοναδική αυτόματη εστίαση
FOCUS_MODE_CONTINUOUSAUTO	Ενεργοποιεί την συνεχόμενη αυτόματη εστίαση, βάση των οδηγών της κάμερας
FOCUS_MODE_INFINITY	Εστίαση σε αντικείμενα που είναι "άπειρη" απόσταση μακριά
FOCUS_MODE_MACRO	Εστίαση σε πολύ κοντινή απόσταση

Οι δύο τελευταίες λειτουργίες εστίασης είναι πολύ εξειδικευμένες και χρησιμοποιούνται κυρίως για επαγγελματικές λήψεις φωτογραφιών και χρησιμοποιούνται σπάνια για επαύξηση πραγματικότητας.

Αν και συνιστάται η χρήση της συνεχούς αυτόματης εστίασης, διότι μας παρέχει συνεχόμενα, μία εικόνα με υψηλή ευκρίνεια, μπορεί να υπάρξουν περιπτώσεις που να χρειαστούμε εξειδικευμένους τρόπους εστίασης για βέλτιστα αποτελέσματα. Είναι λοιπόν σοφό να χρησιμοποιούμε τις σχετικές επιλογές που μας παρέχει η βιβλιοθήκη ανάλογα με τη χρήση της εφαρμογής μας.

Αν εξαιρέσουμε τις περιπτώσεις στις οποίες χρησιμοποιούμε ήδη παραγόμενες εικόνες ως Image Targets για την ανίχνευσή μας, όπως π.χ. το εξώφυλλο ενός περιοδικού, πρέπει να λάβουμε υπόψιν και την εκτύπωση της εικόνας που θα χρησιμοποιήσουμε για ανίχνευση. Όπως και στη δικιά μας περίπτωση που χρησιμοποιούμε ένα QR ως εικόνα ανίχνευσης, πρέπει να προσέξουμε την ποιότητα της εκτύπωσης, για να μην αλλοιωθεί η εικόνα και δημιουργήσει προβλήματα στην ανίχνευση.

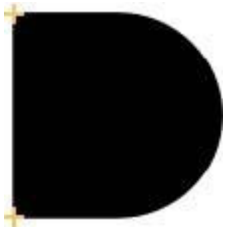
Ενδεικτικά συνιστάται να μην χρησιμοποιείται πλαστικοποιημένο χαρτί για εκτύπωση, εφόσον μπορεί να προκαλέσει αντανακλάσεις στο φως και να δυσκολέψει την αναγνώριση της εικόνας.

Αξίζει να σημειωθεί πως ένας επίσης σημαντικός παράγοντας για την ανίχνευση των στόχων που χρησιμοποιούμε για την επαύξηση είναι και η κλίση στην οποία βρίσκονται όταν η εφαρμογή μας προσπαθεί να τους ανιχνεύσει. Εάν οι εικόνες- στόχοι βρίσκονται υπό κλίση αλλοιώνονται τα οπτικά χαρακτηριστικά της εικόνας με αποτέλεσμα να μην μπορεί να εντοπιστεί από την εφαρμογή. Για βέλτιστα αποτελέσματα θα πρέπει ο στόχος να είναι καλά ευθυγραμμισμένος σε σχέση με την κάμερα και κατά προτίμηση να βρίσκεται σε επίπεδο.

Παραδείγματα εικόνων- στόχων για βέλτιστα αποτελέσματα

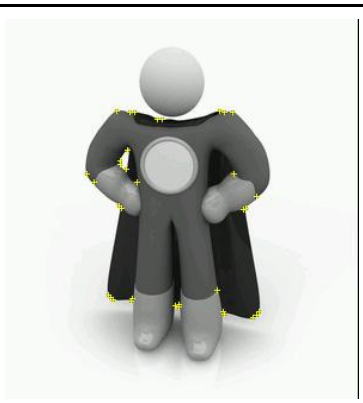
Όπως αναφέρθηκε και προηγουμένως, η Vuforia αναλύει τις εικόνες όπου χρησιμοποιούμε ως στόχους για την επαύξηση, αναλύοντας τις εικόνες αυτές και εντοπίζοντας σημεία κλειδιά (Features) πάνω στην εικόνα. Τα σημεία αυτά είναι λεπτομέρειες στην εικόνα όπως γωνίες, αντιθέσεις στα χρώματα κλπ και αναπαριστώνται από τον αναλυτή της εικόνας με μικρούς κίτρινους σταυρούς.

Σχήματα	Σημεία- Κλειδιά (Features)

	Ένα απλό τετράγωνο έχει 4 σημεία- κλειδιά, ένα για κάθε γωνία του.
	Ένας κύκλος δεν έχει σημεία εφόσον δεν έχει γωνίες.
	Αυτό το σχήμα έχει έχει δύο σημεία- κλειδιά.

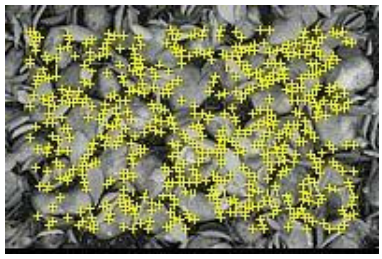
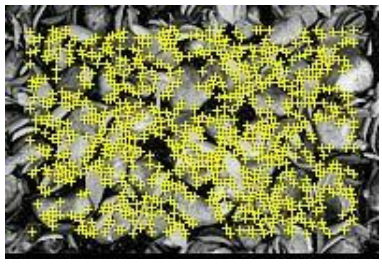
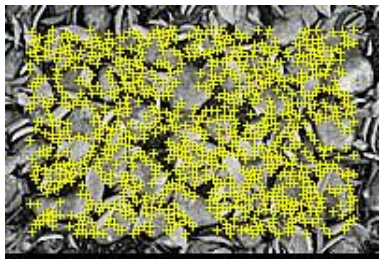
Ανάλογα με το πόσα σημεία- κλειδιά έχει η εικόνα ο αναλυτής την αξιολογεί με έναν βαθμό από το 0 έως το 5, με το 0 να σημαίνει ότι η εικόνα δεν μπορεί να αναγνωριστεί καθόλου από την βιβλιοθήκη και 5 να υποδηλώνει ότι η εικόνα είναι πολύ εύκολα ανιχνεύσιμη.

Κάνοντας πειραματισμούς και μελετώντας τα αποτελέσματα του αναλυτή για διάφορες εικόνες και λαμβάνοντας υπόψιν τις πιο πάνω παρατηρήσεις, παρατηρούμε αρχικά πως οργανικά, καμπύλα και στρογγυλά σχήματα καθιστούν δύσκολη την ανίχνευση τους.

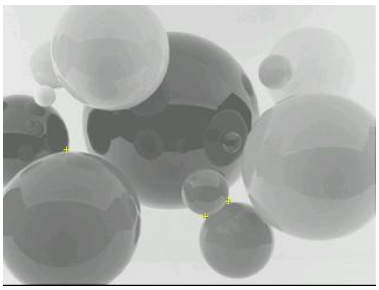
Αρχική Εικόνα	Παραγόμενη εικόνα από τον Αναλυτή	Βαθμολογία
		★☆☆☆☆ Πολύ λίγα σημεία λόγω των καμπύλων σχημάτων

Στον παρακάτω πίνακα παρατηρούμε το πως επηρεάζει η αντίθεση χρωμάτων τον αναλυτή:

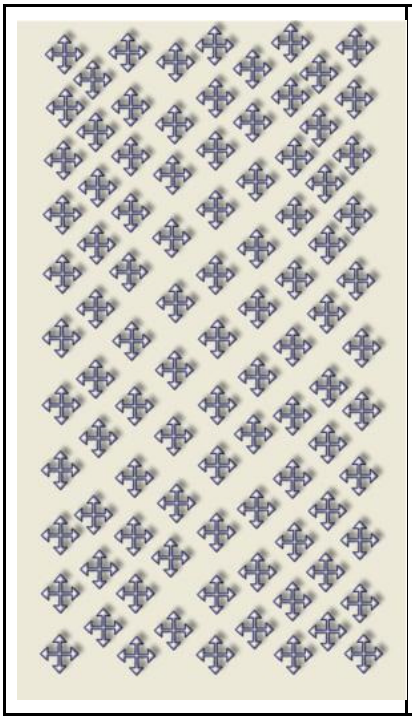
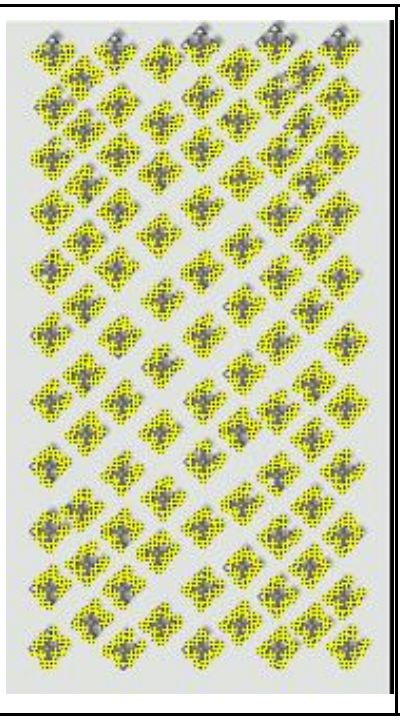
Αρχική Εικόνα	Παραγόμενη εικόνα από τον	Βαθμολογία

	Αναλυτή	
 Εικόνα χωρίς επεξεργασία		★★★☆☆
 Αύξηση αντίθεσης χρωμάτων		★★★★☆
 Υπερβολική αύξηση αντίθεσης		★★★★★

Παρατηρούμε πως πολύ καμπύλα σχήματα στην εικόνα την καθιστούν δύσκολα ανιχνεύσιμη:

Αρχική Εικόνα	Παραγόμενη εικόνα από τον Αναλυτή	Βαθμολογία
		☆☆☆☆☆ Λόγο των καμπύλων σχημάτων αλλά και του χαμηλού κοντράστ στα χρώματα ο αναλυτής δεν εντόπισε κανένα σημείο-κλειδί

Αρχική Εικόνα	Παραγόμενη εικόνα από τον Αναλυτή	Βαθμολογία

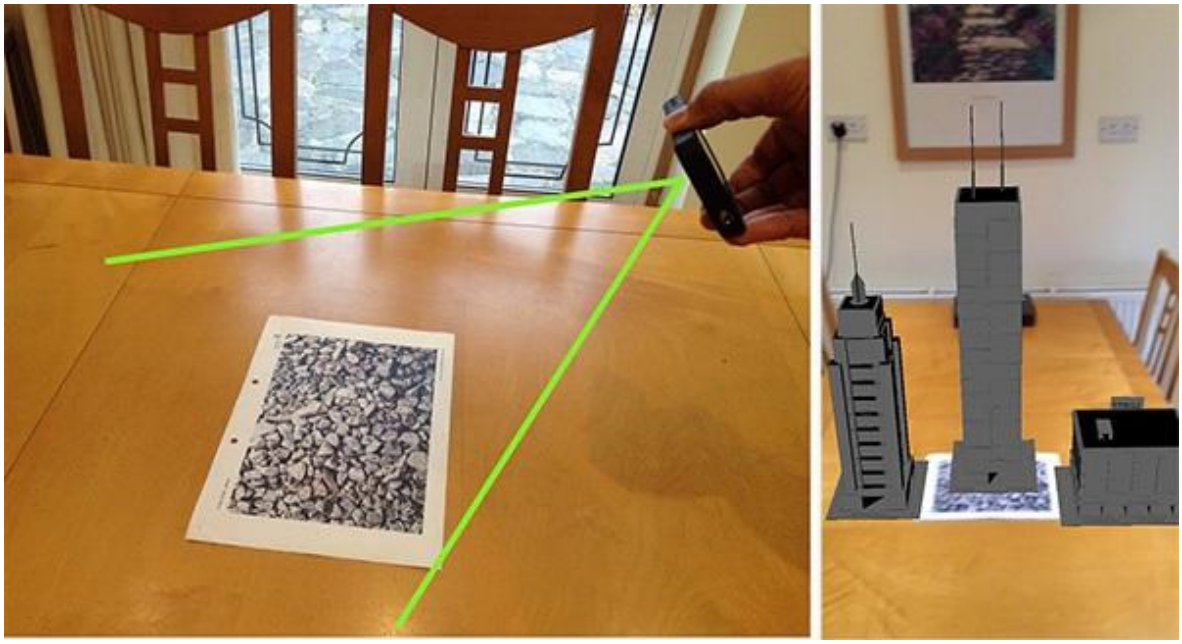
		☆☆☆☆☆ Αν και ο αναλυτής εντοπίζει πολλαπλά σημεία, το επαναλαμβανόμενο μοτίβο της εικόνας καθιστά αδύνατη την ανίχνευση της.

