ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Διαδραστικός χάρτης επαυξημένης πραγματικότητας της Αλεξάνδρειας Πανεπιστημιούπολης του ΔΙΠΑΕ»

Του φοιτητή

Ασιήκκαλη Σταύρου

Αρ. Μητρώου: 2020201

Επιβλέπων

Κεραμόπουλος Ευκλείδης

Καθηγητής

Ημερομηνία 23/1/2026

Τίτλος Δ.Ε. Διαδραστικός χάρτης επαυξημένης πραγματικότητας της Αλεξάνδρειας Πανεπιστημιούπολης του ΔΙΠΑΕ

Κωδικός Δ.Ε.: 25271

Ονοματεπώνυμο φοιτητή/τών: Ασιήκκαλης Σταύρος

Ονοματεπώνυμο εισηγητή: Κεραμόπουλος Ευκλείδης

Ημερομηνία ανάληψης Δ.Ε.: 17/06/2025

Ημερομηνία περάτωσης Δ.Ε. 23/01/2026

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Πρόλογος

Η επιλογή του θέματος της διπλωματικής μου εργασίας δεν ήταν τυχαία αφού από τα πρώτα εξάμηνα των σπουδών μου, μου άρεσε η δημιουργία εφαρμογών που συνδυάζουν τεχνική γνώση και δημιουργικότητα. Φτιάχνοντας αυτή την εφαρμογή έκανα την έρευνά μου πάνω στην επαυξημένη πραγματικότητα η οποία αποτέλεσε για εμένα μια ευκαιρία να μάθω νέες τεχνολογίες και νέες δεξιότητες, καθώς εισάγει μια εντελώς διαφορετική προσέγγιση στην ψηφιακή αλληλεπίδραση και απαιτεί βαθύτερη κατανόηση. Ωστόσο, η παρούσα εργασία, μου έδωσε την ευκαιρία να εξερευνήσω μια διαφορετική πτυχή του Unity, της επαυξημένης πραγματικότητας, η οποία παρουσιάζει νέες προκλήσεις και απαιτεί διαφορετικές προσεγγίσεις στον σχεδιασμό και την τεχνική υλοποίηση. Επιπλέον το γεγονός ότι το θέμα της εργασίας συνδέεται με τον χώρο του ΔΙΠΑΕ αποτελεί ισχυρό κίνητρο. Η ανάπτυξη ενός διαδραστικού AR χάρτη για την Πανεπιστημιούπολη μπορεί να προσφέρει πρακτική αξία σε φοιτητές και προσωπικό, διευκολύνοντας τον προσανατολισμό και την πρόσβαση σε πληροφορίες μέσω μιας σύγχρονης και ευχάριστης ψηφιακής εμπειρίας. Στο πλαίσιο αυτό, η εργασία εστιάζει στην αξιοποίηση της εκτεταμένης πραγματικότητας για τη δημιουργία ενός λειτουργικού και χρήσιμου εργαλείου που ενισχύει τον τρόπο με τον οποίο ο χρήστης αντιλαμβάνεται και ενημερώνεται μέσω ψηφιακών αντικειμένων που προβάλλονται στον φυσικό χώρο της Πανεπιστημιούπολης.

Περίληψη

Η παρούσα διπλωματική εργασία ασχολείται με την ανάπτυξη μιας διαδραστικής εφαρμογής επαυξημένης πραγματικότητας (AR) για την Αλεξάνδρεια Πανεπιστημιούπολη του ΔΙΠΑΕ. Ο στόχος της εφαρμογής είναι να προσφέρει έναν σύγχρονο και αποτελεσματικό τρόπο ενημέρωσης, επιτρέποντας στους χρήστες να έχουν άμεση πρόσβαση σε πληροφορίες για τα μαθήματα που γίνονται στις αίθουσες, οι οποίες προβάλλονται στο πραγματικό περιβάλλον μέσω φορητών συσκευών.

Για την υλοποίηση της εφαρμογής αξιοποιήθηκαν το Unity και το AR Foundation, σε συνδυασμό με το ARCore της Google, προκειμένου να επιτευχθεί ανίχνευση του φυσικού χώρου και τοποθέτηση σταθερών AR Anchors. Επίσης, για το hosting των Anchors αξιοποιήθηκε το Google Cloud Platform και για την αποθήκευση των πληροφοριών και των ψηφιακών αντικειμένων χρησιμοποιήθηκε το Firebase της Google. Μέσω αυτών των σημείων αναφοράς αποδίδονται στο περιβάλλον τρισδιάστατα στοιχεία και πληροφορίες, τα οποία ενισχύουν την κατανόηση της χωρικής διάταξης της Πανεπιστημιούπολης.

Η εφαρμογή δοκιμάστηκε σε πραγματικές συνθήκες, όπου αξιολογήθηκαν η ακρίβεια των anchors, η σταθερότητα της AR απεικόνισης και η χρηστικότητα του συστήματος. Τα αποτελέσματα των δοκιμών δείχνουν ότι η AR μπορεί να αποτελέσει ένα χρήσιμο εργαλείο ενημέρωσης, προσφέροντας μια σύγχρονη και κατανοητή μορφή πλοήγησης στον χώρο.

Συνολικά, η εργασία αποδεικνύει την αξία της AR σε εκπαιδευτικά και πληροφοριακά περιβάλλοντα και θέτει τις βάσεις για μελλοντικές επεκτάσεις, όπως αλληλεπίδραση των χρηστών με τα AR μοντέλα, επιπλέον 3D περιεχόμενο και διασύνδεση με υπηρεσίες του Πανεπιστημίου, καθώς και σύστημα πλοήγησης στην Πανεπιστημιούπολη. Επίσης η προσέγγιση αυτή μπορεί να προσαρμοστεί και να εφαρμοστεί σε άλλα τμήματα ή πανεπιστήμια με κατάλληλο σχεδιασμό και παραμετροποίηση.

«Interactive AR Campus Map of International Hellenic University Alexandria Campus»

«Ashikkalis Stavros»

Abstract

This diploma thesis focuses on the development of an interactive Augmented Reality (AR) application for the Alexandria Campus of the International Hellenic University. The purpose of the application is to offer a modern and efficient way for students, staff, and visitors to navigate the campus and access information about its buildings and facilities through their mobile devices.

The implementation was carried out using Unity in combination with AR Foundation and Google ARCore, enabling real-time environment detection and placement of stable AR Anchors in the physical space. Additionally, Google Cloud Platform were employed for anchor hosting, and Firebase was used as the backend database for storing textual information and AR anchors. These anchors serve as reference point upon which digital elements and informational content are displayed. In addition, 3D models and visual assets were designed to enhance the user's understanding of the campus layout, creating a more immersive and intuitive experience.

The application was tested in real-world conditions to evaluate anchor accuracy, AR visual stability, and overall usability. The test results show that augmented reality can be useful tool for information delivery, providing a modern and clear way to present information in a physical environment.

Overall, the thesis highlights the potential of AR technologies in educational and informational environments. The approach presented in this work can be adapted and applied to other university campuses or institutions with appropriate customization, paving the way for future extensions such as step-by-step navigation, additional 3D content, and integration with university services.

Ευχαριστίες

Ευχαριστώ τον επιβλέποντα καθηγητή μου, Κ. Κεραμόπουλο Ευκλείδη, για την καθοδήγηση και την υποστήριξη του κατά την εκπόνηση της παρούσας διπλωματικής εργασίας. Ευχαριστώ επίσης τον υποψήφιο διδάκτορα Κ. Καζλάρη Γιώργο, για την συνεχή υποστήριξη και σημβουλή του.

Περιεχόμενα

Κατάλογος Σχημάτων

Συντομογραφίες

Δ.Ε.	Διπλωματική Εργασία
ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
AR	Augmented Reality
XR	Extended Reality
UI	User Interface
3D	Three Dimensional
SDK	Software Development Kit

Κεφάλαιο 1ο: Εισαγωγή

1.1 Εισαγωγή

Η ραγδαία εξέλιξη των τεχνολογιών επαυξημένης πραγματικότητας (Augmented Reality – AR) έχει δημιουργήσει νέες δυνατότητες στον τρόπο με τον οποίο οι χρήστες αλληλοεπιδρούν με το ψηφιακό περιεχόμενο μέσα στον φυσικό χώρο. Η AR επιτρέπει την προβολή ψηφιακών αντικειμένων και πληροφοριών στο πραγματικό περιβάλλον σε πραγματικό χρόνο, αξιοποιώντας φορητές συσκευές, όπως κινητά τηλέφωνα, tablets, καθώς και smart glasses.

Στο πλαίσιο αυτό, η παρούσα διπλωματική εργασία επικεντρώνεται στην ανάπτυξη μιας εφαρμογής AR για την Αλεξάνδρεια Πανεπιστημιούπολη του ΔΙΠΑΕ, με στόχο την προβολή ψηφιακών πληροφοριών σε συγκεκριμένα σημεία του φυσικού χώρου. Η εφαρμογή σχεδιάστηκε ώστε να επιτρέπει τη διαχείριση και τη φιλοξενία αγκυρών AR, καθώς και την προβολή του αντίστοιχου περιεχομένου από τους τελικούς χρήστες.

1.2 Σκοπός και στόχοι της διπλωματικής εργασίας

Ο κύριος σκοπός της παρούσας διπλωματικής εργασίας είναι η ανάπτυξη ενός λειτουργικού και αξιόπιστου συστήματος AR που επιτρέπει την προβολή ψηφιακών πληροφοριών στον φυσικό χώρο της Πανεπιστημιούπολης.

Οι βασικοί στόχοι της εργασίας είναι:

- η μελέτη των βασικών αρχών και τεχνολογιών της επαυξημένης πραγματικότητας,
- η υλοποίηση εφαρμογής σε περιβάλλον Unity με χρήση AR Foundation και ARCore,
- η ανάπτυξη περιβάλλοντος σύνδεσης διαχειριστών (login panel) με σκοπό την αποφυγή αλληλεπίδρασης των χρηστών με τη διαχείριση των αγκυρών,
- η ανάπτυξη διαχειριστικού περιβάλλοντος (admin panel) για την τοποθέτηση και φιλοξενία αγκυρών,
- η ανάπτυξη περιβάλλοντος διαχείρισης αγκυρών (anchor manager panel) το οποίο επιτρέπει την εύκολη και αποδοτική διαχείρισή τους,
- η ανάπτυξη περιβάλλοντος χρήστη (student panel) για την ανάκτηση και προβολή του περιεχομένου,
- η αποθήκευση και διαχείριση πληροφοριών μέσω της βάσης δεδομένων Firebase,
- η αξιολόγηση της λειτουργικότητας και της σταθερότητας της εφαρμογής σε πραγματικές συνθήκες.

1.3 Μεθοδολογία υλοποίησης

Για την υλοποίηση της εφαρμογής χρησιμοποιήθηκε το περιβάλλον ανάπτυξης Unity σε συνδυασμό με το AR Foundation και το ARCore της Google. Η διαδικασία ανάπτυξης χωρίστηκε σε διακριτά στάδια, τα οποία περιλαμβάνουν τον σχεδιασμό της αρχιτεκτονικής του συστήματος, την υλοποίηση των βασικών λειτουργιών AR και την ανάπτυξη των δύο ξεχωριστών περιβαλλόντων χρήσης.

Στο διαχειριστικό περιβάλλον πραγματοποιείται η τοποθέτηση και φιλοξενία των αγκυρών AR, ενώ στο περιβάλλον φοιτητή γίνεται η ανάκτηση και προβολή του ψηφιακού περιεχομένου στον φυσικό χώρο. Για την αποθήκευση των δεδομένων αξιοποιήθηκε βάση δεδομένων, επιτρέποντας την διαχείριση των πληροφοριών.

1.4 Δομή της διπλωματικής εργασίας

Η δομή της παρούσας διπλωματικής εργασίας οργανώνεται ως εξής:

Στο πρώτο κεφάλαιο παρουσιάζεται η εισαγωγή, οι στόχοι και μεθοδολογία της εργασίας.

Στο δεύτερο κεφάλαιο αναλύεται το θεωρητικό υπόβαθρο της επαυξημένης πραγματικότητας και οι τεχνολογίες που αξιοποιήθηκαν.

Στο τρίτο κεφάλαιο παρουσιάζεται ο σχεδιασμός και η αρχιτεκτονική του συστήματος.

Στο τέταρτο κεφάλαιο περιγράφεται αναλυτικά η υλοποίηση της εφαρμογής.

Τέλος, στο πέμπτο παρατίθενται τα συμπεράσματα και προτάσεις για μελλοντική επέκταση.

1.5 Επίλογος

Στο κεφάλαιο αυτό παρουσιάστηκε το πλαίσιο στο οποίο εντάσσεται η παρούσα διπλωματική εργασία, οι στόχοι της και η μεθοδολογία που ακολουθήθηκε. Παράλληλα, δόθηκε συνοπτική εικόνα της δομής του κειμένου, ώστε να διευκολυνθεί η κατανόηση των κεφαλαίων που ακολουθούν.

Κεφάλαιο 2ο: Θεωρητικό Υπόβαθρο

2.1 Εισαγωγή

Σε αυτό το κεφάλαιο παρουσιάζεται το θεωρητικό υπόβαθρο που σχετίζεται με την διπλωματική εργασία. Αναλύονται οι βασικές έννοιες της AR, οι τεχνολογίες που χρησιμοποιούνται για την υλοποίηση εφαρμογών AR και τα εργαλεία ανάπτυξης που αξιοποιήθηκαν. Στόχος του κεφαλαίου είναι η κατανόηση των τεχνολογιών στις οποίες βασίστηκε ο σχεδιασμός και η υλοποίηση της εφαρμογής.

2.2 Περιβάλλον Ανάπτυξης Εφαρμογής (Unity)

Το Unity είναι ένα ισχυρό περιβάλλον που χρησιμοποιείται για την ανάπτυξη παιχνιδιών και έγινε γνωστό στους προγραμματιστές παιχνιδιών, αφού τους δίνει την δυνατότητα να συνεργάζονται σε ένα έργο με αποτελεσματικό τρόπο. Υπάρχουν διάφοροι λόγοι που έγινε γνωστό το Unity. Συγκεκριμένα, περιλαμβάνει φιλικό περιβάλλον χρήστη, ευέλικτο πλαίσιο προγραμματισμού με την γλώσσα C#, καθώς και μεγάλη βιβλιοθήκη υλικού σε συνδυασμό με την πληθώρα εκπαιδευτικού υλικού και πρόσθετων. Παράλληλα, περιλαμβάνει κατάστημα (Assets Store) που παρέχει η ίδια εταιρία Unity, στην οποία υπάρχουν τρισδιάστατα μοντέλα, ηχητικά εφέ και κινούμενα σχέδια κάνοντας πιο εύκολη την προσβασιμότητα και την αποτελεσματικότητα. Επιπλέον, με το Unity υπάρχει η δυνατότητα να αναπτυχθούν παιχνίδια που εκτελούνται σε διάφορα λειτουργικά συστήματα, όπως είναι τα Windows, macOS, iOS και Android. Συνοψίζοντας, το περιβάλλον Unity αποτελεί μια ιδιαίτερα διαδεδομένη πλατφόρμα ανάπτυξης παιχνιδιών, καθώς υποστηρίζει τη συνεργατική ανάπτυξη, είναι εύκολα προσβάσιμη σε προγραμματιστές και ενισχύεται από μια ενεργή και μεγάλη κοινότητα χρηστών [1][2].

2.3 Επαυξημένη Πραγματικότητα (Augmented Reality)

Η επαυξημένη πραγματικότητα (AR) αποτελεί μια τεχνολογία που επιτρέπει την προβολή ψηφιακών στοιχείων στο πραγματικό περιβάλλον σε πραγματικό χρόνο. Η AR δεν αντικαθιστά τον φυσικό χώρο όπως κάνει η εικονική πραγματικότητα, αλλά τον εμπλουτίζει με πρόσθετες πληροφορίες, διατηρώντας την άμεση οπτική επαφή του χρήστη με το περιβάλλον [3][8].

Η λειτουργία της AR βασίζεται στη χρήση αισθητήρων των φορητών συσκευών, όπως τις κάμερες και τους αισθητήρες κίνησης, οι οποίοι επιτρέπουν τον εντοπισμό του περιβάλλοντος και τον υπολογισμό της θέσης της συσκευής. Μέσω της διαδικασίας, τα ψηφιακά αντικείμενα προβάλλονται με τέτοιο τρόπο ώστε να φαίνονται ενσωματωμένα στον φυσικό χώρο [4].

Η εξάπλωση των έξυπνων συσκευών έχει συμβάλει σημαντικά στην ανάπτυξη εφαρμογών AR, οι οποίες βρίσκουν εφαρμογή σε τομείς όπως η εκπαίδευση, η πληροφόρηση και η ψυχαγωγία. Η AR αποτελεί πλέον ένα δυναμικά εξελισσόμενο πεδίο, προσφέροντας νέες δυνατότητες παρουσίασης και αλληλεπίδρασης με ψηφιακές πληροφορίες [3].

2.4 AR Foundation

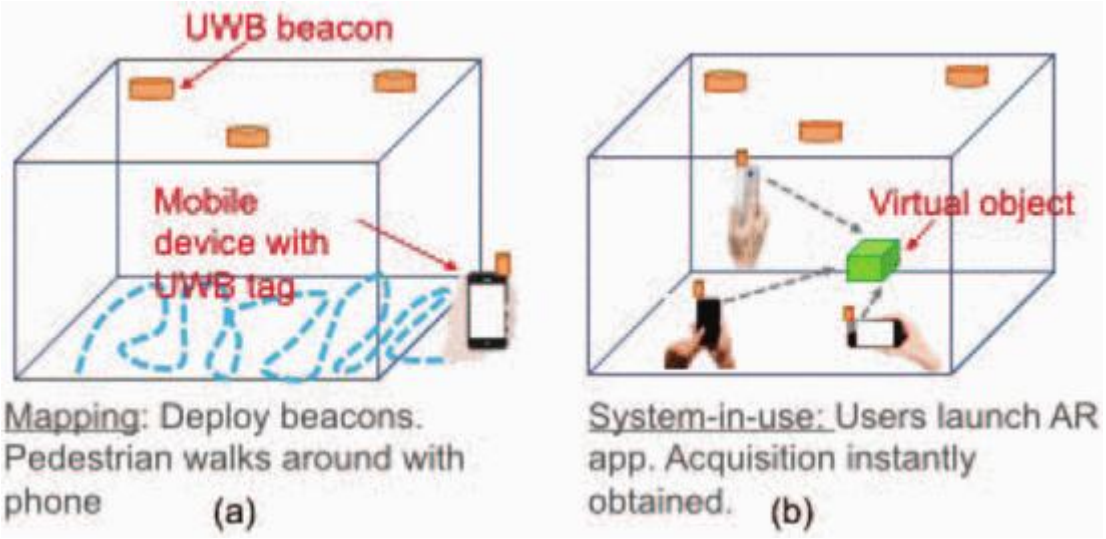
Το AR Foundation αποτελεί ένα ενιαίο πλαίσιο ανάπτυξης AR που παρέχεται από το Unity και επιτρέπει τη δημιουργία εφαρμογών AR για διαφορετικές πλατφόρμες μέσω κοινής βάσης κώδικα. Η λειτουργία που βασίζεται στην ενοποίηση των δυνατοτήτων των ARKit και ARCore, παρέχει στον προγραμματιστή πρόσβαση σε βασικές λειτουργίες AR ανεξάρτητα από το λειτουργικό σύστημα της συσκευής [4].

Μέσω του AR Foundation υποστηρίζονται λειτουργίες όπως ο εντοπισμός και η παρακολούθηση της κίνησης της συσκευής, η κατανόηση του περιβάλλοντος και η τοποθέτηση ψηφιακών αντικειμένων στον φυσικό χώρο. Η ενοποίηση αυτή μειώνει την πολυπλοκότητα της ανάπτυξης εφαρμογών AR, καθώς δεν απαιτείται ξεχωριστή υλοποίηση για κάθε πλατφόρμα [4][8].

Η χρήση του AR Foundation σε συνδυασμό με το Unity επιτρέπει την αξιοποίηση των δυνατοτήτων των σύγχρονων κινητών συσκευών, προσφέροντας ένα ευέλικτο και επεκτάσιμο περιβάλλον ανάπτυξης για εφαρμογές AR. Με τον τρόπο αυτό, διευκολύνεται η ανάπτυξη αξιόπιστων και σταθερών AR εφαρμογών που μπορούν να λειτουργούν σε πολλές και διάφορες συσκευές [4].

2.5 AR Anchors και Cloud Anchors

Οι άγκυρες AR αποτελούν μηχανισμό σύνδεσης ψηφιακών αντικειμένων με συγκεκριμένα σημεία του φυσικού χώρου, επιτρέποντας τη σταθερή προβολή τους, ανεξάρτητα από την κίνηση ή την γωνιά θέασης του χρήστη. Μέσω των anchors, τα ψηφιακά στοιχεία διατηρούν τη χωρική τους θέση, προσφέροντας μια πιο αξιόπιστη και ρεαλιστική εμπειρία AR [5].



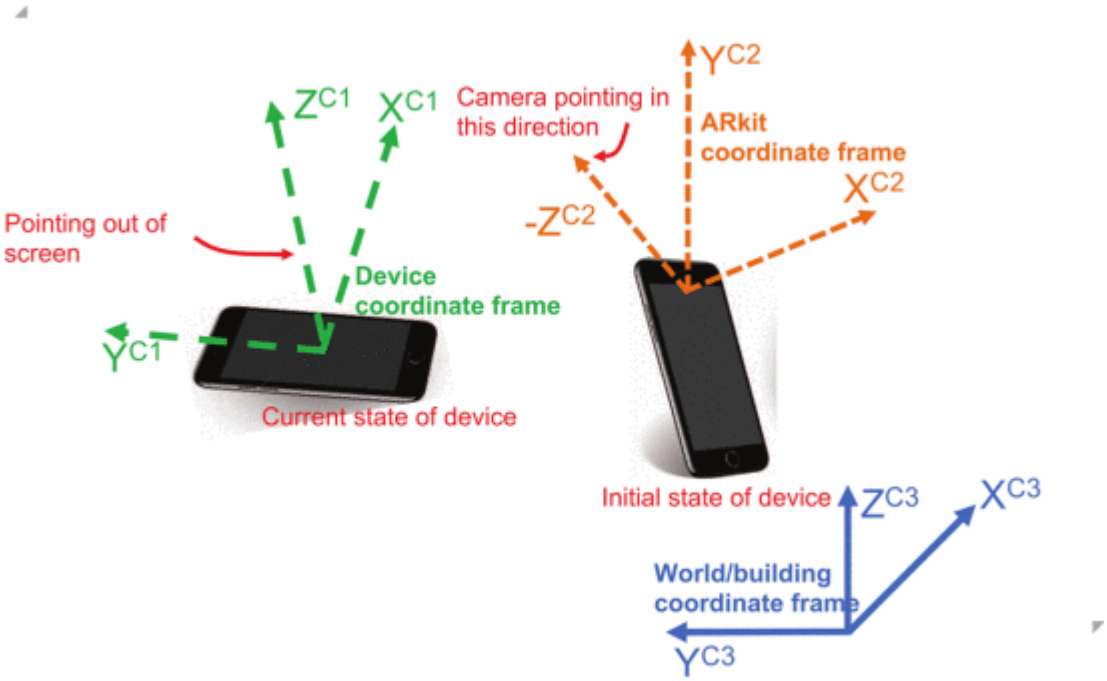
Σχήμα 2.1: Παράδειγμα πολυχρηστικής επαυξημένης πραγματικότητας με χρήση cloud.

Σε τυπικές εφαρμογές AR, οι άγκυρες συσχετίζονται με τοπικές συνεδρίες χρήσης και με σύστημα αναφοράς που ορίζεται κατά την έναρξη της εφαρμογής. Αυτό έχει ως αποτέλεσμα το ψηφιακό περιεχόμενο να μην παραμένει σταθερό στον χώρο όταν η εφαρμογή τερματιστεί ή όταν χρησιμοποιηθεί διαφορετική συσκευή. Στο Σχήμα 2.1, παρουσιάζεται ενδεικτικά η ανάγκη ύπαρξης κοινού συστήματος αναφοράς, έτσι ώστε πολλαπλοί χρήστες να μπορούν να αλληλοεπιδρούν με το ίδιο ψηφιακό περιεχόμενο στον ίδιο φυσικό χώρο [6].

Η τεχνική βάση των AR Anchors στηρίζεται στον χωρικό εντοπισμό και τη χαρτογράφηση του περιβάλλοντος, μέσω της ανίχνευσης χαρακτηριστικών σημείων (feature points) και της δημιουργίας σημειακών νεφών (point clouds). Μέσω αυτών των διαδικασιών, το σύστημα αποκτά κατανόηση της

γεωμετρίας του χώρου, επιτρέποντας τη σταθερή τοποθέτηση ψηφιακών αντικειμένων σε πραγματικές επιφάνειες, ακόμη και σε εσωτερικά περιβάλλοντα [7].

Για την αντιμετώπιση του περιορισμού της τοπικής συνεδρίας έχουν αναπτυχθεί οι Cloud Anchors, οι οποίες επιτρέπουν την αποθήκευση πληροφοριών χωρικού εντοπισμού στο cloud και την ανάκτηση τους από διαφορετικές συσκευές και χρήστες. Με αυτόν τον τρόπο επιτυγχάνεται επίμονη επαυξημένη πραγματικότητα και υποστηρίζονται πολυχρηστικά σενάρια χρήσης [6].



Σχήμα 2.2: Διαφορετικά συστήματα συντεταγμένων σε εφαρμογές AR.

Στο Σχήμα 2.2 απεικονίζονται τα διαφορετικά συστήματα συντεταγμένων που χρησιμοποιούνται σε εφαρμογές AR, όπου το σύστημα της συσκευής και της συνεδρίας AR διαφέρει από το παγκόσμιο σύστημα αναφοράς. Η διαφοροποίηση αυτή καθιστά αναγκαία τη χρήση cloud-based μηχανισμών αγκύρωσης, έτσι ώστε το ψηφιακό περιεχόμενο να παραμένει σταθερό και κοινό για όλους τους χρήστες [6].

Συνολικά, οι AR Anchors και οι Cloud Anchors αποτελούν βασικά δομικά στοιχεία για τη δημιουργία σύγχρονων εφαρμογών AR, καθώς υποστηρίζουν τη σταθερότητα, τη διάρκεια και τη συνεργατική χρήση ψηφιακών αντικειμένων στον φυσικό χώρο.

2.6 Firebase ως υποστηρικτική Υποδομή

Για την αποθήκευση και τον συγχρονισμό δεδομένων σε εφαρμογές AR, μπορούν να αξιοποιηθούν cloud πλατφόρμες τύπου Backend-as-a-Service (BaaS), οι οποίες παρέχουν έτοιμες υποδομές για τη διαχείριση δεδομένων σε πραγματικό χρόνο, χωρίς την ανάγκη ανάπτυξης και συντήρησης αυτόνομου backend συστήματος. Οι πλατφόρμες αυτές διευκολύνουν τη δημιουργία εφαρμογών που απαιτούν ανταλλαγή πληροφοριών μεταξύ πολλαπλών χρηστών και συσκευών [9].

Μια ευρέως διαδεδομένη πλατφόρμα BaaS είναι το Firebase, η οποία προσφέρεται από την Google και παρέχει υπηρεσίες, όπως βάσεις δεδομένων πραγματικού χρόνου, μηχανισμούς συγχρονισμού

δεδομένων και υποστήριξης cloud υποδομών. Το Firebase προσφέρει άμεση διασύνδεση με το Unity, γεγονός που το καθιστά κατάλληλο για εφαρμογές AR που αναπτύσσονται σε περιβάλλοντα παιχνιδιών και διαδραστικών εφαρμογών [9].

Η χρήση cloud υποδομών, όπως το Firebase, επιτρέπει την αποθήκευση και την ανάκτηση δεδομένων που σχετίζονται με ψηφιακό περιεχόμενο και τις χωρικές πληροφορίες, υποστηρίζοντας σενάρια όπου απαιτείται κοινή πρόσβαση σε δεδομένα από διαφορετικές συσκευές. Με αυτόν τον τρόπο, ενισχύεται η λειτουργικότητα εφαρμογών που βασίζονται σε επίμονες και πολυχρηστικές εμπειρίες AR, σύμφωνα με τις δυνατότητες που περιγράφονται στην επίσημη τεκμηρίωση της πλατφόρμας [9].

