ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Σχεδίαση και ανάπτυξη διαδραστικής εφαρμογής επαυξημένης πραγματικότητας (AR) σε περιβάλλον κινητών συσκευών Android (για την ξενάγηση στην πόλη της Καστοριάς), με αξιοποίηση τεχνολογιών επαυξημένης πραγματικότητας και τρισδιάστατων απεικονίσεων.

Πτυχιακή Εργασία

του

Μπαμπούλη Δημήτριου

(Α.Μ. 2927)

Επιβλέπων: ***Νικολάου Σπυρίδων***

Λέκτορας

Καστοριά, Σεπτέμβριος 2024

ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Σχεδίαση και ανάπτυξη διαδραστικής εφαρμογής επαυξημένης πραγματικότητας (AR) σε περιβάλλον κινητών συσκευών Android (για την ξενάγηση στην πόλη της Καστοριάς), με αξιοποίηση τεχνολογιών επαυξημένης πραγματικότητας και τρισδιάστατων απεικονίσεων.

Πτυχιακή Εργασία

του

Μπαμπούλη Δημήτριου

(Α.Μ. 2927)

Επιβλέπων: ***Νικολάου Σπυρίδων***

Λέκτορας

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 3η Οκτωβρίου 2024

.....	.....	.....
Βέργαδος Δημήτριος	Δημόκας Νικόλαος	Νικολάου Σπυρίδων
Αναπληρωτής Καθηγητής	Επίκουρος Καθηγητής	Λέκτορας

Καστοριά, Σεπτέμβριος 2024

Ευχαριστίες

Ευχαριστώ όλους τους καθηγητές του Τμήματος Πληροφορικής του Πανεπιστημίου Δυτικής Μακεδονίας για τις γνώσεις που έχω λάβει κατά τη διάρκεια των σπουδών μου και ιδιαίτερα τον κύριο Σπυρίδων Νικολάου για την εποπτεία της παρούσας πτυχιακής εργασίας.

Περίληψη

Η συγκεκριμένη πτυχιακή εργασία έχει ως θέμα την ανάπτυξη μίας εφαρμογής επαυξημένης πραγματικότητας (Augmented Reality - AR) με τίτλο "Kastoria3D", για έξυπνες κινητές συσκευές σε περιβάλλον Android, για την ενίσχυση της εμπειρίας τουριστικής περιήγησης στα αξιοθέατα και τα μουσεία της πόλης της Καστοριάς.

Για τη δημιουργία της συγκεκριμένης εφαρμογής χρησιμοποιήθηκε η μηχανή ανάπτυξης παιχνιδιών Unity καθώς και το εργαλείο επαυξημένης πραγματικότητας "AR Foundation" το οποίο είναι ενσωματωμένο σε αυτήν. Χρησιμοποιήθηκε επίσης, το εργαλείο Mapbox για την ενσωμάτωση του χάρτη πλοήγησης σε πραγματικό χρόνο, με βάση την τοποθεσία του χρήστη. Τέλος, το περιβάλλον ανάπτυξης της εφαρμογής είναι το Android SDK και το Microsoft Visual Studio.

Λέξεις Κλειδιά: Επαυξημένη Πραγματικότητα, Εικονική Πραγματικότητα, Μεικτή Πραγματικότητα, Kastoria3D, Unity Game Engine, AR Foundation, Mapbox, SDK, API, Microsoft Visual Studio, App, C#

Abstract

This thesis focuses on the development of an augmented reality (AR) application entitled "Kastoria3D", for smart mobile devices based on Android OS, to enhance the experience of tourist sightseeing and museums in the city of Kastoria.

For the creation of this application, the Unity game development engine was used as well as the integrated AR Foundation tool. The Mapbox tool was also used to integrate the real-time navigation map based on the user's location. Finally, the development environment of the application is the Android SDK and Microsoft Visual Studio.