Εκτεταμένη Παρακολούθηση (Extended Tracking)

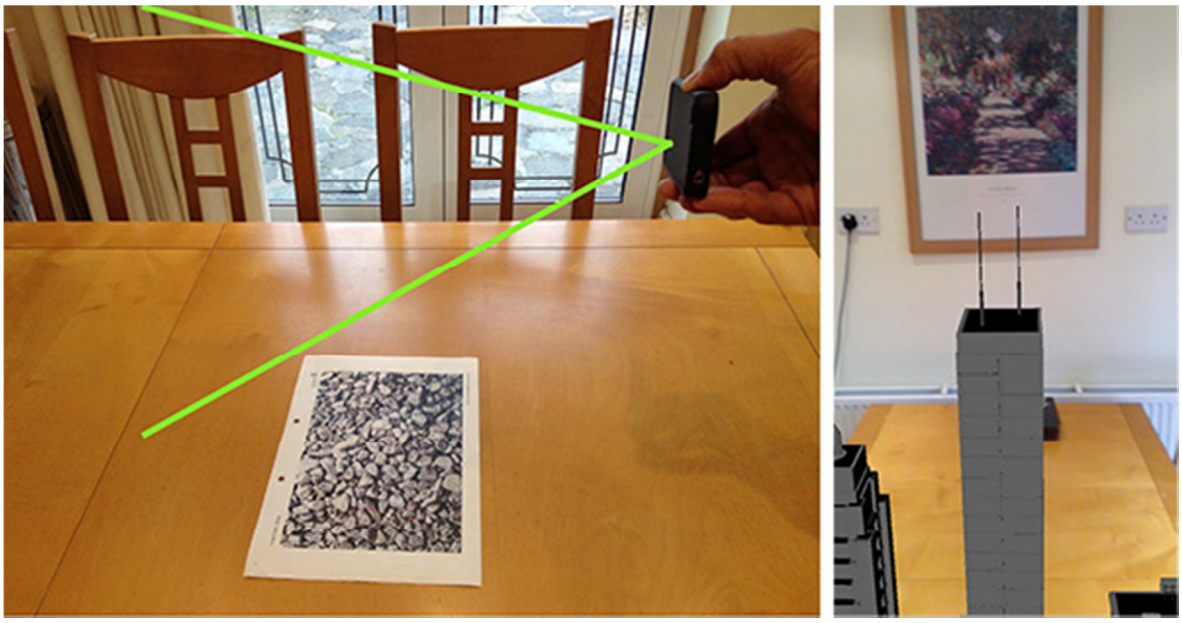
Εκτός από τις ενέργειες που είδαμε πιο πάνω που μας επιτρέπουν καλύτερη ανίχνευση και αναγνώριση, η βιβλιοθήκη της Vuforia μας παρέχει και μερικά επιπλέον προγραμματιστικά εργαλεία τα οποία μπορούν να βοηθήσουν στην ποιότητα της επαύξησης. Ένα από αυτά είναι και η λειτουργία **Εκτεταμένης Παρακολούθηση (Extended Tracking)**.

Η Εκτεταμένη Παρακολούθηση χρησιμοποιεί χαρακτηριστικά του περιβάλλοντος για τη βελτίωση της απόδοσης παρακολούθησης και τη διατήρηση της επαύξησης, ακόμη και όταν ο στόχος δεν είναι πλέον ορατός. Καθώς ο στόχος εξαλείφεται, η Vuforia χρησιμοποιεί άλλες πληροφορίες από το περιβάλλον για να συμπεράνει τη θέση στόχου παρακολουθώντας οπτικά το περιβάλλον.

Η λειτουργία επιτρέπει τη δημιουργία πιο εύρωστων εφαρμογών, επειδή οι επαυξήσεις που συνδέονται με τους στόχους που εφαρμόζουν Εκτεταμένη Παρακολούθηση θα διατηρηθούν. Στην πράξη αυτό σημαίνει ότι αφού εστιάσουμε τη συσκευή μας μακριά από τον αρχικό στόχο, η βιβλιοθήκη θα προσπαθήσει να διατηρήσει την επαύξηση με βάση τη θέση του σε σχέση με τον πραγματικό κόσμο. Όσο πιο λεπτομερές και πλούσιο σε χαρακτηριστικά το περιβάλλον, τόσο καλύτερα και τα αποτελέσματα της Εκτεταμένης Παρακολούθησης.



Εικ.5α: Η συσκευή εστιάζει στην εικόνα στόχο και η επαύξηση επιτυγχάνεται φυσιολογικά.



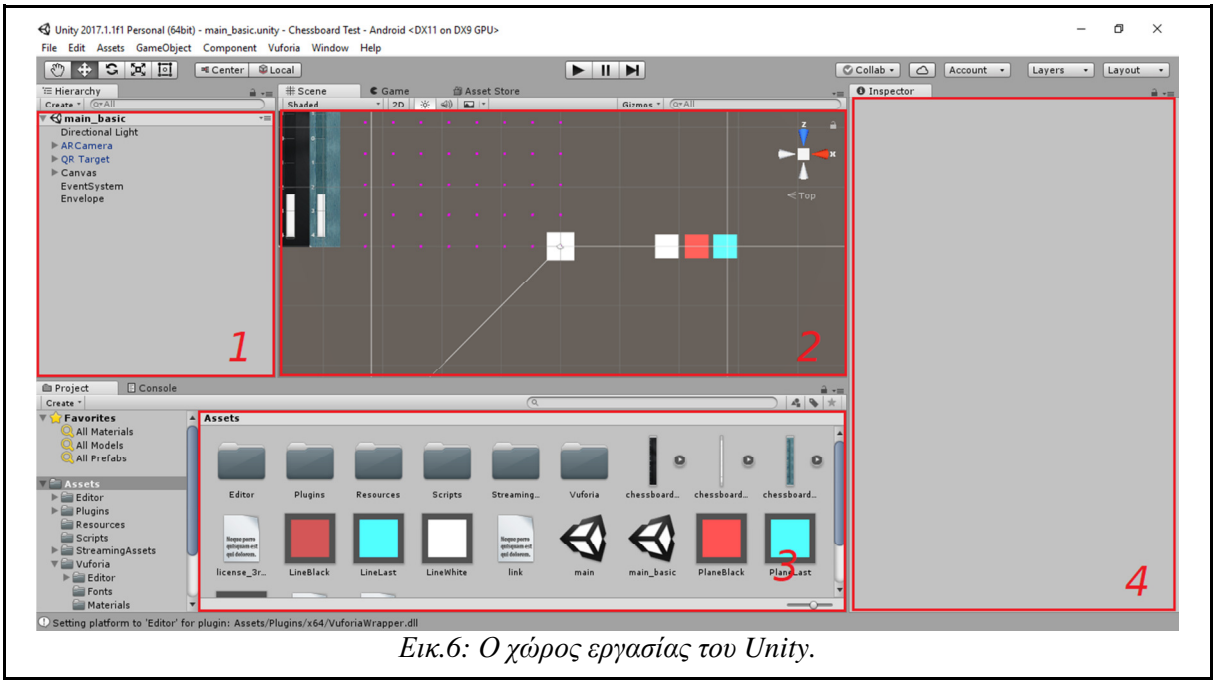
Εικ.5β: Η επαύξηση εξακολουθεί να υλοποιείται λόγω της Εκτεταμένης Παρακολούθησης παρόλο που ο στόχος δεν είναι ορατός από τη συσκευή.

Αφού η επαύξηση ολοκληρωθεί με τη χρήση Εκτεταμένης Παρακολούθησης δεν χρειάζεται πλέον να είναι ορατός ο αρχικός στόχος και επίσης μπορούν και άλλα εικονικά αντικείμενα να εισαχθούν στον "χώρο". Τα νέα αντικείμενα αυτά θα τοποθετηθούν ανάλογα με τη θέση του αρχικού στόχου και του περιβάλλοντος γύρω του. Αν και η λειτουργία αυτή προσφέρει πολύ καλά αποτελέσματα επαύξησης αξίζει να σημειωθεί πως απαιτείται μεγαλύτερη επεξεργαστική ισχύς από τη συσκευή.

Το Περιβάλλον του Unity

Εφόσον έχουμε τα προαπαιτούμενα για την εφαρμογή μας και έχουμε λάβει υπόψιν τις μεθόδους βέλτιστης επαύξησης, ας δούμε λίγο και το περιβάλλον στο οποίο θα δουλέψουμε.

Όπως αναφέρθηκε και στις προηγούμενες σελίδες η εφαρμογή μας θα υλοποιηθεί με την μηχανή Unity. Ας ρίξουμε λοιπόν μία γρήγορη ματιά στον **χώρο εργασίας (workspace)** που το Unity μας παρέχει:

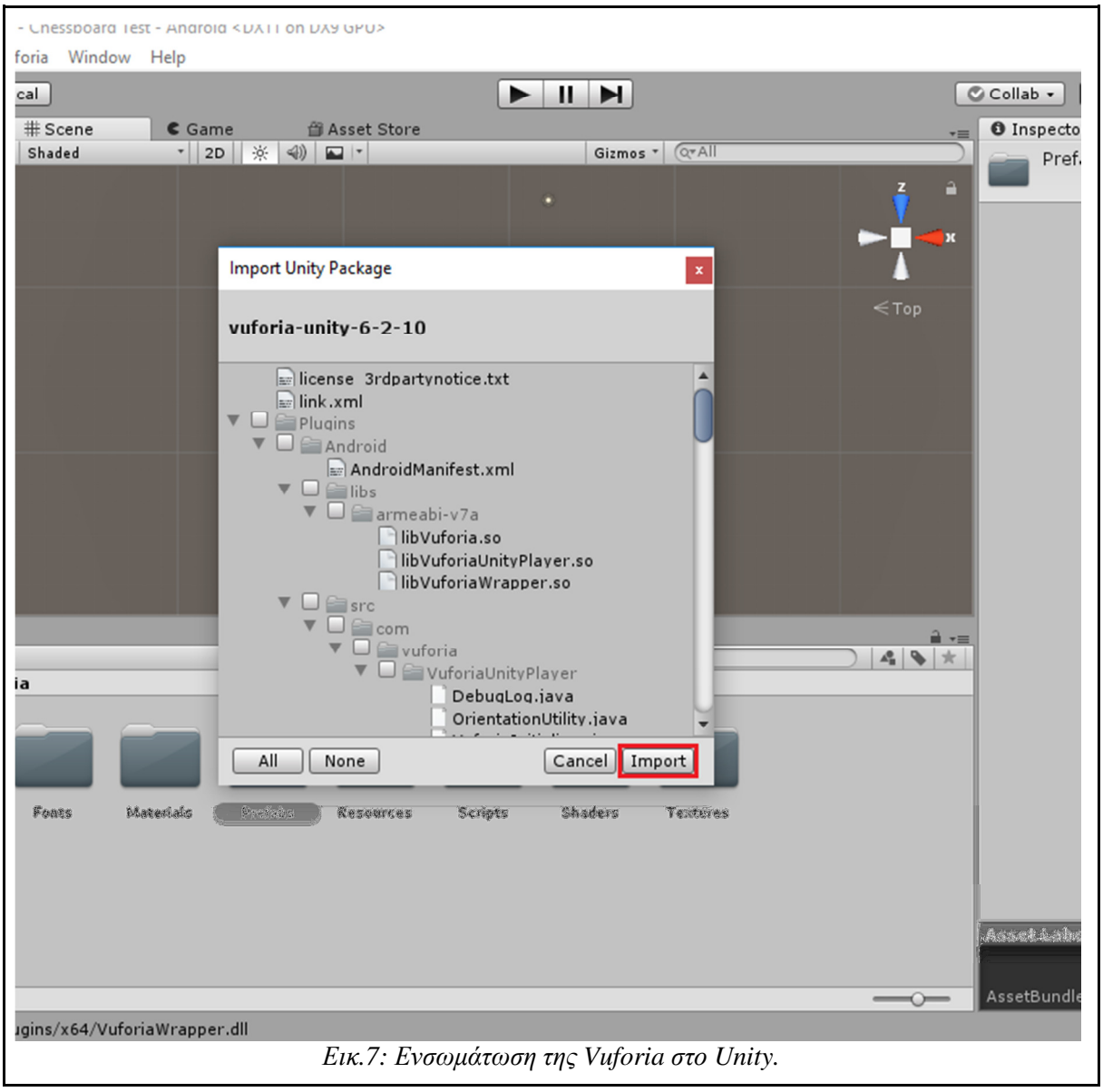


Εικ.6: Ο χώρος εργασίας του Unity.

1 - Ιεραρχία (Hierarchy)	Εδώ απεικονίζονται όλα τα **Αντικείμενα (Objects)** που υπάρχουν στη Σκηνή. Μέσω της Ιεραρχίας μπορούμε να δημιουργήσουμε Γονείς και Τέκνα μέσω των Αντικειμένων. **(Parents & Childs)**
2 - Σκηνή (Scene View)	Αυτός είναι ο "κόσμος" μας. Ό,τι τοποθετείται εδώ θα εμφανίζεται και στην εφαρμογή μας.
3 - Εργαλεία (Assets)	Εδώ βρίσκονται όλα τα εργαλεία που χρειάζεται η εφαρμογή μας, από μικρά scripts μέχρι και αρχεία 3D μοντέλων.
4 - Επιθεωρητής (Inspector)	Σε αυτό το πάνελ μπορούμε να βλέπουμε τα παραμετροποιήσημα χαρακτηριστικά των Αντικείμενα που επιλέγουμε στη σκηνή.