2.7 Επίλογος

Στο παρόν κεφάλαιο παρουσιάστηκε το θεωρητικό υπόβαθρο της AR και αναλύθηκαν οι βασικές τεχνολογίες που σχετίζονται με την ανάπτυξη σύγχρονων εφαρμογών AR. Ειδικότερα, έγινε αναφορά στις θεμελιώδεις αρχές της AR, στα συστήματα αγκύρωσης ψηφιακού περιεχομένου στον φυσικό χώρο, καθώς και στη χρήση cloud-based μηχανισμών για την υποστήριξη επίμονων και πλυχρηστικών εμπειριών.

Παράλληλα παρουσιάστηκε ο ρόλος του Unity και του AR Foundation ως βασικών εργαλείων ανάπτυξης εφαρμογών AR, τα οποία παρέχουν και ευέλικτο πλαίσιο για την αξιοποίηση των δυνατοτήτων των σύγχρονων κινητών συσκευών. Επιπλέον, έγινε αναφορά σε cloud πλατφόρμες τύπου Backend-as-a-Service (BaaS), όπως το Firebase, οι οποίες λειτουργούν ως υποστηρικτική υποδομή για την αποθήκευση και τον συγχρονισμό δεδομένων, διευκολύνοντας τη διαχείριση πληροφοριών σε εφαρμογές που βασίζονται σε πολυχρηστικά και επίμονα σενάρια AR. Το θεωρητικό υπόβαθρο αποτελεί τη βάση για την κατανόηση σχεδιασμού και της υλοποίησης της εφαρμογής που παρουσιάζεται στα επόμενα κεφάλαια.

Στο επόμενο κεφάλαιο ακολουθεί η ανάλυση και ο σχεδιασμός του συστήματος, όπου περιγράφεται αναλυτικά η αρχιτεκτονική και οι λειτουργίες της εφαρμογής που αναπτύχθηκε στο πλαίσιο της παρούσας διπλωματικής εργασίας.

Κεφάλαιο 3ο: Υλοποίηση εργασίας

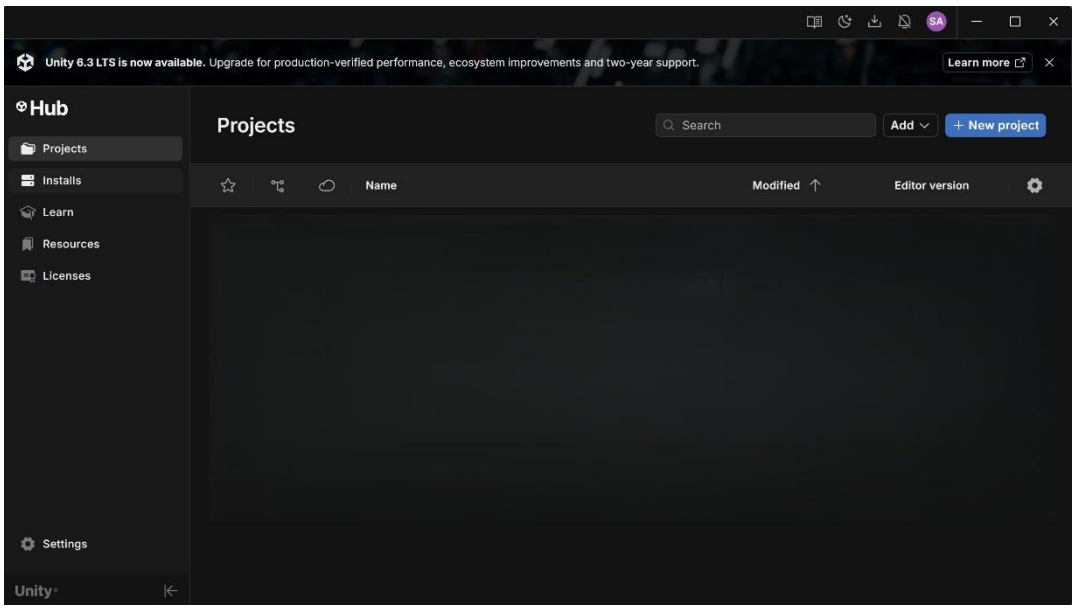
3.1 Εισαγωγή

Στο παρόν κεφάλαιο παρουσιάζονται τα εργαλεία, τα πακέτα και οι τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής της παρούσας διπλωματικής εργασίας. Η υλοποίηση πραγματοποιήθηκε κυρίως στο περιβάλλον ανάπτυξης Unity [10], το οποίο χρησιμοποιήθηκε τόσο για τον προγραμματισμό της εφαρμογής όσο και για την διαχείριση του διαδραστικού περιβάλλοντος και του τρισδιάστατου περιεχομένου.

Για την υλοποίηση των λειτουργιών AR αξιοποιήθηκε το πλαίσιο AR Foundation [11], σε συνδυασμό με τις δυνατότητες του ARCore [12], επιτρέποντας τη χρήση αγκυρών και μηχανισμών χωρικού εντοπισμού στον φυσικό χώρο. Επιπλέον, για την αποθήκευση και τον συγχρονισμό δεδομένων που σχετίζονται με την εφαρμογή χρησιμοποιήθηκε η πλατφόρμα Firebase [9], η οποία λειτουργεί ως υποστηρικτική υποδομή cloud. Τέλος, για την αποθήκευση και φιλοξενία των cloud anchors, αξιοποιήθηκε η υποδομή Google Cloud [13], η οποία επιτρέπει την απομακρυσμένη αποθήκευση και ανάκτηση χωρικών δεδομένων από πολλαπλές συσκευές.

3.2 Δημιουργία έργου στο Unity

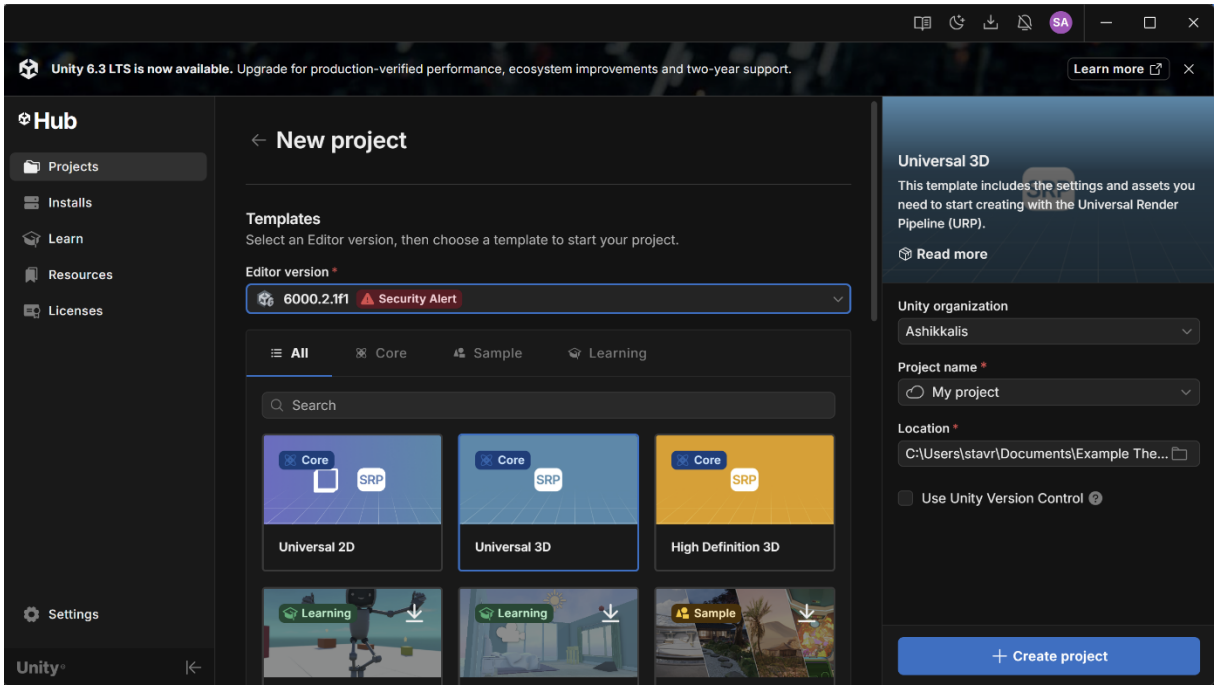
Για την δημιουργία ενός νέου έργου στο Unity, αρχικά έγινε η εγκατάσταση του από την επίσημη σελίδα [10], και έπειτα αφού ανοίχτηκε η εφαρμογή, επιλέχθηκε το “New project”, όπως φαίνεται στο Σχήμα 3.1.



Σχήμα 3.1: Δημιουργία νέου έργου.

Στην συνέχεια αφού επιλέχθηκε το κουμπί “New project” που αναφέρθηκε πιο πάνω, εμφανίστηκε ένα νέο παράθυρο, όπως φαίνεται στο Σχήμα 3.2. Σε αυτό το παράθυρο μπορεί να επιλεχθεί η έκδοση που

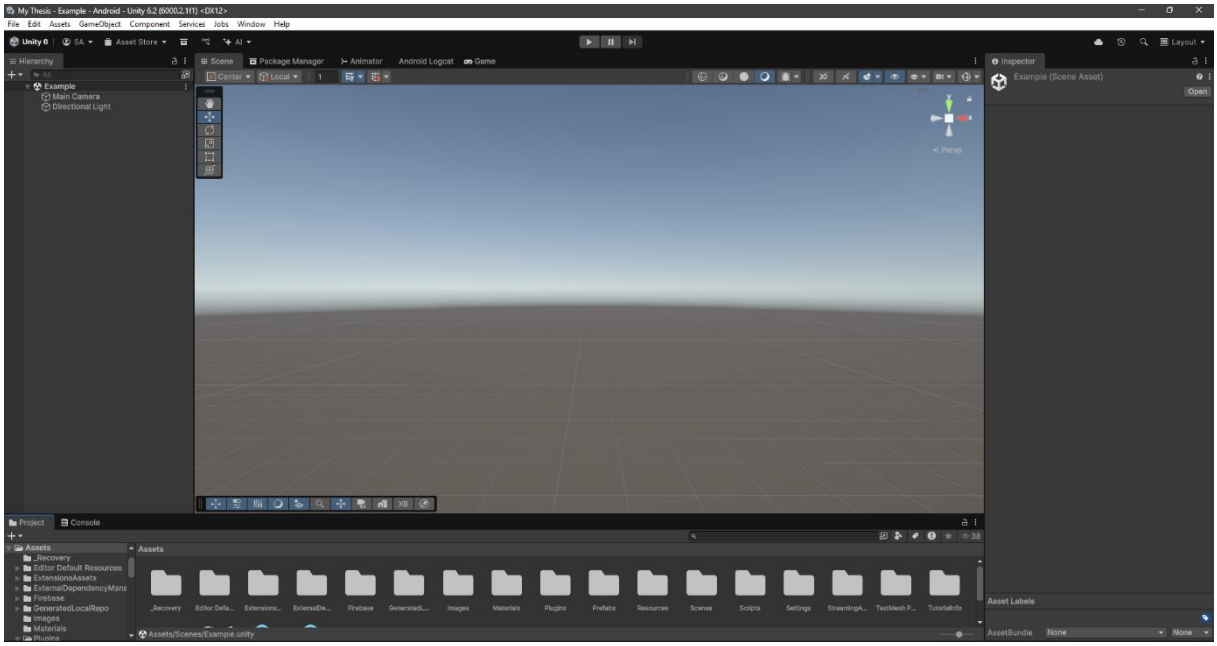
ο χρήστης επιθυμεί να χρησιμοποιήσει , το είδος του project, η ονομασία του και ο χώρος αποθήκευσης του. Επίσης, με το κουμπί “Create Project”, δημιουργείται το έργο.



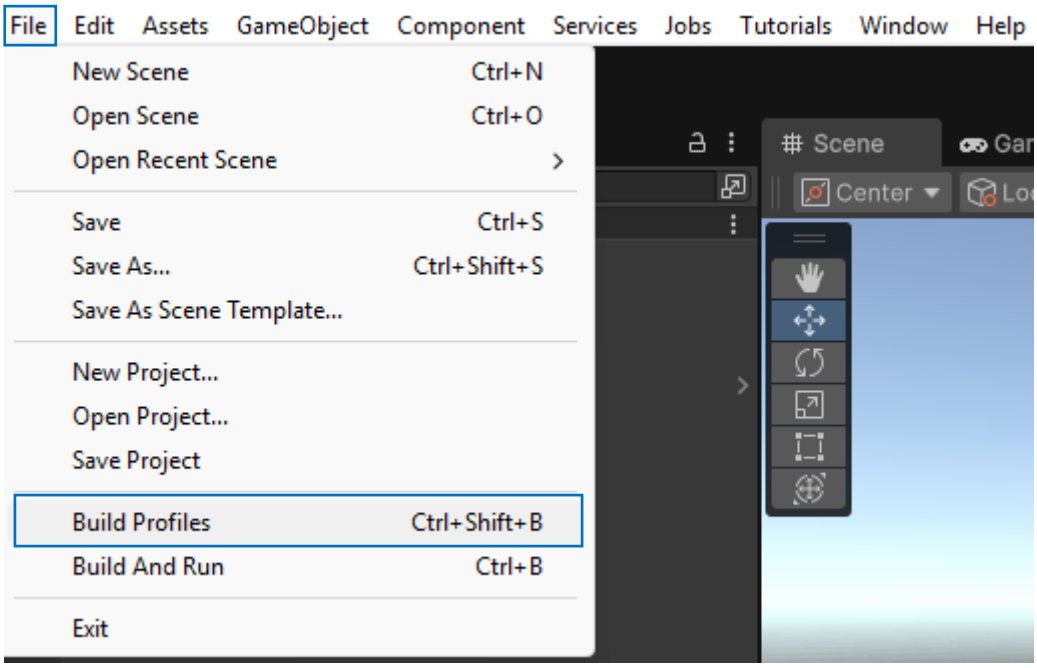
Σχήμα 3.2: Ονομασία, έκδοση και αποθήκευση του νέου έργου.

3.2.1 Ρυθμίσεις έργου

Στο Σχήμα 3.3, φαίνεται η άδεια σκηνή που εμφανίστηκε όταν δημιουργήθηκε το έργο. Συγκεκριμένα, στα αριστερά της εικόνας φαίνεται η ιεραρχία, στα δεξιά ο inspector, στην μέση η σκηνή και στο κάτω μέρος όλα τα αρχεία του έργου, καθώς και το console.



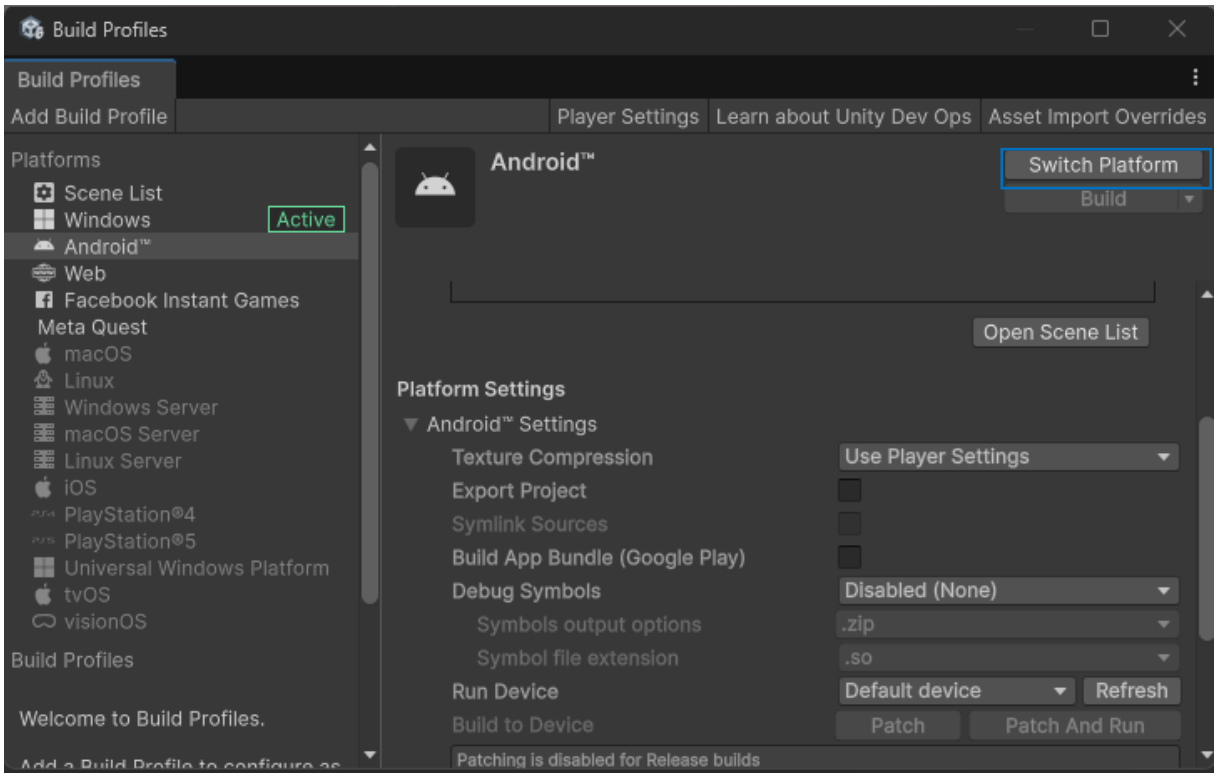
Σχήμα 3.3: Άδεια Σκηνή.



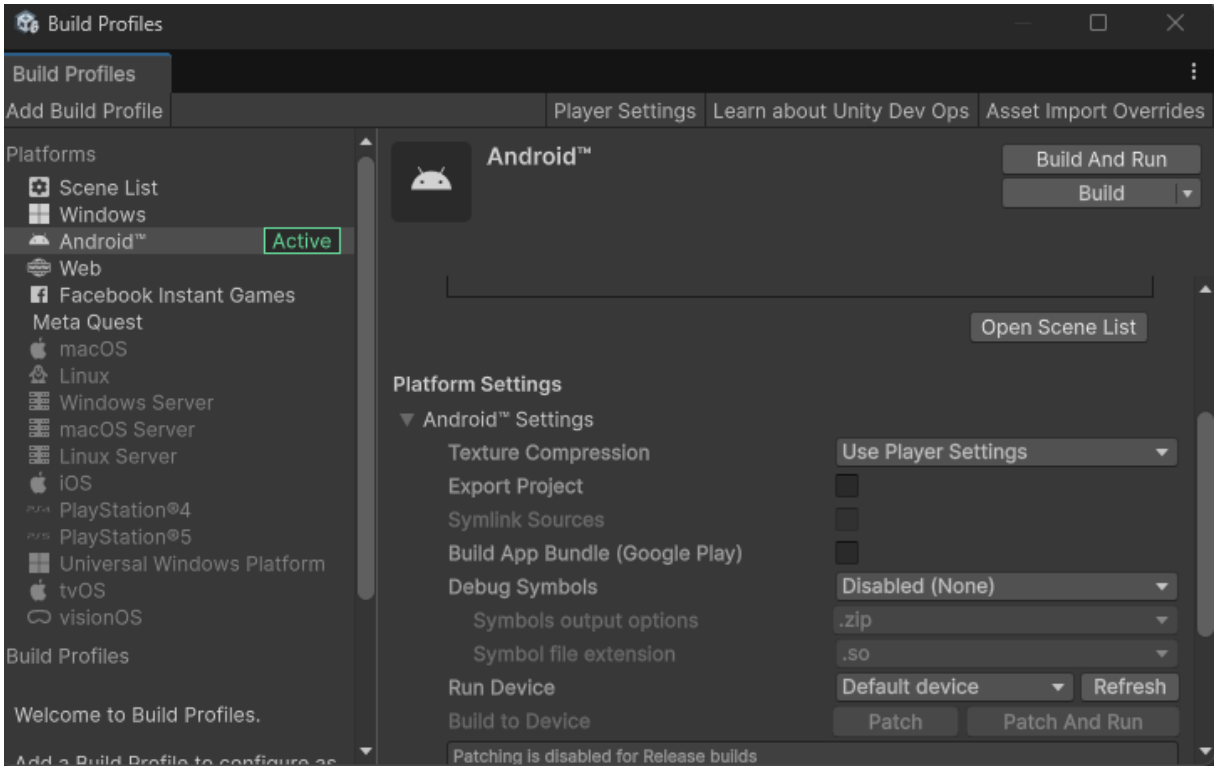
Σχήμα 3.4: Εύρεση ρύθμησης.

Αρχικά, πριν να ξεκινήσει η ανάπτυξη της εφαρμογής, έπρεπε να γίνουν κάποιες ρυθμίσεις για να προσαρμοστεί η εφαρμογή στο επιθυμητό λογισμικό. Συγκεκριμένα, η πρώτη ρύθμιση που έγινε ήταν η επιλογή του λογισμικού που θα ήταν συμβατή η εφαρμογή. Για να πραγματοποιηθεί αυτό, επιλέχθηκε το “File” και μετά το “Build Profiles”, όπως φαίνεται στο Σχήμα 3.4.

Στην συνέχεια, όπως δείχνει το Σχήμα 3.5, επιλέχθηκε το Android, εφόσον η εφαρμογή θα στηρίζει Android συσκευές, και έπειτα επιλέχθηκε το “Switch platform”. Η ένδειξη “Activate” στο Σχήμα 3.6, επιβεβαιώνει ότι το λογισμικό Android είναι ενεργοποιημένο.

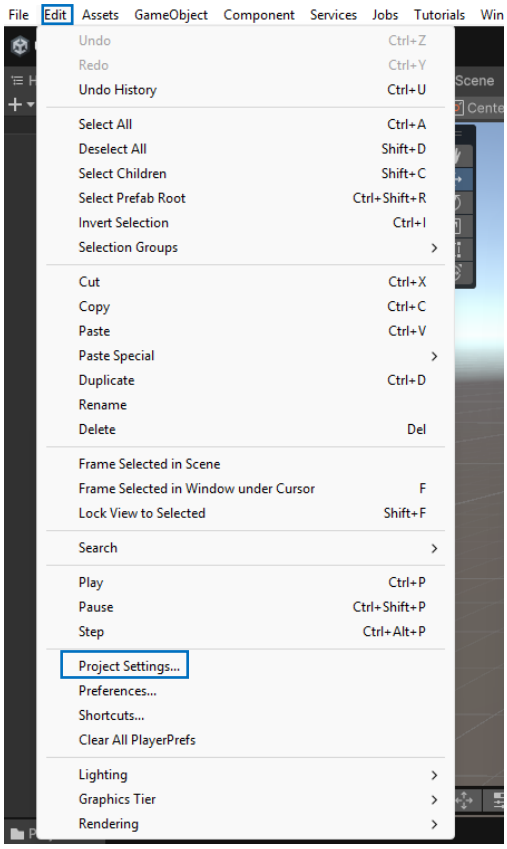


Σχήμα 3.6: Ενεργοποίηση Android λογισμικού.

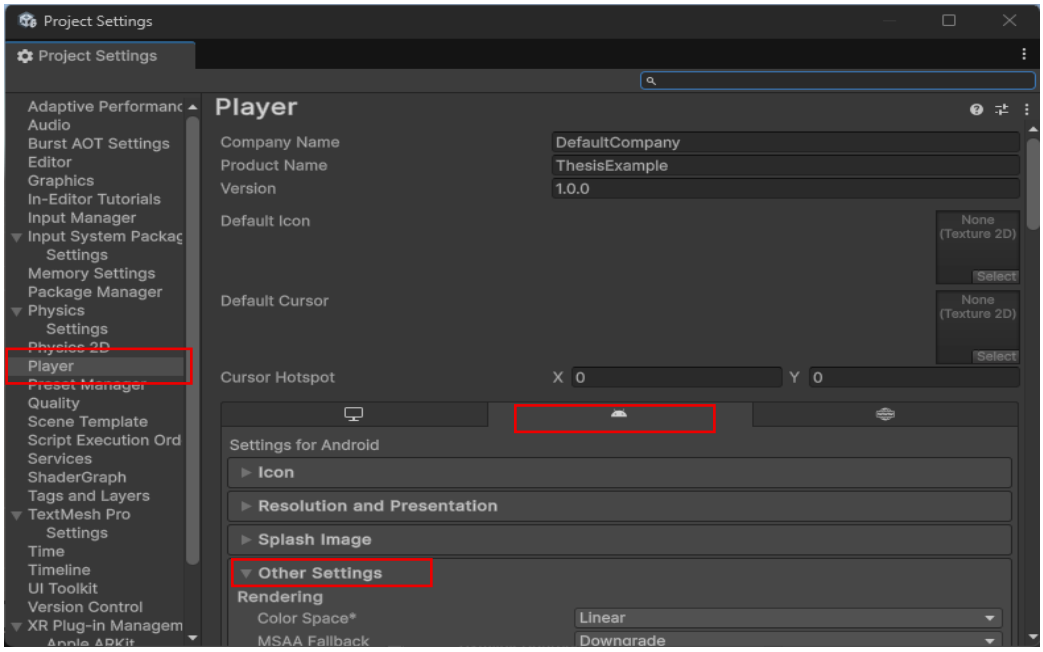


Σχήμα 3.5: Επιλογή λογισμικού.

Στη συνέχεια, έπρεπε να γίνουν επιπλέον ρυθμίσεις πριν να ξεκινήσει το έργο. Όπως φαίνεται στο Σχήμα 3.7, επιλέχθηκε το “Edit” και μετά το “Project Settings”. Έπειτα, από την καρτέλα που άνοιξε όπως φαίνεται στο Σχήμα 3.8, επιλέχθηκε το “Player”, έπειτα το Android και τέλος η κατηγορία “Other Settings”.

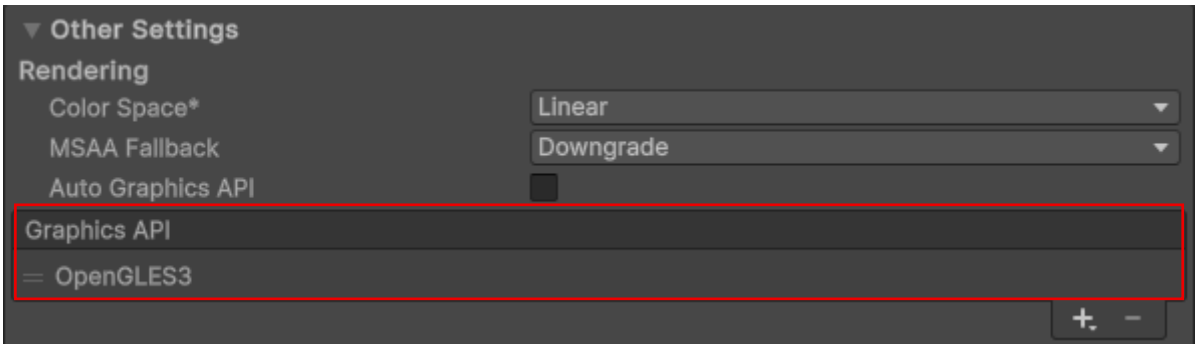


Σχήμα 3.7: Εύρεση Project settings.



