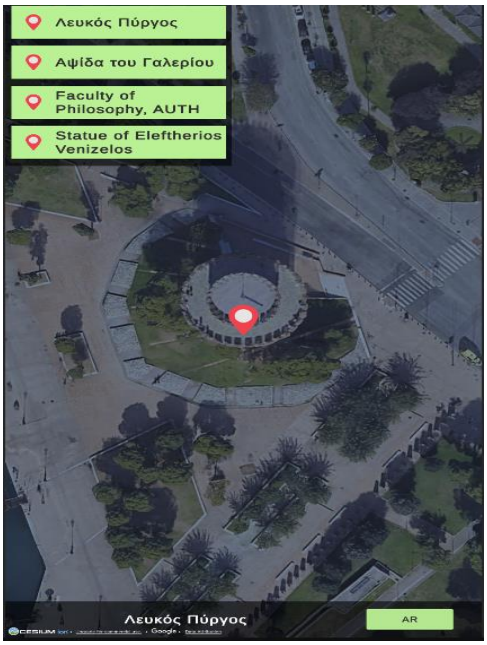
ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Εφαρμογή Επαυξημένης Πραγματικότητας (AR) με Χρήση GPS Τοποθεσίας



Του φοιτητή
ΑΚΡΙΒΟΥ ΝΙΚΟΛΑΟΥ
Αρ. Μητρώου: 174995

Επιβλέπων
Ευκλείδης Κεραμόπουλος
Καθηγητής

Ημερομηνία 30/8/2025

Εφαρμογή Επαυξημένης Πραγματικότητας (AR) με Χρήση GPS Τοποθεσίας
Κωδικός Δ.Ε. 24182
Ονοματεπώνυμο φοιτητή/των ΑΚΡΙΒΟΣ ΝΙΚΟΛΑΟΣ
Ονοματεπώνυμο εισηγητή ΧΡΙΣΤΙΝΑ ΒΟΛΙΩΤΗ
Ημερομηνία ανάληψης Δ.Ε. 30-03-2024
Ημερομηνία περάτωσης Δ.Ε. 30-08-2025

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.

Η έγκριση της διπλωματικής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

«Αφιερώνεται στην οικογένειά μου, για τη σταθερή τους στήριξη, την πίστη τους σε εμένα και την ηρεμία που μου πρόσφεραν σε κάθε στάδιο αυτής της διαδρομής.

Και σε εκείνους τους ανθρώπους που ήταν δίπλα μου με κατανόηση, λόγια ενθάρρυνσης και αληθινό ενδιαφέρον, συμβάλλοντας ουσιαστικά στην ολοκλήρωση αυτής της προσπάθειας.»

«Αφιέρωση»

Πρόλογος

Επέλεξα το συγκεκριμένο θέμα διπλωματικής εργασίας καθώς, ως φοιτητής του τμήματός μου, με εντυπωσίασε ιδιαίτερα η δυνατότητα δημιουργίας τρισδιάστατων αντικειμένων και η προσοχή στη λεπτομέρεια που απαιτείται για να αποκτήσουν ρεαλισμό και αισθητική πληρότητα.Μέσα από μαθήματα που παρακολούθησα κατά τη διάρκεια των σπουδών μου, αναπτύχθηκε σε μεγάλο βαθμό η αγάπη μου για τον κώδικα και τον προγραμματισμό, αλλά και η επιθυμία μου να συνδέσω την τεχνολογία με τον κόσμο του πολιτισμού και των ταξιδιών. Η ανάγκη να κατανοήσω βαθύτερα την ιστορία πίσω από τα μέρη που επισκέπτομαι, με οδήγησε στην επιλογή μιας εφαρμογής που ενώνει την τεχνολογία με την πολιτιστική εμπειρία.Στόχος μου ήταν να δημιουργήσω μια εφαρμογή που να συνδυάζει αυτά τα στοιχεία και να προσφέρει στον χρήστη μια καθηλωτική εμπειρία, μέσα από ρεαλιστικά, αληθοφανή ψηφιακά αντικείμενα, εμπλουτίζοντας με αυτόν τον τρόπο την επαφή του με τον τόπο και την ιστορία.

Περίληψη

Η παρούσα πτυχιακή εργασία επικεντρώνεται στην ανάπτυξη μιας τουριστικής εφαρμογής Επαυξημένης Πραγματικότητας (AugmentedReality - AR) για συσκευές Android, η οποία βασίζεται στην τοποθεσία του χρήστη μέσω GPS.Στόχος της εφαρμογής είναι η εμπλουτισμένη εμπειρία περιήγησης, επιτρέποντας στους χρήστες να λαμβάνουν πληροφορίες σχετικές με τα σημεία που επισκέπτονται. Η εφαρμογή ανιχνεύει τη γεωγραφική θέση του χρήστη και προβάλλει, μέσω AR, ένα εικονικό πλαίσιο που περιέχει κείμενο με ιστορικές και πολιτιστικές πληροφορίες για τον αντίστοιχο χώρο.Η εφαρμογή υλοποιήθηκε στο περιβάλλον Unity και αξιοποιεί το GPS για την αντιστοίχιση της φυσικής θέσης με τα σημεία ενδιαφέροντος. Σκοπός είναι να προσφέρει στον χρήστη μια νέα οπτική της τοποθεσίας που επισκέπτεται, ενισχύοντας την κατανόηση και το ενδιαφέρον του για το περιβάλλον μέσα από διακριτικά και ουσιαστικά ψηφιακά στοιχεία.

«Location-based Augmented Reality (AR) Using GPS»

NIKOLAOSAKRIVOS

Abstract

This undergraduate thesis focuses on the development of a tourism-oriented Augmented Reality (AR) application for Android devices, which is based on the user's location via GPS. The goal of the application is to enhance the user’s exploration experience by providing information related to the places they visit. The application detects the user’s geographic position and displays, through AR, a virtual frame containing text with historical and cultural information relevant to the specific location.The application was developed in the Unity environment and utilizes GPS to match the user's physical location with predefined points of interest. Its purpose is to offer users a new perspective on the places they visit, enriching their understanding and interest in the environment through subtle yet meaningful digital elements.

Ευχαριστίες

Θα ήθελα να εκφράσω την ειλικρινή μου ευγνωμοσύνη στην οικογένειά μου για τη διαρκή τους στήριξη, την κατανόηση και την ηθική ενίσχυση που μου προσέφεραν σε όλη τη διάρκεια των σπουδών μου. Η παρουσία τους, η πίστη τους στις δυνατότητές μου και η ηρεμία που μου μετέδιδαν, αποτέλεσαν για εμένα σταθερό σημείο αναφοράς. Θερμές ευχαριστίες και στους φίλους μου, που με συνόδευσαν σε αυτήν την πορεία με κατανόηση, ενθάρρυνση και διακριτική συμπαράσταση στις απαιτητικές στιγμές. Ευχαριστώ επίσης θερμά τον επιβλέποντα καθηγητή μου Ευκλείδη Κεραμόπουλο αλλά και την εξωτερική συνεργάτιδα Χριστίνα Βολιώτη για τον χρόνο που μου διέθεσεσαν, την καθοδήγηση που μου παρείχαν και κυρίως για την κατανόηση που μου δείξανε. Οι συμβολή τους ήταν ουσιαστική στην ολοκλήρωση αυτής της προσπάθειας. Τέλος, ευχαριστώ το Πανεπιστήμιο και το Τμήμα μου για τις γνώσεις, τις εμπειρίες και τα εφόδια που μου προσέφεραν, τα οποία αποτέλεσαν τη βάση για την υλοποίηση αυτής της πτυχιακής εργασίας.

Περιεχόμενα

Κατάλογος Εικόνων

Συντομογραφίες

Δ.Ε.	Διπλωματική Εργασία
ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
Π.Ε.	Πτυχιακή Εργασία
AR	AugmentedReality (ΕπαυξημένηΠραγματικότητα)
GPS	GlobalPositioningSystem (ΠαγκόσμιοΣύστημαΕντοπισμού)
UI	UserInterface (ΔιεπαφήΧρήστη)
UX	UserExperience (ΕμπειρίαΧρήστη)
API	ApplicationProgrammingInterface (ΔιεπαφήΠρογραμματιστικώνΕφαρμογών)
SDK	SoftwareDevelopmentKit (ΠακέτοΑνάπτυξηςΛογισμικού)
APK	AndroidPackageKit (ΜορφήεγκατάστασηςεφαρμογώνAndroid)
IDE	IntegratedDevelopmentEnvironment (ΟλοκληρωμένοΠεριβάλλονΑνάπτυξης)
Unity	Unity Game Engine (ΜηχανήΑνάπτυξηςΠαιχνιδιών Unity)
AR	AugmentedReality

ΕΙΣΑΓΩΓΗ

Η παρούσα πτυχιακή εργασία εντάσσεται στο πεδίο της Επαυξημένης Πραγματικότητας (AR) και της τουριστικής τεχνολογίας. Βασικός σκοπός είναι η ανάπτυξη μιας εφαρμογής που αξιοποιεί τα δεδομένα GPS για την παροχή πληροφοριών πολιτισμικού ενδιαφέροντος μέσω ψηφιακής απεικόνισης, ενισχύοντας έτσι την εμπειρία του χρήστη κατά την περιήγησή του.Οι στόχοι της εργασίας περιλαμβάνουν τη διερεύνηση των τεχνολογιών AR, τον σχεδιασμό και την υλοποίηση της εφαρμογής σε περιβάλλον Unity, και την ενσωμάτωση των απαραίτητων λειτουργιών για τον εντοπισμό τοποθεσίας και την εμφάνιση σχετικού περιεχομένου.Τα παραδοτέα της εργασίας είναι η εφαρμογή Android και η παρούσα γραπτή αναφορά που περιγράφει αναλυτικά κάθε βήμα της διαδικασίας.Η εργασία χωρίζεται σε έξι κεφάλαια: το Κεφάλαιο 1 παρουσιάζει το αντικείμενο, τους στόχους και τη μεθοδολογία. Το Κεφάλαιο 2 αναλύει τις απαιτήσεις του συστήματος. Στο Κεφάλαιο 3 περιγράφονται οι τεχνολογίες και τα εργαλεία που χρησιμοποιήθηκαν. Το Κεφάλαιο 4 αφορά τον κύκλο ζωής της εφαρμογής, ενώ στο Κεφάλαιο 5 αναλύεται η υλοποίηση του κώδικα. Το Κεφάλαιο 6 παρουσιάζει τα σημεία ενδιαφέροντος που χρησιμοποιούνται στην εφαρμογή. Τέλος, παρατίθενται τα συμπεράσματα της εργασίας και προτάσεις για μελλοντικές βελτιώσεις.

Κεφάλαιο1ο :Αντικείμενο της εργασίας

1.1 Εισαγωγή

Η σύγχρονη εποχή χαρακτηρίζεται από ραγδαίες εξελίξεις στην τεχνολογία, οι οποίες επηρεάζουν όλους τους τομείς της ανθρώπινης δραστηριότητας. Μία από τις τεχνολογίες που έχουν τραβήξει ιδιαίτερα το ενδιαφέρον των ερευνητών, των προγραμματιστών και των τελικών χρηστών είναι η επαυξημένη πραγματικότητα (AugmentedReality - AR). Η επαυξημένη πραγματικότητα συνδυάζει ψηφιακά δεδομένα και πολυμέσα με τον φυσικό κόσμο σε πραγματικό χρόνο, προσφέροντας μια δυναμική και διαδραστική εμπειρία στον χρήστη. Σε αντίθεση με την εικονική πραγματικότητα, η οποία απομονώνει τον χρήστη σε έναν πλήρως ψηφιακό κόσμο, η AR λειτουργεί ως «επικάλυψη» πληροφορίας πάνω στην πραγματικότητα, ενισχύοντας την αντίληψη του χρήστη για τον χώρο που τον περιβάλλει.Η πρόσβαση σε τεχνολογίες AR έχει γίνει πλέον εφικτή μέσω των φορητών συσκευών που χρησιμοποιούμε καθημερινά, όπως smartphones και tablets. Οι συσκευές αυτές διαθέτουν κάμερες, αισθητήρες κίνησης και γεωεντοπισμού, καθώς και επεξεργαστική ισχύ ικανή να υποστηρίξει εφαρμογές με έντονο διαδραστικό χαρακτήρα. Το αποτέλεσμα είναι η δημιουργία εφαρμογών που μπορούν να χρησιμοποιηθούν για εκπαίδευση, ψυχαγωγία, προβολή προϊόντων, υποστήριξη επαγγελματιών, ακόμα και για τουριστικούς σκοπούς.Ο τουρισμός αποτελεί έναν από τους πιο ενδιαφέροντες και παραγωγικούς τομείς εφαρμογής της AR, καθώς δίνει τη δυνατότητα στον επισκέπτη να αλληλεπιδρά με το πολιτιστικό και ιστορικό περιβάλλον με τρόπο μοναδικό. Η χρήση τεχνολογιών όπως η AR και το GPS επιτρέπει τη δημιουργία εφαρμογών οι οποίες μετατρέπουν μια απλή περιήγηση σε βιωματική εμπειρία, προσφέροντας πληροφορίες, πολυμέσα και εικονικά αντικείμενα που εναρμονίζονται με το φυσικό τοπίο και τη γεωγραφική θέση του χρήστη.Η παρούσα πτυχιακή εργασία εντάσσεται σε αυτό το πλαίσιο και στοχεύει στην ανάπτυξη μιας εφαρμογής που να συνδυάζει επαυξημένη πραγματικότητα, χαρτογραφικά δεδομένα και γεωεντοπισμό, ώστε να εμπλουτίζει την εμπειρία του επισκέπτη με διαδραστικό και καινοτόμο τρόπο.

1.2 Σκοπός και Στόχοι

Ο βασικός σκοπός της πτυχιακής εργασίας είναι η υλοποίηση μιας τουριστικής εφαρμογής επαυξημένης πραγματικότητας που επιτρέπει στους χρήστες να εξερευνούν σημεία ιστορικού και πολιτιστικού ενδιαφέροντος μέσα από το κινητό τους τηλέφωνο ή άλλη φορητή συσκευή. Η εφαρμογή αξιοποιεί το GPS για τον προσδιορισμό της θέσης του χρήστη και εμφανίζει σχετικές πληροφορίες ανάλογα με την τοποθεσία του. Στοχεύει στη δημιουργία ενός διαδραστικού περιβάλλοντος, όπου ο χρήστης δεν είναι απλός παρατηρητής, αλλά συμμετέχει ενεργά στην ανακάλυψη του χώρου που τον περιβάλλει.

Οι κύριοι στόχοι της εργασίας συνοψίζονται ως εξής:

- Να προσφέρει μια καθηλωτική και διαδραστική εμπειρία στον χρήστη, ενισχύοντας το ενδιαφέρον του για τον χώρο που επισκέπτεται.
- Να αξιοποιήσει τη δύναμη της επαυξημένης πραγματικότητας για την παρουσίαση πολυμεσικού περιεχομένου (εικόνες, βίντεο, πληροφοριακά κείμενα) σχετικού με τα σημεία ενδιαφέροντος.
- Να προωθήσει τον πολιτιστικό πλούτο και την ιστορική αξία τοπικών αξιοθέατων, δίνοντας έμφαση στην καινοτόμα τεχνολογική προβολή τους.

- Να καταστήσει την πλοήγηση απλή και κατανοητή ακόμη και για χρήστες χωρίς προηγούμενη τεχνική εμπειρία.

Η χρήση τεχνολογίας GPS επιτρέπει την ακριβή παρακολούθηση της θέσης του χρήστη σε πραγματικό χρόνο, ενώ η εφαρμογή αντιδρά άμεσα στις κινήσεις του, ενεργοποιώντας ή απενεργοποιώντας περιεχόμενο ανάλογα με το πού βρίσκεται. Έτσι, η εμπειρία παραμένει σχετική με το περιβάλλον, φυσική, και ουσιαστικά «αβίαστη», χωρίς να απαιτείται χειροκίνητη παρέμβαση.

1.3 Μεθοδολογία

Η μεθοδολογία που ακολουθήθηκε για την υλοποίηση της εφαρμογής βασίστηκε στην αξιοποίηση σύγχρονων τεχνολογιών ανάπτυξης για κινητές συσκευές. Η κύρια πλατφόρμα που χρησιμοποιήθηκε ήταν η Unity, ένα ισχυρό εργαλείο για ανάπτυξη διαδραστικών εφαρμογών με δυνατότητες 2D και 3D απεικόνισης, καθώς και υποστήριξη για mobile πλατφόρμες όπως το Android.Για την ενσωμάτωση της τεχνολογίας επαυξημένης πραγματικότητας, χρησιμοποιήθηκε η Vuforia, μία από τις πιο διαδεδομένες πλατφόρμες AR που επιτρέπει την αναγνώριση εικόνων (imagetargets) και την απόδοση ψηφιακού περιεχομένου στην κάμερα της συσκευής. Παράλληλα, έγινε χρήση του Cesium for Unity, μιας βιβλιοθήκης που υποστηρίζει την απόδοση γεωγραφικών δεδομένων και τρισδιάστατων χαρτών σε πραγματικό χρόνο, προσφέροντας ένα ιδιαίτερα ρεαλιστικό περιβάλλον πλοήγησης.Η διεπαφή χρήστη σχεδιάστηκε με γνώμονα την απλότητα, την προσβασιμότητα και τη φιλικότητα, ώστε η χρήση της εφαρμογής να είναι εύκολη και κατανοητή ακόμα και από άτομα που δεν έχουν ιδιαίτερη επαφή με την τεχνολογία. Κατά την ανάπτυξη, λήφθηκαν υπόψη η εργονομία της οθόνης, η ευκολία μετακίνησης από έναν χάρτη σε περιβάλλον AR και η ομαλή εμπειρία πλοήγησης.Η εργασία περιλαμβάνει τόσο την τεχνική τεκμηρίωση της υλοποίησης, όσο και την παρουσίαση των δυσκολιών που αντιμετωπίστηκαν, των τρόπων αντιμετώπισής τους, καθώς και την τελική δοκιμή της εφαρμογής σε πραγματικές συνθήκες, σε επιλεγμένα σημεία ενδιαφέροντος.

1.4 Επίλογος

Στο παρόν πρώτο κεφάλαιο έγινε μια αναλυτική παρουσίαση του θέματος και της κατεύθυνσης της πτυχιακής εργασίας. Αναδείχθηκε η σημασία της τεχνολογίας επαυξημένης πραγματικότητας στη σύγχρονη εποχή και η εφαρμογή της στον τομέα του τουρισμού. Παρουσιάστηκε ο βασικός στόχος της εργασίας, ο οποίος είναι η ανάπτυξη μιας φορητής εφαρμογής που αξιοποιεί την AR και την τεχνολογία γεωεντοπισμού για να προσφέρει μια νέα, καθηλωτική εμπειρία ξενάγησης.Επιπλέον, αναλύθηκαν οι γενικές μεθοδολογικές αρχές που ακολουθήθηκαν για την ανάπτυξη της εφαρμογής, όπως η επιλογή κατάλληλων τεχνολογιών, η σχεδίαση της διεπαφής και η εφαρμογή σε πραγματικές συνθήκες. Στα επόμενα κεφάλαια θα παρουσιαστούν αναλυτικά οι τεχνολογικές επιλογές, η δομή και η λειτουργία του συστήματος, ο πηγαίος κώδικας, καθώς και τα αποτελέσματα των δοκιμών και αξιολογήσεων που πραγματοποιήθηκαν.

Κεφάλαιο 2ο:Προσδιορισμός Αναγκών και Απαιτήσεων της Εφαρμογής

2.1 Εισαγωγή

Η ανάπτυξη μιας εφαρμογής δεν ξεκινά ποτέ απλώς με την κωδικοποίηση. Πριν από αυτό, είναι απαραίτητο να γίνει μια ξεκάθαρη ανάλυση του προβλήματος που επιδιώκουμε να λύσουμε και των στόχων που θέλουμε να πετύχουμε. Με απλά λόγια, πρέπει πρώτα να προσδιοριστεί ποιος θα τη χρησιμοποιήσει, σε ποιο πλαίσιο και για ποιο λόγο.Στο συγκεκριμένο κεφάλαιο γίνεται μια προσπάθεια να αποτυπωθούν όλες οι βασικές ανάγκες που οδήγησαν στην ιδέα για την εφαρμογή, οι στόχοι που τέθηκαν εξ αρχής, αλλά και οι τεχνικές και λειτουργικές απαιτήσεις που έπρεπε να καλυφθούν ώστε η εφαρμογή να είναι πρακτική, χρηστική και λειτουργική. Παράλληλα, παρουσιάζονται σενάρια χρήσης που δείχνουν πώς ακριβώς θα αξιοποιηθεί στην πράξη από τον τελικό χρήστη.

2.2 Προσδιορισμός Προβλήματος

Στον τομέα του τουρισμού και της ξενάγησης, παρατηρείται τις τελευταίες δεκαετίες μια στασιμότητα σε ό,τι αφορά τον τρόπο παρουσίασης της πληροφορίας. Παρά την τεχνολογική πρόοδο, η εμπειρία του επισκέπτη σε πολιτιστικά σημεία συχνά περιορίζεται σε πινακίδες, έντυπα ή απλές ηχητικές ξεναγήσεις. Παράλληλα, οι περισσότεροι τουρίστες χρησιμοποιούν smartphones καθημερινά, τα οποία προσφέρουν δυνατότητες που παραμένουν αναξιοποίητες.Το πρόβλημα που εντοπίστηκε είναι η έλλειψη διαδραστικότητας και προσωποποίησης στην ξενάγηση, καθώς και η απουσία δυναμικής πληροφόρησης που να συνδέεται με τη θέση του χρήστη στον χώρο. Η τεχνολογία επαυξημένης πραγματικότητας (AR), σε συνδυασμό με τον γεωεντοπισμό (GPS), μπορεί να προσφέρει μια λύση σε αυτό, δημιουργώντας έναν πιο «ζωντανό» τρόπο παρουσίασης πληροφοριών, προσαρμοσμένο στο περιβάλλον και την κίνηση του επισκέπτη.

2.3 Στόχοι της Εφαρμογής

Με βάση τα παραπάνω, οι βασικοί στόχοι που τέθηκαν κατά την υλοποίηση της εφαρμογής είναι οι εξής:

Λειτουργικοί στόχοι:Να εμφανίζεται στον χρήστη σχετική πληροφορία όταν βρίσκεται κοντά σε ένα σημείο ενδιαφέροντος.Να δίνεται η δυνατότητα προβολής περιεχομένου μέσω AR (εικόνα, βίντεο, περιγραφή).Να ακολουθείται η πραγματική του θέση μέσω GPS και να ενημερώνεται η εφαρμογή σε πραγματικό χρόνο.

Εμπειρικοί στόχοι:Να νιώθει ο χρήστης ότι αλληλεπιδρά με τον χώρο και δεν ακολουθεί απλώς μια στατική παρουσίαση.Να είναι η εμπειρία όσο το δυνατόν πιο φυσική και "αβίαστη", χωρίς τεχνικές δυσκολίες ή πολύπλοκα μενού.

Πολιτιστικοί στόχοι:Να αναδεικνύονται σημεία με ιστορικό ή πολιτιστικό ενδιαφέρον με έναν πιο σύγχρονο και ελκυστικό τρόπο.Να ενισχυθεί το ενδιαφέρον των επισκεπτών, ιδιαίτερα των νεότερων ηλικιών, μέσα από ένα πιο τεχνολογικό μέσο ξενάγησης.

2.4 Ανάλυση Απαιτήσεων

Κατά τον σχεδιασμό της εφαρμογής, έγινε προσπάθεια να καταγραφούν και να ικανοποιηθούν συγκεκριμένες απαιτήσεις, που κατηγοριοποιούνται ως εξής:

2.4.1 Λειτουργικές Απαιτήσεις

Οι λειτουργικές απαιτήσεις περιγράφουν τις βασικές δυνατότητες που πρέπει να προσφέρει η εφαρμογή ώστε να θεωρείται επιτυχής:Εντοπισμός θέσης χρήστη με χρήση GPS, για δυναμική παρακολούθηση κατά τη διάρκεια της περιήγησης.Προβολή ψηφιακού περιεχομένου μέσω κάμερας όταν ο χρήστης φτάνει σε συγκεκριμένα σημεία.Αλληλεπίδραση με το περιβάλλον με χρήση imagetarget (Vuforia).Επιλογή τρόπου παρουσίασης (π.χ. εικόνα, βίντεο, περιγραφή).Επιστροφή στον χάρτη ανά πάσα στιγμή μέσω κουμπιού ("Map") για συνέχιση της περιήγησης.Απλή και κατανοητή διεπαφή για να μπορούν να τη χρησιμοποιήσουν όλοι, χωρίς τεχνικές γνώσεις.

2.4.2 Μη Λειτουργικές Απαιτήσεις

Εκτός από τις βασικές λειτουργίες, υπάρχουν και κάποιες επιπλέον προδιαγραφές που σχετίζονται με την εμπειρία χρήσης συνολικά:
Η εφαρμογή πρέπει να είναι συμβατή με Android συσκευές.Να έχει γρήγορο χρόνο απόκρισης και να μην καθυστερεί κατά την αλλαγή θέσης.Να λειτουργεί χωρίς σύνδεση στο διαδίκτυο, εφόσον το περιεχόμενο έχει ήδη φορτωθεί (όπου είναι δυνατό).Να διαχειρίζεται σωστά σφάλματα, όπως π.χ. αν δεν υπάρχει GPS σήμα ή δεν αναγνωρίζεται εικόνα.

2.5 Σενάρια Χρήσης

Για καλύτερη κατανόηση της λειτουργίας της εφαρμογής, ακολουθούν παραδείγματα σεναρίων χρήσης που περιγράφουν πώς αλληλεπιδρά ένας χρήστης με το σύστημα, σε πραγματικές συνθήκες.

Σενάριο 1 – Περιήγηση στο κέντρο της Θεσσαλονίκης (Λευκός Πύργος)

Ένας επισκέπτης που περπατά στο κέντρο της Θεσσαλονίκης ανοίγει την εφαρμογή και δίνει άδεια πρόσβασης στο GPS. Η εφαρμογή εντοπίζει αυτόματα τη θέση του και εμφανίζει στον χάρτη έναν δείκτη που υποδηλώνει πού βρίσκεται. Καθώς πλησιάζει τον Λευκό Πύργο, και όταν βρεθεί σε ακτίνα 10 μέτρων, ενεργοποιείται ένα πλαίσιο στο κάτω μέρος της οθόνης με τον τίτλο του σημείου (π.χ. "Λευκός Πύργος") και μια επιλογή για να ανοίξει την κάμερα.Αν ο χρήστης πατήσει την επιλογή, η εφαρμογή ανοίγει την κάμερα και του ζητά να σκανάρει ένα imagetarget (π.χ. αφίσα ή κάρτα τοποθετημένη κοντά στο σημείο). Μόλις ολοκληρωθεί επιτυχώς η σάρωση, εμφανίζεται στην οθόνη του κινητού είτε μια εικόνα, είτε ένα βίντεο, είτε μια περιγραφή σχετική με τον Λευκό Πύργο — ανάλογα με το τι επιλέξει ο ίδιος από τα διαθέσιμα κουμπιά στο κάτω μέρος.Ανά πάσα στιγμή, ο χρήστης μπορεί να επιστρέψει στον χάρτη, πατώντας το κουμπί "Map" που βρίσκεται κάτω δεξιά, εφόσον βρίσκεται μέσα στη λειτουργία κάμερας.

Σενάριο 2 – Επίσκεψη στην Πλατεία Αριστοτέλους (Άγαλμα Βενιζέλου)

Σε άλλη περίπτωση, ένας φοιτητής θέλει να δοκιμάσει την εφαρμογή στο πλαίσιο παρουσίασης. Βρίσκεται στην Πλατεία Αριστοτέλους, κοντά στο άγαλμα του Ελευθερίου Βενιζέλου. Η εφαρμογή τον εντοπίζει μέσω GPS και, όταν πλησιάσει σε απόσταση μικρότερη των 10 μέτρων από το σημείο, εμφανίζεται στο κάτω μέρος της οθόνης η σχετική ένδειξη για τοποθεσία ("Πλατεία Αριστοτέλους") και ενεργοποιείται η δυνατότητα για χρήση της κάμερας.Ο χρήστης επιλέγει να σκανάρει το ειδικό imagetarget που έχει εκτυπώσει ή τοποθετηθεί προσωρινά στο σημείο, και αμέσως μετά του εμφανίζονται πληροφορίες για την Πλατεία Αριστοτέλους, οι οποίες μπορεί να είναι με τη μορφή εικόνας, σύντομου βίντεο ή περιγραφικού κειμένου.Όπως και στο προηγούμενο σενάριο, υπάρχει η δυνατότητα επιστροφής στον διαδραστικό χάρτη με το πάτημα του κουμπιού "Map" κάτω δεξιά στην οθόνη.

2.6 Επίλογος

Στο παρόν κεφάλαιο παρουσιάστηκε αναλυτικά το πρόβλημα που εντοπίστηκε στον χώρο της τουριστικής εμπειρίας, καθώς και η αναγκαιότητα για μια πιο μοντέρνα, διαδραστική προσέγγιση μέσω της επαυξημένης πραγματικότητας και του GPS. Περιγράφηκαν οι στόχοι της εφαρμογής, οι λειτουργικές και μη λειτουργικές απαιτήσεις της, καθώς και παραδείγματα ρεαλιστικής χρήσης που βοηθούν στο να κατανοηθεί το πώς λειτουργεί στην πράξη.Ακολουθεί το επόμενο κεφάλαιο, όπου θα παρουσιαστεί η αρχιτεκτονική της εφαρμογής, ο κύκλος ζωής της και η τεχνολογική βάση πάνω στην οποία χτίστηκε.

Κεφάλαιο 3ο : Τεχνολογίες Υλοποίησης Συστήματος

3.1 Εισαγωγή

Η ταχύτατη εξέλιξη των τεχνολογιών επαυξημένης πραγματικότητας (AR), των γεωχωρικών δεδομένων και της κινητής ανάπτυξης έχει ανοίξει τον δρόμο για τη δημιουργία εφαρμογών που συνδυάζουν τον πραγματικό και τον ψηφιακό κόσμο με τρόπους που μέχρι πρόσφατα θεωρούνταν αδύνατοι. Σε αυτό το πλαίσιο, η υλοποίηση της παρούσας εφαρμογής δεν αποτελεί απλώς έναν τεχνικό πειραματισμό, αλλά μια απόπειρα αξιοποίησης σύγχρονων και ευέλικτων εργαλείων με στόχο τη δημιουργία ενός καινοτόμου ψηφιακού ξεναγού.

Για την επίτευξη αυτού του στόχου, επιλέχθηκε ένα σύνολο τεχνολογιών που προσφέρουν υψηλή διαλειτουργικότητα, επεκτασιμότητα, καθώς και ενεργή υποστήριξη από τη διεθνή κοινότητα ανάπτυξης. Ο τεχνολογικός πυρήνας της εφαρμογής αποτελείται από τις πλατφόρμες Unity, Vuforia, Cesium for Unity και την τεχνολογία GPS, ενώ καθοριστικό ρόλο διαδραματίζει και η έννοια των σημείων ενδιαφέροντος (Points of Interest – POIs), τα οποία ενσωματώνονται στο περιβάλλον και ενεργοποιούν το περιεχόμενο AR ανάλογα με τη θέση του χρήστη.

Η Unity χρησιμοποιήθηκε ως κεντρικό περιβάλλον ανάπτυξης, παρέχοντας τα εργαλεία για δημιουργία διαδραστικών σκηνών και υποστήριξη για φορητές συσκευές. Η Vuforia αξιοποιήθηκε για την αναγνώριση εικόνων στον φυσικό χώρο και την απεικόνιση επαυξημένου περιεχομένου, βασισμένου σε imagetargets. Η χρήση του Cesium πρόσφερε την απαραίτητη υποδομή για ρεαλιστική αναπαράσταση του παγκόσμιου χαρτογραφικού περιβάλλοντος, ενώ το GPS συνέδεσε την εικονική εμπειρία με την πραγματική θέση του χρήστη.

Η σύνθεση αυτών των τεχνολογιών δεν έγινε αυθαίρετα, αλλά με βάση λειτουργικά και εμπειρικά κριτήρια, όπως η ακρίβεια στον εντοπισμό θέσης, η δυναμική απόδοση περιεχομένου και η προσαρμοστικότητα της εμπειρίας σε διαφορετικά σημεία και συνθήκες. Έτσι, η τελική εφαρμογή δεν περιορίζεται στην παθητική παρουσίαση πληροφορίας, αλλά προσφέρει στον χρήστη μια ενεργή, εξερευνητική και καθηλωτική αλληλεπίδραση με τον χώρο.

3.2 Τεχνολογίες που Χρησιμοποιήθηκαν

3.2.1Unity

Η Unity αποτέλεσε τη βασική πλατφόρμα ανάπτυξης της εφαρμογής μου, καθώς προσφέρει ένα εξαιρετικά ευέλικτο και δυναμικό περιβάλλον για τη δημιουργία διαδραστικών εμπειριών. Πρόκειται για μια από τις πιο ευρέως χρησιμοποιούμενες μηχανές ανάπτυξης παιχνιδιών και εφαρμογών παγκοσμίως, με πλούσια υποστήριξη για 2D και 3D γραφικά, εργαλεία χρήστη (UI), υποστήριξη για επαυξημένη πραγματικότητα (AR), και τεράστια ποικιλία από plugins και third-party βιβλιοθήκες.