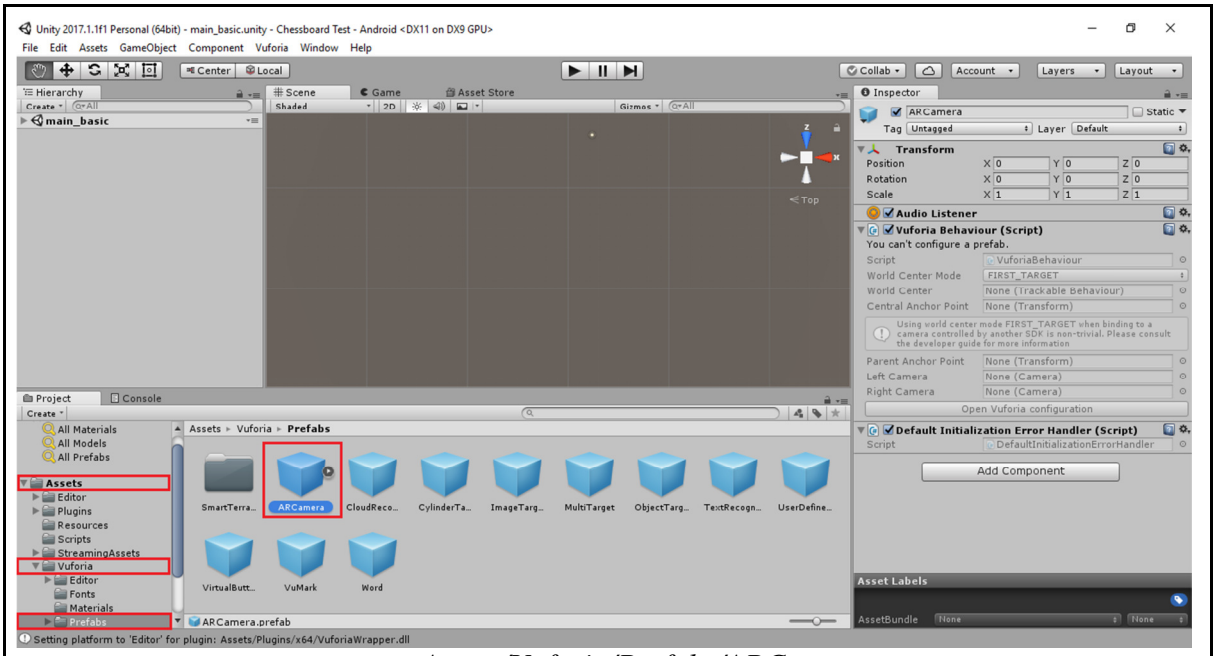
Επαυξημένη Κάμερα (AR Camera) στο Unity

Αφού έχουμε κατεβάσει την βιβλιοθήκη και με επιτυχία εγκαταστήσουμε το Unity απλά "ρίχνουμε" (Drag & Drop) το πακέτο της Vuforia μέσα στο Unity για να την ενσωματώσουμε και πατάμε *Import* στο αντίστοιχο παράθυρο που μας παρουσιάζεται.



Εικ.7: Ενσωμάτωση της Vuforia στο Unity.

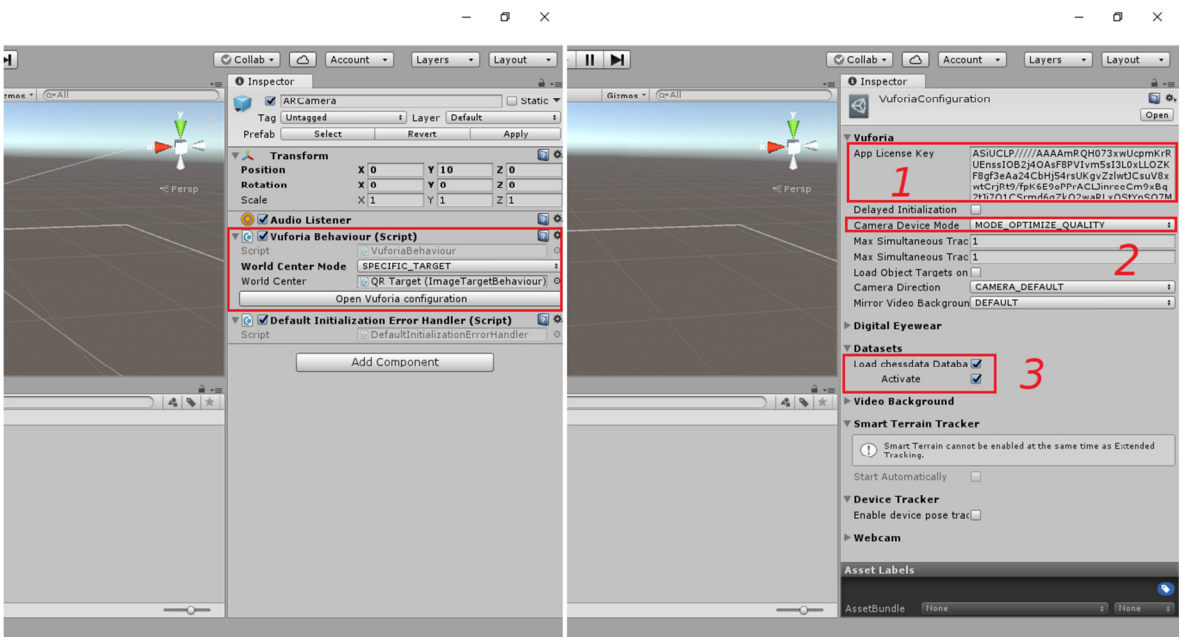
Τώρα λοιπόν έχουμε πρόσβαση στη βιβλιοθήκη και στα εργαλεία που αυτή μας παρέχει μέσω του παραθύρου *Assets*. Τώρα μπορούμε να αρχίσουμε να στήνουμε την *Σκηνή* μας. Αρχικά θα χρειαστούμε να τοποθετήσουμε την κάμερά μας στην *Σκηνή*. Η Vuforia μας παρέχει ένα έτοιμο Αντικείμενο το οποίο διαχειρίζεται την κάμερα της συσκευής και όλες τις λειτουργίες της Vuforia για την επαύξηση. Το μόνο που έχουμε να κάνουμε είναι να το εντοπίσουμε στο μονοπάτι: *"Assets/Vuforia/Prefabs/ARCamera"* και να την "σύρρουμε" στην σκηνή.



Εικ.8: Assets/Vuforia/Prefabs/ARCamera.

Τέλος αρκεί να κάνουμε μια παραμετροποίηση στην κάμερα και να τοποθετήσουμε το "κλειδί" που δημιουργήσαμε κατά την εγγραφή μας στο site της Vuforia (σελ.16). Μέσω του Επιθεωρητή μπορούμε να τροποποιήσουμε τις ρυθμίσεις της κάμερας όπως στην εικόνα.

1) Τοποθετούμε το μοναδικό κλειδί. Έτσι ενεργοποιείται η κάμερα μας.
2) Η παράμετρος Camera Device Mode μας παρέχει μερικές λειτουργίες οι οποίες εστιάζουν στην ποιότητα εικόνας της κάμερας και της απόδοσης. Επιλέγουμε την λειτουργία MODE_OPTIMIZE_QUALITY για καλύτερη ευκρίνεια.
3) Στο σημείο αυτό επιλέγουμε την βάση δεδομένων που εμπεριέχει μέσα την εικόνα-στόχο για να μπορέσουμε να επαυξήσουμε εικονικά αντικείμενα. Αξίζει να σημειωθεί πως πριν μας επιτραπεί να ενεργοποιήσουμε την βάση πρέπει να την εισάγουμε (Import) στο Unity όπως εισάγαμε και την βιβλιοθήκη της Vuforia προηγουμένως.



Εικ.9: Ρυθμίσεις της κάμερας.

Η ρύθμιση World Center Mode δείχνει το σημείο αναφοράς για τα επαυξημένα αντικείμενα. η Vuforia μας επιτρέπει 4 διαφορετικές επιλογές για να καθορίσουμε το "κέντρο" της Σκηνής μας:

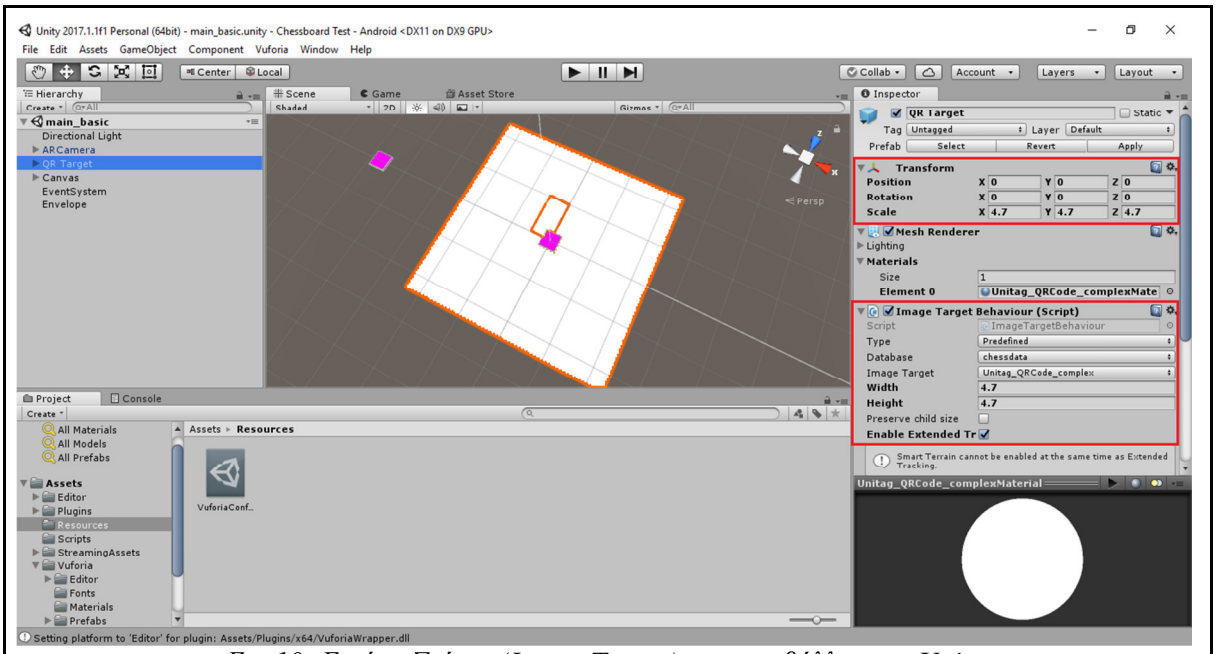
SPECIFIC_TARGET	Όλα τα αντικείμενα δημιουργούνται αναλογικά με μία συγκεκριμένη εικόνα-στόχο την οποία ορίζουμε εμείς και οι αποστάσεις των αντικειμένων παρουσιάζονται εν συναρτήσει αυτής όπως τα τοποθετούμαι στην Σκηνή. (Στην προκειμένη περίπτωση έχουμε ορίσει ως στόχο το QR που δημιουργήσαμε.)
FIRST_TARGET	Ίδια λειτουργία με την SPECIFIC_TARGET με την εξαίρεση ότι δεν καθορίζουμε την εικόνα στόχο εξαρχής αλλά η πρώτη εικόνα που ανιχνεύεται από την κάμερα λειτουργεί ως κέντρο.
CAMERA	Το κέντρο του κόσμου είναι η ίδια η κάμερα. Όλα τα αντικείμενα στη Σκηνή μας θα μετακινηθούν αναλογικά με τη θέση της κάμερας.
DEVICE_TRACKING	Χρησιμοποιείται κυρίως για VR εφαρμογές. Δίνει την δυνατότητα να αναγνωρίζει η Vuforia περιστροφικές κινήσεις.

Αξίζει να σημειωθεί πως η επιλογή DEVICE_TRACKING μπορεί να χρησιμοποιηθεί για την δημιουργία AR μενού επιλογών εφόσον επιτρέπει την αναγνώριση περιστροφικών κινήσεων.

Τέτοιες κινήσεις μπορούν να είναι ο προσανατολισμός της κεφαλής του χρήστη (Smart Glasses) ή το χέρι του.

Τοποθέτηση της Εικόνας Στόχου QR

Στη συνέχεια πρέπει να στήσουμε και την εικόνα στόχο για να μπορέσουμε να απεικονίσουμε αργότερα εικονικά αντικείμενα. Όπως και με την κάμερά μας προηγουμένως έτσι και για την εικόνα μας θα χρησιμοποιήσουμε ένα έτοιμο αντικείμενο (Prefab) το οποίο βρίσκεται στη θέση: "*Assets/Vuforia/Prefabs/ImageTarget*". Κάνοντας το drag & drop στην Σκηνή μας, ο Επιθεωρητής θα μας εμφανίσει τις ρυθμίσεις του:

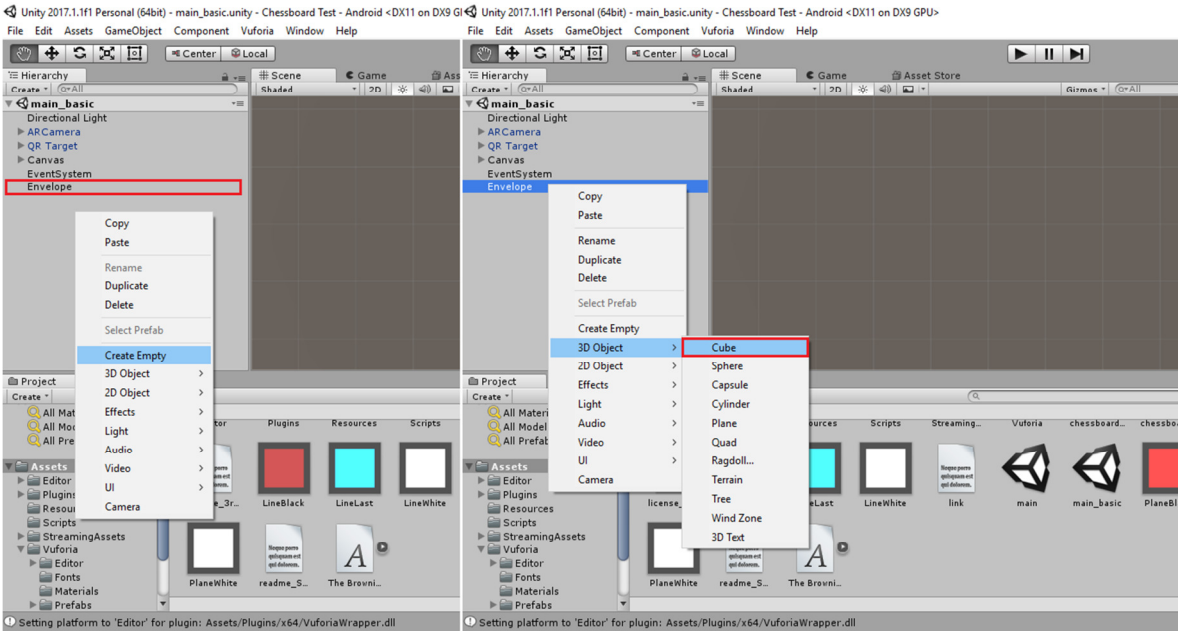


Εικ.10: Εικόνα Στόχος (Image Target) στο περιβάλλον του Unity.