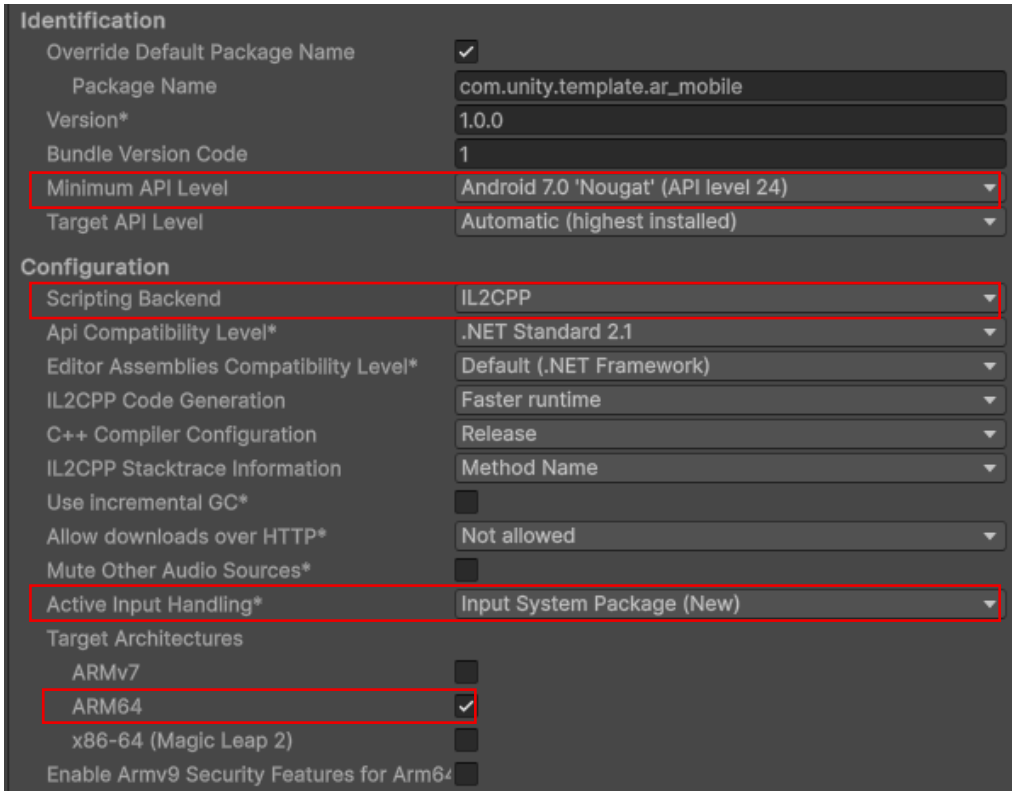
Σχήμα 3.8: Επιλογή Player και Android.

Έπειτα, όπως φαίνεται στο Σχήμα 3.9, αφαιρέθηκε η επιλογή “Vulkan” και έμεινε μόνο το “OpenGLES3”.



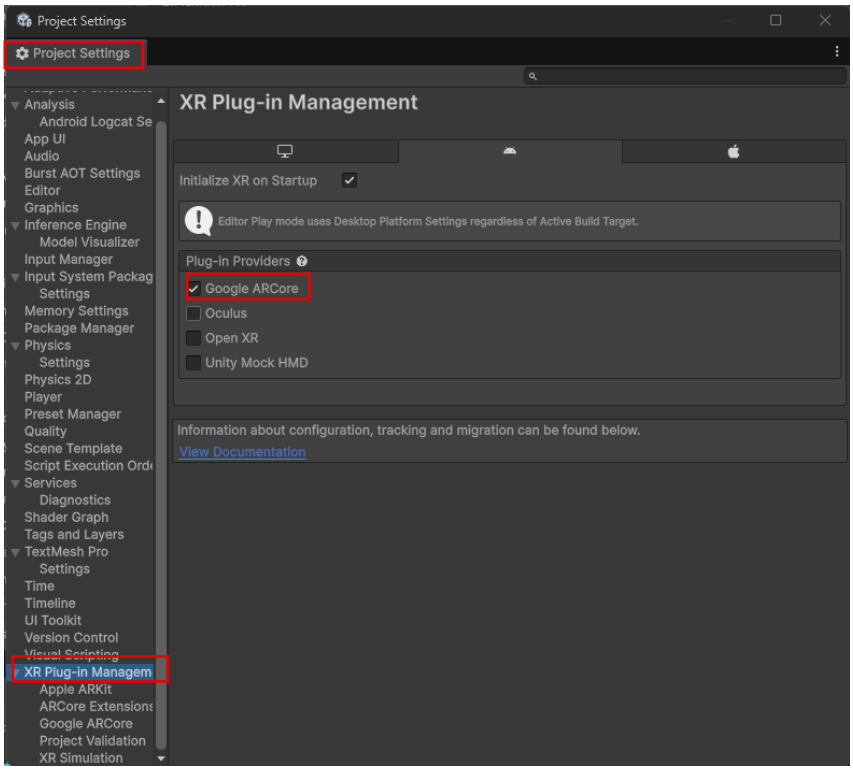
Σχήμα 3.9: Αφαίρεση επιλογής "Vulkan".

Στη συνέχεια, όπως παρατηρείται στο Σχήμα 3.10, έγιναν κάποιες επιλογές. Συγκεκριμένα, επιλέχθηκε το επιθυμητό Minimum API Level, στο Scripting Backend επιλέχθηκε το “IL2CPP”, στο Active Input Handling επιλέχθηκε το “Input System Package (new)”, και στο Target Architectures επιλέχθηκε το “ARM64”.



Σχήμα 3.10: Ρυθμίσεις.

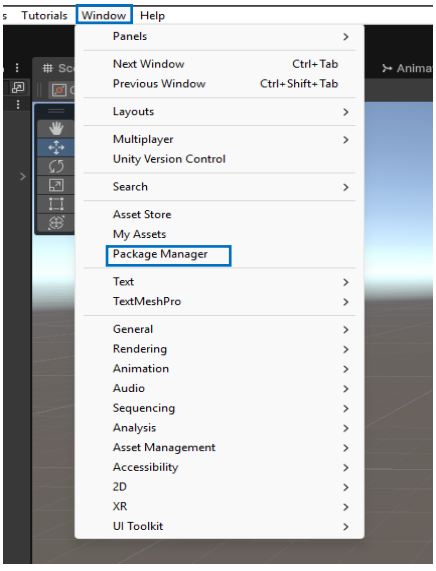
Τέλος, όπως δείχνει το Σχήμα 3.11, ενεργοποιήθηκε η επιλογή Google ARCore από την καρτέλα Project Settings, στην οποία επιλέχθηκε το “XR Plug-in Management” και έπειτα η επιλογή “Google ARCore”.



Σχήμα 3.11: Ενεργοποίηση του Google ARCore.

3.3 Εγκατάσταση πακέτων

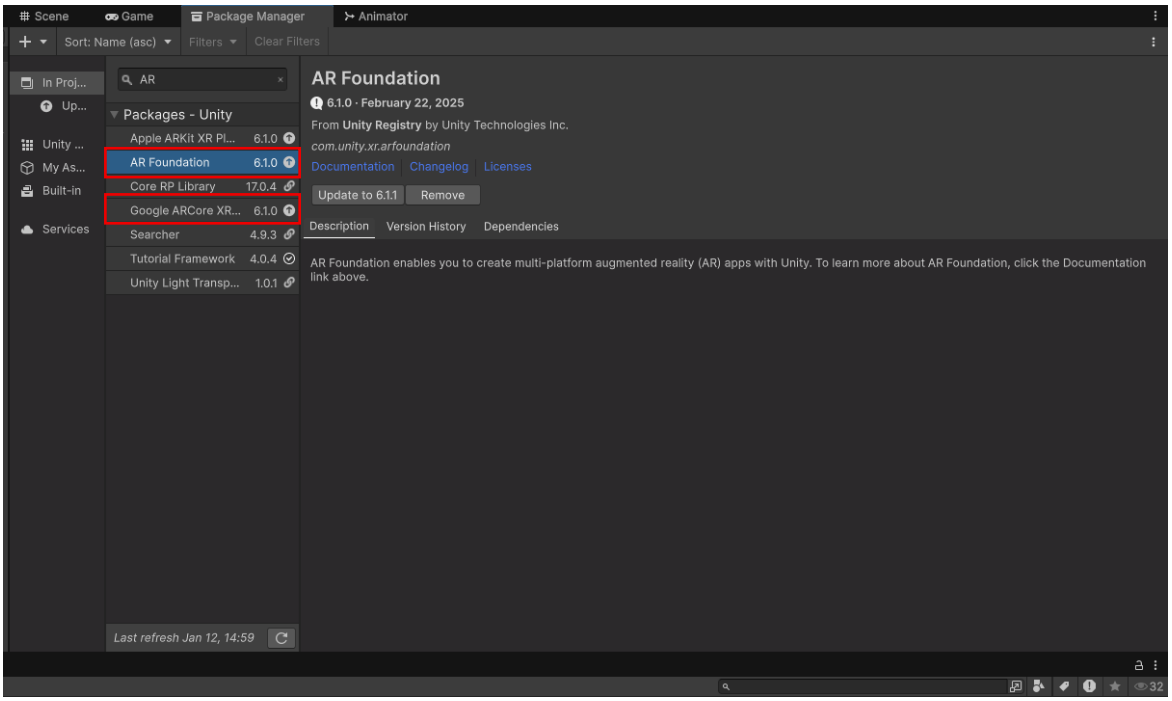
Για την εγκατάσταση πακέτων που χρειάζονται για την εφαρμογή, επιλέχθηκε από την καρτέλα Window, το Package Manager, όπως φαίνεται στο Σχήμα 3.12.



Σχήμα 3.12: Εύρεση Package Manager.

Στη συνέχεια, από το Package Manager, κατέβηκαν τα AR Foundation και Google ARCore XR Plugin, όπως φαίνεται στο Σχήμα 3.13, έτσι ώστε να υπάρχουν τα απαραίτητα εργαλεία για την ανάπτυξη μιας

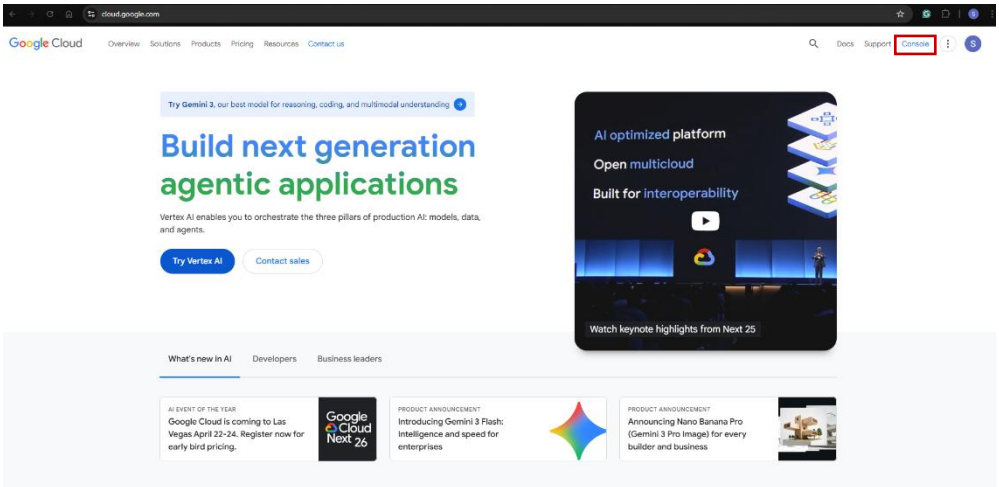
εφαρμογής με AR. Τέλος, από το Package Manager έγινε εισαγωγή το ARCore Extensions plugin με ένα GitHub URL που παρέχει η Google [14].



Σχήμα 3.13: Εγκατάσταση πρόσθετων.

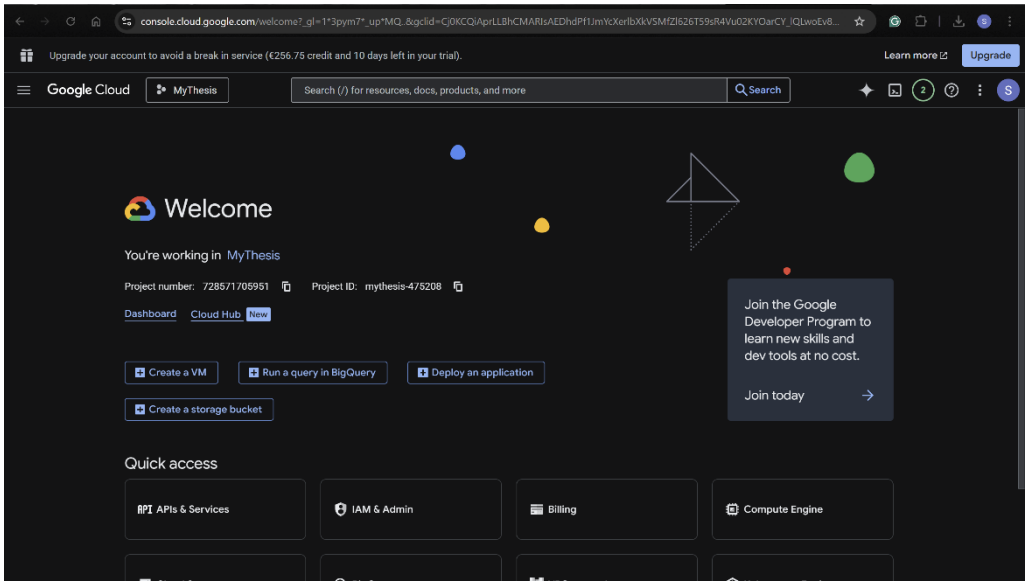
3.4 Google Cloud

Για να πραγματοποιηθεί η ένωση του Google Cloud με το έργο στο Unity, έπρεπε να γίνει μέσω της επίσημης σελίδας του Google Cloud [13], στην οποία επιλέχθηκε το Console για τη δημιουργία έργου (Σχήμα 3.14).

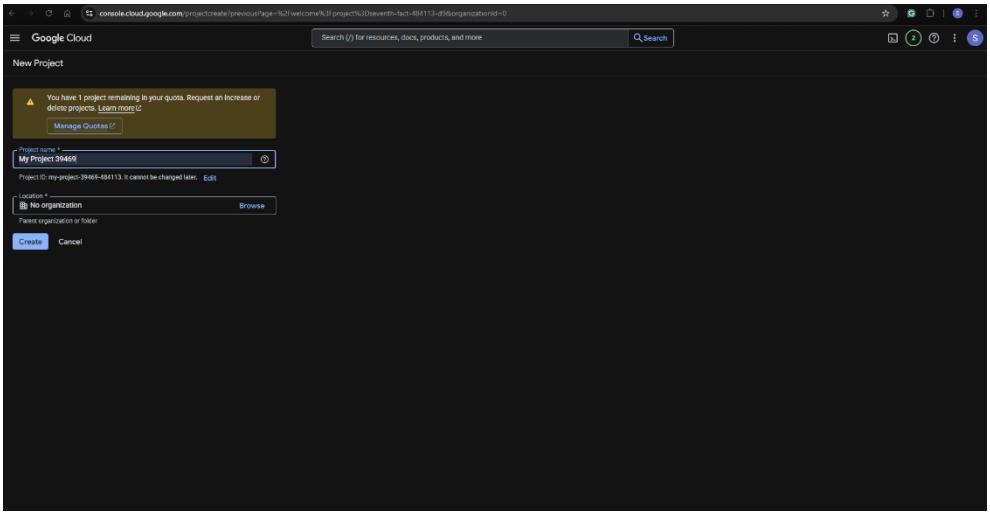


Σχήμα 3.14: Σελίδα Google Cloud.

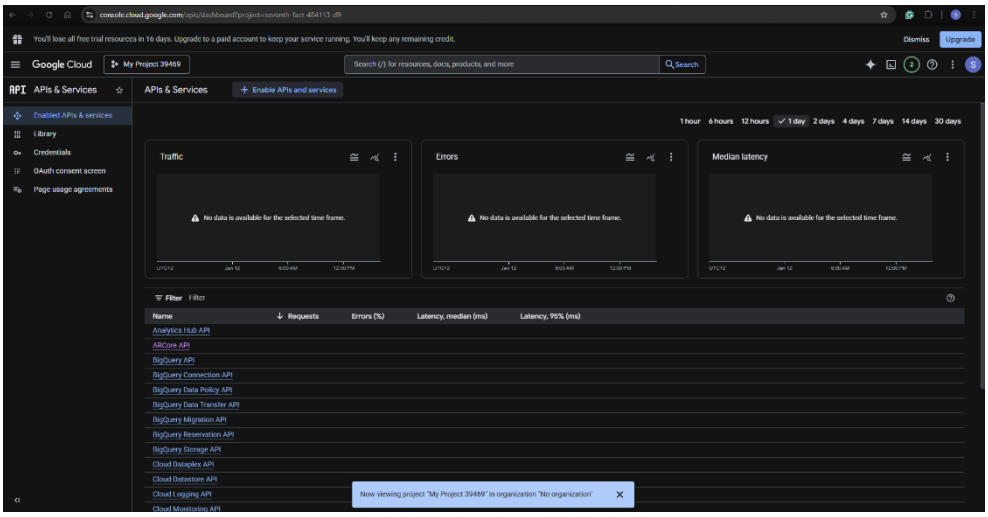
Στο Σχήμα 3.15 φαίνεται η συνέχεια της επιλογής Console από το Σχήμα 3.14. Για την δημιουργία ενός νέου έργου, έγινε επιλογή ονόματος έργου όπως φαίνεται στο Σχήμα 3.16, και έπειτα, από το APIs & Services το οποίο φαίνεται στο Σχήμα 3.17, επιλέχθηκε το "+ Enable APIs and Services", έγινε αναζήτηση "ARCore API" και ενεργοποιήθηκε όπως φαίνεται στο Σχήμα 3.18.



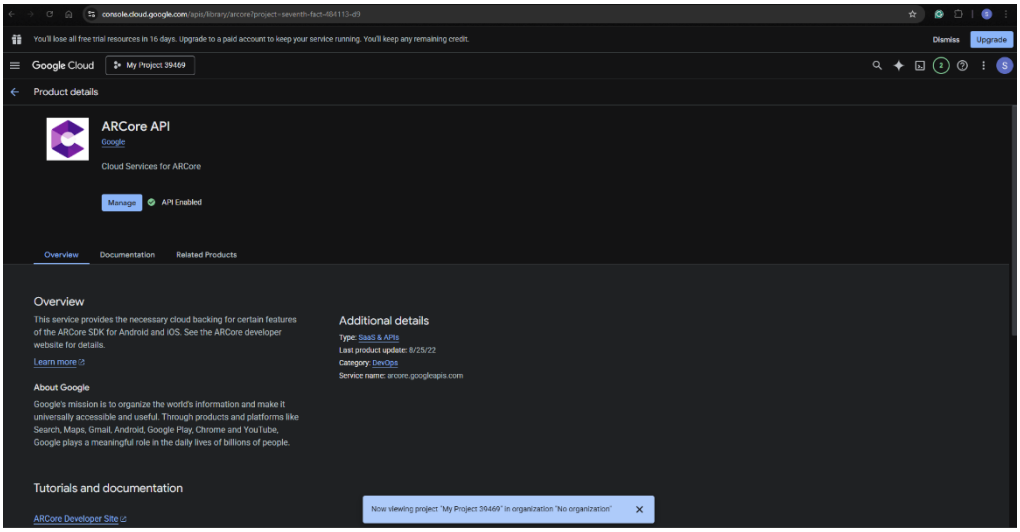
Σχήμα 3.16: Google Cloud Console.



Σχήμα 3.15: Δημιουργία έργου.



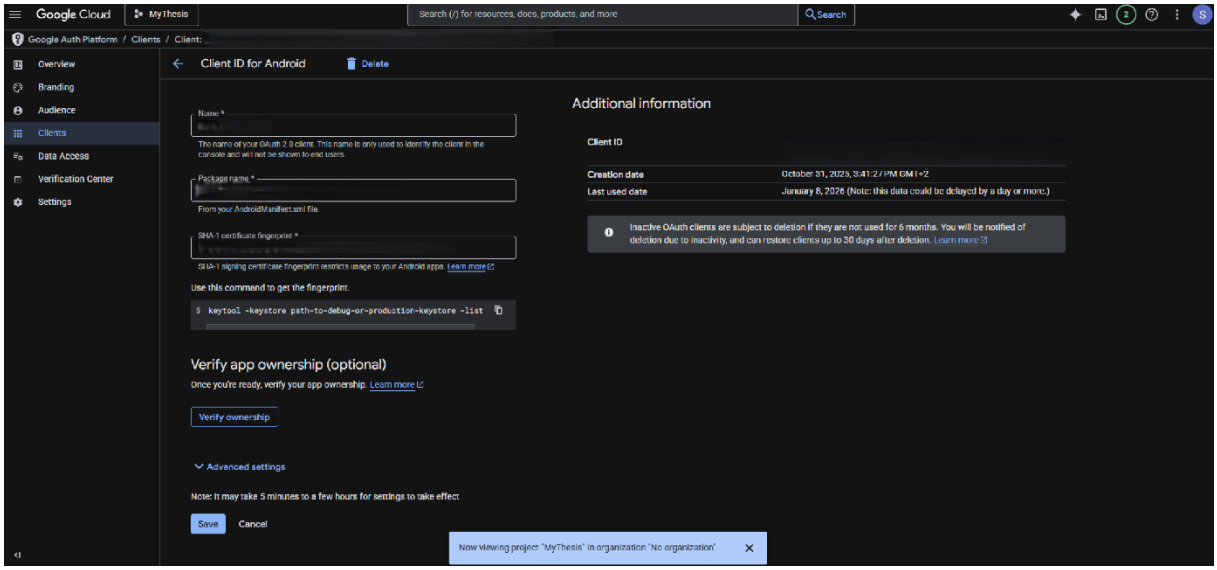
Σχήμα 3.18: APIs & Services.



Σχήμα 3.17: Ενεργοποίηση του ARCore API.

Αφού ενεργοποιήθηκε το ARCore API, επιλέχθηκε ο “keyless” τρόπος σύνδεσης με το έργο στο Unity. Με αυτόν τον τρόπο, οι άγκυρες θα μπορούν να φιλοξενηθούν στο Google Cloud μέχρι 365 μέρες, σε αντίθεση με το να φτιαχνόταν ένα API key το οποίο θα μπορεί να φιλοξενήσει τις άγκυρες μόνο για μία μέρα.

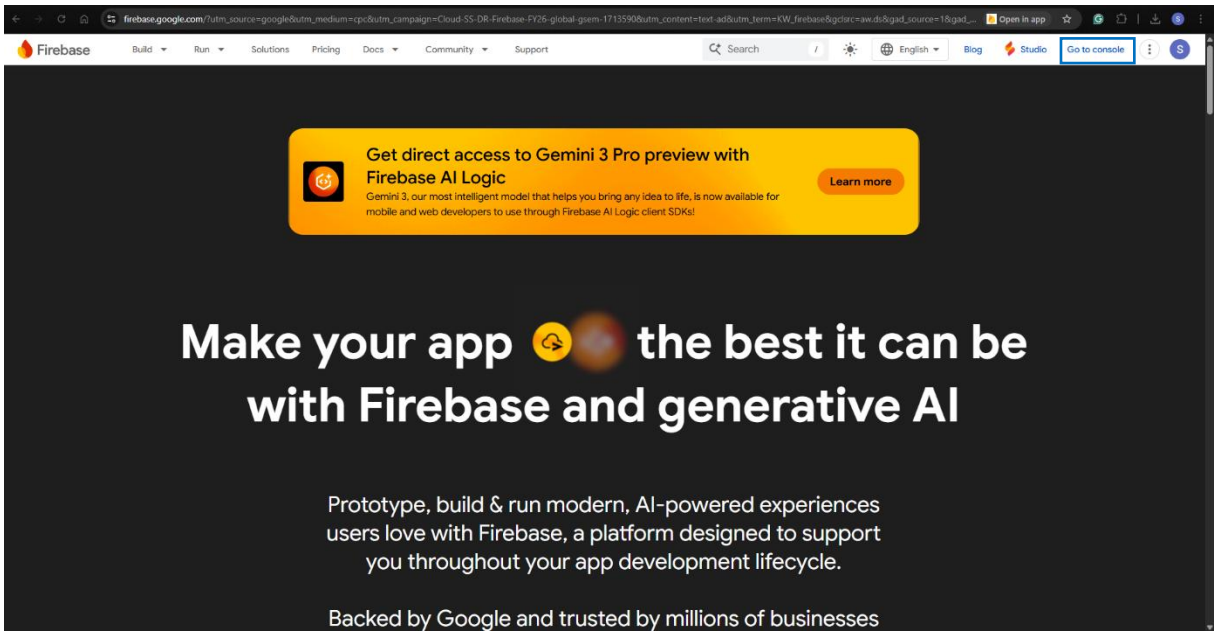
Επομένως, δημιουργήθηκε ένα νέο credential και επιλέχθηκε το “OAuth2.0 Client IDs” όπου είναι ο keyless τρόπος που αναφέρθηκε πιο πάνω. Στην συνέχεια, όπως φαίνεται στο Σχήμα 3.19, έπρεπε να δοθεί ένα όνομα, το package name του project στο Unity και το SHA-1 certificate fingerprint το οποίο μπορεί να βρεθεί από το cmd με τις εντολές που υπάρχουν στην σελίδα του ARCore [15].



Σχήμα 3.19: Δημιουργία Client ID.

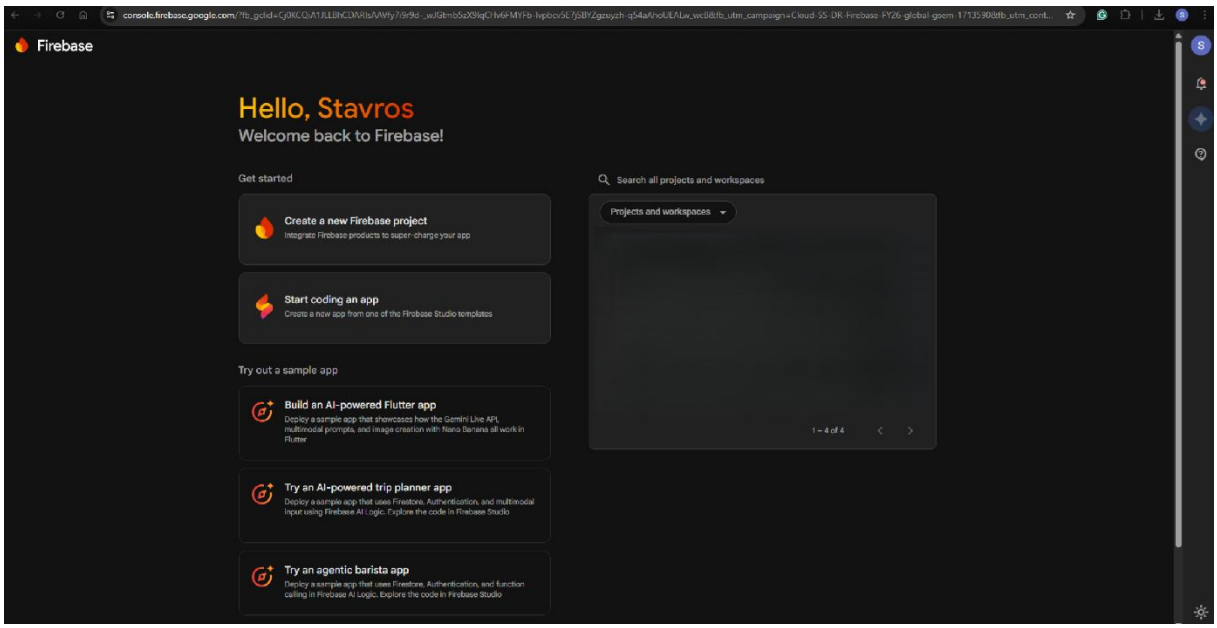
3.5 Δημιουργία βάσης δεδομένων (Firebase)

Για την δημιουργία βάσης δεδομένων χρησιμοποιήθηκε το Firebase, στο οποίο έπρεπε να δημιουργηθεί ένα νέο έργο έτσι ώστε να συνδεθεί με το έργο του Unity. Για να γίνει αυτό, από την αρχική σελίδα του Firebase επιλέχθηκε το κουμπί “Go to console” στο πάνω δεξιά μέρος όπως φαίνεται στο Σχήμα 3.20.



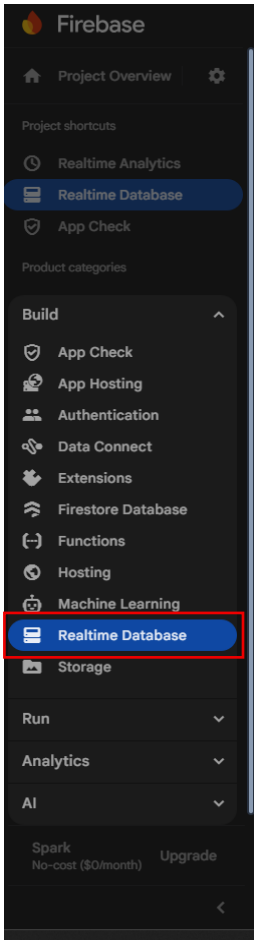
Σχήμα 3.20: Αρχική σελίδα Firebase.