Αυτό που κάνει τη Unity να ξεχωρίζει είναι η δυνατότητα ενσωμάτωσης πολλών τεχνολογιών σε ένα κοινό περιβάλλον. Από φυσική αλληλεπίδραση με αντικείμενα, μέχρι διαχείριση σκηνών, animation και scripting, όλα μπορούν να υλοποιηθούν στο ίδιο interface, χωρίς ανάγκη για εξωτερικά εργαλεία. Επιπλέον, η ευκολία με την οποία κάποιος μπορεί να ενσωματώσει plugins, όπως για παράδειγμα το

Cesium και το Vuforia, καθιστά τη Unity εξαιρετικά προσαρμόσιμη σε σύνθετα project, όπως μια εφαρμογή επαυξημένης πραγματικότητας με χρήση GPS.

Αξίζει να αναφερθεί ότι η Unity υποστηρίζει και ανάπτυξη για πολλαπλές πλατφόρμες (multiplatformdevelopment), δηλαδή μπορεί να δημιουργήσει κανείς εφαρμογές που «τρέχουν» σε Android, iOS, Windows και άλλες πλατφόρμες με κοινή βάση κώδικα. Αν και η συγκεκριμένη εφαρμογή αναπτύχθηκε αποκλειστικά για Android συσκευές, η ύπαρξη αυτής της δυνατότητας παραμένει σημαντική, καθώς ενισχύει τη μελλοντική επεκτασιμότητα της εργασίας, σε περίπτωση που χρειαστεί προσαρμογή για άλλες πλατφόρμες.

Ένα άλλο βασικό πλεονέκτημα της Unity είναι η πλούσια βιβλιοθήκη εργαλείων UI που διαθέτει, επιτρέποντας την εύκολη δημιουργία διαδραστικών και φιλικών προς τον χρήστη διεπαφών. Αν και το κομμάτι του UI με δυσκόλεψε αρχικά, ιδιαίτερα όσον αφορά την ευθυγράμμιση των στοιχείων στην οθόνη και τη σωστή προσαρμογή τους σε διαφορετικές αναλύσεις κινητών, με την πάροδο του χρόνου κατάφερα να εξοικειωθώ περισσότερο και να δημιουργήσω ένα λειτουργικό και σταθερό περιβάλλον χρήστη, προσαρμοσμένο στις ανάγκες της εφαρμογής.

Η σύνταξη του κώδικα έγινε με τη γλώσσα προγραμματισμού C#, η οποία χρησιμοποιείται ευρέως στο οικοσύστημα της Unity. Μέσα από C# scripts υλοποιήθηκαν όλες οι βασικές λειτουργίες της εφαρμογής: από την παρακολούθηση της θέσης του χρήστη μέσω GPS, μέχρι τη δυναμική εμφάνιση κουμπιών και πληροφοριών, καθώς και την εναλλαγή μεταξύ των διαφορετικών οθονών (χάρτης, AR κάμερα, πληροφοριακό περιεχόμενο).

Ένα ακόμα σημαντικό πλεονέκτημα της Unity είναι η δυναμική διαχείριση αντικειμένων σε πραγματικό χρόνο. Καθώς η εφαρμογή βασίζεται σε αλληλεπίδραση με τον φυσικό χώρο, ήταν σημαντικό όλα τα ψηφιακά στοιχεία να μπορούν να "αντιδρούν" άμεσα στις κινήσεις του χρήστη, τόσο σε επίπεδο πλοήγησης όσο και στην ενεργοποίηση πληροφοριών. Η Unity, χάρη στο scenesystem και το objecthierarchy της, καθιστά αυτή τη διαχείριση εξαιρετικά απλή, κάτι που με βοήθησε να υλοποιήσω πιο γρήγορα και αποτελεσματικά τη λογική της εφαρμογής.

Φυσικά, υπάρχουν και άλλες μηχανές ανάπτυξης που θα μπορούσαν να χρησιμοποιηθούν, όπως το UnrealEngine ή το Godot, ωστόσο η Unity είχε δύο πολύ βασικά πλεονεκτήματα για μένα:Είχα ήδη προηγούμενη τριβή με το περιβάλλον της, οπότε δεν ξεκίνησα εντελώς από το μηδέν.Διαθέτει τεράστια κοινότητα και εκτενή τεκμηρίωση, με tutorials, videos, forums και οδηγούς για οποιοδήποτε πρόβλημα αντιμετωπίσει κάποιος.

Αναγκαστικά, για την εργασία αυτή, έπρεπε να χρησιμοποιηθεί Unity, καθώς υποστήριζε άμεσα τα εργαλεία που χρειαζόμουν (Vuforia, Cesium, GPS integration). Ωστόσο, αυτή η "αναγκαιότητα" λειτούργησε τελικά θετικά, αφού μου έδωσε την ευκαιρία να εξερευνήσω εις βάθος μια από τις πιο ισχυρές πλατφόρμες δημιουργίας εφαρμογών.

Στο μέλλον, θα με ενδιέφερε να εξερευνήσω περισσότερο το AR Foundation, το ενιαίο framework της Unity για ανάπτυξη επαυξημένης πραγματικότητας που συνδυάζει δυνατότητες από Vuforia, ARCore και ARKit, και παρέχει ακόμη μεγαλύτερη ευελιξία. Για την παρούσα εργασία, όμως, η κλασική προσέγγιση Unity σε συνδυασμό με Vuforia και Cesium αποδείχθηκε επαρκής και πλήρως λειτουργική.

Συνοψίζοντας, η Unity αποτέλεσε τη ραχοκοκαλιά της εφαρμογής, προσφέροντας το κατάλληλο τεχνικό υπόβαθρο ώστε να ενσωματωθούν διαφορετικές τεχνολογίες και να προσφερθεί στον τελικό χρήστη μια συνεκτική, σύγχρονη και καθηλωτική εμπειρία..

3.2.2 Χάρτες και Cesium

Η αξιοποίηση χαρτογραφικών δεδομένων σε εφαρμογές επαυξημένης πραγματικότητας αποτελεί έναν από τους πιο εντυπωσιακούς και χρήσιμους τρόπους για να συνδέσουμε τον ψηφιακό κόσμο με τον φυσικό χώρο. Ιδιαίτερα όταν μιλάμε για εφαρμογές που βασίζονται σε τοποθεσίες, όπως η δική μου, η χρήση ενός ψηφιακού χάρτη που απεικονίζει την πραγματική γεωγραφική θέση του χρήστη σε συνδυασμό με επιπλέον πληροφορίες, είναι απολύτως καθοριστική.

Για αυτό τον σκοπό, χρησιμοποίησα το Cesium for Unity, μια πολύ ισχυρή πλατφόρμα ανοιχτού κώδικα που δίνει τη δυνατότητα απόδοσης γεωχωρικών δεδομένων σε 3D μέσα στο περιβάλλον της Unity. Σε αντίθεση με άλλες χαρτογραφικές βιβλιοθήκες, όπως το GoogleMaps SDK ή το Mapbox, το Cesium έχει το πλεονέκτημα ότι δεν απαιτεί τη χρήση API keys ή συνδρομές, γεγονός που το καθιστά ιδιαίτερα βολικό για προσωπικές, πανεπιστημιακές ή ερευνητικές εφαρμογές.

Η βασική λειτουργία του Cesium είναι ότι μετατρέπει τους κλασικούς δισδιάστατους χάρτες σε πλήρως τρισδιάστατο γεωγραφικό περιβάλλον, προσφέροντας έτσι μία πιο ρεαλιστική και καθηλωτική εμπειρία στον χρήστη. Αυτό ήταν ακριβώς που χρειαζόμουν για την εφαρμογή μου: να μπορώ να εντάξω μέσα στο ψηφιακό περιβάλλον της Unity ένα μοντέλο που να αναπαριστά την περιοχή της Θεσσαλονίκης με ακρίβεια, ώστε να τοποθετήσω σημεία ενδιαφέροντος (POIs) σε συγκεκριμένα σημεία πάνω στον χάρτη.

Το Cesium υποστηρίζει δορυφορικές εικόνες υψηλής ανάλυσης, ανάγλυφα εδάφους (terrain) και συντεταγμένες που μπορούν να ευθυγραμμιστούν με πραγματικά γεωγραφικά δεδομένα. Μέσω της ενσωμάτωσής του στην Unity, είχα τη δυνατότητα να «καρφιτσώσω» εικονικά αντικείμενα, μοντέλα ή περιεχόμενο σε ακριβείς τοποθεσίες, όπως ο Λευκός Πύργος, η Πλατεία Αριστοτέλους ή άλλα σημεία που ήθελα να αναδείξω.

Πρακτικά, το Cesium λειτούργησε σαν ο "χάρτης βάσης" της εφαρμογής. Πάνω σε αυτόν τοποθέτησα τα POIs, τα οποία συνοδεύονταν από περιεχόμενο όπως εικόνες, περιγραφές ή βίντεο. Η λειτουργία του σε συνδυασμό με το GPS της συσκευής επέτρεψε την αναγνώριση της θέσης του χρήστη και τη δυναμική προβολή πληροφοριών καθώς αυτός μετακινούνταν στον φυσικό χώρο.

Αυτό που μου έκανε ιδιαίτερη εντύπωση ήταν πόσο ομαλά και λειτουργικά ενσωματώνεται το Cesium μέσα στο Unity. Παρά το γεγονός ότι δεν είχα προηγούμενη εμπειρία με αυτό το εργαλείο, κατάφερα να το ενσωματώσω αρκετά γρήγορα, ακολουθώντας τις οδηγίες της κοινότητας και με δοκιμές μέσα στο project. Ειδικά το γεγονός ότι μπορούσα να προβάλλω τη Θεσσαλονίκη με τέτοια λεπτομέρεια, μου έδωσε την αίσθηση πως δημιουργώ μια εφαρμογή που λειτουργεί «πάνω στον πραγματικό κόσμο», και όχι απλώς σε ένα τεχνητό περιβάλλον.

Εκτός από την τεχνική του ακρίβεια, το Cesium μου έδωσε τη δυνατότητα να δημιουργήσω μια εμπειρία που είναι οπτικά πλούσια και εντυπωσιακή. Οι χάρτες δεν είναι πλέον κάτι στατικό — μετατρέπονται σε κάτι «ζωντανό», το οποίο αλληλεπιδρά με τον χρήστη. Αυτό δίνει άλλη διάσταση

στην επαυξημένη πραγματικότητα: δεν πρόκειται απλώς για προβολή περιεχομένου πάνω σε ένα σημείο, αλλά για μια συνολική εμπειρία πλοήγησης μέσα σε έναν εμπλουτισμένο χώρο.

Η χρήση του Cesium είχε και πρακτικά οφέλη στην ανάπτυξη: μπορούσα να πειραματίζομαι με τις τοποθεσίες και να μετακινώ τα POIs εύκολα, να ρυθμίζω το zoom και την κάμερα, αλλά και να απεικονίζω την τοποθεσία του χρήστη με σχετική ακρίβεια. Επιπλέον, επειδή πρόκειται για open-source πλατφόρμα, υπήρχε αρκετό υλικό online που με βοήθησε όταν προέκυπταν τεχνικές δυσκολίες, ιδιαίτερα στην ευθυγράμμιση συντεταγμένων και στην προβολή υλικού σε διαφορετικά επίπεδα υψομέτρου.

Τέλος, το Cesium, σε συνδυασμό με το GPS, έδωσε μια αίσθηση εξερεύνησης στον χρήστη: καθώς προχωρούσε στον φυσικό χώρο, μπορούσε να ανακαλύπτει σταδιακά τα σημεία ενδιαφέροντος, τα οποία ενεργοποιούνταν με βάση την τοποθεσία του. Αυτό προσέφερε μία πιο φυσική και λιγότερο «τεχνητή» εμπειρία στον χρήστη, κάτι που ήταν εξ αρχής ζητούμενο για την εφαρμογή.

Συνοψίζοντας, το Cesium for Unity αποτέλεσε ένα από τα πιο κρίσιμα τεχνικά εργαλεία στην ανάπτυξη της εφαρμογής. Μου επέτρεψε να δώσω γεωγραφική ακρίβεια, ρεαλισμό και οπτικό βάθος στην εφαρμογή, δημιουργώντας ένα αποτέλεσμα που συνδυάζει τεχνολογία χαρτών και επαυξημένης πραγματικότητας με τρόπο φυσικό, λειτουργικό και εντυπωσιακό.

3.2.3Vuforia και Επαυξημένη Πραγματικότητα(AR)

Η επαυξημένη πραγματικότητα (AugmentedReality – AR) είναι μια από τις πιο εντυπωσιακές τεχνολογίες της εποχής μας, καθώς μας δίνει τη δυνατότητα να «δούμε» τον φυσικό κόσμο γύρω μας εμπλουτισμένο με ψηφιακές πληροφορίες. Εικόνες, βίντεο, τρισδιάστατα μοντέλα ή απλό κείμενο μπορούν να προβάλλονται πάνω σε πραγματικά αντικείμενα ή σημεία του περιβάλλοντος, αρκεί να κοιτάξουμε μέσα από την κάμερα του κινητού ή του tablet μας. Αυτό το μείγμα ψηφιακού και φυσικού στοιχείου είναι που κάνει την AR τόσο μοναδική και ελκυστική, ειδικά όταν εφαρμόζεται στον χώρο του τουρισμού και της πολιτιστικής ξενάγησης.

Για την ανάπτυξη της εφαρμογής, χρησιμοποίησα την πλατφόρμα Vuforia, μια από τις πιο γνωστές και αξιόπιστες τεχνολογίες στον χώρο της επαυξημένης πραγματικότητας, ιδιαίτερα όταν πρόκειται για image-based AR (με markers). Το βασικό της πλεονέκτημα είναι ότι αναγνωρίζει εικόνες μέσα από την κάμερα και πάνω σε αυτές προβάλλει περιεχόμενο που ορίζουμε εμείς. Έτσι, μπορεί κανείς να "σκανάρει" μια αφίσα, ένα ταμπελάκι ή μια κάρτα, και να εμφανιστεί πάνω της ένα σχετικό βίντεο, μια φωτογραφία ή μια σύντομη περιγραφή.

Η επιλογή της Vuforia έγινε κυρίως για λόγους αξιοπιστίας και συμβατότητας με την πλατφόρμα ανάπτυξης Unity. Το γεγονός ότι λειτουργεί απρόσκοπτα σε Android και iOS, αλλά και ότι υποστηρίζει άμεσα ενοποίηση με το Unity, με έκανε να την προτιμήσω έναντι άλλων επιλογών. Παράλληλα, υποστηρίζει τόσο με markers (imagetargets) όσο και χωρίς markers (surfacetracking, planedetection), αν και στην παρούσα εργασία αξιοποιήθηκε η πρώτη μέθοδος — δηλαδή η αναγνώριση συγκεκριμένων εικόνων που έχουν σχεδιαστεί για να «ξεκλειδώνουν» περιεχόμενο.

Η λογική ήταν απλή αλλά αποτελεσματική: όταν ο χρήστης πλησιάζει ένα προκαθορισμένο σημείο ενδιαφέροντος και ενεργοποιεί την κάμερα, καλείται να σκανάρει μία εικόνα (imagetarget) που

έχουμε τοποθετήσει στο φυσικό περιβάλλον. Μόλις γίνει η αναγνώριση, εμφανίζεται στην οθόνη η επιλεγμένη πληροφορία — είτε πρόκειται για φωτογραφία, είτε για περιγραφή, είτε για βίντεο. Έτσι, η εμπειρία γίνεται ζωντανή και αλληλεπιδραστική, αφού ο χρήστης δεν βλέπει απλώς έναν χάρτη ή ένα κουμπί, αλλά συμμετέχει ενεργά στην ανακάλυψη του περιεχομένου.

Ένα σημαντικό στοιχείο της AR και ειδικά της Vuforia, είναι η σταθερότητα και η απόδοση του περιεχομένου στην κάμερα. Ακόμη κι αν ο χρήστης κουνήσει ελαφρώς τη συσκευή ή αλλάξει οπτική γωνία, η εφαρμογή συνεχίζει να εμφανίζει σωστά την πληροφορία, χωρίς να "τρέμει" ή να αποσυντονίζεται. Αυτό είναι πολύ βασικό όταν θέλουμε να προσφέρουμε μια θετική εμπειρία σε έναν χρήστη που δεν έχει τεχνικές γνώσεις, όπως ένας τουρίστας ή ένας επισκέπτης σε ανοιχτό χώρο.

Αυτό που μου άρεσε στην Vuforia ήταν ότι δεν χρειάζονταν περίπλοκα setups για να λειτουργήσει. Μέσα από το VuforiaEngine και το VuforiaConfigurationPanel στην Unity, μπορούσα να ανεβάσω τα imagetargets μου, να τα συνδέσω με scripts και να επιλέξω τι περιεχόμενο θα προβάλλεται. Παράλληλα, το σύστημα υποστηρίζει την τοποθέτηση διαφορετικών τύπων περιεχομένου: από απλές εικόνες και ήχο, μέχρι animations ή ακόμα και μοντέλα 3D.

Ένα ακόμα πλεονέκτημα ήταν η δυνατότητα ελέγχου της εμπειρίας από το χρήστη. Στο δικό μου project, ο χρήστης μπορεί να επιλέξει τι θέλει να δει μόλις σκανάρει ένα σημείο: είτε εικόνα, είτε περιγραφή, είτε βίντεο. Αυτή η ευελιξία δεν ήταν δεδομένη, και η Vuforia βοήθησε σημαντικά στο να το εφαρμόσω χωρίς περιπλοκές.

Γενικότερα, η AR δεν είναι απλά ένα "διακοσμητικό" στοιχείο της εφαρμογής, αλλά ένα από τα βασικά χαρακτηριστικά της. Μέσω της AR, ο χρήστης δεν παρατηρεί απλώς τον χώρο — συμμετέχει. Μαθαίνει με τρόπο βιωματικό, "παίζει" με τον χώρο, και δημιουργεί μια εμπειρία που θυμάται περισσότερο. Αυτός είναι και ο λόγος που η επαυξημένη πραγματικότητα αποτελεί πλέον ένα εργαλείο που χρησιμοποιείται ευρέως, όχι μόνο στον τουρισμό, αλλά και στην εκπαίδευση, την αρχιτεκτονική, την τέχνη και τη βιομηχανία.

Συνοψίζοντας, η τεχνολογία AR και η πλατφόρμα Vuforia έπαιξαν καθοριστικό ρόλο στην επιτυχία της εφαρμογής. Χάρη στην ευκολία ενσωμάτωσης, τη σταθερότητα και τις δυνατότητες διαχείρισης περιεχομένου, κατάφερα να προσφέρω μια εμπειρία πλοήγησης πολύ πιο διαδραστική, ζωντανή και ελκυστική για τον τελικό χρήστη.

3.2.4GPS

Το GPS (GlobalPositioningSystem) είναι μια από τις πιο κρίσιμες τεχνολογίες που αξιοποιήθηκαν στην εφαρμογή μου, αφού όλη η λογική της πλοήγησης και της ενεργοποίησης των σημείων ενδιαφέροντος (POIs) βασίζεται στην τοποθεσία του χρήστη. Χάρη στο GPS, η εφαρμογή «γνωρίζει» πού ακριβώς βρίσκεται ο χρήστης στον φυσικό χώρο και μπορεί να του εμφανίσει τις κατάλληλες πληροφορίες τη σωστή στιγμή, δημιουργώντας μια δυναμική και προσωποποιημένη εμπειρία.

Η ενσωμάτωση του GPS στην Unity, παρόλο που υπάρχουν αρκετά παραδείγματα και βιβλιοθήκες, δεν ήταν όσο απλή φανταζόμουν στην αρχή. Ο κύριος λόγος ήταν ότι το GPS «αγγίζει» προσωπικά δεδομένα, δηλαδή τη ζωντανή τοποθεσία του χρήστη, και επομένως χρειαζόταν να διαχειριστώ με μεγάλη προσοχή τα δικαιώματα πρόσβασης. Έπρεπε να μελετήσω καλά πώς να ζητάω άδεια χρήσης

τοποθεσίας, τόσο μέσα από τα settings της Android συσκευής όσο και από τα permissions στο Unityproject, και να βεβαιωθώ ότι όλα λειτουργούν σωστά πριν περάσω στη βασική λογική.

Η υλοποίηση απαιτούσε ένα script που παρακολουθούσε συνεχώς την τρέχουσα θέση του χρήστη, σε σχέση με τις γεωγραφικές συντεταγμένες κάθε POI. Με βάση αυτά τα δεδομένα, η εφαρμογή υπολόγιζε την απόσταση σε πραγματικό χρόνο και όταν αυτή έπεφτε κάτω από ένα όριο — συγκεκριμένα τα 10 μέτρα — ενεργοποιούσε τη δυνατότητα προβολής περιεχομένου. Πρακτικά, εμφανιζόταν ένα κουμπί "AR" στο κάτω μέρος της οθόνης, το οποίο μπορούσε να πατήσει ο χρήστης για να ανοίξει η κάμερα και να ξεκινήσει η εμπειρία επαυξημένης πραγματικότητας.

Αυτό το είδος λειτουργίας, όπου το περιεχόμενο «ξεκλειδώνει» μόνο όταν πλησιάζεις αρκετά, δίνει μια αίσθηση εξερεύνησης και παιχνιδιού. Δεν προσφέρεται όλη η πληροφορία από την αρχή, αλλά παρουσιάζεται οργανωμένα και τη σωστή στιγμή, δίνοντας την εντύπωση πως ο χρήστης "ανακαλύπτει" το περιβάλλον του.

Το GPS δεν λειτούργησε μόνο ως μηχανισμός ενεργοποίησης, αλλά και ως πυξίδα για πλοήγηση, αφού η εφαρμογή ενημέρωνε διαρκώς τη θέση του χρήστη στον χάρτη. Έτσι, μπορούσε να δει σε ποιο σημείο βρίσκεται, ποια POIs είναι κοντά του και να οργανώσει τη διαδρομή του ανάλογα. Αυτή η λειτουργία ήταν ιδιαίτερα χρήσιμη σε εξωτερικούς χώρους, όπου η ακρίβεια του GPS είναι υψηλή και το περιεχόμενο μπορούσε να ενεργοποιηθεί χωρίς καθυστέρηση.

Πέρα όμως από την τεχνική υλοποίηση, αυτό που με εντυπωσίασε περισσότερο ήταν το πώς η τεχνολογία αυτή άλλαξε τελείως τον τρόπο που αντιλαμβάνεσαι μια απλή βόλτα στην πόλη. Από εκεί που ο χάρτης είναι απλώς ένα στατικό εργαλείο, με την προσθήκη του GPS και της AR τεχνολογίας, μετατρέπεται σε ένα ζωντανό, διαδραστικό περιβάλλον. Ο χρήστης γίνεται μέρος της εμπειρίας, καθώς περπατάει και βλέπει το περιεχόμενο να ξεδιπλώνεται γύρω του.

Αναγνωρίζω βέβαια ότι υπάρχουν και περιορισμοί, όπως σε περιοχές με κακή δορυφορική κάλυψη ή σε εσωτερικούς χώρους, όπου το GPS χάνει την ακρίβεια του. Παρ' όλα αυτά, για τη χρήση που προοριζόταν η εφαρμογή, δηλαδή σε εξωτερικά, αναγνωρίσιμα σημεία της πόλης όπως ο Λευκός Πύργος και η Πλατεία Αριστοτέλους, η λειτουργία ήταν σταθερή, ακριβής και λειτουργική.

Συμπερασματικά, η χρήση του GPS προσέθεσε βάθος και ρεαλισμό στην εμπειρία του χρήστη. Χάρη σε αυτό, το σύστημα μπορούσε να "καταλαβαίνει" πού βρίσκεται ο χρήστης και να του εμφανίζει σχετικό περιεχόμενο με φυσικό τρόπο, χωρίς να χρειάζεται ο ίδιος να κάνει περιττές ενέργειες. Ήταν ένας κρίσιμος κρίκος που συνέδεσε την τεχνολογία της AR με την καθημερινή εμπειρία του φυσικού χώρου.

3.2.5 Προσθήκη Σημείων Ενδιαφέροντος (PointsofInterest –POIs)

Τα σημεία ενδιαφέροντος, γνωστά και ως POIs (Points of Interest), αποτελούν τον πυρήνα της εμπειρίας σε τουριστικές και πολιτιστικές εφαρμογές επαυξημένης πραγματικότητας. Είναι τα σημεία στον χάρτη που «κουβαλούν» πληροφορία: ιστορική, πολιτιστική, αρχιτεκτονική ή απλώς πρακτική. Χωρίς την ύπαρξή τους, μια εφαρμογή AR αυτού του τύπου δεν θα είχε λόγο ύπαρξης, αφού δεν θα υπήρχε περιεχόμενο να παρουσιαστεί.

Στο πλαίσιο της εφαρμογής μου, τα POIs λειτουργούν ως ψηφιακές στάσεις ξενάγησης. Είναι τοποθετημένα σε χαρακτηριστικά σημεία της Θεσσαλονίκης, όπως ο Λευκός Πύργος, η Πλατεία Αριστοτέλους ή η Καμάρα, και σχεδιάστηκαν ώστε να ενεργοποιούν πληροφοριακό περιεχόμενο όταν ο χρήστης πλησιάζει. Αυτό το περιεχόμενο μπορεί να είναι μια σύντομη περιγραφή, μια παλιά φωτογραφία ή ακόμα και ένα σχετικό βίντεο, όλα προσαρμοσμένα για να εμφανίζονται μέσα από την κάμερα της συσκευής μέσω επαυξημένης πραγματικότητας.

Για να υλοποιηθεί αυτό, πρώτα έγινε συλλογή και καταγραφή των POIs. Για κάθε σημείο, καταχωρήθηκαν οι ακριβείς γεωγραφικές συντεταγμένες (latitude και longitude), καθώς και το περιεχόμενο που θα το συνοδεύει (τίτλος, περιγραφή, εικόνα ή video). Οι πληροφορίες αυτές συνδέθηκαν με την εφαρμογή και αποθηκεύτηκαν ώστε να είναι διαθέσιμες offline, προκειμένου να λειτουργεί και χωρίς συνεχή σύνδεση στο διαδίκτυο.

Η ενεργοποίηση των σημείων βασίστηκε στον συνδυασμό GPS και AR. Όταν ο χρήστης πλησιάζει ένα POI, η εφαρμογή ελέγχει την απόσταση και αν βρίσκεται εντός 10 μέτρων, εμφανίζεται στην οθόνη ένα κουμπί AR στο κάτω μέρος της διεπαφής. Αν ο χρήστης το πατήσει, ενεργοποιείται η κάμερα και ζητείται από τον χρήστη να σκανάρει ένα imagetarget που έχουμε σχεδιάσει για το συγκεκριμένο σημείο. Μόλις αναγνωριστεί η εικόνα, προβάλλεται στην οθόνη το αντίστοιχο υλικό: είτε περιγραφή, είτε εικόνα, είτε βίντεο, ανάλογα με την επιλογή του.

Αυτή η διαδικασία σχεδιάστηκε για να είναι όσο το δυνατόν πιο φυσική και απλή. Ο χρήστης δεν χρειάζεται να αναζητήσει ή να πλοηγηθεί περίπλοκα μέσα σε μενού. Αρκεί να βρίσκεται στο σωστό σημείο, να στρέψει το κινητό του και να «ξεκλειδώσει» την πληροφορία που είναι συνδεδεμένη με το περιβάλλον του. Έτσι, το κάθε POI δεν είναι απλώς ένα κουμπί ή μια καταχώρηση, αλλά μια διαδραστική εμπειρία που «αναδύεται» τη στιγμή που πρέπει.

Η δημιουργία αυτών των σημείων είχε και τεχνικές απαιτήσεις. Έπρεπε να οριστεί πώς θα αντιστοιχίζονται οι συντεταγμένες στον τρισδιάστατο χάρτη, πώς θα εντοπίζονται τα imagetargets και πώς θα διαχειρίζεται το UI ανάλογα με την απόσταση του χρήστη. Μέσα από τη Unity, κατάφερα να συνδέσω όλα τα παραπάνω, φτιάχνοντας ένα σύστημα που ελέγχει σε κάθε frame τη θέση του χρήστη και «ξυπνάει» το ανάλογο POI όταν χρειάζεται.

Ο τελικός στόχος ήταν να δημιουργηθεί μια ολοκληρωμένη και καθηλωτική εμπειρία περιήγησης. Τα POIs δεν είναι απλώς στατικά σημεία στον χάρτη, αλλά διαδραστικά σημεία αναφοράς που εμπλουτίζουν την πραγματικότητα. Μέσω αυτών, ο χρήστης γνωρίζει καλύτερα τον χώρο, μαθαίνει ιστορία και πληροφορίες με τρόπο άμεσο και όχι βαρετό, και νιώθει ότι συμμετέχει ενεργά σε μια ξενάγηση φτιαγμένη για αυτόν.

Σε επόμενα βήματα, τα POIs μπορούν να εμπλουτιστούν ακόμα περισσότερο: με δυνατότητα βαθμολογίας, προσθήκη ήχου, υποστήριξη πολλών γλωσσών ή δυνατότητα καθοδήγησης από σημείο σε σημείο. Όμως ακόμα και στη βασική τους μορφή, αποδείχθηκαν το πιο ουσιαστικό στοιχείο της εφαρμογής, αφού είναι αυτοί οι «κόμβοι» που δίνουν πραγματικό νόημα στην επαυξημένη πραγματικότητα μέσα στον αστικό ιστό.

3.3Επίλογος

Η επιλογή των τεχνολογιών που χρησιμοποιήθηκαν στην παρούσα εφαρμογή δεν ήταν τυχαία, αλλά αποτέλεσμα διερεύνησης των αναγκών του έργου και της επιθυμίας για δημιουργία μιας σύγχρονης, ολοκληρωμένης και επεκτάσιμης εμπειρίας για τον χρήστη. Κάθε μία από τις πλατφόρμες που αξιοποιήθηκαν, καλύπτει έναν κρίσιμο άξονα της εφαρμογής, τόσο σε τεχνικό όσο και σε λειτουργικό επίπεδο.

Η Unity αποτέλεσε τον πυρήνα ανάπτυξης της εφαρμογής, προσφέροντας ένα ευέλικτο και ισχυρό περιβάλλον για τη δημιουργία διαδραστικού περιεχομένου. Η ενσωμάτωση εργαλείων όπως η Vuforia για την επαυξημένη πραγματικότητα, κατέστησε εφικτή την άμεση σύνδεση του ψηφιακού περιεχομένου με τον πραγματικό κόσμο, μέσω της αναγνώρισης εικόνων στο φυσικό περιβάλλον. Η επιλογή του Vuforia έγινε κυρίως λόγω της σταθερότητας, της καλής τεκμηρίωσης και της συμβατότητάς του με κινητές συσκευές.

Ο Cesium και οι διαδραστικοί χάρτες έπαιξαν καταλυτικό ρόλο στην γεωγραφική απεικόνιση του χρήστη και των σημείων ενδιαφέροντος. Η ενσωμάτωση τρισδιάστατων χαρτών ενίσχυσε την αίσθηση προσανατολισμού και έδωσε στη χρήση μια αίσθηση πραγματικού περιβάλλοντος. Η σύνδεση με το GPS κατέστησε εφικτή τη δυναμική παρακολούθηση της θέσης του χρήστη και την ενεργοποίηση περιεχομένου με βάση την πραγματική του τοποθεσία.

Τα σημεία ενδιαφέροντος ήταν η καρδιά της εμπειρίας. Η έξυπνη αξιοποίησή τους σε συνδυασμό με το AR πρόσφερε στον χρήστη μια ελκυστική μορφή πληροφόρησης, η οποία δεν περιοριζόταν σε απλό κείμενο, αλλά εμπλουτίστηκε με εικόνες, βίντεο και περιγραφή, δημιουργώντας ένα πολυδιάστατο περιβάλλον αλληλεπίδρασης.

Η συνέργεια όλων αυτών των τεχνολογιών είχε ως αποτέλεσμα την υλοποίηση μιας εφαρμογής που γεφυρώνει το χάσμα μεταξύ ψηφιακής και φυσικής πραγματικότητας. Η εφαρμογή δεν λειτουργεί απλώς ως ένας χάρτης ή ένας οδηγός, αλλά ως ένας ψηφιακός συνοδός περιήγησης, που αξιοποιεί σύγχρονα μέσα για να μεταφέρει πληροφορία με τρόπο κατανοητό, εντυπωσιακό και λειτουργικό.

Επιπλέον, η επιλογή τεχνολογιών που έχουν υποστήριξη, συνεχή ανάπτυξη και μεγάλη κοινότητα χρηστών (όπως το Unity ή το Cesium) διασφαλίζει τη δυνατότητα επέκτασης και βελτίωσης της εφαρμογής στο μέλλον, καθιστώντας την μακροπρόθεσμα βιώσιμη.

Συνολικά, η τεχνολογική κατεύθυνση της εργασίας αυτής δεν περιορίστηκε στην απλή υλοποίηση μιας εφαρμογής, αλλά απέδειξε πως η κατάλληλη σύνθεση εργαλείων και τεχνολογιών μπορεί να δημιουργήσει καινοτόμες λύσεις με ουσιαστική προστιθέμενη αξία για τον χρήστη.

Κεφάλαιο4ο:Κύκλος Ζωής της Εφαρμογής

4.1 Εισαγωγή

Η κατανόηση του τρόπου με τον οποίο λειτουργεί η εφαρμογή από τη στιγμή που την ανοίγει ο χρήστης μέχρι την ολοκλήρωση της εμπειρίας AR, είναι βασικό στοιχείο για την πλήρη αποτύπωση της λειτουργικότητάς της. Ο κύκλος ζωής της εφαρμογής περιλαμβάνει όλα τα στάδια χρήσης, από την εκκίνηση και τον έλεγχο δικαιωμάτων, μέχρι την πλοήγηση στους χάρτες και την ενεργοποίηση του περιεχομένου επαυξημένης πραγματικότητας.