Key Words: Augmented Reality, Virtual Reality, Mixed Reality, Kastoria3D, Unity Game Engine, AR Foundation, Mapbox, SDK, API, Microsoft Visual Studio, App, C#

Πίνακας Περιεχομένων

Λίστα Εικόνων

ΕΙΣΑΓΩΓΗ

Η παρούσα πτυχιακή εργασία έχει ως θέμα την ανάπτυξη μίας εφαρμογής επαυξημένης και μεικτής πραγματικότητας για έξυπνες κινητές συσκευές σε περιβάλλον Android, με σκοπό την ενίσχυση της εμπειρίας τουριστικής περιήγησης στα αξιοθέατα και μουσεία της πόλης της Καστοριάς.

Η εργασία αποτελείται από τρία κεφάλαια, στα οποία καλύπτεται και το θεωρητικό υπόβαθρο, αλλά και το πρακτικό μέρος για τη δημιουργία μίας εφαρμογής που περιέχει λειτουργίες επαυξημένης και μεικτής πραγματικότητας.

Στο πρώτο κεφάλαιο γίνονται αναφορές για το τί είναι το android και ποιά η αρχιτεκτονική του, όπως και για το android SDK αλλά και το API framework.

Στο δεύτερο κεφάλαιο γίνεται αναφορά στην επαυξημένη πραγματικότητα και στην ιστορική εξέλιξη της, Επίσης αναφέρεται και το τι είναι η μεικτή πραγματικότητα ,δίνονται μερικές κατηγορίες τέτοιων εφαρμογών και μερικά παραδείγματα απο εφαρμογές επαυξημένης και μεικτής πραγματικότητας στην τουριστική περιήγηση,επίσης αναφέρονται και περιγράγονται μερικά απο τα λογισμικά που χρησιμοποιούνται για την δημιουργία εφαρμογών επαυξημένης και μεικτής πραγματικότητας.

Στο τρίτο κεφάλαιο παρουσιάζεται ο κύκλος ζωής ανάπτυξης λογισμικού (SDLC), η δομή και το διάγραμμα ροής της εφαρμογής Kastoria3D, καθώς επίσης, αναλύονται τα στάδια ανάπτυξης της εφαρμογής μαζί με μια συνοπτική περιγραφή της.

Στη συγκεκριμένη εφαρμογή ο χρήστης, μέσω της έξυπνης κινητής συσκευής του και με την αξιοποίηση τεχνολογιών επαυξημένης πραγματικότητας, καθοδηγείται σε πραγματικό χρόνο προς το επιλεγμένο αξιοθέατο ή μουσείο της πόλης της Καστοριάς και φτάνοντας σε αυτό, ανακαλύπτει περαιτέρω πληροφορίες.

Στο τέλος, αναφέρονται τα συμπεράσματα που προκύπτουν από την εκπόνηση της παρούσας πτυχιακής εργασίας και παρατίθεται η σχετική βιβλιογραφία και το παράρτημα με τα κυριότερα τμήματα κώδικα της εφαρμογής.

1. Λειτουργικό Σύστημα Android

Τι είναι το Android

Το 2003 δημιουργήθηκε η εταιρία Android [1] η οποία δημιούργησε ένα δικό της λειτουργικό σύστημα για τις ψηφιακές κάμερες και το 2004 έφτασε και για τις έξυπνες κινητές συσκευές.Το 2005 η εταιρία αγοράστηκε απο την Google. Η ανάπτυξη λογισμικού για το Android γίνεται με την χρήση βιβλιοθηκών ανεπτυγμένων από την Google που βασίζονται στην γλώσσα προγραμματισμού Java. Οι συσκευές στις οποίες συναντάται συχνότερα το Android είναι έξυπνα κινητά τηλέφωνα, τάμπλετ, τηλεοράσεις, αυτοκίνητα, ρολόγια χειρός, κονσόλες παιχνιδιών, ψηφιακές φωτογραφικές μηχανές, ηλεκτρονικοί υπολογιστές καθώς και σε πολλές άλλες συσκευές.

