**ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ**

**ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ**

**ΤΜΗΜΑ ΑΓΡΟΝΟΜΩΝ ΚΑΙ ΤΟΠΟΓΡΑΦΩΝ ΜΗΧΑΝΙΚΩΝ**

**ΤΟΜΕΑΣ ΚΤΗΜΑΤΟΛΟΓΙΟΥ, ΦΩΤΟΓΡΑΜΜΕΤΡΙΑΣ ΚΑΙ ΧΑΡΤΟΓΡΑΦΙΑΣ**

## ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

# ΑΝΑΠΤΥΞΗ ΠΑΚΕΤΟΥ ΟΠΤΙΚΟΠΟΙΗΣΗΣ ΧΩΡΙΚΩΝ ΔΕΔΟΜΕΝΩΝ

**Εμμανουήλ Κ. Χρυσανίδης**

**Επιβλέπων:**
**Πέτρος Πατιάς, Καθηγητής, ΤΑΤΜ, ΑΠΘ**

**Τριμελής Εξεταστική Επιτροπή:**
**Πέτρος Πατιάς, Καθηγητής, ΤΑΤΜ, ΑΠΘ**
**Βασίλειος Τσιούκας, Καθηγητής, ΤΑΤΜ, ΑΠΘ**
**Δημήτριος Καϊμάρης, Αν. Καθηγητής, ΤΜΧΑ, ΑΠΘ**

**Θεσσαλονίκη, Μάρτιος 2021**

**ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ**
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
**ΤΜΗΜΑ ΑΓΡΟΝΟΜΩΝ ΚΑΙ ΤΟΠΟΓΡΑΦΩΝ ΜΗΧΑΝΙΚΩΝ**
ΤΟΜΕΑΣ ΚΤΗΜΑΤΟΛΟΓΙΟΥ, ΦΩΤΟΓΡΑΜΜΕΤΡΙΑΣ ΚΑΙ ΧΑΡΤΟΓΡΑΦΙΑΣ

| | | |
|---|---|---|
| **ΙΔΡΥΜΑ** | Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης | |
| **ΣΧΟΛΗ** | Πολυτεχνική Σχολή | |
| **ΤΜΗΜΑ** | Αγρονόμων και Τοπογράφων Μηχανικών | |
| **ΕΡΓΑΣΙΑ** | Διπλωματική Εργασία | |
| **ΤΙΤΛΟΣ ΕΡΓΑΣΙΑΣ** | Ανάπτυξη Πακέτου Οπτικοποίησης Χωρικών Δεδομένων | |
| **ΕΚΠΟΝΗΣΗ** | Εμμανουήλ Κ. Χρυσανίδης | |
| **ΗΜΕΡΟΜΗΝΙΑ** | 18/03/2021 | |
| **ΕΠΙΒΛΕΠΩΝ** | Π. Πατιάς, Καθηγητής ΤΑΤΜ | |
| **ΤΡΙΜΕΛΗΣ ΕΠΙΤΡΟΠΗ** | Π. Πατιάς, Καθηγητής ΤΑΤΜ,<br>Βασίλειος Τσιούκας, Καθηγητής, ΤΑΤΜ<br>Δημήτριος Καϊμάρης, Αν. Καθηγητής, ΤΜΧΑ | |
| **ΟΝΟΜΑ ΗΛΕΚΡΟΝΙΚΟΥ ΑΡΧΕΙΟΥ** | | |
| **ΣΥΝΝΗΜΕΝΑ ΑΡΧΕΙΑ** | | |
| **ΠΕΡΙΓΡΑΦΗ ΠΑΡΑΔΟΤΕΟΥ** | **Αναλογική μορφή** | **Ψηφιακή μορφή** |
| | Το παρόν τεύχος αποτελεί τη Διπλωματική εργασία, που περιλαμβάνει την αναλυτική περιγραφή της μεθοδολογίας, την εφαρμογή, τα αποτελέσματα και τα συμπεράσματα της εργασίας. | Στο CD (ή DVD) που συνοδεύει το τεύχος περιλαμβάνονται σε ψηφιακή μορφή: τα κείμενα της εργασίας σε μορφή αρχείων doc και pdf, οι μετρήσεις πεδίου σε μορφή αρχείου .xls ή .txt, οι ορθοφωτοχάρτες σε μορφή αρχείου .tif, κτλ. |

# Δήλωση Συγγραφικής Ιδιότητας

Εγώ, ο Εμμανουήλ Χρυσανίδης, δηλώνω ότι αυτή η πτυχιακή εργασία με τίτλο, Ανάπτυξη Πακέτου Οπτικοποίησης Χωρικών Δεδομένων, και η δουλειά που παρουσιάζεται σε αυτή είναι δικά μου. Επιβεβαιώνω ότι:

- Αυτή η δουλειά πραγματοποιήθηκε ολοκληρωτικά ή κυρίως κατά την υποψηφιότητά μου για τίτλο προπτυχιακών σπουδών σε αυτό το πανεπιστήμιο.
- Όπου οποιοδήποτε μέρος αυτής της πτυχιακής εργασίας έχει προηγουμένως κατατεθεί για την απόκτηση πτυχίου ή άλλου τίτλου σε αυτό ή άλλο πανεπιστήμιο, αυτό διατυπώνεται ξεκάθαρα.
- Όπου έχω συμβουλευτεί την δημοσιευμένη δουλειά τρίτων, αυτό αποδίδεται ορθώς.
- Όπου έχω παραθέσει από δουλειά τρίτων, η πηγή δίνεται πάντα. Με εξαίρεση αυτές τις παραθέσεις, αυτή η πτυχιακή εργασία είναι εξ ολοκλήρου προσωπική μου δουλειά.
- Έχω παραθέσει όλες τις κύριες πηγές βοήθειας.
- Όπου αυτή η πτυχιακή εργασία είναι βασισμένη σε συνεργατική δουλειά δική μου και τρίτων, έχω καταστήσει ξεκάθαρο ποια κομμάτια έχουν πραγματοποιηθεί από άλλους και πώς συνέβαλα εγώ.

Υπογραφή: _____

Ημερομηνία: _____

# Περίληψη

Η αύξηση της υπολογιστικής ισχύος φορητών ηλεκτρονικών συσκευών (κινητών τηλεφώνων και τάμπλετ) τα τελευταία χρόνια έδωσε τη δυνατότητα δημιουργίας εφαρμογών με εξαιρετικά ρεαλιστικά γραφικά. Μία κατηγορία εφαρμογών που απαιτούν ρεαλιστικά γραφικά, είναι οι εφαρμογές Εικονικής, Επαυξημένης ή ακόμη και Μικτής Πραγματικότητας. Οι εφαρμογές αυτές έχουν εφαρμογή σε μία πληθώρα τομέων όπως, του Πολιτισμού, των Τεχνών, της Αγοράς Ακινήτων και του Τουρισμού. Ο σκοπός της παρούσας εργασίας είναι η δημιουργία δύο εφαρμογών Επαυξημένης Πραγματικότητας, μέσω των οποίων μπορεί κάποιος να περιηγηθεί σε τμήματα αρχαιολογικών χώρων στη Λάρισα και στη Τούμπα αντίστοιχα. Συγκεκριμένα δημιουργούνται δύο Πόρτες Επαυξημένης Πραγματικότητας οι οποίες οδηγούν τον χρήστη της εφαρμογής στο εσωτερικό των παραπάνω αρχαιολογικών χώρων. Η διαδικασία δημιουργίας των εφαρμογών ακολουθεί τα εξής τέσσερα στάδια. Πρώτο στάδιο είναι η αποτύπωση των αρχαιολογικών χώρων με σκοπό την δημιουργία δύο Νεφών Σημείων. Νέφος Σημείων είναι ένα σύνολο σημείων με συντεταγμένες $x, y, z$, το οποίο περιέχει χρωματική πληροφορία και απεικονίζει το τελικό αποτέλεσμα της τρισδιάστατης σάρωσης του επιθυμητού τρισδιάστατου μοντέλου. Επόμενο στάδιο είναι η δημιουργία των Τρισδιάστατων Μοντέλων των δύο εφαρμογών. Αυτά δημιουργούνται μετά από την κατάλληλη επεξεργασία των Νεφών Σημείων που λήφθηκαν στο πρώτο στάδιο, με χρήση εναέριας φωτογραμμετρίας αλγόριθμος SfM και σαρωτή λέιζερ. Τρίτο στάδιο αποτελεί η δημιουργία του εικονικού περιβάλλοντος με τη χρήση του λογισμικού Unity. Στο στάδιο αυτό δημιουργούνται ουσιαστικά όλα τα αντικείμενα τα οποία θα υπάρχουν στον εικονικό χώρο της εφαρμογής. Τέταρτο και τελευταίο στάδιο αποτελεί η εισαγωγή των Τρισδιάστατων Μοντέλων των αρχαιολογικών χώρων ως αντικείμενα μέσα στο εικονικό περβάλλον. Τέλος γίνεται μία ανασκόπηση των μεθόδων που χρησιμοποιήθηκαν σε κάθε μία από τις δύο εφαρμογές.

# Abstract

The recent increase of processing power in mobile electronic devices (such as smartphones and tablets) has led to the development of mobile applications with extremely realistic graphics. Applications that take advantage of this progress in electronics are Virtual Reality, Augmented Reality or even Mixed Reality applications. Those applications are applied to a wide variety of sectors, including Culture, Arts, Real Estate and Tourism. The aim of this thesis is the development of two Augmented Reality (AR) applications, which give the user the opportunity to virtually tour two archeological sites, one in Toumba Thessaloniki and the other one in Larissa. This tour is implemented through two AR Portals which lead into the above mentioned archeological sites. The process of the application development consists of four stages. The first stage involves the mapping of the two sites, in order to obtain two Point Clouds, one for each site. A Point Cloud is a set of points, which represent a three dimension (3D) object. Each point has its own set of $x, y, z$ coordinates and color information. The next stage consists of the creation of the 3D Models of the archeological sites. These are created by processing the Point Clouds obtained in the first stage, using aerial photogrammetry (SfM Algorithm) and laser scanner. The virtual environment of the application is created in the third stage, using the Unity Game Engine. This stage involves the creation of all those objects which will exist in the virtual environment of the application. In the fourth and last stage, the 3D Models of the archeological sites are imported in the virtual environment of the application. Lastly there is a review of the methods used in each application.

# Ευχαριστίες

Κατά τη διάρκεια υλοποιήσης της προσπάθειας αυτής έλαβα στήριξη και βοήθεια από διάφορα πρόσωπα, τα οποία θα ήθελα να ευχαριστήσω και να δηλώσω την εκτίμηση μου στον καθένα ξεχωριστά. Αρχικά θα ήθελα να εκφρασω τις ευχαριστίες μου στον επιβλέποντα καθηγητή κ. Πέτρο Πατιά με τη συμβολή του οποίου κατάφερα να υλοποιήσω την παρούσα διπλωματική εργασία. Επίσης θα ήθελα να ευχαριστήσω τον Αν. καθηγητή κ. Δημήτριο Καϊμάρη και τον καθηγητή κ. Βασίλειο Τσιούκα για τις εύστοχες παρατηρήσεις και την καθοδήγηση που μου παρείχανε. Επίσης θα ήθελα να ευχαριστήσω την οικογένειά μου και ιδίως τον αδερφό μου Γιώργο για τη συνεχή στήριξη, καθώς και τους φίλους μου.

# Περιεχόμενα

# Κατάλογος Σχημάτων

# Κεφάλαιο 1

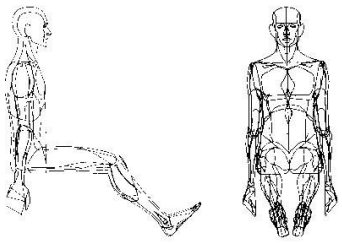
# Εισαγωγή

## 1.1 Σκοπός της Διπλωματικής Εργασίας

Ο σκοπός της παρούσας διπλωματικής εργασίας, είναι η δημιουργία πακέτου οπτικοποιήσης χωρικών δεδομένων. Πιο συγκεριμένα η δημιουργία μιας AR Portal, μέσω της οποίας ο χρήστης περιηγείται σε αρχαιολογικούς χώρους και χώρους ανασκαφών. Η αποτύπωση των επιθυμητών αυτών χώρων( Αρχαίο Θέατρο Λάρισας και χώρος ανασκαφών Τούμπας) έγινε με τη χρήση φωτογραμμετρικών μεθόδων και σαρωτή λέιζερ. Στη συνέχεια, μέσω κατάλληλων λογισμικών, όπως Agisoft Metashape, Scene, Geomagic Studio, Polywork Inspector, δημιουργήθηκαν τα κατάλληλα τρισδιάστατα μοντέλα. Τα μοντέλα αυτά αφού τροποποιήθηκαν κατάλληλα μέσω των λογισμικών CloudCompare και Meshlab εισήχθησαν στο Unity, όπου και δημιουργήθηκαν οι τελικές εφαρμογές για Smartphone

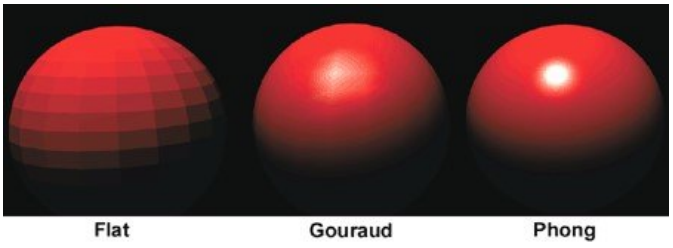
## 1.2 Ιστορική Αναδρομή

Ένας από τους πρωτοπόρους της σχεδίασης τρισδιάστατων σχεδίων(3D) σε υπολογιστή ήταν ο William Fetter το 1960, ο οποίος εργαζόταν για την εταιρία Boeing και σχεδίαζε μοντέλα σε υπολογιστές. Ένα από τα σχέδια πάνω στα οποία δούλευε ήταν «ο Άνθρωπος Boeing», δηλαδή η τρισδιάστατη αναπαράσταση του ανθρώπινου σώματος, στο οποίο ο Fetter εισήγαγε για πρώτη φορά τον όρο «γραφικά του υπολογιστή» [1].



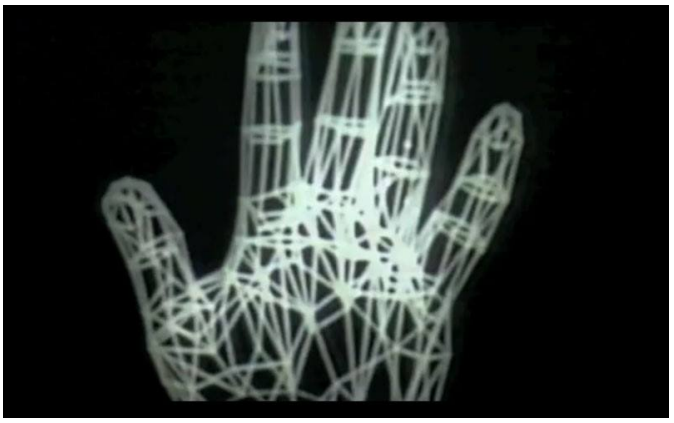
Σχημα 1.1: Τρισδιάστατη αναπαράσταση του ανθρώπινου σώματος σύμφωνα με τον William Fetter [1]

Αργότερα το 1971, ο Henri Gouraud [8] εφάρμοσε μια νέα τεχνική σκίασης, η οποία πήρε και το όνομά του, «Gouraud shading». Η μέθοδος αυτή χρησιμοποιείται για την προσομοίωση των διαφορετικών επιδράσεων του φωτός και του χρώματος στην επιφάνεια ενός αντικειμένου. Βασική αρχή που διέπει την τεχνική αυτή είναι ο υπολογισμός των κανονικών επιφανειών στις κορυφές των πολυγώνων, σε ένα τρισδιάστατο (3D) μοντέλο υπολογιστή. Συγκριτικά με τη μεταγενέστερη μέθοδο σκίασης του Phong,«Phong Shading» η οποία δημιουργήθηκε το 1973, απαιτεί λιγότερη επεξεργασία, αλλά δεν υπολογίζει με ακρίβεια όλα τα αποτελέσματα του φωτισμού. Οι δύο αυτές μέθοδοι ωστόσο, είναι πολύ ανώτερες από την επίπεδη σκίαση, η οποία ήταν προγενέστερη αυτών.




Σχημα 1.2: Οι 3 πιο συνηθισμένοι τύποι σκίασης σε τρισδιάστατα αντικείμενα [2]

Το 1972, οι Edwin Catmull [9] και Fred Parke [10] ανακάλυψαν τη μέθοδο χαρτογράφησης της υφής. Κατασκεύασαν αρχικά ένα τρισδιάστατο μοντέλο του αριστερού χεριού του Edwin Catmull αποτελούμενο από 350 τρίγωνα και πολύγωνα. Στη συνέχεια, το ψηφιοποιήσαν σε ένα τρισδιάστατο πρόγραμμα που δημιούργησε ο Edwin Catmull.



Σχημα 1.3: Τρισδιάστατη απεικόνιση του αριστερού χεριού [3]

Το 1978 ο James Blinn [11], ο οποίος ήταν επιστήμονας υπολογιστών στη NASA, δημιούργησε τη μέθοδο της ανάγλυφης απεικόνισης. Η μέθοδος αυτή παρουσιάζει με άσπρο και μαύρο χρώμα την υφή που χρησιμοποιείται, με αποτέλεσμα να δίνει μια πιο ρεαλιστική εικόνα στο μοντέλο.

Σχημα 1.4: Το πραγματικό αντικείμενο [4]

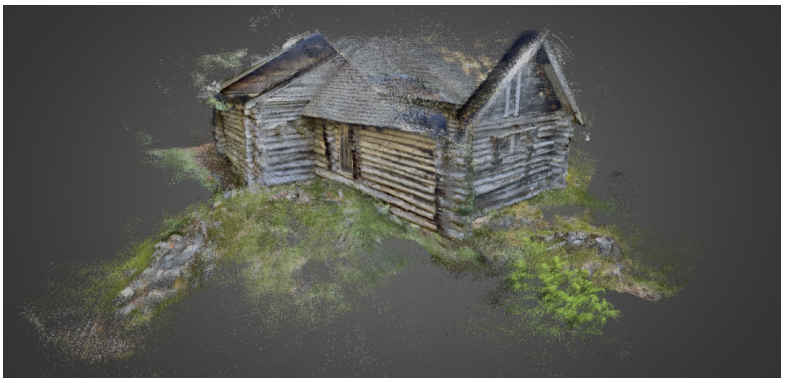
Σχημα 1.5: Το ψηφιακό αντικείμενο με τη μέθοδο της ανάγλυφης απεικόνισης [4]

Η χρήση των γραφικών λειτουργικών συστημάτων άρχισε να γίνεται γνωστή στις αρχές του '90. Στα πρώιμα αυτά στάδια το μεγαλύτερο πρόβλημα ήταν, κυρίως, η μεγάλη υπολογιστική ισχύς που απαιτούσε ένα σύστημα για να μπορέσει να απεικονίσει τέτοιου είδους γραφικά και λεπτομέρειες. Ο λόγος που προκύπτει αυτό το πρόβλημα είναι, ότι για να αποδοθεί ένα αντικείμενο του πραγματικού κόσμου τρισδιάστατα, κατασκευάζεται από πάρα πολλά και μικρά πολύγωνα, μέσω πολύπλοκων μαθηματικών πράξεων. Αυτή η διαδικασία παραγωγής μιας εικόνας, που αποτελείται από δεδομένα που αφορούν τον τρισδιάστατο χώρο και είναι αποθηκευμένα στον υπολογιστή, ονομάζεται Τρισδιάστατη Απόδοση (3D Rendering). Στην πραγματικότητα, με τη διαδικασία αυτή, τρισδιάστατα μοντέλα μετατρέπονται σε δισδιάστατες εικόνες (2D) με τρισδιάστατα φωτορεαλιστικά, ή με εφέ που πλησιάζουν την πραγματικότητα του αντικειμένου που έχει αποτυπωθεί.

Οι απαιτήσεις της εποχής, αναφορικά με το χρόνο και το οικονομικό κόστος, έχουν οδηγήσει στη δημιουργία εφαρμογών που μπορούν να απεικονίσουν και να επεξεργαστούν τρισδιάστατα μοντέλα σε μικρό χρονικό διάστημα, αναλόγως του όγκο του μοντέλου, με σχετικά χαμηλό κόστος. Σ' αυτό βοήθησε το γεγονός ότι η χρήση των τρισδιάστατων γραφικών και μοντέλων έχει υιοθετηθεί από πολλά επαγγέλματα και από πολλούς τομείς της καθημερινής ζωής [12].