Η δομή είναι απλή, αλλά μελετημένη, έτσι ώστε ο χρήστης να μην χρειάζεται τεχνικές γνώσεις για να τη χρησιμοποιήσει. Στόχος ήταν να υπάρχει άμεση κατανόηση και φυσική ροή, χωρίς περίπλοκα μενού ή περιττές πληροφορίες.

4.2 Εκκίνηση της εφαρμογής και έλεγχος αδειών

Με το άνοιγμα της εφαρμογής, πραγματοποιείται ένας αρχικός έλεγχος για τα βασικά δικαιώματα λειτουργίας:

- Πρόσβαση στην τρέχουσα τοποθεσία μέσω GPS
- Πρόσβαση στην κάμερα της συσκευής για τη χρήση AR

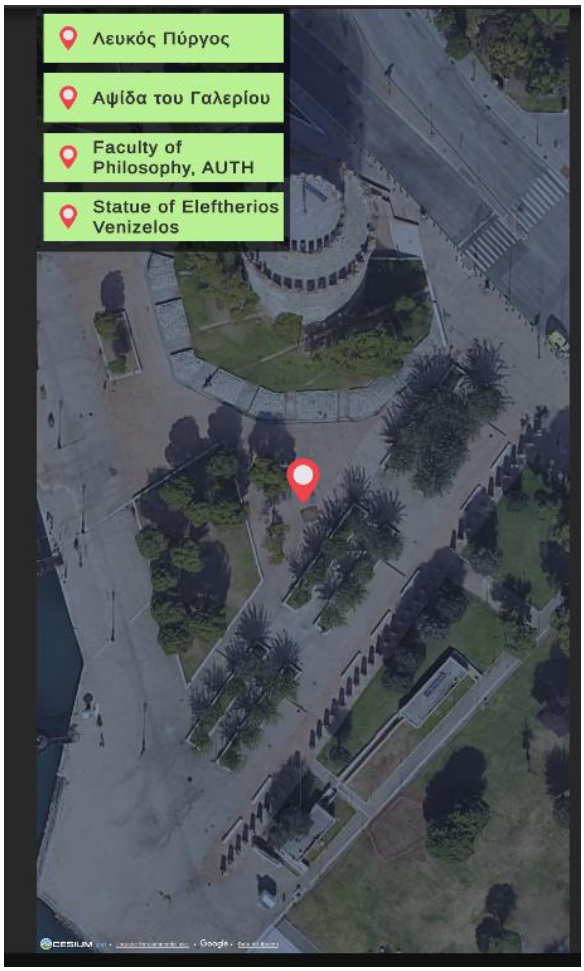
Αν δεν έχουν ήδη δοθεί, η εφαρμογή ζητά την έγκριση του χρήστη με ευγενικά και επεξηγηματικά μηνύματα. Αυτή η φάση είναι απαραίτητη για να διασφαλιστεί η σωστή λειτουργία της εφαρμογής και η προστασία των προσωπικών δεδομένων.

4.3 Προβολή χάρτη και δυναμική πλοήγηση

Αφού γίνει η αποδοχή, εμφανίζεται η κύρια οθόνη της εφαρμογής, που περιλαμβάνει ψηφιακό χάρτη της περιοχής και δείκτη (πινέζα) που αντιστοιχεί στην τρέχουσα θέση του χρήστη. Αυτός ο δείκτης μετακινείται σε πραγματικό χρόνο, καθώς ο χρήστης κινείται.

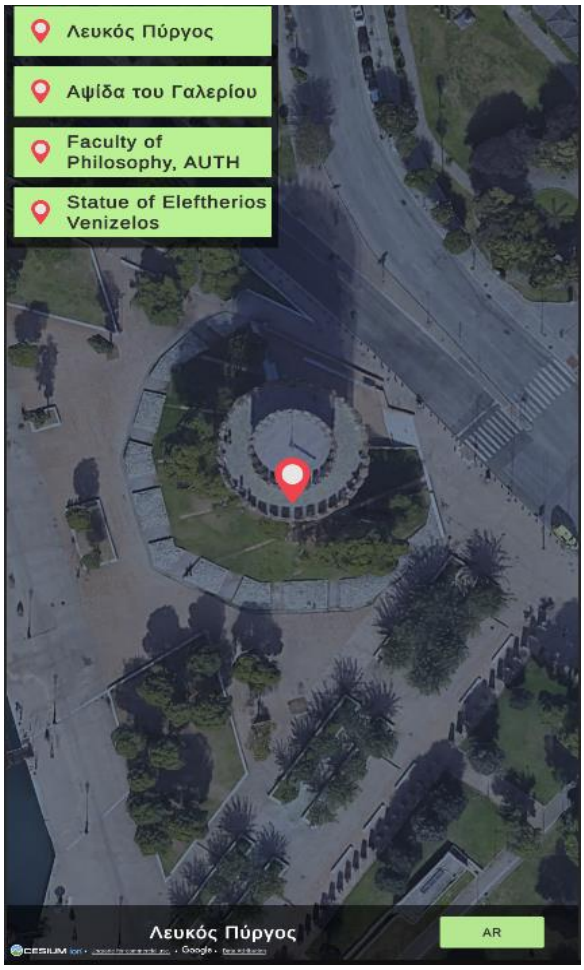
Στην πάνω αριστερή πλευρά της οθόνης εμφανίζεται μια λίστα με όλα τα διαθέσιμα σημεία ενδιαφέροντος (POIs), στα οποία μπορεί να κατευθυνθεί:

- Λευκός Πύργος
- Αψίδα του Γαλερίου (Καμάρα)
- Πλατεία Αριστοτέλους (imagetarget: Άγαλμα Ελευθερίου Βενιζέλου)
- Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης (imagetarget: Φιλοσοφική Σχολή)



Εικόνα 4.1-Αρχική οθόνη με χάρτη, θέση χρήστη και λίστα POIs

Ο χρήστης μπορεί να περπατήσει προς οποιοδήποτε από αυτά τα σημεία. Η εφαρμογή παρακολουθεί τη θέση του σε πραγματικό χρόνο και μόλις πλησιάσει ένα σημείο σε απόσταση μικρότερη των 10 μέτρων, τότε εμφανίζεται στο κάτω μέρος της οθόνης μια κάρτα με τον τίτλο του σημείου και ένα πράσινο κουμπί “AR”.



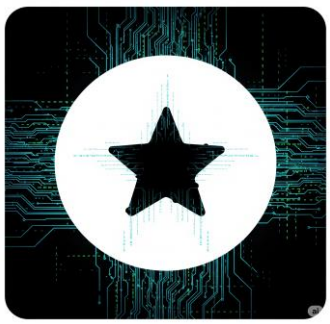
Εικόνα 4.2- Εμφάνιση κουμπιού "AR" όταν ο χρήστης βρίσκεται εντός 10 μέτρων

4.4 Προβολή χάρτη και δυναμική πλοήγηση

Όταν ο χρήστης πατήσει το κουμπί "AR", ενεργοποιείται η κάμερα της συσκευής του, δίνοντάς του τη δυνατότητα να χρησιμοποιήσει τη λειτουργία επαυξημένης πραγματικότητας. Η εφαρμογή περιμένει από τον χρήστη να στοχεύσει ένα συγκεκριμένο imagetarget, δηλαδή μια σταθερή εικόνα που έχει οριστεί από την αρχή και ενεργοποιεί την εμφάνιση του ψηφιακού περιεχομένου.

Αξίζει να σημειωθεί ότι σε όλα τα σημεία ενδιαφέροντος χρησιμοποιείται το ίδιο imagetarget, και αυτό που διαφέρει είναι η τοποθεσία στην οποία ο χρήστης πρέπει να βρίσκεται και να το εντοπίσει:

- Στον Λευκό Πύργο, ο χρήστης καλείται να στοχεύσει την προκαθορισμένη εικόνα, η οποία έχει τοποθετηθεί κοντά στο μνημείο.
- Στην Πλατεία Αριστοτέλους, η ίδια εικόνα βρίσκεται κοντά ή πάνω στο άγαλμα του Ελευθερίου Βενιζέλου.
- Στο Αριστοτέλειο Πανεπιστήμιο, η εικόνα εντοπίζεται στον χώρο της Φιλοσοφικής Σχολής.

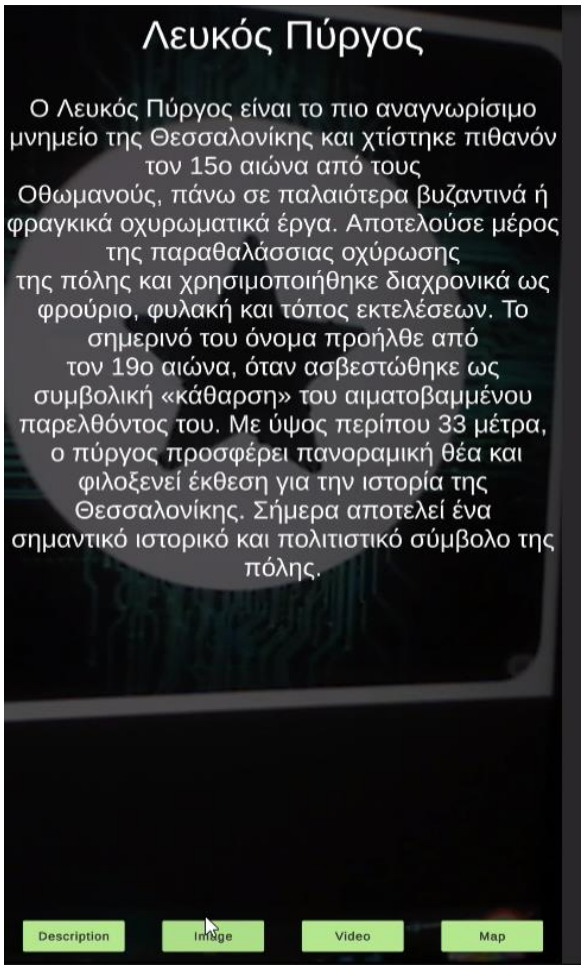


Εικόνα 4.3- Το σταθερό image target που χρησιμοποιείται για την ενεργοποίηση του AR περιεχομένου

Μόλις η εφαρμογή αναγνωρίσει επιτυχώς την εικόνα, τότε εμφανίζεται στην οθόνη το σχετικό περιεχόμενο με το σημείο που έχει επιλεγεί. Ο χρήστης έχει στη διάθεσή του τρεις επιλογές μέσω κουμπιών, για να δει:

- Μια εικόνα σχετική με το σημείο,
- Εναβίντεο μικρής διάρκειας,
- Μια περιγραφή με κείμενο.

Η μετάβαση από τη μία επιλογή στην άλλη γίνεται απλά, πατώντας τα αντίστοιχα κουμπιά που βρίσκονται στο κάτω μέρος της οθόνης. Με αυτό τον τρόπο, ο χρήστης μπορεί να εξερευνήσει το περιεχόμενο με τον ρυθμό και τον τρόπο που προτιμά, επιλέγοντας τι θέλει να μάθει πρώτα ή αν επιθυμεί να επιστρέψει στους χάρτες.



Εικόνα 4.4- Παρουσίαση περιεχομένου επαυξημένης πραγματικότητας με επιλογές εικόνας, βίντεο και περιγραφής

4.5 Επιστροφή στον χάρτη και στην πλοήγηση

Μετά την ολοκλήρωση της εμπειρίας AR, ο χρήστης έχει τη δυνατότητα να επιστρέψει στην προβολή του χάρτη. Αυτό γίνεται με ένα απλό πάτημα στο κουμπί “Map”, το οποίο βρίσκεται πάντα διαθέσιμο στο κάτω δεξί μέρος της οθόνης, ακόμα και κατά την προβολή του AR περιεχομένου. Με αυτόν τον τρόπο, η πλοήγηση στην εφαρμογή γίνεται πιο ευέλικτη, δίνοντας τη δυνατότητα στον χρήστη να αλλάξει σημείο, να μεταβεί σε νέο POI ή να περιηγηθεί ξανά σε ό,τι τον ενδιαφέρει.

Η λειτουργία αυτή είναι σημαντική, γιατί δεν "κλειδώνει" τον χρήστη μέσα στην AR εμπειρία, αλλά του επιτρέπει να επιλέγει τον ρυθμό και τη διαδρομή του μέσα στην πόλη. Η εμπειρία παραμένει καθοδηγούμενη αλλά όχι περιοριστική.

Εικόνα 4.5- Κουμπί επιστροφής στον χάρτη κατά την προβολή επαυξημένης πραγματικότητας

4.6 Τερματισμός της εφαρμογής

Η έξοδος από την εφαρμογή γίνεται με κλασικό τρόπο, όπως σε κάθε εφαρμογή Android, είτε μέσω του κουμπιού επιστροφής της συσκευής, είτε με swipeup από το taskmanager. Ωστόσο, κατά τον τερματισμό, η εφαρμογή διακόπτει άμεσα την πρόσβαση στο GPS και την κάμερα για λόγους ιδιωτικότητας και διαχείρισης ενέργειας.

Αυτό εξασφαλίζει ότι δεν παραμένουν ενεργές λειτουργίες στο παρασκήνιο, προσφέροντας μια ομαλή έξοδο από την εμπειρία χωρίς να επιβαρύνεται η συσκευή ή να παραβιάζονται προσωπικά δεδομένα. Επίσης, σε κάθε επανεκκίνηση, ο χρήστης καλείται εκ νέου να αποδεχθεί τα απαραίτητα δικαιώματα, διασφαλίζοντας έτσι τη διαφάνεια και τον έλεγχο.

4.7 Επίλογος

Ο κύκλος ζωής της εφαρμογής σχεδιάστηκε έτσι ώστε να υποστηρίζει μια εμπειρία απλή, φιλική και λειτουργική. Από την είσοδο στον χάρτη και την επιλογή σημείου, μέχρι την ενεργοποίηση της επαυξημένης πραγματικότητας και την επιστροφή στην πλοήγηση, κάθε βήμα έχει σκεφτεί με γνώμονα την αμεσότητα και την ευχρηστία.

Η τεχνολογία λειτουργεί στο παρασκήνιο με τέτοιο τρόπο ώστε να μην εμποδίζει αλλά να υποστηρίζει την περιήγηση. Στόχος δεν ήταν να εντυπωσιάσει μόνο οπτικά, αλλά να προσφέρει πραγματική χρησιμότητα σε φοιτητές, επισκέπτες και όσους επιθυμούν να εξερευνήσουν την πόλη με έναν νέο, πιο διαδραστικό τρόπο.

Η εμπειρία ολοκληρώνεται φυσικά, αφήνοντας στον χρήστη την αίσθηση ότι "έμαθε" την πόλη με τον δικό του ρυθμό, όχι ότι τον "καθοδήγησε" απλώς μια εφαρμογή.

Κεφάλαιο 5ο: Τεκμηρίωση Κώδικα

5.1 Εισαγωγή στο Doxygen

Κατά την ανάπτυξη ενός έργου που βασίζεται σε πολλά scripts και αλληλεπιδράσεις μεταξύ διαφορετικών στοιχείων, είναι πολύ εύκολο να χαθεί η συνολική εικόνα της αρχιτεκτονικής ή να μην είναι κατανοητό τι κάνει κάθε κομμάτι του κώδικα. Για αυτόν τον λόγο, η τεκμηρίωση παίζει κομβικό ρόλο όχι μόνο για την καλύτερη συντήρηση του έργου στο μέλλον, αλλά και για την ίδια την κατανόηση του συστήματος από τρίτους (ή και από τον ίδιο τον προγραμματιστή, σε μεταγενέστερη φάση).

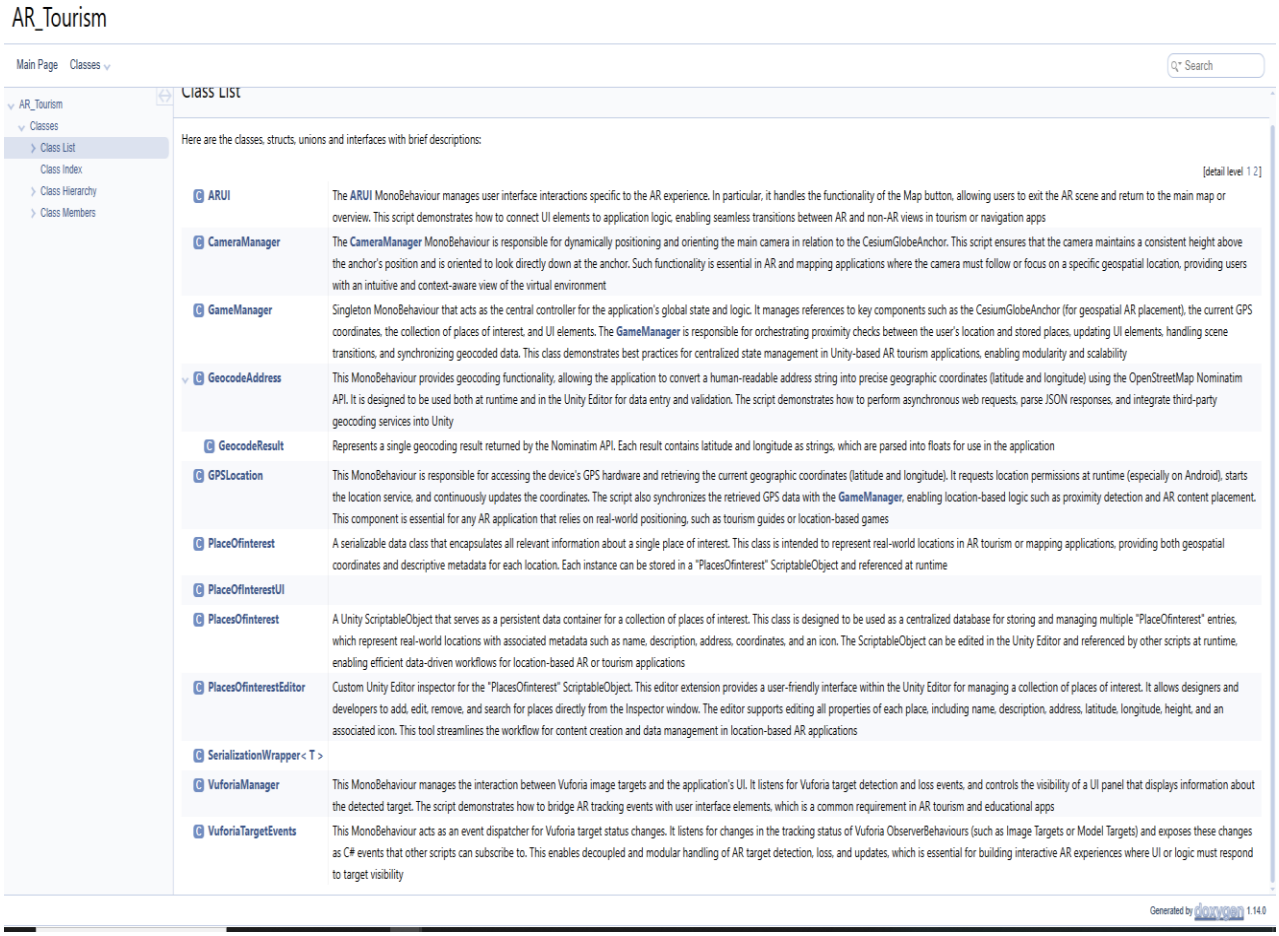
Για τη συγκεκριμένη εφαρμογή, χρησιμοποιήθηκε το Doxygen, ένα πολύ ισχυρό εργαλείο αυτόματης παραγωγής τεκμηρίωσης από σχόλια στον πηγαίο κώδικα. Μέσω των κατάλληλων annotations και της αξιοποίησης της γλώσσας σήμανσης του Doxygen, είναι δυνατόν να δημιουργηθεί μια πλήρως δομημένη τεκμηρίωση σε μορφή HTML, η οποία περιλαμβάνει:

- Περιγραφή κάθε κλάσης και μελών της
- Διαγράμματα συσχετίσεων και ιεραρχίας
- Αναφορές χρήσης και σχέσεων μεταξύ των scripts
- Συνεχώς ενημερωμένη τεκμηρίωση καθώς εξελίσσεται ο κώδικας

Η διαδικασία ξεκίνησε με τον σχολιασμό των σημαντικών τμημάτων του project χρησιμοποιώντας ειδική σύνταξη (όπως ///, @brief, @param, κ.λπ.). Στη συνέχεια, μέσω της εφαρμογής του Doxygen και με τη βοήθεια του εργαλείου Graphviz, δημιουργήθηκαν αυτόματα αρχεία HTML που παρουσιάζουν όλες τις απαραίτητες πληροφορίες για τον κώδικα με ευανάγνωστο τρόπο.

Η κεντρική σελίδα της τεκμηρίωσης (index.html) περιλαμβάνει λίστα με όλες τις κλάσεις, ιεραρχίες, μεθόδους και διαγράμματα συνεργασίας μεταξύ των scripts. Η πλοήγηση είναι απλή και άμεση, προσφέροντας στον χρήστη μια πλήρη εικόνα του έργου χωρίς να χρειάζεται να ανατρέχει στον κώδικα γραμμή-γραμμή.

Παρακάτω παρατίθεται ένα ενδεικτικό στιγμιότυπο από την αρχική σελίδα της τεκμηρίωσης:



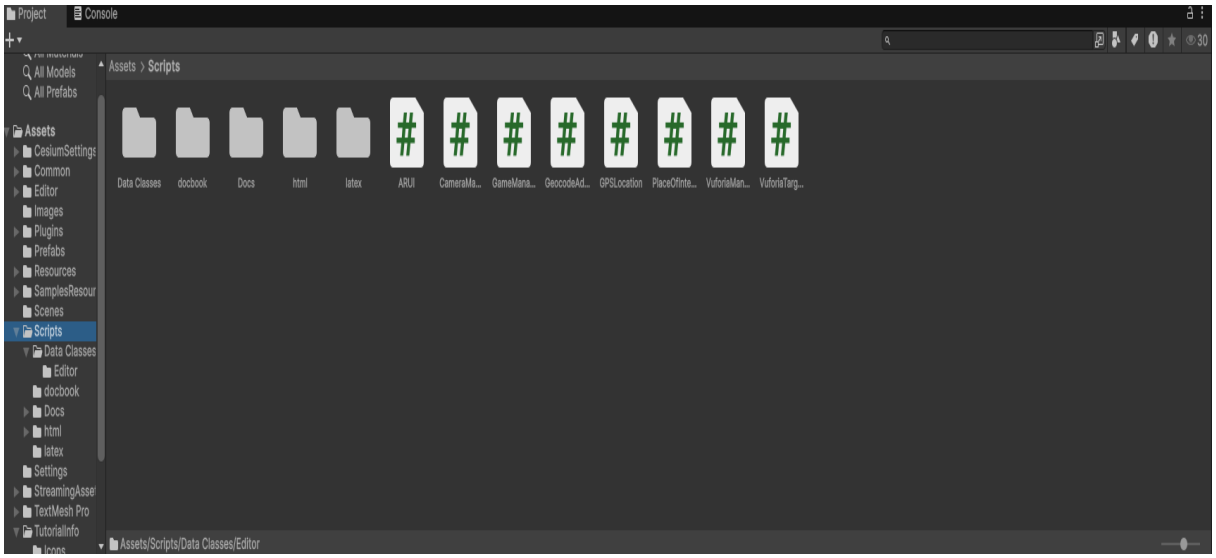
Εικόνα 5.1- Παρουσιάζει την αρχική σελίδα της τεκμηρίωσης, όπου παρατίθενται οι κύριες κλάσεις του project μαζί με συνοπτικές περιγραφές των ρόλων τους

5.2 Οργάνωση του Project

Ένα από τα πρώτα και πιο σημαντικά βήματα στην ανάπτυξη της εφαρμογής ήταν να οργανώσω σωστά τα αρχεία και τα scripts στο Unity, ώστε το project να παραμείνει διαχειρίσιμο όσο μεγάλωνε. Σε αυτή την ενότητα παρουσιάζω πώς είναι δομημένο το project, ποια είναι τα βασικά του components και πώς όλα αυτά συνεργάζονται.

5.2.1 Δομή Φακέλων

Η δομή που ακολουθήθηκε δεν είναι πολύπλοκη, αλλά είναι αρκετά πρακτική. Όλα τα scripts της εφαρμογής τοποθετήθηκαν συγκεντρωμένα μέσα στον φάκελο Assets/Scripts, χωρίς διαχωρισμό σε υποφακέλους ανά λειτουργία (όπως AR, GPS, UI κ.λπ.).
Αυτό έγινε εσκεμμένα, ειδικά στα πρώτα στάδια ανάπτυξης, για να μπορώ να βλέπω και να επεξεργάζομαι πιο εύκολα όλα τα scripts, χωρίς να μπαίνω σε υποφακέλους και να χάνω χρόνο.



Εικόνα 5.2- Screenshot τουφακέλου Assets/Scripts στο Unity Editor

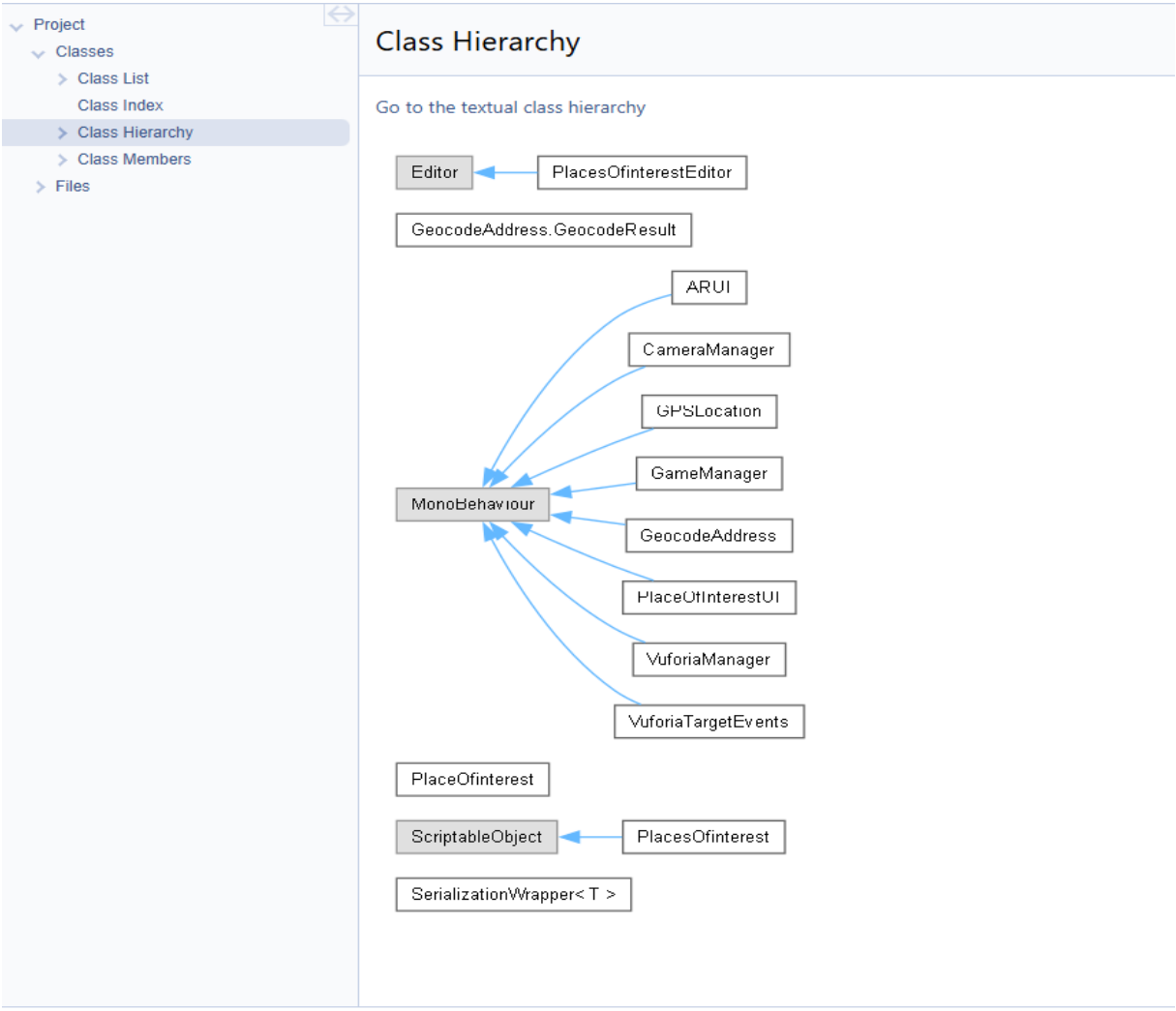
Όπως φαίνεται, ο φάκελος περιέχει όλα τα βασικά αρχεία της εφαρμογής:

- GameManager.cs: Το «κέντρο ελέγχου» της εφαρμογής. Συντονίζει τα πάντα: GPS, σημεία ενδιαφέροντος, αλλαγές σκηνών, καταστάσεις.
- GPSLocation.cs: Εντοπίζει και ενημερώνει συνεχώς τη θέση του χρήστη, με χρήση του GPS της συσκευής.
- CameraManager.cs: Ρυθμίζει τη θέση και την κατεύθυνση της κάμερας με βάση τις συντεταγμένες του χρήστη (χρήση Cesium).
- ARUI.cs: Διαχειρίζεται το περιβάλλον χρήστη όταν ενεργοποιείται το AR περιεχόμενο.
- VuforiaManager.cs&VuforiaTargetEvents.cs: Χειρίζονται την τεχνολογία Vuforia και την εμφάνιση των imagetargets.
- GeocodeAddress.cs: Αν μεταφράζεις μια διεύθυνση (π.χ. «Λευκός Πύργος») σε γεωγραφικές συντεταγμένες, αυτό το script το κάνει με χρήση OpenStreetMap.
- PlaceOfinterest.cs&PlacesOfinterest.cs: Το πρώτο είναι ένα μοντέλο για ένα POI, ενώ το δεύτερο είναι ένα ScriptableObject που περιέχει όλα τα POIs της εφαρμογής – σαν μικρή «βάση δεδομένων» σε μορφή asset.

Η απλότητα αυτής της δομής μου έδωσε ευελιξία και ταχύτητα, ειδικά στα πρώτα στάδια που έπρεπε συνεχώς να προσθέτω, να δοκιμάζω και να αλλάζω πράγματα.

5.2.2 Ιεραρχία Κλάσεων (από το Doxygen)

Για να κατανοήσουμε καλύτερα τη σχέση των κλάσεων μεταξύ τους, χρησιμοποίησα το εργαλείο Doxygen, το οποίο δημιουργεί αυτόματα ένα διάγραμμα κληρονομικότητας και εξαρτήσεων. Έτσι, είναι εύκολο να δεις τι επεκτείνει τι, και ποια scripts εξαρτώνται από άλλα.

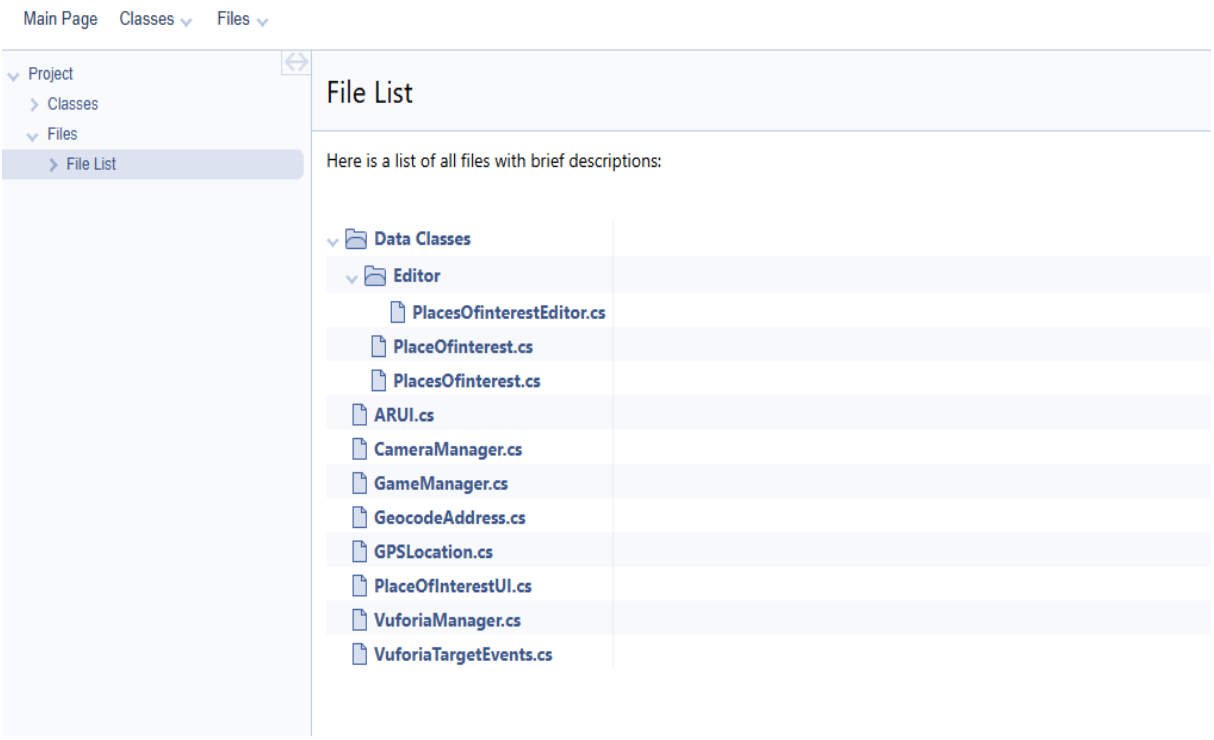


Εικόνα 5.3- Screenshot από το διάγραμμα ιεραρχίας κλάσεων του Doxygen

Στο διάγραμμα φαίνεται πως σχεδόν όλες οι βασικές κλάσεις (GameManager, ARUI, GPSLocation, CameraManager, VuforiaManager κ.λπ.) κληρονομούν από το MonoBehaviour, όπως είναι αναμενόμενο στο Unity. Από την άλλη, το PlaceOfinterest είναι μια απλή Serializable κλάση που περιγράφει ένα POI, ενώ το PlacesOfinterest είναι ScriptableObject και λειτουργεί ως repository δεδομένων.