Αρχικά τοποθετούμε την εικόνα στο σημείο (0,0,0) στη Σκηνή μας επιλέγοντας τις αντίστοιχες παραμέτρους στο Position. Το Scale τοποθετείται αυτόματα στο 4.7 εφόσον τόσο είναι το πλάτος της εικόνας μας. Τέλος επιλέγουμε την λειτουργία Extended Tracking για βέλτιστα αποτελέσματα. Αξίζει να σημειωθεί πως το Width και Height στο Image Target Behaviour είναι ίδια με τα X και Y Scale στο Transform. Τοποθετούμε τον στόχο στο (0,0,0) για μεγαλύτερη ευκολία στο να τοποθετήσουμε αργότερα τα αντικείμενα που θέλουμε να επαυξήσουμε. Έτσι θα έχουμε ως σημείο αναφοράς την εικόνα για να πετύχουμε αναλογικές αποστάσεις μεταξύ των αντικειμένων. Αφού όλα τα Αντικείμενα τοποθετηθούν στη Σκηνή, αργότερα μπορούμε να μετακινήσουμε την εικόνα σε ένα άλλο σημείο εάν θέλουμε.

Όπως αναφέρθηκε και στην περιγραφή, η εφαρμογή θα απεικονίζει με ένα βέλος την τελευταία και την βέλτιστη κίνηση του αντίστοιχου παίχτη στον γύρο πάνω στο ταμπλό.

Θα χρειαστεί να δημιουργήσουμε αντικείμενα τα οποία θα αντιστοιχούν σε κάθε κελί της σκακιέρας. Με αυτόν τον τρόπο θα μπορούμε να δημιουργήσουμε το βέλος που θα απεικονίζει την κίνηση έχοντας ως είσοδο δυο κελιά του ταμπλό, ένα για την αφετηρία του βέλους και ένα για τον προορισμό.



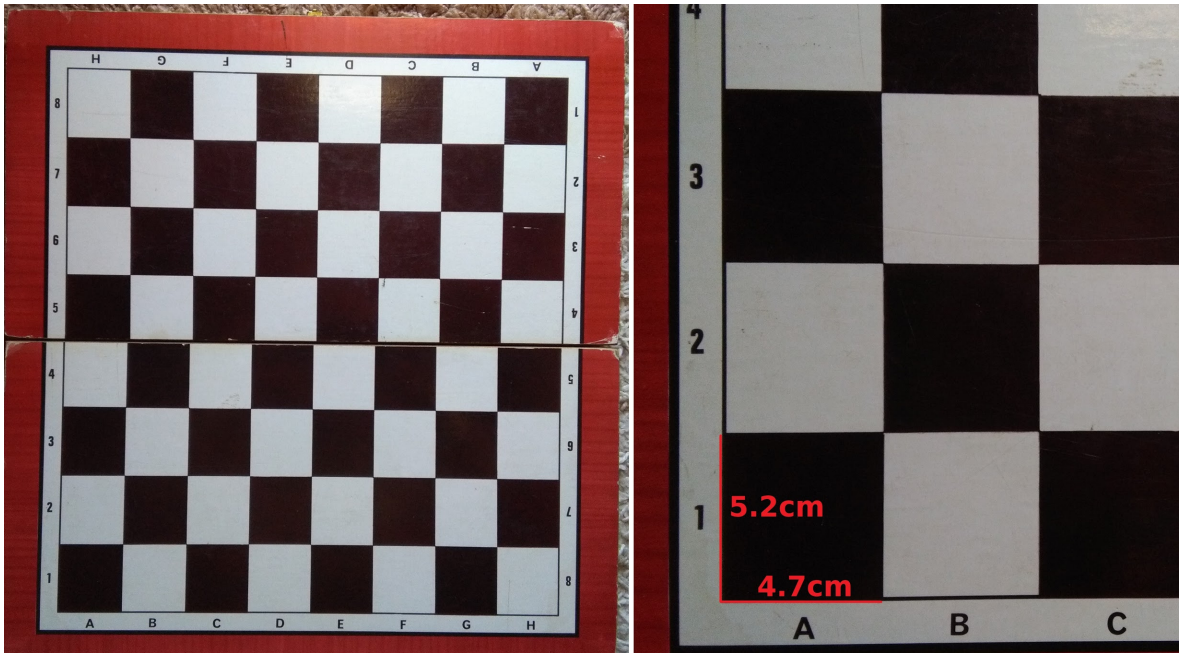
Εικ.11: Δημιουργία Κενού Αντικειμένου και 3D Κύβου.

Κάνοντας δεξί κλικ στην Ιεραρχία μπορούμε να δημιουργήσουμε ένα "Κενό Αντικείμενο" (Envelope) το οποίο θα το χρησιμοποιήσουμε ως "φάκελο" για να τοποθετήσουμε μέσα τα αντικείμενα τα οποία θα αντιστοιχούν στα κελιά της σκακιέρας. Στη συνέχεια κάνουμε δεξί κλικ στο κενό αντικείμενο δημιουργούμε έναν κύβο για κάθε κελί.

Λόγω του ότι τα κελιά τις σκακιέρας είναι 64, τόσα πρέπει να είναι και τα αντικείμενα που θα δημιουργήσουμε. Τοποθετώντας τα μέσα σε ένα κενό αντικείμενο, οργανώνουμε καλύτερα την Ιεραρχία μας και επίσης είναι πιο εύκολη η διαχείρισή τους εντός Σκηνής.

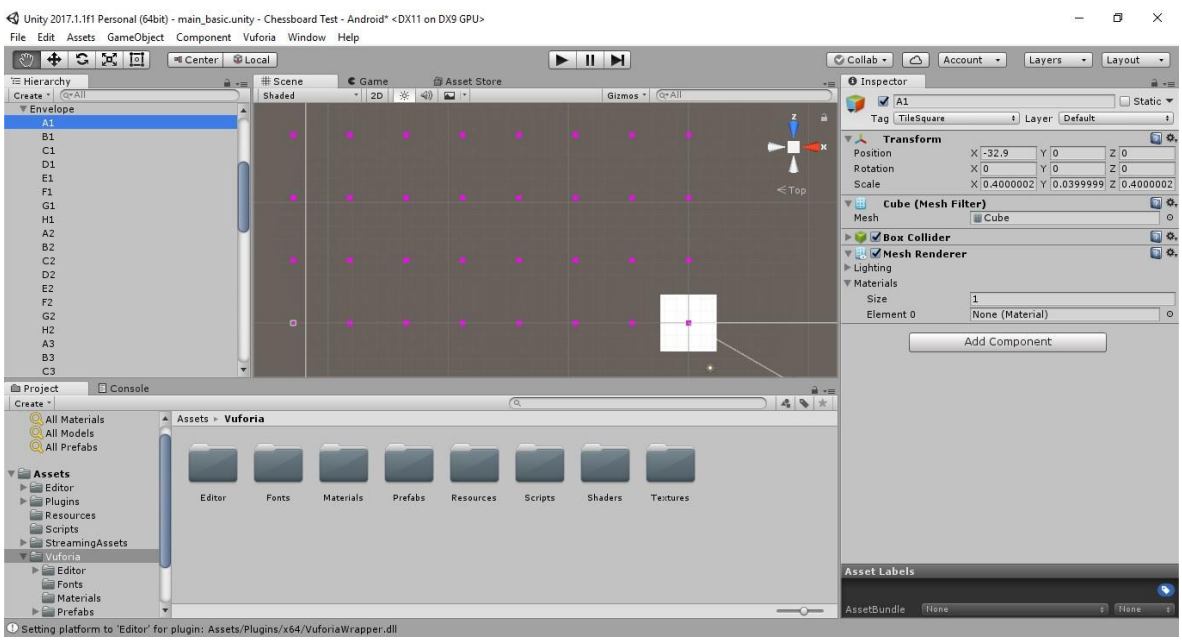
Ίσως πιο σωστό θα ήταν να δημιουργήσουμε προγραμματιστικά αυτά τα αντικείμενα και όχι χειροκίνητα. Η αλήθεια είναι όμως πως με αυτόν τον τρόπο η εφαρμογή χρειάζεται λιγότερη επεξεργαστική ισχύς για να λειτουργήσει διότι τα αντικείμενα είναι τοποθετημένα εξαρχής ανάλογα με τη διάσταση της σκακιέρας.

Το αμέσως επόμενο ερώτημα είναι: *"Ποίες είναι οι αποστάσεις ανάμεσα στα Αντικείμενα αυτά για να αναπαριστούν τα κέντρα των κελιών της σκακιέρας;"*



Εικ.12: Η σκακιέρα που χρησιμοποιήσαμε ως αναφορά.

Η εικονιζόμενη σκακιέρα χρησιμοποιήθηκε για της δοκιμές της εφαρμογής. Το μέγεθος του κάθε κελιού είναι 4.7cm πλάτος και 5.2cm ύψος. Επομένως αυτές θα είναι και οι αποστάσεις από τα κέντρα κάθε κελιού. Έτσι το μόνο που μένει είναι να δημιουργήσουμε τα αντικείμενα στο Unity και να τα ονοματίσουμε αντίστοιχα με τα ονόματα στη σκακιέρα (A1,B1,κλπ).



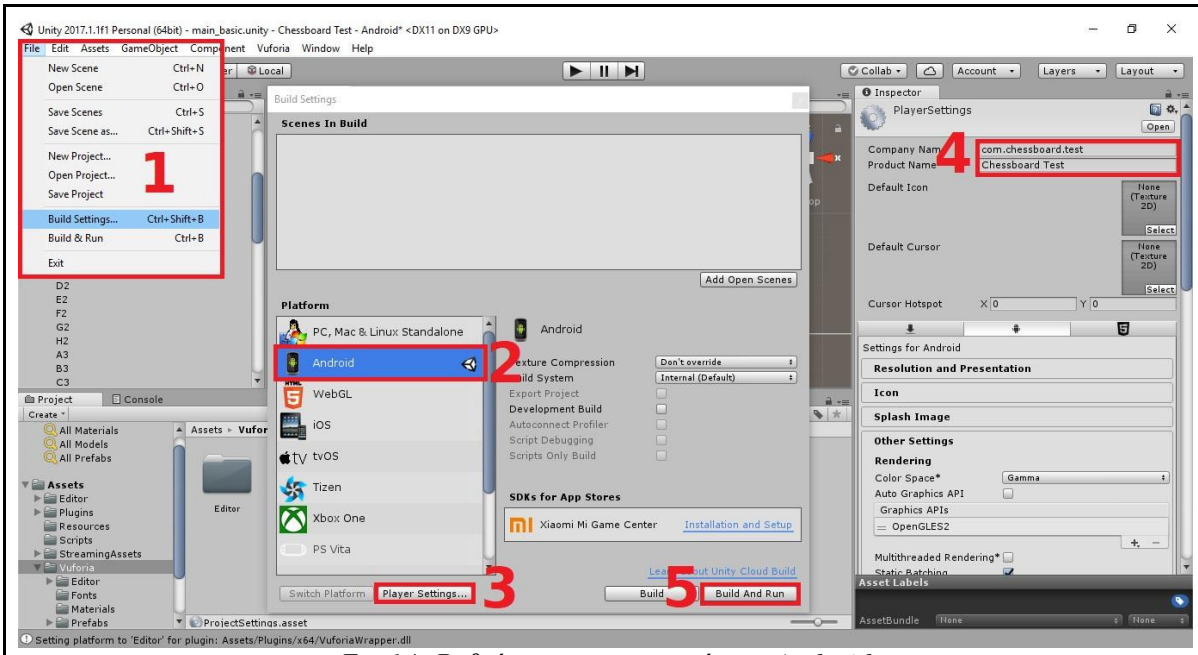
Εικ.13: Και τα 64 αντικείμενα που αντιστοιχούν στα κελιά της σκακιέρας τοποθετημένα μέσα στο κενό αντικείμενο (Envelope).

Όπως φαίνεται και στην εικόνα για λόγους ευκολίας τοποθετήσαμε το αντικείμενο για το κελί H1, καθώς και την εικόνα του QR στο (0,0,0). Έτσι μπορούμε να υπολογίσουμε καλύτερα τις αποστάσεις των αντικειμένων μέσο του Επιθεωρητή. Επίσης για να γίνει η επαύξηση των

αντικειμένων αυτών πρέπει να τοποθετήσουμε μέσω της Ιεραρχίας, το Envelope μέσα QR Target.