Σήμερα με την ανάπτυξη της τεχνολογίας, η απεικόνιση των αντικειμένων ή των χώρων γίνεται πέρα από τους παραπάνω τρόπους και με τη χρήση Νεφών Σημείων (Point Cloud). Το νέφος σημείων είναι ένα πακέτο δεδομένων το οποίο περιλαμβάνει στοιχεία για κάθε σημείο στο χώρο. Τα σημεία αυτά αποτυπώνονται μέσω ειδικών οργάνων, όπως Laser Scanner, Light Detection and Ranging (LiDAR) ή με καθαρά φωτογραμμετρικές μεθόδους. Το πακέτο δεδομένων των σημείων αυτών δημιουργεί τις επιφάνειες του μοντέλου, και κατ' επέκταση το ίδιο το μοντέλο. Κάθε σημείο έχει τις δικιές του συντεταγμένες στο χώρο. Οι συντεταγμένες αυτές αρχικά αναφέρονται στο σύστημα του ειδικού οργάνου, ωστόσο στη συνέχεια μέσω ειδικών προγραμμάτων μπορούν να γεωαναφερθούν στο επιθυμητό προβολικό σύστημα. Η ραγδαία ανάπτυξη της τεχνολογίας καθιστά

εφικτή τη δημιουργία Νεφών Σημείων ακόμη και μέσω έξυπνου κινητού τηλεφώνου (Smartphone), ή ταμπλέτας (Tablet), τα οποία αποδίδουν αρκετά ικανοποιητικά το επιθυμητό αντικείμενο ή χώρο.



Σχημα 1.6: Point cloud από ipad 2020 [5]

## 1.3 Επαυξημένη Πραγματικότητα - Εικονική Πραγματικοτητα - Μικτή Πραγματικότητα

Με τον όρο Επαυξημένη Πραγματικότητα (Augmented Reality - AR) εννοείται η παρουσίαση ενός ή περισσότερων ψηφιακών αντικειμένων ή πληροφοριών, που αναπαράγονται μέσω συσκευών, όπως κινητές συσκευές Android και iOS, υπολογιστές και Tablet, στον πραγματικό κόσμο, σε πραγματικό χρόνο. Με την τεχνολογία αυτή, το πραγματικό περιβάλλον μετατρέπεται σε διαδραστικό, στο οποίο ο χρήστης μπορεί να εμφανίσει οτιδήποτε επιθυμεί, ανάλογα με την εφαρμογή του. Πέρα από την Επαυξημένη Πραγματικότητα, υπάρχουν και η Εικονική Πραγματικότητα (Virtual Reality - VR), καθώς και η Μικτή Πραγματικότητα (Mixed Reality - MR) [13]. Η πρώτη αφορά τη δημιουργία ενός τεχνητού περιβάλλοντος, το οποίο είναι και η βασική διαφορά με την Επαυξημένη Πραγματικότητα, η οποία αναφέρεται στον πραγματικό κόσμο. Ο χρήστης μπορεί να αλληλεπιδρά, με το τεχνητό αυτό περιβάλλον, μέσω ειδικών συσκευών, όπως είναι τα VR γυαλιά, συσκευές χειρός και γάντια με αισθητήρες, ώστε να έχει μια πιο ρεαλιστική αίσθηση, Τέλος είναι η Μικτή Πραγματικότητα, η οποία είναι ο συνδυασμός Επαυξημένης και Εικονικής Πραγματικότητας. Τα φυσικά και ψηφιακά αντικείμενα συνυπάρχουν, καθώς ο χρήστης εναλλάσσεται μεταξύ πραγματικού και εικονικού κόσμου. Οι τρεις αυτές Πραγματικότητες έχουν ποικίλες χρήσεις, από τη ψυχαγωγία και τη δουλειά, έως τη χρήση τους σε στρατιωτικές εκπαιδεύσεις και σε προσομοιώσεις.

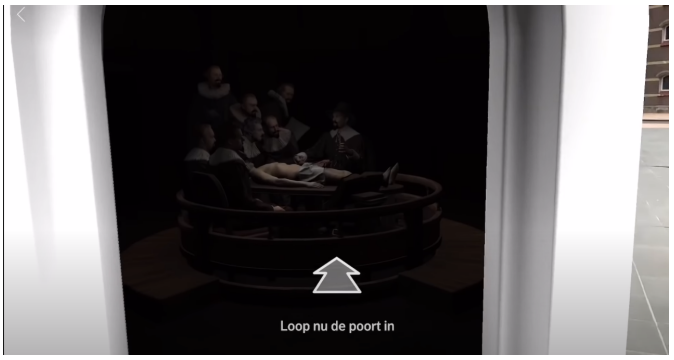
## 1.4 Βιβλιογραφική Ανασκόπηση Εφαρμογών Επαυξημένης Πραγματικότητας

Στην παρούσα διπλωματική γίνεται χρήση της Επαυξημένης Πραγματικότητας για την εικονική περιήγηση σε έναν χώρο ανασκαφών, ο οποίος έχει αποτυπωθεί και παρουσιάζεται με τη μορφή νέφους σημείων. Η χρήση Επαυξημένης Πραγματικότητας δεν περιορίζεται μόνο στον τομέα αυτόν, αλλά και σε πολλούς άλλους, όπως ο τουρισμός, η αγορά ακινήτων και γενικότερα στον τομέα του Πολιτισμού.

Μία εφαρμογή η οποία χρησιμοποιεί Επαυξημένη Πραγματικότητα είναι η *Rembrandt Reality* [14]. Μέσω της εφαρμογής αυτής γίνεται περιήγηση μέσα στο έργο του Ολλανδού ζωγράφου Rembrandt, ο οποίος απεικονίζει τον Dr Nicolaes Tulp να εξηγεί την ανθρώπινη ανατομία σε μια ομάδα γιατρών, όπως φαίνεται στην Εικόνα 1.7

Σχημα 1.7: Πίνακας του Rembrandt ¨Μάθημα ανατομίας του Dr Tulp'

Πιο συγκεκριμένα κατά τη χρήση της εφαρμογής ο χρήστης, αφού σαρώσει το επίπεδο για να το εντοπίσει η συσκευή (Plane Detection), τοποθετεί την AR Portal πάνω σε αυτό και στη συνέχεια εισέρχεται στο δωμάτιο, στο οποίο μπορεί να περιηγηθεί, ενώ ταυτόχρονα ακούει μια ηχητική διήγηση της ιστορίας του πίνακα.




Σχημα 1.8: AR Portal

Σχημα 1.9: AR Portal Interior

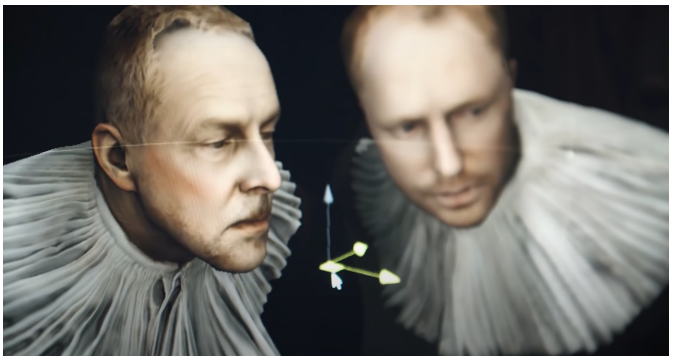
Για την υλοποίηση της εφαρμογής αυτής, αποτυπώθηκε ο χώρος με χρήση της φωτογραφικής μηχανής MatterPort, όπως φαίνεται στην Εικόνα 1.10, ο οποίος στη συνέχεια αποδόθηκε με τη μορφή τρισδιάστατου μοντέλου, Εικόνα 1.11.

Σχημα 1.10: Φωτογραφική μηχανή Matterport

Σχημα 1.11: Τρισδιάστατο μοντέλο χώρου

Επίσης χρησιμοποιήθηκαν άτομα, τα οποία, αφού φόρεσαν τις κατάλληλες ενδυμασίες, φωτογραφήθηκαν κατά 360 μοίρες, ώστε να αποδοθεί το τρισδιάστατο μοντέλο τους, μέσω φωτογραμμετρικών μεθόδων, όπως χρήση αλγορίθμων Sfm [15], [16]. Η μέθοδος αυτή βοηθάει στη δημιουργία τρισδιάστατων μοντέλων, μέσω λήψης RGB φωτογραφιών. Πιο συγκεκριμένα χρησιμοποιεί επικαλυπτόμενες δισδιάστατες φωτογραφίες για την ανακατασκευή της τρισδιάστατης μορφής των αντικειμένων. Όσο μεγαλύτερη επικάλυψη υπάρχουν μεταξύ των φωτογραφιών, τόσο καλύτερα ποιοτικά αποδίδεται το τρισδιάστατο μοντέλο. Αφού φωτογραφήθηκαν, με τον κατάλληλο εξοπλισμό, όπως φαίνεται στην Εικόνα 1.12, στη συνέχεια με τη βοήθεια φωτογραμμετρικών και σχεδιαστικών προγραμμάτων (RealityCapture, Blender) δημιουργήθηκαν τα τελικά μοντέλα της εφαρμογής, Εικόνα 1.13, 1.14.

Σχημα 1.12: Αποτύπωση και δημιουργία τρισδιάστατων ανθρώπινων μοντέλων

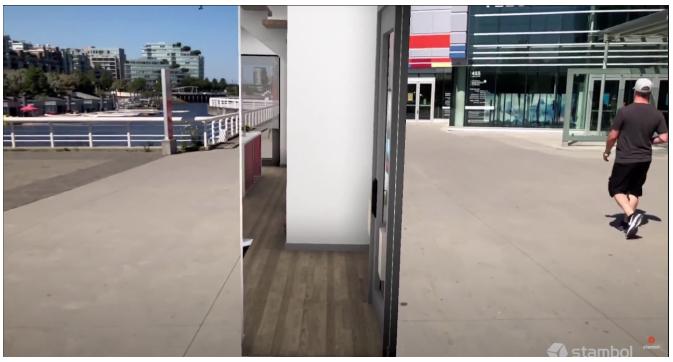


Σχημα 1.13: Αποτύπωση και δημιουργία τρισδιάστατων ανθρώπινων μοντέλων

Σχημα 1.14: Τρισδιάστατη Απεικόνιση του Πίνακα

Η εφαρμογή αυτή, έχει μεγάλη ακρίβεια στα τρισδιάστατα μοντέλα, των οποίων οι λεπτομέρειες αποδίδονται με υψηλή ευκρίνεια, ωστόσο το κόστος και ο χρόνος για να αποδοθεί ένα τέτοιο θέμα με τόσο υψηλή ακρίβεια είναι αρκετά μεγάλο.

Μία άλλη εφαρμογή που κάνει χρήση της Επαυξημένης Πραγματικότητας αυτή τη φορά στον τομέα της αγοράς ακινήτων είναι η AR Portal for Real Estate Marketing [17]. Η εφαρμογή αυτή είναι πιο απλή από την προηγούμενη, καθώς με ένα απλό άγγιγμα στην οθόνη της συσκευής τοποθετείται η AR Portal στο έδαφος, Εικόνα 1.15. Το εσωτερικό αποτελείται από μοντέλα που έχουν κατασκευαστεί σε σχεδιαστικό πρόγραμμα.



Σχημα 1.15: AR Portal

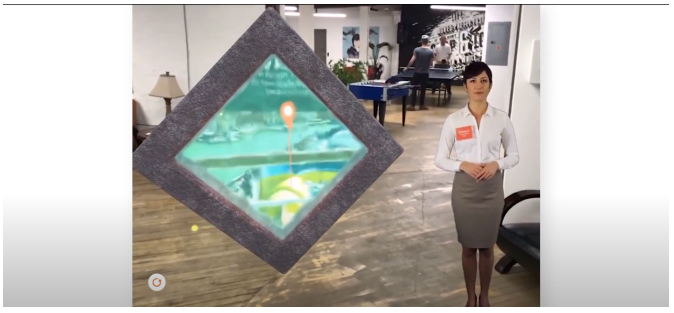
Στην πραγματικότητα είναι μια εφαρμογή που δίνει τη δυνατότητα στον χρήστη να περιηγείται στο ακίνητο που τον ενδιαφέρει σαν να βρίσκεται μέσα σε αυτό σε πραγματικό χρόνο, αντί να βλέπει το ακίνητο μέσω φωτογραφιών όπως συνήθως γίνεται, Εικόνα 1.16, Εικόνα 1.17, Εικόνα 1.18. Η κατασκευή μιας τέτοιας εφαρμογής είναι αρκετά απλή, καθώς το τρισδιάστατο μοντέλο του ακινήτου μπορεί να αποδοθεί μέσω σχεδιαστικών λογισμικών, είτε μέσω λογισμικών τα οποία δημιουργούν το τρισδιάστατο μοντέλο του ακινήτου ακόμα και από φωτογραφίες που έχουν ληφθεί με ένα κινητό τηλέφωνο.


Σχημα 1.16: AR Portal Interior

Σχημα 1.17: AR Portal Interior

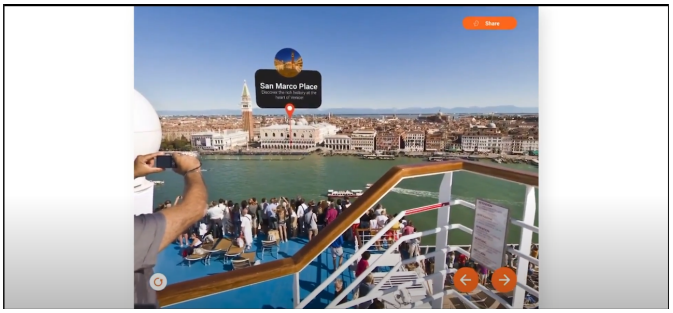
Σχημα 1.18: AR Portal Interior

Τέλος οι τομείς στους οποίους βρίσκονται οι περισσότερες εφαρμογές Επαυξημένης Πραγματικότητας είναι οι τομείς του τουρισμού και του πολιτισμού. Η εφαρμογή Metavrse - AR Travel Portal Demo [18] παρουσιάζει το πρόγραμμα που ακολουθείται σε μία κρουαζιέρα. Πιο συγκεκριμένα, ο χρήστης μπαίνοντας σε μια AR Portal βγαίνει στο κατάστρωμα του κρουαζιερόπλοιου και μπορεί να δει τους ταξιδιωτικούς προορισμούς από τους οποίους θα περάσει, με ταυτόχρονη ηχητική περιγραφή, από μια εικονική ξεναγό, Εικόνα 1.19.

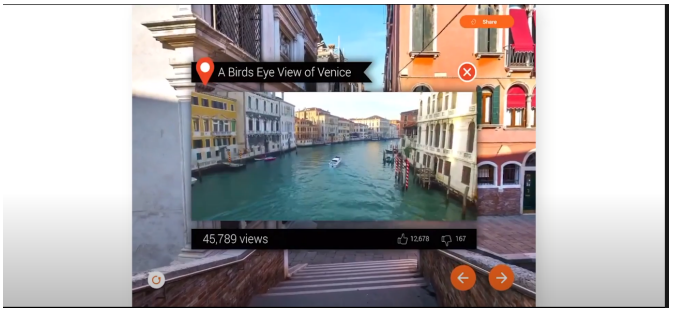


Σχημα 1.19: AR Portal - Guide

Επίσης, με τη χρήση πολυμέσων (multimedia), όπως βίντεο και φωτογραφίες, η ξεναγός παρουσιάζει στο χρήστη τα σημεία ενδιαφέροντος (HotSpots) των εν λόγω προορισμών, ώστε να έχει μια ολοκληρωμένη εικόνα πριν την επίσκεψή του σε αυτούς, Εικόνα 1.20, Εικόνα 1.21.

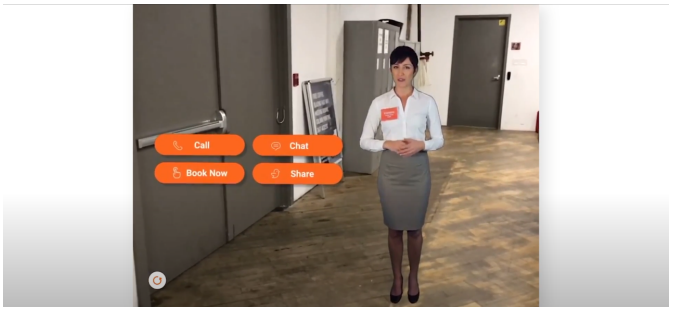



Σχημα 1.20: Cruise Destinations




Σχημα 1.21: Multimedia (Video)

Επίσης δίνεται η δυνατότητα στο χρήστη να επικοινωνήσει με την εταιρία, μέσω ενός διαδραστικού μενού, είτε να κλείσει μια θέση για οποιαδήποτε κρουαζιέρα επιθυμεί.

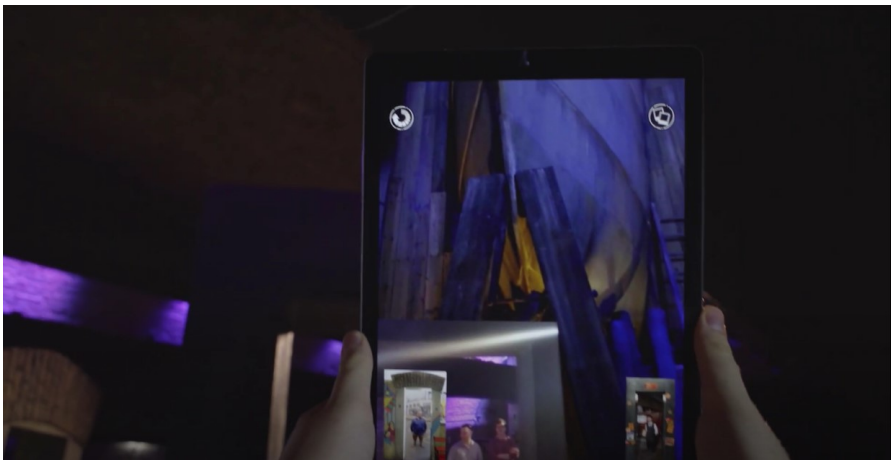



Σχημα 1.22: Menu

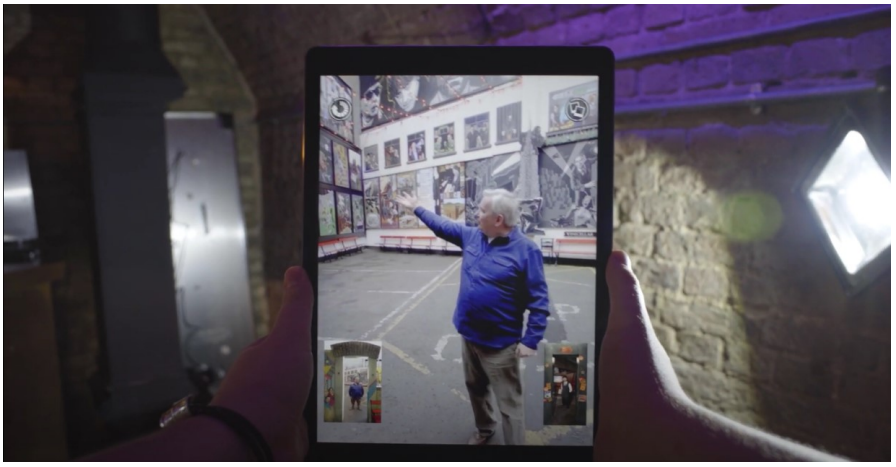
Η εφαρμογή αυτή, σχεδιάστηκε στη μηχανή MetaVRse Engine, που κατασκεύασε η ομώνυμη εταιρία, της οποίας το περιβάλλον είναι παρόμοιο με του Unity. Το βασικό πλεονέκτημα της μηχανής αυτής, είναι ότι για την κατασκευή εφαρμογών δεν απαιτείται χρήση κώδικα, γεγονός που την κάνει πιο προσιτή σε μεγαλύτερη μερίδα κόσμου.