5.2.3 Ανάγνωση και κατανόηση της δομής του κώδικα

Μια ακόμα πολύ χρήσιμη δυνατότητα του Doxygen είναι η προβολή FileList. Μέσα από αυτό το interface βλέπεις όλα τα αρχεία του project συγκεντρωμένα, με τη δυνατότητα να κάνεις κλικ σε κάθε ένα για να δεις την τεκμηρίωσή του.



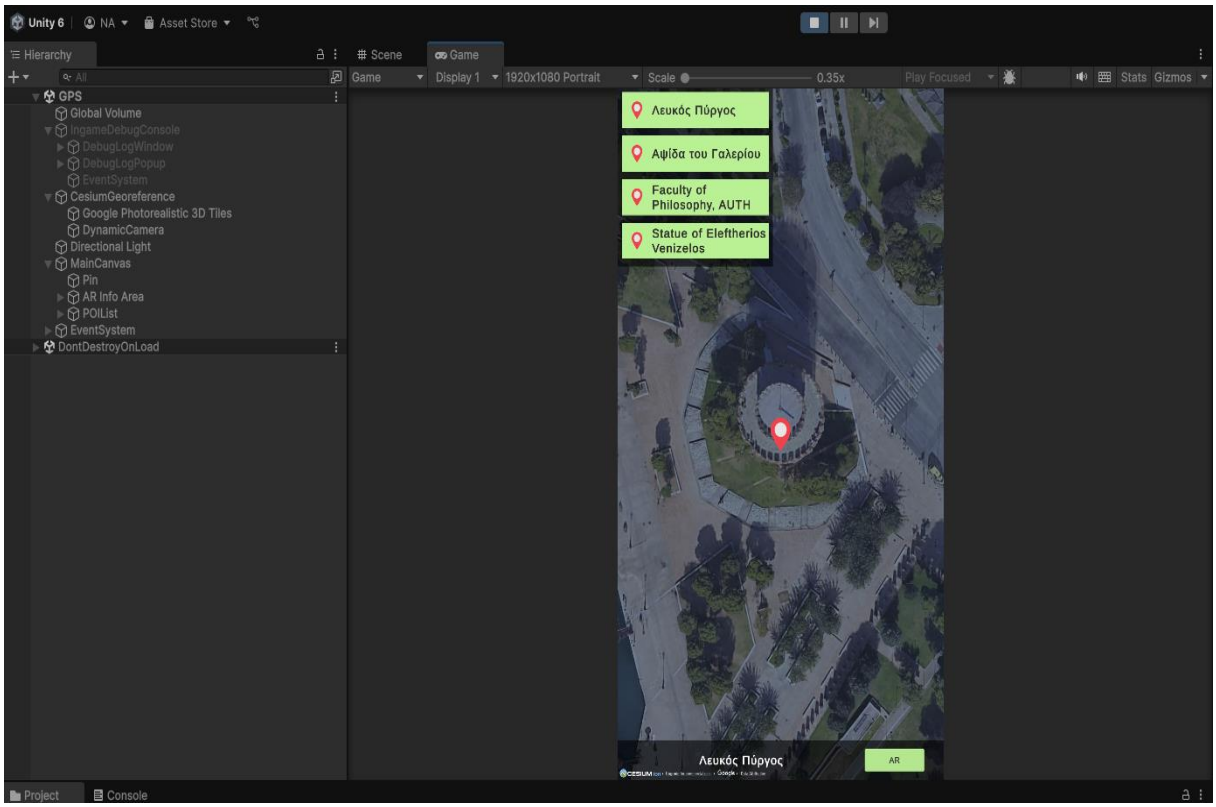
Εικόνα 5.4- Screenshot από το File List του Doxygen

Αυτό δεν είναι απλώς ένας φάκελος – είναι μια πολύτιμη πύλη αναφοράς. Ακόμα και αν έχεις να δεις τον κώδικά σου εβδομάδες ή εργάζεσαι με ομάδα, με ένα κλικ μπορείς να ξαναθυμηθείς τι κάνει το κάθε script, να εντοπίσεις γρήγορα ποιο αρχείο να επεξεργαστείς και να πλοηγηθείς απευθείας στις μεθόδους ή μεταβλητές του.

5.2.4 Ροή δεδομένων και συνεργασία των components

Για να καταλάβουμε καλύτερα πώς "κουμπώνουν" όλα μεταξύ τους, αξίζει να δούμε πώς ρέουν τα δεδομένα και ποιος ρόλος αντιστοιχεί σε κάθε component της εφαρμογής.

1. Εντοπισμός θέσης: Το GPSLocation.cs ξεκινά τη διαδικασία με την ανάκτηση των γεωγραφικών συντεταγμένων της συσκευής.
2. Διαχείριση λογικής: Ο GameManager.cs λαμβάνει τη θέση και ελέγχει αν ο χρήστης βρίσκεται κοντά σε κάποιο σημείο ενδιαφέροντος, μέσω της λίστας από PlacesOfinterest.cs.
3. Απεικόνιση & πλοήγηση: Αν πλησιάσουμε κάποιο POI, τότε:
    - Ενεργοποιείται το ARUI.cs για να εμφανίσει την κάρτα πληροφοριών
    - Το CameraManager.cs φροντίζει ώστε η κάμερα να είναι ευθυγραμμισμένη με το σημείο
    - Το VuforiaManager.cs ενεργοποιεί την AR αναγνώριση μέσω imagetargets



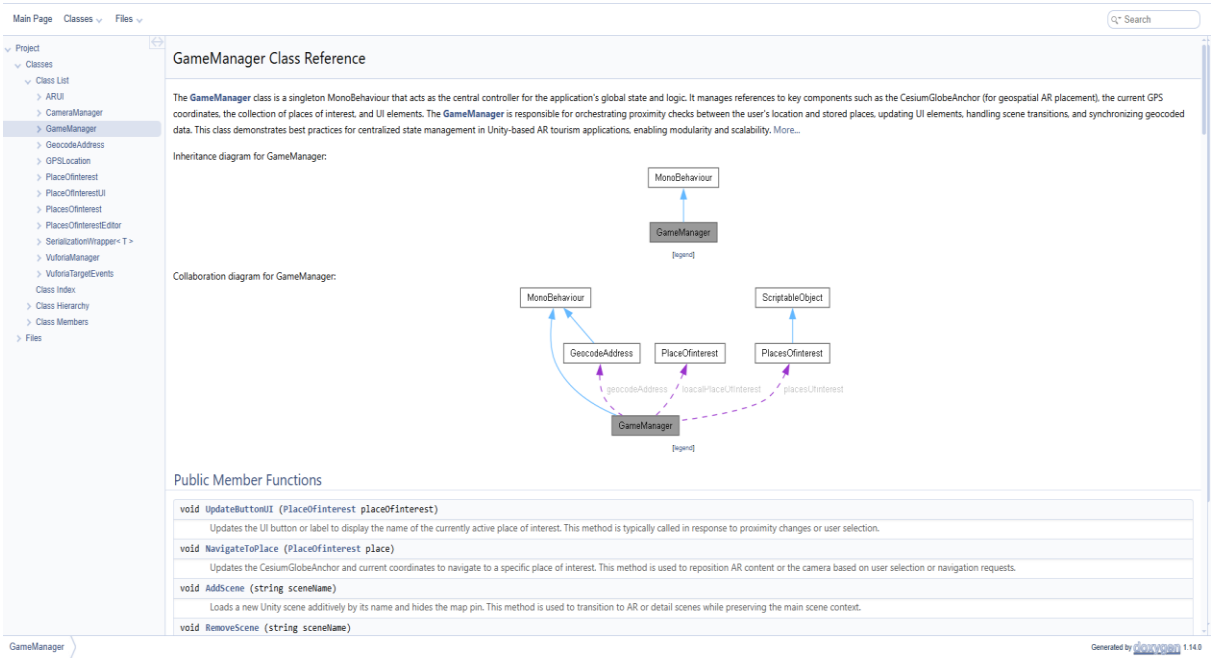
Εικόνα 5.5- Screenshot από τη βασική σκηνή Unity με συνδεδεμένα GameObjects (Canvas, ARCamera, GPS, κ.λπ.)

Με αυτόν τον τρόπο, δημιουργείται ένας κύκλος συνεχούς ελέγχου και απόκρισης: από την τοποθεσία, στην αναγνώριση, στην εμφάνιση περιεχομένου, προσφέροντας μια δυναμική εμπειρία επαυξημένης περιήγησης.

5.3 Επεξήγηση scripts

5.3.1 GameManager.cs

Ο GameManager είναι το μυαλό της εφαρμογής. Είναι ένα singletonscript που "τρέχει" από την αρχή ως το τέλος και φροντίζει για όλα τα βασικά: πού βρίσκεται ο χρήστης, ποια σημεία ενδιαφέροντος υπάρχουν γύρω του, ποιο εμφανίζεται στο UI, ποια σκηνή πρέπει να φορτωθεί και πότε.
Μόλις ξεκινήσει η εφαρμογή, ο GameManager ξεκινά δύο βασικές διεργασίες:
Μετατροπή διευθύνσεων σε συντεταγμένες (geocoding), ώστε τα σημεία ενδιαφέροντος να τοποθετηθούν σωστά στον χώρο.Έλεγχο εγγύτητας: κάθε δευτερόλεπτο, υπολογίζει αν ο χρήστης είναι κοντά σε κάποιο σημείο και, αν ναι, ενεργοποιεί το AR περιεχόμενο.Επιπλέον, διαχειρίζεται τη μετάβαση μεταξύ σκηνών, κρύβει/εμφανίζει panels και στέλνει ειδοποιήσεις (Events) σε άλλα scripts όταν αλλάζει το ενεργό POI.



Εικόνα 5.6- Doxygen αναφορά της κλάσης GameManager.cs με overview, διαγράμματα και public μεθόδους

Σχόλιο:Το collaboration diagram είναι ιδιαίτερα χρήσιμο για να καταλάβει κανείς με μια ματιά πώς ο GameManager αποτελεί το σημείο σύγκλισης μεταξύ GPS, UI, δεδομένων POI και γεωκωδικοποίησης – και δείχνει τη φιλοσοφία modular αρχιτεκτονικής που ακολουθήθηκε.

```
5       /// </summary>
6       void Awake()
7       {
8           // Ensure that there is only one instance of GameManager
9           if (Instance == null)
0           {
1               Instance = this;
2               DontDestroyOnLoad(gameObject); // Keep this instance across scenes
3           }
4           else if (Instance != this)
5           {
6               Destroy(gameObject); // Destroy duplicate instances
7               return;
8           }
9
0       }
```

Εικόνα 5.7- Απόσπασμα του κώδικα Awake() – εφαρμογή του μοτίβου Singleton

Σχόλιο:Αυτή η μέθοδος Awake() εφαρμόζει το μοτίβο Singleton. Εξασφαλίζει ότι ο GameManager παραμένει μοναδικός και σταθερός κατά την αλλαγή σκηνών, διατηρώντας όλα τα κρίσιμα δεδομένα, όπως την τοποθεσία του χρήστη και την κατάσταση του UI.

```
164     private IEnumerator ProximityCheckCoroutine()
165     {
166         while (true)
167         {
168             CheckProximityToPlaces();
169             yield return new WaitForSeconds(1f); // Check every 1 second
170         }
171     }
```

Εικόνα 5.0.8- Απόσπασμα από τη μέθοδο ProximityCheckCoroutine() και CheckProximityToPlaces()

Το ProximityCheckCoroutine() καλείται με την έναρξη της εφαρμογής και τρέχει συνεχώς κάθε 1 δευτερόλεπτο. Ελέγχει αν ο χρήστης βρίσκεται κοντά σε κάποιο αποθηκευμένο POI. Αν ναι, εμφανίζει το κατάλληλο AR περιεχόμενο.

```
178     private IEnumerator UpdateAllPlacesLatLonFromAddress()
179     {
180         if (geocodeAddress == null)
181         {
182             geocodeAddress = GetComponent<GeocodeAddress>();
183             if (geocodeAddress == null)
184             {
185                 Debug.LogError("No GeocodeAddress component found in the Game Manager.");
186                 yield break;
187             }
188         }
```

Εικόνα 5.9- Απόσπασμα από τη μέθοδο UpdateAllPlacesLatLonFromAddress()

Σχόλιο:Η μέθοδος αυτή σαρώνει όλα τα σημεία ενδιαφέροντος (POIs) και αν κάποιο δεν έχει συντεταγμένες, κάνει γεωκωδικοποίηση βάσει της διεύθυνσης με χρήση της GeocodeAddress κλάσης. Έτσι, όλα τα σημεία αποκτούν γεωγραφική θέση αυτόματα.

```
230     public void NavigateToPlace(PlaceOfinterest place)
231     {
232         // Logic to navigate to a specific place using the provided latitude and longitude
233         Debug.Log($"Navigating to Place at Latitude: {place.Latitude}, Longitude: {place.Longitude}");
234 
235         // Here you would typically update the CesiumGlobeAnchor or other navigation logic
236         currentLatitude = place.Latitude;
237         currentLongitude = place.Longitude;
238 
239         cesiumGlobeAnchor.latitude = place.Latitude;
240         cesiumGlobeAnchor.longitude = place.Longitude;
241         cesiumGlobeAnchor.height = place.Height; // Use the place's height
242     }
```

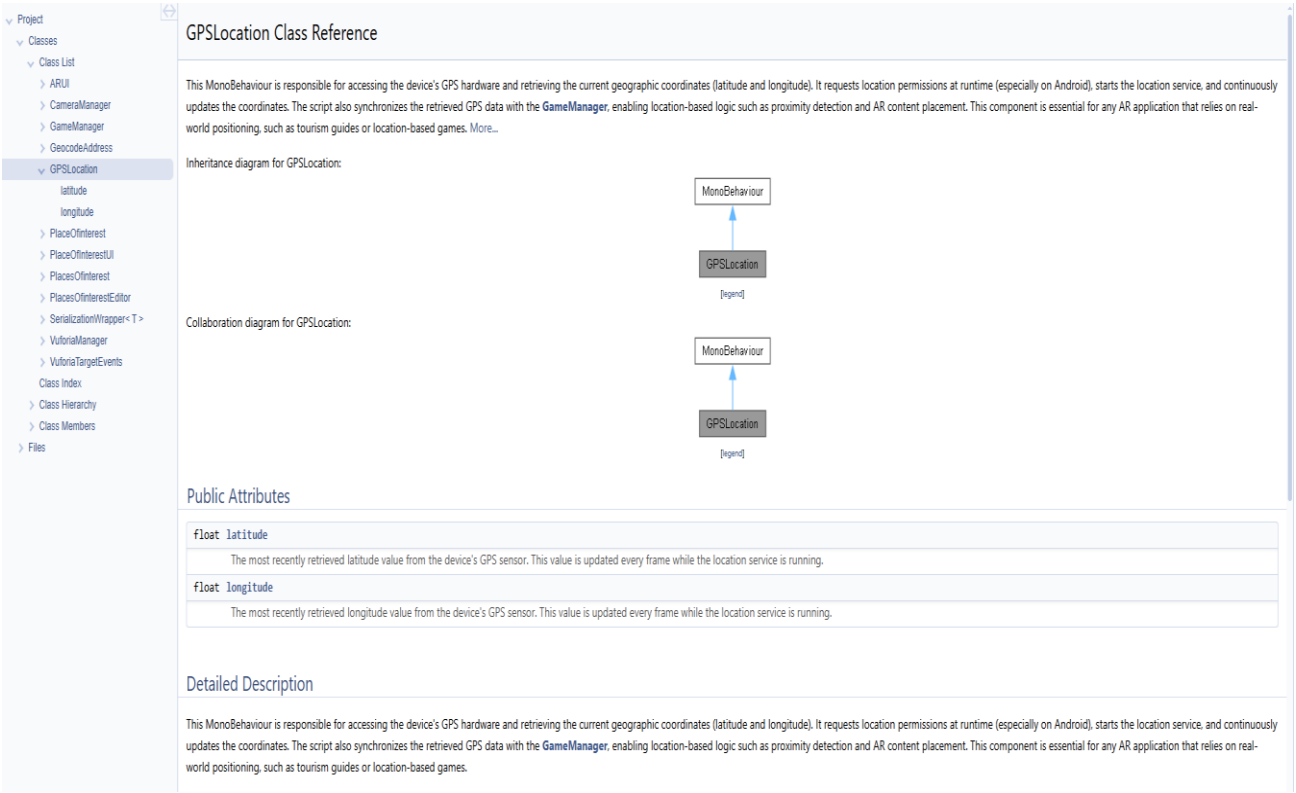
Εικόνα 5.10- Απόσπασμα από τη μέθοδο NavigateToPlace()

Σχόλιο:Ο GameManager διαχειρίζεται και την πλοήγηση σε σημεία, είτε μέσω επιλογής από τον χρήστη είτε μέσω εγγύτητας, ενώ παράλληλα φορτώνει νέες σκηνές για την εμπειρία AR. Με το NavigateToPlace() γίνεται επανευθυγράμμιση της κάμερας και του AR περιεχομένου πάνω στο POI που επιλέχθηκε, με χρήση Cesium.

5.3.2 GPSLocation.cs

Το GPSLocation.cs είναι το script που ενεργοποιεί και διαχειρίζεται τον εντοπισμό της τοποθεσίας του χρήστη μέσω του GPS της συσκευής. Είναι ουσιαστικά το σημείο επαφής ανάμεσα στον πραγματικό κόσμο και την εφαρμογή μας, αφού χωρίς αυτό δεν θα μπορούσαμε να γνωρίζουμε πού βρίσκεται ο χρήστης και αν πλησιάζει κάποιο POI.

Η λειτουργία του είναι απλή αλλά κομβική: ζητάει άδεια πρόσβασης στη θέση, ξεκινά την υπηρεσία GPS, και μετά – σε κάθε frame – παίρνει τις νέες συντεταγμένες και τις μεταβιβάζει στον GameManager. Με αυτό τον τρόπο, όλα τα υπόλοιπα components (AR εμφάνιση, χάρτης, imagetargets) έχουν επίκαιρα δεδομένα.



Εικόνα 5.11– Τεκμηρίωση της κλάσης GPSLocation.cs στο Doxygen

```
#if UNITY_ANDROID
        // Request location permission at runtime on Android
        if (!UnityEngine.Android.Permission.HasUserAuthorizedPermission(UnityEngine.Android.Permission.FineLocation))
        {
            UnityEngine.Android.Permission.RequestUserPermission(UnityEngine.Android.Permission.FineLocation);
        }
#endif
```

Εικόνα 5.12- Αίτημα για άδεια πρόσβασης τοποθεσίας (Android)

Σχόλιο: Αυτό το κομμάτι είναι υπεύθυνο για να ζητήσει από τον χρήστη την άδεια για χρήση του GPS απαραίτητο σε Android συσκευές. Είναι σημαντικό να τοποθετείται στο Start() ώστε να εκτελείται νωρίς.

```
63          // Start service before querying location
64          Input.location.Start();
65
66          // Wait until service initializes
67          int maxWait = 20;
68          while (Input.location.status == LocationServiceStatus.Initializing && maxWait > 0)
69          {
70              yield return new WaitForSeconds(1);
71              maxWait--;
72          }
73
```

Εικόνα 5.13- Coroutine εκκίνησης και αναμονής για ενεργοποίηση GPS

Σχόλιο: Εκκινεί την υπηρεσία εντοπισμού και περιμένει έως και 20 δευτερόλεπτα για να πάρει απάντηση. Αν το GPS δεν ανταποκριθεί, ορίζεται ότι απέτυχε. Αυτός ο έλεγχος προστατεύει την εφαρμογή από ατέρμονη αναμονή.

```
if (Input.location.status == LocationServiceStatus.Running)
{
    latitude = Input.location.lastData.latitude;
    longitude = Input.location.lastData.longitude;
    Debug.Log("Lat: " + latitude + " Lon: " + longitude);

    // Update GameManager's lat/lon if instance exists
    if (GameManager.Instance != null)
    {
        GameManager.Instance.currentLatitude = latitude;
        GameManager.Instance.currentLongitude = longitude;
    }
}
```

Εικόνα 5.14– Real-time συγχρονισμός συντεταγμένων με τον GameManager

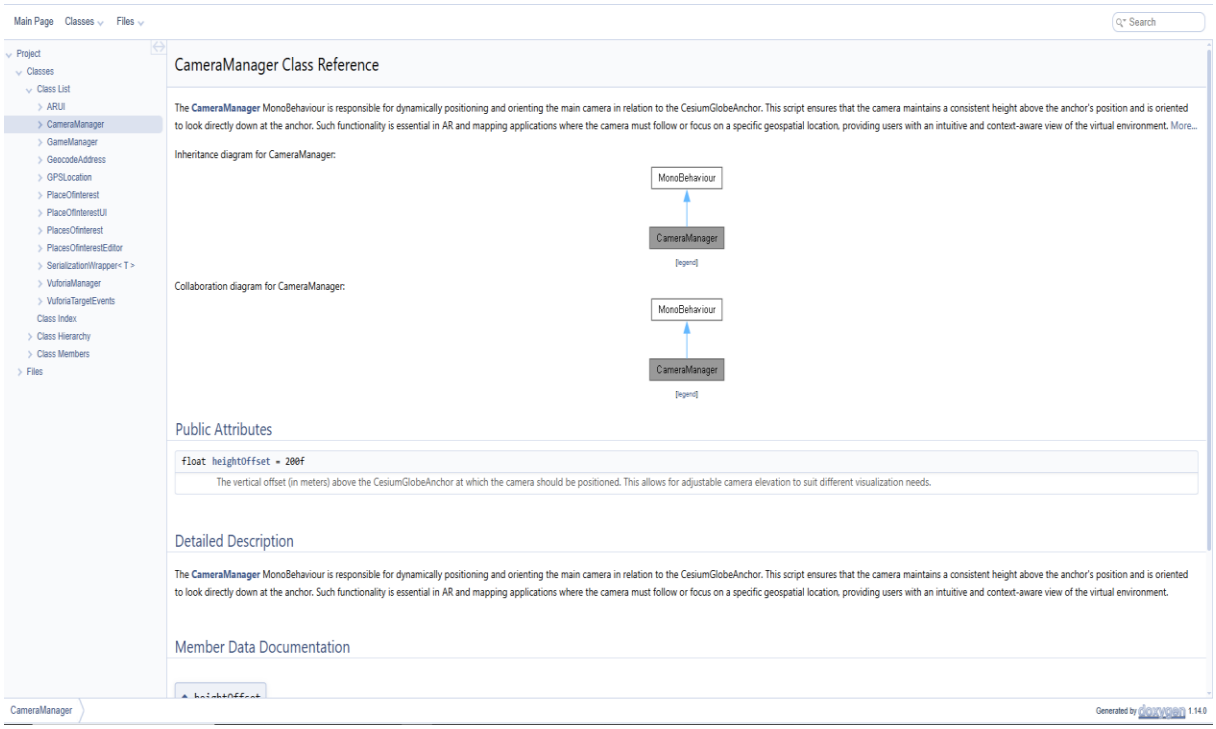
Σχόλιο:Αυτό το τμήμα του κώδικα είναι υπεύθυνο για τη συνεχή ενημέρωση των συντεταγμένων, και για τη συγχώνευσή τους με τον GameManager. Με αυτόν τον τρόπο, όλες οι υπόλοιπες λειτουργίες της εφαρμογής (π.χ. εγγύτητα σε POIs, εμφάνιση AR περιεχομένου) έχουν πάντα ενημερωμένα δεδομένα.

5.3.3 CameraManager.cs

Το CameraManager είναι υπεύθυνο για τη διαχείριση της θέσης και του προσανατολισμού της κύριας κάμερας της εφαρμογής. Η βασική του δουλειά είναι να τοποθετεί την κάμερα ακριβώς πάνω από το σημείο ενδιαφέροντος του χρήστη, σε μια συγκεκριμένη απόσταση ύψους, και να την στρέφει κάθετα προς τα κάτω.

Αυτό το είδος προβολής (top-down) είναι ιδανικό σε εφαρμογές επαυξημένης πραγματικότητας που βασίζονται σε χάρτη ή σε γεωχωρική πληροφορία, γιατί προσφέρει στον χρήστη μια σταθερή και κατανοητή οπτική της τοποθεσίας του μέσα στον χώρο.

Το script λειτουργεί σε άμεση συνεργασία με τον GameManager, καθώς βασίζεται σε αυτόν για να λάβει τις πιο πρόσφατες γεωγραφικές συντεταγμένες.



Εικόνα 5.15– DoxygenαναφοράτηςκλάσηςCameraManager.cs

```
// Set the camera position to the Cesium Globe Anchor's position
if (GameManager.Instance.currentLatitude != null && GameManager.Instance.currentLongitude != null)
{
    GameManager.Instance.cesiumGlobeAnchor.latitude = GameManager.Instance.currentLatitude;
    GameManager.Instance.cesiumGlobeAnchor.longitude = GameManager.Instance.currentLongitude;
    GameManager.Instance.cesiumGlobeAnchor.height = heightOffset;
}
```

Εικόνα 5.16– Απόσπασμα κώδικα: Ενημέρωση CesiumGlobeAnchor

Σχόλιο:Αυτό το σημείο είναι όπου η κάμερα "συγχρονίζεται" με την τοποθεσία του χρήστη. Το CesiumGlobeAnchor ενημερώνεται δυναμικά, ώστε η κάμερα να παρακολουθεί πάντα το σωστό γεωγραφικό σημείο.

```
        // Position the camera above the anchor
        transform.position = anchorTransform.position + Vector3.up * heightOffset;
        // Look directly down at the anchor
        transform.rotation = Quaternion.Euler(90f, 0f, 0f);
    }
```

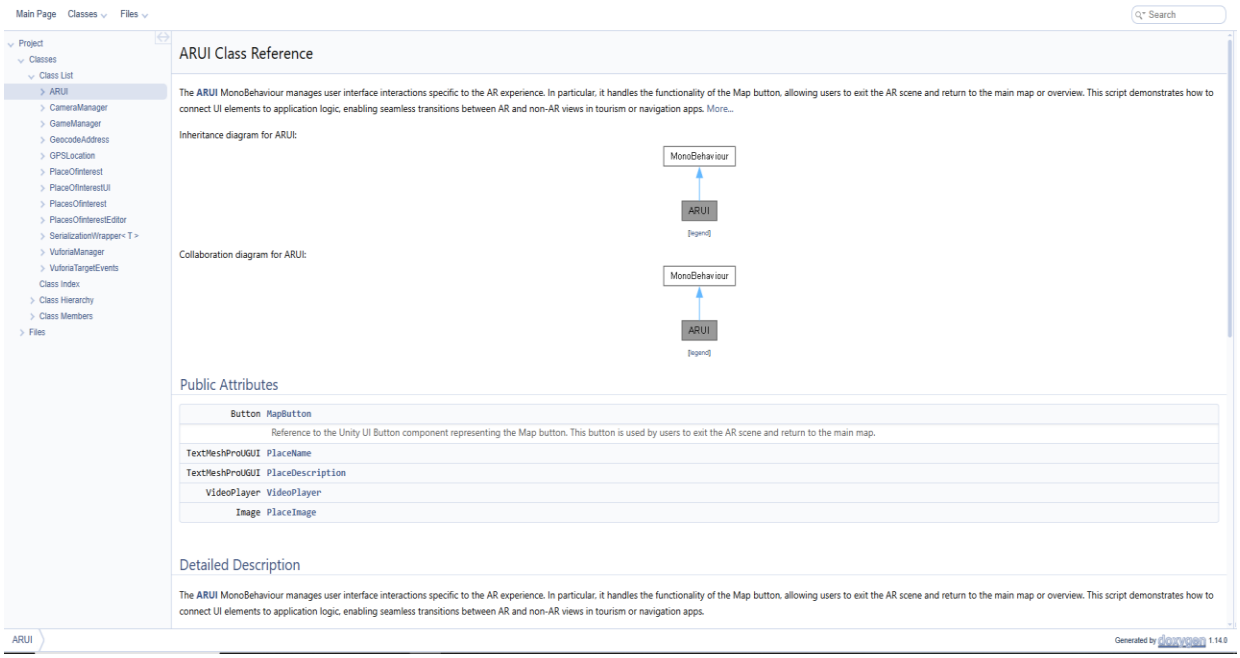
Εικόνα 5.17– Απόσπασμα κώδικα: Τοποθέτηση & προσανατολισμός κάμερας

Σχόλιο:Με αυτόν τον κώδικα, η κάμερα ανεβαίνει κατακόρυφα πάνω από το anchor και στρέφεται προς τα κάτω (90 μοίρες) για να έχει το χαρακτηριστικό top-downview. Έτσι ο χρήστης βλέπει πάντα το POI από ψηλά.

5.3.4ARUI.cs

Η ARUI είναι το script που φροντίζει για το UserInterface μέσα στην AR σκηνή. Όταν ο χρήστης πλησιάσει ένα σημείο ενδιαφέροντος (POI), ενεργοποιείται η AR λειτουργία και το UI γεμίζει με πληροφορίες για το σημείο: όνομα, περιγραφή, εικόνα και βίντεο. Παράλληλα, δίνει τη δυνατότητα επιστροφής στον χάρτη μέσω ενός κουμπιού ("Map").

Η λειτουργία του είναι ξεκάθαρα υποστηρικτική, αλλά πολύ κρίσιμη για την εμπειρία του χρήστη είναι η γέφυρα ανάμεσα στο τεχνολογικό κομμάτι της AR και τη "φιλική" παρουσίασή του.



Εικόνα 5.18– Doxygen αναφορά της κλάσης ARUI.cs

Σχόλιο:Μέσα από το Doxygen βλέπουμε τις βασικές μεταβλητές της UI (PlaceName, MapButton, Video κ.λπ.) και τη βασική ροή του κώδικα. Ειδικά για ομάδες ανάπτυξης, αυτό βοηθά να εντοπιστούν γρήγορα οι σύνδεσμοι με το GameManager.

```
43      if (MapButton != null) MapButton.onClick.AddListener(() => GameManager.Instance.RemoveScene("AR")); // Add listener to the Map button
44
```

Εικόνα 5.19– Απόσπασμα κώδικα: MapButton με Listener

Σχόλιο:Αυτό το απλό, αλλά πολύ ουσιαστικό κομμάτι κώδικα δένει το κουμπί "Map" με τη μέθοδο RemoveScene() του GameManager. Δηλαδή, όταν ο χρήστης πατήσει το κουμπί, βγαίνει από τη σκηνή AR και επιστρέφει στο βασικό χάρτη. Είναι ένα από τα βασικά στοιχεία πλοήγησης της εφαρμογής.

```
if (PlaceName != null)
{
    PlaceName.text = GameManager.Instance.loacalPlaceOfInterest.Name; // Set the place name text
}
if (PlaceDescription != null)
{
    PlaceDescription.text = GameManager.Instance.loacalPlaceOfInterest.Description; // Set the place description text
}
```

Εικόνα 5.20– Απόσπασμα κώδικα: Φόρτωση περιεχομένου στο UI

Σχόλιο:Εδώ το script εμφανίζει το όνομα και την περιγραφή του σημείου ενδιαφέροντος στην AR UI σκηνή. Παίρνει τα δεδομένα από το ενεργό POI στον GameManager και τα εμφανίζει στα αντίστοιχα πεδία του UI. Έτσι, κάθε φορά που πλησιάζεις κάποιο μέρος, βλέπεις άμεσα τις σχετικές πληροφορίες.

```
3       if (VideoPlayer != null && GameManager.Instance.loacalPlaceOfInterest.Video != null)
4       {
5           VideoPlayer.clip = GameManager.Instance.loacalPlaceOfInterest.Video; // Set the video clip for the Video Player
6           VideoPlayer.Play(); // Start playing the video
7       }
8       if (PlaceImage != null && GameManager.Instance.loacalPlaceOfInterest.Icon != null)
9       {
0           PlaceImage.sprite = GameManager.Instance.loacalPlaceOfInterest.Icon; // Set the place icon image
1       }
2
```

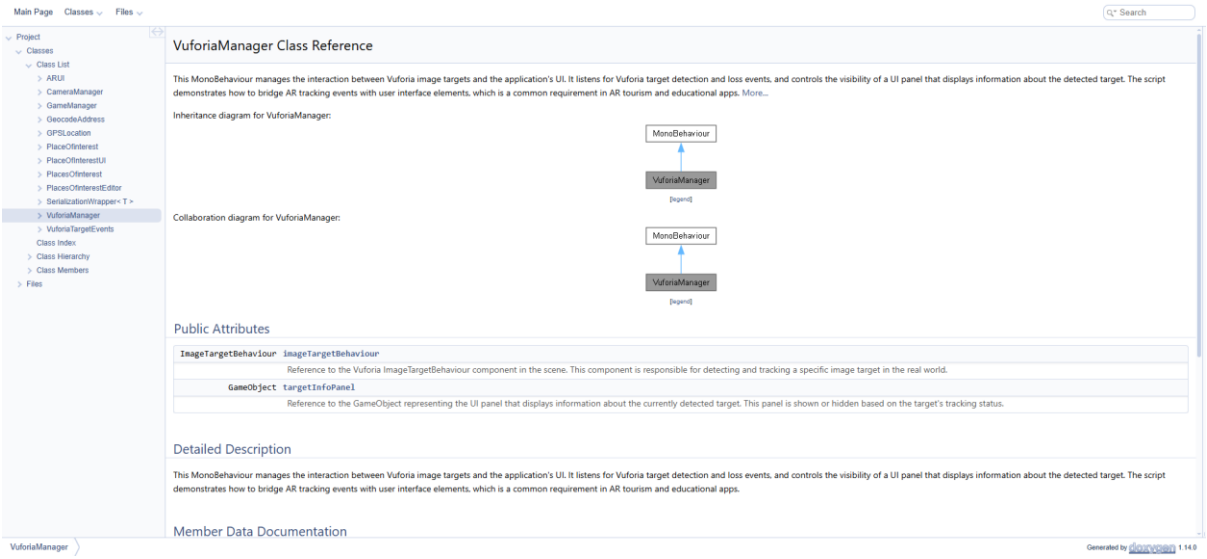
Εικόνα 5.21– Απόσπασμα κώδικα: Προβολή βίντεο και εικόνας

Σχόλιο:Αυτό το μπλοκ φροντίζει να φορτώσει (αν υπάρχουν) το βίντεο και την εικόνα του σημείου ενδιαφέροντος. Είναι ενδεικτικό του πώς τα πολυμέσα ενσωματώνονται στο UI με ασφάλεια — δηλαδή μόνο αν υπάρχουν δεδομένα και references.