Αφού μεταφέρθηκε στο console, δημιουργήθηκε ένα νέο έργο, όπως μπορεί να φανεί στο Σχήμα 3.21, επιλέχθηκε ένα όνομα και το create.



Σχήμα 3.21: Δημιουργία έργου.

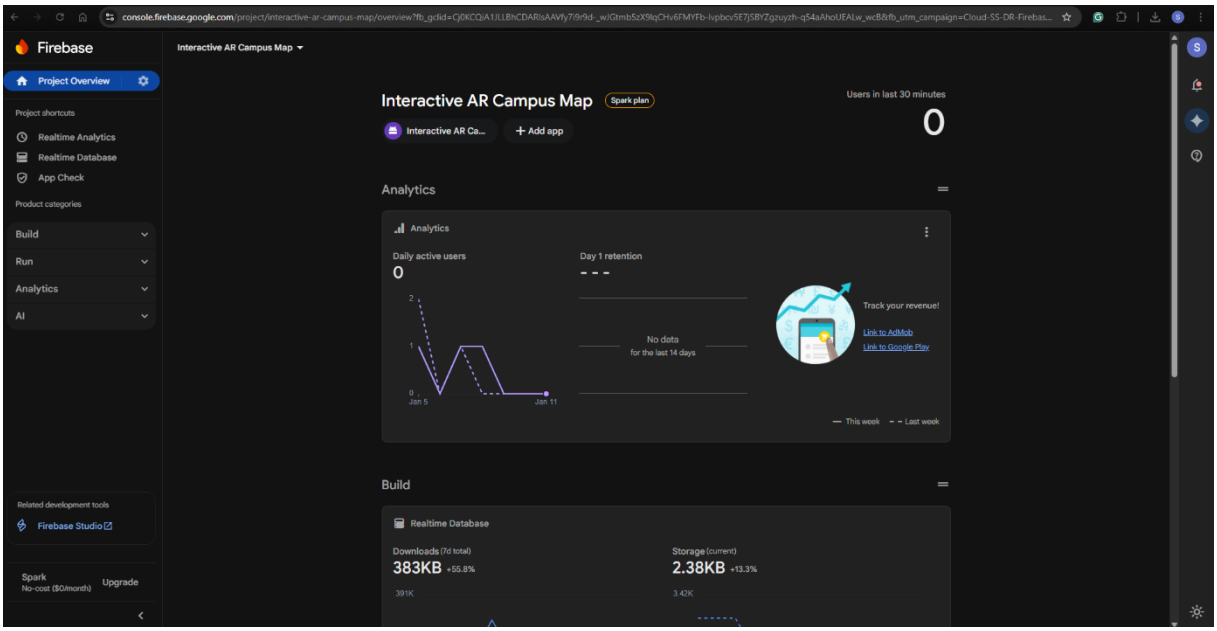
Έπειτα, από το project που δημιουργήθηκε, επιλέχθηκε το “Build” , όπως φαίνεται στο Σχήμα 3.22, και έπειτα το Realtime Database.



Σχήμα 3.22: Realtime Database

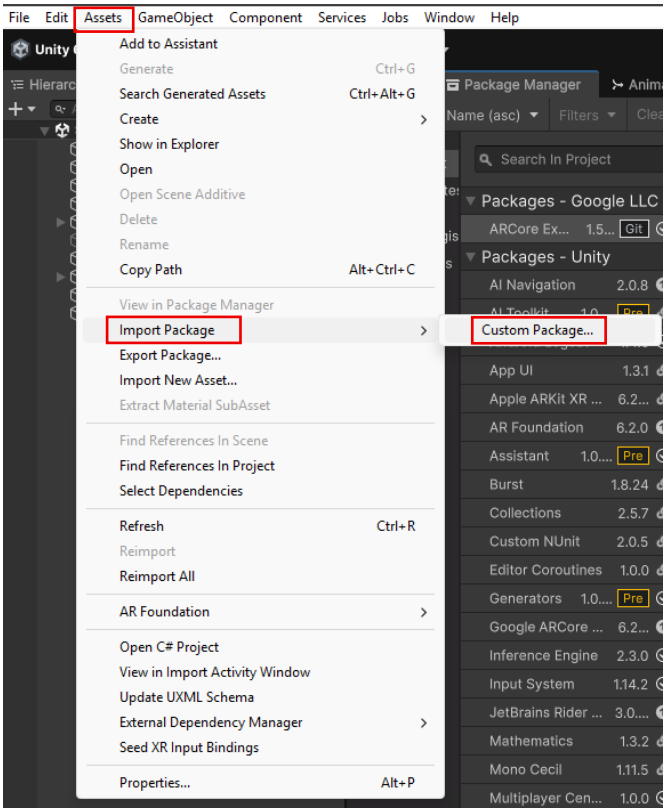
Από εκεί, ανέβηκε ένα JSON αρχείο με την δομή της βάσης και τα επιθυμητά στοιχεία που χρειαζόταν να τοποθετηθούν και μετά έγινε η σύνδεση της βάσης με το Unity project.

Αφού δημιουργήθηκε το project στο Firebase, μετά έγινε εγγραφή της εφαρμογής του Unity. Αυτό επιτεύχθηκε μέσα από το Firebase console στο οποίο έγινε επιλογή του“Add App”, όπως φαίνεται στο Σχήμα 3.23 και στην συνέχεια η επιλογή εικονιδίου του Unity. Στην συνέχεια επιλέχθηκε το Android και το όνομα πακέτου της εφαρμογής του Unity.



Σχήμα 3.23: Προσθήκη εφαρμογής Unity στο Firebase.

Παράλληλα, έγινε λήψη ενός JSON αρχείου, το οποίο τοποθετήθηκε στον φάκελο Assets στο Unity. Έπειτα, ακολούθησε η εγκατάσταση του Firebase Unity SDK. Αυτό επιτεύχθηκε από την επιλογή Unity →Assets → Import package → Custom package, όπως φαίνεται στο Σχήμα 3.24, και επιλέχθηκε το SDK που εγκαταστάθηκε πατώντας το Import.



Σχήμα 3.24: Προσθήκη SDK πακέτου στο Unity.

Για την διαδικασία σύνδεσης του Firebase με το Unity βοήθησαν οι οδηγίες από τον ιστότοπο του Firebase [17].

3.6 Επίλογος

Σε αυτό το κεφάλαιο αναλύθηκαν και εξηγήθηκαν οι διαδικασίες σύνδεσης άλλων τεχνολογιών και πακέτων με το κύριο project στο Unity. Αυτό το κεφάλαιο θα διευκολύνει την κατανόηση των λειτουργιών της εφαρμογής στο επόμενο κεφάλαιο.

Κεφάλαιο 4ο: Η εφαρμογή AR

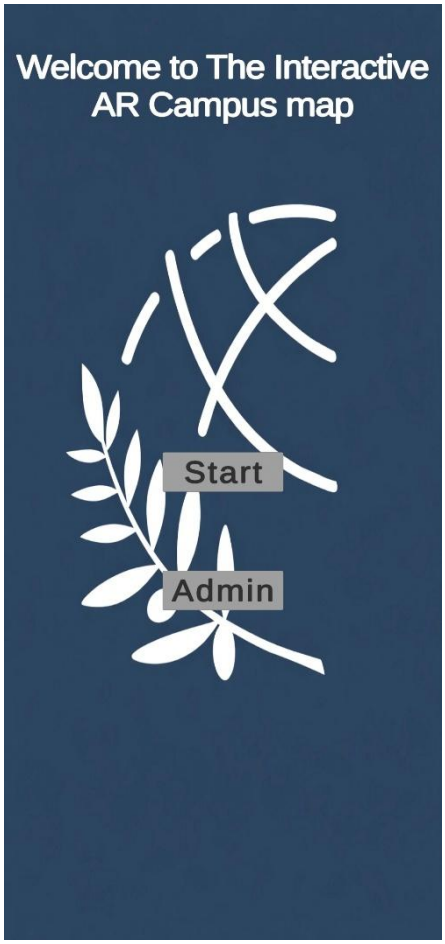
4.1 Εισαγωγή

Αυτό το κεφάλαιο επικεντρώνεται στην παρουσίαση εφαρμογής, αναλύοντας τους στόχους της, τους ορισμούς, και το πως θα βοηθήσει τους φοιτητές που θα την χρησιμοποιούν.

4.2 Σκοπός σχεδίασης της εφαρμογής

Η εφαρμογή αυτή φτιάχτηκε για τους φοιτητές της Αλεξάνδρειας Πανεπιστημιούπολης και προς το παρόν για τους φοιτητές του τμήματος Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων. Η εφαρμογή έχει στόχο να βοηθήσει τους φοιτητές να ενημερώνονται για τα μαθήματα που γίνονται σε κάθε αίθουσα στα κτήρια της πληροφορικής και της ηλεκτρονικής. Συγκεκριμένα, οι φοιτητές θα έχουν την δυνατότητα να βλέπουν τρισδιάστατα μοντέλα στον πραγματικό χώρο μέσω των κινητών συσκευών τους. Τα τρισδιάστατα μοντέλα είναι οι πίνακες που θα εμφανίζονται έξω από κάθε αίθουσα, και στους οποίους θα μπορούν οι φοιτητές να δουν συγκεκριμένα ποια μαθήματα γίνονται στην αίθουσα, από ποιον καθηγητή, την ημέρα και την ώρα, καθώς και το email του καθηγητή.

Το περιβάλλον ανάπτυξης που χρησιμοποιήθηκε για την ανάπτυξη της εφαρμογής είναι το Unity.Το σχήμα 4.1 δείχνει την αρχική οθόνη καλωσορίσματος η οποία έχει δύο επιλογές, “Student” και “Admin”.



Σχήμα 4.1: Οθόνη καλωσορίσματος.

Η επιλογή “Admin” είναι η πιο βασική, αφού αφορά τους υπεύθυνους που θα διαχειρίζονται τις άγκυρες της εφαρμογής. Πατώντας το κουμπί “Admin”, ο χρήστης θα μεταβεί σε ένα Login form όπως φαίνεται στο Σχήμα 4.2 όπου θα πρέπει να εισάγει το Username και Password, έτσι ώστε να μπορέσει να μεταβεί στην οθόνη του διαχειριστή.

Σχήμα 4.2: Admin's login form.

Αφού ο χρήστης βάλει τα στοιχεία του, με το κουμπί “Login”, γίνεται ο έλεγχος ορθότητας των στοιχείων από την βάση, και αν είναι σωστά τότε ο χρήστης θα μεταβεί στην σκηνή διαχειριστή η οποία φαίνεται στο Σχήμα 4.3.

Σχήμα 4.3: Σκηνή διαχειριστή.

Μόλις ο χρήστης μεταβεί στην σκηνή του διαχειριστή ξεκινάει η σάρωση και η χαρτογράφηση του πραγματικού χώρου όπου εμφανίζονται τα "planes" τα οποία είναι χρώμα κίτρινο (Σχήμα 4.3). Συγκεκριμένα, τα planes αναγνωρίζουν τις επιφάνειες, τους τοίχους, την οροφή και στη συνέχεια τα χαρτογραφούν. Όπου υπάρχει χρώμα κίτρινο σημαίνει ότι είναι τα αναγνωρισμένα στοιχεία στον πραγματικό χώρο και έχουν χαρτογραφηθεί. Επίσης, τα planes δίνουν την δυνατότητα τοποθέτησης τρισδιάστατων μοντέλων.

Για την τοποθέτηση μιας άγκυρας, γίνεται η επιλογή του dropdown μενού που φαίνεται στο Σχήμα 4.4, όπου μπορεί να επιλεχθεί το επιθυμητό κτίριο.

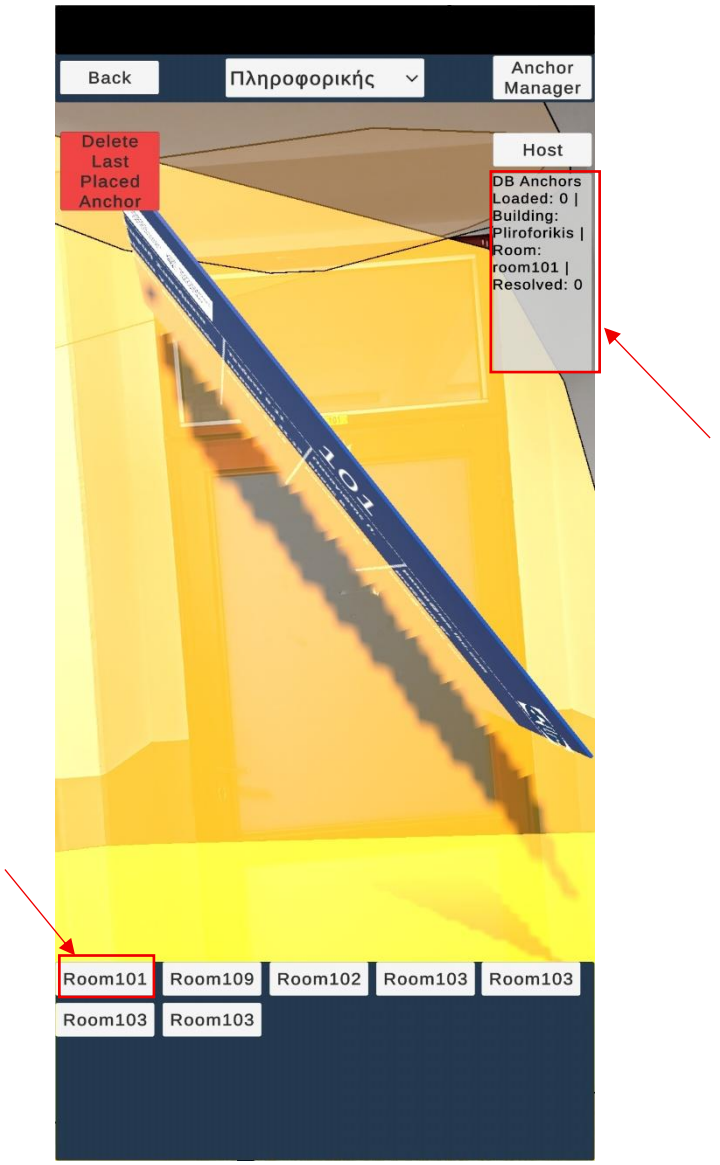
Σχήμα 4.4: Dropdown menu.

Επιλέγοντας κάποιο κτίριο, εμφανίζονται στο κάτω μέρος οι επιλογές αιθουσών του συγκεκριμένου κτιρίου, όπως μπορεί να φανεί στα Σχήματα 4.5 και 4.6. Στα παρόν στιγμιότυπα οθόνης δεν βρίσκονται όλες οι αίθουσες.

Σχήμα 4.5: Επιλογές κτιρίου πληροφορικής.

Σχήμα 4.6: Επιλογές κτιρίου ηλεκτρονικής.

Όταν επιλεχθεί μία αίθουσα, φαίνεται στο κουτάκι κάτω από το κουμπί “Host”, για να βεβαιωθεί ότι έγινε σωστή επιλογή, και τότε ο χρήσης είναι έτοιμος να πατήσει στο σημείο που θέλει για να τοποθετήσει την άγκυρα. Στην συνέχεια, μπορεί να πατήσει και να τοποθετήσει τον πίνακα. Αυτά μπορούν να φανούν στο Σχήμα 4.7. Τέλος, όταν γίνεται επιλογή μίας αίθουσας, τα στοιχεία της συγκεκριμένης αίθουσας εισέρχονται από την βάση.

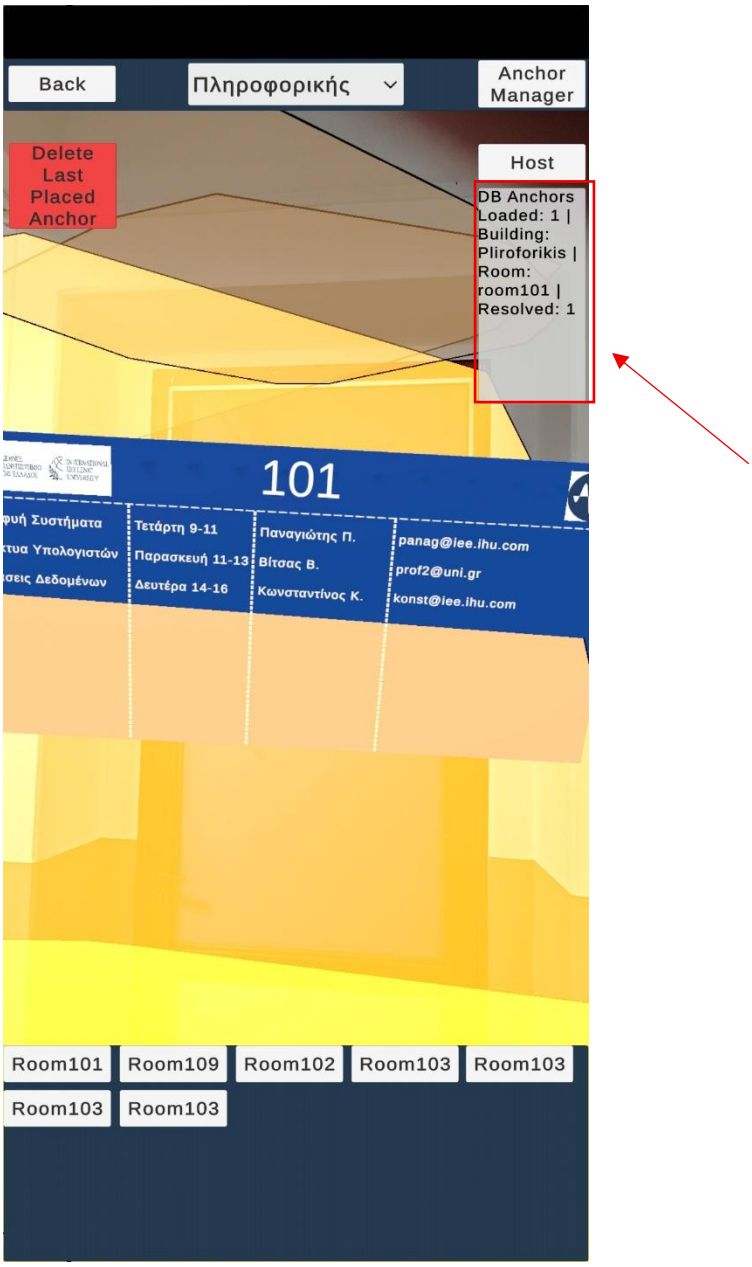


Σχήμα 4.7: Επιλογή και τοποθέτηση άγκυρας.

Αν σε περίπτωση που ο χρήστης τοποθετήσει σε λάθος μέρος την άγκυρα, τότε επιλέγει το κουμπί “Delete Last Placed Anchor” και την αφαιρεί. Όπως παρατηρείται, ο πίνακας δεν είναι σωστά τοποθετημένος, αλλά υπάρχει η δυνατότητα να περιστραφεί ο πίνακας και να εφαρμοστεί όπως επιθυμεί ο χρήστης. Στο σχήμα 4.8, φαίνεται η δημιουργία ενός ευδιάκριτου πίνακα.

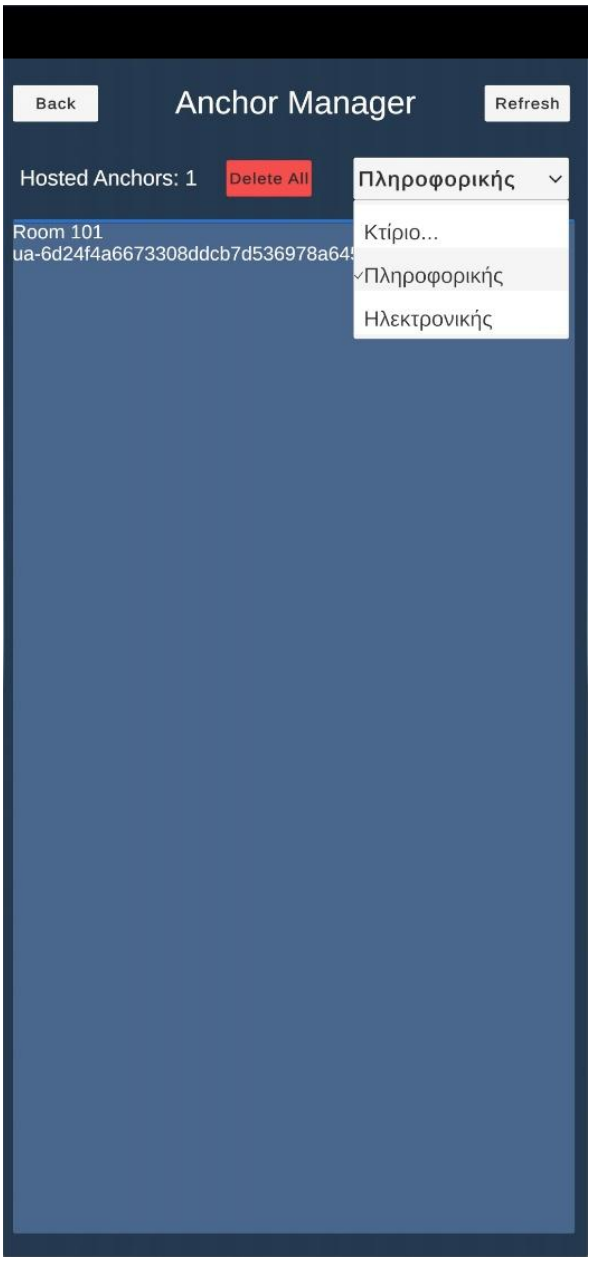
Σχήμα 4.8: Περιστροφή.

Στην συνέχεια πατώντας το κουμπί “Host” γίνεται host η άγκυρα στο Google Cloud έτσι ώστε να αποθηκευτεί η χαρτογράφηση και η περιστροφή που έγινε με ένα μοναδικό Id. Έπειτα, γίνεται αποθήκευση του Id στην βάση δεδομένων. Στο σχήμα 4.9, μπορεί να φανεί ότι το hosting έγινε με επιτυχία, βλέποντας στο κουτάκι κάτω από το κουμπί “Host”, ότι ο αριθμός των αγκυρών στην βάση αυξήθηκε κατά ένα.



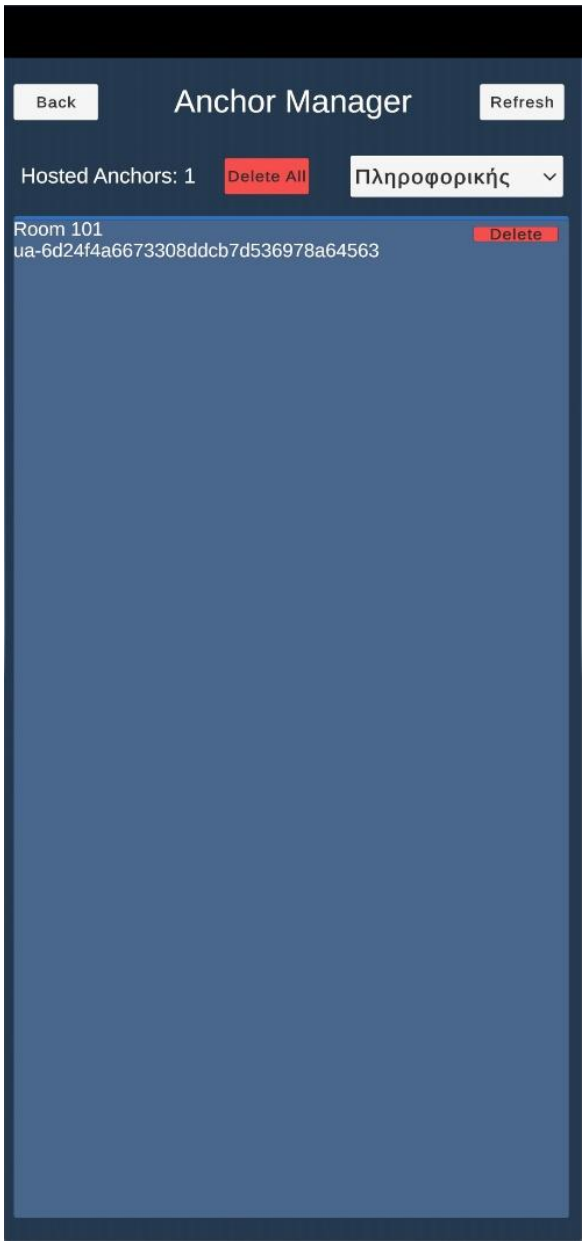
Σχήμα 4.9: Host με επιτυχία.

Για την διαχείριση των αγκυρών που υπάρχουν στην βάση, γίνεται επιλογή του “Anchor Manager” όπου μεταφέρει τον χρήστη στην σκηνή διαχείρισης αγκύρων. Στο σχήμα 4.10 φαίνεται η σκηνή, στην οποία υπάρχει ένα dropdown μενού και μπορεί να επιλεχθεί κάποιο κτίριο για να φανούν οι αποθηκευμένες αίθουσες που είναι στην βάση.

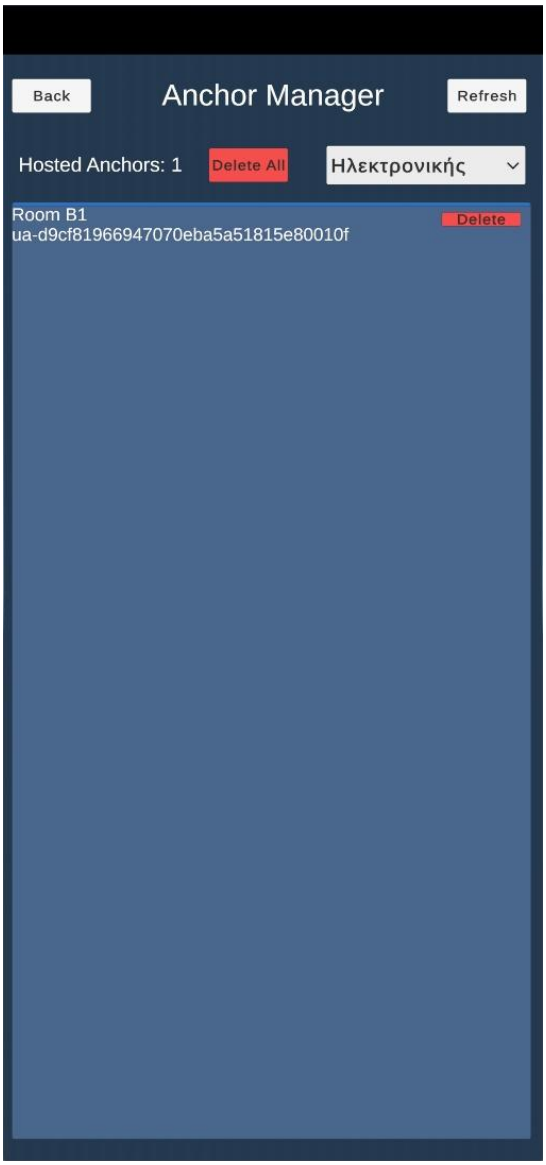


Σχήμα 4.10: Dropdown μενού.

Σε αυτή την σκηνή μπορεί να γίνει διαγραφή κάποιας μη επιθυμητής άγκυρας με την επιλογή “Delete” που βρίσκετε στα δεξιά του Id. Επίσης μπορεί να γίνει διαγραφή όλων των αγκυρών του συγκεκριμένου κτιρίου με την επιλογή “Delete all” που είναι δίπλα από το πλήθος των αγκύρων. Το κουμπί “Refresh” χρησιμοποιείτε για να φορτωθούν ξανά τα δεδομένα από την βάση σε περίπτωση που κάτι δεν φορτωθεί σωστά στην αρχή (Σχήμα 4.11, 4.12).



Σχήμα 4.11: Διαχείριση αγκύρων πληροφορικής.