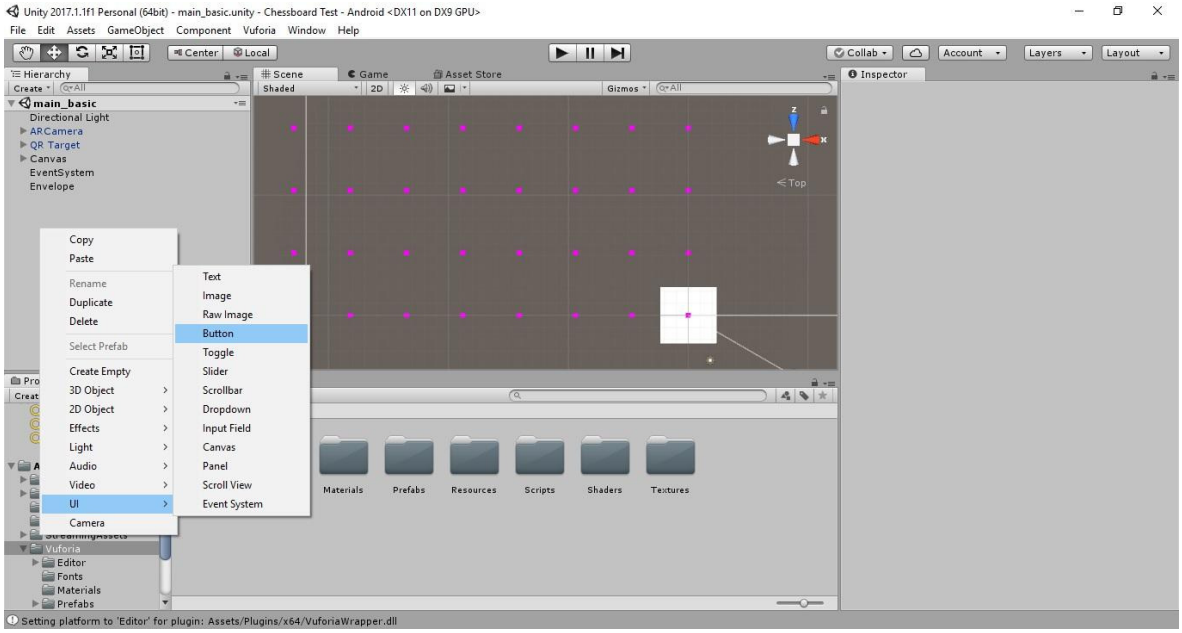
Στο σημείο αυτό μπορούμε να κάνουμε την πρώτη δοκιμή της εφαρμογής μας:
Στο μενού "*File/Build Settings*" μπορούμε να βρούμε τις ρυθμίσεις για να τρέξουμε την εφαρμογή μας. Επιλέγουμε την πλατφόρμα Android και στη συνέχεια πατάμε Player Settings για να ανοίξουμε τις ρυθμίσεις Android. Στον Επιθεωρητή πρέπει να δώσουμε ένα μοναδικό όνομα στην εφαρμογή μας και ένα μοναδικό όνομα πακέτου. Ως προεπιλογή χρησιμοποιείται το ανάποδο όνομα ιστοσελίδων, εφόσον κάθε ιστοσελίδα είναι μοναδική. Τέλος πατάμε Build And Run και περιμένουμε να ολοκληρωθεί η διαδικασία.

Σημείωση: Στη συσκευή που επιλέγουμε να τρέξουμε την εφαρμογή πρέπει να έχουμε ενεργοποιήσει τις ρυθμίσεις: "Εγκατάσταση από Άγνωστες Πηγές" και "Εντοπισμός Σφαλμάτων USB" από τις "Επιλογές Για Προγραμματιστές".



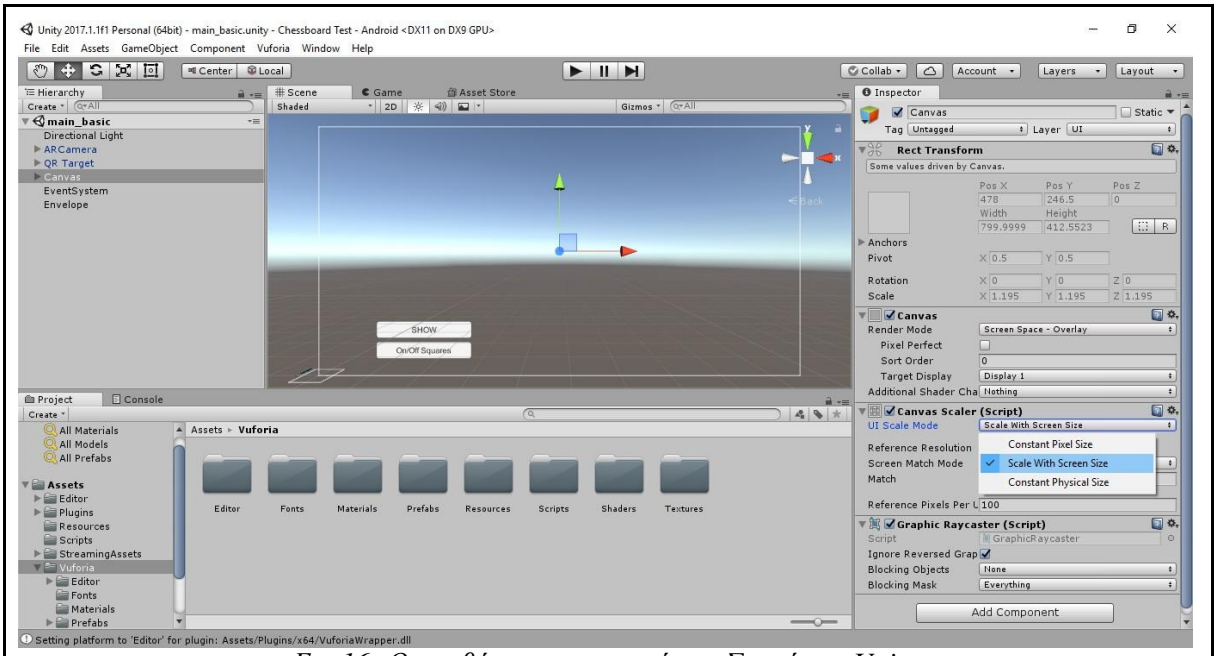
Εικ.14: Ρυθμίσεις της εφαρμογής για Android.

Για την εφαρμογή θα χρειαστούμε ένα περιβάλλον διεπαφής (User Interface) το οποίο θα μπορεί να χρησιμοποιεί ο χρήστης. Το περιβάλλον αυτό θα είναι απλό, παρέχοντας ένα κουμπί το οποίο θα ενεργοποιεί την επαύξηση όταν πατηθεί. Επίσης θα περιέχει ένα κουμπί επιπλέον το οποίο θα αφαιρεί τα κουτάκια από τα κελιά του ταμπλό και θα χρησιμοποιείται κυρίως για τον εντοπισμό σφαλμάτων (Debugging).



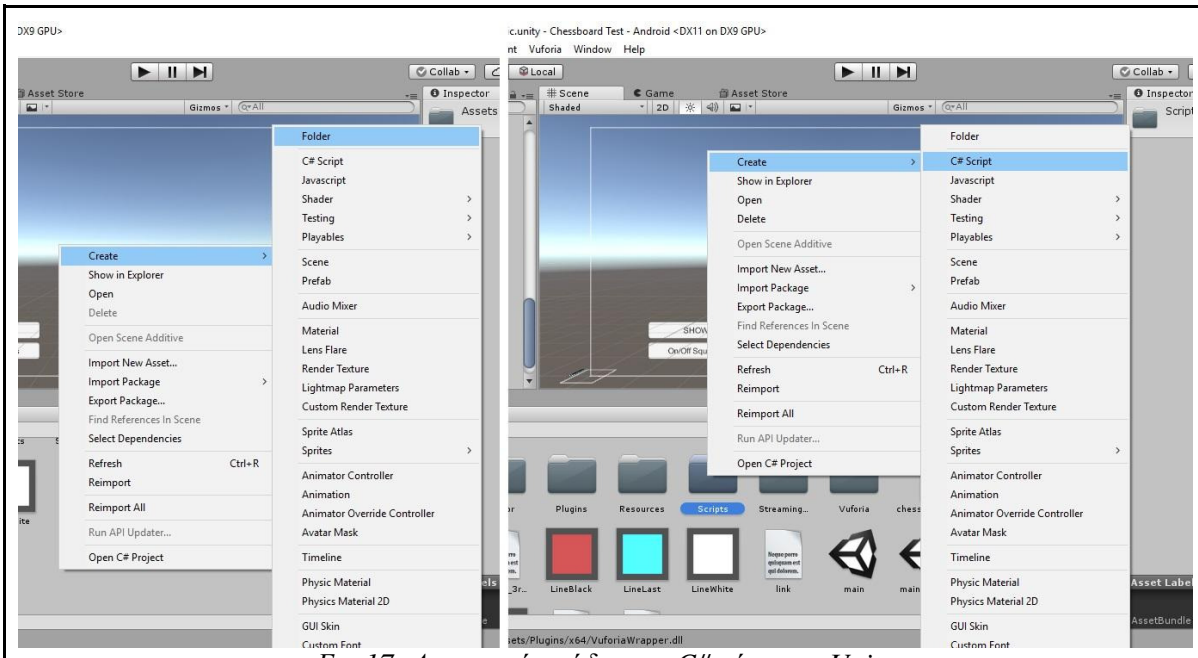
Εικ.15: Τοποθέτηση περιβάλλοντος διεπαφής.

Μέσο της Ιεραρχίας δημιουργούμε δύο κουμπιά. Αυτομάτως το Unity μας δημιουργεί έναν 2D Αντικείμενο με το όνομα Canvas οπού εκεί τοποθετούνται τα κουμπιά και ότι άλλο δισδιάστατο αντικείμενο δημιουργήσουμε. Στον Επιθεωρητή επιλέγουμε τη ρύθμιση Scale With Screen Size στο UI Scale Mode για να καλύψουμε τις περιπτώσεις χρήσης σε όλες τις οθόνες φορητών συσκευών, καθώς η λειτουργία αυτή επιτρέπει στο UI να διαμορφώνεται ανάλογα με την οθόνη της συσκευής. Εντός της Σκηνής τοποθετούμε τα κουμπιά σε όποιο σημείο του καμβά θέλουμε.



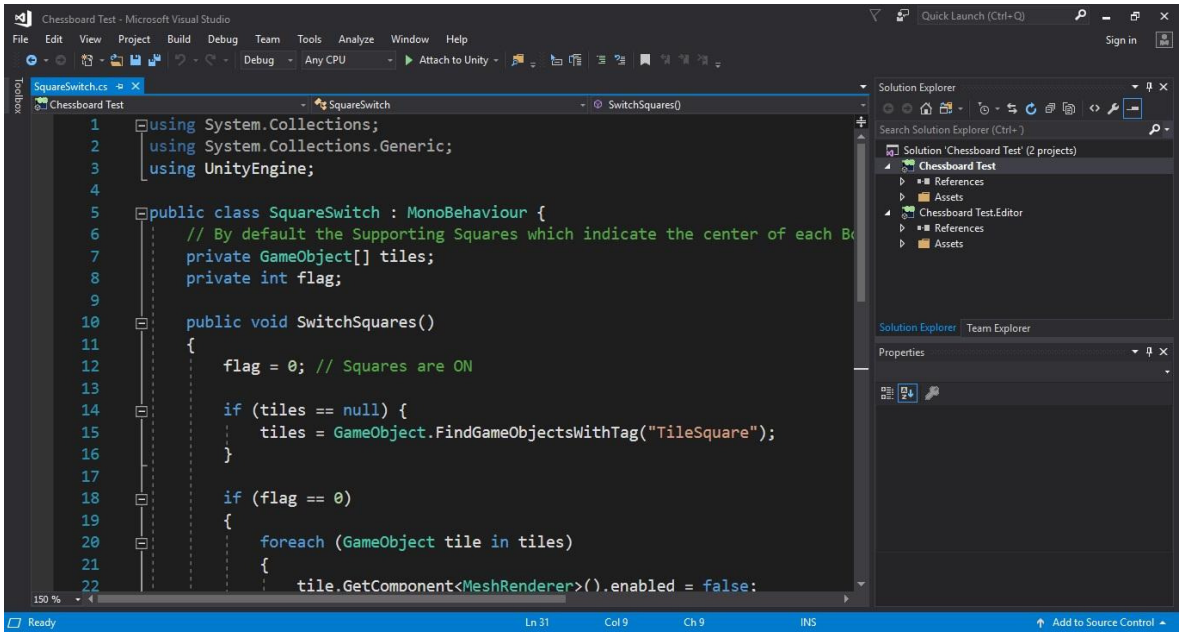
Εικ.16: Ο καμβάς με τα κουμπιά στη Σκηνή του Unity.

Ας δημιουργήσουμε στο κουμπί "On/Off Squares" που φτιάξαμε, τον κώδικα για να αφαιρεί τα κουτάκια από τα κελιά της σκακιέρας. Αρχικά κάνοντας δεξί κλικ στα Εργαλεία (Assets) δημιουργούμε ένα νέο φάκελο που τον ονομάζουμε "Scripts" και εκεί στη συνέχεια δημιουργούμε ένα κομμάτι κώδικα C# (C# Script). Στον φάκελο αυτόν θα τοποθετούμε όλους τους κώδικες που θα γράφουμε.



Εικ.17: Δημιουργία κώδικα σε C# μέσα στο Unity.

Ονομάζουμε το αρχείο "SquareSwitch" και κάνουμε διπλό κλικ. Αυτό θα ανοίξει το Microsoft Visual Studio οπού εκεί μπορούμε να επεξεργαστούμε τον κώδικα.



Εικ.18: Το περιβάλλον του Visual Studio.