Ενδιαφέρον παρουσιάζει και το project περιήγησης στο Μπλέλφαστ, Ιρλανδία. Πιο συγκεκριμένα, το project αφορά δύο πραγματικές πόρτες, όπου οι επισκέπτες, αφού προμηθεύονται αρχικά με ακουστικά και ένα tablet καλούνται να περάσουν από μέσα. Η μια πόρτα οδηγεί τον επισκέπτη στην εποχή, όπου κατασκευάζεται ο Τιτανικός, καθώς το Μπελφαστ ήταν και η γενέτειρα του Τιτανικού, (Εικόνα 1.23, Εικόνα 1.24), ενώ η άλλη πόρτα φανερώνει στον επισκέπτη την ιστορία της τέχνης του δρόμου (street art), με ταυτόχρονη διήγηση από ξεναγό και στις δυο (Εικόνα 1.25, Εικόνα 1.26). Το περιεχόμενο που εμφανίζεται σε κάθε πόρτα είναι ένα βίντεο 360 μοιρών, το οποίο δίνει στον επισκέπτη την αίσθηση ότι τα βλέπει πραγματικά μπροστά του. Η λογική και ο τρόπος λειτουργίας της εφαρμογής αυτής είναι ίδιος με της παρούσας διπλωματικής εργασίας, με τη διαφορά ότι στη συγκεκριμένη εφαρμογή οι πόρτες είναι πραγματικές.

Σχημα 1.23: Πρώτη Πόρτα - Τιτανικός - Ξεναγός



Σχημα 1.24: Πρώτη Πόρτα - Τιτανικός



Σχημα 1.25: Δεύτερη Πόρτα - Street Art - Ξεναγός

Σχημα 1.26: Δεύτερη Πόρτα - Street Art

# Κεφάλαιο 2

# Unity

## 2.1 Περιγραφή του Unity

Το Unity αποτελεί μια μηχανή δημιουργίας παιχνιδιών, η οποία σχεδιάστηκε το 2004 από την αμερικάνικη εταιρία Unity Technologies, η οποία αρχικά είχε την ονομασία Over the Edge Entertainment (OTEE). Ιδρυτές της εταιρίας αυτής είναι οι David Helgason, Nicholas Francis και Joachim Ante [19]. Η μηχανή αυτή δίνει τη δυνατότητα στους χρήστες να κατασκευάσουν παιχνίδια και εφαρμογές σε δισδιάστατο περιβάλλον (2D), σε τρισδιάστατο (3D), σε Εικονική Πραγματικότητα (Virtual Reality - VR) και σε Επαυξημένη Πραγματικότητα (Augmented Reality - AR). Το Unity έγινε ιδιαίτερα γνωστό το 2008, μέσω του iPhone, της εταιρίας Apple, επειδή ήταν η πρώτη εταιρεία που παρείχε πλήρη υποστήριξη κατασκευής παιχνιδιών για το λειτουργικό σύστημα iOS. Την επόμενη χρονιά (2009), διατέθηκε στο κοινό μια δωρεάν έκδοση της μηχανής αυτής, γεγονός το οποίο προσέλκυσε ακόμα περισσότερους προγραμματιστές και κατασκευαστές παιχνιδιών. Σήμερα χαρακτηρίζεται ως μια "Cross-platform" μηχανή δημιουργίας παιχνιδιών, αφού καταφέρνει να εξυπηρετεί και να συνδράμει στην κατασκευή εφαρμογών και παιχνιδιών για πολλές πλατφόρμες, όπως:

- iOS
- Android
- Windows
- Universal Windows Platform
- Mac
- Linux/Steam OS
- WebGL

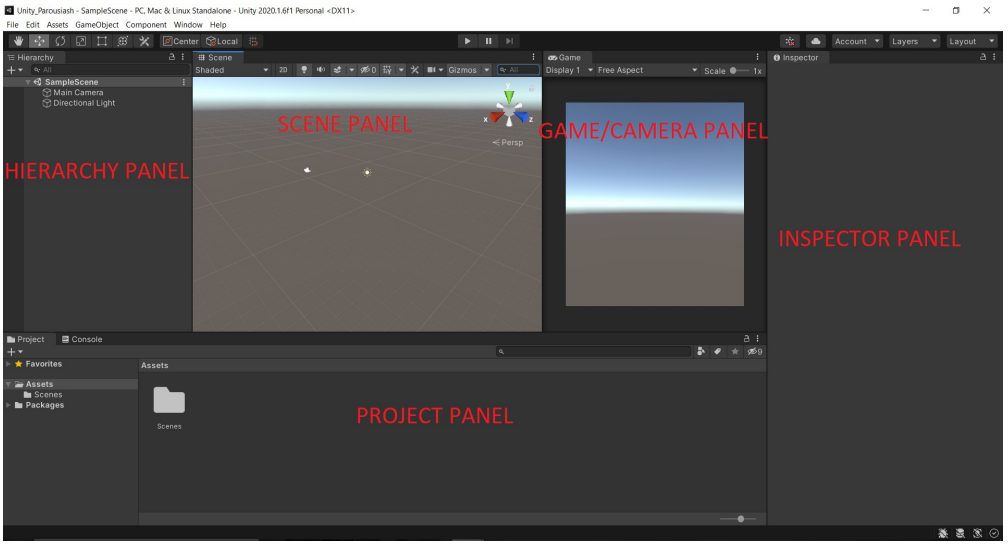
- Playstation
- Xbox One
- Nintendo 3DS
- Oculus Rift
- Google Cardboard Android & iOS
- Steam VR PC & Max
- Playstation VR
- Gear VR
- Windows Mixed Reality
- Daydream
- Android TV
- tvOS
- Nintendo Switch
- Facebook Gameroom
- Apple AR Kit
- Google ARCore
- Vuforia


Σχημα 2.1: Συμβατές Πλατφόρμες [6]

Οι βασικές γλώσσες προγραμματισμού που χρησιμοποιούνται στη μηχανή αυτή είναι οι C#, C++ και Java. Τέλος, η κατασκευή της εν λόγω εφαρμογής έγινε μέσω της πλατφόρμας ανάπτυξης ανοιχτού κώδικα "Mono", η οποία βασίζεται στο .NET Framework και επιτρέπει τη δημιουργία εφαρμογών πολλαπλών πλατφορμών.

## 2.2 Περιβάλλον του Unity

Το Unity αποτελείται από Panels, τα οποία δίνουν τη δυνατότητα στο χρήστη να διαμορφώσει το περιβάλλον του προγράμματος, όπως επιθυμεί ο ίδιος. Τα κυριότερα panels είναι τα Hierarchy Panel, Scene Panel, Camera/Game Panel, Inspector Panel και Project Panel, τα οποία παρουσιάζονται στην παρακάτω εικόνα.




Σχημα 2.2: Περιβάλλον του Unity

### 2.2.1 Hierarchy Panel

Στο εν λόγω παράθυρο παρουσιάζονται όλα τα αντικείμενα που βρίσκονται στη σκηνή. Τα αντικείμενα αυτά μπορεί να είναι η κάμερα, ο φωτισμός, κουμπιά με λειτουργίες και κάποια μοντέλα (Prefabs) που έχουν εισαχθεί.

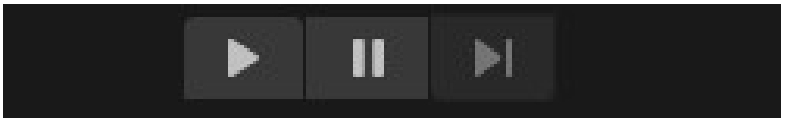
### 2.2.2 Scene Panel

Το παράθυρο αυτό απεικονίζει το χώρο, όπου ο χρήστης κατασκευάζει την εφαρμογή ή το παιχνίδι του. Δίνεται η δυνατότητα πλοήγησης στο χώρο, μέσω της κάμερας του παιχνιδιού, περιστροφής των αντικειμένων, αλλαγής κλίμακας και άλλων δυνατοτήτων, μέσω κάποιων κουμπιών.

Σχημα 2.3: Unity Tools

### 2.2.3 Camera/Game Panel

Το παράθυρο αυτό παρουσιάζει την τελική μορφή που θα έχει το παιχνίδι/εφαρμογή, όταν θα δημιουργηθεί και θα κατασκευαστεί (Build) για την πλατφόρμα που επιθυμεί ο χρήστης.



Σχημα 2.4: Game Panel Buttons

Τα κουμπιά που είναι διαθέσιμα στο παράθυρο αυτό είναι το Play Button, με το οποίο εκτελείται το πρόγραμμα/παιχνίδι, ώστε να έχει ο χρήστης την εικόνα (preview) για το πώς δουλεύει το προϊόν του. Στη συνέχεια υπάρχει το Pause Button, με το οποίο το πρόγραμμα/παιχνίδι σταματάει να τρέχει, ή πιο σωστά «παγώνει», ώστε ο χρήστης να μπορεί να επεξεργαστεί τις επιθυμητές παραμέτρους καθ' όλη τη διάρκεια που τρέχει και δοκιμάζει το προϊόν του. Τέλος το Next Frame Button χρησιμοποιείται, ώστε να μπορεί ο χρήστης να μεταβαίνει από καρέ (frame) σε καρέ με ευκολία.

### 2.2.4 Project Panel

Στο παράθυρο αυτό απεικονίζονται όλα τα Assets του προγράμματος ή του παιχνιδιού. Με τον όρο Assets εννοούνται όλα τα Scripts, Textures, Shaders, Materials και άλλα, τα οποία περιέχονται στο project.

### 2.2.5 Inspector Panel

Στο παράθυρο αυτό εμφανίζονται όλα τα χαρακτηριστικά κάθε αντικειμένου που επιλέγεται. Τα χαρακτηριστικά αυτά προέρχονται τόσο από το Hierarchy panel όσο και από το Project panel.

## 2.3 Εργαλεία που Χρησιμοποιήθηκαν

Τα βασικά εργαλεία που χρησιμοποιήθηκαν στην εκπόνηση της εργασίας αυτής αναλύονται παρακάτω:

- **Terrain (Χώρος)**: Πρόκειται για τον «κόσμο» πάνω στον οποίον θα τοποθετηθούν όλα τα αντικείμενα (3D models). Τα εργαλεία του terrain επιτρέπουν τη δημιουργία εδαφών και χώρων με ιδιαίτερη λεπτομέρεια. Επίσης, δίνουν τη δυνατότητα εισαγωγής υφών (textures) και υψωμέτρων στα διάφορα τμήματα του terrain. Τέλος επιτρέπουν την εισαγωγή και κατασκευή αντικείμενων, όπως δεντρα και πέτρες.

- **3D Models (Τρισδιάστατα Μοντέλα)**: Ένα τρισδιάστατο μοντέλο ή αλλιώς Mesh είναι ένα σύνολο από πολύγωνα, τα οποία διαμορφώνονται κατάλληλα, ώστε να αποτελέσουν ένα αντικείμενο που θα τοποθετηθεί στη σκηνή. Τα αντικείμενα αυτά δημιουργούνται μέσω λογισμικών σχεδίασης , όπως είναι το 3D Studio Max, το Blender, Autocad, Sketcup κ.α. Πολλά από αυτά τα λογισμικά έχουν βιβλιοθήκες, όπου προσφέρουν στους χρήστες έτοιμα μοντέλα.

- **Textures (Υφές)**: Ένα τρισδιάστατο μοντέλο εκτός από την κατάλληλη μορφοποίηση χρειάζεται και μια υφή. Η υφή όσον αφορά το 3D Modeling είναι ουσιαστικά μια επίπεδη εικόνα που ¨ντύνει' ένα τρισδιάστατο μοντέλο για να του δώσουν χρώμα, λεπτομέρεια και γενικότερα μια πιο ρεαλιστική υπόσταση. Όσο περισσότερη λεπτομέρεια έχει ένα τρισδιάστατο μοντέλο τόσο μεγαλύτερη είναι η απαίτηση για μια καλύτερη και υψηλότερης ανάλυσης υφή κατά την απεικόνιση. Η δημιουργία ενός Texture μπορεί να επιτευχθεί με κατάλληλα προγραμματιστικά εργαλεία επεξεργασίας εικόνας, όπως είναι το Photoshop, Mudbox, ZBrush κ.α.

- **Lighting (Φωτισμός)**: Ένας κατάλληλος φωτισμός προσφέρει στο μοντέλο μια πιο καλαίσθητη και ρεαλιστική απεικόνιση. Ο φωτισμός είναι σημαντικός για την καλή απόδοση ενός μοντέλου, επειδή βοηθάει και στην κατάλληλη απόδοση των σκιάσεων στα αντικείμενα, έτσι ώστε να δημιουργείται η αίσθηση της πραγματικότητας. Στο Unity παρέχονται τρεις κύριοι τύποι φωτισμών (lights): Point Light, Directional Light και Spot Light.

- **Animation (Κίνηση)**: Αφορά εφαρμογές στις οποίες πρέπει να συνδυάζονται τα στοιχεία κίνησης και διαδραστικότητας, όπως είναι τα ηλεκτρονικά παιχνίδια. Η λειτουργία αυτή μπορεί να εφαρμοστεί και στα τρισδιάστατα μοντέλα. Το Animation System του Unity, βασίζεται στην έννοια του Animation Clip. Κάθε clip περιέχει πληροφορίες σχετικά με το αντικείμενο, όπως για το ποια είναι η θέση του, η στροφή του και το μέγεθός του. Κατά την εισαγωγή του στη σκηνή, το μοντέλο θα πραγματοποιεί αυτή τη κίνηση αυτόματα ή όταν ο χρήστης το επιθυμεί, μέσω πληκτρολογίου ή του ποντικιού του ή μέσω της οθόνης αφής, όταν μιλάμε για κινητό ή tablet.

- **Scripting (Προγραμματισμός)**: Μέσα από την μηχανή του Unity, παρέχεται ένας τρόπος για να περιγραφούν οι συμπεριφορές των αντικειμένων (game object), όπως κίνηση, περιστροφή, χειρισμός χαρακτήρα κ.α, που βρίσκονται στη σκηνή. Αυτό επιτυγχάνεται μέσω των scripts, τα οποία μπαίνουν στο αντικείμενο, ως στοιχείο του και του δινουν την επιθυμητή ικανότητα/ιδιότητα. Τα scripts μπορούν να προστεθούν στα αντικείμενα της σκηνής και να επαναχρησιμοποιηθούν και σε άλλα projects. Οι κώδικες αυτοί(scripts) κατασκευάζονται σε ένα ολοκληρωμένο περιβάλλον ανάπτυξης για .NET Framework, το οποίο αναλύεται λεπτομερώς στη συνέχεια.[20]Υπάρχουν πολλές γλώσσες προγραμματισμού που υποστηρίζουν αυτές τις διεργασίες όπως είναι η Java 3D, JavaScript και η C#.[21]

- **Cameras (Κάμερα)**: Ένα από τα βασικά στοιχεία επίσης σε μια τρισδιάστατη εφαρμογή είναι η κάμερα. Σε κάθε σκηνή εισάγουμε μια τουλάχιστον κάμερα, μέσα απ' την οποία ο χρήστης μπορεί να αλληλεπιδράσει με τα στοιχεία που βρίσκονται στη σκηνή. Δίνεται, επίσης, η δυνατότητα να προσθέσουμε περισσότερες από μια κάμερες, αν το παιχίδι ή εφαρμογή το απαιτεί, ώστε να μπορεί ο χρήστης να προσεγγίζει τα αντικείμενα από πολλές και διαφορετικές οπτικές γωνίες.. Η χρήση καμερών παρέχεται και υποστηρίζεται από όλες σχεδόν τις πλατφόρμες δημιουργίας τρισδιάστατων εφαρμογών όπως είναι το 3D Studio Max και το Unity.

- **Rendering (Διαδικασία παραγωγής εικόνας - Απόδοση)**: Με τον όρο Rendering, αναφέρεται η διαδικασία προβολής των αντικειμένων στην οθόνη της συσκευής. Η μηχανή αυτή, για να ικανοποιήσει τις ανάγκες τις σε γραφικά χρησιμοποιεί τις εφαρμογές (Application Programming Interface - API, OpenGL, Direct3D και OpenGl ES), για κινητές έξυπνές συσκευές, όπως (Android, iOS). Το Rendering δίνει τη δυνατότητα επεξεργασίας του φωτισμού και της απόδοσης σκιάσεων στο χώρο.[20]

# Κεφάλαιο 3

# Εφαρμογές Διπλωματικής Εργασίας

Στα πλαίσια της εκπόνησης της παρούσας εργασίας, που αφορά τη δημιουργία AR Portal στο χώρο ανασκαφών της Τούμπας, κατασκευάστηκε μια εφαρμογή πρότυπο η οποία απεικονίζει τμήμα της σκηνής του Αρχαίου Θεάτρου της Λάρισας. Σκοπός της πρώτης αυτής εφαρμογής ήταν η αξιολόγηση των διαθέσιμων τεχνικών, λογισμικών, καθώς και της ποιότητας των αποτελεσμάτων που προέκυψαν από τη χρήση τους. Κατά τη διαδικασία δημιουργίας της εφαρμογής επιλύθηκαν προβλήματα (errors) που παρουσιάστηκαν στο λογισμικό του Unity, καθώς επίσης δοκιμάστηκαν διάφορα πακέτα και ρυθμίσεις ώστε να επιτευχθεί το καλύτερο δυνατό αποτέλεσμα. Επίσης χρησιμοποιήθηκαν φωτογραμμετρικές τεχνικές και λογισμικά για την αποτύπωση και δημιουργία του τρισδιάστατου μοντέλου.

## 3.1 AR Portal - Αρχαίο Θέατρο Λάρισας

### 3.1.1 Περιοχή Μελέτης

Το Αρχαίο Θέατρο της Λάρισας, χωρητικότητας δέκα χιλιάδων θεατών, κατασκευάστηκε στο πρώτο μισό του 3ου αιώνα π.Χ., επί βασιλείας Αντίγονου Γονατά, βασιλιάς Μακεδονίας. Αυτό εξηγείται από το γεγονός, πώς μετά το θάνατο του Μεγ. Αλεξάνδρου, η Θεσσαλία αποτέλεσε τμήμα του βασιλείου της Μακεδονίας. Η λειτουργία του Θεάτρου κράτησε περίπου έξι αιώνες και πιο συγκεκριμένα μέχρι τα τέλη του 3ου αιώνα μ.Χ και αρχές 4ου αιώνα μ.Χ. Οι μεγαλύτερες καταστροφές επήλθαν από έναν ισχυρό σεισμό τον 7ο Αιώνα μ.Χ. . Ο χώρος αυτός, πέρα από τις θεατρικές παραστάσεις που φιλοξενούσε, αποτέλεσε χώρο συνέλευσης της Αγοράς, καθώς και χώρο λατρείας του θεού Διονύσου, σύμφωνα με τα ευρήματα βωμών προς τιμήν του. Το Θέατρο έχει την αρχιτεκτονική

του Ελληνιστικού θεάτρου, καθώς αποτελείται από το κοίλο, το επιθέατρο, το κυρίως θέατρο, την ορχήστρα, τους παρόδους, το προσκήνιο και τη σκηνή. Χαρακτηριστικό της αρχιτεκτονικής του είναι ότι το κοίλο του Θεάτρου ήταν η ίδια η πλαγιά του λόφου, η οποία διαμορφώθηκε σε αναβαθμίδες για να τοποθετηθούν τα εδώλια. Επί ρωμαϊκής εποχής, το Θέατρο διαμορφώθηκε, ώστε να φιλοξενεί αγώνες μεταξύ μονομάχων, είτε αγώνες μεταξύ μονομάχων και θηρίων. Για τη μετατροπή του Θεάτρου σε αρένα αφαιρέθηκαν οι τρεις πρώτες σειρές των εδωλίων και κατασκευάστηκε ψηλό τοίχος για την ασφάλεια των θεατών. Σήμερα σώζεται μεγάλο μέρος του Θεάτρου και των εδωλίων του, όπως προέκυψε από τις ανασκαφές και παρουσιάζεται στην Εικόνα 3.1

Σχημα 3.1: Αρχαίο Θέατρο Λάρισας

### 3.1.2 Αποτύπωση - Δημιουργία του Τρισδιάστατου Μοντέλου

Αρχικά σχεδιάστηκε το πλάνο πτήσης του Drone και πιο συγκεκριμένα του Phantom 4 Pro, ώστε να υπάρχει πλήρης κάλυψη του τμήματος που επρόκειτο να αποτυπωθεί. Για να γίνει αυτό ορίστηκαν δύο πτήσεις με αρκετές διαδρομές, ώστε να υπάρχει η κατάλληλη επικάλυψη μεταξύ των φωτογραφιών, μέθοδος Structure From Motion (Sfm). Οι φωτογραμμετρικές αυτές μέθοδοι, χρησιμοποιούν μαθηματικούς και φωτογραμμετρικούς αλγορίθμους, κάνοντας τη διαδικασία πιο αυτοματοποιημένη και επομένως πιο εύκολη για το χρήστη [22]. Γενικότερα η μέθοδος αυτή είναι μια καλή εναλλακτική αντί για τη χρήση laser scanner και μπορεί να αποδώσει μοντέλα με πολύ καλή ακρίβεια [23]. Στην προκειμένη περίπτωση η επικάλυψη ορίστηκε στο 75%. Κατά τη διάρκεια της πρώτης πτήσης, η κάμερα του drone ρυθμίστηκε στις 90 μοίρες (κατακόρυφες φωτογραφίες), όπως φαίνεται στην

Εικόνα 3.2, και στη συνέχεια στις 20 μοίρες (πλάγιες φωτογραφίες), όπως φαίνεται στην Εικόνα 3.3. Η λήψη φωτογραφιών με διαφορετικές πόζες έγινε ούτως ώστε να τηρούνται οι προϋποθέσεις για την απόδοση ενός τρισδιάστατου μοντέλου.