5.3.5VuforiaManager.cs

Το VuforiaManager.cs είναι υπεύθυνο για την ενοποίηση του συστήματος αναγνώρισης εικόνων της Vuforia με το UI της εφαρμογής. Κάθε φορά που εντοπίζεται ένα AR imagetarget, το script ενημερώνει δυναμικά το interface, εμφανίζοντας ή κρύβοντας την αντίστοιχη πληροφοριακή κάρτα. Έτσι, δημιουργείται μια διαδραστική εμπειρία όπου ο χρήστης λαμβάνει άμεσα πληροφορίες όταν «σκανάρει» ένα σημείο ενδιαφέροντος.

Η προσέγγιση βασίζεται σε δύο βασικά callbacks (TargetFound, TargetLost) που ενεργοποιούν ή απενεργοποιούν UI στοιχεία, διατηρώντας ταυτόχρονα καθαρό και modular κώδικα.



Εικόνα 5.22- Doxygen overview γιατο VuforiaManager.cs

Σχόλιο:Στην εικόνα φαίνεται η βασική περιγραφή της κλάσηςκαι τα publicfields.

```
32      void Start()
33      {
34
35          if (imageTargetBehaviour != null)
36          {
37              // Add or get the VuforiaTargetEvents component
38              vuforiaTargetEvents = imageTargetBehaviour.GetComponent<VuforiaTargetEvents>();
39              if (vuforiaTargetEvents == null)
40                  vuforiaTargetEvents = imageTargetBehaviour.gameObject.AddComponent<VuforiaTargetEvents>();
41
42              // Subscribe to target events
43              vuforiaTargetEvents.OnTargetFound += TargetFound;
44              vuforiaTargetEvents.OnTargetLost += TargetLost;
45              // Optionally subscribe to OnTargetUpdated and OnTargetStatusChanged if needed
46          }
47
48
49          // Ensure the target info panel is initially hidden
50          if (targetInfoPanel == null)
51          {
52              // Try to auto-assign by searching for a GameObject named "TargetInfoPanel"
53              var foundPanel = GameObject.Find("TargetInfoPanel");
54              if (foundPanel != null)
55              {
56                  targetInfoPanel = foundPanel;
57                  Debug.LogWarning("Target info panel was not assigned in the inspector. Found and assigned GameObject named 'TargetInfoPanel'.");
58              }
59          }
60
61          if (targetInfoPanel != null)
62          {
63              targetInfoPanel.SetActive(false);
64          }
65
66      }
```

Εικόνα 5.23– Μέθοδος Start() με σύνδεση σε Vuforia events

Σχόλιο: Το αρχικό setup της εφαρμογής. Η μέθοδος Start() κάνει εγγραφή στα events ανίχνευσης και απώλειας στόχου, και διασφαλίζει ότι το πάνελ πληροφοριών ξεκινά απενεργοποιημένο.

```
86      void TargetFound(ObserverBehaviour observer)
87      {
88          // Display the target information panel when a target is found
89          if (targetInfoPanel != null)
90              targetInfoPanel.SetActive(true);
91
92          // Optionally, log or use observer.TargetName if needed
93          Debug.Log("Target found: " + observer.TargetName);
94      }
```

Εικόνα 5.24– Μέθοδος TargetFound()

Σχόλιο: Όταν εντοπιστεί imagetarget, η TargetFound() ενεργοποιεί το UI panel και εμφανίζει σχετικές πληροφορίες για το αντικείμενο στον χρήστη.

```
100     /// <param name= observer >The ObserverBehaviour representing the lost target.</param>
101     void TargetLost(ObserverBehaviour observer)
102     {
103         // Hide the target information panel when a target is lost
104         if (targetInfoPanel != null)
105             targetInfoPanel.SetActive(false);
106
107         Debug.Log("Target lost: " + observer.TargetName);
108     }
109 }
```

Εικόνα 5.25– Μέθοδος TargetFound()

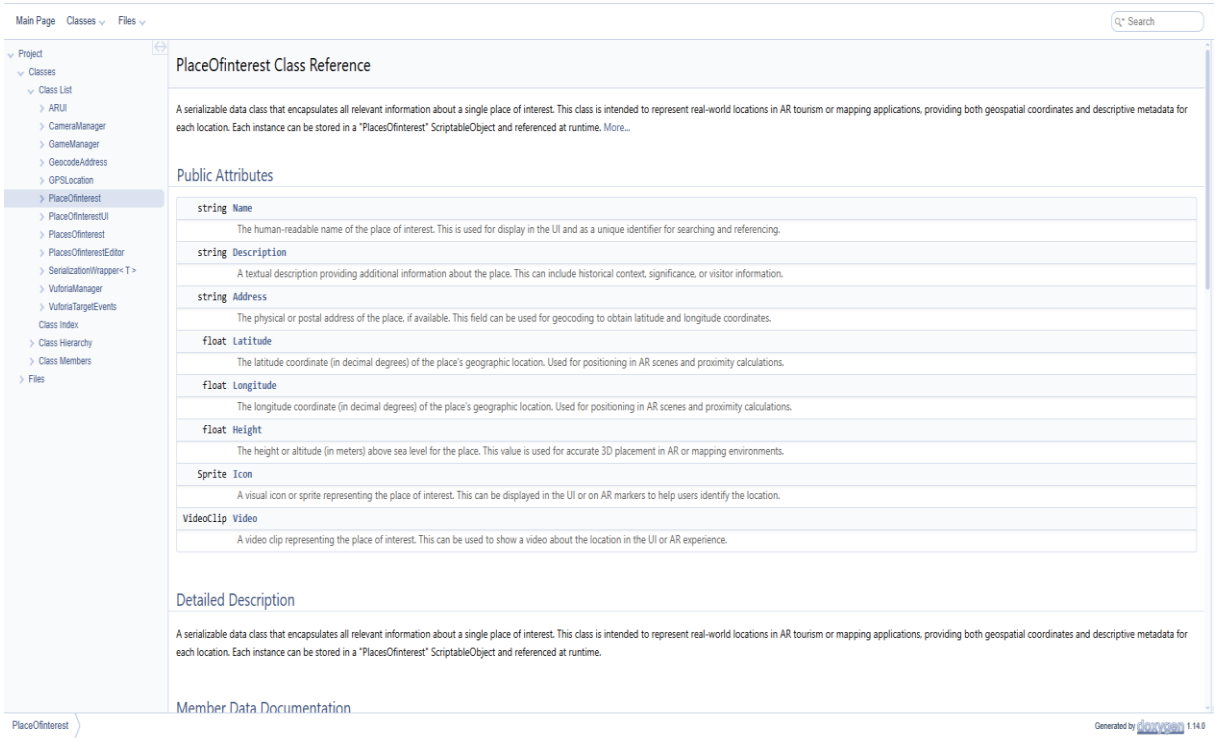
Σχόλιο: Μόλις χαθεί η οπτική επαφή με το target, η TargetLost() κρύβει το πάνελ, αποτρέποντας την προβολή ξεπερασμένων ή άσχετων δεδομένων στο περιβάλλον επαυξημένης πραγματικότητας.

5.3.6PlaceOfinterest.cs

Η PlaceOfinterest είναι μια απλή serializable κλάση που περιέχει όλα τα δεδομένα για ένα σημείο ενδιαφέροντος (Point of Interest – POI). Δεν κληρονομεί από MonoBehaviour, δεν είναι script που τοποθετείται σε GameObject, αλλά χρησιμεύει ως δομικό στοιχείο για να αποθηκεύεις και να εμφανίζεις πληροφορίες όπως:

- Όνομα & Περιγραφή
- Συντεταγμένες (Latitude / Longitude / Height)
- Οπτικό περιεχόμενο (εικόνα και βίντεο)
- Ταχυδρομική διεύθυνση

Χρησιμοποιείται μέσα στο PlacesOfinterestScriptableObject για να συγκεντρωθούν όλα τα POIs της εφαρμογής.



Εικόνα 5.26– Doxygen overview γιατο PlaceOfinterest.cs

Σχόλιο:Το στιγμιότυπο από το Doxygen παρουσιάζει αναλυτικά όλα τα δημόσια πεδία της κλάσης PlaceOfinterest. Κάθε πεδίο συνοδεύεται από περιγραφή, καθιστώντας το screenshot ιδανικό για τεκμηρίωση, χωρίς να είναι απαραίτητη η παράθεση όλου του κώδικα.

```
12  public class PlaceOfinterest
13  {
14
15      public string Name;
16      public string Description;
17      public string Address;
18      public float Latitude;
19      public float Longitude;
20      public float Height;
21      public Sprite Icon;
22      public VideoClip Video;
23  }
24
```

Εικόνα 5.27– Μεταβλήτες τις κλάσης

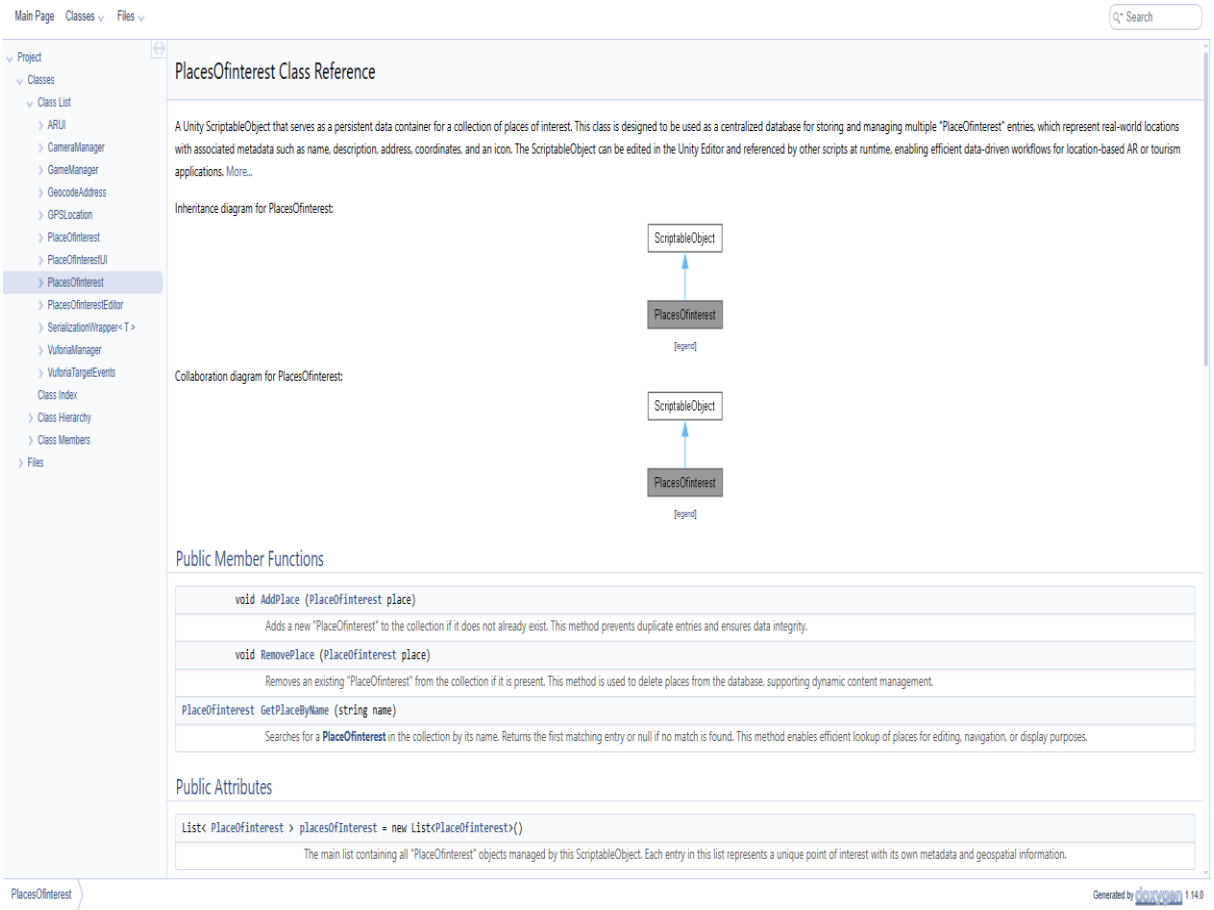
Σχόλιο: Στο screenshot αυτό φαίνεται ότι η PlaceOfinterest είναι μια απλή κλάση που κρατάει όλες τις βασικές πληροφορίες για ένα σημείο ενδιαφέροντος. Περιλαμβάνει το όνομα, την περιγραφή και τη διεύθυνση του μέρους, αλλά και τις γεωγραφικές του συντεταγμένες (γεωγραφικό πλάτος, μήκος και ύψος), ώστε να τοποθετείται σωστά στον χάρτη. Επιπλέον, δίνει τη δυνατότητα να εμφανίζεται και μια εικόνα ή ένα βίντεο για το κάθε σημείο μέσα στην εφαρμογή. Είναι δηλαδή η βάση για να "γεμίσουμε" τον AR χάρτη μας με νοηματικό και οπτικό περιεχόμενο.

5.3.7 PlacesOfinterest.cs

Το scriptPlacesOfinterest είναι ένα ScriptableObject που λειτουργεί σαν μικρή βάση δεδομένων μέσα στο Unity. Περιέχει μια λίστα από σημεία ενδιαφέροντος (PlaceOfinterest) και προσφέρει βασικές λειτουργίες όπως:

- να προσθέτεις νέα σημεία (π.χ. μουσεία, μνημεία),
- να διαγράφεις υπάρχοντα,
- και να βρίσκεις κάποιο με βάση το όνομά του.

Το μεγάλο πλεονέκτημα είναι ότι μπορούμε να συντηρούμε και να οργανώνουμε όλα τα μέρη μας κεντρικά, μέσα από το UnityEditor, χωρίς να τα γράφουμε στον κώδικα κάθε φορά. Είναι το ιδανικό εργαλείο για εφαρμογές AR ή τουριστικές, όπου κάθε σημείο έχει πληροφορίες όπως όνομα, περιγραφή, εικόνα, βίντεο και συντεταγμένες GPS.



Εικόνα 5.28– Doxygen overview γιατο PlacesOfinterest.cs

Σχόλιο: Η εικόνα αυτή δείχνει ότι το PlacesOfinterest είναι ένα ScriptableObject που λειτουργεί σαν "βάση δεδομένων" για όλα τα σημεία ενδιαφέροντος της εφαρμογής. Μέσα του αποθηκεύεται μια λίστα από αντικείμενα τύπου PlaceOfinterest, και από άλλες κλάσεις μπορούμε να τα προσθέτουμε, να τα αφαιρούμε ή να τα αναζητάμε με βάση το όνομα. Είναι πολύ χρήσιμο για να οργανώνουμε τα μέρη μας με εύκολο και επαναχρησιμοποιήσιμο τρόπο στο Unity.

```
public void AddPlace(PlaceOfinterest place)
{
    if (place != null && !placesOfInterest.Contains(place))
    {
        placesOfInterest.Add(place);
    }
}
```

Εικόνα 5.29 Απόσπασμα κώδικα από τη μέθοδο AddPlace

Σχόλιο: Η μέθοδος AddPlace προσθέτει ένα νέο σημείο ενδιαφέροντος στη λίστα, εφόσον δεν υπάρχει ήδη. Μια απλή προστασία για να μην έχουμε διπλοεγγραφές, χρήσιμη αν τα δεδομένα μας προέρχονται από εξωτερικές πηγές ή φορτώνονται πολλές φορές.

```
public void RemovePlace(PlaceOfinterest place)
{
    if (place != null && placesOfInterest.Contains(place))
    {
        placesOfInterest.Remove(place);
    }
}
```

Εικόνα 5.30- Απόσπασμα κώδικα από τη μέθοδο RemovePlace

Σχόλιο: Με αυτή τη μέθοδο μπορούμε να αφαιρέσουμε ένα συγκεκριμένο POI από τη λίστα. Είναι απαραίτητο αν θέλουμε να δίνουμε στον χρήστη τη δυνατότητα να αφαιρέσει ένα σημείο ή να το επεξεργαστούμε δυναμικά κατά την εκτέλεση.

```
53      public PlaceOfinterest GetPlaceByName(string name)
54      {
55          return placesOfInterest.Find(place => place.Name == name);
56      }
57  }
```

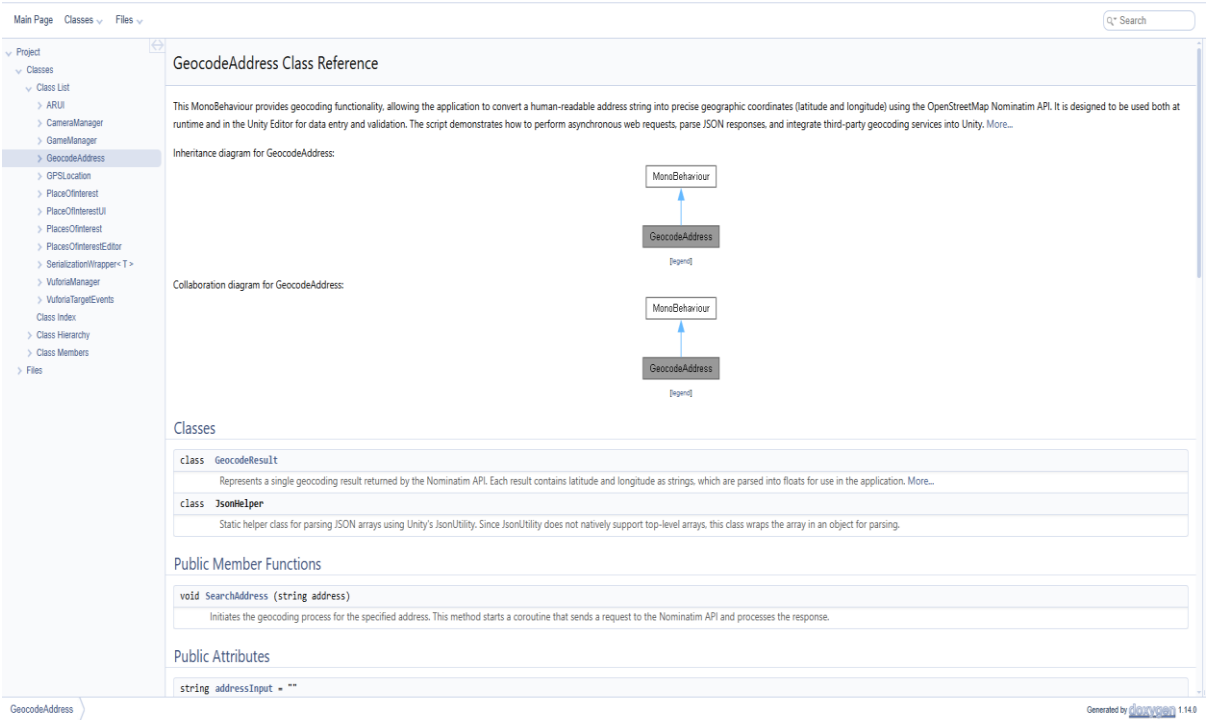
Εικόνα 5.31– Απόσπασμα κώδικα από τη μέθοδο GetPlaceByName

Σχόλιο:Πολύ χρήσιμη μέθοδος για να βρίσκουμε ένα συγκεκριμένο POI με βάση το όνομά του. Χρησιμοποιείται συχνά όταν θέλουμε να δείξουμε πληροφορίες στον χρήστη ή να μεταφερθούμε σε εκείνο το σημείο στην AR σκηνή.

5.3.8GeocodeAddress.cs

Αυτό το script είναι υπεύθυνο για να μετατρέπει μια διεύθυνση (π.χ. "Ακρόπολη, Αθήνα") σε συντεταγμένες GPS (γεωγραφικό πλάτος και μήκος). Ουσιαστικά, «ρωτάει» μια online υπηρεσία (OpenStreetMap / Nominatim) και παίρνει πίσω τα αντίστοιχα αριθμητικά δεδομένα, τα οποία μπορούν μετά να χρησιμοποιηθούν για να εμφανιστεί περιεχόμενο AR σε εκείνη την τοποθεσία.

Χρησιμοποιείται μέσα στο GameManager, και πιο συγκεκριμένα όταν εκτελείται η εντολή UpdateAllPlacesLatLonFromAddress() ώστε κάθε σημείο ενδιαφέροντος να αποκτήσει αυτόματα τις γεωγραφικές του συντεταγμένες.



Εικόνα 5.32– Doxygen overview για το GeocodeAddress.cs

Σχόλιο:Η εικόνα δείχνει την τεκμηρίωση Doxygen για την κλάση GeocodeAddress, με το διάγραμμα κληρονομίας, τις βοηθητικές εσωτερικές κλάσεις (GeocodeResult, JsonHelper) και τη λίστα των δημόσιων μεθόδων/μεταβλητών (π.χ. SearchAddress, addressInput, latitude).

```
public void SearchAddress(string address)
{
    if (!string.IsNullOrEmpty(address))
    {
        StartCoroutine(GetCoordinatesFromAddress(address));
    }
}
```

Εικόνα 5.33– Απόσπασμα κώδικα από τη μέθοδο SearchAddress

Σχόλιο:Αυτή η μέθοδος ξεκινάει τη διαδικασία γεωκωδικοποίησης μόνο όταν ο χρήστης έχει εισάγει έγκυρη διεύθυνση. Εκκινεί το coroutineGetCoordinatesFromAddress, το οποίο στέλνει το αίτημα στο Nominatim API.

```
57      IEnumerator GetCoordinatesFromAddress(string address)
58      {
59          string url = $"https://nominatim.openstreetmap.org/search?q={UnityWebRequest.EscapeURL(address)}&format=json";
60          UnityWebRequest request = UnityWebRequest.Get(url);
61          request.SetRequestHeader("User-Agent", "UnityGeocodeTest/1.0 (nikosakr99@gmail.com)"); // Replace with your email
62
63          yield return request.SendWebRequest();
64
65  #if UNITY_2020_1_OR_NEWER
66          if (request.result != UnityWebRequest.Result.Success)
67  #else
68          if (request.isNetworkError || request.isHttpError)
69  #endif
70          {
71              Debug.LogError("Request error: " + request.error);
72              resultText = "Error: " + request.error;
73          }
74          else
75          {
76              string json = request.downloadHandler.text;
77              GeocodeResult[] results = JsonHelper.FromJson<GeocodeResult>(json);
78
79              if (results != null && results.Length > 0)
80              {
81                  float lat = float.Parse(results[0].lat);
82                  float lon = float.Parse(results[0].lon);
83                  latitude = lat;
84                  longitude = lon;
85                  resultText = $"Lat: {lat}, Lon: {lon}";
86                  Debug.Log($"Lat: {lat}, Lon: {lon}");
87              }
88              else
89              {
90                  resultText = "No results found.";
91                  latitude = 0f;
92                  longitude = 0f;
93              }
94          }
95      }
```

Εικόνα 5.34– Απόσπασμακώδικααπότο coroutine GetCoordinatesFromAddress (request & error handling)

Σχόλιο:Εδώ κατασκευάζεται το URL με σωστή escape της διεύθυνσης και αποστέλλεται το HTTP GET. Σε περίπτωση σφάλματος (π.χ. δικτύου), ενημερώνεται το resultText για εμφάνιση στον inspector ή στο UI. Αφού λάβουμε την απάντηση, τη μετατρέπουμε σε πίνακα GeocodeResult με το JsonHelper. Αν υπάρχει τουλάχιστον ένα αποτέλεσμα, αποθηκεύουμε lat/lon στα αντίστοιχα πεδία και ενημερώνουμε το resultText. Διαφορετικά μηδενίζουμε τις συντεταγμένες και εμφανίζουμε μήνυμα «χωρίς αποτελέσματα».

```
public static class JsonHelper
{
    public static T[] FromJson<T>(string json)
    {
        string wrapped = "{\"array\":" + json + "}";
        Wrapper<T> wrapper = JsonUtility.FromJson<Wrapper<T>>(wrapped);
        return wrapper.array;
    }

    [System.Serializable]
    private class Wrapper<T>
    {
        public T[] array;
    }
}
```

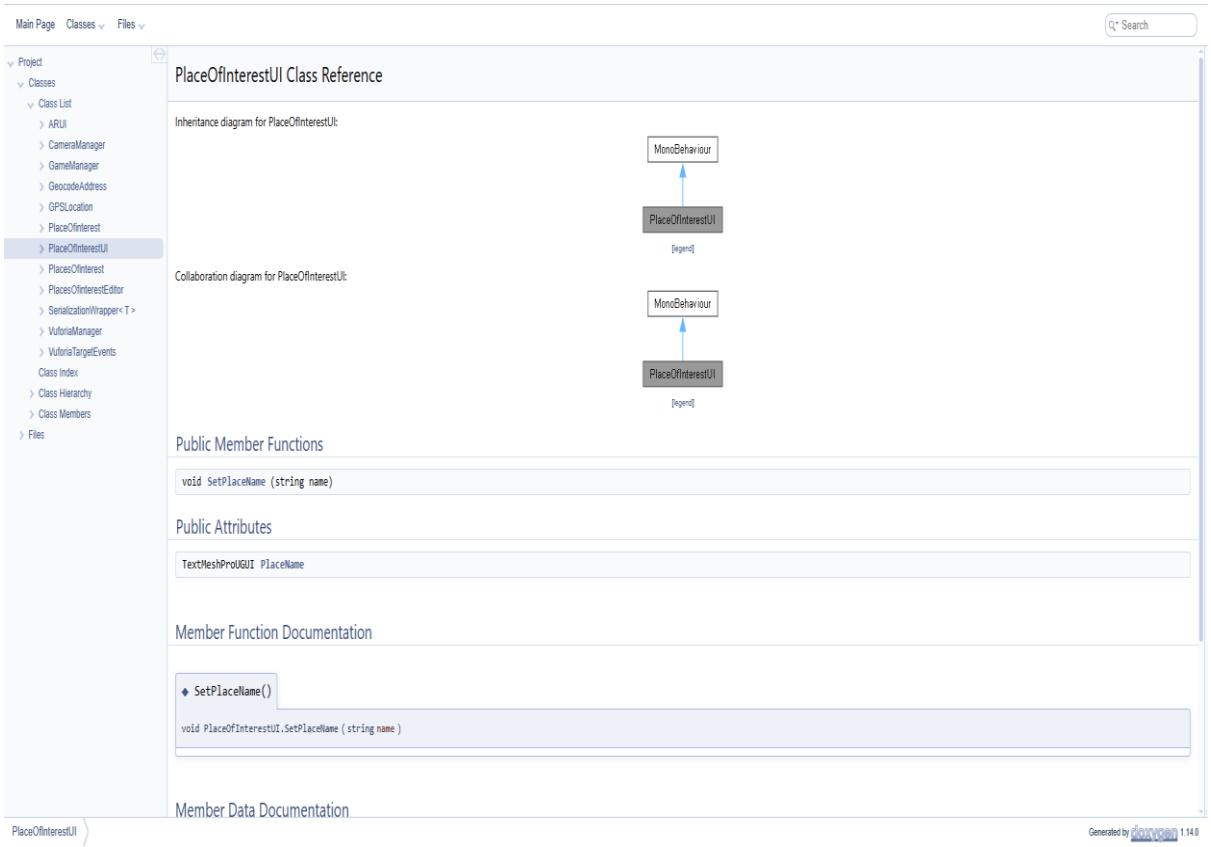
Εικόνα 5.35– Απόσπασμα κώδικα από την εσωτερική κλάση JsonHelper

Σχόλιο:Το Unity Json Utility δεν υποστηρίζει απευθείας κορυφαία JSON arrays, γι’ αυτό τυλίγουμε το JSON με ένα αντικείμενο { "array": [...] } και το κάνουμε deserialize στην εσωτερική Wrapper<T>. Στη συνέχεια επιστρέφουμε το πεδίο array.

5.3.9 PlaceOfInterestUI.cs

Το συγκεκριμένο script έχει πολύ απλό ρόλο: προβάλλει το όνομα ενός σημείου ενδιαφέροντος στην οθόνη. Παίρνει ένα string (το όνομα) και το εμφανίζει σε ένα TextMeshPro πεδίο. Αν για κάποιο λόγο το TextMeshProUGUI δεν έχει οριστεί, εμφανίζεται σχετικό μήνυμα στο Console.

Είναι ένα τυπικό παράδειγμα UI helperscript, που απομονώνει τη λογική εμφάνισης από τα δεδομένα.



Εικόνα 5.36– Τεκμηρίωση Doxygen της PlaceOfInterestUI.cs

Σχόλιο: Στο Doxygen φαίνεται καθαρά ότι η κλάση περιλαμβάνει μία μέθοδο SetPlaceName() και ένα μόνο publicattribute (PlaceName). Είναι ένα απλό helperscript για το UI, το οποίο δείχνει απλά το όνομα του σημείου ενδιαφέροντος πάνω από το αντικείμενο στην AR σκηνή.

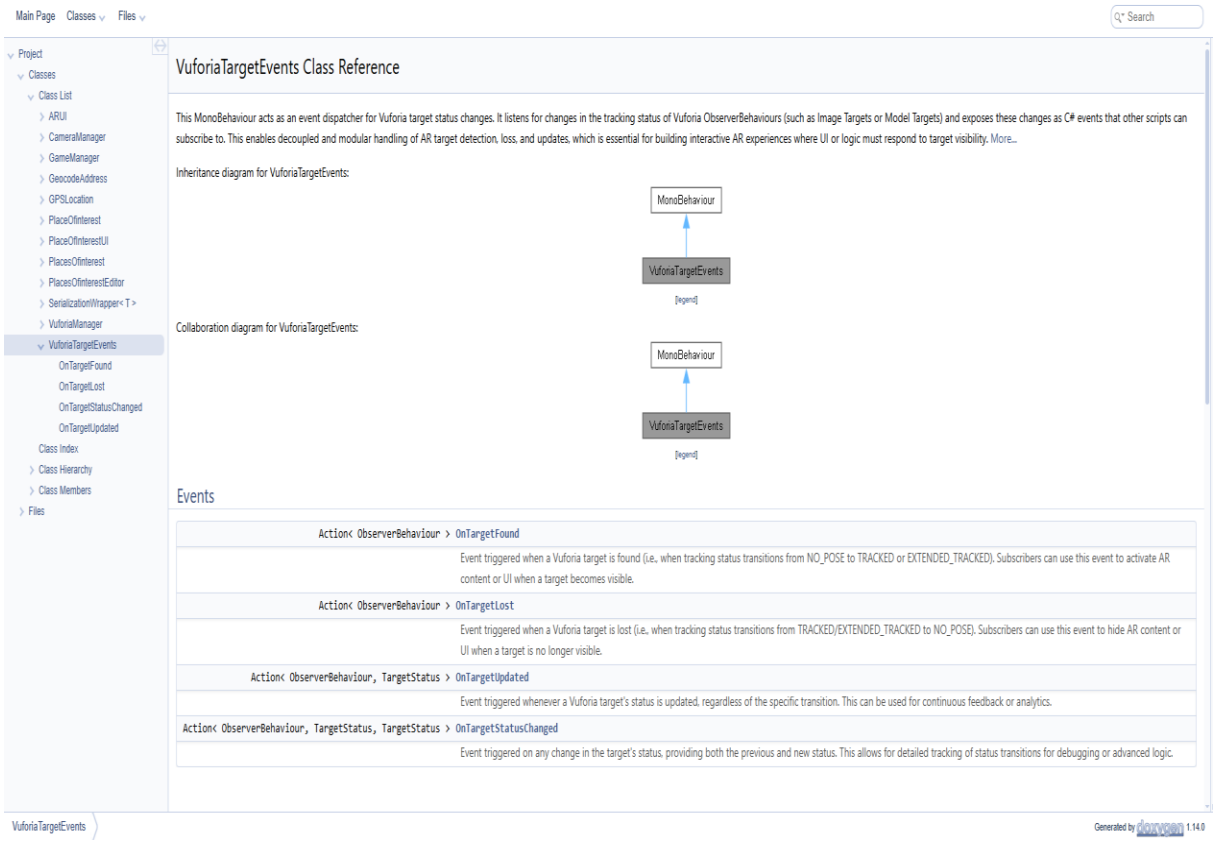
```
1  using UnityEngine;
2  using TMPro;
3
4  public class PlaceOfInterestUI : MonoBehaviour
5  {
6      public TextMeshProUGUI PlaceName; // Reference to the TextMeshProUGUI component for the place name
7
8      public void SetPlaceName(string name)
9      {
10         if (PlaceName != null)
11         {
12             PlaceName.text = name; // Set the place name text
13         }
14         else
15         {
16             Debug.LogWarning("PlaceName TextMeshProUGUI component is not assigned.");
17         }
18     }
19 }
20
```

Εικόνα 5.37 -Απόσπασμα από τη μέθοδο SetPlaceName

Σχόλιο: Ελέγχει αν έχει οριστεί το UI αντικείμενο και το ενημερώνει. Αν όχι, εμφανίζει προειδοποίηση στο Console. Απλός και καθαρός χειρισμός UI ενημέρωσης.