Ο κώδικας που γράφουμε είναι ο εξής:

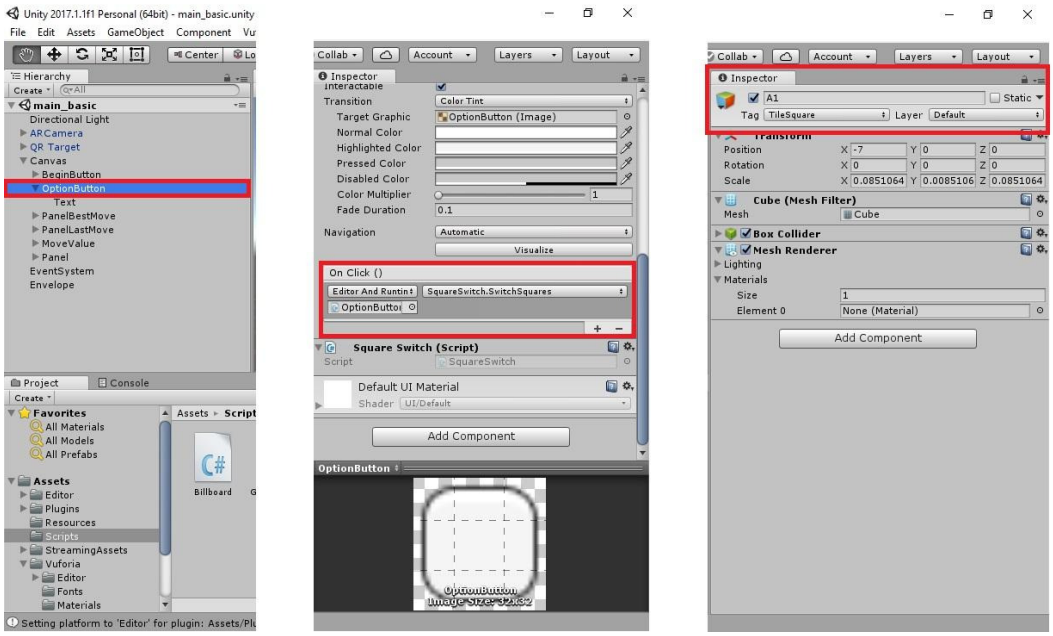
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class SquareSwitch : MonoBehaviour {

// By default the Supporting Squares which indicate the
// center of each Board Tile are ON

private GameObject[] tiles;
private int flag;

public void SwitchSquares()
{
  flag = 0; // Squares are ON
  if (tiles == null) {
  tiles = GameObject.FindGameObjectsWithTag("TileSquare");
  }
  if (flag == 0){
    foreach (GameObject tile in tiles){
      tile.GetComponent<MeshRenderer>().enabled = false;
    }
  } else{
    foreach (GameObject tile in tiles){
      tile.GetComponent<MeshRenderer>().enabled = true;
    }
  }
```

```
  tiles = null;
  flag = 1; // Squares are OFF
  }
}
```

Η λειτουργία του κώδικα είναι ως εξής: Αρχικά όλα τα GameObjects με ετικέτα “TileSquare” αποθηκεύονται σε έναν πίνακα. στη συνέχεια γίνεται ένας έλεγχος αν τα κουτάκια είναι φανερά ή όχι (flag = 0 ή flag = 1) και αναλόγως το script ενεργοποιεί ή απενεργοποιεί τον MeshRenderer από κάθε Αντικείμενο.

Ο MeshRenderer είναι μία ιδιότητα που μπορούν να έχουν τα Αντικείμενα που τους επιτρέπει το να έχουν σχήμα, υλικό κλπ. απενεργοποιώντας τον MeshRenderer το Unity τα κάνει να φαίνονται αόρατα.

Τέλος για να λειτουργήσει ο κώδικας μας πρέπει να δημιουργήσουμε και να τοποθετήσουμε μία νέα ετικέτα σε όλα τα αντικείμενα - κελιά στον Επιθεωρητή και να ενημερώσουμε το κουμπί στον καμβά μας να ενεργοποιεί το script αυτό.



Εικ.19: Ενεργοποίηση του Button και επιλογή “Ετικετών”.

Αναπαράσταση Κινήσεων

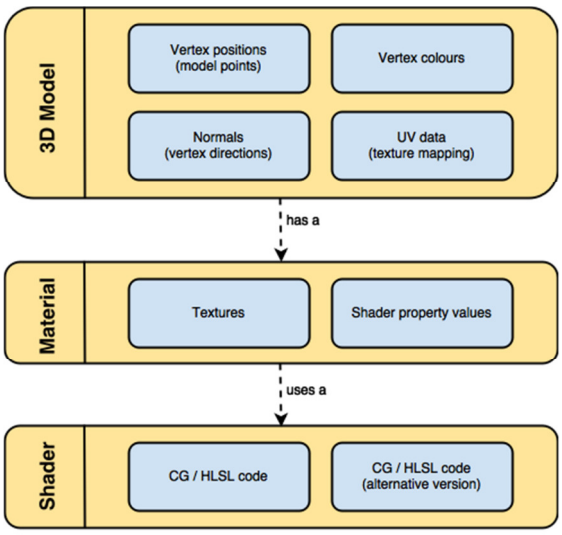
Όπως αναφέρθηκε και στην περιγραφή της εφαρμογής, οι πληροφορίες όπου θα αναπαρίστανται στην οθόνη θα παρέχονται ως είσοδο στην εφαρμογή. Η γενική ιδέα είναι πως τις πληροφορίες θα τις αποκτούμε μέσω HTML Scraping από ιστοσελίδες που τρέχουν live αγώνες σκακιού. Έτσι η εφαρμογή μας θα δέχεται ως είσοδο 2 κελιά του ταμπλό και θα αναπαριστά την κίνηση από το πιόνι που βρίσκεται στο αντίστοιχο κελί.

Δημιουργούμε λοιπόν στα Assets ένα νέο κομμάτι κώδικα σε C# και του δίνουμε το όνομα "RenderingLine". Επίσης δημιουργούμε στην Ιεραρχία δύο νέα αντικείμενα και τους δίνουμε από τον Επιθεωρητή την ιδιότητα Particles/Additive στο Script των Shaders καθώς και από ένα χρώμα. Πατώντας το κουμπί "Add Component" τους δίνουμε το LineRenderer script, επίσης από τον Επιθεωρητή. Ο κώδικας LineRenderer που παρέχεται από το Unity μας επιτρέπει να δημιουργούμε τρισδιάστατες γραμμές στη Σκηνή μας. Τέλος κάνουμε Drag & Drop το αρχείο σε C# που μόλις δημιουργήσαμε.

Προτού προχωρήσουμε περαιτέρω, πρέπει πρώτα να κατανοήσουμε την λειτουργία των Shaders:

Τα **Shaders** είναι κομμάτια κώδικα τα οποία τρέχουν εξολοκλήρου από την κάρτα γραφικών της συσκευής και έχουν ως σκοπό την απεικόνιση των 3D μοντέλων.

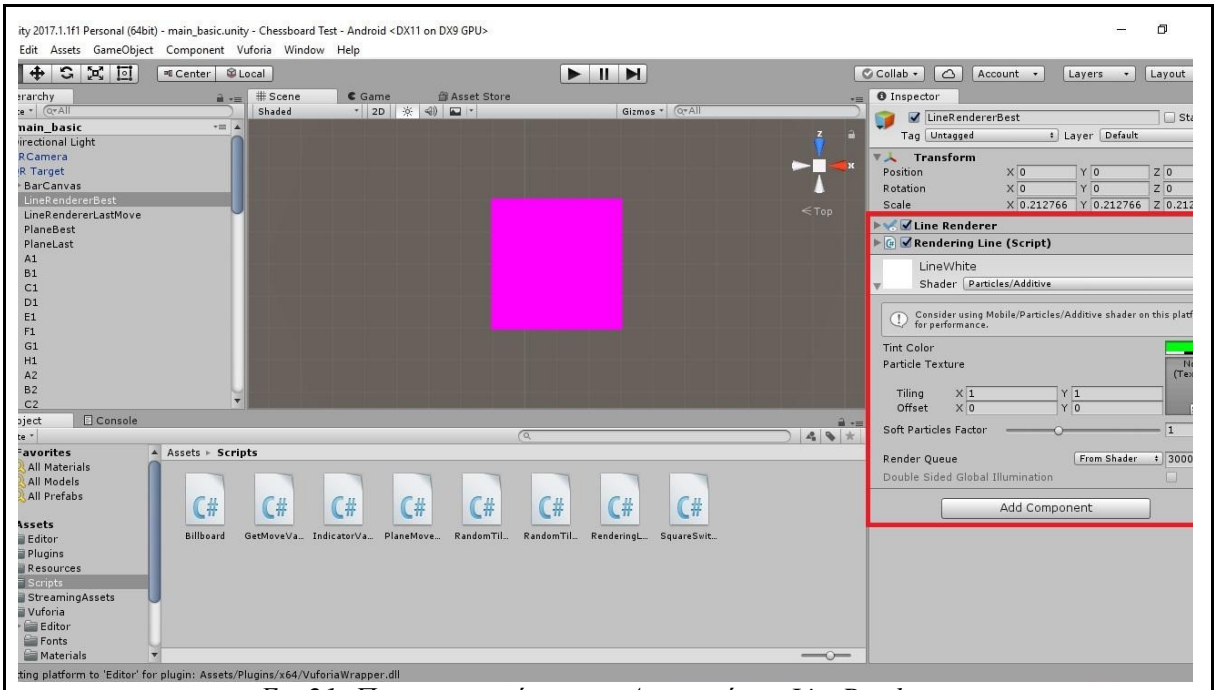
Τα Shaders δημιουργούν ουσιαστικά τα πολύγωνα από τα οποία αποτελούνται τα 3D αντικείμενα, καθώς και διάφορα φίλτρα εικόνας τα οποία εξυπηρετούν στην διαμόρφωση των μοντέλων αυτών.



Εικ.20: Σχήμα αναπαράστασης 3D Μοντέλου.

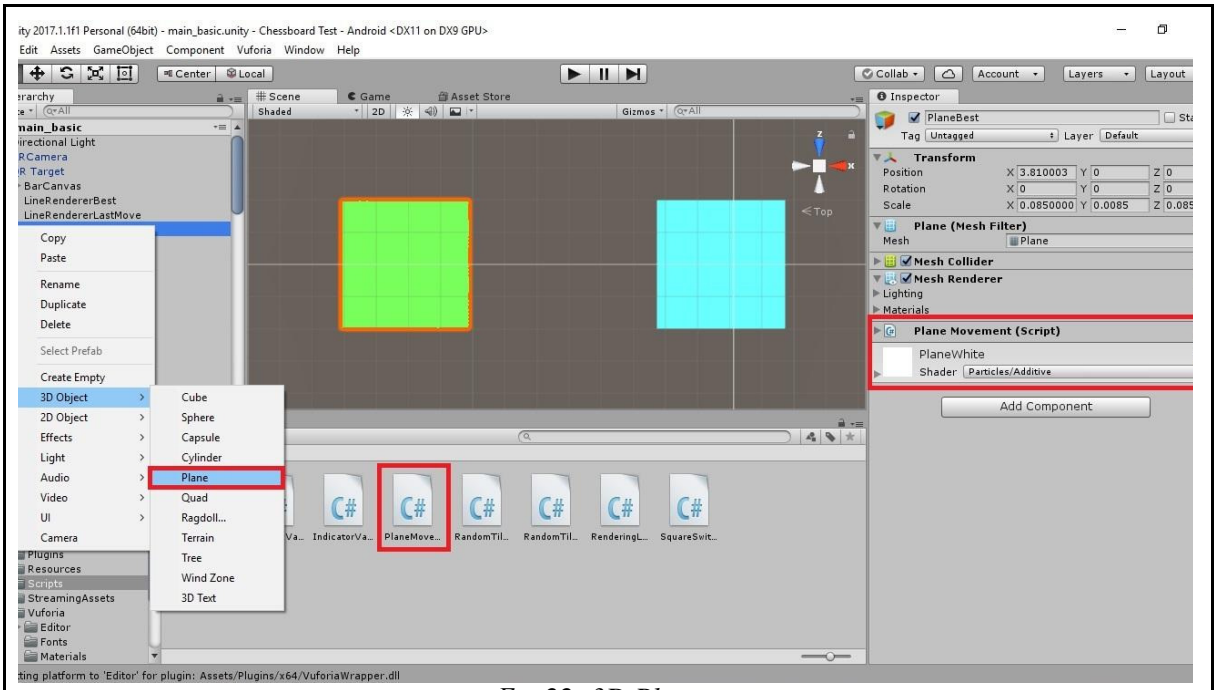
Ο λόγος τώρα που δημιουργούμε 2 LineRenderer Αντικείμενα είναι διότι θέλουμε να αναπαραστήσουμε 2 κινήσεις επάνω στη σκακιέρα. Μία βέλτιστη κίνηση για τον παίκτη του γύρου καθώς και την τελευταία κίνηση που έγινε.

Για καλύτερη αναπαράσταση των κινήσεων θα δημιουργήσουμε και στο κελί του προορισμού, ένα αχνό τετράγωνο που θα σημάνει το τέλος της κίνησης.



Εικ.21: Παραμετροποίηση των Αντικειμένων LineRenderer.

Δημιουργούμε τώρα δυο νέα 3D Plane Αντικείμενα στην Ιεραρχία καθώς και ένα νέο κομμάτι κώδικα C# στα Assets. Του δίνουμε το όνομα “PlaneMovement”. Έπειτα κάνουμε Drag & Drop το script μας στον Επιθεωρητή και δίνουμε στα Αντικείμενα αυτά Particle/Additive Shadders.



Εικ.22: 3D Planes.

Στη συνέχεια επεξεργαζόμαστε στο Visual Studio τα 2 scripts που φτιάξαμε.

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

    private LineRenderer lineRenderer;
    private float counter;
    private float dist;
    private int flag = 0;

    private GameObject o;
    private GameObject d;

    private Transform origin;
    private Transform destination;

    public string inputOrigin;
    public string inputDestination;

    public float lineDrawSpeed = 12f;

    void Start () {
        InvokeRepeating("UpdateLine", 3f, 3f);
    }

     void Update() {
        //Simple boolean activating script by button
        if (flag == 1) {
            if (counter < dist) {
                counter += .1f / lineDrawSpeed;
                float x = Mathf.Lerp(0, dist, counter);

                Vector3 pointA = origin.position;
                Vector3 pointB = destination.position;
                Vector3 pointAlongLine = x *
                Vector3.Normalize(pointB - pointA) + pointA;
                lineRenderer.SetPosition(1, pointAlongLine);
            }
        }
    }
    private void UpdateLine()
    {
```

```
        counter = 0;
    }

    public void Ignition()
    {
        Debug.Log(inputOrigin);
        Debug.Log(inputDestination);

        o = GameObject.Find(inputOrigin);
        d = GameObject.Find(nputDestination));

        origin = o.transform;
        destination = d.transform;

        lineRenderer = GetComponent<LineRenderer>();
        lineRenderer.SetPosition(0, origin.position);
        lineRenderer.SetWidth(.45f, .45f);
        dist = Vector3.Distance(origin.position,
                                destination.position);

        flag = 1;
        Debug.Log("Linerenderer is activated.");
    }
}
```