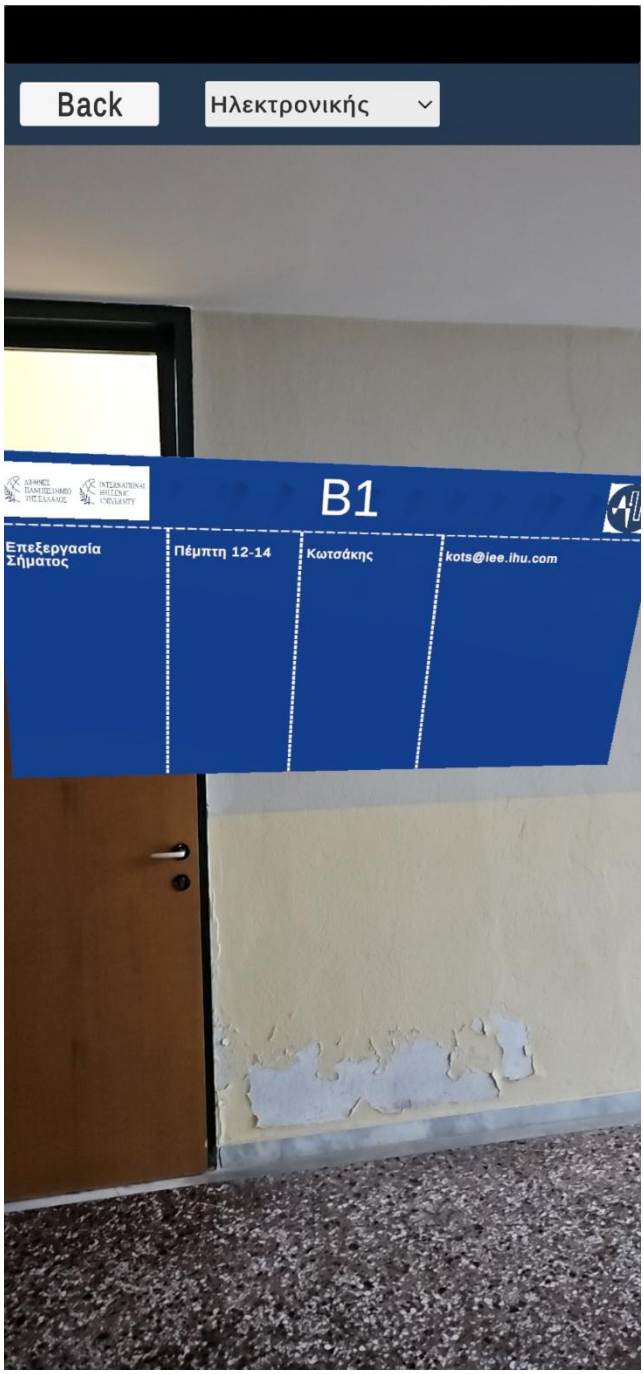


Σχήμα 4.12: Διαχείριση αγκυρών ηλεκτρονικής.

Τα πιο πάνω περιγράφουν την προβολή της εφαρμογής με την επιλογή “Admin”. Ωστόσο, σημαντική επίσης είναι και η εφαρμογή από την πλευρά του φοιτητή. Πατώντας το κουμπί “Student” (Σχήμα 4.1), προβάλλεται η σκηνή του φοιτητή η οποία φαίνεται και στο Σχήμα 4.13. Σε αυτή την σκηνή ο φοιτητής επιλέγει το κτίριο που βρίσκετε εκείνη την στιγμή και αυτόματα γίνεται η διευθέτηση για όλες τις άγκυρες του συγκεκριμένου κτιρίου, οι οποίες φαίνονται στα Σχήματα 4.14 και 4.15.

Σχήμα 4.13: Σκηνή φοιτητή και dropdown μενού.

Σχήμα 4.14: Επιλογή και resolving αιθουσών πληροφορικής



Σχήμα 4.15: Επιλογή και επίλυση αιθουσών ηλεκτρονικής.

Όλα τα δεδομένα που παρουσιάστηκαν πιο πάνω (Μαθήματα, καθηγητές, ημέρες, ώρες, emails καθηγητών), είναι ενδεικτικά και δεν ισχύουν. Θα γίνει η εισαγωγή των πραγματικών στοιχείων αργότερα για όλες τις αίθουσες και των δύο κτιρίων.

4.3 Λειτουργία της εφαρμογής στο πίσω μέρος

Στην συνέχεια θα γίνει η ανάλυση της λογικής του κώδικα που βρίσκετε από πίσω για την καλύτερη κατανόηση των λειτουργιών που παρουσιάστηκαν πιο πάνω.

Στο έργο υπάρχουν εννέα διαφορετικά scripts:

CloudAnchorPlacer.cs

SimpleAnchorManipulator.cs

AnchorManagerUI.cs

RoomInfoDB.cs

RoomSelectButton.cs

AnchorRowItem.cs

AdminUILogin.cs

DropDownManager.cs

UIManager.cs

4.3.1 Βάση Δεδομένων και Οργάνωση Πληροφορίας

Η εφαρμογή αξιοποιεί τη Firebase Realtime Database ως κεντρικό μηχανισμό αποθήκευσης και ανάκτησης δεδομένων. Η βάση δεδομένων είναι οργανωμένη σε ιεραρχική δομή τύπου JSON, η οποία επιτρέπει την αποδοτική ομαδοποίηση της πληροφορίας με βάση τη φυσική διάταξη του πανεπιστημιακού χώρου, δηλαδή κτίριο – αίθουσα – περιεχόμενο. Η συγκεκριμένη προσέγγιση διευκολύνει τόσο τη διαχείριση των δεδομένων όσο και τη δυναμική ανάκτησή τους από την εφαρμογή επαυξημένης πραγματικότητας.

Στο ανώτερο επίπεδο της βάσης περιλαμβάνονται δύο βασικές οντότητες: οι χρήστες και τα κτίρια. Η οντότητα users στο Σχήμα 4.16, χρησιμοποιείται για την αποθήκευση στοιχείων αυθεντικοποίησης του διαχειριστή της εφαρμογής, ο οποίος διαθέτει αυξημένα δικαιώματα, όπως η δημιουργία και φιλοξενία Cloud Anchors. Η οντότητα buildings περιλαμβάνει όλα τα διαθέσιμα κτίρια του συστήματος και αποτελεί το βασικό σημείο εκκίνησης για την πλοήγηση του χρήστη στο περιεχόμενο της εφαρμογής.

Κάθε κτίριο αναπαρίσταται από ένα μοναδικό αναγνωριστικό, όπως “Pliroforikis” και “Hlektronikis” και περιλαμβάνει περιγραφικά στοιχεία, καθώς και μια συλλογή αιθουσών (rooms). Κάθε αίθουσα διαθέτει τον αριθμό ή τον κωδικό της (roomNumber) και μια συλλογή μαθημάτων (courses) (Σχήμα 4.17). Για κάθε μάθημα αποθηκεύονται πληροφορίες όπως το όνομα του μαθήματος, η χρονική περίοδος διεξαγωγής, το όνομα του διδάσκοντα και το ηλεκτρονικό του ταχυδρομείο. Τα δεδομένα αυτά χρησιμοποιούνται για την ενημέρωση του χρήστη μέσω της διεπαφής της εφαρμογής, όταν αυτός αλληλοεπιδρά με μια συγκεκριμένη αίθουσα στο περιβάλλον AR.

Επιπλέον σε κάθε αίθουσα υπάρχει το node anchors, το οποίο προορίζεται για την αποθήκευση των Cloud Anchors που δημιουργούνται από τον διαχειριστή της εφαρμογής (Σχήμα 4.17). Κάθε Cloud Anchor αποθηκεύεται με μοναδικό αναγνωριστικό (cloudAnchorId) και συνοδεύεται από τοπικές παραμέτρους θέσης και προσανατολισμού, οι οποίες επιτρέπουν την ακριβή επανατοποθέτηση των εικονικών αντικειμένων στο φυσικό περιβάλλον κατά τη διαδικασία επίλυσης. Με τον τρόπο αυτό, η

βάση δεδομένων λειτουργεί ως συνδετικός κρίκος μεταξύ του φυσικού χώρου και των ψηφιακών αντικειμένων που προβάλλονται σε αυτόν.

Η συγκεκριμένη σχεδίαση της βάσης δεδομένων επιτρέπει την επεκτασιμότητα του συστήματος, καθώς μπορούν να προστεθούν νέα κτίρια, αίθουσες ή Cloud Anchors. Παράλληλα, διασφαλίζεται η ομαλή συνεργασία μεταξύ της βάσης δεδομένων και των υποσυστημάτων AR, προσφέροντας μια συνεκτική και δυναμική εμπειρία χρήστη.

```
{
  "users": {
    "admin1": {
      "id": "admin1",
      "username": "admin1",
      "password": "123456"
    }
  },
```

Σχήμα 4.17: Οντότητα Users.

```
{
  "buildings": {
    "Hlektronikis": {
      "name": "Τμήμα Ηλεκτρονικής",
      "rooms": {
        "B1": {
          "anchors": {
            "ua-d9cf81966947070eba5a51815e80010f": {
              "cloudAnchorId": "ua-d9cf81966947070eba5a51815e80010f",
              "createdAt": 1768320912,
              "localEuler": {
                "x": 69.7738265991211,
                "y": 81.68557739257812,
                "z": 0
              },
              "localPos": {
                "x": 0,
                "y": 0,
                "z": 0
              }
            }
          },
          "courses": {
            "course1": {
              "courseName": "Επεξεργασία Σήματος",
              "courseTime": "Πέμπτη 12-14",
              "teacherEmail": "kots@iee.ihu.com",
              "teacherName": "Κωτσάκης"
            }
          },
          "roomNumber": "B1"
        },
```

Σχήμα 4.16: Δομή κτιρίων, αιθουσών, μαθημάτων και Cloud Anchors.

```
9   public class RoomInfoRealtimeDBSimple : MonoBehaviour
10  {
11      [Header("Realtime DB")]
12      [Tooltip("Firebase building key")]
            8 references
13      public string buildingId;
14
15      [Tooltip("Room node key")]
            9 references
16      public string roomId;
17
18      [Header("UI Texts")]
            2 references
19      public TMP_Text roomNumberText;
            2 references
20      public TMP_Text courseNameText;
            2 references
21      public TMP_Text courseTimeText;
            2 references
22      public TMP_Text teacherNameText;
            2 references
23      public TMP_Text teacherEmailText;
24
            3 references
25      private DatabaseReference _dbRoot;
```

Σχήμα 4.18: Παράμετροι RoomInfoDB.

Το script RoomInfoDB είναι υπεύθυνο για την ανάκτηση και προβολή πληροφοριών μιας αίθουσας από το Firebase Realtime Database στην διεπαφή της εφαρμογής (TextMeshPro). Η λειτουργία του βασίζεται στο ζεύγος παραμέτρων buildingId και RoomId, οι οποίες καθορίζουν ποιο node της βάσης θα ανακτηθεί (Σχήμα 4.18). Οι τιμές αυτές δεν ορίζονται υποχρεωτικά μέσα στο ίδιο script, αλλά μπορούν να περαστούν δυναμικά από το CloudAnchoraPlacer μέσω της μεθόδου SetTarget(), ώστε το script να προσαρμόζεται στην αίθουσα που επιλέγεται ή επιλύεται κάθε φορά.

Κατά την εκκίνηση, το script εκτελεί ελέγχους μέσω της εντολής FirebaseApp.CheckAndFixDependenciesAsync() και στη συνέχεια αρχικοποιεί τη σύνδεση με τη Realtime Database, δημιουργώντας αναφορά στη ρίζα της βάσης (_dbRoot). Η πρόσβαση στα δεδομένα πραγματοποιείται σωστά μόνο όταν το Firebase έχει οριστεί έγκυρος στόχος. Ο έλεγχος εγκυρότητας υλοποιείται στη μέθοδο IsValidTarget(), η οποία αποτρέπει κλήσεις προς τη βάση δεδομένων σε περίπτωση κενών ή μη έγκυρων τιμών (Σχήμα 4.19).

```
63      private bool IsValidTarget()
64      {
65          if (string.IsNullOrWhiteSpace(buildingId) || string.IsNullOrWhiteSpace(roomId))
66          {
67              Debug.LogWarning($"[RoomInfo] Invalid target. buildingId='{buildingId}', roomId='{roomId}'");
68              return false;
69          }
70          return true;
71      }
```

Σχήμα 4.19: Μέθοδος IsValidTarget.

Η κύρια διαδικασία ανάκτησης υλοποιείται στη μέθοδο LoadRoomData() στο Σχήμα 4.20, όπου εκεί πραγματοποιείτε η ανάγνωση από το μονοπάτι buildings/{buildingId}/rooms/{roomId} με τη χρήση του GetValueAsync(). Όταν ολοκληρωθεί η ανάγνωση, τα δεδομένα που επιστρέφονται,

```
73      private void LoadRoomData()
74      {
75          if (_dbRoot == null)
76          {
77              Debug.LogWarning("[RoomInfo] DB root not ready yet.");
78              return;
79          }
80
81          Debug.Log($"[RoomInfo] Loading: buildings/{buildingId}/rooms/{roomId}");
82
83          _dbRoot.Child("buildings").Child(buildingId).Child("rooms").Child(roomId).GetValueAsync()
84              .ContinueWithOnMainThread(task =>
85              {
86                  if (task.IsFaulted)
87                  {
88                      Debug.LogError("[RoomInfo] Firebase read failed: " + task.Exception);
89                      return;
90                  }
91
92                  DataSnapshot snap = task.Result;
93                  if (!snap.Exists)
94                  {
95                      Debug.LogWarning($"[RoomInfo] Room not found in DB: buildings/{buildingId}/rooms/{roomId}");
96                      return;
97                  }
98
99                  string roomNumber = snap.Child("roomNumber").Value?.ToString() ?? roomId;
100                 if (roomNumberText != null)
101                     roomNumberText.text = roomNumber;
102
103                 string allCourseNames = "";
104                 string allCourseTimes = "";
105                 string allTeacherNames = "";
106                 string allTeacherEmails = "";
107
108                 DataSnapshot coursesNode = snap.Child("courses");
109
```

Σχήμα 4.20: Μέθοδος LoadRoomData.

χρησιμοποιούνται για την ενημέρωση των UI πεδίων. Συγκεκριμένα, ανακτάται ο αριθμός της αίθουσας και στην συνέχεια διαβάζεται ο κόμβος courses, ο οποίος μπορεί να περιέχει περισσότερες από μία εγγραφές μαθημάτων. Για κάθε μάθημα συλλέγονται τα πεδία όπως courseName, coursTime, teacherName και teacherEmail, τα οποία στη συνέχεια εμφανίζονται συγκεντρωτικά στο UI.

Τέλος, η χρήση του ContinueWithOnMainThread() στο σχήμα 4.20, διασφαλίζει ότι η ενημέρωση των στοιχείων της διεπαφής πραγματοποιείται στο κύριο νήμα της Unity, όπως απαιτείται για ασφαλή αλληλεπίδραση με UI αντικείμενα.

4.3.2 Αυθεντικοποίηση χρήστη και διαχωρισμός ρόλων

Η εφαρμογή περιλαμβάνει μηχανισμό αυθεντικοποίησης για τον διαχειριστή, έτσι ώστε να διαχωρίζονται οι λειτουργίες που σχετίζονται με τη δημιουργία και φιλοξενία των Cloud Anchors από τη λειτουργία προβολής και επίλυσης που χρησιμοποιείται από άλλους χρήστες. Κατά την είσοδο, ο χρήστης εισάγει username και password μέσω της διεπαφής. Η εφαρμογή ανακτά από την βάση δεδομένων την αντίστοιχη εγγραφή χρήστη και εκτελεί έλεγχο εγκυρότητας συγκρίνοντας τον αποθηκευμένο κωδικό με την τιμή εισόδου. Η διαδικασία αυτή γίνεται με την μέθοδο OnLoginButtonPressed() στο script AdminUILogin που φαίνεται στο σχήμα 4.21 και 4.22.

```
41      public void OnLoginButtonPressed()
42      {
43          if (!dbReady || db == null)
44          {
45              statusText.text = "Database not ready yet. Try again.";
46              return;
47          }
48
49          string username = usernameInput.text.Trim();
50          string password = passwordInput.text;
51
52          if (string.IsNullOrEmpty(username) || string.IsNullOrEmpty(password))
53          {
54              statusText.text = "Enter username and password.";
55              return;
56          }
57
58          statusText.text = "Checking login...";
59
60          db.Child("users").Child(username).GetValueAsync().ContinueWithOnMainThread(task =>
61          {
62              if (!task.IsCompleted)
63              {
64                  statusText.text = "Database error.";
65                  Debug.LogError(task.Exception);
66                  return;
67              }
68
69              DataSnapshot snapshot = task.Result;
70
71              if (!snapshot.Exists)
72              {
73                  statusText.text = "User not found.";
```

Σχήμα 4.21: Μέθοδος OnLoginButtonPressed.

```
77              string dbPassword = snapshot.Child("password").Value.ToString();
78
79              if (password == dbPassword)
80              {
81                  statusText.text = "Login successful!";
82                  Debug.Log("ADMIN LOGIN OK");
83                  SceneManager.LoadScene("SampleScene");
84              }
85              else
86              {
87                  statusText.text = "Wrong password.";
88              }
89          });
90      }
```

Σχήμα 4.22: Έλεγχος εγκυρότητας.

4.3.3 Επιλογή αίθουσας για έναρξη διαδικασίας

Η επιλογή αίθουσας στην εφαρμογή υλοποιείται μέσω UI κουμπιών, όπου κάθε κουμπί αντιστοιχεί σε μία συγκεκριμένη αίθουσα και κτίριο. Το script RoomSelectButton αναλαμβάνει να συνδέσει την ενέργεια του χρήστη, με το κεντρικό σύστημα διαχείρισης Cloud Anchors (CloudAnchorPlacer), μεταφέροντας τις απαραίτητες παραμέτρους επιλογής.

Κατά την αρχικοποίηση του αντικειμένου (Awake()), το script εντοπίζει το Button component και καταχωρεί callback στη λειτουργία onClick(). Έτσι, κάθε φορά που ο χρήστης πατά το αντίστοιχο UI κουμπί, καλείτε η μέθοδος onClicked().

Στη μέθοδο onClicked(), πραγματοποιείται αρχικά έλεγχος ότι έχει ανατεθεί σωστά αναφορά στο CloudAnchorPlacer (Σχήμα 4.23). Στη συνέχεια, ορίζεται το κτίριο και η αίθουσα που επέλεξε ο χρήστης μέσω δύο κλήσεων:

- SelectBuilding(buildingId): ενημερώνει το ενεργό κτίριο και ξεκινά τη διαδικασία φόρτωσης και προετοιμασίας των αγκυρών που σχετίζονται με το συγκεκριμένο κτίριο (Σχήμα 4.24).
- SetCurrentRoomId(roomId): ορίζει την τρέχουσα αίθουσα, ώστε οι επόμενες ενέργειες να αποθηκεύονται κάτω από το σωστό node στη βάση δεδομένων (Σχήμα 4.25).

Με αυτόν τον τρόπο, η επιλογή αίθουσας λειτουργεί ως «σύνδεσμος» μεταξύ της διεπαφής χρήστη και της λογικής του AR συστήματος, καθορίζοντας σε ποιο κτίριο/αίθουσα θα γίνει ανάκτηση δεδομένων, επίλυση των αγκυρών ή δημιουργία νέων αγκυρών απο τον διαχειριστή.

```
1   using UnityEngine;
2   using UnityEngine.UI;
3
    0 references
4   public class RoomSelectButton : MonoBehaviour
5   {
        1 reference
6       public string roomId = "room101";
        1 reference
7       public string buildingId = "Pliroforikis";
        3 references
8       public CloudAnchorPlacer cloudAnchorPlacer;
9
        0 references
10      private void Awake()
11      {
12          var btn = GetComponent<Button>();
13          if (btn != null)
14              btn.onClick.AddListener(OnClicked);
15      }
16
        1 reference
17      private void OnClicked()
18      {
19          if (cloudAnchorPlacer == null)
20          {
21              Debug.LogWarning("[RoomSelectButton] No CloudAnchorPlacer assigned.");
22              return;
23          }
24
25          cloudAnchorPlacer.SelectBuilding(buildingId);
26          cloudAnchorPlacer.SetCurrentRoomId(roomId);
27      }
```

Σχήμα 4.23: RoomSelectButton.

```
346     public void SelectBuilding(string buildingId)
347     {
348         if (string.IsNullOrEmpty(buildingId))
349         {
350             Debug.LogWarning("[CloudAnchors] SelectBuilding called with empty buildingId.");
351             return;
352         }
353
354         _currentBuildingId = buildingId;
355         Debug.Log("[CloudAnchors] Selected building: " + _currentBuildingId);
356
357         ClearResolvedOnly();
358         UpdateDebugStatus();
359
360         StartCoroutine(LoadAnchorsForBuildingThenQueueResolve(_currentBuildingId));
361     }
```

Σχήμα 4.24: Μέθοδος SelectBuilding.

```
671     public void SetCurrentRoomId(string roomId)
672     {
673         if (string.IsNullOrEmpty(roomId))
674         {
675             Debug.LogWarning("[CloudAnchors] Tried to set empty roomId.");
676             return;
677         }
678         _currentRoomId = roomId;
679         Debug.Log("[CloudAnchors] Placement roomId set to: " + _currentRoomId);
680     }
```

Σχήμα 4.25: Μέθοδος SetCurrentRoomId.

4.3.4 Τοποθέτηση άγκυρας μέσω αλληλεπίδρασης χρήστη (Tap & Place)

Η διαδικασία τοποθέτησης άγκυρας στο περιβάλλον AR πραγματοποιείται μέσω άμεσης αλληλεπίδρασης του διαχειριστή με την οθόνη της κινητής συσκευής. Η λειτουργικότητα αυτή θλοποιείται στο script CloudAnchorPlacer και ενεργοποιείται αποκλειστικά στη λειτουργία διαχειριστή, ώστε να αποτρέπεται η ακούσια δημιουργία αγκυρών από απλούς χρήστες.

Κατά την εκτέλεση της μεθόδου update(), το σύστημα ελέγχει αρχικά αν η εφαρμογή βρίσκεται σε Admin mode και αν πληρούνται οι απαραίτητες προϋποθέσεις, για παράδειγμα, δεν βρίσκεται σε εξέλιξη η διαδικασία hosting και επιτρέπεται η τοποθέτηση νέας άγκυρας. Όταν ο χρήστης αγγίξει την οθόνη, η εφαρμογή πραγματοποιεί raycast από τη θέση αφής προς το AR περιβάλλον, με στόχο την ανίχνευση επιπέδων επιφανειών (planes) που έχουν εντοπιστεί από το AR subsystem.

Εφόσον το raycast επιτύχει και εντοπίσει έγκυρη επιφάνεια, δημιουργείται τοπική άγκυρα (ARanchor) η οποία συνδέεται με το αντίστοιχο plane. Πάνω στην άγκυρα τοποθετείται ένα τοπικό αντικείμενο (prefab), το οποίο λειτουργεί ως αναπαράσταση της άγκυρας στο φυσικό περιβάλλον. Το αντικείμενο αυτό μπορεί να περιστραφεί ή να προσαρμοστεί από τον χρήστη πριν την τελική διαδικασία φιλοξενίας.

Η τοποθετημένη άγκυρα αποθηκεύεται προσωρινά στην μνήμη της εφαρμογής και χαρακτηρίζεται ως ενεργή άγκυρα, επιτρέποντας στον χρήστη να πραγματοποιήσει περαιτέρω ενέργειες, όπως περιστροφή ή επιβεβαίωση της τοποθέτησης. Η διαδικασία ολοκληρώνεται με την αποθήκευση των τοπικών μετασχηματισμών, οι οποίοι θα χρησιμοποιηθούν αργότερα κατά την φιλοξενία και επίλυση της άγκυρας.

Με αυτόν τον μηχανισμό, η εφαρμογή παρέχει έναν φυσικό και διαισθητικό τρόπο τοποθέτησης ψηφιακών ανυικειμένων στον πραγματικό χώρο, εξασφαλίζοντας παράλληλα ακρίβεια στη συσχέτιση του ψηφιακού περιεχομένου με το φυσικό περιβάλλον.

Στο Σχήμα 4.26 φαίνονται οι περιορισμοί πριν την τοποθέτηση του αντικειμένου, στην αρχή του Update(), ώστε να αποτραπεί το placement όταν ο χρήστης δεν είναι διαχειριστής, γίνεται ήδη hosting ή quality check, ή όταν έχει ήδη τοποθετηθεί άγκυρα και δεν επιτρέπεται δεύτερη.

```
592     private void Update()
593     {
594         UpdateDebugStatus();
595
596         if (!isAdminMode) return;
597         if (_checkingFeatureMapQuality) return;
598         if (!_canPlace) return;
599
```

Σχήμα 4.26: Ελέγχοι περιορισμών.

Η εφαρμογή χρησιμοποιεί raycasting από το σημείο αφής προς το AR περιβάλλον. Το touch αγνοείται όταν αφορά UI στοιχεία, ώστε να μην δημιουργούνται άγκυρες κατά τη χρήση κουμπιών ή dropdown. Αυτά γίνονται με την μέθοδο TouchBeganOnPlane(), όπως φαίνεται στο Σχήμα 4.27.

```
836     private bool TouchBeganOnPlane(out Pose pose)
837     {
838         pose = default;
839         if (ETouch.activeTouches.Count == 0) return false;
840
841         var t = ETouch.activeTouches[0];
842         if (t.phase != UnityEngine.InputSystem.TouchPhase.Began) return false;
843
844         if (EventSystem.current != null && EventSystem.current.IsPointerOverGameObject(t.touchId))
845             return false;
846
847         if (!_raycaster.Raycast(t.screenPosition, _hits, TrackableType.PlaneWithinPolygon))
848             return false;
849
850         pose = _hits[0].pose;
851         return true;
852     }
```

Σχήμα 4.27: Μέθοδος TouchBeganOnPlane.

Στο Σχήμα 4.28, με την εντολή var localAnchor = _anchorManager.AttachAnchor(plane, hit.pose) που βρίσκετε στην μέθοδο Update(), η άγκυρα δημιουργείται συσχετισμένη με το plane που ανιχνεύθηκε, ώστε να ακολουθεί τη χωρική κατανόηση του AR συστήματος και να είναι σταθερή στον πραγματικό χώρο. Αμέσως μετά στις γραμμές 625-627, γίνεται instance του prefab ως «παιδί» της άγκυρας, για να φαίνεται στον χρήστη πού τοποθετήθηκε.

```
600         if (TouchBeganOnPlane(out Pose pose))
601         {
602             if (_lastTapFrame == Time.frameCount) return;
603             _lastTapFrame = Time.frameCount;
604 
605             var hit = _hits[0];
606             var plane = _planeManager.GetPlane(hit.trackableId);
607             if (plane == null)
608             {
609                 Debug.LogWarning("[CloudAnchors] No plane for this hit.");
610                 return;
611             }
612 
613             var localAnchor = _anchorManager.AttachAnchor(plane, hit.pose);
614             if (localAnchor == null)
615             {
616                 Debug.LogWarning("[CloudAnchors] AttachAnchor failed.");
617                 return;
618             }
619 
620             _localAnchors.Add(localAnchor);
621 
622             GameObject vis;
623             if (prefab != null)
624             {
625                 vis = Instantiate(prefab, localAnchor.transform);
626                 vis.transform.localPosition = Vector3.zero;
627                 vis.transform.localRotation = Quaternion.identity;
```

Σχήμα 4.28: Δημιουργία άγκυρας.

Μετά την τοποθέτηση, η άγκυρα αποθηκεύεται ως η τελευταία τοποθετημένη άγκυρα και απενεργοποιείται προσωρινά η δυνατότητα νέας τοποθέτησης, ώστε να αποφευχθούν πολλαπλές άγκυρες χωρίς επιβεβαίωση. Οι εντολές βρίσκονται στο Σχήμα 4.29.

```
653                 _lastPlacedAnchor = localAnchor;
654                 _canPlace = false;
```

Σχήμα 4.29: Μπλοκάρισμα τοποθέτησης νέας άγκυρας.

4.3.5 Περιστροφή του αντικειμένου πριν το Hosting

Μετά την τοποθέτηση της τοπικής άγκυρας, η εφαρμογή επιτρέπει στον διαχειριστή να προσαρμόσει τον προσανατολισμό του αντικειμένου πριν αυτο φιλοξενηθεί ως Cloud Anchor. Η λειτουργικότητα περιστροφής υλοποιείτε μέσω του SimpleAnchorManipulator, το οποίο εφαρμόζεται πάνω στο οπτικό αντικείμενο της άγκυρας και επιτρέπει την περιστροφή με χειρονομία drag στην οθόνη.