Σχημα 3.2: Φωτογραφία 90 μοιρών

Σχημα 3.3: Φωτογραφία 20 μοιρών

Επόμενο βήμα ήταν η δημιουργία του νέφους σημείων με τη χρήση του λογισμικού Agisoft Metashape. Αρχικά έγινε η ευθυγράμμιση των φωτογραφιών (Align Photos). Κατά τη διαδικασία αυτή εξακριβώνονται τα σημεία κάθε φωτογραφίας και καθένα από αυτά αντιστοιχίζεται σε δύο ή περισσότερες φωτογραφίες, ανάλογα σε πόσες εντοπίζεται.

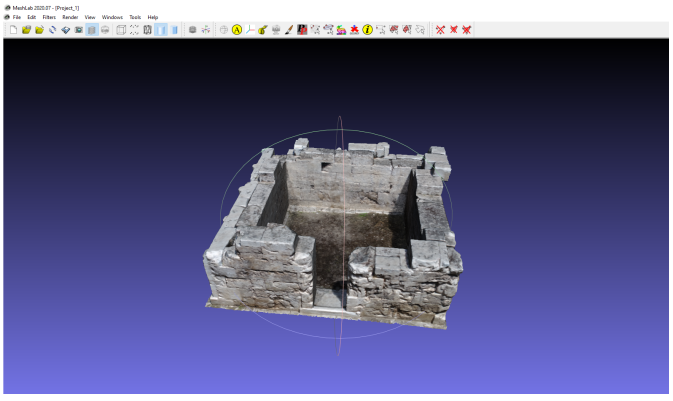
Έπειτα είναι η διαδικασία διόρθωσης των δεδομένων με τη χρήση φωτοσταθερών σημείων (GCPs). Διορθώνοντας τη θέση των φωτογραφιών, το οποίο γίνεται με τη μετακίνηση των εικόνων στα φωτοσταθερά, επιτυγχάνεται η καλύτερη ακρίβεια του μοντέλου , καθώς και η γεωαναφορά του.

Έτσι, όσο μικρότερο είναι το σφάλμα τόσο καλύτερη είναι και η μετριτική ακρίβεια του τρισδιάστατου μοντέλου. Η μετριτική αυτή ακρίβεια ωστόσο εξαρτάται και από την ποιότητα των φωτογραφιών, ώστε να βγει ένα καθαρό μοντέλο, χωρίς ανακρίβειες στις επιφάνειες και τις γωνίες του [24].

Όσον αφορά τη δημιουργία του νέφους σημείων επιλέχθηκε η ποιότητα να είναι υψηλή και το φιλτράρισμα βάθους (Depth Filtering) να είναι επιθετικό (aggressive), ώστε να ταξινομηθούν τα περισσότερα από τα ακραία σημεία του νέφους.

Έπειτα έγινε εισαγωγή του νέφους σημείων στο πρόγραμμα CloudCompare, στο οποίο κόπηκε το επιθυμητό τμήμα της σκηνής. Στη συνέχεια έγινε καθαρισμός του νέφους σημείων από θορύβους, χρησιμοποιώντας τις επιλογές Noise Filter και SOR Filter (Statistic Outlier Removal) της βιβλιοθήκης PCL. Το πρώτο φίλτρο λαμβάνει υπόψιν την απόσταση από μια θεωρητική επιφάνεια που σχηματίζουν τα σημεία. Αν κάποιο σημείο είναι μακριά από την επιφάνεια αυτή διαγράφεται [25]. Το φίλτρο αυτό θεωρείται ένα φίλτρο χαμηλής διέλευσης και η απόσταση από την επιφάνεια μπορεί να αλλάξει ανάλογα με τις ανάγκες του χρήστη. Η δεύτερη μέθοδος, υπολογίζει αρχικά τη μέση απόσταση κάθε σημείου από τα γειτονικά του και στη συνέχεια διαγράφει αυτά που είναι σε απόσταση μεγαλύτερη από τη μέση αυτή απόσταση [26]. Τέλος, και αφού έχουν ολοκληρώθηκαν οι παραπάνω διαδικασίες, έγινε η εξαγωγή του νέφους σημείων με τη μορφή .ply και σε δυαδικό σύστημα (binary system), καθώς αυτό υποστηρίζεται από το Unity.

Το τελικό νέφος σημείων, το οποίο προέκυψε, φαίνεται στην Εικόνα 3.4.



Σχημα 3.4: Point Cloud

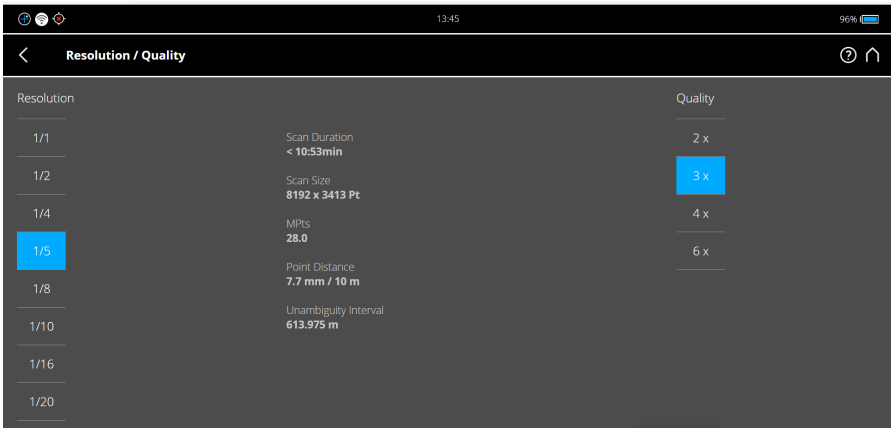
## 3.2 AR Portal - Χώρος Ανασκαφών Τούμπα, Θεσσαλονίκη

### 3.2.1 Περιοχή Μελέτης

Το Νέφος Σημείων αναφέρεται στην ανασκαφική περίοδο Ιουλίου 2019, όπου και πραγματοποιήθηκε τεκμηρίωση των αρχαιολογικών καταλοίπων που ανακαλύφθηκαν σε πέντε ανασκαφικές τομές, διαστάσεων τέσσερα επί τέσσερα, στο βορειοανατολικό άκρο του ανασκαφικού χώρου στην περιοχή Τούμπα Θεσσαλονίκης. Στα σκάμματα αυτά έχουν αποκαλυφθεί τμήμα του νεκροταφείου της Μεσοβυζαντινής περιόδου (7ος-12ος αιώνας μ.Χ.), με ταφές ενήλικων και ανήλικων ατόμων, καθώς και αποσπασματικά σωζόμενα οικοδομικά κατάλοιπα που ανήκουν σε δύο πολύχωρα κτιριακά συγκροτήματα, τα οποία χωρίζονται το ένα από το άλλο από στενό δρόμο. Τα οικοδομικά κατάλοιπα ανήκουν σε δύο οικιστικές περιόδους. Τα κατάλοιπα της νεότερης οικιστικής περιόδου, τα οποία περιλαμβάνουν λίθινα θεμέλια οικοδομημάτων των οποίων η πλήρης κάτοψη δεν μπορεί ακόμα να αποκατασταθεί, με τα μέχρι στιγμής δεδομένα τοποθετούνται στους κλασικούς/πρώιμους ελληνιστικούς χρόνους (5ος-4ος αιώνας π.Χ.). Τα οικοδομικά κατάλοιπα της αρχαιότερης οικιστικής φάσης, που περιλαμβάνουν τα κατώτερα τμήματα τοίχων κτισμένων με ωμοπλίνθους πάνω σε χαμηλά λίθινα θεμέλια, τοποθετούνται στην Πρώιμη Εποχή του Σιδήρου (10ος-9ος αιώνας π.Χ.).

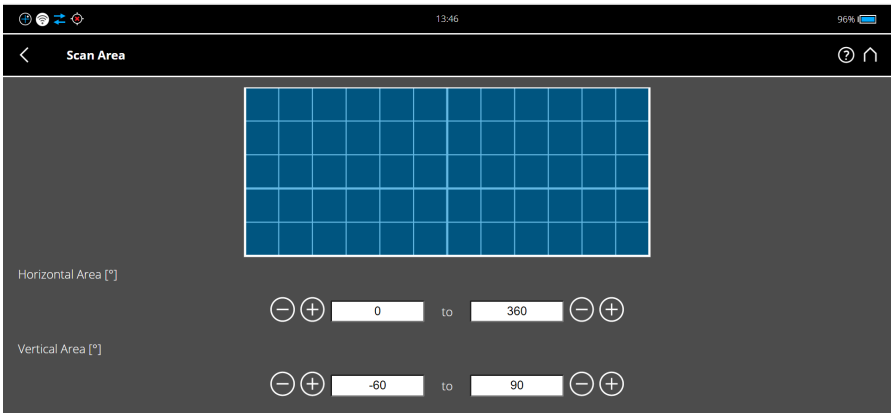
### 3.2.2 Δεδομένα αρχείου – Διαδικασία Αποτύπωσης καταγραφής δεδομένων - Εξαγωγή Νεφών Σημείων

Η αποτύπωση του χώρου των ανασκαφών της Τούμπας έγινε από το Εργαστήριο Φωτογραμμετρίας και Τηλεπισκόπησης του ΚΦΧ του ΤΑΤΜ, ΑΠΘ. Σε αυτή την ενότητα θα παρουσιαστούν οι διαδικασίες συλλογής των δεδομένων, που έγιναν με χρήση Laser Scanner και πιο συγκεκριμένα με το Faro Focus 3D X120, καθώς και με επίγειες λήψεις φωτογραφιών με τη φωτογραφική κάμερα Canon EOS 1200D με φακό 18mm. Η διαδικασία των μετρήσεων με FARO ακολούθησε τα παρακάτω βήματα. Αρχικά στήθηκε το όργανο, στη συνέχεια οριζοντιώθηκε και στο τέλος καθορίστηκαν το όνομα του Project και οι παράμετροι των μετρήσεων. Η ανάλυση που επιλέχθηκε ήταν 1/5 και η ποιότητα $3x$.



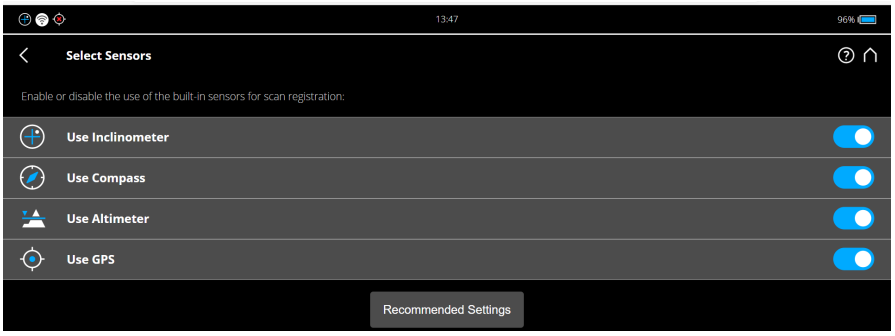

Σχημα 3.5: Ανάλυση και ποιότητα αποτύπωσης

Στη συνέχεια επιλέχθηκε η περιοχή αποτύπωσης, η οποία ήταν από 0 έως 360 μοίρες στο οριζόντιο επίπεδο, και από -60 έως 90 μοίρες στο κατακόρυφο επίπεδο.




Σχημα 3.6: Πεδίο μέτρησης

Κατά τη μέτρηση έγινε χρήση και των αισθητήρων του οργάνου, όπως το κλινόμετρο, το οποίο ελέγχει την κλίση του οργάνου και ελέγχει αν το όργανο είναι σωστά οριζοντιωμένο. Οι υπόλοιποι αισθητήρες είναι η πυξίδα, το αλτίμετρο και το GPS.




Σχημα 3.7: Αισθητήρες Οργάνου

Έγινε επιλογή του χρωματισμού της αποτύπωσης. Όσον αφορά τις φωτογραφίες που παίρνει το όργανο μετά το τέλος των μετρήσεων, έχει επιλεγεί το φως να λαμβάνεται με βάση το οριζόντιο πεδίο και οι φωτογραφίες να έχουν τι μέγιστη ανάλυση.

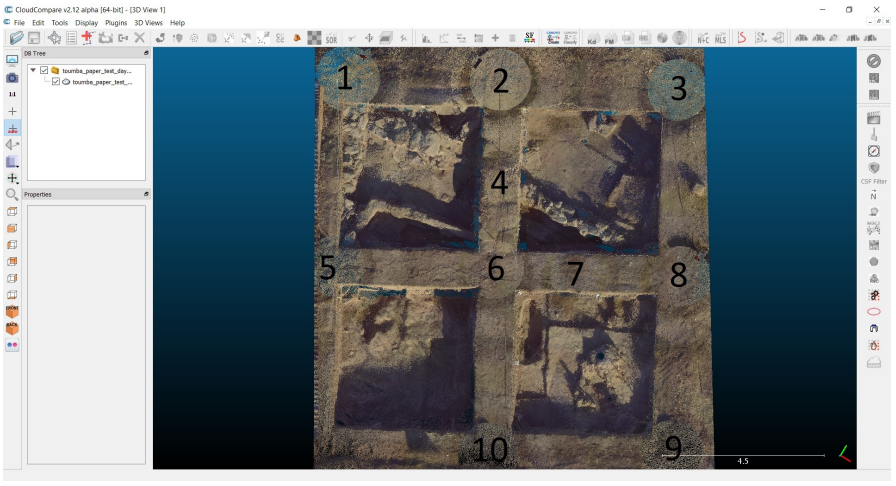

Σχημα 3.8: Επιλογή χρωματισμού αποτύπωσης


Σχημα 3.9: Ρυθμίσεις για τη λήψη φωτογραφιών του FARO

Τέλος, επιλέχθηκαν το καθάρισμα του ουρανού και το καθάρισμα περιγράμματος, το οποίο διαγράφει σημεία που είναι εκτός αυτού.


Σχημα 3.10: Επιλογές καθαρισμού αποτύπωσης

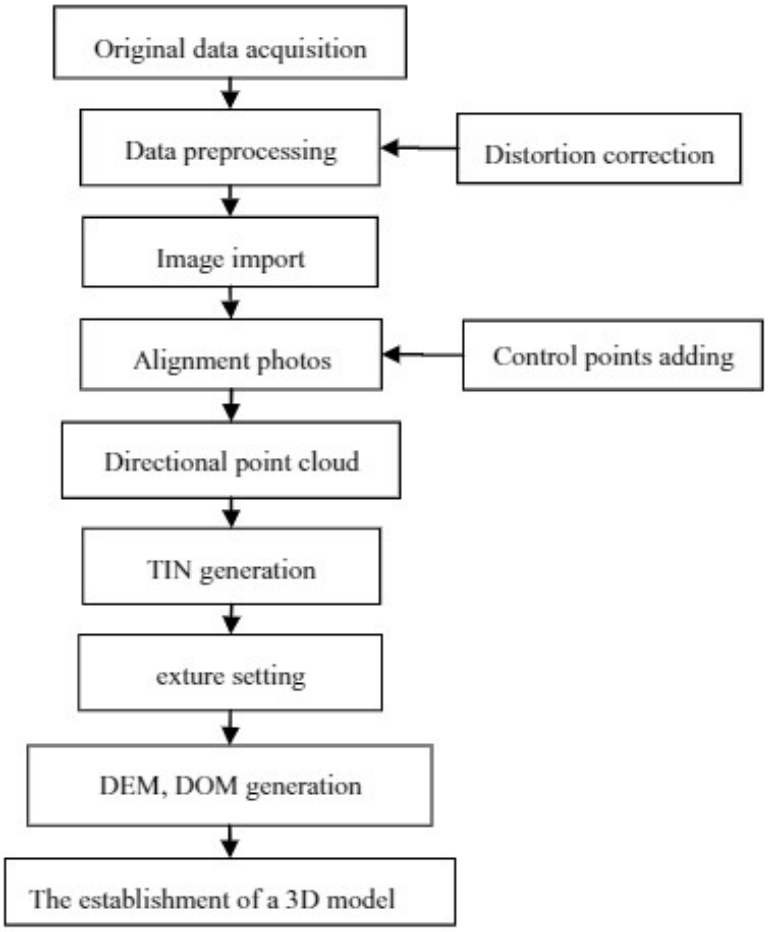
Αφού τελείωσε η μέτρηση, ακολουθήθηκε η ίδια διαδικασία και για τις υπόλοιπες στάσεις του οργάνου. Στη συνέχεια, από το Εργαστήριο Φωτογραμμετρίας και Τηλεπισκόπησης του ΚΦΧ του ΤΑΤΜ, ΑΠΘ έγινε επεξεργασία των μετρήσεων στο λογισμικό Scene της FARO, όπου τα σημεία χρωματίστηκαν. Στο λογισμικό Geomagic Studio έγινε η καταχώριση των σημείων και η συνένωσή τους (Registration) με την μέθοδο Cloud to Cloud, η οποία βασίζεται στην ύπαρξη των κοινών σημείων μεταξύ των αποτυπώσεων, και τέλος, έγινε εξαγωγή του ενιαίου τρισδιάστατου μοντέλου. Έπειτα μέσω του λογισμικού PolyWorks Inspector έγινε η γεωαναφορά του μοντέλου, στο Ελληνικό Γεωδαιτικό Σύστημα Αναφοράς 1987 (ΕΓΣΑ87 ή GGRS87). Το ΕΓΣΑ87 χρησιμοποιεί το ελλειψοειδές του ΓΡΣ80 και ως προβολικό σύστημα την Εγκάρσια Μερκατορική Προβολή με κεντρικό μεσημβρινό λο=24 μοίρες, όπου είναι το βάθρο του Διονύσου, Αθήνα. Για να αποφευχθούν οι αρνητικές τιμές στις συντεταγμένες, ορίστηκε ο κεντρικός μεσημβρινός να έχει ως τετμημένη 500000μ και η αρχή των τεταγμένων θεωρείται ο ισημερινός. Επίσης συντελεστής κλίμακας είναι μο=0.9996. Έτσι η Ελλάδα απεικονίζεται σε μια ζώνη. [27], [28]




Σχημα 3.11: Στάσεις Faro

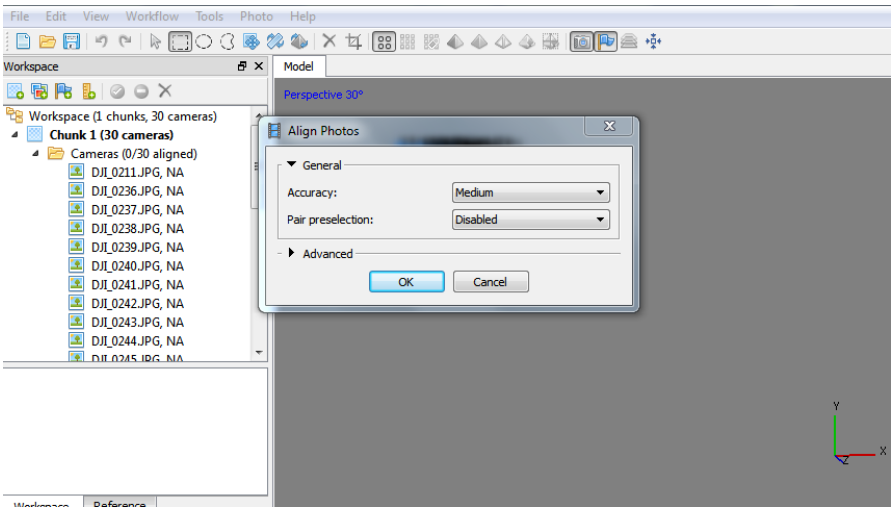
Όσον αφορά στη μέθοδο της φωτογραμμετρίας, ελήφθησαν φωτογραφίες γύρω από την επιθυμητή περιοχή, καθώς και πάνω από αυτή καλύπτοντας νοητά όλο το χώρο, ώστε το μοντέλο που προκύπτει

να έχει όσο το δυνατόν λιγότερα κενά. Αφού ολοκληρώθηκε η διαδικασία αυτή, στη συνέχεια οι φωτογραφίες για κάθε τμήμα του χώρου εισήχθησαν στο λογισμικό Agisoft, ώστε να δημιουργηθεί το τρισδιάστατο μοντέλο, ακολουθώντας τη διαδικασία που περιγράφηκε στην πρώτη εφαρμογή. Τα βήματα για τη δημιουργία του μοντέλου παρουσιάζονται στην Εικόνα 3.12.