Το παραπάνω κομμάτι κώδικα αντιστοιχεί στο "RenderingLine" script. Αρχικά ο κώδικας **αναζητά** Αντικείμενα στη Σκηνή με βάση **το όνομά** τους και στη συνέχεια **τροποποιεί** το Mesh του Αντικειμένου LinerRender για να δημιουργήσει μία γραμμή από το ένα Αντικείμενο στο άλλο. Αφού ολοκληρώσουμε το script, στον Επιθεωρητή θα εμφανίζεται η επιλογή να τοποθετήσουμε ποιά αντικείμενα θέλουμε να αναζητήσουμε. Σε μεταγενέστερο στάδιο η χειροκίνητη αυτή είσοδος θα αντικατασταθεί με τις πληροφορίες που θα αποκομίσουμε από το HTML Scraping.

Εικ.23: Properties του Αντικειμένου LineRenderer.

Μπορούμε έτσι στον Επιθεωρητή να τοποθετούμε χειροκίνητα τα ονόματα από τα Αντικείμενα τα οποία αναπαριστούν τα κελιά στη σκακιέρα.

Ας δούμε τώρα το "PlaneMovement":

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

public class PlaneMovement : MonoBehaviour {
    private GameObject d;

    public string inputDestination;
    private Transform destination;

    public void Ignition()
    {

        d = GameObject.Find(inputDestination);
        destination = d.transform;

        transform.position = d.transform.position;
        transform.parent = d.transform;
        Debug.Log("Plane Moved.");
    }
}
```

Ομοίως με τον προηγούμενο κώδικα, μπορούμε να τοποθετήσουμε στον Επιθεωρητή το κελί που θέλουμε να δημιουργήσουμε το επίπεδο αναζητώντας το αντικείμενο με το αντίστοιχο όνομα.

Εικ.24: Το PlaneMovement Script στον Επιθεωρητή.

Και τα δύο κομμάτια κώδικά μας έχουν από μία κοινή συνάρτηση, την Ignition(). Η συνάρτηση αυτή είναι η συνάρτηση που ενεργοποιεί ουσιαστικά τον κώδικα. Για να λειτουργήσουν οι κώδικες πρέπει να την καλέσουμε μέσω του δεύτερου κουμπιού ("SHOW") που δημιουργήσαμε προηγουμένως (σελ. 32). Όπως λοιπόν και με το SquareSwitch script (σελ. 41) ενσωματώνουμε και τις Ignition() συναρτήσεις.

Δείκτης Απόδοσης Των Παικτών

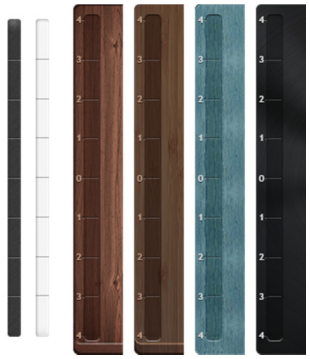
Στο σκάκι, έχει αναπτυχθεί ένας δείκτης απόδοσης των παικτών ο οποίος εξαρτάται από τις κινήσεις που επιλέγουν οι παίκτες, από τα πιόνια που ελέγχουν ακόμα και τις θέσεις όπου βρίσκονται τα πιόνια τους. Ο δείκτης αυτός παίρνει τιμές από το -4 έως το +4. Όσο πιο κοντά

στο +4 βρίσκεται ο δείκτης, υποδεικνύει ότι έχει προβάδισμα στην παρτίδα ο λευκός παίκτης και αντίστοιχα όσο περισσότερο τείνει στο -4, το προβάδισμα πηγαίνει στον μαύρο παίκτη.

Πολλές σκακιστικές ιστοσελίδες αναπαριστούν αυτόν τον συντελεστή με ένα γραφικό μίας μπάρας η οποία αυξομειώνεται με κάθε κίνηση και έχει ως τιμές το εύρος [-4,+4].

Εικ.25:
Το γραφικό που χρησιμοποιείται από το chess24.com για την αναπαράσταση του δείκτη απόδοσης.

Με ένα μικρό ψάξιμο στο HTML του chess24 μπορούμε να βρούμε τα γραφικά που χρησιμοποιούνται για τον δείκτη. Από την εικόνα αυτή θα χρειαστούμε 2 γραφικά:

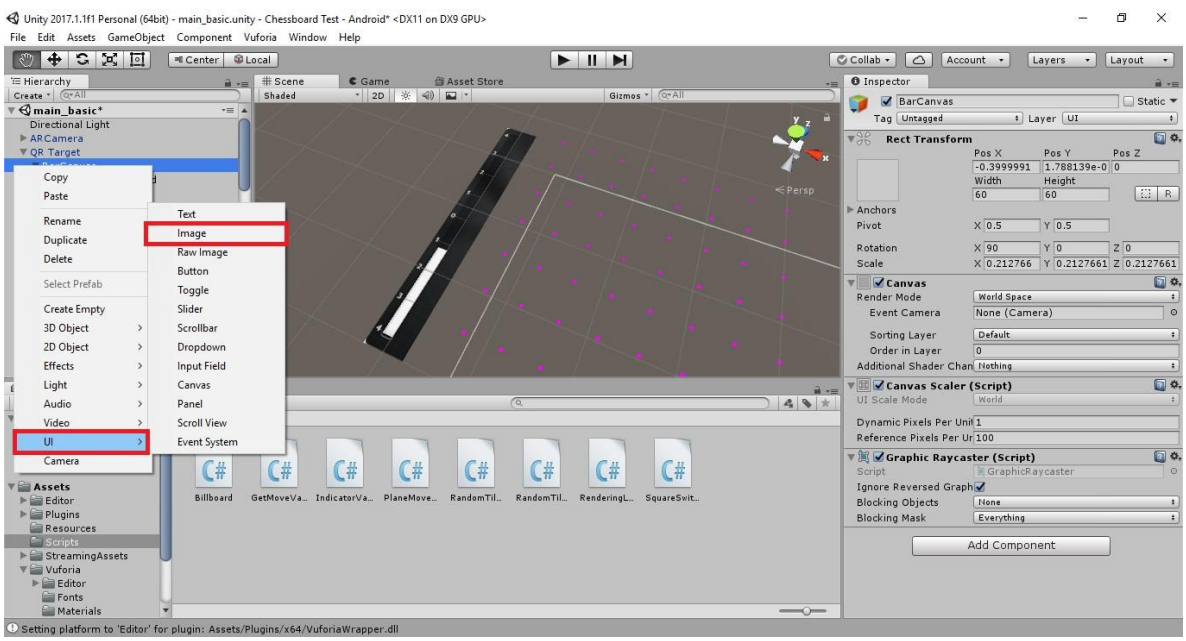


Εικ.26: Τα γραφικά της μπάρας.

ένα για το περίγραμμα της μπάρας και ένα για το γέμισμά της. Δημιουργούμε λοιπόν δύο ξεχωριστά αρχεία εικόνων, ένα για το περίγραμμα και ένα για το γέμισμα αντίστοιχα και στη συνέχεια τα εισάγουμε στο Unity "σέρνοντάς" τα στα Assets.

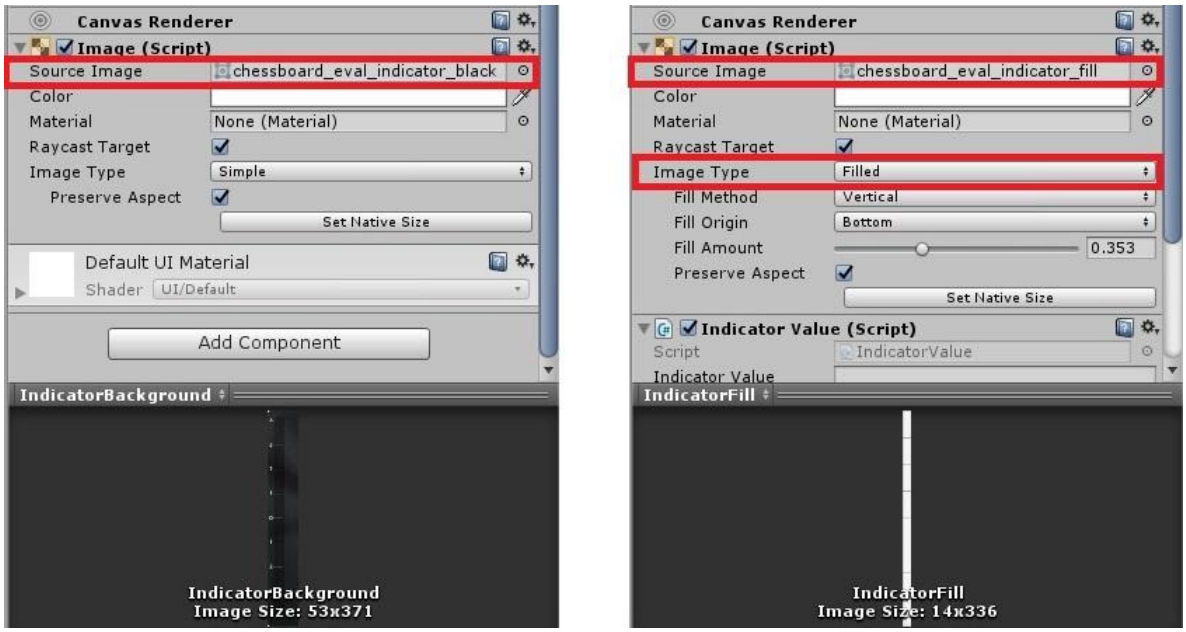
Στη συνέχεια θα προσπαθήσουμε να επαυξήσουμε την μπάρα με τον δείκτη απόδοσης στην εφαρμογή μας.

Για να αναπαραστήσουμε τη μπάρα με τον δείκτη απόδοσης, θα δημιουργήσουμε στη Σκηνή δυο UI Στοιχεία Εικόνας, μέσο της Ιεραρχίας. Το Unity έτσι θα μας ετοιμάσει έναν νέο Καμβά και θα δημιουργήσουμε μέσα του τις δύο εικόνες.



Εικ.27: Δημιουργόντας τα Στοιχεία UI στην Ιεραρχία.

Στη συνέχεια επιλέγουμε τα γραφικά που εισάγαμε στα Assets ως γραφικά των Στοιχείων μας και τοποθετούμε τον καμβά σε σημείο τέτοιο ώστε όταν τρέξει η εφαρμογή μας να εμφανιστεί σε εμφανές σημείο χωρίς να εμποδίζει στην επαύξηση, όπως στην πιο πάνω εικόνα τον τοποθετήσαμε αριστερά της σκακιέρας (όπως και στο chess24).



Εικ.28: Οι ρυθμίσεις των εικόνων στον Επιθεωρητή.

Επιλέγουμε να κάνουμε "παιδί" την λευκή μπάρα στο περίγραμμα και όπως φαίνεται στην εικόνα επιλέξαμε το Image Type να είναι *Filled*. Έτσι μπορούμε να αυξομειώνουμε το μέγεθος της εικόνας ώστε να φαίνεται ότι η μπάρα γεμίζει και αδειάζει. Το ποσοστό "γεμίσματος" υποδεικνύεται από τη μεταβλητή Fill Amount, η οποία παίρνει τιμές από 0-1.

Το μόνο που πρέπει να κάνουμε τώρα είναι να φτιάξουμε έναν αλγόριθμο ο οποίος θα

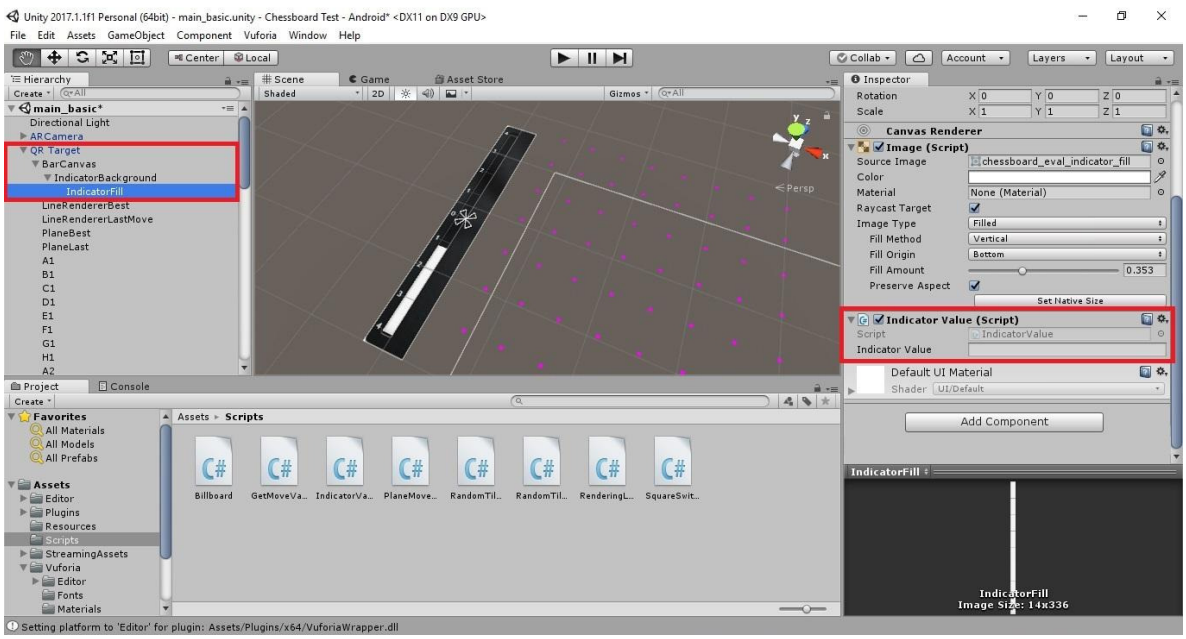
μετατρέπει το εύρος τιμών [-4,4] σε [0,1], έτσι ώστε να φαίνεται ότι η μπάρα γεμίζει. Αν και ακούγεται πολύπλοκο, δεν είναι ιδιαίτερα να υλοποιηθεί ένας τέτοιος αλγόριθμος. Φτιάχνουμε ένα νέο κομμάτι κώδικα το οποίο το ονομάζουμε IndicatorValue και το "σέρνουμε" στη λευκή μπάρα. Ο κώδικας που εισάγουμε είναι ο εξής:

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

public class IndicatorValue : MonoBehaviour {

    Image IndicatorBar;
    float maxValue = 4f;
    float minValue = -4f;
    public static float indicator;
    public string indicatorValue;
    float v;
    float value;

    void Start () {
        IndicatorBar = GetComponent<Image>();
        indicator = 0.4f;
    }

    private float Map(float value, float inMin, float inMax,
                      float outMin, float outMax)
    {

        return (value - inMax)*(outMax - outMin) /
                               (inMax - inMin) + outMin;
    }

    public void Ignition()
    {
        v = float.Parse(indicatorValue);
        value = v * (-1);

        IndicatorBar.fillAmount = Mathf.Abs(Map(value, -4f,
        4f, 0f, 1f));
    }
}
```