Εικόνα 1. Λογότυπο Android

Πηγή: https://1000logos.net/android-logo/

1.2. Αρχιτεκτονική Android

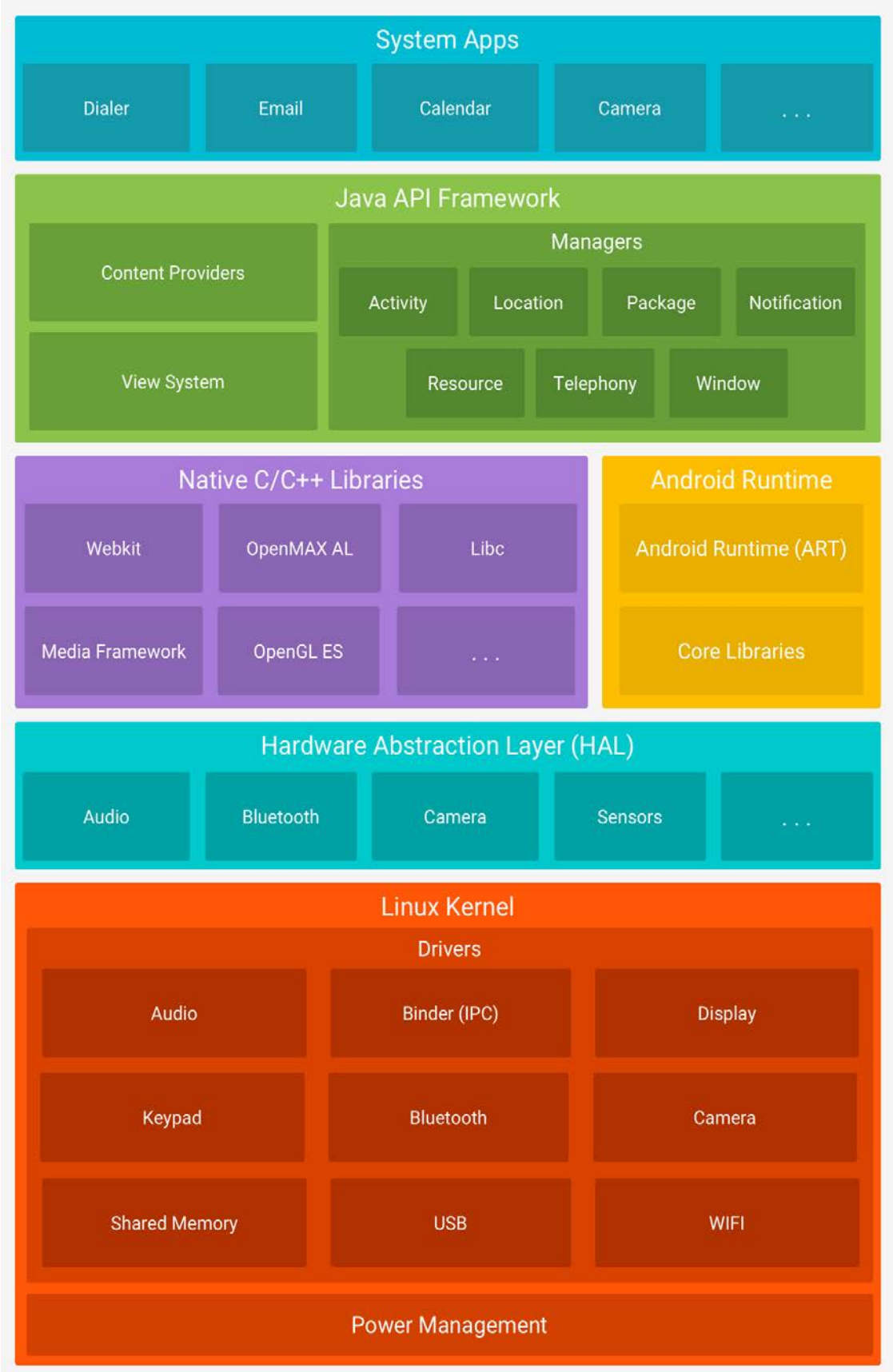
Η αρχιτεκτονική του Android [2] περιλαμβάνει τα εξής επίπεδα, αρχίζοντας από το ψηλότερο και πηγαίνοντας στο χαμηλότερο:

- **Επίπεδο Εφαρμογών (Applications):** Το Android είναι εξαρχής εφοδιασμένο με μια σειρά απο εφαρμογές που χρησιμοποιούνται για τις βασικές λειτουργίες και αυτές αποτελούνται απο ένα email client, ένα πρόγραμμα για SMS μηνύματα, ημερολόγιο, χάρτες (Google Maps), περιηγητή ιστού, πρόγραμμα για δομημένη αποθήκευση των επαφών και άλλα. Όλες οι εφαρμογές είναι γραμμένες στην γλώσσα προγραμματισμού Java.
- **Επίπεδο Πλαισίου Εφαρμογών (Applications Framework):** Παρέχοντας μια ανοικτή πλατφόρμα ανάπτυξης, το Android προσφέρει στην κοινότητα των προγραμματιστών δυνατότητα να κατασκευάσουν πλούσιες και καινοτόμες εφαρμογές. Οι ίδιοι έχουν την ελευθερία να εκμεταλλευτούν πλήρως το hardware της συσκευής, να έχουν πρόσβαση σε υπηρεσίες εντοπισμού

θέσης, να τρέξουν υπηρεσίες και εφαρμογές στο background, να θέσουν χρονοδιακόπτες για εμφάνιση ειδοποιήσεων και πολλά άλλα. Ακόμα έχουν πλήρη πρόσβαση στο ίδιο πλαίσιο από APIs που έχουν οι βασικές εφαρμογές του Android. Η αρχιτεκτονική του είναι διαμορφωμένη έτσι που κάθε εφαρμογή μπορεί να χρησιμοποιήσει τις δυνατότητες μιας άλλης και επίσης δομημένη με τέτοιο τρόπο που δίνει την δυνατότητα στον χρήστη να αλλάξει τα συστατικά κάθε εφαρμογής. Κάτω από το πλαίσιο των εφαρμογών υπάρχει ένα σύστημα από υπηρεσίες και συστήματα τα οποία περιλαμβάνουν:

- Ένα σύνολο από γραφικά στοιχεία (Views) για την δημιουργία γραφικού περιβάλλοντος συμπεριλαμβανομένων λιστών (lists), πλεγμάτων (grids), κουτιών κειμένου (text boxes), κουμπιών (buttons) και άλλων.
- Ένα διαχειριστή περιεχομένου (Content Manager) ο οποίος επιτρέπει στις εφαρμογές να αποκτούν πρόσβαση σε δεδομένα και υπηρεσίες άλλων εφαρμογών ή τον διαμοιρασμό των δικών τους δεδομένων με άλλες εφαρμογές.
- Ένα διαχειριστή πόρων (Resource Manager) για την πρόσβαση και την διαχείριση των πόρων όπως strings, εικόνες, layout files.
- Έναν διαχειριστή ειδοποιήσεων (Notification Manager) ο οποίος επιτρέπει την προβολή ειδοποιήσεων στην μπάρα κατάστασης (status bar).
- Έναν διαχειριστή δραστηριοτήτων (Activity Manager) ο οποίος διαχειρίζεται τον κύκλο ζωής των εφαρμογών.