5.3.10 – VuforiaTargetEvents.cs

ToVuforiaTargetEvents.cseίναιέναscriptπουαναλαμβάνειναδιαχειρίζεταιτιςαλλαγέςκατάστασης (status) τωνARtargetsπουπαρακολουθείηVuforia. ΠαρακολουθείταλεγόμεναObserverBehaviours (όπωςεικόνεςήμοντέλαστονARχώρο) καιενεργοποιείαντίστοιχαγεγονότα (C# events) όπωςOnTargetFound, OnTargetLost, OnTargetUpdatedκαιOnTargetStatusChanged. Αυτός ο μηχανισμός είναι ιδανικός για να απεμπλακεί η λογική του συστήματος από την τεχνολογία εντοπισμού — επιτρέποντας καθαρό, modular σχεδιασμό. Τα υπόλοιπα scripts απλά κάνουν subscribe στα events και αντιδρούν (π.χ. εμφάνιση UI όταν εμφανιστεί ένα target).



Εικόνα 5.38– Doxygen τεκμηρίωση για το VuforiaTargetEvents.cs

Σχόλιο:Η εικόνα παρουσιάζει την τεκμηρίωση του Doxygen για την κλάση VuforiaTargetEvents. Παρατηρούμε το διάγραμμα κληρονομικότητας, τις διαθέσιμες μεθόδους και τα events της κλάσης. Αυτό επιβεβαιώνει την οργανωμένη και τεκμηριωμένη προσέγγιση στον σχεδιασμό του script, καθώς και τη χρήση των C# delegates/events για αποσυζευγμένη επικοινωνία μεταξύ των components.

```
    public event Action<ObserverBehaviour> OnTargetFound;
    /// <summary>
    /// Event triggered when a Vuforia target is lost (i.e., when tracking status transitions from TRACKED/EXTENDED_TRACKED to NO_POSE).
    /// Subscribers can use this event to hide AR content or UI when a target is no longer visible.
    /// </summary>
    public event Action<ObserverBehaviour> OnTargetLost;
    /// <summary>
    /// Event triggered whenever a Vuforia target's status is updated, regardless of the specific transition.
    /// This can be used for continuous feedback or analytics.
    /// </summary>
    public event Action<ObserverBehaviour, TargetStatus> OnTargetUpdated;
    /// <summary>
    /// Event triggered on any change in the target's status, providing both the previous and new status.
    /// This allows for detailed tracking of status transitions for debugging or advanced logic.
    /// </summary>
    public event Action<ObserverBehaviour, TargetStatus, TargetStatus> OnTargetStatusChanged;

    /// <summary>
    /// Stores the previous tracking status of the observed target.
    /// This is used to detect transitions between different tracking states.
    /// </summary>
    private TargetStatus previousStatus;

    /// <summary>
    /// Unity lifecycle method called when the script instance is being loaded.
    /// Subscribes to the Vuforia ObserverBehaviour's OnTargetStatusChanged event to monitor tracking changes.
    /// </summary>
```

Εικόνα 5.39– Δήλωση των C# Events.

Σχόλιο:Εδώορίζονταιτατέσσεραβασικάγεγονότα (OnTargetFound, OnTargetLost, OnTargetUpdated, OnTargetStatusChanged) πουμπορούνναχρησιμοποιηθούναπόάλλαscripts.

```
    /// </summary>
    void Start()
    {
        var observer = FindAnyObjectByType<ObserverBehaviour>();


        previousStatus = observer.TargetStatus;
        observer.OnTargetStatusChanged += HandleTargetStatusChanged;

    }
```

Εικόνα 5.40– Start() και εγγραφή στο συμβάν αλλαγής κατάστασης

Σχόλιο: Στο Start(), το script βρίσκει τον πρώτο διαθέσιμο VuforiaObserver (π.χ. ένα ImageTarget) και αποθηκεύει την αρχική του κατάσταση. Στη συνέχεια, κάνει subscribe στο γεγονός OnTargetStatusChanged ώστε να λαμβάνει ενημερώσεις κάθε φορά που αλλάζει η κατάσταση εντοπισμού του στόχου. Αυτή είναι η βασική «γέφυρα» που επιτρέπει στην εφαρμογή να αντιδρά σε AR συμβάντα.

```
    private void HandleTargetStatusChanged(ObserverBehaviour behaviour, TargetStatus status)
    {
        // OnTargetStatusChanged (previous, new)
        OnTargetStatusChanged?.Invoke(behaviour, previousStatus, status);

        // OnTargetFound
        if ((previousStatus.Status == Status.NO_POSE) &&
            (status.Status == Status.TRACKED || status.Status == Status.EXTENDED_TRACKED))
        {
            OnTargetFound?.Invoke(behaviour);
        }
        // OnTargetLost
        else if ((previousStatus.Status == Status.TRACKED || previousStatus.Status == Status.EXTENDED_TRACKED) &&
                 status.Status == Status.NO_POSE)
        {
            OnTargetLost?.Invoke(behaviour);
        }

        // OnTargetUpdated
        OnTargetUpdated?.Invoke(behaviour, status);

        previousStatus = status;
    }
}
```

Εικόνα 5.41– Η μέθοδος HandleTargetStatusChanged

Σχόλιο: Εδώ γίνεται ο κύριος έλεγχος των μεταβάσεων κατάστασης του στόχου. Αν ο στόχος εντοπίστηκε, καλείται OnTargetFound, αν χάθηκε καλείται OnTargetLost, και κάθε φορά που υπάρχει αλλαγή, εκτελείται και OnTargetUpdated. Τέλος, το OnTargetStatusChanged καλείται πάντα, δίνοντας αναλυτική εικόνα για debugging ή πιο σύνθετη λογική.

5.4 Περιγραφή AR και GPS Σκηνών

Σε αυτό το σημείο παρουσιάζονται οι δύο βασικές σκηνές που αναπτύχθηκαν στην εφαρμογή: η σκηνή GPS, η οποία αφορά τον γεωεντοπισμό του χρήστη και την απεικόνιση POIs στον χάρτη, και η σκηνή AR, που ενεργοποιεί την εμπειρία επαυξημένης πραγματικότητας μέσω αναγνώρισης εικόνας.

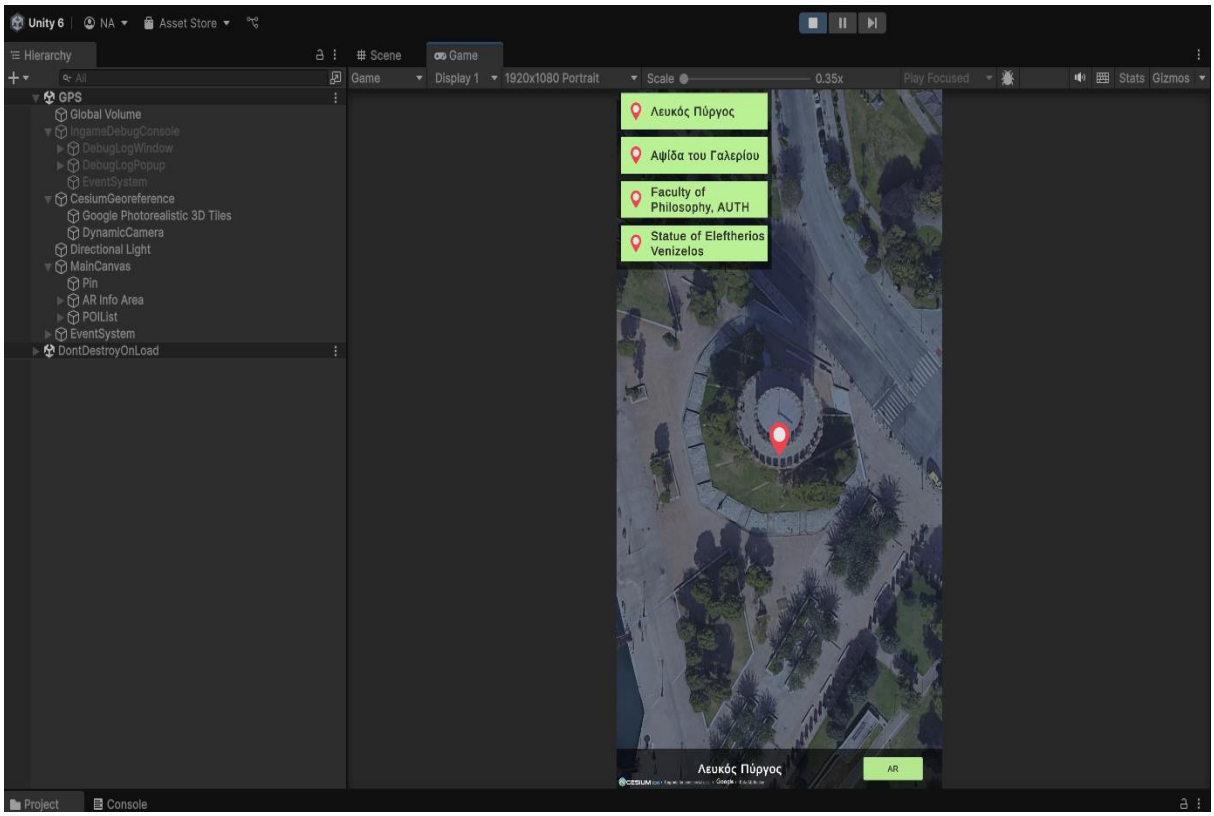
5.4.1 GPSScene

Η GPS σκηνή είναι η πρώτη που βλέπει ο χρήστης. Εδώ εντοπίζεται η τοποθεσία του μέσω GPS και προβάλλονται σημεία ενδιαφέροντος πάνω σε δορυφορικό χάρτη με πλήρη ακρίβεια. Η σκηνή είναι πλήρως λειτουργική χωρίς την ανάγκη ενεργοποίησης AR, προσφέροντας αυτόνομη εμπειρία πλοήγησης.

Περιεχόμενα της σκηνής:

CesiumGeoreference + GooglePhotorealistic 3D Tiles: Επιτρέπουν την ακριβή και ρεαλιστική απεικόνιση του χάρτη με 3D δορυφορικά δεδομένα.
DynamicCamera: Η κάμερα της σκηνής, η οποία κινείται και κάνει zoom ανάλογα με τη θέση και τις ενέργειες του χρήστη.
MainCanvas:Περιέχει όλα τα στοιχεία του γραφικού περιβάλλοντος (UI), όπως την AR λίστα σημείων ενδιαφέροντος, το κουμπί ενεργοποίησης AR και το Pin.
Pin:Δείχνει τη θέση του χρήστη στον χάρτη.

AR Info Area / POIList:Το πλαίσιο με τα POIs που βρίσκονται κοντά στη θέση του χρήστη. Μπορεί να αλληλεπιδράσει με αυτά για να μεταβεί στην AR εμπειρία.
EventSystem:Χειρίζεται τα inputevents του Unity UI (π.χ. αγγίγματα, κλικ).
DontDestroyOnLoad:Το συγκεκριμένο GameObject είναι υπεύθυνο για τη διατήρηση κρίσιμων αντικειμένων (όπως δεδομένα GPS ή settings) όταν γίνεται εναλλαγή σκηνών. Χάρη σε αυτό, η πληροφορία του χρήστη (π.χ. επιλεγμένο POI) δεν χάνεται όταν μεταφερόμαστε στη σκηνή AR.



Εικόνα 5.42– Άποψη της GPS σκηνής στο Unity Editor με εμφανή την τοποθεσία και λίστα POIs

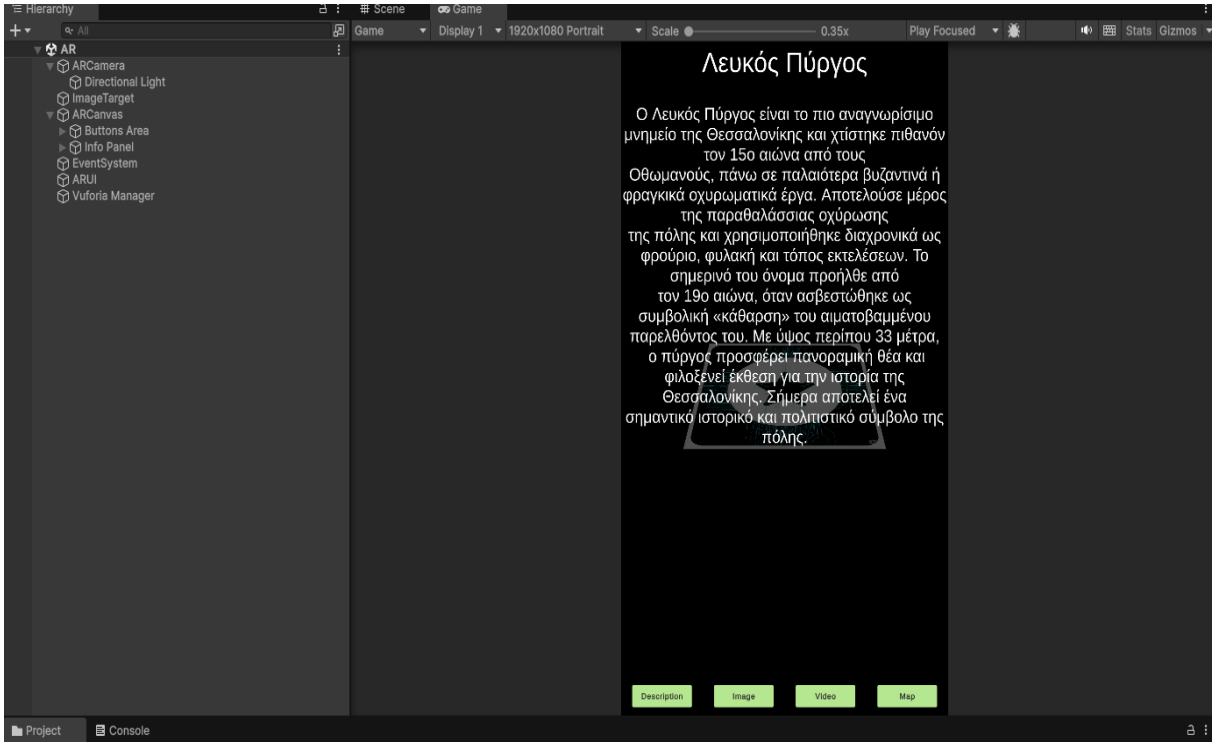
5.4.2 ARScene

Η σκηνή AR ενεργοποιείται όταν ο χρήστης επιλέξει κάποιο σημείο ενδιαφέροντος και πατήσει το κουμπί "AR". Εκεί εμφανίζονται επαυξημένα γραφικά στοιχεία πάνω σε ένα φυσικό imagetarget, προσφέροντας επιπλέον πληροφορίες και δυνατότητες για κάθε σημείο ενδιαφέροντος.

Περιεχόμενα της σκηνής:

ARCamera:Η κάμερα που διαχειρίζεται τη λήψη και την απεικόνιση AR περιεχομένου μέσω του Vuforia SDK.
ImageTarget:Το φυσικό marker (π.χ. αφίσα, εικόνα) που αναγνωρίζεται από τη Vuforia ώστε να ενεργοποιηθεί η προβολή του AR περιεχομένου.
ARCanvas:Το UI της AR σκηνής, περιλαμβάνει πληροφοριακά πάνελ, κουμπιά και άλλες πληροφορίες για τον χρήστη.

InfoPanel:Το κεντρικό πεδίο εμφάνισης πληροφοριών (τίτλος, περιγραφή, εικόνες, βίντεο) για το POI.
ButtonsArea:Περιλαμβάνει τα κουμπιά που ενεργοποιούν την εμφάνιση περιγραφής, φωτογραφίας, βίντεο ή χάρτη.
ARUI:Toscript που χειρίζεται τη λογική πίσω από τα UI interactions.
VuforiaManager:Βασικόcomponent της Vuforia για την αρχικοποίηση και διαχείριση της AR εμπειρίας.



Εικόνα 5.43– Η AR σκηνή όπως φαίνεται στο Game view του Unity, με ενεργοποιημένο περιεχόμενο για τον Λευκό Πύργο.

5.5 Επίλογος Κεφαλαίου

Το παρόν κεφάλαιο είχε ως στόχο την παρουσίαση και ανάλυση της εσωτερικής λειτουργίας της εφαρμογής μέσα από την τεκμηρίωση του κώδικα. Αρχικά, έγινε εισαγωγή στη χρήση του Doxygen ως εργαλείο για αυτόματη παραγωγή τεκμηρίωσης, αλλά το κύριο βάρος δόθηκε στην ουσιαστική κατανόηση της λογικής πίσω από την υλοποίηση.Αναλύθηκαν όλα τα βασικά scripts της εφαρμογής, όπως ο GameManager, το GPSLocation, τα UI components και η διαχείριση στόχων με τη Vuforia. Κάθε κλάση παρουσιάστηκε με περιγραφή, σχολιασμό επιλεγμένων αποσπασμάτων κώδικα και εικόνες από το Doxygen για να γίνει πιο κατανοητή η λειτουργία της. Παράλληλα, παρουσιάστηκε η ροή των δεδομένων ανάμεσα στα components, η αρχιτεκτονική του project και η αλληλεπίδραση των σκηνών μεταξύ τους.

Η όλη διαδικασία βοήθησε να αναδειχθεί το πώς η εφαρμογή λειτουργεί εσωτερικά, πώς διαχειρίζεται δεδομένα πραγματικού χρόνου, και πώς είναι οργανωμένη σε λογικό και επεκτάσιμο επίπεδο. Η τεκμηρίωση, σε συνδυασμό με την εικόνα της σκηνής και των scripts, ολοκλήρωσε μια πιο σφαιρική αποτύπωση της δουλειάς που έγινε στο τεχνικό κομμάτι της εφαρμογής.Μέσα από αυτή τη διαδικασία, κατάλαβα καλύτερα πόσο σημαντικό είναι να γράφεται καθαρός, οργανωμένος και

τεκμηριωμένος κώδικας όχι μόνο για να δουλεύει κάτι σωστά, αλλά και για να μπορεί να εξελιχθεί και να συντηρηθεί με ευκολία στο μέλλον.

Κεφάλαιο 6ο: Παρουσίαση Σημείων Ενδιαφέροντος (POIs)

Στο κεφάλαιο αυτό παρουσιάζονται τα βασικά σημεία ενδιαφέροντος (Points of Interest - POIs) που έχουν ενσωματωθεί στην εφαρμογή. Τα σημεία αυτά επιλέχθηκαν με γνώμονα την ιστορική, πολιτιστική και γεωγραφική σημασία τους για την πόλη της Θεσσαλονίκης. Η πληροφορία που τα συνοδεύει αντλείται από αξιόπιστες πηγές και έχει προσαρμοστεί σε φιλική μορφή για τον χρήστη.

Ο ρόλος των POIs είναι κομβικός για την εμπειρία του τελικού χρήστη, καθώς προσφέρουν:

- Πληροφοριακό περιεχόμενο (περιγραφή, φωτογραφία, βίντεο)
- Διαδραστικότητα μέσω AR (εφόσον πλησιάσει στον χώρο)
- Πολιτιστική και τουριστική αξία, προσελκύοντας τόσο κατοίκους όσο και επισκέπτες της πόλης

Η προβολή κάθε POI υποστηρίζεται από το σύστημα γεωεντοπισμού της εφαρμογής, την υποδομή του Unity και το σύστημα Vuforia για AR εμπειρίες. Ακολουθεί αναλυτική παρουσίαση των σημείων που ενσωματώθηκαν στο prototype.

6.1 Λευκός Πύργος

Ο Λευκός Πύργος αποτελεί το πιο εμβληματικό σημείο ενδιαφέροντος της εφαρμογής. Επιλέχθηκε για την ιστορική και πολιτιστική του σημασία και αποτελεί αναπόσπαστο μέρος της ταυτότητας της Θεσσαλονίκης.Κατά την περιήγηση στον χάρτη, ο χρήστης μπορεί να επιλέξει τον Λευκό Πύργο, να δει βασικές πληροφορίες, φωτογραφία και, εφόσον βρίσκεται κοντά, να ενεργοποιήσει την AR προβολή με επιπλέον υλικό.

6.1.1 Πληροφορίες POI

Όνομα: Λευκός Πύργος

Περιγραφή: Λευκός Πύργος είναι το πιο αναγνωρίσιμο μνημείο της Θεσσαλονίκης και χτίστηκε πιθανόν τον 15ο αιώνα από τους Οθωμανούς, πάνω σε παλαιότερα βυζαντινά ή φραγκικά οχυρωματικά έργα. Αποτελούσε μέρος της παραθαλάσσιας οχύρωσης της πόλης και χρησιμοποιήθηκε διαχρονικά ως φρούριο, φυλακή και τόπος εκτελέσεων. Το σημερινό του όνομα προήλθε από τον 19ο αιώνα, όταν ασβεστώθηκε ως συμβολική «κάθαρση» του αιματοβαμμένου παρελθόντος του. Με ύψος περίπου 33 μέτρα, ο πύργος προσφέρει πανοραμική θέα και φιλοξενεί έκθεση για την ιστορία της Θεσσαλονίκης. Σήμερα αποτελεί ένα σημαντικό ιστορικό και πολιτιστικό σύμβολο της πόλης.

Εικόνα 6.1– Λευκός Πύργος, παλαιά φωτογραφία

Πρόσθετο υλικό (video):https://www.youtube.com/watch?v=cMKP-RBIZ3U

6.2 Αψίδα του Γαλερίου (Καμάρα)

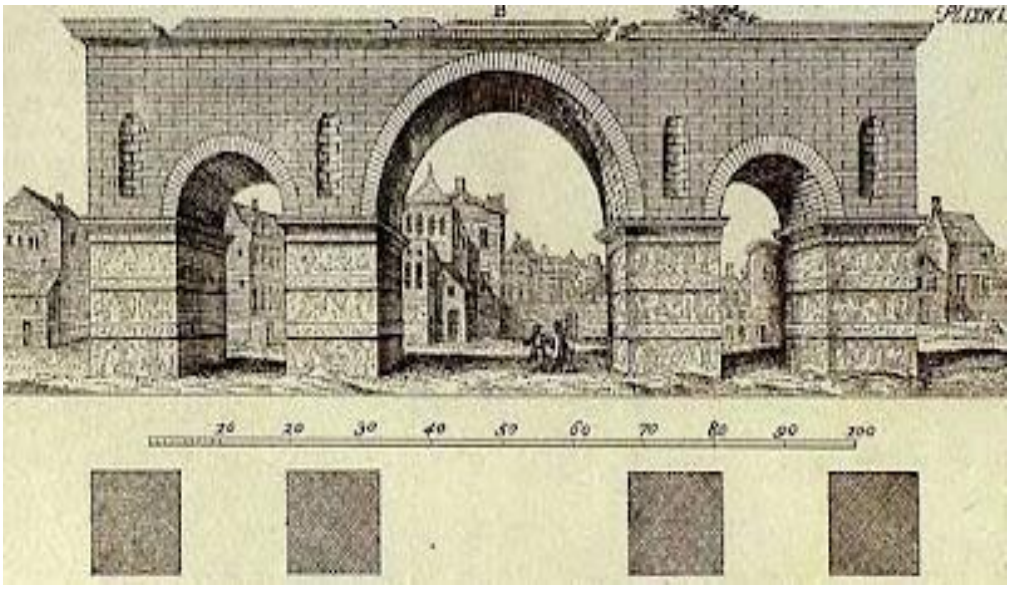
Η Αψίδα του Γαλερίου, γνωστή και ως Καμάρα, αποτελεί ένα από τα πιο χαρακτηριστικά ρωμαϊκά μνημεία της Θεσσαλονίκης και ένα ιδιαίτερο σημείο ενδιαφέροντος μέσα στην εφαρμογή. Επιλέχθηκε για την αρχαιολογική και ιστορική της αξία, καθώς και για τη συμβολική της σύνδεση με το παρελθόν της πόλης.Κατά την περιήγηση στον χάρτη, ο χρήστης εφόσον πλησιάσει το σημείο εντό 10 μέτρων μπορεί να εντοπίσει την Καμάρα, να διαβάσει σχετικές πληροφορίες, να δει εικόνες εποχής και βίντεο.

6.2.1 Πληροφορίες POI

Όνομα: Αψίδα του Γαλερίου (Καμάρα)

Περιγραφή:Η Αψίδα του Γαλερίου, γνωστή και ως Καμάρα, χτίστηκε γύρω στο 298–299 μ.Χ. στη Θεσσαλονίκη για να τιμήσει τη νίκητου αυτοκράτορα Γαλερίου επί των Περσών. Αποτελούσε μέρος

ενός μεγαλύτερου αυτοκρατορικού συγκροτήματος και λειτουργούσε ως μνημείο θριάμβου αλλά και σύμβολο της ρωμαϊκής εξουσίας. Η αρχική της μορφή περιλάμβανε οκτώ κίονεςκαι τρία τόξα, διακοσμημένα με ανάγλυφες παραστάσεις μαχών και αυτοκρατορικών θριάμβων. Συνδεόταν αρχιτεκτονικά με τη Ροτόντα και την παλατιακή κατοικία του Γαλερίου. Παρά τις καταστροφές των αιώνων, η Καμάρα παραμένει ένα από τα σημαντικότερα μνημεία της ρωμαϊκής περιόδου στη Θεσσαλονίκη και εντάχθηκε στα Παγκόσμια Μνημεία της UNESCO το 1988.



Εικόνα 6.2– Σχέδιο της αρχικής μορφής της Καμάρας

Πρόσθετο υλικό (video):Παρουσίαση από το YouTube – Rebuild The Project

6.3 Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης (ΑΠΘ) – Φιλοσοφική Σχολή

Η Φιλοσοφική Σχολή του ΑΠΘ αποτελεί ένα σημείο ιδιαίτερης σημασίας για την πόλη και την εφαρμογή. Επιλέχθηκε όχι μόνο για την ιστορική της βαρύτητα αλλά και ως σύμβολο της ακαδημαϊκής και πνευματικής δραστηριότητας της Θεσσαλονίκης.Κατά την πλοήγηση στον χάρτη, εφόσον πλησιάσει,μπορεί να ενεργοποιήσει την επαυξημένη προβολή για πρόσθετο υλικό.

6.3.1 Πληροφορίες POI

Όνομα: Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης (Φιλοσοφική Σχολή)

Περιγραφή:Το Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης (ΑΠΘ) ιδρύθηκε το 1925 και είναι τομεγαλύτερο πανεπιστήμιο στην Ελλάδα και ένα από τα σημαντικότεραστα Βαλκάνια. Πήρε το όνομά του από τον αρχαίο φιλόσοφο Αριστοτέλη,που καταγόταν από τα ΣτάγειραΧαλκιδικής. Ιδρύθηκε με στόχο την ενίσχυσητης ανώτατης εκπαίδευσης στη Βόρεια Ελλάδα και αποτέλεσε βασικό μοχλόγια την πνευματική, επιστημονική και πολιτιστική ανάπτυξη της Θεσσαλονίκηςκαι της Μακεδονίας. Σήμερα διαθέτει δεκάδες σχολές καιτμήματακαλύπτονταςόλους τους επιστημονικούς τομείς, και αποτελεί κέντρο έρευνας, καινοτομίας και φοιτητικής ζωής.

Εικόνα 6.3– Η Φιλοσοφική Σχολή του ΑΠΘ, ιστορική φωτογραφία

Πρόσθετο υλικό (video):https://www.youtube.com/watch?v=eVlOBtl8BxQ&t=38s

6.4 Πλατεία Αριστοτέλους

Η Πλατεία Αριστοτέλους αποτελεί ένα από τα πιο εμβληματικά αστικά τοπόσημα της Θεσσαλονίκης. Περιλαμβάνεται στην εφαρμογή τόσο για την αρχιτεκτονική της σημασία όσο και για τον ρόλο της ως ζωντανό πολιτιστικό κέντρο της πόλης.Κατά την περιήγηση στον χάρτη, ο χρήστης μπορεί να περπατήσειστην Πλατεία Αριστοτέλους, και εφόσον βρίσκεται κοντά στο άγαλμα του Ελευθερίου Βενιζέλου, να ενεργοποιήσει το AR περιεχόμενο έτσι ώστε να δει το πρόσθετο υλικό.

6.4.1 Πληροφορίες POI

Όνομα: Πλατεία Αριστοτέλους

Περιγραφή:Η πλατεία Αριστοτέλους, το κεντρικό και πιο χαρακτηριστικό σημείο της Θεσσαλονίκης, διαμορφώθηκε μετά τημεγάλη πυρκαγιά του 1917 και σχεδιάστηκε το 1918 από τον Γάλλο αρχιτέκτονα ErnestHébrard με σκοπό να συνδέσει κομψά την παραλία με τον πυρήνα της πόλης .Αν και η αρχική, πιο μεγαλεπίβολη πρόταση δεν υλοποιήθηκε πλήρως λόγω οικονομικών περιορισμών τα δύο κυκλικά κτίρια στις άκρες της και τα γύρω νεοκλασικά μέγαρα ολοκληρώθηκαν στις δεκαετίες του 1950-60, διατηρώντας ταυτόχρονα στοιχεία βυζαντινής και ευρωπαϊκής αρχιτεκτονικής .Σήμερα, η πλατεία φιλοξενεί τη διάσημη προτομή του Αριστοτέλη, τον χώρο συγκέντρωσης για μεγάλες

πολιτικές εκδηλώσεις, συναυλίες, εορτές και καθημερινή ζωή στα καφέ και καταστήματα, ενώ ο μοναδικός «στίβος» της συνδέειτην παραλία (Λ. Νίκης) με την αγορά και το Αρχαίο Φόρουμ.

Εικόνα 6.4– Πλατεία Αριστοτέλους, παλαιά αεροφωτογραφία

Πρόσθετο υλικό (video):https://www.youtube.com/watch?v=53k1rh8S--A

6.5 Επίλογος

Η ενσωμάτωση και παρουσίαση των Σημείων Ενδιαφέροντος (POI) ενισχύει καθοριστικά την εκπαιδευτική και πολιτιστική αξία της εφαρμογής. Μέσα από την επιλογή αντιπροσωπευτικών τοποθεσιών της Θεσσαλονίκης, όπως ο Λευκός Πύργος, η Καμάρα, η Φιλοσοφική Σχολή του ΑΠΘ και η Πλατεία Αριστοτέλους, προσφέρεται στον χρήστη μια διαδραστική εμπειρία εξερεύνησης με ιστορικό και αισθητικό υπόβαθρο.Η χρήση εικόνων, κειμένων και βίντεο, σε συνδυασμό με τη δυνατότητα προβολής επαυξημένης πραγματικότητας (AR), ενισχύει τη σύνδεση του χρήστη με τον χώρο, αναδεικνύοντας τον ρόλο της τεχνολογίας στην προβολή της πολιτιστικής κληρονομιάς. Τα POIs λειτουργούν όχι μόνο ως πληροφοριακά σημεία, αλλά και ως κόμβοι εμβύθισης, προσφέροντας μια πιο βιωματική προσέγγιση στην ιστορία και την ταυτότητα της πόλης.

Κεφάλαιο 7ο:Συμπεράσματα ή/και προτάσεις βελτίωσης

Η παρούσα εργασία αποτέλεσε μια πολύ σημαντική εμπειρία για εμένα, καθώς ήταν το πρώτο μου μεγάλο project και μου έδωσε την ευκαιρία να μάθω πολλά καινούργια πράγματα. Μέσα από τη διαδικασία αυτή ήρθα σε επαφή με τεχνολογίες που δεν είχα χρησιμοποιήσει ξανά και που με ενδιέφεραν πραγματικά, όπως η Επαυξημένη Πραγματικότητα, το Unity και το GPS tracking. Είμαι χαρούμενος με το τελικό αποτέλεσμα, γιατί κατάφερα να ολοκληρώσω μια εφαρμογή που συνδυάζει εντυπωσιακά χαρακτηριστικά και λειτουργεί με τρόπο που μπορεί να φανεί χρήσιμος στον τελικό χρήστη.Η εργασία ήταν μια μεγάλη πρόκληση, κυρίως επειδή έπρεπε να συνδυάσω πολλά διαφορετικά εργαλεία και τεχνολογίες που δεν είχα δουλέψει ξανά. Επίσης, το γεγονός ότι έπρεπε να βρω λύσεις μόνος μου σε αρκετά τεχνικά ζητήματα, με βοήθησε να εξελιχθώ τόσο τεχνικά όσο και ως προς τον τρόπο που οργανώνω και υλοποιώ ένα έργο. Μία από τις βασικές δυσκολίες ήταν πως η εφαρμογή αποδείχτηκε αρκετά «βαριά» και δεν μπορεί να τρέξει σε όλες τις Android συσκευές, κάτι που θα μπορούσε να αντιμετωπιστεί σε μελλοντική έκδοση με βελτιστοποίηση της απόδοσης. Επίσης, η προσπάθεια συγχρονισμού όλων των στοιχείων (AR, χάρτης, UI κτλ.) ήταν αρκετά απαιτητική.Όσον αφορά πιθανές βελτιώσεις στο μέλλον, θα μπορούσε να προστεθεί λειτουργία πλοήγησης για να διευκολύνεται η μετάβαση του χρήστη από το ένα σημείο ενδιαφέροντος στο άλλο. Μια άλλη ενδιαφέρουσα προοπτική θα ήταν η ενσωμάτωση 3D μοντέλων για τα σημεία ενδιαφέροντος, έτσι ώστε η εμπειρία να είναι πιο παραστατική και ζωντανή. Επιπλέον, θα μπορούσε να προστεθεί η δυνατότητα προβολής πληροφοριών χωρίς εικόνες, απλώς στοχεύοντας με το κινητό προς την κατεύθυνση του σημείου, ώστε η εμπειρία να είναι πιο καθαρή και απλή για τον χρήστη.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Milgram, P., &Kishino, F. (1994). A taxonomy of mixed reality visual displays. *IEICE Transactions on Information and Systems*, E77-D(12), 1321–1329.