Όπως μπορεί να φανεί από το Σχήμα 4.30, μέσα στην μέθοδο Update(), γίνεται ο έλεγχος για το αν ο χρήστης είναι ο διαχειριστής, και τότε έχει το δικαίωμα να περιστρέψει το αντικείμενο. Στην μέθοδο setActive(), είναι ο μηχανισμός αποκλειστικότητας ώστε να μην περιστρέφονται πολλά objects ταυτόχρονα. Μέσα στο Update(), στις γραμμές 45-50, δείχνει ότι η περιστροφή ενεργοποιείται μόνο με ένα δάχτυλο και drag. Στην μέθοδο RotateTarget δείχνει το πώς μετατρέπεται το drag σε γωνιακή περιστροφή.

```
29      public void SetActive(bool active)
30      {
31          if (active)
32              Active = this;
33          else if (Active == this)
34              Active = null;
35      }
36
            0 references
37      private void Update()
38      {
39          if (!CloudAnchorPlacer.IsAdminScene)
40              return;
41
42          if (Active != this)
43              return;
44
45          if (Touch.activeTouches.Count != 1)
46              return;
47
48          var touch = Touch.activeTouches[0];
49          if (touch.phase == UnityEngine.InputSystem.TouchPhase.Moved)
50              RotateTarget(touch.delta);
51      }
52
            1 reference
53      private void RotateTarget(Vector2 delta)
54      {
55          if (target == null) return;
56
57          float yawDelta = delta.x * rotationSpeed;
58
59          float pitchDelta = -delta.y * rotationSpeed;
60          currentPitch = Mathf.Clamp(currentPitch + pitchDelta, minDownAngle, maxUpAngle);
61
62          float yaw = target.localEulerAngles.y + yawDelta;
63          target.localRotation = Quaternion.Euler(currentPitch, yaw, 0f);
64      }
```

Σχήμα 4.30: SimpleAnchorManipulator script.

4.3.6 Διαδικασία φιλοξενείας άγκυρας για να μετατραπεί σε Cloud Anchor.

Μετά την τοποθέτηση της τοπικής άγκυρας και την προσαρμογή του προσανατολισμού του αντικειμένου, ο διαχειριστής μπορεί να μετατρέψει την άγκυρα σε Cloud Anchor μέσω της διαδικασίας hosting. Στο σύστημα, το hosting εκκινείτε όταν ο διαχειριστής πατήσει το κουμπί “Host”, το οποίο καλεί την μέθοδο HostCurrentAnchor() του script CloudAnchorPlacer.

Η μέθοδος HostCurrentAnchor() στο Σχήμα 4.31, ελέγχει αρχικά ότι η εφαρμογή βρίσκεται σε Admin mode, υπάρχει τοποθετημένη άγκυρα διαθέσιμη για hosting και δεν εκτελείται ήδη άλλη διαδικασία hosting.

Εφόσον οι παραπάνω συνθήκες ισχύουν, ενεργοποιείται courtine (CheckFeatureMapAndHostCoroutine) που ελέγχει την ποιότητα χαρτογράφησης χαρακτηριστικών (feature map) του περιβάλλοντος.

```
692     public void HostCurrentAnchor()
693     {
694         if (!isAdminMode)
695         {
696             Debug.LogWarning("[CloudAnchors] HostCurrentAnchor called in student mode. Ignored.");
697             return;
698         }
699
700         if (_lastPlacedAnchor == null)
701         {
702             Debug.LogWarning("[CloudAnchors] No placed anchor to host yet.");
703             return;
704         }
705
706         if (_checkingFeatureMapQuality)
707         {
708             Debug.LogWarning("[CloudAnchors] Already checking feature map / hosting in progress.");
709             return;
710         }
711
712         Debug.Log("[CloudAnchors] HOST pressed. Starting feature-map quality check & hosting…");
713         StartCoroutine(CheckFeatureMapAndHostCoroutine(_lastPlacedAnchor));
714     }
```

Σχήμα 4.31: Μέθοδος HostCurrentAnchor.

Η ποιότητα αυτή εκτιμάται μέσω της EstimateFeatureMapQualityForHosting, η οποία βασίζεται στην τρέχουσα θέση της κάμερας και καθορίζει αν υπάρχουν αρκετά οπτικά χαρακτηριστικά στον χώρο έτσι ώστε το Cloud Anchor να μπορεί να επανα-εντοπιστεί αξιόπιστα σε άλλες συσκευές. Το Σχήμα 4.32, δείχνει ότι όταν η ποιότητα κριθεί επαρκής, η εφαρμογή ξεκινά το hosting με την κλήση HostCloudAnchorAsync, χρησιμοποιώντας επίσης τιμή TTL (Days To Live) που ορίζει τη διάρκεια ζωής του anchor στο cloud. Αν το hosting ολοκληρωθεί επιτυχώς, επιστρέφεται ένα μοναδικό αναγνωριστικό για την επίλυση της άγκυρας σε άλλες συσκευές.

```
716     private IEnumerator CheckFeatureMapAndHostCoroutine(ARAnchor anchor)
717     {
718         _checkingFeatureMapQuality = true;
719
720         const float timeoutSeconds = 20f;
721         const float checkInterval = 0.5f;
722         float remaining = timeoutSeconds;
723
724         yield return new WaitForSeconds(0.5f);
725
726         while (remaining > 0f)
727         {
728             var cam = Camera.main;
729             if (cam == null) break;
730
731             var pose = new Pose(cam.transform.position, cam.transform.rotation);
732             var quality = ARAnchorManagerExtensions.EstimateFeatureMapQualityForHosting(_anchorManager, pose);
733             Debug.Log($"[CloudAnchors] FeatureMapQuality: {quality}.");
734
735             if (quality == FeatureMapQuality.Sufficient || quality == FeatureMapQuality.Good)
736             {
737                 int ttl = Mathf.Clamp(daysToLive, 1, 365);
738                 var hp = ARAnchorManagerExtensions.HostCloudAnchorAsync(_anchorManager, anchor, ttl);
```

Σχήμα 4.32: Μέθοδος CheckFeatureMapAndHostQuality.

Μετά από επιτυχές hosting, το σύστημα αποθηκεύει στη βάση δεδομένων, το cloudAnchorId, το roomId, το buildingId ώστε να συνδέεται με την σωστή αίθουσα και αποθηκεύει επίσης την τοπική θέση localPos και τον προσανατολισμό localEuler του οπτικού αντικειμένου στην άγκυρα (Σχήμα 4.33). Έτσι, όταν οι άγκυρες φορτωθούν ξανά απο την βάση και επιλυθούν, το αντικείμενο μπορεί να εμφανιστεί στον ίδιο πραγματικό χώρο με την ίδια ακριβώς ευθυγράμμιση (Σχήμα 4.34).

```
751             if (result.CloudAnchorState == CloudAnchorState.Success)
752             {
753                 string id = result.CloudAnchorId;
754                 if (!string.IsNullOrEmpty(id))
755                 {
756                     _anchorToVisual.TryGetValue(anchor, out var vis);
757
758                     Vector3 localPos = vis ? vis.transform.localPosition : Vector3.zero;
759                     Vector3 localEuler = vis ? vis.transform.localEulerAngles : Vector3.zero;
760
```

Σχήμα 4.34: Επιτυχής διαδικασία hosting.

```
822             SaveAnchorToFirebase(_currentBuildingId, roomIdToSave, id, localPos, localEuler);
823             Debug.Log($"[CloudAnchors] Hosted OK (fallback). ID={id}, building={_currentBuildingId}, roomId={roomIdToSave}");
```

Σχήμα 4.33: Αποθήκευση άγκυρας στην βάση δεδομένων.

4.3.7 Επίλυση αγκυρών και λειτουργία Dropdown

Η επιλογή κτιρίου από τον χρήστη πραγματοποιείται μέσω ενός κοινού στοιχείου διεπαφής dropdown μενού, στο οποίο είναι συνδεδεμένα δύο ανεξάρτητα scripts, το CloudAnchorPlacer και το DropDownManager.

Το CloudAnchorPlacer αξιοποιεί την επιλογή αυτή για την ενεργοποίηση της επιχειρησιακής λογικής της εφαρμογής, όπως η φόρτωση και επίλυση Cloud Anchors από τη βάση δεδομένων. Παράλληλα το DropDownManager χρησιμοποιεί την ίδια πληροφορία αποκλειστικά για την διαχείριση της διεπαφής χρήστη, ενεργοποιώντας ή απενεργοποιώντας τα αντίστοιχα panels των αιθουσών.

Στο Σχήμα 4.35, φαίνονται οι εισόδοι που λαμβάνει το DropDownManager script, μία εκ των οποίων είναι η είσοδος του χρήστη από ένα στοιχείο τύπου dropdown όπως φαίνεται στην γραμμή 7, και ενεργοποιεί τα αντίστοιχα panels αιθουσών στις γραμμές 10 και 11. Επίσης στις γραμμές 14 και 15 λαμβάνει δύο strings τα οποία είναι τα ονόματα των κτιρίων στα ελληνικά.

```
1   using TMPro;
2   using UnityEngine;
3
    0 references
4   public class BuildingPanelSwitcher : MonoBehaviour
5   {
6       [Header("UI")]
        7 references
7       public TMP_Dropdown buildingDropdown;
8
9       [Header("Panels")]
        2 references
10      public GameObject panelPliroforikis;
        2 references
11      public GameObject panelHlektronikis;
12
13      [Header("Dropdown labels")]
        1 reference
14      public string pliroforikisLabel = "Πληροφορικής";
        1 reference
15      public string hlektronikisLabel = "Ηλεκτρονικής";
```

Σχήμα 4.35: Είσοδοι DropDownManager script

Στο σχήμα 4.36, παρουσιάζεται η μέθοδος OnBuildingChanged(), η οποία κάθε φορά που ο χρήστης αλλάζει την επιλογή κτιρίου μέσω του dropdown, όπου ανακτάται το κείμενο της επιλεγμένης επιλογής, όπως αυτή εμφανίζεται στον χρήστη, και χρησιμοποιείται ως κριτήριο λήψης αποφάσεων διεπαφή στην γραμμή 35. Στις γραμμές 37-38 γίνεται η επιλογή του χρήστη και συγκρίνεται με προκαθορισμένες ετικέτες, οι οποίες αντιστοιχούν στα διαθέσιμα κτίρια της εφαρμογής, ώστε να προσδιοριστεί ποιο τμήμα της διεπαφής θα εμφανιστεί. Στις γραμμές 40-41, ενεργοποιούνται η απενεργοποιούνται δυναμικά τα αντίστοιχα panels με βάση το αποτέλεσμα της σύγκρισης, επιτρέποντας τη στοχευμένη προβολή μόνο των σχετικών επιλογών προς τον χρήστη.

```
30      void OnBuildingChanged(int index)
31      {
32          if (buildingDropdown == null || buildingDropdown.options == null || buildingDropdown.options.Count == 0)
33              return;
34
35          string selectedText = buildingDropdown.options[index].text;
36
37          bool showPliro = selectedText == pliroforikisLabel;
38          bool showHlektro = selectedText == hlektronikisLabel;
39
40          if (panelPliroforikis != null) panelPliroforikis.SetActive(showPliro);
41          if (panelHlektronikis != null) panelHlektronikis.SetActive(showHlektro);
42
43
44          Debug.Log($"[BuildingPanelSwitcher] Selected '{selectedText}' -> Pliro={showPliro}, Hlektronikis={showHlektro}");
45      }
46  }
47
```

Σχήμα 4.36: Μέθοδος OnBuildingChange.

Η μέθοδος αυτή λειτουργεί ανεξάρτητα από τη λογική διαχείρισης Cloud Anchors, καθώς περιορίζεται αποκλειστικά στη διαχείριση της διεπαφής χρήστη. Η ίδια ενέργεια επιλογής κτιρίου ενεργοποιεί παράλληλα τη λογική φόρτωσης δεδομένων μέσω του CloudAnchorPlacer, χωρίς άμεση εξάρτηση μεταξύ των δύο scripts.

Στο CloudAnchorPlacer στο Σχήμα 4.37, μέσα στην μέθοδο start(), το script παρακολουθεί το dropdown μενού για να ξεκινήσει τις διαδικασίες δεδομένων των αγκυρών.

```
186         if (buildingDropdown != null)
187         {
188             SetupBuildingDropdown();
189             buildingDropdown.onValueChanged.RemoveListener(OnBuildingDropdownChanged);
190             buildingDropdown.onValueChanged.AddListener(OnBuildingDropdownChanged);
191         }
```

Σχήμα 4.37: Παρακολούθηση του dropdown.

Στο σχήμα 4.38, το απόσπασμα κώδικα δείχνει πως η μέθοδος OnBuildingDropdownChanged(), ενεργοποιείται κάθε φορά που ο χρήστης επιλέγει κάποιο κτίριο μέσω του dropdown. Η μέθοδος λαμβάνει ως είσοδο τη θέση (index) της επιλεγμένης επιλογής και αποτελεί το σημείο διασύνδεσης μεταξύ της διεπαφής χρήστη και της λογικής της εφαρμογής.

```
2 references
264 public void OnBuildingDropdownChanged(int dropdownIndex)
265 {
266     int realIndex = dropdownIndex - 1;
267
268     if (realIndex < 0)
269     {
270         Debug.Log("[CloudAnchors] Dropdown placeholder selected. Not resolving.");
271         return;
272     }
273
274     if (realIndex >= buildingOptions.Count)
275         return;
276
277     string buildingId = buildingOptions[realIndex].buildingId;
278
279     if (string.IsNullOrEmpty(buildingId))
280     {
281         Debug.LogWarning("[CloudAnchors] Dropdown mapped buildingId is empty. Check buildingOptions in Inspector.");
282         return;
283     }
284
285     SelectBuilding(buildingId);
286 }
```

Σχήμα 4.38: Μέθοδος OnBuildingdropdownChanged.

Στη συνέχεια, στο σχήμα 4.39 παρουσιάζεται η επιλογή του χρήστη που αντιστοιχίζεται σε ένα εσωτερικό αναγνωριστικό κτιρίου (buildingId), το οποίο χρησιμοποιείτε από το σύστημα για την ανάκτηση των αντίστοιχων δεδομένων από τη βάση Firebase. Τέλος, καλείται η μέθοδος SelectBuilding(), η οποία ενεργοποιεί τη διαδικασία φόρτωσης και επίλυσης των Cloud Anchors που σχετίζονται με το επιλεγμένο κτίριο.

```
346     public void SelectBuilding(string buildingId)
347     {
348         if (string.IsNullOrEmpty(buildingId))
349         {
350             Debug.LogWarning("[CloudAnchors] SelectBuilding called with empty buildingId.");
351             return;
352         }
353
354         _currentBuildingId = buildingId;
355         Debug.Log("[CloudAnchors] Selected building: " + _currentBuildingId);
356
357         ClearResolvedOnly();
358         UpdateDebugStatus();
359
360         StartCoroutine(LoadAnchorsForBuildingThenQueueResolve(_currentBuildingId));
361     }
362
```

Σχήμα 4.39: Μέθοδος SelectBuilding.

Η μέθοδος LoadAnchorsForBuildingThenQueueResolve που φαίνεται στο σχήμα 4.40 και 4.41, είναι υπεύθυνη για την φόρτωση των αποθηκευμένων Cloud Anchors που αντιστοιχούν σε ένα συγκεκριμένο κτίριο και την προετοιμασία τους για επίλυση στο περιβάλλον AR. Η συνάρτηση υλοποιείται ως courtine, ώστε να επιτρέπεται η ασύγχρονη εκτέλεση των απαιτούμενων ελέγχων και της ανάκτησης δεδομένων χωρίς να διακόπτεται η ομαλή λειτουργία της εφαρμογής.

Αρχικά, η εκτέλεση της μεθόδου καθυστερεί εώς ότου το AR Session βρεθεί σε κατάσταση παρακολούθησης, διασφαλίζοντας ότι το περιβάλλον AR είναι έτοιμο να δεχθεί Cloud Anchors (γραμμή 383-384). Παράλληλα, πραγματοποιείται αναμονή μέχρι την πλήρη αρχικοποίηση της σύνδεσης με τη βάση δεδομένων Firebase, ώστε να είναι δυνατή η αξιόπιστη ανάκτηση των αποθηκευμένων δεδομένων (γραμμή 386-387).

Στη συνέχεια, η συνάρτηση εκτελεί ερωτήματα στη βάση δεδομένων Firebase για την ανάκτηση όλων των αιθουσών που ανήκουν στο επιλεγμένο κτίριο, καθώς και των Cloud Anchors που είναι αποθηκευμένοι σε κάθε αίθουσα. Το ερώτημα αυτό εκτελείτε στην γραμμή 389. Για κάθε εγγραφή anchor που εντοπίζεται, εξάγεται το μοναδικό αναγνωριστικό του Cloud Anchor, καθώς και οι σχετικές πληροφορίες τοπικού μετασχηματισμού.

Τα αναγνωριστικά των Cloud Anchors δεν επιλύονται άμεσα, αλλά προστίθενται σε ουρά επεξεργασίας. Αυτό γίνεται στις γραμμές 426-430. Η προσέγγιση αυτή επιτρέπει τον ελεγχόμενο και σταδιακό χειρισμό των αιτημάτων επίλυσης, αποφεύγοντας την ταυτόχρονη εκτέλεση πολλαπλών επιλύσεων διαδικασιών που θα μπορούσαν να επηρεάσουν αρνητικά την απόδοση ή να οδηγήσουν σε περιορισμούς από το AR σύστημα.

```
380     private IEnumerator LoadAnchorsForBuildingThenQueueResolve(string buildingId)
381     {
382         while (ARSession.state != ARSessionState.SessionTracking)
383             yield return null;
384 
385         while (!_firebaseReady || _dbRoot == null)
386             yield return null;
387 
388         var task = _dbRoot.Child("buildings").Child(buildingId).Child("rooms").GetValueAsync();
389         while (!task.IsCompleted) yield return null;
390 
391         if (task.IsFaulted)
392         {
393             Debug.LogError("[CloudAnchors] DB read failed (building rooms): " + task.Exception);
394             yield break;
395         }
396 
397         var roomsSnap = task.Result;
398         if (!roomsSnap.Exists || !roomsSnap.HasChildren)
399         {
400             Debug.LogWarning($"[CloudAnchors] No rooms found for building '{buildingId}'.");
401             yield break;
402         }
403 
404         int total = 0;
```

Σχήμα 4.41: Μέθοδος LoadAnchorsForBuildingTheQueueResolve 1.

```
405 
406         foreach (var roomChild in roomsSnap.Children)
407         {
408             string roomId = roomChild.Key;
409             var anchorsNode = roomChild.Child("anchors");
410 
411             if (!anchorsNode.Exists || !anchorsNode.HasChildren)
412                 continue;
413 
414             foreach (var anchorChild in anchorsNode.Children)
415             {
416                 string json = anchorChild.GetRawJsonValue();
417                 if (string.IsNullOrEmpty(json)) continue;
418 
419                 AnchorDbRecord rec = JsonUtility.FromJson<AnchorDbRecord>(json);
420                 if (rec == null || string.IsNullOrEmpty(rec.cloudAnchorId)) continue;
421 
422                 if (rec.cloudAnchorId.StartsWith("CLOUD_ANCHOR_ID_EXAMPLE"))
423                     continue;
424 
425                 if (_dbAnchorIds.Add(rec.cloudAnchorId))
426                 {
427                     total++;
428                     EnqueueResolve(roomId, rec.cloudAnchorId, rec.localPos, rec.localEuler);
429                 }
430             }
431         }
432 
433         Debug.Log($"[CloudAnchors] Loaded {total} anchors from building '{buildingId}' (queued).");
434         UpdateDebugStatus();
435         EnsureResolveQueueProcessing();
436     }
```

Σχήμα 4.40: Μέθοδος LoadAnchorsForBuildingTheQueueResolve 2.

Στην μέθοδο EnqueueResolve(), στο Σχήμα 4.42, υλοποιείται ο μηχανισμός ουράς επίλυσης των Cloud Anchors. Για κάθε άγκυρα που ανακτάται από τη βάση δεδομένων, αποθηκεύονται στην ουρά τα βασικά δεδομένα της, συγκεκριμένα το roomId, οπού αντιστοιχεί στην αίθουσα, το cloudId όπου αποτελεί το μοναδικό αναγνωριστικό του Cloud Anchor, καθώς και οι τοπικές πληροφορίες μετασχηματισμού.

Οι πληροφορίες μετασχηματισμού περιλαμβάνουν το localPos, το οποίο αντιστοιχεί στη θέση της άγκυρας στο τοπικό σύστημα αναφοράς και το localEuler, το οποίο περιγράφει τον προσανατολισμό του αντικειμένου, όπως αυτός είχε οριστεί πριν την διαδικασία φιλοξενίας.

Η χρήση ουράς επιτρέπει τη σταδιακή και ελεγχόμενη επίλυση των Cloud Anchors, ανεξάρτητα από τη σειρά αποθήκευσης τους στη βάση δεδομένων, συμβάλλοντας στη σταθερότητα και την αποδοτική διαχείριση των αιτημάτων επίλυσης. Η ουρά μπορεί να φανεί στην γραμμή 444.

```
438     private void EnqueueResolve(string roomId, string cloudId, Vector3 localPos, Vector3 localEuler)
439     {
440         if (string.IsNullOrEmpty(cloudId)) return;
441         if (_spawnedResolved.ContainsKey(cloudId)) return;
442         if (_activeResolves.ContainsKey(cloudId)) return;
443 
444         _resolveQueue.Enqueue((roomId, cloudId, localPos, localEuler));
445     }
```

Σχήμα 4.42: Μέθοδος EnqueueResolve.

4.3.8 Διαχείριση αγκυρών

Η διαχείριση των αγκυρών υλοποιείται με το AnchorManagerUI script, όπου ο διαχειριστής μπορεί να δει το πλήθος των αγκυρών σε κάθε κτίριο, να διαγράψει ξεχωριστά μία άγκυρα ή ακόμα και όλες απο ένα κτίριο απο την βάση δεδομένων.

Η λειτουργία του συστήματος βασίζεται στη δομή της Firebase Realtime Database: buildings/{buildingId}/rooms/{roomId}/anchors/{anchorId}

Έτσι, ο Anchor Manager ανακτά τα rooms του επιλεγμένου κτιρίου, εντοπίζει τους κόμβους anchors και δημιουργεί δυναμικά UI γραμμές σε scrollable λίστα, μια για κάθε anchor, προσφέροντας επίσης τις ενέργειες refresh για επαναφόρτωση της λίστας απο την βάση, delete single anchor διαγράφοντας μία συγκεκριμένη άγκυρα και delete all anchors για την διαγραφή όλων των αγκυρών σε όλα τα rooms του επιλεγμένου κτιρίου.

Τέλος, η επιλογή κτηρίου μπορεί να γίνεται είτε στατικά με το buildingId, είτε δυναμικά μέσω dropdown, με αντιστοίχιση label με Firebase key, έτσι ώστε ο διαχειριστής να μπορεί να αλλάζει κτήριο και να βλέπει και να διαχειρίζεται διαφορετικά δεδομένα.

Στην αρχή το script, κάνει έλεγχο για το αν η εφαρμογή βρίσκεται σε Admin mode, έτσι ώστε ο απλός χρήστης να μην μπορεί να έχει πρόσβαση, όπως φαίνεται στο Σχήμα 4.43.

```
81      private void Start()
82      {
83          if (!CloudAnchorPlacer.IsAdminScene)
84          {
85              Debug.LogWarning("[AnchorManagerUI] Not admin scene. Disabling UI.");
86              gameObject.SetActive(false);
87              return;
88          }
```

Σχήμα 4.43: Έλεγχος Admin mode.

Η επιλογή κτιρίου γίνεται μέσω του dropdown. Με κάθε αλλαγή επιλογής ενημερώνεται το buildingId και γίνεται refresh της λίστας, όπως φαίνεται στα Σχήματα 4.44 και 4.45.

```
96          if (buildingDropdown != null)
97          {
98              buildingDropdown.onValueChanged.RemoveAllListeners();
99              buildingDropdown.onValueChanged.AddListener(OnBuildingDropdownChanged);
100
101             ApplyBuildingFromDropdown(buildingDropdown.value);
102         }
```

Σχήμα 4.45: Αρχικοποίηση dropdown μενού επιλογής κτιρίου.

```
127     private void OnBuildingDropdownChanged(int dropdownIndex)
128     {
129         ApplyBuildingFromDropdown(dropdownIndex);
130         RefreshList();
131     }
```

Σχήμα 4.44: Μέθοδος OnBuildingDropdownChanged.

Στο Σχήμα 4.46, γίνεται αρχικοποίηση της σύνδεσης με την Firebase Realtime Database και στην συνέχεια γίνεται αυτόματα ανάκτηση των αγκυρών για εμφάνιση στη λίστα, μέσω της μεθόδου

```
107 private void InitFirebase()
108 {
109     FirebaseApp.CheckAndFixDependenciesAsync().ContinueWithOnMainThread(task =>
110     {
111         if (task.Result != DependencyStatus.Available)
112         {
113             Debug.LogError("[AnchorManagerUI] Firebase dependencies not available: " + task.Result);
114             _firebaseReady = false;
115             return;
116         }
117
118         var db = FirebaseDatabase.GetInstance(firebaseDbUrl);
119         _dbRoot = db.RootReference;
120         _firebaseReady = true;
121
122         Debug.Log("[AnchorManagerUI] Firebase ready.");
123         RefreshList();
124     });
125 }
```

Σχήμα 4.46: Σύνδεση με την Firebase Realtime Database.

InitFirebase().

Στον κώδικα στο Σχήμα 4.47, χρησιμοποιείται mapping ανάμεσα στο κείμενο που βλέπει ο χρήστης και στο πραγματικό key του Firebase.

```
146         foreach (var opt in buildingOptions)
147         {
148             if (opt != null && opt.label == selectedLabel)
149             {
150                 if (!string.IsNullOrEmpty(opt.buildingId))
151                 {
152                     buildingId = opt.buildingId;
153                     Debug.Log("[AnchorManagerUI] Building set from dropdown: " + buildingId);
154                 }
155                 return;
156             }
157         }
```

Σχήμα 4.47: Υλοποίηση mapping.

Η εφαρμογή διαβάζει όλα τα rooms ενός κτιρίου από τη βάση με την εντολή var task = _dbRoot.Child("buildings").Child(buildingId).Child("rooms").GetValueAsync(), ώστε να βρει όλες τις άγκυρες ανά αίθουσα.

Στο Σχήμα 4.48, για κάθε άγκυρα που εντοπίζεται στην βάση, δημιουργείται δυναμικά μία UI εγγραφή στη λίστα, η οποία περιλαμβάνει την αίθουσα και cloudAnchorId, καθώς και κουμπί διαγραφής.

```
218             var anchorsNode = roomChild.Child("anchors");
219             if (!anchorsNode.Exists || !anchorsNode.HasChildren)
220                 continue;
221
222             foreach (var anchorChild in anchorsNode.Children)
223             {
224                 string nodeKey = anchorChild.Key;
225                 string cloudIdToShow = nodeKey;
226
227                 string rawJson = anchorChild.GetRawJsonValue();
228                 if (!string.IsNullOrEmpty(rawJson))
229                 {
230                     try
231                     {
232                         var rec = JsonUtility.FromJson<AnchorDbRecord>(rawJson);
233                         if (rec != null && !string.IsNullOrEmpty(rec.cloudAnchorId))
234                             cloudIdToShow = rec.cloudAnchorId;
235                     }
236                     catch { }
237                 }
238
239                 var row = Instantiate(rowPrefab, contentParent, false);
240                 row.transform.localScale = Vector3.one;
241
242                 row.Setup(
243                     roomLabel: $"Room {roomNumber}",
244                     anchorId: cloudIdToShow,
245                     onDelete: () => DeleteSingleAnchor(buildingId, roomId, nodeKey, cloudIdToShow)
246                 );
```

Σχήμα 4.48: Δημιουργία UI δυναμικά στη λίστα.