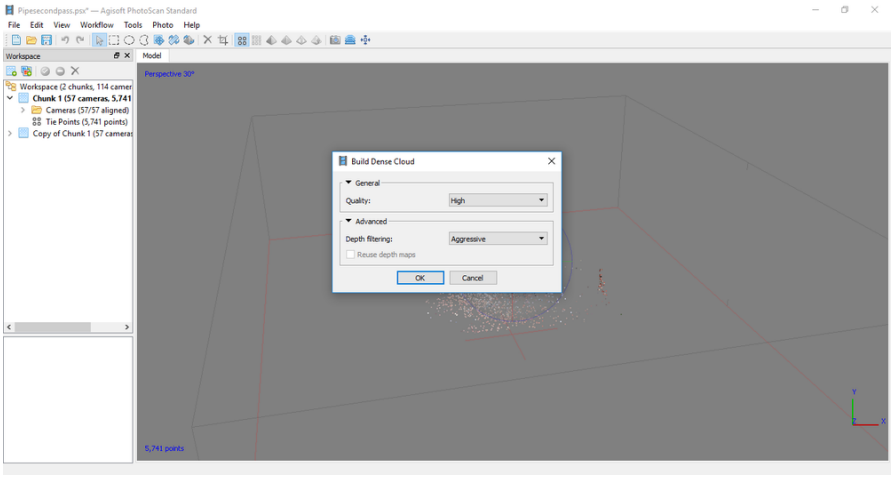

Σχημα 3.12: Διάγραμμα Ροής Δημιουργίας Τρισδιάστατου Μοντέλου - Agisoft [29]

Αρχικά έγινε η διαδικασία ευθυγράμμισης των φωτογραφιών (Align Photos).



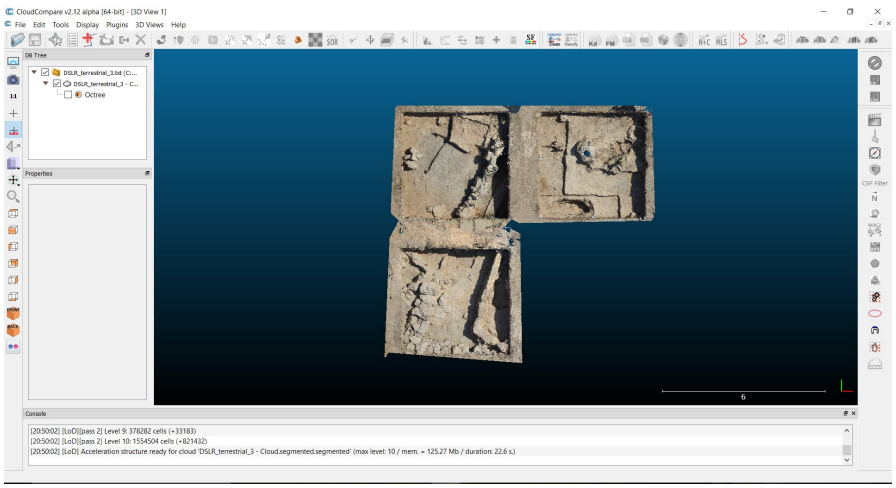

Σχημα 3.13: Align Photos on Agisoft

Ακολούθησε η δημιουργία του Dense Cloud.




Σχημα 3.14: Build Dense Cloud

Το μοντέλο που προκύπτει από την DSLR φωτογραφική μηχανή, φαίνεται παρακάτω.



Σχημα 3.15: Νέφος σημείων από DSLR

### 3.2.3 Επεξεργασία Δεδομένων

Σε εφαρμογές και παιχνίδια που χρησιμοποιούνται Νέφη Σημείων, γίνεται χρήση λογισμικών επεξεργασίας των νεφών αυτών, όπως των Scene της FARO, Cloud Compare, Global Mapper, MeshLab, Blender και άλλων, μέσω των οποίων προκύπτει ένα Νέφος Σημείων πιο καθαρό, πιο ελαφρύ και προσαρμοσμένο στις ανάγκες της εκάστοτε μελέτης. Στην παρούσα πτυχιακή χρησιμοποιήθηκε αρχικά το λογισμικό Global Mapper, ώστε να μετατραπούν τα τρισδιάστατα μοντέλα από μορφές .asc και .txt σε μορφή PLY (Standford Polygon Library). Η μορφή PLY σχεδιάστηκε για να βοηθάει στην αποθήκευση των δεδομένων που προκύπτουν από τους σαρωτές λέιζερ, στις τρεις διαστάσεις. Μπορεί να αποθηκεύσει και να αποδώσει το πλήθος των σημείων που έχουν αποτυπωθεί και αποτελούν το σαρωμένο αντικείμενο (point cloud). Επίσης μπορεί να αποδώσει και τα πολύγωνα που αποτελούν το σαρωμένο τρισδιάστατο μοντέλο (Mesh) [30]. Το αρχείο με κατάληξη .asc είναι τύπου ASCII, δηλαδή ενός συνόλου κωδικοποιημένων χαρακτήρων του λατινικού αλφαβήτου και αναφέρεται στις μετρήσεις που έχουν υλοποιηθεί με το Laser Scanner. Το αρχείο με κατάληξη .txt προέκυψε από τις επίγειες φωτογραφίες που ελήφθησαν με κάμερες DSLR. Στη συνέχεια τα μοντέλα με επέκταση PLY εισήχθησαν στο Cloud Compare, όπου έγινε ο καθαρισμός των μοντέλων, η συνένωσή τους (Merge) και εξήχθησαν πάλι σε μορφή PLY, όχι όμως σε κωδικοποίηση ASCII, αλλά σε κωδικοποίηση Binary, η οποία είναι συμβατή με την εφαρμογή του Unity.

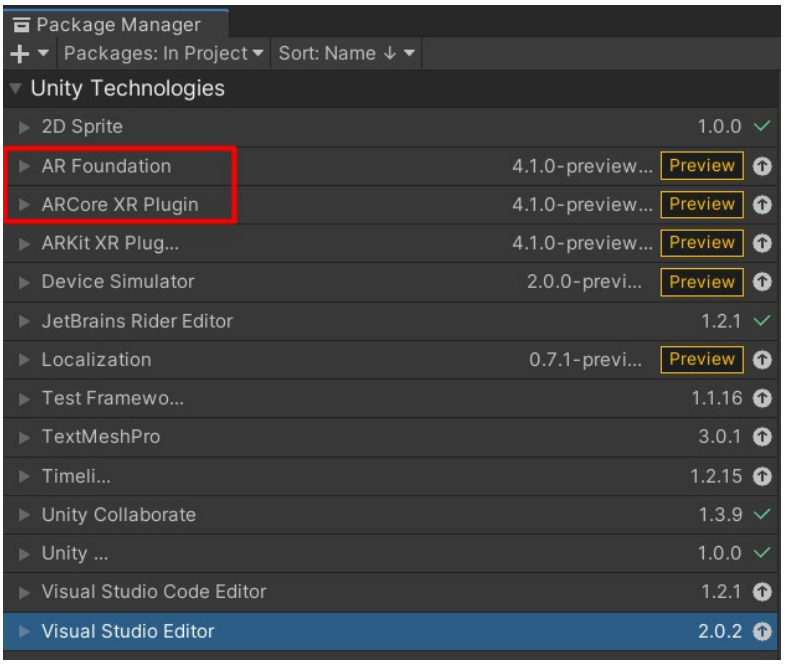
# Κεφάλαιο 4

# Οι εφαρμογές AR Portal

## 4.1 Οι Εφαρμογές στο Unity

### 4.1.1 Πακέτα Unity (Package Manager)

Αρχικά, πριν τη διαδικασία κατασκευής των εφαρμογών, έγινε εγκατάσταση κάποιων πακέτων, μέσω του Unity Package Manager, τα οποία είναι απαραίτητα για τις εφαρμογές Augmented Reality. Από αυτά ξεχωρίζουν το πακέτο AR Foundation και το ARCore XR Plugin, καθώς η εφαρμογή δημιουργήθηκε για κινητά με λειτουργικό σύστημα Android. Αντίστοιχα στην περίπτωση που ο χρήστης επιθυμεί να κατασκευάσει την εφαρμογή για κινητά με λειτουργικό σύστημα iOS, το αντίστοιχο πακέτο του ARCore XR Plugin είναι το ARKit XR Plugin. Το AR Foundation είναι ένα σύνολο MonoBehaviours και APIs, το οποίο διασφαλίζει τη λειτουργικότητα των εφαρμογών Επαυξημένης Πραγματικότητας στις συμβατές πλατφόρμες/συσκευές.

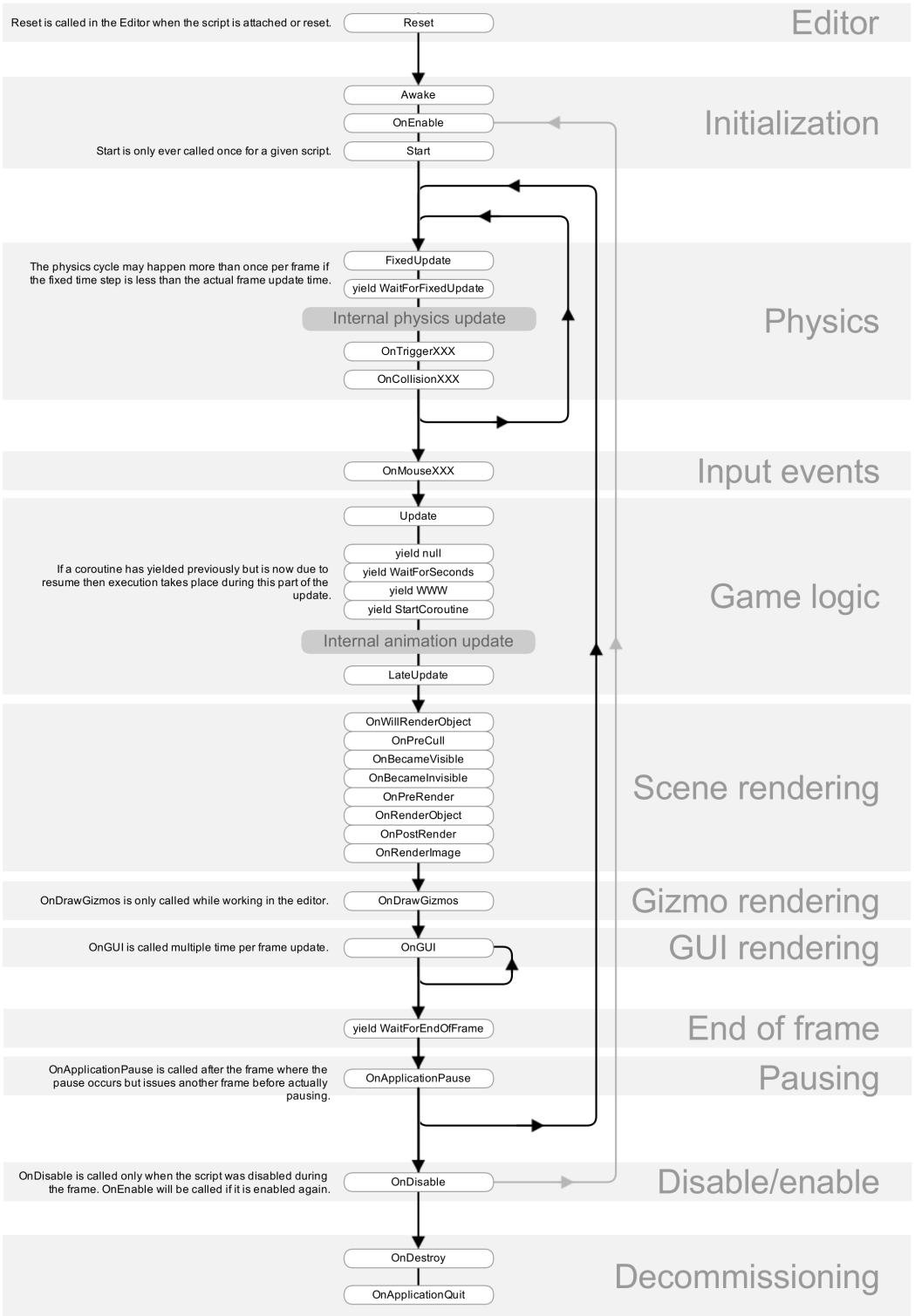



Σχημα 4.1: Package Manager

MonoBehaviour είναι μια κλάση (class) από την οποία αντλούνται οι παράμετροι και οι λειτουργίες που χρησιμοποιούνται σε κάθε κώδικα (script) που χρησιμοποιείται στο Unity. Η χρήση της κλάσης αυτής, όταν ο χρήστης συνθέτει τον κώδικά του σε γλώσσα προγραμματισμού C#, είναι απαραίτητη, καθώς με τις λειτουργίες της βοηθάει στον καλύτερο χειρισμό του κώδικα αυτού. Ωστόσο δίνεται η δυνατότητα στο χρήστη, μέσω του Unity Editor, να απενεργοποιήσει τις λειτουργίες αυτές, οι οποίες είναι οι παρακάτω:

- Start()
- Update()
- FixedUpdate()
- LateUpdate()
- OnGUI()
- OnDisable()
- OnEnable()
- OnDestroy()

Στη συνέχεια παρουσιάζεται το διάγραμμα ροής ενός κώδικα με τις λειτουργίες, οι οποίες και αναλύονται:




Σχημα 4.2: Διάγραμμα Ροής MonoBehaviour

[31]

Οι λειτουργίες αυτές ξεκινάνε μόλις δημιουργηθεί μια νέα σκηνή. Οι χρήσεις των κύριων λειτουργιών αυτών είναι οι εξής [31]:

- Reset(): Χρησιμοποιείται όταν ενεργοποιηθεί το κουμπί Reset ή όταν εισάγονται για πρώτη φορά στοιχεία στη σκηνή.
- Awake(): Χρησιμοποιείται για τη ρύθμιση των αναφορών μεταξύ των scripts και καλείται πάντα πριν τη λειτουργία Start().
- OnEnable(): Χρησιμοποιείται για την ενεργοποίηση και απενεργοποιήση των αντικειμένων που βρίσκονται στη σκηνή.
- Start(): Η λειτουργία αυτή καλείται πάντα μετά τη λειτουργία Awake. Ενεργοποιείται μόνο όταν είναι ενεργοποιημένο το script.
- FixedUpdate(): Η λειτουργία αυτή εξαρτάται από το ρυθμό καρέ (frame rate). Σε χαμηλά frame rates καλείται πολλές φορές, ενώ σε ψηλά δεν καλείται καθόλου. Η λειτουργία αυτή είναι καταλληλότερη για τον υπολογισμό των φυσικών κινήσεων, καθώς σχετίζεται με την απόδοση κάθε λήψης frame.
- Update(): Είναι η λειτουργία που εκτελείται μία μόνο φορά σε κάθε frame. και αποτελεί τη βασική συνάρτηση ενημέρωσης του κάθε frame. Εάν υπάρξει πρόβλημα και το frame κολλήσει, τότε η λειτουργία αυτή δε θα εκτελεστεί ξανά και θα εκτελείται μόνο το FixedUpdate.
- LateUpdate(): Η λειτουργία αυτή ξεκινάει μόλις τελειώσει η λειτουργία Update και εκτελείται σε κάθε frame. Επίσης η λειτουργία αυτή μπορεί να ρυθμίζει τη σειρά εκτέλεσης ενός script.
- OnGUI(): Η λειτουργία αυτή καλείται, κάθε φορά που λαμβάνουν χώρα συμβάντα rendering και processing. Κάθε πρόγραμμα που χρησιμοποιεί GUI (graphical user interface) βασίζεται σε γεγονότα, όπως κλικ στο ποντίκι ή στο πληκτρολόγιο.
- OnDisable(): Η λειτουργία αυτη καλείται όταν το script παύει να λειτουργεί, χωρίς να το καταστρέφει.
- OnDestroy(): Η λειτουργία αυτή καλείται όταν το MonoBehaviour, που αναλύθηκε παραπάνω, καταστρέφεται.

Τα APIs (Application Programming Interface) είναι διαμεσολαβητές που επιτρέπουν σε δύο εφαρμογές να επικοινωνούν μεταξύ τους. Τα APIs ταξινομούνται σύμφωνα με τα namespaces, στα οποία ανήκουν. Το πιο γνωστό API στο Unity είναι το UnityEngine.

Τέλος το AR Foundation αφορά συσκευές που υποστηρίζουν τις εξής έννοιες:

1. World Tracking: Θέση και προσανατολισμός της συσκευής στο περιβάλλον οποιαδήποτε στιγμή.

2. Plane Detection: Ανίχνευση επιφανειών, είτε οριζόντιων, είτε κάθετων, είτε συνδυασμός αυτών.
3. Light Estimation: Εκτίμηση της φωτεινότητας στο φυσικό χώρο.
4. Face Tracking: Εντοπισμός προσώπου
5. Object Tracking: Εντοπισμός τρισδιάστατων αντικειμένων
6. Point Cloud: Απόδοση Νέφους Σημείων

Το επόμενο πακέτο που αναφέρθηκε παραπάνω είναι το ARCore XR Plugin. Το ARCore είναι πλατφόρμα της Google, που χρησιμοποιείται για την κατασκευή εφαρμογών Επαυξημένης Πραγματικότητας, τόσο σε Android, όσο και σε iOS. Αυτό επιτυγχάνεται, καθώς μέσω της χρήσης διάφορων APIs, το ARCore καταφέρνει να καταγράψει, να αποδώσει το περιβάλλον και να αλληλεπιδράσει με διάφορες πληροφορίες και εντολές σε αυτό.

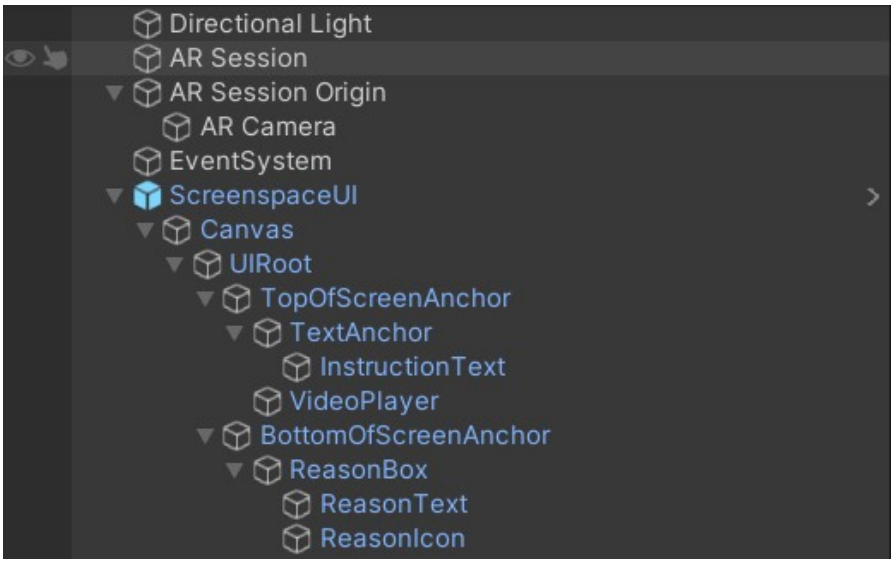
Το ARCore αξιοποιεί τρεις βασικές δυνατότητες του κινητού.