- **Επίπεδο Βιβλιοθηκών (Libraries):** Το οποίο περιλαμβάνει ένα σύνολο από βιβλιοθήκες γραμμένες σε C/C++ οι οποίες χρησιμοποιούνται από διάφορα στοιχεία του συστήματος του Android. Οι βιβλιοθήκες αυτές καθώς και οι δυνατότητες οι οποίες προσφέρουν στους προγραμματιστές είναι προσβάσιμες σε αυτούς μέσω του επιπέδου πλαισίου εφαρμογής.
- **Επίπεδο Εκτέλεσης (Android Runtime):** Το οποίο αποτελείται από ένα σύνολο από βασικές βιβλιοθήκες και την Dalvik Virtual Machine.
- **Πυρήνας του Linux:** Το Android βασίζεται στον πυρήνα Linux έκδοση 2.6 έτσι ώστε να μπορεί να υποστηρίζει τις βασικές υπηρεσίες συστήματος όπως η ασφάλεια, η διαχείριση μνήμης, η διαχείριση διεργασιών, η στοίβα δικτύου, και ο οδηγούς συσκευών. Τέλος, ο πυρήνας λειτουργεί και ως ενδιάμεσος και υπεύθυνος διασύνδεσης μεταξύ του υλικού και του λογισμικού.



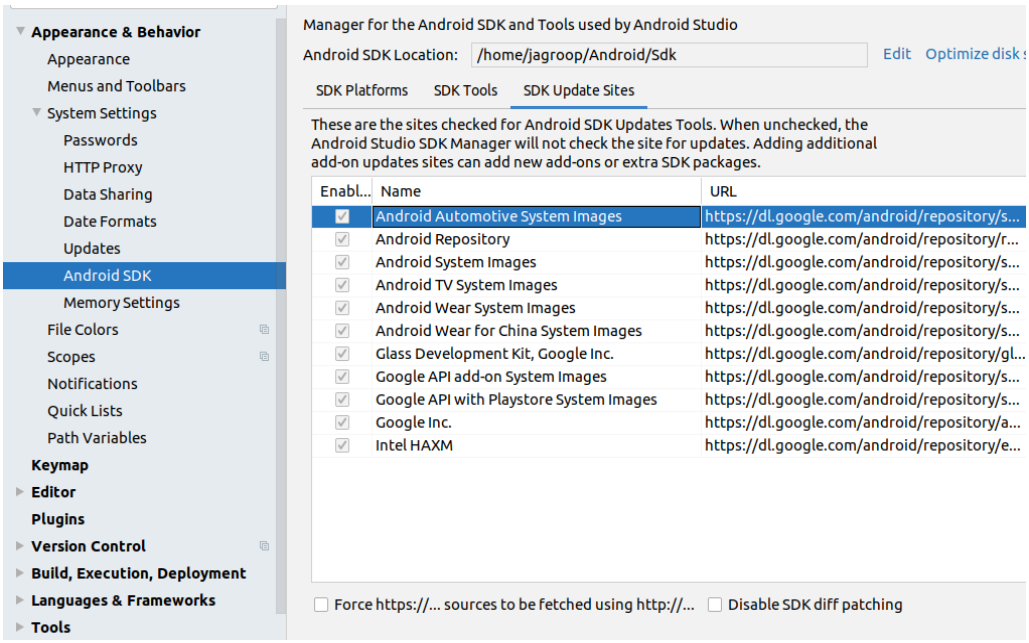
Εικόνα 2.Android Architecture

Πηγή: https://developer.android.com/guide/platform

1.3. Android SDK

Το κιτ ανάπτυξης λογισμικού Android, γνωστό και ως SDK [3], αποτελεί μια ολοκληρωμένη συλλογή εργαλείων ανάπτυξης λογισμικού. Αναλυτικά περιλαμβάνει: έναν debugger (εργαλείο αποσφαλμάτωσης), διάφορες libraries (βιβλιοθήκες), έναν emulator (προσομοιωτή) βασισμένο στο QEM, πλήθος βιβλιογραφίας, δείγματα κώδικα, καθώς και μαθήματα. Η ανάπτυξη εφαρμογών Android (με τη χρήση του Android SDK) γίνεται κατά κύριο λόγο με το Android Studio, το οποίο δημιουργήθηκε από την Google με την υποστήριξη της IntelliJ και αποτελεί το επίσημο IDE για ανάπτυξη λογισμικού Android. Βέβαια, η χρήση του Android Studio δεν είναι υποχρεωτική, καθώς υπάρχει η δυνατότητα χρήσης άλλων IDE όπως το Eclipse IDE και το NetBeans IDE, τα οποία με την εγκατάσταση ορισμένων πρόσθετων Plugins μπορούν να υποστηρίξουν την ανάπτυξη Android εφαρμογών. Τέλος μπορούν να χρησιμοποιηθούν οι διάφοροι επεξεργαστές κειμένου (όπως το Notepad ή το Notepad ++) για να γραφούν τα αρχεία JAVA και XML, τα οποία έπειτα μπορούν μέσω διαφόρων εργαλείων εντολών γραμμής (command line tools) και πάντοτε σε συνδυασμό με το JDK και το Apache Ant, να κατασκευάσουν (build) και να αποσφαλματώσουν (debug) εφαρμογές Android.