[2] Billinghurst, M., Clark, A., & Lee, G. (2015). A survey of augmented reality. *Foundations and Trends® in Human–Computer Interaction*, 8(2-3), 73–272.

[3] Unity Technologies. (2023). *Unity Manual*. Διαθέσιμο στο: https://docs.unity3d.com/Manual/index.html

[4] CesiumGS. (2023). *Cesium for Unreal and CesiumJS Documentation*. Διαθέσιμοστο: https://cesium.com/docs/

[5] Google. (2023). *Google Photorealistic 3D Tiles Overview*. Διαθέσιμο στο: https://developers.google.com/maps/documentation/3d-tiles/

[6] Azuma, R. (1997). A Survey of Augmented Reality. *Presence: Teleoperators and Virtual Environments*, 6(4), 355–385.

[7] Wu, H., Lee, S. W., Chang, H., & Liang, J. (2021). AR Navigation for Smart Cities: Challenges and Opportunities. *IEEE Access*, 9, 15678–15689.

[8] Papagiannakis, G., Singh, G., &Magnenat-Thalmann, N. (2008). A survey of mobile and wireless technologies for augmented reality systems. *Computer Animation and Virtual Worlds*, 19(1), 3–22.

[9] C. Kourouthanassis, G. Giaglis. (2012). "A Design Framework for Mobile Augmented Reality Applications." International Journal of Mobile Communications.

[10] R. T. Azuma et al. (2001). "Recent Advances in Augmented Reality." IEEE Computer Graphics and Applications, 21(6), 34–47.

[11] ISO/IEC 25010:2011. "Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models."

Παραρτήματα

Παράρτημα 1 – Script: GameManager.cs

Το παρόν παράρτημα περιλαμβάνει τον πλήρη πηγαίο κώδικα του scriptGameManager.cs, το οποίο αποτελεί μέρος της λειτουργικότητας της εφαρμογής. Ο κώδικας παρατίθεται αυτούσιος, όπως χρησιμοποιήθηκε στο περιβάλλον ανάπτυξης Unity, και συνοδεύεται από βασικό σχολιασμό και εσωτερική τεκμηρίωση.

Η ενσωμάτωσή του στην εφαρμογή ήταν κρίσιμη για την υλοποίηση λειτουργιών που σχετίζονται με:
•τη διαχείριση της κατάστασης του χρήστη,
•την επικοινωνία με συστήματα γεωγραφικού εντοπισμού (GPS),
• την ενεργοποίηση και απόκριση του περιβάλλοντος επαυξημένης πραγματικότητας (AR),
• ή άλλες υποστηρικτικές διεργασίες, όπως η γεωκωδικοποίηση, η εμφάνιση του UI και ο συγχρονισμός δεδομένων.

Η παράθεση του συγκεκριμένου κώδικα ενισχύει τη διαφάνεια της υλοποίησης και επιβεβαιώνει τις περιγραφές και τα συμπεράσματα που αναπτύχθηκαν στο κύριο σώμα της εργασίας.

GameManager.cs

```
using UnityEngine;
using CesiumForUnity;
using System.Collections;
using UnityEngine.SceneManagement; // Added for scene loading
using System;
using TMPro;

/// <summary>
/// The GameManager class is a singleton MonoBehaviour that acts as the central controller for the application's global state and logic.
/// It manages references to key components such as the CesiumGlobeAnchor (for geospatial AR placement), the current GPS coordinates,
/// the collection of places of interest, and UI elements. The GameManager is responsible for orchestrating proximity checks between the user's location
/// and stored places, updating UI elements, handling scene transitions, and synchronizing geocoded data.
/// This class demonstrates best practices for centralized state management in Unity-based AR tourism applications, enabling modularity and scalability.
/// </summary>
public class GameManager : MonoBehaviour
{
    /// <summary>
    /// Reference to the CesiumGlobeAnchor component, which is used to anchor virtual content to real-world geospatial coordinates.
    /// This enables accurate placement of AR objects based on latitude, longitude, and height.
    /// </summary>
    public CesiumGlobeAnchor cesiumGlobeAnchor;

    /// <summary>
    /// The current latitude value, typically updated by the GPSLocation component.
    /// This value is used for proximity calculations and AR content placement.
    /// </summary>
    public float currentLatitude = 37.7749f; // Example: San Francisco latitude

    /// <summary>
    /// The current longitude value, typically updated by the GPSLocation component.
    /// This value is used for proximity calculations and AR content placement.
    /// </summary>
    public float currentLongitude = -122.4194f; // Example: San Francisco longitude

    /// <summary>
    /// Reference to the currently active or nearby place of interest.
    /// This field is updated when the user is within a certain distance of a stored place.
    /// </summary>
    [Header("Local Place of Interest")]
    public PlaceOfinterest loacalPlaceOfInterest; // Reference to the local PlaceOfinterest scriptable object

    /// <summary>
    /// Reference to the ScriptableObject containing the collection of all places of interest.
    /// This enables centralized management and querying of location data.
    /// </summary>
    public PlacesOfinterest placesOfinterest; // Reference to the PlaceOfinterest scriptable object

    /// <summary>
    /// Reference to the GeocodeAddress component, used for converting addresses to coordinates.
    /// This enables dynamic updating of place data based on user input or external sources.
    /// </summary>
    public GeocodeAddress geocodeAddress; // Reference to the GeocodeAddress component

```

GameManager.cs

```
    /// Reference to the GeocodeAddress component, used for converting addresses to coordinates.
    /// This enables dynamic updating of place data based on user input or external sources.
    /// </summary>
    public GeocodeAddress geocodeAddress; // Reference to the GeocodeAddress component

    /// <summary>
    /// Singleton instance of the GameManager, ensuring only one instance exists throughout the application's lifecycle.
    /// This pattern facilitates global access to game state and logic.
    /// </summary>
    public static GameManager Instance { get; private set; }

    /// <summary>
    /// Event invoked whenever the currently active place of interest is updated.
    /// Other components can subscribe to this event to react to changes in proximity or selection.
    /// </summary>
    public Action<PlaceOfinterest> OnPlaceOfInterestUpdated; // Action to notify when a place of interest is updated

    /// <summary>
    /// Reference to the TextMeshProUGUI component used to display the name of the current place of interest in the UI.
    /// </summary>
    [Header("UI Stuff")]
    public TextMeshProUGUI placeNameText; // Reference to the UI TextMeshProUGUI for displaying the place name

    /// <summary>
    /// Reference to the GameObject representing the map pin or marker in the scene.
    /// This object is shown or hidden based on scene transitions and user interactions.
    /// </summary>
    public GameObject pin;

    /// <summary>
    /// Reference to the GameObject representing the AR panel UI.
    /// This panel is activated when the user is near a place of interest and deactivated otherwise.
    /// </summary>
    public GameObject arPanel;

    public GameObject POIUI;

    public GameObject POIList;

    public GameObject mainCanvas;


    [Header("Cesium Geo Reference")]
    public CesiumGeoreference cesiumGeoreference; // Reference to the CesiumGeoreference component

    /// <summary>
    /// Unity lifecycle method called when the script is enabled.
    /// Subscribes the UpdateButtonUI method to the OnPlaceOfInterestUpdated event, ensuring the UI is refreshed when the active place changes.
    /// </summary>
    void OnEnable()
    {
        OnPlaceOfInterestUpdated += UpdateButtonUI; // Subscribe to the event
    }
```

99 % No issues found

GameManager.cs

```

    /// <summary>
    /// Unity lifecycle method called when the script instance is being loaded.
    /// Implements the singleton pattern by ensuring only one instance of GameManager exists and persists across scene loads.
    /// </summary>
    void Awake()
    {
        // Ensure that there is only one instance of GameManager
        if (Instance == null)
        {
            Instance = this;
            DontDestroyOnLoad(gameObject); // Keep this instance across scenes
        }
        else if (Instance != this)
        {
            Destroy(gameObject); // Destroy duplicate instances
            return;
        }

    }

    /// <summary>
    /// Unity lifecycle method called before the first frame update.
    /// Starts coroutines for updating geocoded data and checking proximity to places of interest.
    /// Also ensures the AR panel is initially hidden.
    /// </summary>
    void Start()
    {
        // Prevent device from sleeping/locking
        Screen.sleepTimeout = SleepTimeout.NeverSleep;

        StartCoroutine(UpdateAllPlacesLatLonFromAddress());
        StartCoroutine(ProximityCheckCoroutine());

        arPanel.SetActive(false); // Initially deactivate the AR panel


        foreach (PlaceOfinterest placeOfinterest in placesOfinterest.placesOfInterest)
        {
            PlaceOfInterestUI clone = Instantiate(POIUI, POIList.transform).GetComponent<PlaceOfInterestUI>(); // Instantiate the POI UI for each place of interest

            clone.SetPlaceName(placeOfinterest.Name); // Set the place name in the UI
        }

    }


    /// <summary>
    /// Coroutine that periodically checks the user's proximity to all stored places of interest.
    /// If the user is within a specified distance (e.g., 10 meters) of a place, the AR panel is activated and the place is set as active.
    /// This enables context-aware AR experiences based on real-world location.
    /// </summary>
    /// <returns>An IEnumerator for coroutine execution.</returns>
    private IEnumerator ProximityCheckCoroutine()
```

No issues found

•

GameManager.cs

```
    /// </summary>
    /// <returns>An IEnumerator for coroutine execution.</returns>
    private IEnumerator ProximityCheckCoroutine()
    {
        while (true)
        {
            CheckProximityToPlaces();
            yield return new WaitForSeconds(1f); // Check every 1 second
        }
    }

    /// <summary>
    /// Coroutine that iterates through all places of interest and updates their latitude and longitude fields by geocoding their addresses.
    /// This ensures that all places have valid coordinates for AR placement and proximity calculations.
    /// </summary>
    /// <returns>An IEnumerator for coroutine execution.</returns>
    private IEnumerator UpdateAllPlacesLatLonFromAddress()
    {
        if (geocodeAddress == null)
        {
            geocodeAddress = GetComponent<GeocodeAddress>();
            if (geocodeAddress == null)
            {
                Debug.LogError("No GeocodeAddress component found in the Game Manager.");
                yield break;
            }
        }

        foreach (var place in placesOfinterest.placesOfInterest)
        {
            if ((place.Latitude == 0f && place.Longitude == 0f) && !string.IsNullOrEmpty(place.Address))
            {
                geocodeAddress.latitude = 0f;
                geocodeAddress.longitude = 0f;
                geocodeAddress.resultText = "";
                geocodeAddress.SearchAddress(place.Address);

                // Wait until the geocoding completes (resultText is set or timeout)
                float timeout = 10f;
                float timer = 0f;
                while (geocodeAddress.resultText == "" && timer < timeout)
                {
                    timer += Time.deltaTime;
                    yield return null;
                }

                // Update place with found coordinates
                place.Latitude = geocodeAddress.latitude;
                place.Longitude = geocodeAddress.longitude;
            }
        }
    }

    /// <summary>
```

99 % No issues found

GameManager.cs

```
    /// <summary>
    /// Updates the CesiumGlobeAnchor and current coordinates to navigate to a specific place of interest.
    /// This method is used to reposition AR content or the camera based on user selection or navigation requests.
    /// </summary>
    /// <param name="place">The place of interest to navigate to.</param>
    public void NavigateToPlace(PlaceOfInterest place)
    {
        // Logic to navigate to a specific place using the provided latitude and longitude
        Debug.Log($"Navigating to Place at Latitude: {place.Latitude}, Longitude: {place.Longitude}");

        // Here you would typically update the CesiumGlobeAnchor or other navigation logic
        currentLatitude = place.Latitude;
        currentLongitude = place.Longitude;

        cesiumGlobeAnchor.latitude = place.Latitude;
        cesiumGlobeAnchor.longitude = place.Longitude;
        cesiumGlobeAnchor.height = place.Height; // Use the place's height
    }

    /// <summary>
    /// Loads a new Unity scene additively by its name and hides the map pin.
    /// This method is used to transition to AR or detail scenes while preserving the main scene context.
    /// </summary>
    /// <param name="sceneName">The name of the scene to load.</param>
    public void AddScene(string sceneName)
    {
        SceneManager.LoadScene(sceneName, LoadSceneMode.Additive);
        pin.SetActive(false); // Activate the pin when a scene is added
        cesiumGeoreference.gameObject.SetActive(false); // Optionally deactivate the CesiumGeoreference if not needed in the new scene
        mainCanvas.SetActive(false); // Optionally deactivate the main canvas if not needed in the new scene
        POIList.SetActive(false); // Optionally deactivate the POI list if not needed in the new scene
        Debug.Log($"Scene '{sceneName}' loaded.");
    }

    /// <summary>
    /// Unloads a Unity scene by its name and shows the map pin.
    /// This method is used to return from AR or detail scenes to the main map or overview scene.
    /// </summary>
    /// <param name="sceneName">The name of the scene to unload.</param>
    public void RemoveScene(string sceneName)
    {
        Scene scene = SceneManager.GetSceneByName(sceneName);
        if (scene.isLoaded)
        {
            SceneManager.UnloadSceneAsync(sceneName);
            cesiumGeoreference.gameObject.SetActive(true); // Reactivate the CesiumGeoreference if it was deactivated
            pin.SetActive(true); // Deactivate the pin when a scene is removed
            mainCanvas.SetActive(true); // Reactivate the main canvas if it was deactivated
            POIList.SetActive(true); // Reactivate the POI list if it was deactivated
            Debug.Log($"Scene '{sceneName}' unloaded.");
        }
        else
        {
            Debug.LogWarning($"Scene '{sceneName}' is not loaded and cannot be unloaded.");
```

GameManager.cs

```
276         else
277         {
278             Debug.LogWarning($"Scene '{sceneName}' is not loaded and cannot be unloaded.");
279         }
280     }
281
282     /// <summary>
283     /// Checks if the user's current location is within a specified distance (e.g., 10 meters) of any stored place of interest.
284     /// If a nearby place is found, it is set as the active place, the AR panel is shown, and the OnPlaceOfInterestUpdated event is invoked.
285     /// If no places are nearby, the AR panel is hidden and the active place is cleared.
286     /// </summary>
287     private void CheckProximityToPlaces()
288     {
289         if (placesOfinterest == null || placesOfinterest.placesOfInterest == null)
290             return;
291
292         bool foundNearby = false;
293         foreach (var place in placesOfinterest.placesOfInterest)
294         {
295             if (place.Latitude == 0f && place.Longitude == 0f)
296                 continue;
297
298             double distance = HaversineDistance(currentLatitude, currentLongitude, place.Latitude, place.Longitude);
299             if (distance <= 10.0)
300             {
301                 Debug.Log($"You are within 10 meters of {place.Name}!"); // Added log message
302                 loacalPlaceOfInterest = place; // Update the local place of interest
303                 foundNearby = true;
304
305                 OnPlaceOfInterestUpdated?.Invoke(place); // Notify subscribers about the updated place of interest
306
307                 arPanel.SetActive(true); // Activate the AR panel when within proximity
308                 break;
309             }
310         }
311         if (!foundNearby)
312         {
313             loacalPlaceOfInterest = null;
314             arPanel.SetActive(false); // Deactivate the AR panel if no places are nearby
315         }
316     }
317
318     /// <summary>
319     /// Calculates the great-circle distance in meters between two geographic coordinates using the Haversine formula.
320     /// This method is used for accurate proximity detection in location-based AR applications.
321     /// </summary>
322     /// <param name="lat1">Latitude of the first point in decimal degrees.</param>
323     /// <param name="lon1">Longitude of the first point in decimal degrees.</param>
324     /// <param name="lat2">Latitude of the second point in decimal degrees.</param>
325     /// <param name="lon2">Longitude of the second point in decimal degrees.</param>
326     /// <returns>The distance between the two points in meters.</returns>
327     private double HaversineDistance(double lat1, double lon1, double lat2, double lon2)
328     {
329         double R = 6371000; // Earth radius in meters
```

99 % No issues found

```
/// <param name="lon2">Longitude of the second point in decimal degrees.</param>
/// <returns>The distance between the two points in meters.</returns>
private double HaversineDistance(double lat1, double lon1, double lat2, double lon2)
{
    double R = 6371000; // Earth radius in meters
    double dLat = Mathf.Deg2Rad * (float)(lat2 - lat1);
    double dLon = Mathf.Deg2Rad * (float)(lon2 - lon1);

    double a =
        Mathf.Sin((float)dLat / 2) * Mathf.Sin((float)dLat / 2) +
        Mathf.Cos(Mathf.Deg2Rad * (float)lat1) * Mathf.Cos(Mathf.Deg2Rad * (float)lat2) *
        Mathf.Sin((float)dLon / 2) * Mathf.Sin((float)dLon / 2);

    double c = 2 * Mathf.Atan2(Mathf.Sqrt((float)a), Mathf.Sqrt((float)(1 - a)));
    double distance = R * c;
    return distance;
}
}
```

Παράρτημα 2 – Script: GPSLocation.cs

Το script`GPSLocation` είναι υπεύθυνο για την ανάκτηση των γεωγραφικών συντεταγμένων (γεωγραφικό πλάτος και μήκος) από τον αισθητήρα GPS της συσκευής. Εκκινεί την υπηρεσία τοποθεσίας κατά την έναρξη της εφαρμογής, ζητά τα απαραίτητα δικαιώματα (σε Android), και ενημερώνει συνεχώς τις τιμές αυτές σε κάθε frame. Παράλληλα, συγχρονίζει τα δεδομένα με τον `GameManager`, ώστε να είναι διαθέσιμα σε όλη την εφαρμογή για έλεγχο εγγύτητας, εμφάνιση περιεχομένου και γεωεντοπισμένη πλοήγηση.

GPSLocation.cs

```
using UnityEngine;
using System.Collections;

/// <summary>
/// This MonoBehaviour is responsible for accessing the device's GPS hardware and retrieving the current geographic coordinates (latitude and longitude).
/// It requests location permissions at runtime (especially on Android), starts the location service, and continuously updates the coordinates.
/// The script also synchronizes the retrieved GPS data with the GameManager, enabling location-based logic such as proximity detection and AR content placement.
/// This component is essential for any AR application that relies on real-world positioning, such as tourism guides or location-based games.
/// </summary>
public class GPSLocation : MonoBehaviour
{
    /// <summary>
    /// The most recently retrieved latitude value from the device's GPS sensor.
    /// This value is updated every frame while the location service is running.
    /// </summary>
    public float latitude;

    /// <summary>
    /// The most recently retrieved longitude value from the device's GPS sensor.
    /// This value is updated every frame while the location service is running.
    /// </summary>
    public float longitude;

    /// <summary>
    /// Unity lifecycle method called once before the first frame update.
    /// Handles requesting location permissions (on Android) and starts the coroutine to initialize the location service.
    /// This ensures that the application has the necessary permissions and that GPS data is available as soon as possible.
    /// </summary>
    void Start()
    {
#if UNITY_ANDROID
        // Request location permission at runtime on Android
        if (!UnityEngine.Android.Permission.HasUserAuthorizedPermission(UnityEngine.Android.Permission.FineLocation))
        {
            UnityEngine.Android.Permission.RequestUserPermission(UnityEngine.Android.Permission.FineLocation);
        }
#endif
        StartCoroutine(StartLocationService());
    }

    /// <summary>
    /// Coroutine that manages the initialization and retrieval of GPS data from the device.
    /// It waits for user permission, checks if location services are enabled, and handles timeouts or errors gracefully.
    /// Upon successful initialization, it retrieves the initial latitude and longitude values.
    /// </summary>
    IEnumerator StartLocationService()
    {
#if UNITY_ANDROID
        // Wait until the user responds to the permission dialog
        while (!UnityEngine.Android.Permission.HasUserAuthorizedPermission(UnityEngine.Android.Permission.FineLocation))
        {
            Debug.Log("Waiting for location permission...");
            yield return null;
        }
```

```
    /// Also updates the GameManager's coordinates to keep the global game state in
    /// </summary>
    void Update()
    {
        // Optionally, print location every frame if available
        if (Input.location.status == LocationServiceStatus.Running)
        {
            latitude = Input.location.lastData.latitude;
            longitude = Input.location.lastData.longitude;
            Debug.Log("Lat: " + latitude + " Lon: " + longitude);

            // Update GameManager's lat/lon if instance exists
            if (GameManager.Instance != null)
            {
                GameManager.Instance.currentLatitude = latitude;
                GameManager.Instance.currentLongitude = longitude;
            }
        }
    }
}
```

Παράρτημα 3 – Script:CameraManager.cs

Ο παρακάτω κώδικας αποτελεί το script CameraManager, το οποίο είναι υπεύθυνο για τη δυναμική τοποθέτηση και προσανατολισμό της κάμερας στην Unity σε σχέση με το CesiumGlobeAnchor. Η κάμερα διατηρεί σταθερό ύψος και παρακολουθεί από ψηλά το σημείο-άγκυρα, προσφέροντας εποπτική προβολή. Η λειτουργία αυτή είναι κρίσιμη σε εφαρμογές Επαυξημένης Πραγματικότητας (AR) και γεωχωρικής απεικόνισης.

CameraManager.cs

```
using UnityEngine;

/// <summary>
/// The CameraManager MonoBehaviour is responsible for dynamically positioning and orienting the main camera in relation to the CesiumGlobeAnchor.
/// This script ensures that the camera maintains a consistent height above the anchor's position and is oriented to look directly down at the anchor.
/// Such functionality is essential in AR and mapping applications where the camera must follow or focus on a specific geospatial location,
/// providing users with an intuitive and context-aware view of the virtual environment.
/// </summary>
[RequireComponent(typeof(GameManager))]
public class CameraManager : MonoBehaviour
{
    /// <summary>
    /// The vertical offset (in meters) above the CesiumGlobeAnchor at which the camera should be positioned.
    /// This allows for adjustable camera elevation to suit different visualization needs.
    /// </summary>
    public float heightOffset = 200f; // Adjustable height above the anchor

    /// <summary>
    /// Unity lifecycle method called once per frame.
    /// Updates the camera's position to be directly above the CesiumGlobeAnchor at the specified height offset,
    /// and sets the camera's rotation to look straight down at the anchor's position.
    /// This ensures a top-down view centered on the current geospatial location.
    /// </summary>
    void Update()
    {
        // Example: Adjust camera position based on GameManager's CesiumGlobeAnchor
        if (GameManager.Instance != null && GameManager.Instance.cesiumGlobeAnchor != null)
        {
            // Set the camera position to the Cesium Globe Anchor's position
            if (GameManager.Instance.currentLatitude != null && GameManager.Instance.currentLongitude != null)
            {
                GameManager.Instance.cesiumGlobeAnchor.latitude = GameManager.Instance.currentLatitude;
                GameManager.Instance.cesiumGlobeAnchor.longitude = GameManager.Instance.currentLongitude;
                GameManager.Instance.cesiumGlobeAnchor.height = heightOffset;
            }

            // Make the camera look straight down at the anchor's position
            Transform anchorTransform = GameManager.Instance.cesiumGlobeAnchor.transform;
            if (anchorTransform != null)
            {
                // Position the camera above the anchor
                transform.position = anchorTransform.position + Vector3.up * heightOffset;
                // Look directly down at the anchor
                transform.rotation = Quaternion.Euler(90f, 0f, 0f);
            }
        }
    }
}

```

Παράρτημα 4 – Script:ARUI.cs

Η ARUI είναι το script που φροντίζει για το User Interface μέσα στην AR σκηνή. Όταν ο χρήστης πλησιάσει ένα σημείο ενδιαφέροντος (POI), ενεργοποιείται η AR λειτουργία και το UI γεμίζει με πληροφορίες για το σημείο: όνομα, περιγραφή, εικόνα και βίντεο. Παράλληλα, δίνει τη δυνατότητα επιστροφής στον χάρτη μέσω ενός κουμπιού ("Map").

.

ARUI.cs

```
using UnityEngine;
using UnityEngine.UI;
using TMPro;
using UnityEngine.Video;

/// <summary>
/// The ARUI MonoBehaviour manages user interface interactions specific to the AR experience.
/// In particular, it handles the functionality of the Map button, allowing users to exit the AR scene and return to the main map or overview.
/// This script demonstrates how to connect UI elements to application logic, enabling seamless transitions between AR and non-AR views in tourism or navigation apps.
/// </summary>
public class ARUI : MonoBehaviour
{
    /// <summary>
    /// Reference to the Unity UI Button component representing the Map button.
    /// This button is used by users to exit the AR scene and return to the main map.
    /// </summary>
    public Button MapButton; // Reference to the Map button (remove static)

    public TextMeshProUGUI PlaceName; // Reference to the Map button text
    public TextMeshProUGUI PlaceDescription; // Reference to the Map button description

    public VideoPlayer VideoPlayer; // Reference to the Video Player component for displaying videos related to the place of interest

    public Image PlaceImage; // Reference to the Image component for displaying the icon of the place of interest
    /// <summary>
    /// Unity lifecycle method called before the first frame update.
    /// Subscribes the Map button's click event to the RemoveScene method of the GameManager, ensuring that clicking the button unloads the AR scene.
    /// This enables intuitive navigation and scene management for end users.
    /// </summary>
    void Start()
    {
        StartCoroutine(SetupMapButton());
    }

    private System.Collections.IEnumerator SetupMapButton()
    {
        if (GameManager.Instance == null)
        {
            Debug.Log("<color=aqua> GameManager instance is not available. Ensure it is initialized before ARUI.</color>");
            yield break; // Exit if GameManager is not available
        }
        yield return new WaitForSeconds(0.1f);
        if (MapButton != null) MapButton.onClick.AddListener(() => GameManager.Instance.RemoveScene("AR")); // Add listener to the Map button

        if (PlaceName != null)
        {
            PlaceName.text = GameManager.Instance.loacalPlaceOfInterest.Name; // Set the place name text
        }
        if (PlaceDescription != null)
        {
            PlaceDescription.text = GameManager.Instance.loacalPlaceOfInterest.Description; // Set the place description text
        }
        if (VideoPlayer != null && GameManager.Instance.loacalPlaceOfInterest.Video != null)
        {
            VideoPlayer.clip = GameManager.Instance.loacalPlaceOfInterest.Video; // Set the video clip for the Video Player
            VideoPlayer.Play(); // Start playing the video
        }
        if (PlaceImage != null && GameManager.Instance.loacalPlaceOfInterest.Icon != null)
        {
            PlaceImage.sprite = GameManager.Instance.loacalPlaceOfInterest.Icon; // Set the place icon image
        }


    }
}
```

Παράρτημα 5– Script:GeocodeAddress.cs

Αυτό το script είναι υπεύθυνο για να μετατρέπει μια διεύθυνση (π.χ. "Ακρόπολη, Αθήνα") σε συντεταγμένες GPS (γεωγραφικό πλάτος και μήκος). Ουσιαστικά, «ρωτάει» μια online υπηρεσία (OpenStreetMap / Nominatim) και παίρνει πίσω τα αντίστοιχα αριθμητικά δεδομένα, τα οποία μπορούν μετά να χρησιμοποιηθούν για να εμφανιστεί περιεχόμενο AR σε εκείνη την τοποθεσία.

GeocodeAddress.cs

```
using UnityEngine;
using UnityEngine.Networking;
using System.Collections;
using UnityEditor;

/// <summary>
/// This MonoBehaviour provides geocoding functionality, allowing the application to convert a human-readable address string
/// into precise geographic coordinates (latitude and longitude) using the OpenStreetMap Nominatim API.
/// It is designed to be used both at runtime and in the Unity Editor for data entry and validation.
/// The script demonstrates how to perform asynchronous web requests, parse JSON responses, and integrate third-party geocoding services into Unity.
/// </summary>
public class GeocodeAddress : MonoBehaviour
{
    /// <summary>
    /// The address string to be geocoded. This field is typically set via the Unity Editor or by other scripts.
    /// </summary>
    [HideInInspector]
    public string addressInput = "";
    /// <summary>
    /// The result of the geocoding operation, formatted as a string for display in the UI or editor.
    /// This field is updated after each geocoding request.
    /// </summary>
    [HideInInspector]
    public string resultText = "";

    /// <summary>
    /// The latitude value obtained from the geocoding service.
    /// This value is set after a successful geocoding request and can be used for AR placement or map visualization.
    /// </summary>
    public float latitude;
    /// <summary>
    /// The longitude value obtained from the geocoding service.
    /// This value is set after a successful geocoding request and can be used for AR placement or map visualization.
    /// </summary>
    public float longitude;

    /// <summary>
    /// Initiates the geocoding process for the specified address.
    /// This method starts a coroutine that sends a request to the Nominatim API and processes the response.
    /// </summary>
    /// <param name="address">The address string to be converted into coordinates.</param>
    public void SearchAddress(string address)
    {
        if (!string.IsNullOrEmpty(address))
        {
            StartCoroutine(GetCoordinatesFromAddress(address));
        }
    }

    /// <summary>
    /// Coroutine that sends an HTTP GET request to the OpenStreetMap Nominatim API to geocode the provided address.
    /// It handles network errors, parses the JSON response, and updates the latitude and longitude fields accordingly.
    /// This method demonstrates best practices for asynchronous web requests and JSON parsing in Unity.
    /// </summary>
    /// <param name="address">The address string to geocode.</param>
    /// <returns>An IEnumerator for coroutine execution.</returns>
    IEnumerator GetCoordinatesFromAddress(string address)
    {
        string url = $"https://nominatim.openstreetmap.org/search?q={UnityWebRequest.EscapeURL(address)}&format=json";
        UnityWebRequest request = UnityWebRequest.Get(url);
        request.SetRequestHeader("User-Agent", "UnityGeocodeTest/1.0 (nikosakr99@gmail.com)"); // Replace with your email

        yield return request.SendWebRequest();

#if UNITY_2020_1_OR_NEWER
        if (request.result != UnityWebRequest.Result.Success)
```

GeocodeAddress.cs

```
64
65  #if UNITY_2020_1_OR_NEWER
66          if (request.result != UnityWebRequest.Result.Success)
67  #else
68          if (request.isNetworkError || request.isHttpError)
69  #endif
70          {
71              Debug.LogError("Request error: " + request.error);
72              resultText = "Error: " + request.error;
73          }
74          else
75          {
76              string json = request.downloadHandler.text;
77              GeocodeResult[] results = JsonHelper.FromJson<GeocodeResult>(json);
78
79              if (results != null && results.Length > 0)
80              {
81                  float lat = float.Parse(results[0].lat);
82                  float lon = float.Parse(results[0].lon);
83                  latitude = lat;
84                  longitude = lon;
85                  resultText = $"Lat: {lat}, Lon: {lon}";
86                  Debug.Log($"Lat: {lat}, Lon: {lon}");
87              }
88              else
89              {
90                  resultText = "No results found.";
91                  latitude = 0f;
92                  longitude = 0f;
93              }
94          }
95      }
96
97      /// <summary>
98      /// Represents a single geocoding result returned by the Nominatim API.
99      /// Each result contains latitude and longitude as strings, which are parsed into floats for use in the application.
100     /// </summary>
101     [System.Serializable]
102     public class GeocodeResult
103     {
104         /// <summary>
105         /// The latitude value as a string, as returned by the API.
106         /// </summary>
107         public string lat;
108         /// <summary>
109         /// The longitude value as a string, as returned by the API.
110         /// </summary>
111         public string lon;
112     }
113
114     /// <summary>
115     /// Static helper class for parsing JSON arrays using Unity's JsonUtility.
116     /// Since JsonUtility does not natively support top-level arrays, this class wraps the array in an object for parsing.
117     /// </summary>
118     public static class JsonHelper
119     {
120         /// <summary>
121         /// Parses a JSON array string into an array of objects of type T.
122         /// This method is essential for handling API responses that return arrays at the root level.
123         /// </summary>
124         /// <typeparam name="T">The type of objects in the array.</typeparam>
125         /// <param name="json">The JSON array string to parse.</param>
126         /// <returns>An array of objects of type T.</returns>
127         public static T[] FromJson<T>(string json)
128         {
129             string wrapped = "{\"array\":" + json + "}";
```

```
        /// Parses a JSON array string into an array of objects of type T.
        /// This method is essential for handling API responses that return arrays at the root level.
        /// </summary>
        /// <typeparam name="T">The type of objects in the array.</typeparam>
        /// <param name="json">The JSON array string to parse.</param>
        /// <returns>An array of objects of type T.</returns>
        public static T[] FromJson<T>(string json)
        {
            string wrapped = "{\"array\":" + json + "}";
            Wrapper<T> wrapper = JsonUtility.FromJson<Wrapper<T>>(wrapped);
            return wrapper.array;
        }

        [System.Serializable]
        private class Wrapper<T>
        {
            public T[] array;
        }
    }
}

// Custom Editor for GeocodeAddress
#if UNITY_EDITOR
/// <summary>
/// Custom Unity Editor inspector for the GeocodeAddress MonoBehaviour.
/// This editor extension provides a convenient interface for testing geocoding functionality directly within the Unity Editor.
/// Users can input an address, trigger the geocoding process, and view the resulting coordinates and status messages.
/// This tool is valuable for validating address data and ensuring correct geocoding before runtime.
/// </summary>
[CustomEditor(typeof(GeocodeAddress))]
public class GeocodeAddressEditor : Editor
{
    /// <summary>
    /// Overrides the default inspector GUI to provide a custom interface for geocoding addresses.
    /// Includes input fields for the address, a search button, and a display area for results.
    /// </summary>
    public override void OnInspectorGUI()
    {
        GeocodeAddress script = (GeocodeAddress)target;

        EditorGUILayout.LabelField("Geocode Address Tester", EditorStyles.boldLabel);

        script.addressInput = EditorGUILayout.TextField("Address", script.addressInput);

        if (GUILayout.Button("Search"))
        {
            script.SearchAddress(script.addressInput);
        }

        EditorGUILayout.Space();
        EditorGUILayout.LabelField("Result:");
        EditorGUILayout.HelpBox(script.resultText, MessageType.None);

        if (GUI.changed)
        {
            EditorUtility.SetDirty(target);
        }
    }
}
#endif

```

Παράρτημα 6 – Script:PlaceOfInterestUI.cs

Το συγκεκριμένο script έχει πολύ απλό ρόλο: προβάλλει το όνομα ενός σημείου ενδιαφέροντος στην οθόνη. Παίρνει ένα string (το όνομα) και το εμφανίζει σε ένα TextMeshPro πεδίο. Αν για κάποιο λόγο το TextMeshProUGUI δεν έχει οριστεί, εμφανίζεται σχετικό μήνυμα στο Console.