1. Motion Tracking: Επιτρέπει στη συσκευή να εντοπίζει τη σχετική της θέση κάθε στιγμή.
2. Environmental Understanding: Δίνει τη δυνατότητα ανίχνευσης και κατανόησης του περιβάλλοντος και των στοιχείων μέσα σε αυτό.
3. Light Estimation: Εκτίμηση της τάξης της φωτεινότητας.

Τέλος προστέθηκε χειροκίνητα ένα ακόμα custom πακέτο, το Pcx.Package [32], το οποίο ήταν απαραίτητο και στις δύο εφρμογές, καθώς επιτρέπει την εισαγωγή point cloud στο Unity σε μορφή PLY. Το εν λόγω πακέτο λήφθηκε και τροποποιήθηκε ώστε να προσαρμοστεί στις ανάγκες της παρούσας διπλωματικής εργασίας.

### 4.1.2 Εισαγωγή Αντικειμένων

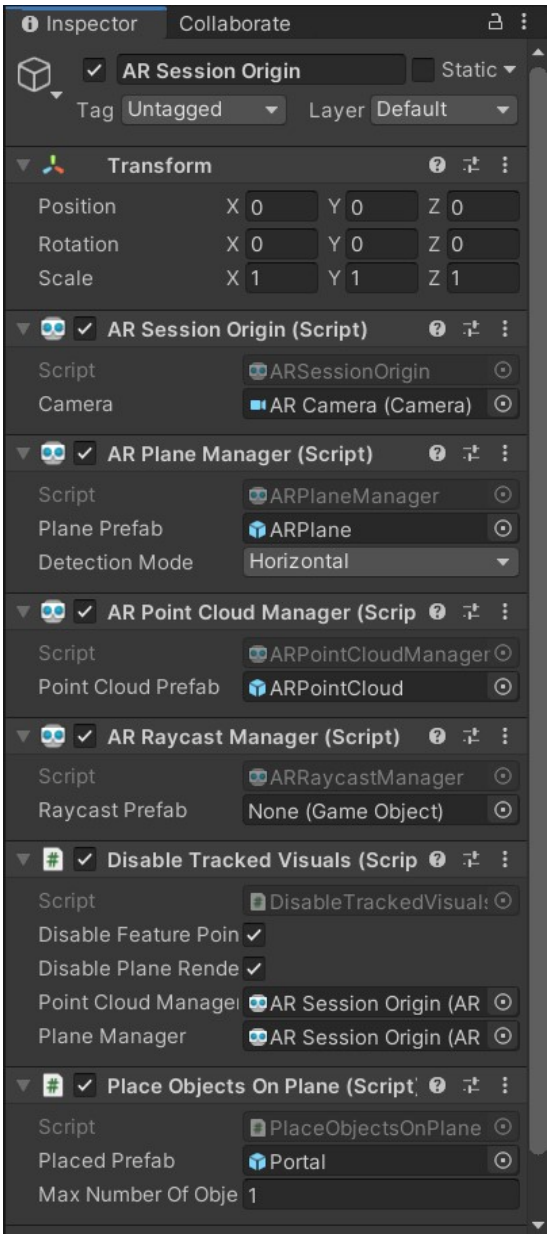
Στη συνέχεια αφού εισήχθησαν τα κατάλληλα πακέτα στο Unity, προστέθηκαν στο Hierarchy Panel τα κατάλληλα αντικείμενα. Τα αντικείμενα αυτά είναι κοινά, καθώς διαφέρει μόνο το νέφος σημείων, θα συμπεριληφθούν στις σκηνές των δύο εφαρμογών και θα αποτελέσουν τα στοιχεία εκείνα που θα συνθέσουν τις τελικές εφαρμογές, όπως φαίνεται στην παρακάτω εικόνα.




Σχημα 4.3: Αντικείμενα (Hierarchy Panel)

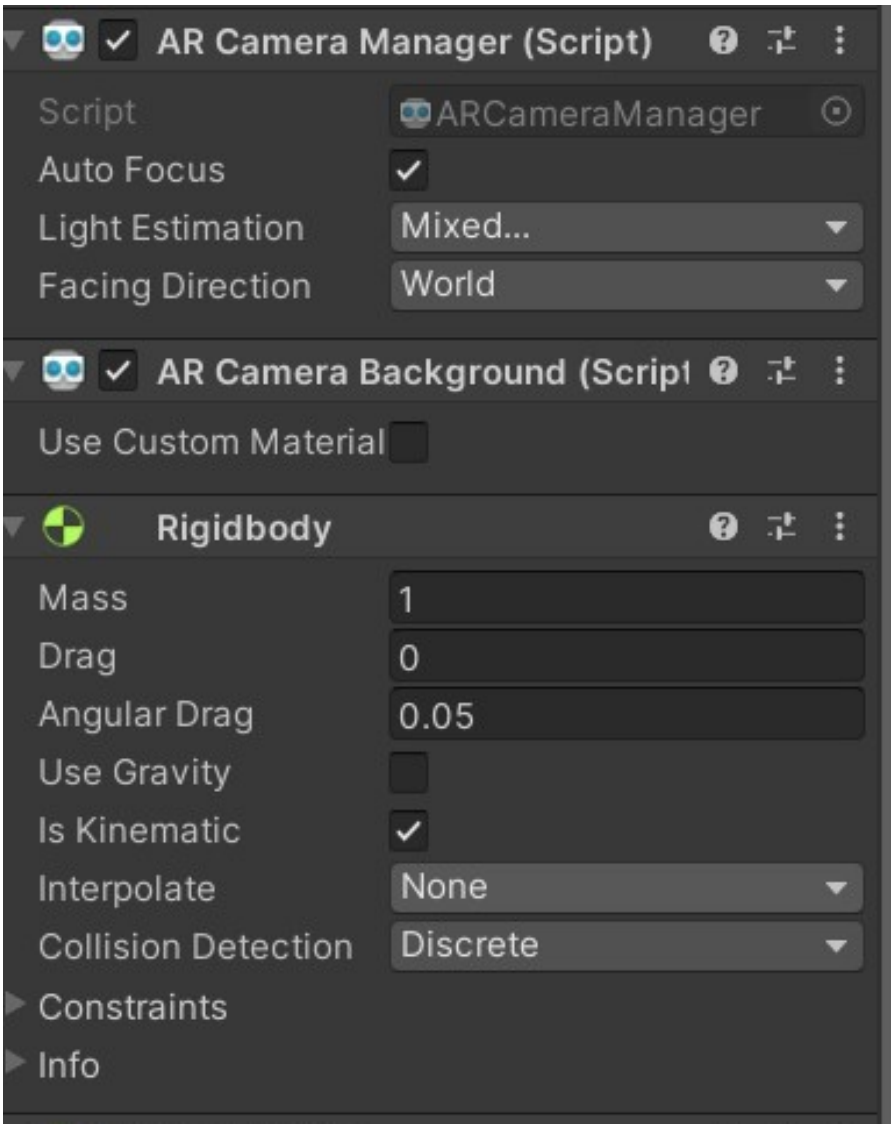
Το AR Session Origin είναι το βασικό τμήμα για μια εφαρμογή AR, καθώς εδώ περιλαμβάνεται, τόσο η AR Camera, όσο και το τρισδιάστατο μοντέλο (Prefab), το οποίο θα προβληθεί στο φυσικό περιβάλλον, μέσω της εφαρμογής. Στο κομμάτι αυτό, όπως φαίνεται και στην παρακάτω εικόνα, περιλαμβάνονται όλα εκείνα τα AR στοιχεία, τα οποία και είναι απαραίτητα για την ομαλή λειτουργία της εφαρμογής. Αρχικά αναφέρεται το AR Plane Manager, το οποίο είναι υπεύθυνο για τον εντοπισμό του επιπέδου, στο οποίο θα προβληθεί το αντικείμενο. Στην πραγματικότητα, το AR Plane είναι μια επίπεδη επιφάνεια, της οποίας η προβολή εξαρτάται από την πόζα, τις διαστάσεις και τα οριακά σημεία. Η ανίχνευση του Plane μπορεί να γίνει σε οριζόντια επιφάνεια, σε κάθετη και σε συνδυασμό αυτών. Στις εφαρμογές της παρούσας εργασίας ο εντοπισμός γίνεται μόνο σε οριζόντια επιφάνεια. Στη συνέχεια μέσω του AR Point Cloud Manager, γίνεται εφικτή η δημιουργία και προβολή του Νέφους Σημείων, μέσω διάφορων χαρακτηριστικών σημείων (feature points). Τα σημεία αυτά δεν είναι τυχαία σημεία, αλλά μοναδικά και βοηθούν τη συσκευή, ώστε να προσδιορίσει, με όσο το δυνατόν μεγαλύτερη ακρίβεια τη θέση του νέφους στο πραγματικό κόσμο. Επόμενο στη λίστα είναι το AR Raycast Manager. Ο διαχειριστής αυτός είναι υπεύθυνος για τη λειτουργία λήψης ακτίνας (raycasting). Η λειτουργία του Raycasting είναι ουσιαστικά η εκπομπή μιας ακτίνας από ένα σημείο σε μια κατεύθυνση μέγιστου μήκους μέχρι να συναντήσει κάποιο αντικείμενο που υπάρχει στη σκηνή του Unity.

Ακολουθούν δύο τμήματα κώδικα (scripts) με ονομασίες Disabled Tracked Visuals και Place Objects on Plane, τα οποία αναλύονται στο Κεφ. 4.1.3.




Σχημα 4.4: AR Session Origin Components

Στο AR Session Origin παρατηρείται ως «παιδί» του (child) η AR Camera, η οποία αποτελείται από τα στοιχεία που φαίνονται στην παρακάτω εικόνα.




Σχημα 4.5: AR Camera

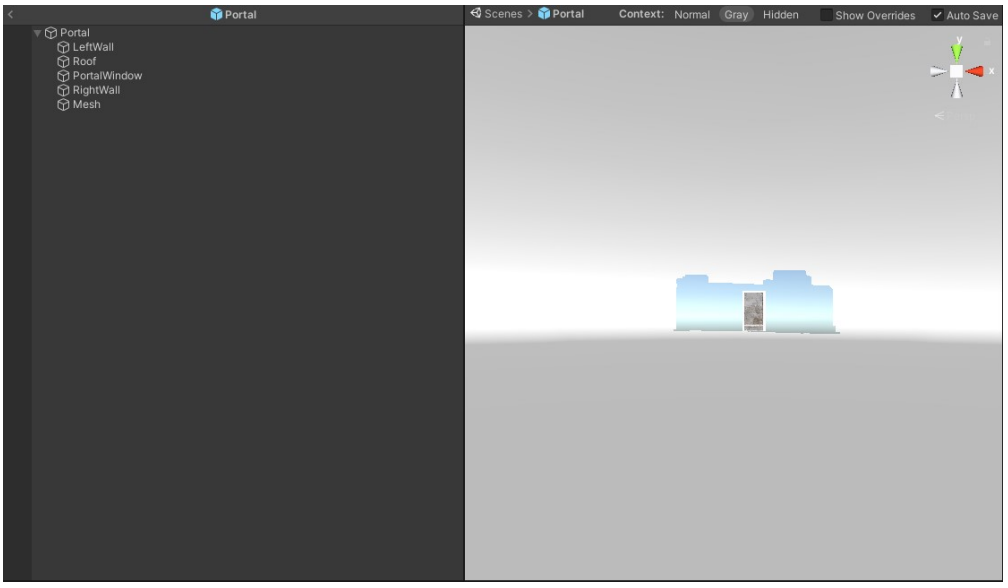
Αρχικά το AR Camera Manager είναι από τα βασικά στοιχεία (components), καθώς είναι ο διαχειριστής αυτός που ενεργοποιεί τις λειτουργίες της AR Cameras, όπως αυτές της λειτουργίας εκτίμησης του φωτός. Επόμενο στη λίστα είναι το AR Camera Background, το οποίο επιτρέπει στην κάμερα κατά τη διάρκεια λειτουργίας της εφαρμογής να παρουσιάζει πίσω από το μοντέλο το τμήμα του πραγματικού κόσμου, όπως αυτό αποτυπώνεται από την κάμερα.

Το Rigidbody, που ακολουθεί, είναι το κομμάτι (component) αυτό, το οποίο κάνει τα αντικείμενα, όπως και την AR Camera, να ενεργούν σύμφωνα με τους κανόνες της Φυσικής. Αυτό έχει σαν συνέπεια όλα τα αντικείμενα που έχουν το component αυτό, να κινούνται με ρεαλιστικό τρόπο. Μέσω αυτού και με τη χρήση κάποιων script μπορούν τα αντικείμενα να αλληλεπιδρούν μεταξύ τους, είτε να προστεθούν επιπλέον δυνάμεις σε αυτά. Στη συνέχεια στο Hierarchy Panel βρίσκεται το EventSystem, το οποίο, λειτουργεί συνδυαστικά με μια σειρά από ενότητες (modules), και

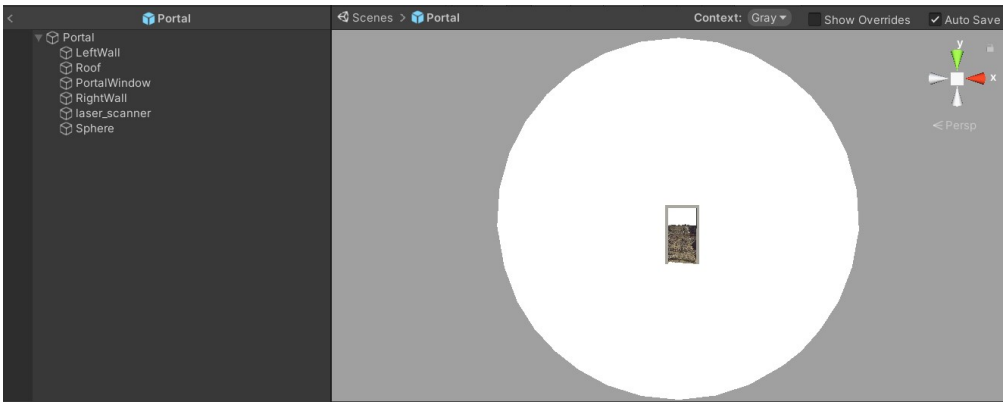
διαχειρίζεται τα συμβάντα που προκύπτουν στη σκηνή του Unity. Γενικά σε κάθε σκηνή περιέχεται ένα μόνο EventSystem.

Το αμέσως επόμενο αντικείμενο είναι το Screenspace UI. Το αντικείμενο αυτό είναι ουσιαστικά ένας καμβάς, τον οποίο μπορεί ο χρήστης να δημιουργήσει μέσα από το Unity. Ο καμβάς αυτός περιέχει όλα τα στοιχεία διεπαφής, τα οποία και στην διάταξη πρέπει να είναι «παιδιά» (children) του, όπως φαίνεται στην Εικόνα 4.3. Ουσιαστικά το αντικείμενο αυτό δίνει οδηγίες στο χρήστη για το πώς πρέπει να χρησιμοποιήσει την εφαρμογή, δηλαδή πού και πότε να ακουμπήσει την οθόνη για να εμφανίσει το αντικείμενο που θέλει, να μετατοπίσει το κινητό πιο αργά, στην περίπτωση που ψάχνει υφές (textures) κ.α. Το TopOfScreenAnchor δηλώνει ότι το κείμενο (text) που ακολουθεί (InstructionText) θα βρίσκεται στο πάνω μέρος της οθόνης της συσκευής, ενώ το BottomOfScreenAnchor δηλώνει ότι το κείμενο (text) που ακολουθεί (ReasonText) θα βρίσκεται στο κάτω μέρος της οθόνης της συσκευής. Ο καμβάς γενικότερα κατά τη λειτουργία του χρησιμοποιεί το EventSystem που αναφέρθηκε προηγουμένως.

Τέλος αυτό που βρίσκεται στο Hierarchy Panel είναι το μοντέλο. Τόσο στην πρώτη, όσο και στη δεύτερη εφαρμογή κατασκευάστηκε μία AR Portal, από την οποία όταν διέρχεται ο χρήστης, δημιουργείται ο εικονικός χώρος και στη συγκεκριμένη περίπτωση τμήμα της σκηνής του αρχαίου θεάτρου της Λάρισας και ο χώρος ανασκαφών της Τούμπας, αντίστοιχα. Τα μοντέλα αυτά απεικονίζονται παρακάτω.




Σχημα 4.6: Portal - Prefab - Larisa

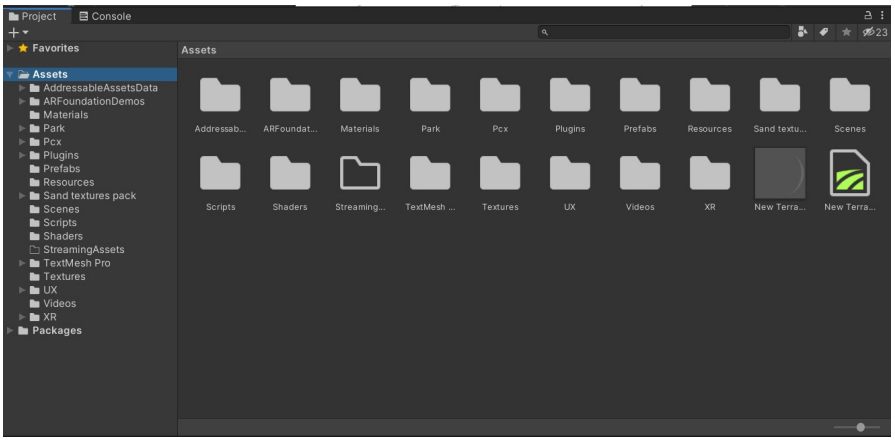



Σχημα 4.7: Portal - Prefab - Toumba

Τα μοντέλα με ονομασία Portal αποτελούνται από τα δοκάρια με ονομασίες LeftWall, RightWall, Roof και από ένα plane ανάμεσα σε αυτά με την ονομασία PortalWindow. Επίσης περιλαμβάνονται σε αυτά και τα νέφη σημείων, όπως προεκυψαν από τις μετρήσεις με χρήση drone και laser scanner, αντίστοιχα. Το δεύτερο prefab περιέχρι ένα τρισδιάστατο μοντέλο σφαίρας (Sphere), στο οποίο τοποθετήθηκε ένα Video Player Component, ώστε όταν ξεκινάει η εφαρμογή να εμφανίζεται γύρω από το νέφος σημείων ο ουρανός, ενώ στο πρώτο εμφανίζεται το πραγματικό περιβάλλον στο οποίο βρίσκεται ο χρήστης. Αφού τα μοντέλα κατασκευάστηκαν στο περιβάλλον του Unity στη συνέχεια με ένα drag & drop στον υποφάκελο Prefabs, ο οποίος βρίσκεται στο φάκελο Assets, μετατράπηκαν σε prefabs. Το prefab είναι ένας τύπος αποθήκευσης του τρισδιάστατου μοντέλου, ώστε ο χρήστης να μπορεί να το επαναχρησιμοποιήσει, είτε να το εισάγει σε κάποιο άλλο project, χωρίς να χρειαστεί να το φτιάξει από την αρχή.

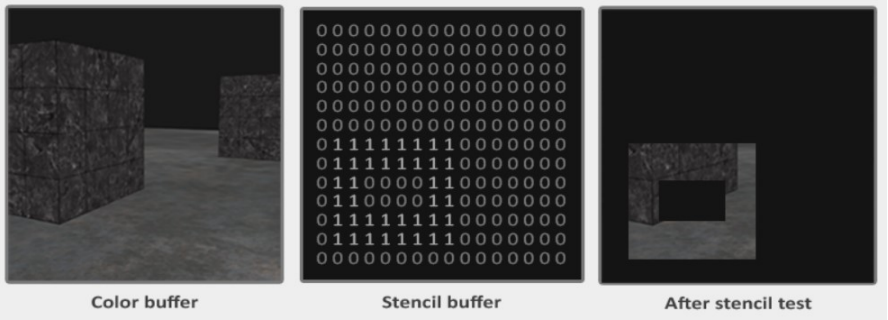
Ο φάκελος Assets, που αναφέρθηκε παραπάνω, είναι ο φάκελος εκείνος που περιέχει όλα τα απαραίτητα «συστατικά» για να κατασκευαστεί μια εφαρμογή ή παιχνίδι στο Unity, όπως τα Materials, Shaders, Packages, Scenes κ.α..