Ο διαχειριστής μπορεί να διαγράψει μεμονωμένη άγκυρα, αφαιρώντας τον κόμβο anchorsRef.Child(nodeKey).RemoveValueAsync().ContinueWithOnMainThread, από την βάση, και στην συνέχεια γίνεται ανανέωση της λίστας. Ενώ η μαζική διαγραφή αγκυρών ενός κτιρίου γίνεται με την εντολή “_ dbRoot. Child("buildings"). Child(buildingId). Child("rooms"). Child(roomId). Child("anchors"). RemoveValueAsync()”.

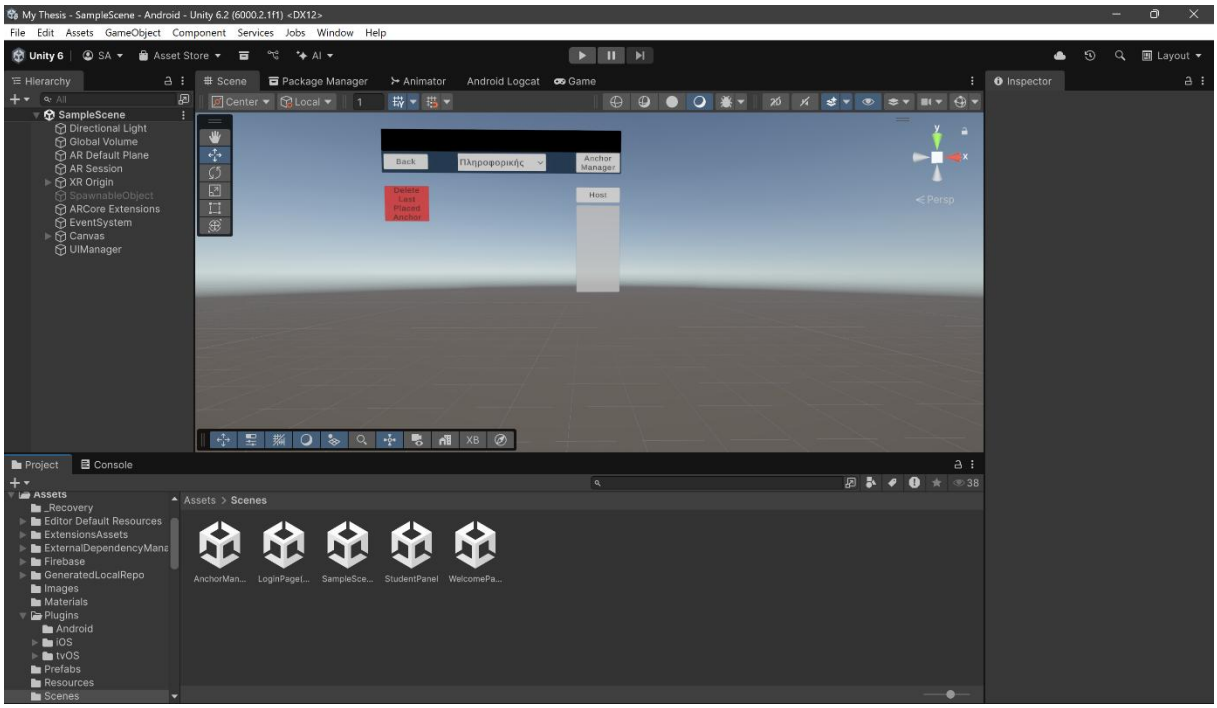
Ο Anchor Manager δεν δημιουργεί άγκυρες, απλώς διαχειρίζεται της ήδη hosted άγκυρες που έχουν αποθηκευτεί στη βάση δεδομένων από τη διαδικασία φιλοξενίας στο CloudAnchorPlacer.

4.4 Δημιουργία και Οργάνωση Σκηνών στο Unity

Για την υλοποίηση της εφαρμογής AR, χρησιμοποιήθηκε η μηχανή ανάπτυξης Unity, η οποία επιτρέπει τη δημιουργία και διαχείριση πολλαπλών σκηνών. Η χρήση διαφορετικών σκηνών κρίθηκε απαραίτητη ώστε να διαχωριστούν οι λειτουργικές ενότητες της εφαρμογής και να επιτευχθεί καλύτερη οργάνωση, συντηρητικότητα και επεκτασιμότητα του συστήματος.

Στο πλαίσιο της παρούσας εργασίας δημιουργήθηκαν διακριτές σκηνές, κάθε μία απο τις οποίες εξυπηρετεί συγκεκριμένο ρόλο στη ροή της εφαρμογής.

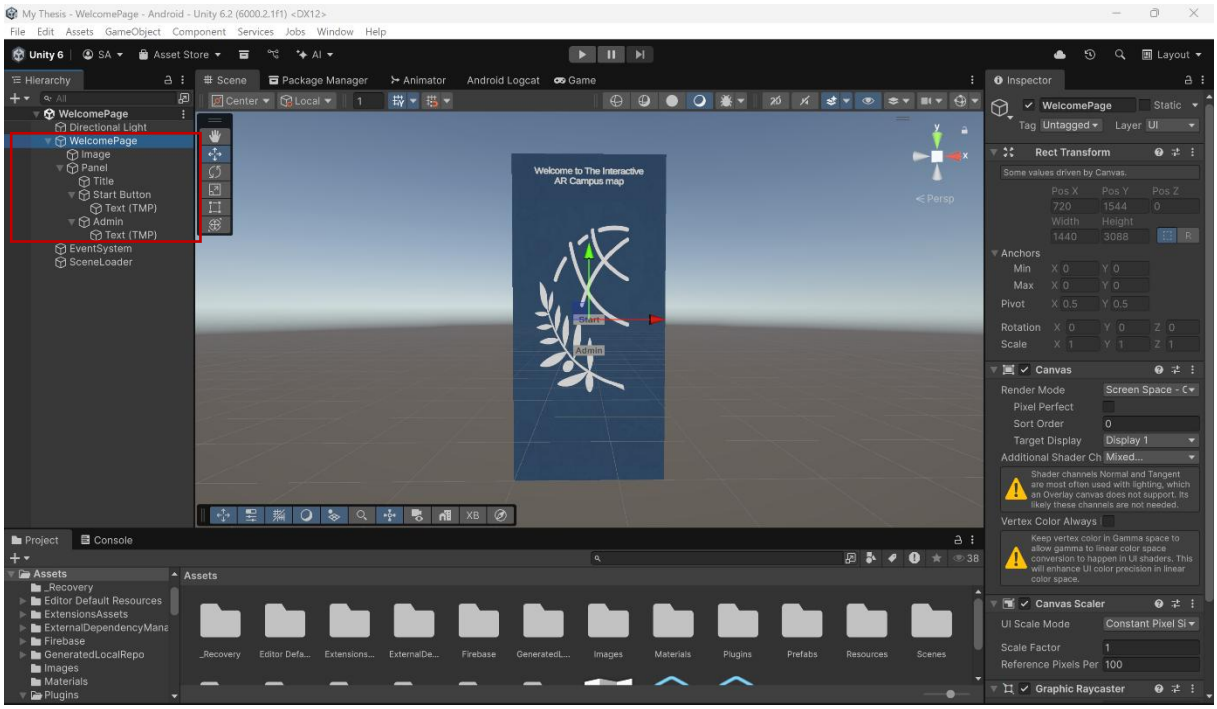
Στο σχήμα 4.49, φαίνονται οι πέντε σκηνές στα Assets.



Σχήμα 4.49: Σκηνές στο Unity

4.4.1 Σκηνή Εκκίνησης (Welcome Scene)

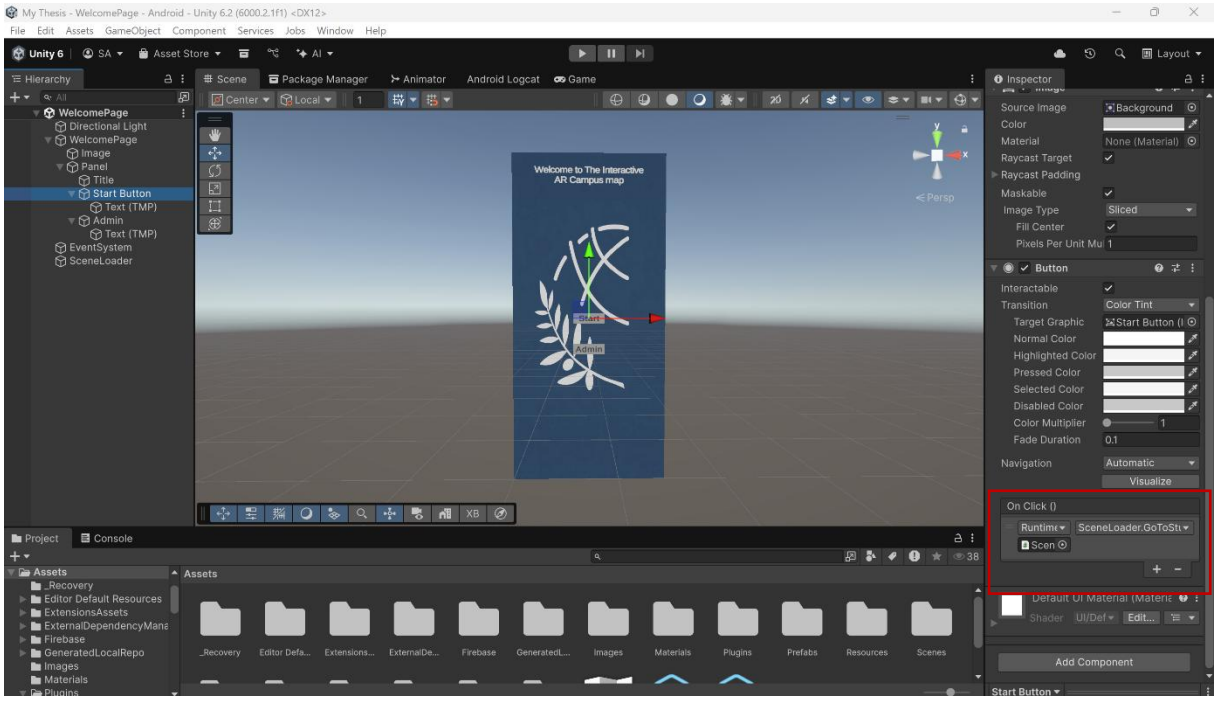
Η πρώτη σκηνή της εφαρμογής αποτελεί τη σκηνή εκκίνησης (Welcome Page). Στη σκηνή αυτή ο χρήστης έχει τη δυνατότητα να επιλέξει τον ρόλο του, δηλαδή αν θα εισέλθει στην εφαρμογή ως διαχειριστής (Admin) ή ως φοιτητής (Student). Η επιλογή αυτή καθορίζει τη λειτουργικότητα που θα είναι διαθέσιμη στις επόμενες σκηνές (Σχήμα 4.50).



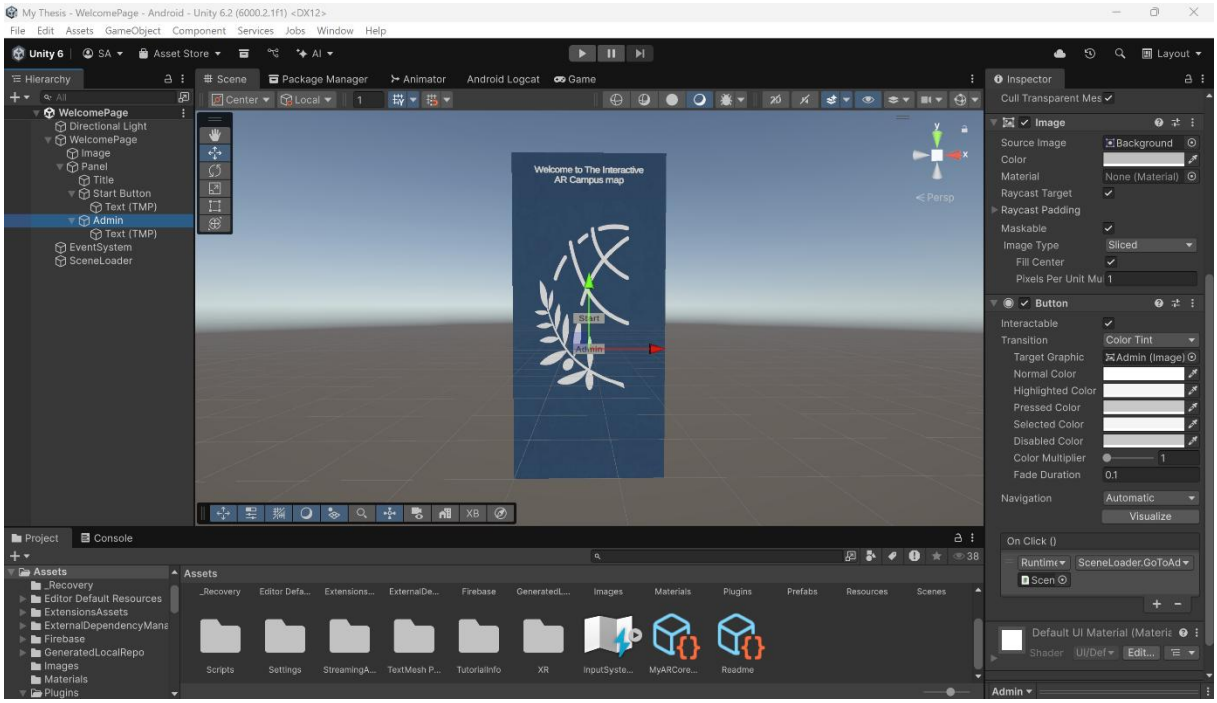
Σχήμα 4.50: Welcome Scene.

Η σκηνή περιλαμβάνει γραφικά στοιχεία διεπαφής (UI), όπως εικόνα για το φόντο, κείμενο για τον τίτλο και τα δύο κουμπιά, τα οποία υλοποιήθηκαν μέσω του συστήματος Canvas της Unity (Σχήμα 4.50).

Στα κουμπιά Start και Admin, στην μέθοδο OnClicked(), ανατίθεται το script UIManager (Σχήμα 4.51 και 4.52), το οποίο είναι ανατεθειμένο στο SceneLoader αντικείμενο.

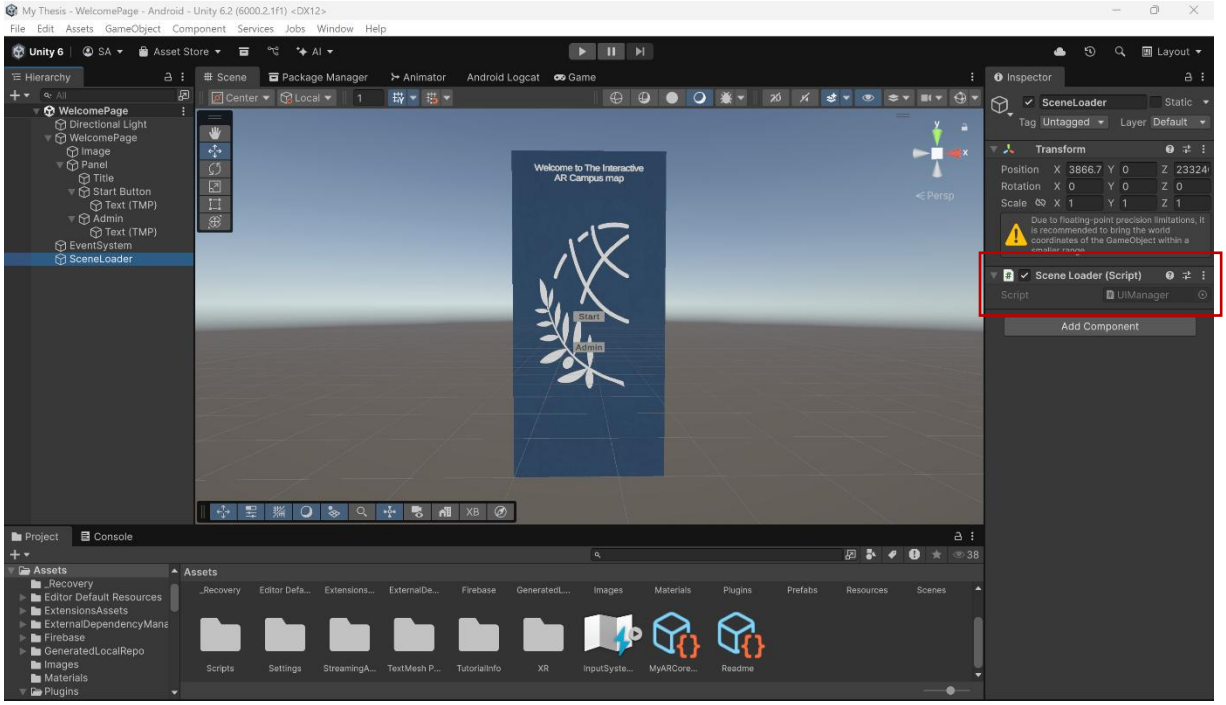


Σχήμα 4.51: Start Button OnClicked μέθοδος.



Σχήμα 4.52: Admin Button OnClicked μέθοδος.

Η μετάβαση στις επόμενες σκηνές πραγματοποιείται με την χρήση του SceneManager της Unity (Σχήμα 4.53).

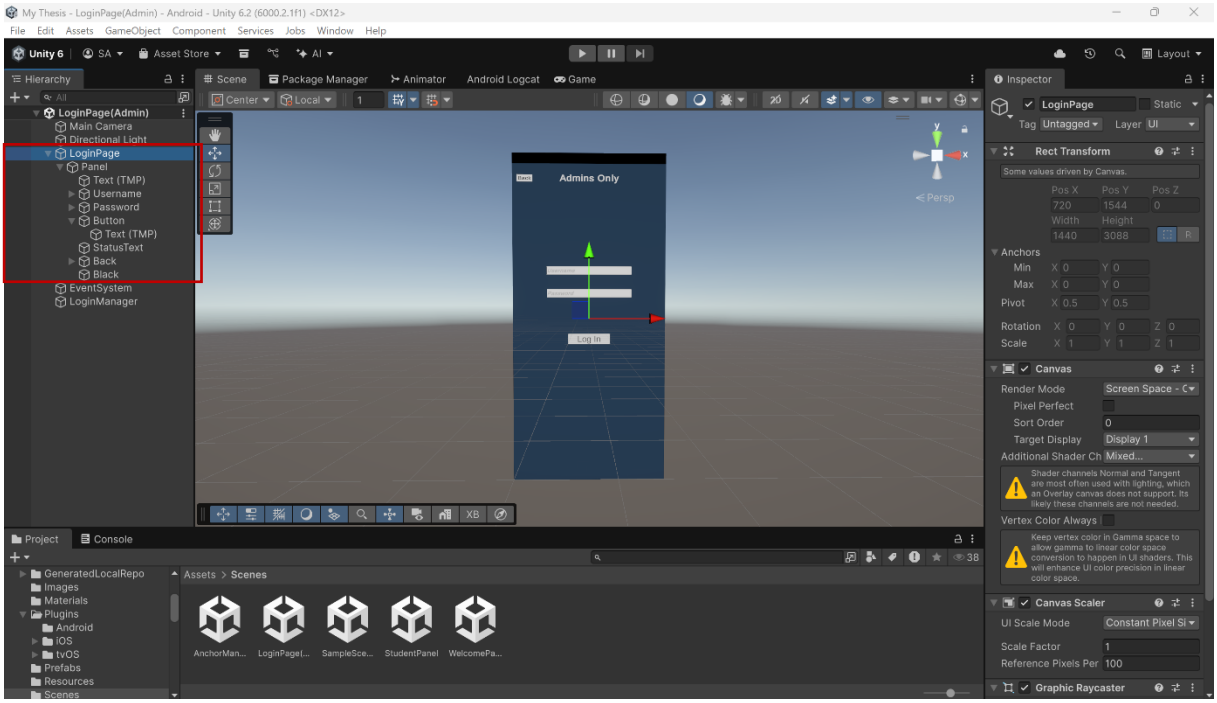


Σχήμα 4.53: SceneLoader αντικείμενο.

4.4.2 Σκηνή Αυθεντικοποίησης Διαχειριστή (Admin Login Scene)

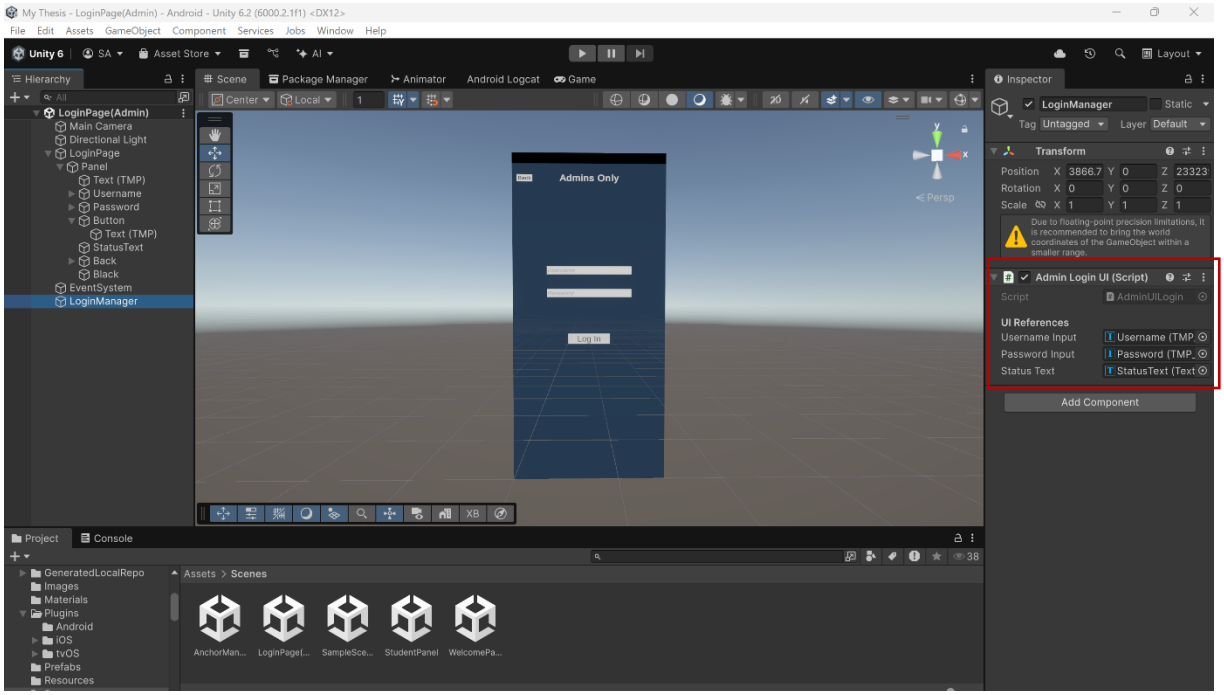
Για την πρόσβαση στις λειτουργίες διαχείρισης δημιουργήθηκε ξεχωριστή σκήνη αυθεντικοποίησης διαχειριστή. Στη σκηνή αυτή, ο χρήστης εισάγει όνομα χρήστη και κωδικό πρόσβασης, τα οποία ελέγχονται μέσω της Firebase Realtime Database.

Η σκηνή περιλαμβάνει πεδία εισαγωγής κειμένου (Input Fields), κουμπί “Log In” και μηνύματα κατάστασης, ώστε να ενημερώνεται ο χρήστης για την επιτυχία ή αποτυχία της διαδικασίας σύνδεσης (Σχήμα 4.54).



Σχήμα 4.54: Σκηνή αυθεντικοποίησης διαχειριστή.

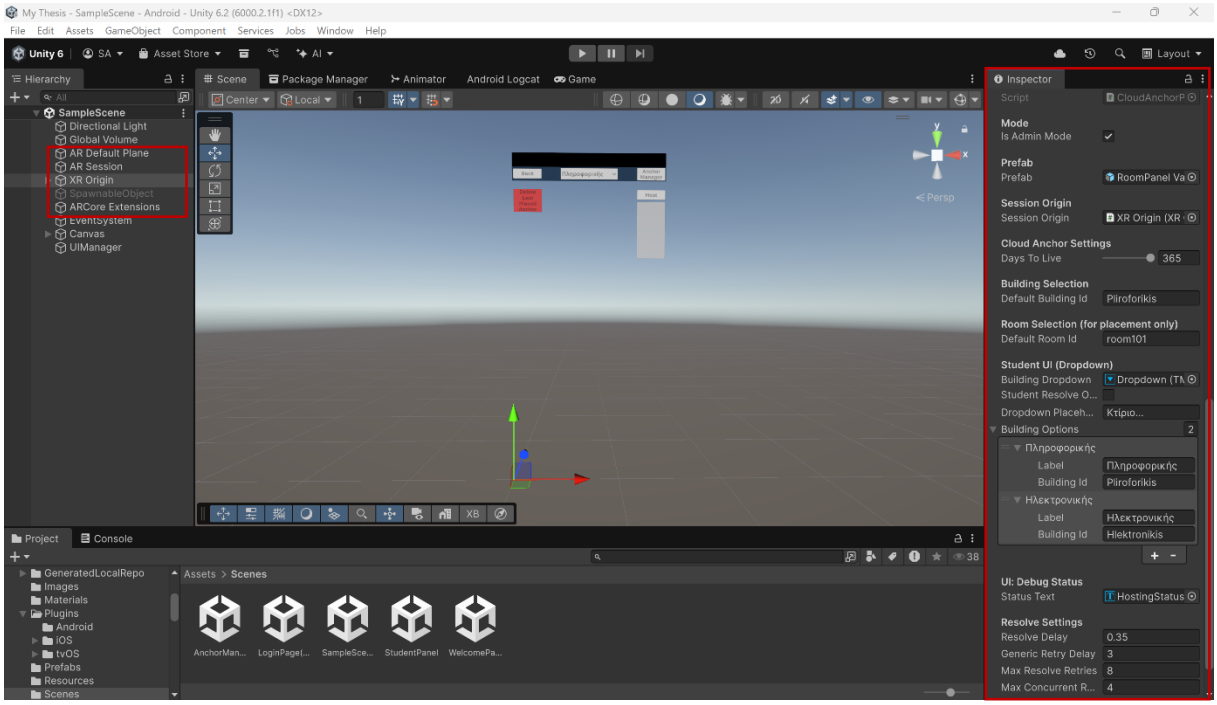
Σε περίπτωση επιτυχούς αυθεντικοποίησης, ο διαχειριστής μεταφέρεται στην κύρια σκηνή AR. Όλα αυτά γίνονται με την βοήθεια του script AdminUILogin το οποίο ανατέθηκε στο LoginManager αντικείμενο (Σχήμα 4.55).



Σχήμα 4.55: Αντικείμενο LginManager.

4.4.3 Κύρια Σκηνή Επαυξημένης Πραγματικότητας (AR Scene) και Σκηνή Φοιτητή (Student Scene)

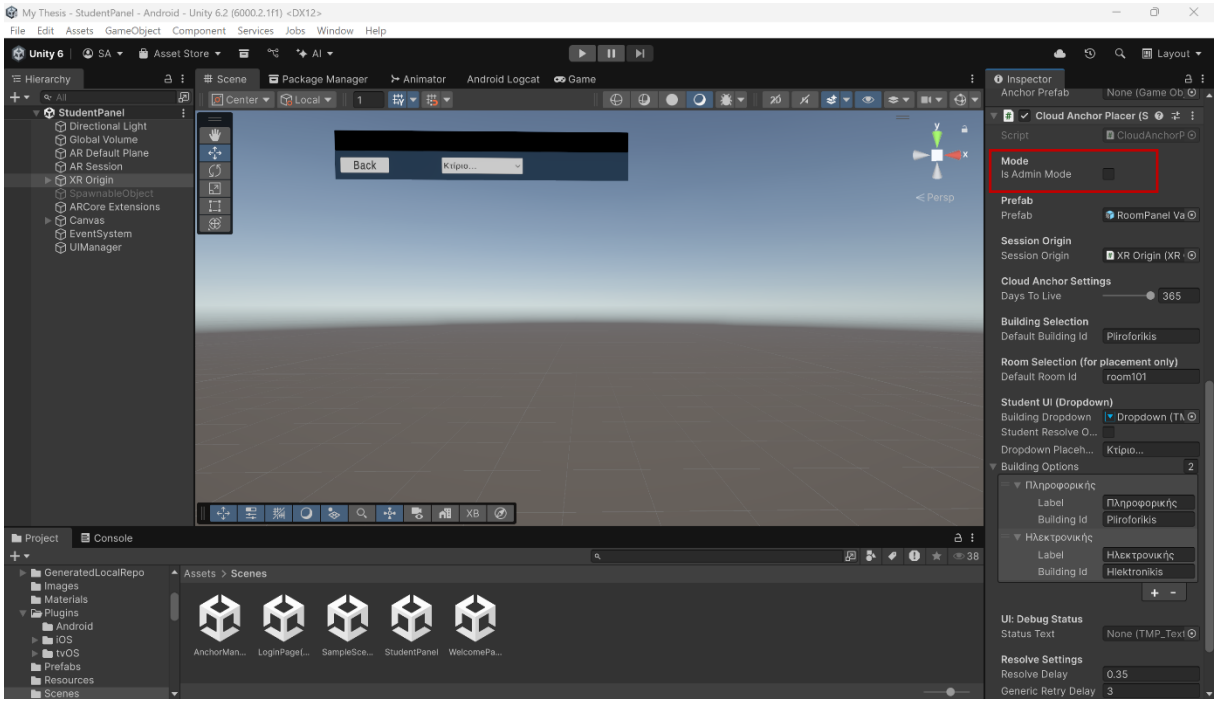
Η βασική λειτουργικότητα της εφαρμογής υλοποιήθηκε στην κύρια σκηνή AR (SampleScene). Η σκηνή αυτή περιλαμβάνει όλα τα απαραίτητα στοιχεία για τη λειτουργία του AR, όπως το AR Session, το XR Origin όπου ανατέθηκε το CloudAnchorPlacer script, καθώς και τα απαραίτητα managers του AR Foundation για ανίχνευση επιφανειών, αγκυρώσεις και αλληλεπίδραση με το περιβάλλον (Σχήμα 4.56).