PlaceOfInterestUI.cs

```
using UnityEngine;
using TMPro;

public class PlaceOfInterestUI : MonoBehaviour
{
    public TextMeshProUGUI PlaceName; // Reference to the TextMeshProUGUI component for the place name

    public void SetPlaceName(string name)
    {
        if (PlaceName != null)
        {
            PlaceName.text = name; // Set the place name text
        }
        else
        {
            Debug.LogWarning("PlaceName TextMeshProUGUI component is not assigned.");
        }
    }
}

```

Παράρτημα 7 – Script: VuforiaManager.cs

Το VuforiaManager.cs είναι υπεύθυνο για την ενοποίηση του συστήματος αναγνώρισης εικόνων της Vuforia με το UI της εφαρμογής. Κάθε φορά που εντοπίζεται ένα AR image target, το script ενημερώνει δυναμικά το interface, εμφανίζοντας ή κρύβοντας την αντίστοιχη πληροφοριακή κάρτα. Έτσι, δημιουργείται μια διαδραστική εμπειρία όπου ο χρήστης λαμβάνει άμεσα πληροφορίες όταν «σκανάρει» ένα σημείο ενδιαφέροντος.

VuforiaManager.cs

```
using UnityEngine;
using Vuforia;

/// <summary>
/// This MonoBehaviour manages the interaction between Vuforia image targets and the application's UI.
/// It listens for Vuforia target detection and loss events, and controls the visibility of a UI panel that displays information about the detected target.
/// The script demonstrates how to bridge AR tracking events with user interface elements, which is a common requirement in AR tourism and educational apps.
/// </summary>
public class VuforiaManager : MonoBehaviour
{
    /// <summary>
    /// Reference to the Vuforia ImageTargetBehaviour component in the scene.
    /// This component is responsible for detecting and tracking a specific image target in the real world.
    /// </summary>
    public ImageTargetBehaviour imageTargetBehaviour;
    /// <summary>
    /// Reference to the GameObject representing the UI panel that displays information about the currently detected target.
    /// This panel is shown or hidden based on the target's tracking status.
    /// </summary>
    public GameObject targetInfoPanel; // Reference to the UI panel that displays target information

    /// <summary>
    /// Holds a reference to the VuforiaTargetEvents component, which dispatches target tracking events.
    /// </summary>
    private VuforiaTargetEvents vuforiaTargetEvents;

    /// <summary>
    /// Unity lifecycle method called on the frame when the script is enabled.
    /// Sets up event subscriptions for Vuforia target found and lost events, and ensures the info panel is initially hidden.
    /// This method demonstrates best practices for initializing AR event handling and UI state.
    /// </summary>
    void Start()
    {

        if (imageTargetBehaviour != null)
        {
            // Add or get the VuforiaTargetEvents component
            vuforiaTargetEvents = imageTargetBehaviour.GetComponent<VuforiaTargetEvents>();
            if (vuforiaTargetEvents == null)
                vuforiaTargetEvents = imageTargetBehaviour.gameObject.AddComponent<VuforiaTargetEvents>();

            // Subscribe to target events
            vuforiaTargetEvents.OnTargetFound += TargetFound;
            vuforiaTargetEvents.OnTargetLost += TargetLost;
            // Optionally subscribe to OnTargetUpdated and OnTargetStatusChanged if needed
        }


        // Ensure the target info panel is initially hidden
        if (targetInfoPanel == null)
        {
            // Try to auto-assign by searching for a GameObject named "TargetInfoPanel"
            var foundPanel = GameObject.Find("TargetInfoPanel");
            if (foundPanel != null)
            {
                targetInfoPanel = foundPanel;
                Debug.LogWarning("Target info panel was not assigned in the inspector. Found and assigned GameObject named 'TargetInfoPanel'.");
            }
        }

        if (targetInfoPanel != null)
        {
            targetInfoPanel.SetActive(false);
        }

    }
```

VuforiaManager.cs

```
64          }
65
66      }
67
68      /// <summary>
69      /// Unity lifecycle method called when the MonoBehaviour will be destroyed.
70      /// Cleans up event subscriptions to prevent memory leaks and unintended behavior.
71      /// </summary>
72      void OnDestroy()
73      {
74          if (vuforiaTargetEvents != null)
75          {
76              vuforiaTargetEvents.OnTargetFound -= TargetFound;
77              vuforiaTargetEvents.OnTargetLost -= TargetLost;
78          }
79      }
80
81      /// <summary>
82      /// Callback method invoked when a Vuforia target is detected and tracked.
83      /// Activates the target information panel and logs the event for debugging or analytics.
84      /// </summary>
85      /// <param name="observer">The ObserverBehaviour representing the detected target.</param>
86      void TargetFound(ObserverBehaviour observer)
87      {
88          // Display the target information panel when a target is found
89          if (targetInfoPanel != null)
90              targetInfoPanel.SetActive(true);
91
92          // Optionally, log or use observer.TargetName if needed
93          Debug.Log("Target found: " + observer.TargetName);
94      }
95
96      /// <summary>
97      /// Callback method invoked when a Vuforia target is lost (no longer tracked).
98      /// Deactivates the target information panel and logs the event for debugging or analytics.
99      /// </summary>
100     /// <param name="observer">The ObserverBehaviour representing the lost target.</param>
101     void TargetLost(ObserverBehaviour observer)
102     {
103         // Hide the target information panel when a target is lost
104         if (targetInfoPanel != null)
105             targetInfoPanel.SetActive(false);
106
107         Debug.Log("Target lost: " + observer.TargetName);
108     }
109 }
110
```

Παράρτημα 8 – Script :VuforiaTargetEvents.cs

Το VuforiaTargetEvents.cs είναι ένα script που αναλαμβάνει να διαχειρίζεται τις αλλαγές κατάστασης (status) των AR targets που παρακολουθεί η Vuforia. Παρακολουθεί τα λεγόμενα ObserverBehaviours (όπως εικόνες ή μοντέλα στον AR χώρο) και ενεργοποιεί αντίστοιχα γεγονότα (C# events) όπως OnTargetFound, OnTargetLost, OnTargetUpdated και OnTargetStatusChanged. Αυτός ο μηχανισμός είναι ιδανικός για να απεμπλακεί η λογική του συστήματος από την τεχνολογία εντοπισμού — επιτρέποντας καθαρό, modular σχεδιασμό. Τα υπόλοιπα scripts απλά κάνουν subscribe στα events και αντιδρούν (π.χ. εμφάνιση UI όταν εμφανιστεί ένα target).

VuforiaTargetEvents.cs

```cs
using UnityEngine;
using Vuforia;
using System;

/// <summary>
/// This MonoBehaviour acts as an event dispatcher for Vuforia target status changes.
/// It listens for changes in the tracking status of Vuforia ObserverBehaviours (such as Image Targets or Model Targets)
/// and exposes these changes as C# events that other scripts can subscribe to.
/// This enables decoupled and modular handling of AR target detection, loss, and updates,
/// which is essential for building interactive AR experiences where UI or logic must respond to target visibility.
/// </summary>
[RequireComponent(typeof(VuforiaManager))]
public class VuforiaTargetEvents : MonoBehaviour
{
    /// <summary>
    /// Event triggered when a Vuforia target is found (i.e., when tracking status transitions from NO_POSE to TRACKED or EXTENDED_TRACKED).
    /// Subscribers can use this event to activate AR content or UI when a target becomes visible.
    /// </summary>
    public event Action<ObserverBehaviour> OnTargetFound;
    /// <summary>
    /// Event triggered when a Vuforia target is lost (i.e., when tracking status transitions from TRACKED/EXTENDED_TRACKED to NO_POSE).
    /// Subscribers can use this event to hide AR content or UI when a target is no longer visible.
    /// </summary>
    public event Action<ObserverBehaviour> OnTargetLost;
    /// <summary>
    /// Event triggered whenever a Vuforia target's status is updated, regardless of the specific transition.
    /// This can be used for continuous feedback or analytics.
    /// </summary>
    public event Action<ObserverBehaviour, TargetStatus> OnTargetUpdated;
    /// <summary>
    /// Event triggered on any change in the target's status, providing both the previous and new status.
    /// This allows for detailed tracking of status transitions for debugging or advanced logic.
    /// </summary>
    public event Action<ObserverBehaviour, TargetStatus, TargetStatus> OnTargetStatusChanged;

    /// <summary>
    /// Stores the previous tracking status of the observed target.
    /// This is used to detect transitions between different tracking states.
    /// </summary>
    private TargetStatus previousStatus;

    /// <summary>
    /// Unity lifecycle method called when the script instance is being loaded.
    /// Subscribes to the Vuforia ObserverBehaviour's OnTargetStatusChanged event to monitor tracking changes.
    /// </summary>
    void Start()
    {
        var observer = FindAnyObjectByType<ObserverBehaviour>();


        previousStatus = observer.TargetStatus;
        observer.OnTargetStatusChanged += HandleTargetStatusChanged;

    }

    /// <summary>
    /// Unity lifecycle method called when the MonoBehaviour will be destroyed.
    /// Unsubscribes from the Vuforia ObserverBehaviour's event to prevent memory leaks or null references.
    /// </summary>
    void OnDestroy()
    {
        var observer = GetComponent<ObserverBehaviour>();
        if (observer != null)
        {
            observer.OnTargetStatusChanged -= HandleTargetStatusChanged;
        }
```

```
60      void OnDestroy()
61      {
62          var observer = GetComponent<ObserverBehaviour>();
63          if (observer != null)
64          {
65              observer.OnTargetStatusChanged -= HandleTargetStatusChanged;
66          }
67      }
68
69      /// <summary>
70      /// Handles the logic for all Vuforia target status transitions.
71      /// Invokes the appropriate C# events based on the change in tracking status, enabling other scripts to react accordingly.
72      /// This method distinguishes between target found, lost, updated, and general status changes.
73      /// </summary>
74      /// <param name="behaviour">The Vuforia ObserverBehaviour whose status has changed.</param>
75      /// <param name="status">The new tracking status of the target.</param>
76      private void HandleTargetStatusChanged(ObserverBehaviour behaviour, TargetStatus status)
77      {
78          // OnTargetStatusChanged (previous, new)
79          OnTargetStatusChanged?.Invoke(behaviour, previousStatus, status);
80
81          // OnTargetFound
82          if ((previousStatus.Status == Status.NO_POSE) &&
83              (status.Status == Status.TRACKED || status.Status == Status.EXTENDED_TRACKED))
84          {
85              OnTargetFound?.Invoke(behaviour);
86          }
87          // OnTargetLost
88          else if ((previousStatus.Status == Status.TRACKED || previousStatus.Status == Status.EXTENDED_TRACKED) &&
89                   status.Status == Status.NO_POSE)
90          {
91              OnTargetLost?.Invoke(behaviour);
92          }
93
94          // OnTargetUpdated
95          OnTargetUpdated?.Invoke(behaviour, status);
96
97          previousStatus = status;
98      }
99  }
```

Παράρτημα 9 – Script: PlaceOfinterest.cs

Η κλάση PlaceOfinterest είναι ένα σειριοποιήσιμο μοντέλο δεδομένων που χρησιμοποιείται για να περιγράψει πλήρως ένα σημείο ενδιαφέροντος σε εφαρμογές επαυξημένης πραγματικότητας. Περιλαμβάνει πληροφορίες όπως το όνομα, την περιγραφή, τη διεύθυνση, τις γεωγραφικές συντεταγμένες (πλάτος, μήκος, ύψος), καθώς και σχετικό οπτικοακουστικό υλικό (εικόνα και βίντεο). Το script αυτό λειτουργεί ως δομικό στοιχείο αποθήκευσης και ανάκτησης δεδομένων τοποθεσιών, διευκολύνοντας τη σύνδεση του περιεχομένου με το UI και τη σκηνή AR.

PlaceOfinterest.cs

```cs
using System;
using UnityEngine;
using UnityEngine.Video;

/// <summary>
/// A serializable data class that encapsulates all relevant information about a single place of interest.
/// This class is intended to represent real-world locations in AR tourism or mapping applications,
/// providing both geospatial coordinates and descriptive metadata for each location.
/// Each instance can be stored in a "PlacesOfinterest" ScriptableObject and referenced at runtime.
/// </summary>
[Serializable]
public class PlaceOfinterest
{
    /// <summary>
    /// The human-readable name of the place of interest.
    /// This is used for display in the UI and as a unique identifier for searching and referencing.
    /// </summary>
    public string Name;
    /// <summary>
    /// A textual description providing additional information about the place.
    /// This can include historical context, significance, or visitor information.
    /// </summary>
    public string Description;
    /// <summary>
    /// The physical or postal address of the place, if available.
    /// This field can be used for geocoding to obtain latitude and longitude coordinates.
    /// </summary>
    public string Address;
    /// <summary>
    /// The latitude coordinate (in decimal degrees) of the place's geographic location.
    /// Used for positioning in AR scenes and proximity calculations.
    /// </summary>
    public float Latitude;
    /// <summary>
    /// The longitude coordinate (in decimal degrees) of the place's geographic location.
    /// Used for positioning in AR scenes and proximity calculations.
    /// </summary>
    public float Longitude;
    /// <summary>
    /// The height or altitude (in meters) above sea level for the place.
    /// This value is used for accurate 3D placement in AR or mapping environments.
    /// </summary>
    public float Height;

    /// <summary>
    /// A visual icon or sprite representing the place of interest.
    /// This can be displayed in the UI or on AR markers to help users identify the location.
    /// </summary>
    public Sprite Icon;

    /// <summary>
    /// A video clip representing the place of interest.
    /// This can be used to show a video about the location in the UI or AR experience.
    /// </summary>
    public VideoClip Video;
}

```

Παράρτημα 10 – Script: PlacesOfinterest.cs

Η κλάση PlacesOfinterest είναι ένα ScriptableObject της Unity που λειτουργεί ως βάση δεδομένων σημείων ενδιαφέροντος. Περιλαμβάνει μια λίστα από αντικείμενα τύπου PlaceOfinterest, τα οποία περιέχουν πληροφορίες για κάθε τοποθεσία (π.χ. όνομα, περιγραφή, συντεταγμένες, εικονίδιο). Παρέχει μεθόδους για προσθήκη, διαγραφή και αναζήτηση σημείων βάσει ονόματος.

PlacesOfinterest.cs

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

/// <summary>
/// A Unity ScriptableObject that serves as a persistent data container for a collection of places of interest.
/// This class is designed to be used as a centralized database for storing and managing multiple "PlaceOfinterest" entries,
/// which represent real-world locations with associated metadata such as name, description, address, coordinates, and an icon.
/// The ScriptableObject can be edited in the Unity Editor and referenced by other scripts at runtime, enabling efficient data-driven workflows
/// for location-based AR or tourism applications.
/// </summary>
[CreateAssetMenu(fileName = "Empty Data", menuName = "Places Of interest")]
public class PlacesOfinterest : ScriptableObject
{
    /// <summary>
    /// The main list containing all "PlaceOfinterest" objects managed by this ScriptableObject.
    /// Each entry in this list represents a unique point of interest with its own metadata and geospatial information.
    /// </summary>
    public List<PlaceOfinterest> placesOfInterest = new List<PlaceOfinterest>();

    /// <summary>
    /// Adds a new "PlaceOfinterest" to the collection if it does not already exist.
    /// This method prevents duplicate entries and ensures data integrity.
    /// </summary>
    /// <param name="place">The PlaceOfinterest instance to add to the list.</param>
    public void AddPlace(PlaceOfinterest place)
    {
        if (place != null && !placesOfInterest.Contains(place))
        {
            placesOfInterest.Add(place);
        }
    }

    /// <summary>
    /// Removes an existing "PlaceOfinterest" from the collection if it is present.
    /// This method is used to delete places from the database, supporting dynamic content management.
    /// </summary>
    /// <param name="place">The PlaceOfinterest instance to remove from the list.</param>
    public void RemovePlace(PlaceOfinterest place)
    {
        if (place != null && placesOfInterest.Contains(place))
        {
            placesOfInterest.Remove(place);
        }
    }

    /// <summary>
    /// Searches for a PlaceOfinterest in the collection by its name.
    /// Returns the first matching entry or null if no match is found.
    /// This method enables efficient lookup of places for editing, navigation, or display purposes.
    /// </summary>
    /// <param name="name">The name of the place to search for.</param>
    /// <returns>The matching PlaceOfinterest instance, or null if not found.</returns>
    public PlaceOfinterest GetPlaceByName(string name)
    {
        return placesOfInterest.Find(place => place.Name == name);
    }
}

```

Παράρτημα 11 – Script: PlacesOfinterestEditor.cs

Το script PlacesOfinterestEditor είναι ένα custom εργαλείο για το Unity Editor, το οποίο επιτρέπει την εύκολη διαχείριση των σημείων ενδιαφέροντος (PlacesOfinterest). Μέσα από το παράθυρο του Inspector, ο χρήστης μπορεί να προσθέτει, τροποποιεί ή διαγράφει σημεία, καθώς και να αποθηκεύει ή να φορτώνει δεδομένα σε μορφή JSON. Διευκολύνει την εισαγωγή πληροφορίας χωρίς αλλαγή σκηνής ή χρήση εξωτερικών εργαλείων.

PlacesOfinterestEditor.cs

```
61      /// Custom GUI style for section headers in the inspector.
62      /// </summary>
63      private GUIStyle headerStyle;
64      /// <summary>
65      /// Custom GUI style for boxed sections in the inspector.
66      /// </summary>
67      private GUIStyle boxStyle;
68      /// <summary>
69      /// Custom GUI style for buttons in the inspector.
70      /// </summary>
71      private GUIStyle buttonStyle;
72      /// <summary>
73      /// Custom GUI style for labels in the inspector.
74      /// </summary>
75      private GUIStyle labelStyle;
76
77      /// <summary>
78      /// Initializes custom GUI styles for the inspector if they have not already been created.
79      /// This method ensures consistent and visually appealing formatting for all custom UI elements.
80      /// </summary>
81      private void InitStyles()
82      {
83          if (headerStyle == null)
84          {
85              headerStyle = new GUIStyle(EditorStyles.boldLabel)
86              {
87                  fontSize = 16,
88                  normal = { textColor = new Color(0.2f, 0.5f, 0.8f) }
89              };
90          }
91          if (boxStyle == null)
92          {
93              boxStyle = new GUIStyle(GUI.skin.box)
94              {
95                  padding = new RectOffset(10, 10, 8, 8),
96                  margin = new RectOffset(0, 0, 5, 5)
97              };
98          }
99          if (buttonStyle == null)
100         {
101             buttonStyle = new GUIStyle(GUI.skin.button)
102             {
103                 fontStyle = FontStyle.Bold,
104                 fontSize = 12,
105                 fixedHeight = 28
106             };
107         }
108         if (labelStyle == null)
109         {
110             labelStyle = new GUIStyle(EditorStyles.label)
111             {
112                 fontSize = 12
113             };
114         }
115     }
116
117     // Foldout and scroll state
118     private bool placesFoldout = true;
119     private Vector2 placesScroll;
120     private List<bool> expandedStates = new List<bool>();
121
122     /// <summary>
123     /// Overrides the default Unity inspector GUI for the "PlacesOfinterest" ScriptableObject.
124     /// Provides a comprehensive interface for viewing, editing, adding, and searching places of interest.
125     /// The inspector displays all existing places in a scrollable list, each with editable fields for all properties.
126     /// Users can add new places by filling out the provided form and clicking the "Add Place" button.
```

PlacesOfinterestEditor.cs

```
    /// The search section allows users to quickly locate a place by name and view its details.
    /// All changes are recorded for undo functionality and marked as dirty for proper asset saving.
    /// </summary>
    public override void OnInspectorGUI()
    {
        InitStyles();
        PlacesOfinterest poiAsset = (PlacesOfinterest)target;

        EditorGUILayout.Space();
        EditorGUILayout.LabelField("📍 Places Of Interest", headerStyle);
        EditorGUILayout.Space();

        // Backup/Restore buttons in a horizontal group, centered
        EditorGUILayout.BeginHorizontal();
        GUILayout.FlexibleSpace();
        if (GUILayout.Button("⬇ Backup to JSON", GUILayout.Width(140), GUILayout.Height(24)))
        {
            string path = EditorUtility.SaveFilePanel("Backup Places Of Interest", "", "places_backup.json", "json");
            if (!string.IsNullOrEmpty(path))
            {
                try
                {
                    string json = JsonUtility.ToJson(new SerializationWrapper<PlaceOfinterest>(poiAsset.placesOfInterest), true);
                    File.WriteAllText(path, json);
                    EditorUtility.DisplayDialog("Backup Complete", "Places backed up to JSON successfully.", "OK");
                }
                catch (Exception ex)
                {
                    EditorUtility.DisplayDialog("Backup Failed", ex.Message, "OK");
                }
            }
        }
        if (GUILayout.Button("⬆ Restore from JSON", GUILayout.Width(140), GUILayout.Height(24)))
        {
            string path = EditorUtility.OpenFilePanel("Restore Places Of Interest", "", "json");
            if (!string.IsNullOrEmpty(path))
            {
                try
                {
                    string json = File.ReadAllText(path);
                    var wrapper = JsonUtility.FromJson<SerializationWrapper<PlaceOfinterest>>(json);
                    if (wrapper != null && wrapper.items != null)
                    {
                        Undo.RecordObject(poiAsset, "Restore Places");
                        poiAsset.placesOfInterest = new List<PlaceOfinterest>(wrapper.items);
                        EditorUtility.SetDirty(poiAsset);
                        expandedStates = new List<bool>(new bool[poiAsset.placesOfInterest.Count]);
                        EditorUtility.DisplayDialog("Restore Complete", "Places restored from JSON successfully.", "OK");
                    }
                }
                catch (Exception ex)
                {
                    EditorUtility.DisplayDialog("Restore Failed", ex.Message, "OK");
                }
            }
        }
        GUILayout.FlexibleSpace();
        EditorGUILayout.EndHorizontal();

        EditorGUILayout.Space();

        // Foldout for places list
        placesFoldout = EditorGUILayout.Foldout(placesFoldout, $"Places List ({poiAsset.placesOfInterest.Count})", true, headerStyle);

        // Expand/Collapse All
        if (placesFoldout && poiAsset.placesOfInterest.Count > 0)
```

PlacesOfinterestEditor.cs

```
190
191         // Expand/Collapse All
192         if (placesFoldout && poiAsset.placesOfInterest.Count > 0)
193         {
194             EditorGUILayout.BeginHorizontal();
195             GUILayout.FlexibleSpace();
196             if (GUILayout.Button("Expand All", GUILayout.Width(90)))
197             {
198                 EnsureExpandedStates(poiAsset.placesOfInterest.Count);
199                 for (int i = 0; i < expandedStates.Count; i++) expandedStates[i] = true;
200             }
201             if (GUILayout.Button("Collapse All", GUILayout.Width(90)))
202             {
203                 EnsureExpandedStates(poiAsset.placesOfInterest.Count);
204                 for (int i = 0; i < expandedStates.Count; i++) expandedStates[i] = false;
205             }
206             GUILayout.FlexibleSpace();
207             EditorGUILayout.EndHorizontal();
208         }
209
210         // List all places in scroll view
211         if (placesFoldout)
212         {
213             if (poiAsset.placesOfInterest.Count == 0)
214             {
215                 EditorGUILayout.HelpBox("No places added yet.", MessageType.Info);
216             }
217             else
218             {
219                 EnsureExpandedStates(poiAsset.placesOfInterest.Count);
220                 placesScroll = EditorGUILayout.BeginScrollView(placesScroll, GUILayout.MinHeight(180), GUILayout.MaxHeight(320));
221                 for (int i = 0; i < poiAsset.placesOfInterest.Count; i++)
222                 {
223                     var place = poiAsset.placesOfInterest[i];
224                     expandedStates[i] = EditorGUILayout.Foldout(expandedStates[i], $"#{i + 1}  {place.Name}", true, EditorStyles.foldoutHeader);
225                     if (expandedStates[i])
226                     {
227                         EditorGUILayout.BeginVertical(boxStyle);
228
229                         // Top row: Remove button, Name, Icon
230                         EditorGUILayout.BeginHorizontal();
231                         if (GUILayout.Button("🗑", GUILayout.Width(28), GUILayout.Height(28)))
232                         {
233                             Undo.RecordObject(poiAsset, "Remove Place");
234                             poiAsset.placesOfInterest.RemoveAt(i);
235                             EditorUtility.SetDirty(poiAsset);
236                             EditorGUILayout.EndHorizontal();
237                             EditorGUILayout.EndVertical();
238                             expandedStates.RemoveAt(i);
239                             i--;
240                             continue;
241                         }
242                         EditorGUILayout.LabelField("Name", GUILayout.Width(38));
243                         place.Name = EditorGUILayout.TextField(place.Name, GUILayout.Width(120));
244                         GUILayout.FlexibleSpace();
245                         place.Icon = (Sprite)EditorGUILayout.ObjectField(place.Icon, typeof(Sprite), false, GUILayout.Width(60), GUILayout.Height(60));
246                         EditorGUILayout.EndHorizontal();
247
248                         // Video
249                         place.Video = (VideoClip)EditorGUILayout.ObjectField("Video", place.Video, typeof(VideoClip), false);
250
251                         // Description
252                         EditorGUILayout.LabelField("Description", labelStyle);
253                         place.Description = EditorGUILayout.TextArea(place.Description, GUILayout.MinHeight(80), GUILayout.MaxHeight(120));
254
255                         // Address
```

PlacesOfinterestEditor.cs

```
                    place.Description = EditorGUILayout.TextArea(place.Description, GUILayout.MinHeight(80), GUILayout.MaxHeight(120));

                    // Address
                    place.Address = EditorGUILayout.TextField("Address", place.Address);

                    // Lat/Lon/Height row
                    EditorGUILayout.BeginHorizontal();
                    EditorGUILayout.LabelField("Lat", GUILayout.Width(24));
                    EditorGUILayout.SelectableLabel(place.Latitude.ToString("F6"), EditorStyles.textField, GUILayout.Width(70));
                    if (GUILayout.Button("Copy", GUILayout.Width(36)))
                        EditorGUIUtility.systemCopyBuffer = place.Latitude.ToString("F6");
                    EditorGUILayout.LabelField("Lon", GUILayout.Width(24));
                    EditorGUILayout.SelectableLabel(place.Longitude.ToString("F6"), EditorStyles.textField, GUILayout.Width(70));
                    if (GUILayout.Button("Copy", GUILayout.Width(36)))
                        EditorGUIUtility.systemCopyBuffer = place.Longitude.ToString("F6");
                    EditorGUILayout.LabelField("H", GUILayout.Width(14));
                    place.Height = EditorGUILayout.FloatField(place.Height, GUILayout.Width(48));
                    EditorGUILayout.EndHorizontal();

                    EditorGUILayout.EndVertical();
                    EditorGUILayout.Space(2);
                }
            }
            EditorGUILayout.EndScrollView();
        }
    }

    // Add new place section
    EditorGUILayout.Space(8);
    EditorGUILayout.LabelField("+ Add New Place", headerStyle);
    EditorGUILayout.BeginVertical(boxStyle);

    EditorGUILayout.BeginHorizontal();
    EditorGUILayout.LabelField("Name", GUILayout.Width(38));
    newPlaceName = EditorGUILayout.TextField(newPlaceName, GUILayout.Width(120));
    GUILayout.FlexibleSpace();
    newPlaceIcon = (Sprite)EditorGUILayout.ObjectField(newPlaceIcon, typeof(Sprite), false, GUILayout.Width(60), GUILayout.Height(60));
    EditorGUILayout.EndHorizontal();

    // Video
    newPlaceVideo = (VideoClip)EditorGUILayout.ObjectField("Video", newPlaceVideo, typeof(VideoClip), false);

    EditorGUILayout.LabelField("Description", labelStyle);
    newPlaceDescription = EditorGUILayout.TextArea(newPlaceDescription, GUILayout.MinHeight(80), GUILayout.MaxHeight(120));
    newPlaceAddress = EditorGUILayout.TextField("Address", newPlaceAddress);

    EditorGUILayout.BeginHorizontal();
    EditorGUILayout.LabelField("Lat", GUILayout.Width(24));
    newPlaceLatitude = EditorGUILayout.FloatField(newPlaceLatitude, GUILayout.Width(70));
    EditorGUILayout.LabelField("Lon", GUILayout.Width(24));
    newPlaceLongitude = EditorGUILayout.FloatField(newPlaceLongitude, GUILayout.Width(70));
    EditorGUILayout.LabelField("H", GUILayout.Width(14));
    newPlaceHeight = EditorGUILayout.FloatField(newPlaceHeight, GUILayout.Width(48));
    EditorGUILayout.EndHorizontal();

    GUI.backgroundColor = new Color(0.5f, 0.85f, 0.5f);
    if (GUILayout.Button("Add Place", GUILayout.Height(28)))
    {
        var newPlace = new PlaceOfinterest
        {
            Name = newPlaceName,
            Description = newPlaceDescription,
            Address = newPlaceAddress,
            Latitude = newPlaceLatitude,
            Longitude = newPlaceLongitude,
            Height = newPlaceHeight,
```

PlacesOfinterestEditor.cs

```
307
308         GUI.backgroundColor = new Color(0.5f, 0.85f, 0.5f);
309         if (GUILayout.Button("Add Place", GUILayout.Height(28)))
310         {
311             var newPlace = new PlaceOfinterest
312             {
313                 Name = newPlaceName,
314                 Description = newPlaceDescription,
315                 Address = newPlaceAddress,
316                 Latitude = newPlaceLatitude,
317                 Longitude = newPlaceLongitude,
318                 Height = newPlaceHeight,
319                 Icon = newPlaceIcon,
320                 Video = newPlaceVideo
321             };
322             Undo.RecordObject(poiAsset, "Add Place");
323             poiAsset.AddPlace(newPlace);
324             EditorUtility.SetDirty(poiAsset);
325             newPlaceName = "";
326             newPlaceDescription = "";
327             newPlaceAddress = "";
328             newPlaceLatitude = 0f;
329             newPlaceLongitude = 0f;
330             newPlaceHeight = 0f;
331             newPlaceIcon = null;
332             newPlaceVideo = null;
333             expandedStates.Add(true);
334         }
335         GUI.backgroundColor = Color.white;
336         EditorGUILayout.EndVertical();
337
338         // Search section
339         EditorGUILayout.Space(8);
340         EditorGUILayout.LabelField("🔎 Find Place By Name", headerStyle);
341         EditorGUILayout.BeginVertical(boxStyle);
342         searchName = EditorGUILayout.TextField("Search Name", searchName);
343         if (GUILayout.Button("Find", GUILayout.Height(24)))
344         {
345             foundPlace = poiAsset.GetPlaceByName(searchName);
346         }
347         if (foundPlace != null)
348         {
349             EditorGUILayout.HelpBox(
350                 $"Found: {foundPlace.Name}\nDescription: {foundPlace.Description}", MessageType.Info);
351
352             EditorGUILayout.BeginHorizontal();
353             EditorGUILayout.LabelField("Lat", GUILayout.Width(24));
354             EditorGUILayout.SelectableLabel(foundPlace.Latitude.ToString("F6"), EditorStyles.textField, GUILayout.Width(70));
355             if (GUILayout.Button("Copy", GUILayout.Width(36)))
356                 EditorGUIUtility.systemCopyBuffer = foundPlace.Latitude.ToString("F6");
357             EditorGUILayout.LabelField("Lon", GUILayout.Width(24));
358             EditorGUILayout.SelectableLabel(foundPlace.Longitude.ToString("F6"), EditorStyles.textField, GUILayout.Width(70));
359             if (GUILayout.Button("Copy", GUILayout.Width(36)))
360                 EditorGUIUtility.systemCopyBuffer = foundPlace.Longitude.ToString("F6");
361             EditorGUILayout.LabelField("H", GUILayout.Width(14));
362             EditorGUILayout.SelectableLabel(foundPlace.Height.ToString("F2"), EditorStyles.textField, GUILayout.Width(48));
363             EditorGUILayout.EndHorizontal();
364         }
365         EditorGUILayout.EndVertical();
366
367         EditorGUILayout.Space();
368
369         if (GUI.changed)
370         {
371             EditorUtility.SetDirty(poiAsset);
372         }
```

```
            EditorGUILayout.EndVertical();

            EditorGUILayout.Space();

            if (GUI.changed)
            {
                EditorUtility.SetDirty(poiAsset);
            }
        }

        // Ensure expandedStates matches the number of places
        private void EnsureExpandedStates(int count)
        {
            while (expandedStates.Count < count) expandedStates.Add(true);
            while (expandedStates.Count > count) expandedStates.RemoveAt(expandedStates.Count - 1);
        }
    }

    // Helper for serializing lists with JsonUtility
    [Serializable]
    public class SerializationWrapper<T>
    {
        public List<T> items;
        public SerializationWrapper(List<T> items) { this.items = items; }
    }
```