Όσον αφορά τα Materials χρησιμοποιήθηκε το Default Point και στις δύο εφαρμογές, το οποίο προήλθε από το custom πακέτο (Pcx), και το οποίο είχε δύο shaders (point, disk), ώστε να «ντύνει» το εν λόγω υλικό (material). Στην πρώτη εφαρμογή (Θέατρο Λάρισας) χρησιμοποιήθηκε το Disk Shader, καθώς απέδιδε το νέφος σημείων καλύτερα. Αντίθετα στη δεύτερη εφαρμογή χρησιμοποιήθηκε το Point Shader, καθώς απέδιδε καλύτερα το νέφος σημείων και με καλύτερες ακρίβειες στις επιφάνειες του. Επίσης χρησιμοποιήθηκε ακόμα ένα material, το οποίο κατασκευάστηκε για το PortalWindow και έχει την ίδια ονομασία. Το material αυτό «ντύθηκε» με το PortalWindow Shader. Στο φάκελο Materials βρίσκεται και το material Video360, το οποίο αφορά τη δεύτερη εφαρμογή (χώρος ανασκαφών Τούμπας), και αναφέρεται στο Video Player Component, και «ντύθηκε» με το ανάλογο Shader. Σημειώνεται πως, ό,τι διαγράφεται από το φάκελο Assets, αυτομάτως διαγράφεται και από το Project και πρέπει να σχεδιαστεί, αν πρόκειται για μοντέλο, ή να επανατοποθετηθεί, αν είναι κάποιο άλλο στοιχείο.




Σχημα 4.8: Assets Folder

Στο PortalWindow Shader, το οποίο είναι και αυτό που μπαίνει στην πόρτα, αρχικά επιλέγεται το ZWrite και το CULL να είναι OFF, καθώς και το ColorMask να είναι 0, καθώς πρέπει η πόρτα να είναι εντελώς διάφανη, και να μη φαίνεται το 3D Plane το οποίο χρησιμοποιήθηκε για να κατασκευαστεί.. Στη συνέχεια χρησιμοποιήθηκε η έννοια του Stencil Buffer για να κάνει το Plane αυτό ικανό να απεικονίζει το τι βρίσκεται μέσα στο χώρο. Πιο συγκεκριμένα το Stencil Buffer περιέχει 8 bits ανά τιμή Stencil, δηλαδή ένα σύνολο 256 διαφορετικών τιμών Stencil ανά Pixel. Αυτές οι τιμές Stencil μπορούν να καθοριστούν από το χρήστη ανάλογα με τις απαιτήσεις του. Ακολουθεί ένα παράδειγμα που επεξηγεί την λογική χρήσης του Stencil Buffer. Στην παρούσα εργασία πρέπει να είναι ορατό μόνο το εσωτερικό κομμάτι της πόρτας, και όλα τα υπόλοιπα να κόβονται. Για αυτό το λόγο τέθηκε η τιμή 1 σε όλα τα pixels της πόρτας, και η τιμή 0 σε όλα τα υπόλοιπα pixels.

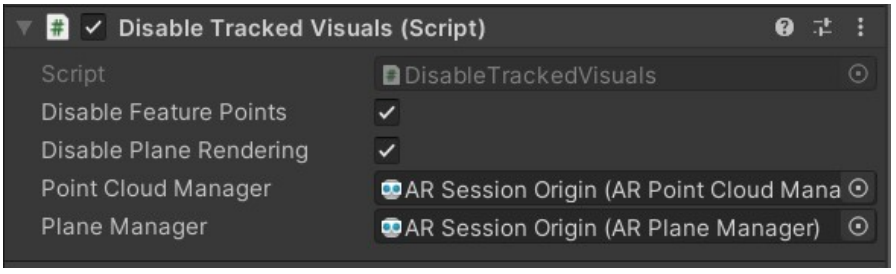



Σχημα 4.9: Stencil Example [7]

### 4.1.3 Κώδικας

Για να επιτευχθούν οι διάφορες λειτουργίες στις δύο εφαρμογές έγινε η χρήση των παρακάτω τμημάτων κώδικα (Scripts).

Αρχικά το πρώτο τμήμα κώδικα που συντάχθηκε ήταν το DisableTrackedVisuals script. Το εν λόγω script υλοποιεί την απενεργοποίηση των Points και Planes, αφού πρώτα επιλεχθεί μέσω της οθόνης του κινητού ή του tablet η επιθυμητή θέση, στην οποία θα τοποθετηθεί το αντικείμενό. Στο πρώτο κομμάτι του script δημιουργούνται οι επιλογές απενεργοποίησης των Feature Point και του Plane Rendering, με τη χρήση public Boolean μεταβλητών, οι οποίες μπορούν να πάρουν τις λογικές τιμές (values) true και false. Στη συνέχεια ορίζονται οι μεταβλητές Point Cloud Manager και Plane Manager, σέρνοντας το αντικείμενο AR Session Origin από το Hierarchy Panel, όπου ήδη περιέχονται οι διαχειριστές αυτοί. Οι τιμές των διαχειριστών εξαρτώνται από τις τιμές των παραπάνω μεταβλητών. Πιο συγκεκριμένα, όταν ξεκινήσει η εφαρμογή και οι τιμές Disable Feature Points και Disable Plane Rendering είναι επιλεγμένες, τότε οι Point Cloud Manager και Plane Manager αποκτούν τιμή false μόλις τοποθετηθεί το αντικείμενο, και απενεργοποιούνται.




Σχημα 4.10: Disable Tracked Visuals

Επόμενο script είναι το PlaceObjectsOnPlane, το οποίο ευθύνεται για την τοποθέτηση του εκάστοτε μοντέλου στο χώρο. Αρχικά δημιουργείται ένα πλαίσιο με την ονομασία Placed Prefab, στο οποίο τοποθετείται το prefab που θα εμφανιστεί στο άγγιγμα της οθόνης (spawn) και το οποίο το σέρνουμε στο πλαίσιο αυτό, ώστε να το αντιληφθεί το πρόγραμμα. Στο κάτω πλαίσιο ορίζουμε τον μέγιστο αριθμό μοντέλων που θα τοποθετηθούν, όπου στην κάθε εφαρμογή είναι ένα. Η βασική λειτουργία του script είναι ότι, με το άγγιγμα στην οθόνη και με την προϋπόθεση ότι ισχύει η ανισότητα:

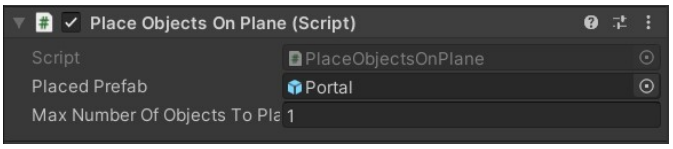
$$m\_NumberOfPlacedObjects < m\_MaxNumberOfObjectsToPlace$$

τότε το αντικείμενο εμφανίζεται στο Plane στη θέση που επέλεξε ο χρήστης, και με στροφή ανάλογη με αυτή της κάμερας τη στιγμή της επιλογής.

Αλλιώς η ανισότητα που ισχύει είναι η ακόλουθη:

$$m\_NumberOfPlacedObjects > m\_MaxNumberOfObjectsToPlace$$

τότε απλά αλλάζει η θέση και η στροφή του αντικειμένου που έχει ήδη τοποθετηθεί.




Σχημα 4.11: Place Objects On Plane

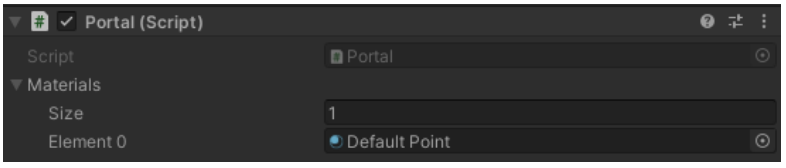
Τρίτο και πολύ βασικό script είναι το Portal. Το script αυτό εισήχθη πάνω στο 3D Plane με ονομασία PortalWindow που έχει ορισθεί ως η είσοδος της πόρτας. Για να κατασκευαστεί ο κώδικας χρησιμοποιήθηκε η λειτουργία σύγκρισης του Stencil, όπως φαίνεται και από την παρακάτω εικόνα. Στην ουσία μέσα σε όλα τα Shaders που χρησιμοποιήθηκαν στα projects γίνεται αναφορά του Stencil, το οποίο έχει τη τιμή Equal. Αρχικά, όταν ξεκινάει η εφαρμογή, δίνεται σε όλα τα materials η τιμή Equal, δηλαδή τα materials αυτά να είναι ορατά μόνο μέσα από το Portal. Στη συνέχεια ανάλογα με τη θέση της κάμερας στο χώρο, και συγκεκριμένα αν είναι έξω από το χώρο, για την ακρίβεια μέχρι λίγο πριν βρεθεί κάτω από τα δοκάρια της πόρτας, ορίζεται τα αντικείμενα να είναι ορατά μόνο μέσω της πόρτας. Όταν όμως τη στιγμή που η κάμερα βρεθεί ακριβώς κάτω από τα δοκάρια τότε η τιμή αλλάζει και από Equal γίνεται NotEqual, δηλαδή πλέον ο χώρος ανοίγεται και μπορεί ο χρήστης να δει τα πάντα σαν να είναι πραγματικά μέσα σε ένα δωμάτιο. Στο πλαίσιο Materials, στην πρώτη εφαρμογή, επιλέχθηκε ο αριθμός ένα, καθώς μας ενδιέφερε μόνο το στοιχείο με το material Default Point, δηλαδή το νέφος σημείων, ενώ στη δεύτερη εφαρμογή βάλαμε το δύο, καθώς τα αντικείμενα, τα οποία θέλαμε να έχουν τις παραπάνω δυνατότητες ήταν δύο με δύο διαφορετικά materials, δηλαδή το νέφος σημείων και η σφαίρα.

**Comparison Function**

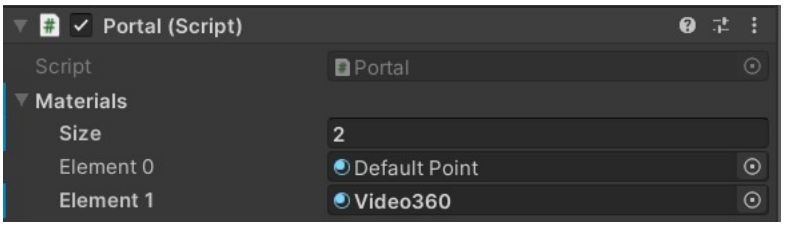
Comparison function is one of the following:

| | |
|---|---|
| **Greater** | Only render pixels whose reference value is greater than the value in the buffer. |
| **GEqual** | Only render pixels whose reference value is greater than or equal to the value in the buffer. |
| **Less** | Only render pixels whose reference value is less than the value in the buffer. |
| **LEqual** | Only render pixels whose reference value is less than or equal to the value in the buffer. |
| **Equal** | Only render pixels whose reference value equals the value in the buffer. |
| **NotEqual** | Only render pixels whose reference value differs from the value in the buffer. |
| **Always** | Make the stencil test always pass. |
| **Never** | Make the stencil test always fail. |

Σχημα 4.12: Stencil




Σχημα 4.13: Portal Script - Αρχαίο Θέατρο Λάρισας




Σχημα 4.14: Portal Script - Χώρος Ανασκαφών Τούμπας

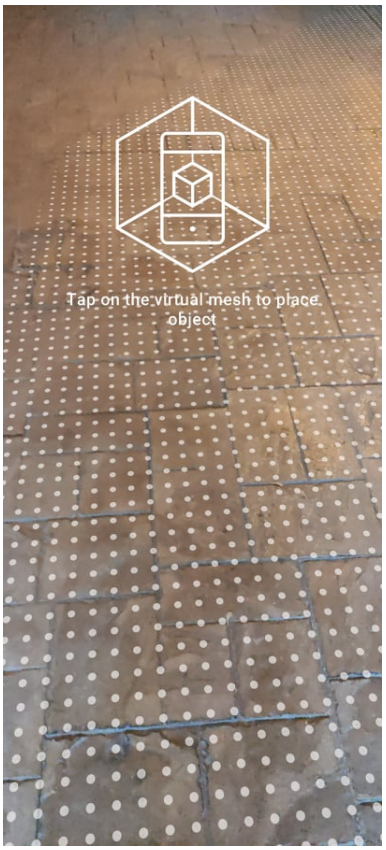
Ο πηγαίος κώδικας που χρησιμοποιήθηκε παρατίθεται στο Παράρτημα.

## 4.2 Στιγμιότυπα Εφαρμογών

Στο κεφάλαιο αυτό παρουσιάζονται στιγμιότυπα από τις δύο εφαρμογές εξωτερικά και εσωτερικά των AR Portals.

Στις παρακάτω εικόνες παρουσιάζονται οι διαδικασίες ανίχνευσης του επιπέδου και της τοποθέτησης του αντικειμένου στο επίπεδο αυτό.


(α′) Plane Detection

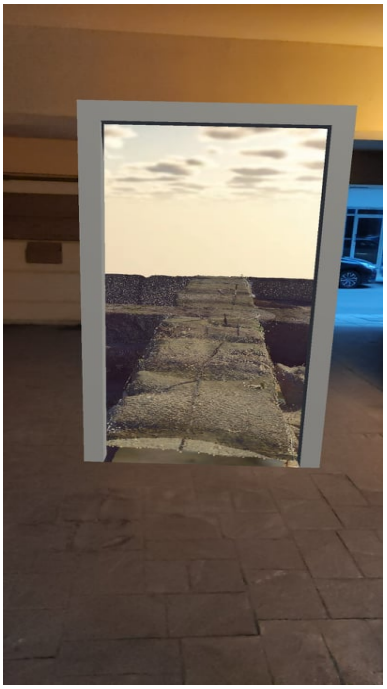



(β′) Place Object

Σχήμα 4.15: Διαδικασία Εντοπισμού Επιπέδου

Έπειτα απεικονίζονται οι δύο πόρτες στο επίπεδο και τα επόμενα στιγμιότυπα αφορούν το εξωτερικό και το εσωτερικό τους περιβάλλον.

(α′) Λάρισα AR Portal



(β′) Τούμπα AR Portal

Σχημα 4.16: Εμφάνιση Αντικειμένου στο επίπεδο

(α′) Λάρισα AR Portal - Εξωτερικό

(β′) Τούμπα AR Portal - Εξωτερικό

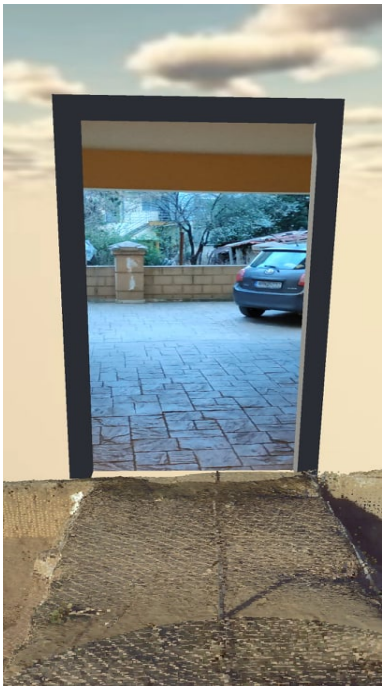
Σχημα 4.17: Πλάγια θέαση των ARPortals

(α΄) Λάρισα - AR Portal - Εσωτερικό

(β΄) Τούμπα - AR Portal - Εσωτερικό

Σχημα 4.18: Περιήγηση στο εσωτερικό των ARPortals

(α΄) Λάρισα - AR Portal - Εσωτερικό



(β΄) Τούμπα - AR Portal - Εσωτερικό

Σχημα 4.19: Θέαση εξωτερικού περιβάλλοντος μέσω των ARPortals

# Κεφάλαιο 5

# Συμπεράσματα

Η αποτύπωση του μοντέλου στην πρώτη εφαρμογή (Αρχαίο Θέατρο Λάρισας) έγινε με τη χρήση drone (Phantom 4 Pro), ενώ στη δεύτερη εφαρμογή έγινε με τη χρήση laser scanner (Faro Focus 3D X 1200D) και με κάμερα (Canon EOS 1200D), από το Εργαστήριο Φωτογραμμετρίας και Τηλεπισκόπησης του ΚΦΧ του ΤΑΤΜ, ΑΠΘ. Για την επεξεργασία των δεδομένων και τη δημιουργία του τρισδιάστατου μοντέλου στην πρώτη εφαρμογή χρησιμοποιήθηκε το λογισμικό του Agisoft Metashape. Για τη δημιουργία του μοντέλου στη δεύτερη εφαρμογή από τις μετρήσεις του laser scanner χρησιμοποιήθηκε το Scene της Faro, το Geomagic Studio και το PolyWorks Inspector. Το νέφος σημείων που προέκυψε από τη φωτογραφική μηχανή δεν αξιοποιήθηκε, καθώς το μέγεθός του ήταν περίπου 2GB και δε μπορούσε να τρέξει ομαλά η εφαρμογή στη συσκευή Android (Xiaomi Redmi Note 8T), που χρησιμοποιήθηκε.

Έπειτα χρησιμοποιήθηκε το λογισμικό Unity στο οποίο εισήχθησαν τα κατάλληλα πακέτα και παράμετροι, που βοήθησαν στην επίλυση προβλημάτων συμβατότητας. Τα προβλήματα συμβατότητας αφορούσαν κατά κύριο λόγο τύπους δεδομένων που εξήχθησαν από τα λογισμικά που αναφέρθηκαν παραπάνω με το Unity. Το Unity δεν υποστηρίζει την εισαγωγή νέφους σημείων επομένως εγινε χρήση του πακέτου Pcx για την εισαγωγή τους. Στην πρώτη εφαρμογή, για την απόδοση του νέφους σημείων στο Unity, επιλέχθηκε το Disk Shader του custom πακέτου με ονομασία Pcx, καθώς απέδιδε καλύτερα το μοντέλο, ενώ στη δεύτερη εφαρμογή επιλέχθηκε το Point Shader του ίδιου πακέτου. Τέλος έγινε εξαγωγή της εφαρμογής με μορφή .apk, ώστε αφού μεταφερθεί στη συμβατή με το Unity, κινητή συσκευή Android, ή tablet, να μπορεί ο χρήστης να περιηγηθεί στους αντίστοιχους χώρους.

## 5.1 Πεδία Εφαρμογής

Η εφαρμογή που προέκυψε ως αποτέλεσμα της παρούσας διπλωματικής εργασίας, μπορεί να υιοθετηθεί από αρχαιολογικούς χώρους, μουσεία, πινακοθήκες κ.α, ώστε να δίνεται η δυνατότητα στον κάθε ενδιαφερόμενο να περιπλανηθεί στους χώρους αυτούς, χωρίς να βρίσκεται απαραίτητα εκεί σε πραγματικό χρόνο. Η περιήγηση συνοδευόμενη με διαδραστικά πολυμέσα, όπως βίντεο, εικόνες, ηχητικό υλικό, δίνει στο χρήστη μια πλήρη αίσθηση του χώρου, προκαλεί το ενδιαφέρον του και παρέχει όλες τις απαραίτητες πληροφορίες που χρειάζεται για να έχει μία ολοκληρωμένη εμπειρία.

Το Unity αποτελεί ένα ολοκληρωμένο λογισμικό σχεδίασης παιχνιδιών ή εφαρμογών σε Επαυξημένη Πραγματικότητα, Εικονική Πραγματικότητα και σε Μικτή Πραγματικότητα, που θα διαδραματίσει ιδιαίτερο ρόλο στο μέλλον των εφαρμογών αυτών. Η ανάπτυξη της τεχνολογίας, τα επόμενα χρόνια θα φέρει κινητές συσκευές smartphones και tablets, οι οποίες αφενός θα υποστηρίζουν εφαρμογές με μεγαλύτερο όγκο δεδομένων, αφετέρου θα διαθέτουν μεγαλύτερη υπολογιστική ισχύ. Με αυτό τον τρόπο η εφαρμογή που κατασκευάστηκε στα πλαίσια της παρούσας διπλωματικής εργασίας, δίνει τη δυνατότητα σε χρήστες από όλον τον κόσμο να επισκεφτούν από το σπίτι τους πολιτισμικά μνημεία άλλων χωρών και άλλους χώρους ενδιαφέροντος, ειδικότερα αν λάβει κανείς υπόψη του την νέα πραγματικότητα η οποία έχει διαμορφωθεί λόγω της πανδημίας.