Σχήμα 4.56: Sample Scene.

Ανάλογα με τον ρόλο του χρήστη, η σκηνή αυτή λειτουργεί με διαφορετικό τρόπο. Στην περίπτωση του διαχειριστή, παρέχεται η δυνατότητα τοποθέτησης, περιστροφής και φιλοξενίας αγκυρών AR, καθώς και διαχείρισης αυτών μέσω ειδικού περιβάλλοντος. Στην περίπτωση του φοιτητή, η σκηνή λειτουργεί αποκλειστικά σε λειτουργία προβολής, όπου οι αποθηκευμένες αγκυρώσεις επιλύονται και προβάλλονται αυτόματα στον χώρο. Επίσης, η σκηνή του φοιτητή λειτουργεί ακριβώς με το ίδιο script, απλα δεν γίνεται επιλογή του “Is Admin Mode”, όπως φαίνεται στο Σχήμα 4.57.

Η διεπαφή χρήστη της σκηνής περιλαμβάνει επιλογή κτιρίου μέσω dropdown μενού, προβολή πληροφοριών αιθουσών και μαθημάτων.



Σχήμα 4.57: “Is Admin Mode” disabled.

4.4.4 Διαχείριση Μετάβασης Μεταξύ Σκηνών

Η μετάβαση μεταξύ των σκηνών υλοποιήθηκε με χρήση του μηχανισμού Scene Management της Unity (Σχήμα 4.58). Με τον τρόπο αυτό διασφαλίζεται ομαλή ροή της εφαρμογής, καθώς και σωστή ενεργοποίηση ή απενεργοποίηση των αντίστοιχων λειτουργιών ανά σκηνή και ρόλο χρήστη.

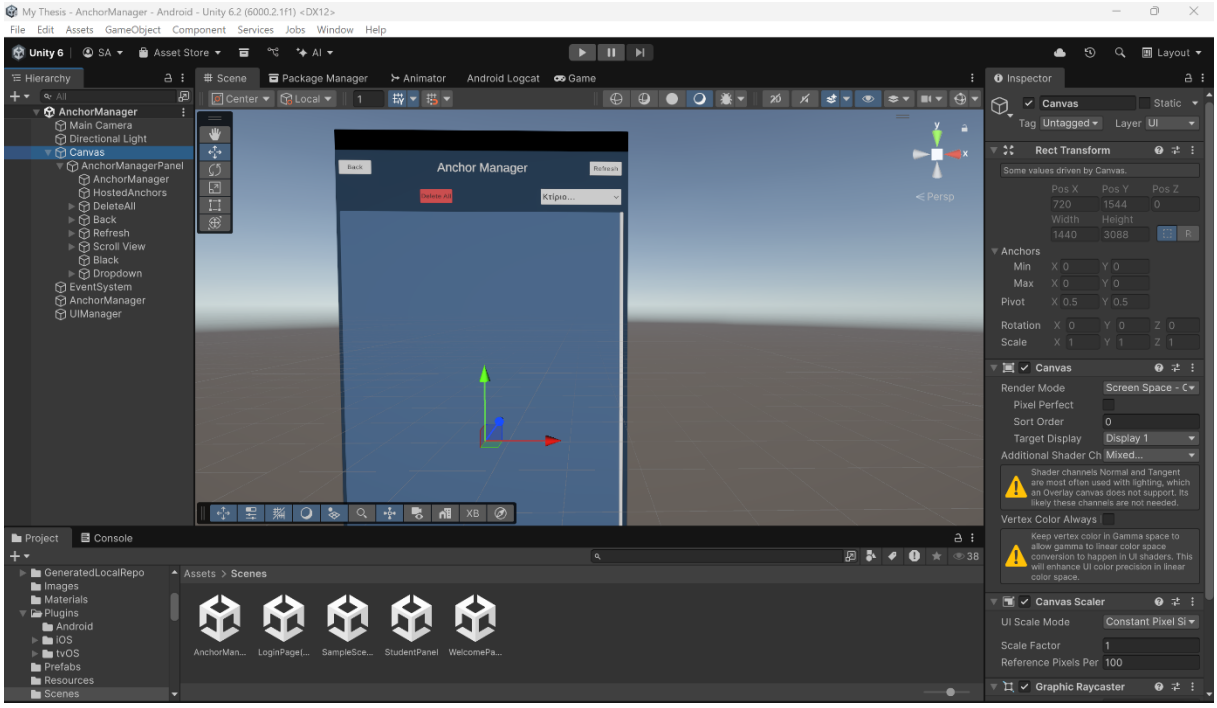
Ο διαχωρισμός της εφαρμογής σε πολλαπλές σκηνές, συνέβαλε στη σαφή δομή του έργου και επέτρεψε την ανεξάρτητη ανάπτυξη και δοκιμή κάθε λειτουργικής ενότητας.

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class SceneLoader : MonoBehaviour
{
    // Loads admin login scene
    public void GoToAdminLogin()
    {
        SceneManager.LoadScene("LoginPage(Admin)");
    }

    // Loads welcome page (optional for later)
    public void GoToWelcomePage()
    {
        SceneManager.LoadScene("WelcomePage");
    }

    public void GoToStudentPage()
    {
        SceneManager.LoadScene("StudentPanel");
    }

    public void GoToAnchorManager()
    {
        SceneManager.LoadScene("AnchorManager");
    }

    public void GoToCloudAnchorPlacer()
    {
        SceneManager.LoadScene("SampleScene");
    }
}

```

Σχήμα 4.58: UIManager script.

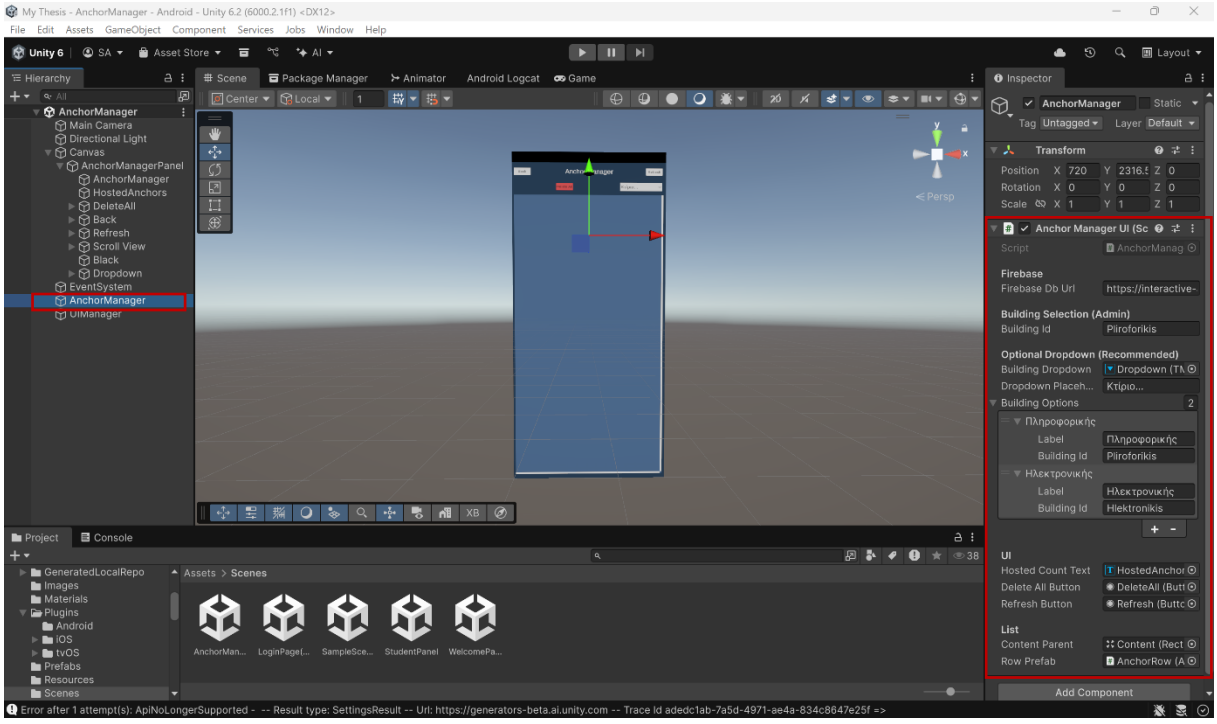
4.4.5 Σκηνή Διαχείρισης Αγκυρών

Στην σκηνή διαχειριστή υπάρχουν αντικείμενα κειμένου, κουμπιά, dropdown μενού και Scroll View όπως φαίνεται στο Σχήμα 4.59.



Σχήμα 4.59: Καμβάς της AnchorManager σκηνής.

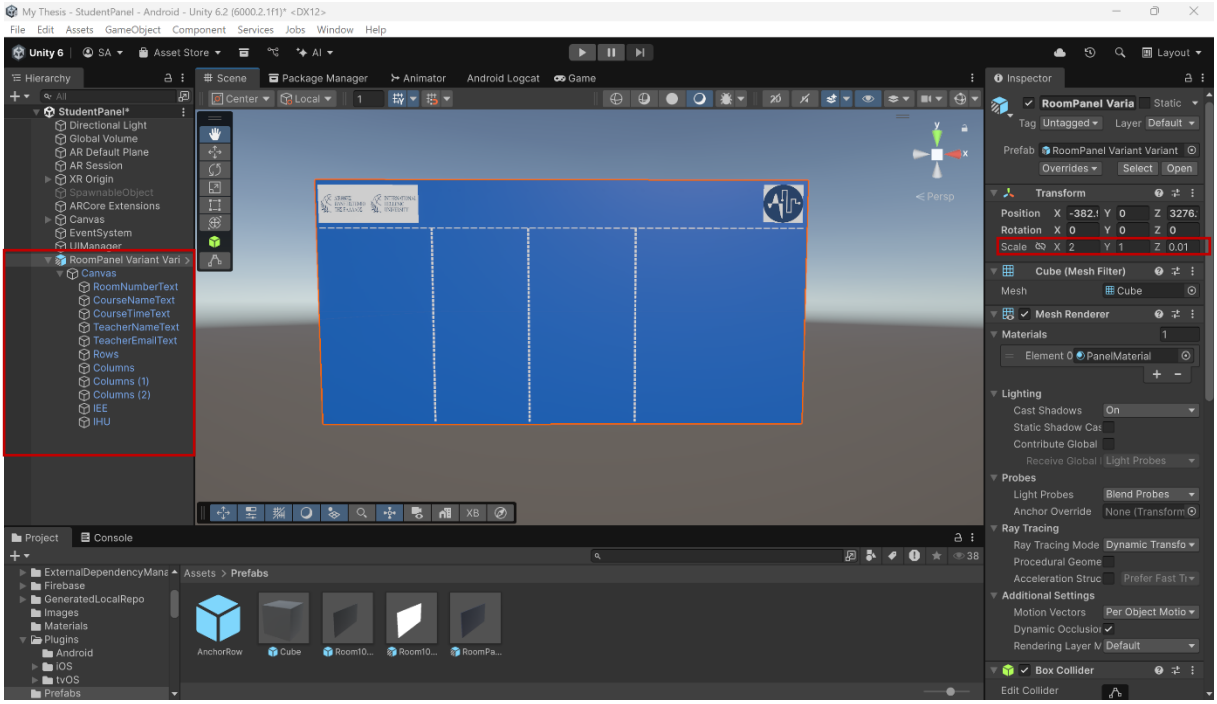
Η σκηνή AnchorManager λειτουργεί με το AnchorManagerUI script το οποίο ανατέθηκε στο AnchorManager αντικείμενο που φαίνεται στο Σχήμα 4.60.



Σχήμα 4.60: Ανάθεση AnchorManagerUI script στο AnchorManager αντικείμενο.

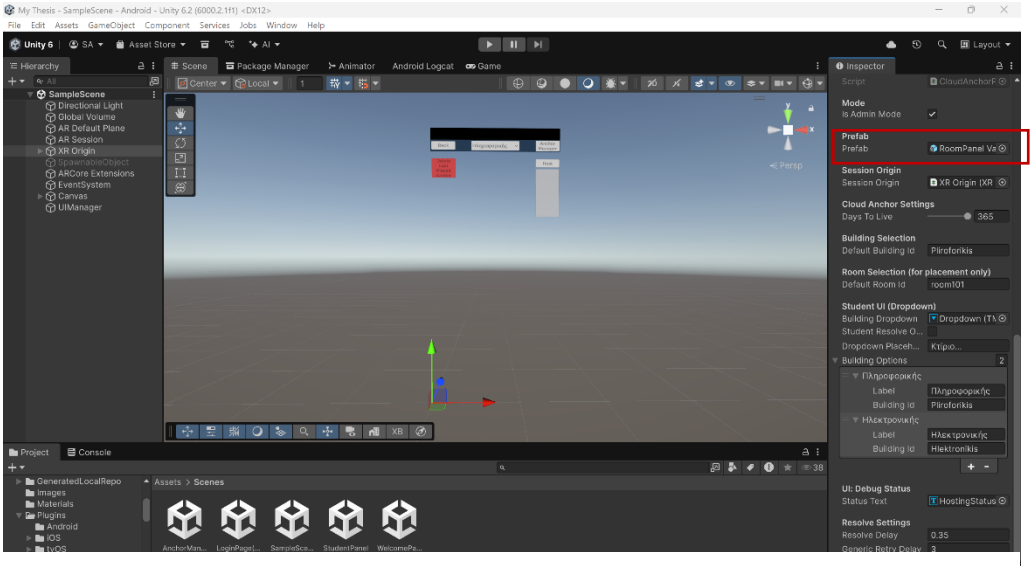
4.4.6 Δημιουργία Τρισδιάστατου Μοντέλου

Για την δημιουργία του τρισδιάστατου αντικειμένου δημιουργίθηκε ένας 3D κύβος, αλλάχτηκε το scale του σε (X: 2, Y: 1, Z: 0.01), έτσι ώστε να μετατραπεί σε ένα πίνακα πληροφοριών. Το μοντέλο περιέχει αντικείμενα Text για τον αριθμό της αίθουσας, όνομα μαθήματος, όνομα καθηγητή, ημέρα και ώρα και το email του καθηγητή, τα οποία εμφανίζονται δυναμικά από την βάση. Δημιουργήθηκαν στήλες και γραμμές με text αντικείμενα για τη διαχώριση της πληροφορίας (Σχήμα 4.61). Προστέθηκαν επίσης δύο εικόνες, μία για το Πανεπιστήμιο και μία για το τμήμα.



Σχήμα 4.61: Δημιουργία τρισδιάστατου μοντέλου.

Στο τέλος, το μοντέλο αποθηκεύτηκε ως prefab και ανατέθηκε στο CloudAnchorPlacer script (Σχήμα 4.62).



Σχήμα 4.62: Ανάθεση prefab στο script.

Κεφάλαιο 5ο: Συμπεράσματα και μελλοντικές επεκτάσεις

Στην παρούσα διπλωματική εργασία υλοποιήθηκε μια διαδραστική εφαρμογή επαυξημένης πραγματικότητας (Augmented Reality – AR) με σκοπό την παροχή χωρικής και εκπαιδευτικής πληροφόρησης στην Αλεξάνδρεια Πανεπιστημιούπολη του Διεθνούς Πανεπιστημίου της Ελλάδος. Η εφαρμογή αξιοποιεί σύγχρονες τεχνολογίες AR και cloud υποδομές, προκειμένου να συνδέσει ψηφιακό περιεχόμενο με συγκεκριμένα σημεία του φυσικού χώρου.

Η ανάπτυξη της εφαρμογής βασίστηκε στο Unity σε συνδιασμό με το AR Foundation και το Google ARCore, επιτρέποντας την ανίχνευση επιφανειών, τη χωρική κατανόηση του περιβάλλοντος και την τοποθέτηση σταθερών ψηφιακών αντικειμένων μέσω AR Anchors. Η χρήση των Cloud Anchors επέτρεψε την επίμονη αποθήκευση των αγκυρών στο cloud, διασφαλίζοντας ότι το ίδιο ψηφιακό περιεχόμενο μπορεί να επανα-εντοπιστεί και να προβληθεί από διαφορετικές συσκευές και χρήστες στον ίδιο φυσικό χώρο.

Παράλληλα η Firebase Realtime Database αξιοποιήθηκε ως κεντρική υποδομή αποθήκευσης και συγχρονισμού δεδομένων. Μέσω της ιεραρχικής δομής της βάσης δεδομένων (κτίριο – αίθουσα – περιεχόμενο – άγκυρες), επιτεύχθηκε οργανωμένη και επεκτάσιμη διαχείριση της πληροφορίας, καθώς και δυναμική ανάκτηση των δεδομένων κατά την εκτέλεση της εφαρμογής. Η βάση δεδομένων λειτουργεί ως συνδετικός κρίκος μεταξύ του φυσικού χώρου και του ψηφιακού περιεχομένου που προβάλλεται στην εφαρμογή.

Η εφαρμογή σχεδιάστηκε με διαχωρισμό ρόλων, παρέχοντας διαφορετικές λειτουργίες για διαχειριστές και τελικούς χρήστες. Ο διαχειριστής διαθέτει τη δυνατότητα αυθεντικοποίησης, επιλογής κτιρίου και αίθουσας, τοποθέτησης και περιστροφής αγκυρών, φιλοξενίας τους ως Cloud Anchors και διαχείρισης τους μέσω ειδικού περιβάλλοντος (Anchor Manager). Αντίστοιχα, ο φοιτητής μπορεί να επιλέξει κτίριο και να δει τις αποθηκευμένες αγκυρώσεις να επιλύονται αυτόματα, προβάλλοντας πληροφορίες για τις αίθουσες και τα μαθήματα στον πραγματικό χώρο.

Η λειτουργικότητα της εφαρμογής δοκιμάστηκε σε πραγματικές συνθήκες χρήσης και τα αποτελέσματα έδειξαν ότι οι Cloud Anchors παρέχουν ικανοποιητική σταθερότητα και ακρίβεια επανα-τοποθέτησης, εφόσον ο χώρος διαθέτει επαρκή οπτικά χαρακτηριστικά. Επιπλέον, η χρήση ουράς επίλυσης αγκυρών και μηχανισμών επανα-εκτέλεσης συνέβαλε στη σταθερότητα της εφαρμογής και στην αποφυγή προβλημάτων απο υπερφόρτωση ή αποτυχία επίλυσης.

Ωστόσο, σε ορισμένες περιπτώσεις παρατηρήθηκε ότι η ομοιομορφία του φυσικού περιβάλλοντος μπορεί να περιορίσει την αποτελεσματικότητα της αναγνώρισης χώρου. Συγκεκριμένα, όταν οι τοίχοι είναι λευκοί και δεν διαθέτουν επαρκή οπτικά χαρακτηριστικά, το σύστημα δυσκολεύεται να εξάγει διακριτά χαρακτηριστικά απο την ανίχνευση επιφανειών (planes), γεγονός που ενδέχεται να επηρεάσει τη διαδικασία επανατοποθέτησης των αγκυρών.

Για την αντιμετώπιση του ζητήματος αυτού, προτείνεται και δοκιμάστηκε η χρήση τεχνητών οπτικών δεικτών, όπως αφίσες ή QR codes, οι οποίοι εμπλουτίζουν το feature map με πρόσθετα γεωμετρικά χαρακτηριστικά και βελτιώνουν την αξιοπιστία της επανατοποθέτησης.

Συνολικά, η παρούσα εργασία καταδεικνύει ότι η AR μπορεί να αποτελέσει ένα αποτελεσματικό εργαλείο πληροφόρησης και χωρικής κατανόησης σε πανεπιστημιακά περιβάλλοντα. Η προτεινόμενη λύση είναι λειτουργική, επεκτάσιμη και μπορεί να προσαρμοστεί εύκολα σε άλλα κτίρια, τμήματα ή ακόμα και διαφορετικά εκπαιδευτικά ιδρύματα.

Παρόλο που η εφαρμογή καλύπτει επιτυχώς τους αρχικούς στόχους της, υπάρχουν αρκετές δυνατότητες για μελλοντική βελτίωση και επέκταση. Ενδεικτικά:

- Ενσωμάτωση συστήματος πλοήγησης (AR navigation) για καθοδήγηση χρηστών σε διάφορα μέρη στην Πανεπιστιμιούπολη.
- Υποστήριξη περισσότερων τύπων ψηφιακού περιεχομένου, όπως 3D μοντέλα, βίντεο ή διαδραστικά στοιχεία.
- Επέκταση του συστήματος αυθεντικοποίησης με χρήση Firebase Authentication και διαφορετικά επίπεδα δικαιωμάτων.
- Βελτιστοποίηση της εμπειρίας χρήστη μέσω καλύτερου UI/UX σχεδιασμού και προσαρμογής σε διαφορετικές συσκευές.
- Εφαρμογή μηχανισμών ανάλυσης χρήσης για αξιολόγηση της αποδοτικότητας της εφαρμογής σε πραγματικές συνθήκες.
- Δημιουργεία UI για αλλαγή δεδομένων των αιθουσών και αυτόματη ενημέρωση JSON αρχείου στην βάση.
- Σύνδεση υπηρεσιών τμημάτων π.χ. Ανακοινώσεις.

Η υλοποίηση των παραπάνω θα μπορούσε να ενισχύσει περαιτέρω τη χρηστικότητα και τη λειτουργική αξία της εφαρμογής, καθιστώντας την ένα ολοκληρωμένο εργαλείο ψηφιακής πλοήγησης και πληροφόρησης σε πανεπιστημιακούς χώρους.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Επιστημονικά άρθρα

[1] S. Jangra, G. Singh, A. Mantri, S. Angra and B. Sharma, "Interactivity Development Using Unity 3D Software and C # Programming," *2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, Delhi, India, 2023, pp. 1-6, doi: 10.1109/ICCCNT56998.2023.10308030.

[2] N. Sebiraj, K. Advaith and P. S. Priyadharshini, "Enhancing User Experience and Design Exploration using Augmented Reality (AR) and Virtual Reality (VR)," *2024 5th International Conference on Smart Electronics and Communication (ICOSEC)*, Trichy, India, 2024, pp. 1620-1625, doi: 10.1109/ICOSEC61587.2024.10722546.

[3] Y. Wang, Y. Wang and Z. Fan, "Current Status and Prospects of Mobile AR Applications," *2021 International Conference on Culture-oriented Science & Technology (ICCST)*, Beijing, China, 2021, pp. 34-37, doi: 10.1109/ICCST53801.2021.00018.

[4] Z. Oufqir, A. El Abderrahmani and K. Satori, "ARKit and ARCore in serve to augmented reality," *2020 International Conference on Intelligent Systems and Computer Vision (ISCV)*, Fez, Morocco, 2020, pp. 1-7, doi: 10.1109/ISCV49265.2020.9204243.

[5] R. Takaseki, R. Nagashima, H. Kashima and T. Okazaki, "Development of Anchoring Support System Using with AR Toolkit," *2015 7th International Conference on Emerging Trends in Engineering & Technology (ICETET)*, Kobe, Japan, 2015, pp. 123-127, doi: 10.1109/ICETET.2015.22.

[6] N. Rajagopal, J. Miller, K. K. Reghu Kumar, A. Luong and A. Rowe, "Demo Abstract: Welcome to My World: Demystifying Multi-User AR with the Cloud," *2018 17th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, Porto, Portugal, 2018, pp. 146-147, doi: 10.1109/IPSN.2018.00036.

[7] D. Tsoukalos, V. Drosos and D. K. Tsolis, "Attempting to reconstruct a 3D indoor space scene with a mobile device using ARCore," *2021 12th International Conference on Information, Intelligence, Systems & Applications (IISA)*, Chania Crete, Greece, 2021, pp. 1-6, doi: 10.1109/IISA52424.2021.9555529.

[8] J. Zhou, Z. Xu, H. Yan, B. Gao, O. Yang and Z. Zhao, "AR Creator: A Mobile Application of Logic Education Based on AR," *2020 International Conference on Virtual Reality and Visualization (ICVRV)*, Recife, Brazil, 2020, pp. 379-380, doi: 10.1109/ICVRV51359.2020.00109.

Διαδικτυακοί τόποι

[9] «Firebase» [Online]. Διαθέσιμο στο: https://firebase.google.com/-docs/data-base/unit-y/st-a-rt?_-gl=1*sicca9*_up*MQ..&gclid=Cj0KCQiAyP3KBhD9ARIsAAJLnnZey7dRQ2kdqAjvU46sDmD5nU5wHYfQDm5QMCZOfKoKwd9khPDUGm4aAr1JEALw_wcB&gclsrc=aw.ds&gbraid=0AAAADpUDOizd9KM31u2r1F6jQWGrpA2X. Προσπελάστηκε: 11/01/2026

[10] «Unity» [Online]. Διαθέσιμο στο: https://unity.com/. Προσπελάστηκε: 11/01/2026

[11] «AR Foundation» [Online]. Διαθέσιμο στο: https://dev-elo-per-s.google.-com-/ar/-deve-lop/u-nity-arf/getting-started-ar-foundation. Προσπελάστηκε: 11/01/2026

[12] «ARCore» [Online]. Διαθέσιμο στο: https://developers.google.com/ar. Προσπελάστηκε: 11/01/2026

[13] «Google Cloud» [Online]. Διαθέσιμο στο: https://cloud.google.com/. Προσπελάστηκε: 11/01/2026

[14] «ARCore Extensions» [Online]. Διαθέσιμο στο: https://developers.google.com/ar/develop/unity-arf/getting-started-extensions?ar_foundations_version=6#ar-foundations-version. Προσπελάστηκε: 11/01/2026

[15] «ARCore API on Google Cloud» [Online]. Διαθέσιμο στο: https://de-v-elo-pe-rs.go-ogle.c-om/a-r/d-evelop/authorization?platform=unity-arf. Προσπελάστηκε: 11/01/2026

[16] «Firebase home page» [Online]. Διαθέσιμο στο: https://fireba--se.goo-gl--e.c-om/?t-m_so-urc-e=-google&utm_medi-um=cpc-&utm_ca-mpaign=Cloud-SS-DR-Firebase-FY26-global-gsem-1713590&utm_content=text-ad&utm_term=KW_firebase&gclsrc=aw.ds&gad_source=1&gad_campaignid=23417478215&gbraid=0AAAADpUDOh-NMS8ylkMHZaao86yisKR8&gclid=Cj0KCQiA1JLLBhCDARIsAAVfy7i9r9d-_wJGtmb5zX9lqCHv6FMYFb-lvpbcv5E7jSBYZgzuyzh-q54aAhoUEALw_wcB. Προσπελάστηκε: 11/01/2026

[17] «Firebase connect with Unity» [Online]. Διαθέσιμο στο: https://fir-eb-ase.g-oog-le.c-om/d-oc-s/u-ni-ty/setup. Προσπελάστηκε: 11/01/2026