Η συνάρτηση `float Map()` παίρνει ως παραμέτρους την τιμή που θέλουμε να μετατρέψουμε σε ποσοστό, την μέγιστη τιμή και την ελάχιστη τιμή που μπορεί να πάρει (`inMax, inMin`), όπου στην προκειμένη περίπτωση είναι -4 και 4 και την μέγιστη και ελάχιστη τιμή που πρέπει να δώσει ο αλγόριθμος (`outMin, outMax`), για εμάς 0 και 1, εφόσον αυτές τις τιμές παίρνει η μεταβλητή `fillAmount`.

Παρατήρηση: Τοποθετούμε και εδώ μία συνάρτηση `Ignition()` ώστε να την

καλέσουμε από το κουμπί και να θέσει σε λειτουργία τον αλγόριθμο. Οπότε δεν πρέπει να ξεχάσουμε να κάνουμε τη σύνδεση με το κουμπί.



Εικ.29: Οι ρυθμίσεις του κώδικα IndicatorValue που εμφανίζονται στον Επιθεωρητή.

Μέσω του επιθεωρητή μπορούμε να αναφέρουμε το ποσοστό που θέλουμε να αναπαραστήσουμε στην μπάρα, με εύρος [-4,4]. Τέλος για να επαυξηθεί η μπάρα πρέπει να τοποθετήσουμε στην Ιεραρχία τον Καμβά που εμπεριέχει τις εικόνες, μέσα στο QR Target, όπου βρίσκονται όλα τα αντικείμενα που θέλουμε να επαυξηθούν.

Στον κώδικα του αλγορίθμου υπάρχει ένα παράδοξο. Ο αλγόριθμος λειτουργεί για θετικές τιμές. Εμείς αναπαριστούμε ποιος παίκτης έχει το προβάδισμα μέσω του πρόσημου του δείκτη, αρνητικό (-) για τον μαύρο παίκτη, θετικό (+) για τον λευκό. Παρ' όλα αυτά, αν αντιστρέψουμε τα πρόσημα στην είσοδο του αλγορίθμου για τον κάθε παίκτη, τότε ο αλγόριθμος λειτουργεί κανονικά.

Αυτή είναι μία σημαντική παρατήρηση για το πως θα πάρουμε τις πληροφορίες μέσω Scrapping. Ο δείκτης του αλγορίθμου θα πρέπει να έρχεται με αντίθετο πρόσημο ή σε μεταγενέστερη έκδοση της εφαρμογής να γίνεται η αλλαγή του πρόσημου μέσα στον αλγόριθμο.

Επιπλέον Λειτουργίες: Virtual Buttons

Στο σημείο αυτό έχουμε ολοκληρώσει την εφαρμογή μας όσο αναφορά την επαύξηση των πληροφοριών. Έχοντας ολοκληρωμένες τις λειτουργίες επαύξησης, το μόνο που μένει είναι μέσο Scrapping να παρέχουμε στην εφαρμογή μας ό,τι πληροφορίες θέλουμε. Από εκεί και και πέρα οι διαδικασίες για την αναπαράσταση των πληροφοριών αυτών είναι η ίδια.

Παρ' όλα αυτά αξίζει να σημειωθεί μία περαιτέρω δυνατότητα που μπορεί να μας παρέχει η βιβλιοθήκη Vuforia, η οποία στο μέλλον μπορεί να φανεί χρήσιμη και να χρησιμοποιηθεί για την βελτίωση της εφαρμογής.

Η λειτουργία που θα αναφερθούμε στην ενότητα αυτή είναι τα **Εικονικά Κουμπιά (Virtual Buttons)**. Τα Virtual Buttons εξυπηρετούν στη διεπαφή με τον χρήστη. Μπορούμε να ορίσουμε μία εικόνα στόχο ως Εικονικό Κουμπί και να δώσουμε έτσι σήμα στην εφαρμογή να εκτελέσει μία **διαδικασία**.

Εικ.30: Τυπική Λειτουργία των Εικονικών Κουμπιών.

Αφού Δημιουργήσουμε μία τυπική Σκηνή στο Unity με όλα τα απαραίτητα για επαύξηση Αντικείμενα (ARCamera, ImageTarget), μέσο των *Assets/Vuforia/Prefabs* τοποθετούμε επιπλέον στη Σκηνή μας ένα Virtual Button και το ορίζουμε και αυτό ως παιδί της Εικόνας.

Τέλος Δημιουργούμε ένα κομμάτι κώδικα σε C# και το τοποθετούμε μέσα στο Εικονικό Κουμπί. Εδώ θα υλοποιήσουμε την διαδικασία που θα εκτελείτε μόλις ο χρήστης περάσει το χέρι του πάνω από το Virtual Button.

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

public class vButtonBehaviour : MonoBehaviour,
                                IVirtualButtonEventHandler {
   public GameObject vbObject;

   void Start(){
```

```
        vbObject = GameObject.Find("VirtualButton");
        vbObject.GetComponent<VirtualButtonBehaviour>().
        RegisterEventHandler(this);
    }

    public void onButtonPressed(VirtualButtonBehaviour vb){
        // Line Of Code To be Excecuted When Button is pressed.
    }

    public void onButtonReleased(VirtualButtonBehaviour vb){
        // Line Of Code To be Excecuted When Button is
        // released.
    }
```

Ο πιο πάνω κώδικας αποτελεί την "ραχοκοκαλιά" για την υλοποίηση ενός Virtual Button.

Αρχικά με βρίσκουμε μέσο της συνάρτησης `Start()` το Εικονικό Κουμπί στη Σκηνή και το ενεργοποιούμε με την εντολή `GetComponent<VirtualButtonBehaviour>()` `.RegisterEventHandler(this)`.

Τέλος μέσα στις συναρτήσεις `onButtonPressed()` και `onButtonReleased()` γράφουμε τον κώδικα όπου θέλουμε να εκτελείται όταν πατάμε και όταν αφήνουμε το κουμπί αντίστοιχα.

Future Work

Όσο αναφορά την μελλοντική εργασία για την πτυχιακή, ως σκοπός στο μέλλον, είναι να επιτυγχάνεται αναγνώριση των πιονιών και της σκακέρας χωρίς την χρήση Εικόνας- Στόχου. Αυτό πιστεύουμε ότι μπορεί να επιτευχθεί χρησιμοποιώντας 3D αναγνώριση για κάθε πιόνι ώστε να εμφανίζεται η κίνηση του, απαλείφοντας έτσι την χρήση του QR. Επίσης, με τον τρόπο αυτό η εφαρμογή μπορεί να χρησιμοποιηθεί όχι μόνο σε οργανωμένα παιχνίδια σκακιού, όπου μπορεί να γίνει προεργασία για να στηθούν οι παρτίδες και να λειτουργεί η εφαρμογή, αλλά και σε ερασιτεχνικά παιχνίδια όπως π.χ. σε έναν μικρό σκακιστικό σύλλογο ή ακόμα και σε ένα καφενείο.

Επίσης, εκτός από την 3D αναγνώριση, μπορεί να χρησιμοποιηθεί μία σταθερή κάμερα από την αρχή της παρτίδας η οποία θα αναγνωρίζει την σκακιέρα και θα εκτελεί ουσιαστικά, μία μικρή επεξεργασία βίντεο, σε πραγματικό χρόνο, για να απεικονίσει τις πληροφορίες της παρτίδας. Αυτό βέβαια αποτελεί μία πιο γενικευμένη άποψη σχετικά με την Επαύξηση της Πραγματικότητας. Υπάρχουν τέτοιες βιβλιοθήκες, οι οποίες είναι ανοιχτού κώδικα και εκτελούν αναγνώριση σκακιέρας μέσο βίντεο. Για περισσότερες πληροφορίες μπορεί να ανατρέξει κανείς στο Παράρτημα με τις πηγές που χρησιμοποιήθηκαν για την εργασία αυτή και συμπεριλαμβάνονται στον οπτικό δίσκο (cd) της πτυχιακής.

Επίλογος

Η πτυχιακή αυτή εργασία παρέχει σε γνώστες και μη τις πιο βασικές γνώσεις για να μπορέσει κανείς να αναπτύξει την δικιά του Εφαρμογή Επαυξημένης πραγματικότητας με την χρήση της Vuforia. Φυσικά οι επιλογές και οι περιπτώσεις χρήσης είναι αμέτρητες στον κλάδο της Επαυξημένης Πραγματικότητας επομένως ο καθένας μπορεί να χρησιμοποιήσει την βιβλιοθήκη και την ίδια την τεχνολογία της επαύξησης της πραγματικότητας με τον δικό του μοναδικό τρόπο, δημιουργώντας έτσι καινοτομίες.

Όπως αναλύθηκε και στην πρώτη ενότητα, η Επαυξημένη Πραγματικότητα είναι μια ραγδαία αναπτυσσόμενη τεχνολογία με πολλά πεδία εφαρμογής. Μπορεί να χρησιμοποιηθεί όχι μόνο από τον κλάδο της Πληροφορικής αλλά και από πολλούς άλλους όπως την Ιατρική, την Εκπαίδευση, την Διασκέδαση ακόμα και στις Τέχνες. Στο σημείο αυτό θα ήθελα να αναφέρω πως και για αυτόν τον λόγο, ότι δηλαδή είναι μία πολύ πολύπλευρη τεχνολογία, την επέλεξα ως θέμα για την πτυχιακή μου. Πιστεύω πως μπορεί να χρησιμοποιηθεί για να εκφράσει μία εμπειρία ή γνώση με δημιουργικό τρόπο ακόμα και υπό τη μορφή τέχνης.

Βιβλιογραφία

Infographic: The History of Augmented Reality, (2016). Ανακτήθηκε 3 Δεκεμβρίου, 2017. http://www.augment.com/blog/infographic-lengthy-history-augmented-reality/

The Ultimate Guide To Augmented Reality (AR) Technology, (2016). Ανακτήθηκε 13 Μαΐου, 2017. http://www.realitytechnologies.com/augmented-reality

Jaewoon, L. (2015). *Real-Time Projection-Based Augmented Reality System for Dynamic Objects in the Performing Arts*. Symetry - Open Access Journal: Symmetry. doi:10.3390/sym7010182

Vuforia: Developer Library, (2011). Ανακτήθηκε 10 Μαΐου, 2017. https://library.vuforia.com/

Sutherland, I. (1963). *Sketchpad, A Man - Machine Graphical Communication System.* Massachusett: Massachusett's Institute Of Technology