## 5.2 Προτάσεις για Μελλοντική Εργασία

Υφίστανται ορισμένες βελτιώσεις και προσθήκες, ώστε η εφαρμογή να γίνει ακόμα πιο ενδιαφέρουσα και χρήσιμη στον επισκέπτη. Αρχικά μία προσθήκη είναι η εισαγωγή σημείων ενδιαφέροντος (HotSpots), όπου ο επισκέπτης αγγίζοντας απλά το σημείο, στην οθόνη της συσκευής του, θα προβάλλεται ένα βίντεο ή εικόνα ή ακόμα και μικρότερα τρισδιάστατα μοντέλα, τα οποία θα μπορεί να τα επεξεργάζεται λεπτομερώς. Ακόμα για να παρουσιαστούν στον επισκέπτη τμήματα που δεν υπάρχουν πλέον, αλλά είναι γνωστή η αρχιτεκτονική τους, αρκεί να αγγίξει την οθόνη στο σημείο που ήταν το κτίριο και αυτό να προβάλλεται μπροστά του. Τέλος μια ακόμα λειτουργία που θα μπορούσε να προστεθεί στην εφαρμογή, αλλάζοντάς την από Επαυξημένη Πραγματικότητα σε Μικτή Πραγματικότητα, θα ήταν η χρήση ειδικών γυαλιών VR, όπου ενώ στην αρχή ο επισκέπτης θα βρίσκεται στον πραγματικό κόσμο, θα μεταφέρεται σε ένα εικονικό περιβάλλον βλέποντας τα γεγονότα να εξελίσσονται μπροστά του.

# Παράρτημα Α΄

# Πηγαίος Κώδικας

## Α΄.1 Scripts

### Α΄.1.1 Portal

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

public class Portal : MonoBehaviour
{
    public Material[] materials;

    // Start is called before the first frame update
    void Start()
    {
        foreach (var mat in materials)
        {
            mat.SetInt("stest", (int)CompareFunction.Equal);
        }
    }

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

```
}



private void OnTriggerStay(Collider collider)
{
    if (collider.tag != "MainCamera")
    {
        return;
    }


    Vector3 user_position = Camera.main.transform.position + Camera.main.
        ↪ transform.forward * Camera.main.nearClipPlane;
    Vector3 relativePosition = transform.InverseTransformPoint(user_position
        ↪ );


    //outside
    if (relativePosition.z < 0)
    {
        foreach (var mat in materials)
        {
            mat.SetInt("stest", (int)CompareFunction.Equal);
        }

    }
    //inside
    else
    {
        foreach (var mat in materials)
        {
            mat.SetInt("stest", (int)CompareFunction.NotEqual);
        }


    }
```

```
    }


}
```

### A′.1.2 DisabledTrackedVisuals

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

public class DisableTrackedVisuals : MonoBehaviour
{
    [SerializeField]
    [Tooltip("Disables spawned feature points and the ARPointCloudManager")]
    bool m_DisableFeaturePoints;

    public bool disableFeaturePoints
    {
        get => m_DisableFeaturePoints;
        set => m_DisableFeaturePoints = value;
    }

    [SerializeField]
    [Tooltip("Disables spawned planes and ARPlaneManager")]
    bool m_DisablePlaneRendering;

    public bool disablePlaneRendering
    {
        get => m_DisablePlaneRendering;
        set => m_DisablePlaneRendering = value;
    }

    [SerializeField]
    ARPointCloudManager m_PointCloudManager;

    public ARPointCloudManager pointCloudManager
    {
        get => m_PointCloudManager;
        set => m_PointCloudManager = value;
    }
```

```
    [SerializeField]
    ARPlaneManager m_PlaneManager;

    public ARPlaneManager planeManager
    {
        get => m_PlaneManager;
        set => m_PlaneManager = value;
    }

    void OnEnable()
    {
        PlaceObjectsOnPlane.OnObjectPlaced += OnPlacedObject;
    }

    void OnDisable()
    {
        PlaceObjectsOnPlane.OnObjectPlaced -= OnPlacedObject;
    }

    void OnPlacedObject()
    {
        if (m_DisableFeaturePoints)
        {
            m_PointCloudManager.SetTrackablesActive(false);
            m_PointCloudManager.enabled = false;
        }

        if (m_DisablePlaneRendering)
        {
            m_PlaneManager.SetTrackablesActive(false);
            m_PlaneManager.enabled = false;
        }
    }
}
```

### A′.1.3 PlaceObjectsOnPlane

```
using System;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.XR.ARFoundation;
using UnityEngine.XR.ARSubsystems;

[RequireComponent(typeof(ARRaycastManager))]
public class PlaceObjectsOnPlane : MonoBehaviour
{
    [SerializeField]
    [Tooltip("Instantiates this prefab on a plane at the touch location.")]
    GameObject m_PlacedPrefab;

    /// <summary>
    /// The prefab to instantiate on touch.
    /// </summary>
    public GameObject placedPrefab
    {
        get { return m_PlacedPrefab; }
        set { m_PlacedPrefab = value; }
    }

    /// <summary>
    /// The object instantiated as a result of a successful raycast
        ↪ intersection with a plane.
    /// </summary>
    public GameObject spawnedObject { get; private set; }

    /// <summary>
    /// Invoked whenever an object is placed in on a plane.
    /// </summary>
    public static event Action OnObjectPlaced;

    ARRaycastManager m_RaycastManager;

    static List<ARRaycastHit> s_Hits = new List<ARRaycastHit>();
```

```
[SerializeField]
int m_MaxNumberOfObjectsToPlace = 1;

int m_NumberOfPlacedObjects = 0;

void Awake()
{
    m_RaycastManager = GetComponent<ARRaycastManager>();
}

void Update()
{
    if (Input.touchCount > 0)
    {
        Touch touch = Input.GetTouch(0);

        if (touch.phase == TouchPhase.Began)
        {
            if (m_RaycastManager.Raycast(touch.position, s_Hits,
                ↪ TrackableType.PlaneWithinPolygon))
            {
                Pose hitPose = s_Hits[0].pose;

                if (m_NumberOfPlacedObjects < m_MaxNumberOfObjectsToPlace)
                {
                    spawnedObject = Instantiate(m_PlacedPrefab, hitPose.
                        ↪ position, hitPose.rotation);

                    m_NumberOfPlacedObjects++;
                }
                else
                {
                    spawnedObject.transform.SetPositionAndRotation(hitPose.
                        ↪ position, hitPose.rotation);
                }

                OnObjectPlaced?.Invoke();
```

```
            }
        }
    }
}
}
```

### A′.1.4 PlaneController

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

public class PlaneController : MonoBehaviour
{
    ARPlaneManager m_ARPlaneManager;


    public Text buttonText;


    private void Awake()
    {
        m_ARPlaneManager = GetComponent<ARPlaneManager>();
    }

    // Start is called before the first frame update
    void Start()
    {

    }

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

```
    }



    public void TogglePlaneDetectionAndVisibility()
    {
        m_ARPlaneManager.enabled = !m_ARPlaneManager.enabled;

        if (m_ARPlaneManager.enabled)
        {
            SetAllPlanesActiveOrDeactive(true);
            GetComponent<PortalPlacer>().enabled = true;
            buttonText.text = "Disable Plane Detection And Hide Existing";

        }else
        {
            SetAllPlanesActiveOrDeactive(false);
            GetComponent<PortalPlacer>().enabled = false;
            buttonText.text = "Enable Plane Detection And Show Existing";
        }
    }


    void SetAllPlanesActiveOrDeactive(bool value)
    {
        {
            foreach (var plane in m_ARPlaneManager.trackables)
            {
                plane.gameObject.SetActive(value);
            }
        }


    }

}
```

### A′.1.5 PortalPlacer

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Experimental.XR;
using UnityEngine.XR.ARFoundation;

public class PortalPlacer : MonoBehaviour
{
    ARRaycastManager m_ARRaycastManager;


    List<ARRaycastHit> raycast_hits = new List<ARRaycastHit>();

    //This is the prefab to be instantiated
    public GameObject PortalPrefab;


    //This is the gameobject reference instantiated after a succesfull raycast
        ↪ intersection with a plane
    private GameObject spawnedPortal;



    private void Awake()
    {
        m_ARRaycastManager = GetComponent<ARRaycastManager>();

    }

    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
```

```
void Update()
{
    if (Input.touchCount > 0)
    {
        //Getting touch inputs
        Touch touch = Input.GetTouch(0);

        if (m_ARRaycastManager.Raycast(touch.position,raycast_hits,
            ↪ UnityEngine.XR.ARSubsystems.TrackableType.PlaneWithinPolygon)
            ↪ )
        {
            //Getting the pose of the hit
            Pose pose = raycast_hits[0].pose;

            if (spawnedPortal==null)
            {
                spawnedPortal = Instantiate(PortalPrefab, pose.position,
                    ↪ Quaternion.Euler(0, 0, 0));

                var rotationOfPortal = spawnedPortal.transform.rotation.
                    ↪ eulerAngles;
                spawnedPortal.transform.eulerAngles = new Vector3(
                    ↪ rotationOfPortal.x, Camera.main.transform.rotation.
                    ↪ eulerAngles.y, rotationOfPortal.z);


            }else
            {
                spawnedPortal.transform.position = pose.position;

                var rotationOfPortal = spawnedPortal.transform.rotation.
                    ↪ eulerAngles;
                spawnedPortal.transform.eulerAngles = new Vector3(
                    ↪ rotationOfPortal.x, Camera.main.transform.rotation.
                    ↪ eulerAngles.y, rotationOfPortal.z);
            }
        }
```

```
        }
    }
}
```

## A′.2 Shaders

### A′.2.1 PortalWindow

```
/*
 * To be attached to the mesh of the Portal
 * Should set render queue to before geometry, however this will
 * Result in objects closer than the portal to still be rendererd
 */



Shader "Custom/PortalWindow"
{
	SubShader
{

	//render objects behind the portal
	ZWrite off
	//absolutely transparent
	ColorMask 0
	//Bidirectional behaviour
	Cull off



	Stencil{
		Ref 1
		//set all pixels in the portal to 1
		Pass replace
	}


		Pass
		{
		}
	}
}
```

### A′.2.2 Video360

```
    Shader "Custom/Video360"
{
    Properties
    {
        _MainTex ("Texture", 2D) = "white" {}
    }
    SubShader
    {
        Tags { "RenderType"="Opaque" }
        LOD 100

                Cull front

                Stencil{
                        Ref 1
                        comp[stest]
                }

        Pass
        {
            CGPROGRAM
            #pragma vertex vert
            #pragma fragment frag
            // make fog work
            #pragma multi_compile_fog

            #include "UnityCG.cginc"

            struct appdata
            {
                float4 vertex : POSITION;
                float2 uv : TEXCOORD0;
            };

            struct v2f
            {
```

```
            float2 uv : TEXCOORD0;
            UNITY_FOG_COORDS(1)
            float4 vertex : SV_POSITION;
        };

        sampler2D _MainTex;
        float4 _MainTex_ST;

        v2f vert (appdata v)
        {
            v2f o;
            o.vertex = UnityObjectToClipPos(v.vertex);
            o.uv = TRANSFORM_TEX(v.uv, _MainTex);
            UNITY_TRANSFER_FOG(o,o.vertex);
            return o;
        }

        fixed4 frag (v2f i) : SV_Target
        {
            // sample the texture
                        float2 uv = float2(1. - i.uv.x,i.uv.y);

            fixed4 col = tex2D(_MainTex, i.uv);
            // apply fog
            UNITY_APPLY_FOG(i.fogCoord, col);
            return col;
        }
        ENDCG
    }
  }
}
```

### A′.2.3 Disk Shader

```
// Pcx - Point cloud importer & renderer for Unity
// https://github.com/keijiro/Pcx
```

```
Shader "Point Cloud/Disk"
{
    Properties
    {
        _Tint("Tint", Color) = (0.5, 0.5, 0.5, 1)
        _PointSize("Point Size", Float) = 0.05
        [Enum(Equal,3,NotEqual,6)] stest("Stencil Test",int) = 3

    }
    SubShader
    {
         Stencil {
        Ref 1
        comp[stest]
        }
        Tags { "RenderType"="Opaque" }
        Cull Off
        Pass
        {
            Tags { "LightMode"="ForwardBase" }
            CGPROGRAM
            #pragma vertex Vertex
            #pragma geometry Geometry
            #pragma fragment Fragment
            #pragma multi_compile_fog
            #pragma multi_compile _ UNITY_COLORSPACE_GAMMA
            #pragma multi_compile _ _COMPUTE_BUFFER
            #include "Disk.cginc"
            ENDCG
        }
        Pass
        {
            Tags { "LightMode"="ShadowCaster" }
            CGPROGRAM
            #pragma vertex Vertex
            #pragma geometry Geometry
            #pragma fragment Fragment
            #pragma multi_compile _ _COMPUTE_BUFFER
```

```
            #define PCX_SHADOW_CASTER 1
            #include "Disk.cginc"
            ENDCG

        }
    }
    CustomEditor "Pcx.DiskMaterialInspector"
}
```

### A′.2.4 Point Shader

```
// Pcx - Point cloud importer & renderer for Unity
// https://github.com/keijiro/Pcx

Shader "Point Cloud/Point"
{
    Properties
    {
        _Tint("Tint", Color) = (0.5, 0.5, 0.5, 1)
        _PointSize("Point Size", Float) = 0.007
        [Toggle] _Distance("Apply Distance", Float) = 1
                [Enum(Equal,3,NotEqual,6)] stest("Stencil Test",int) = 3
    }
    SubShader
    {
                Stencil {
        Ref 1
        comp[stest]
        }
        Tags { "RenderType"="Opaque" }
        Pass
        {
            CGPROGRAM

            #pragma vertex Vertex
            #pragma fragment Fragment

            #pragma multi_compile_fog
            #pragma multi_compile _ UNITY_COLORSPACE_GAMMA
            #pragma multi_compile _ _DISTANCE_ON
            #pragma multi_compile _ _COMPUTE_BUFFER

            #include "UnityCG.cginc"
            #include "Common.cginc"

            struct Attributes
            {
```

```
        float4 position : POSITION;
        half3 color : COLOR;
    };

    struct Varyings
    {
        float4 position : SV_Position;
        half3 color : COLOR;
        half psize : PSIZE;
        UNITY_FOG_COORDS(0)
    };

    half4 _Tint;
    float4x4 _Transform;
    half _PointSize;

#if _COMPUTE_BUFFER
    StructuredBuffer<float4> _PointBuffer;
#endif

#if _COMPUTE_BUFFER
    Varyings Vertex(uint vid : SV_VertexID)
#else
    Varyings Vertex(Attributes input)
#endif
    {
    #if _COMPUTE_BUFFER
        float4 pt = _PointBuffer[vid];
        float4 pos = mul(_Transform, float4(pt.xyz, 1));
        half3 col = PcxDecodeColor(asuint(pt.w));
    #else
        float4 pos = input.position;
        half3 col = input.color;
    #endif

    #ifdef UNITY_COLORSPACE_GAMMA
        col *= _Tint.rgb * 2;
    #else
```

```
            col *= LinearToGammaSpace(_Tint.rgb) * 2;
            col = GammaToLinearSpace(col);
        #endif

            Varyings o;
            o.position = UnityObjectToClipPos(pos);
            o.color = col;
        #ifdef _DISTANCE_ON
            o.psize = _PointSize / o.position.w * _ScreenParams.y;
        #else
            o.psize = _PointSize;
        #endif
            UNITY_TRANSFER_FOG(o, o.position);
            return o;
        }

        half4 Fragment(Varyings input) : SV_Target
        {
            half4 c = half4(input.color, _Tint.a);
            UNITY_APPLY_FOG(input.fogCoord, c);
            return c;
        }

        ENDCG
    }
  }
  CustomEditor "Pcx.PointMaterialInspector"
}
```

# Βιβλιογραφία


[1] 3d computer animation in the 60s. https://grayfire93.wordpress.com/2015/12/07/3d-computer-animation-in-the-60s/.

[2] Gouraud shading. https://www.pcmag.com/encyclopedia/term/gouraud-shading/.

[3] 40 year old 3d computer graphics pixar. http://socks-studio.com/2011/09/04/40-year-old-3d-computer-graphics-pixar-1972/.

[4] Founders series: industry legend jim blinn. https://www.fxguide.com/fxfeatured/founders-series-industry-legend-jim-blinn/.

[5] Sitescape launches a beta lidar scanning app for ipad leveraging arkit 4. https://www.spar3d.com/news/lidar/sitescape-launches-a-beta-lidar-scanning-app-for-ipad-leveraging-arkit-4/.

[6] Unity real time developement platform |2d, 3d, vr & ar engine. https://unity.com/.

[7] Stencil testing. https://learnopengl.com/Advanced-OpenGL/Stencil-testing.

[8] Henri gouraud biography. https://peoplepill.com/people/henri-gouraud/.

[9] Edwin catmull biography. https://www.computer.org/profiles/edwin-catmull.

[10] Frederic ira parke biography. https://peoplepill.com/people/frederic-parke.

[11] Jim blinn biography. https://www.jimblinn.com/biography/.

[12] Μπολωνάκης Κωνσταντίνος. Ανάπτυξη εφαρμογής προσομοίωσης εικονικού μουσείου σε τρισδιάστατο περιβάλλον με στοιχεία adventure game, 2012.

[13] Francesco Benvenuto, Andrea Bardini, Saul Mosanghini, and Alberto Battistutti. Augmented reality, mixed reality and virtual reality are enabling technologies for digital transformation, permitting to reshape operations.

[14] Rembrandy reality ar portal app. https://play.google.com/store/apps/details?id=nl.nn.mauritshuis.

[15] M.J. Westoby, J. Brasington, N.F. Glasser, M.J. Hambrey, and J.M. Reynolds. 'structure-from-motion' photogrammetry: A low-cost, effective tool for geoscience applications. *Geomorphology*, 179:300–314, 2012.

[16] M.W. Smith, J.L. Carrivick, and D.J. Quincey. Structure from motion photogrammetry in physical geography. *Progress in Physical Geography: Earth and Environment*, 40(2):247–275, 2016.

[17] Ar portal for real estate marketing. https://www.stambol.com/2018/07/30/3-ways-to-use-ar-for-real-estate-marketing/.

[18] Ar travel portal demo. https://metavrse.com/what-is-xr/.

[19] Eric Peckham. How unity built the world's most popular game engine. *Progress in Physical Geography: Earth and Environment*, 40(2), 2019.

[20] Ismail Buyuksalih, Serdar Bayburt, Gurcan Buyuksalih, AP Baskaraca, Hairi Karim, and Alias Abdul Rahman. 3d modelling and visualization based on the unity game engine–advantages and challenges. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, 4, 2017.

[21] Antonín Šmíd. Comparison of unity and unreal engine. 2017.

[22] Enoc Sanz-Ablanedo, Jim H. Chandler, José Ramón Rodríguez-Pérez, and Celestino Ordóñez. Accuracy of unmanned aerial vehicle (uav) and sfm photogrammetry survey as a function of the number and location of ground control points used. *Remote Sensing*, 10(10), 2018.

[23] S. Cucchiaro. 4d-sfm photogrammetry for monitoring sediment dynamics in a debris-flow catchment: Software testing and results comparison. *Progress in Physical Geography: Earth and Environment*, 40(2), 2018.

[24] Wenang Anurogo, Muhammad Zainuddin Lubis, Hanah Khoirunnisa, DSPA Hanafi, Fajar Rizki, Ganda Surya, and NA Dewanti. A simple aerial photogrammetric mapping system overview and image acquisition using unmanned aerial vehicles (uavs). *Geospatial Information*, 1(1), 2017.

[25] Noise filter. http://www.cloudcompare.org/doc/wiki/index.php?title=Noise_filter.

[26] Sor filter. http://www.cloudcompare.org/doc/wiki/index.php?title=SOR_filter.

[27] Coordinate reference systems gr. http://mapref.org/CoordinateReferenceSystemsGR.html.

[28] Map projections gr. http://mapref.org/MapProjectionsGR.html.

[29] Xiu quan Li, Zhu an Chen, Li ting Zhang, and Dan Jia. Construction and accuracy test of a 3d model of non-metric camera images using agisoft photoscan. *Procedia Environmental Sciences*, 36:184–190, 2016.

[30] Guenter Pomaska. Utilization of photosynth point clouds for 3d object reconstruction. In *Proceedings of the 22nd CIPA symposium, Kyoto, Japan*, pages 1–5, 2009.

[31] Unity user manual (2019.4 lts), monobehaviour. https://docs.unity3d.com/Manual/ExecutionOrder.html.

[32] Keijiro Takahashi. Pcx. https://github.com/keijiro/Pcx, 2020.