Το κιτ ανάπτυξης λογισμικού Android, υποστηρίζει και παλαιότερες εκδόσεις Android, για περιπτώσεις στις οποίες κάποια εφαρμογή στοχεύει σε παλαιότερες συσκευές. Όλες οι εφαρμογές Android βρίσκονται συμπιεσμένες σε μορφή .apk, περιλαμβάνουν αρχεία κατάληξης .dex και είναι τοποθετημένες στον φάκελο /data/app στο λειτουργικο σύστημα του Android. Επίσης αξίζει να σημειωθεί πως αυτοί οι φάκελοι είναι για λόγους ασφαλείας προσβάσιμοι μόνο από τον χρήστη root.



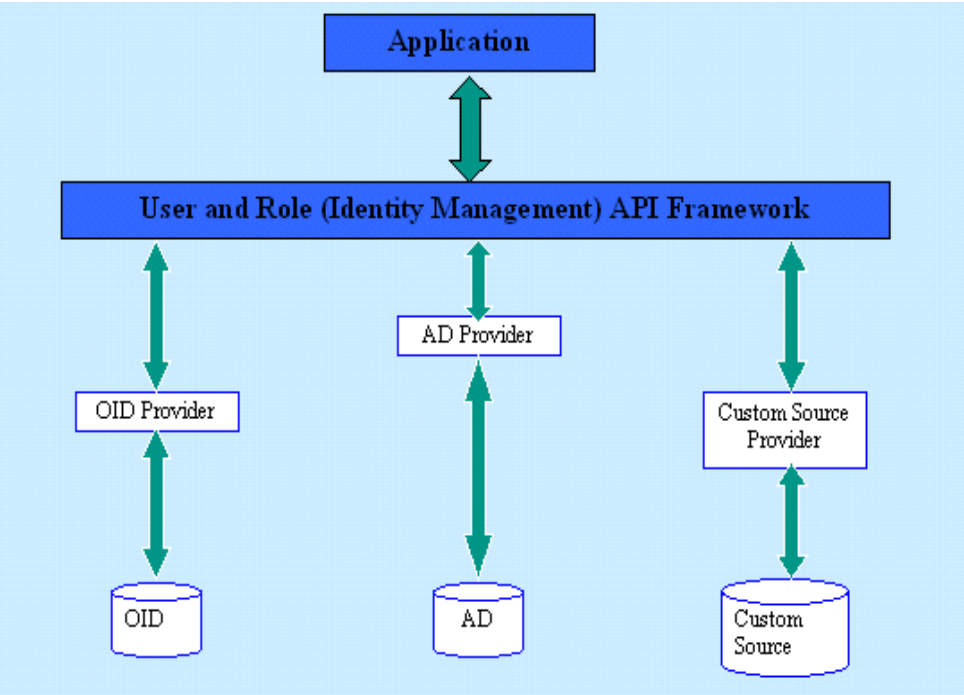
Εικόνα 3.Android SDK

Πηγή: https://www.geeksforgeeks.org/android-sdk-and-its-components/

1.4. API Framework

Το λειτουργικό σύστημα Android με το σύνολο των δυνατοτήτων του είναι διαθέσιμο στον προγραμματιστή με τα APIs [4], τα οποία βρίσκονται γραμμένα στην προγραμματιστική γλώσσα Java και αποτελούν σύνολα κώδικα που απλοποιούν την επαναχρησιμοποίηση του επεξεργαστή, των διαφόρων αισθητήρων και εξαρτημάτων, καθώς και υπηρεσιών των συσκευών. Μέσω των APIs δίνεται πρόσβαση στα ακόλουθα:

- Στο View System, το οποίο χρησιμοποιείται για την κατασκευή του γραφικού περιβάλλοντος (UI) των εφαρμογών,περιλαμβάνοντας λίστες, πλέγματα, κουτιά κειμένου, κουμπιά, αλλά και έναν ενσωματοποιημένο περιηγητή ιστού.
- Στον Resource Manager, παρέχοντας πρόσβαση σε πόρους που δεν σχετίζονται με κώδικα, όπως τοπικές συμβολοσειρές, γραφικά ή αρχεία διάταξης.
- Στο Notification Manager, επιτρέποντας τις εφαρμογές να εμφανίζουν ειδοποιήσεις στην συσκευή.
- Στον Activity Manager, ο οποίος είναι υπεύθυνος της διαχείρισης του κύκλου ζωής των εφαρμογών.
- Στους Content Providers, υπεύθυνοι λήψης δεδομένων από άλλες εφαρμογές ή αποστολής δεδομένων προς αυτές.

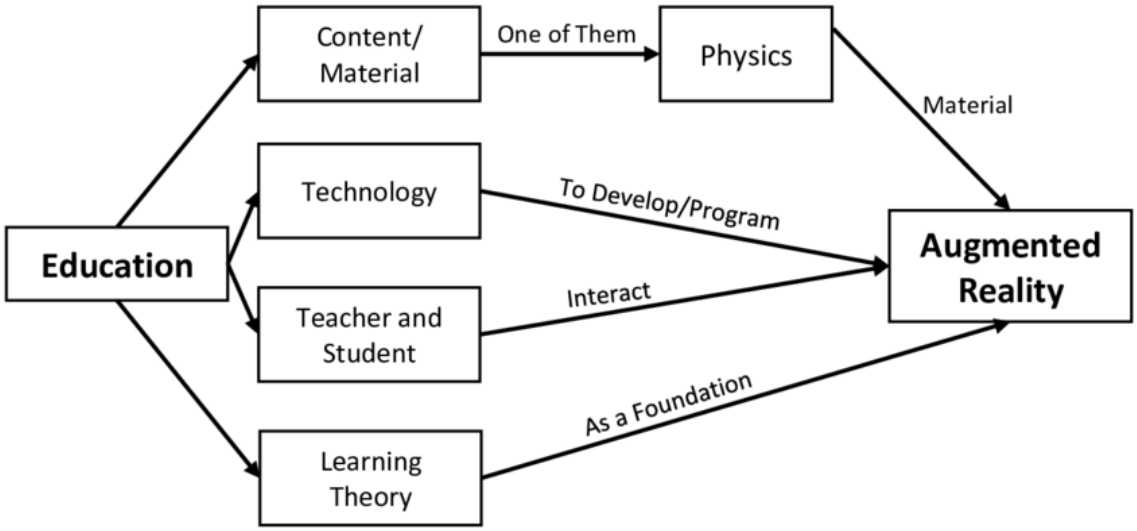


Εικόνα 4. API Framework

Πηγή: https://docs.oracle.com/cd/E16439_01/doc.1013/e13977/userrole.htm

2. Επαυξημένη & Μεικτή πραγματικότηταΕπαυξημένη πραγματικότητα

Επαυξημένη Πραγματικότητα, γνωστή και ως Augmented Reality ή AR [5] , είναι η τεχνολογία που μας επιτρέπει να βλέπουμε ταυτόχρονα ζωντανές εικόνες από την πραγματική ζωή σε συνδυασμό με ψηφιακές προσθήκες

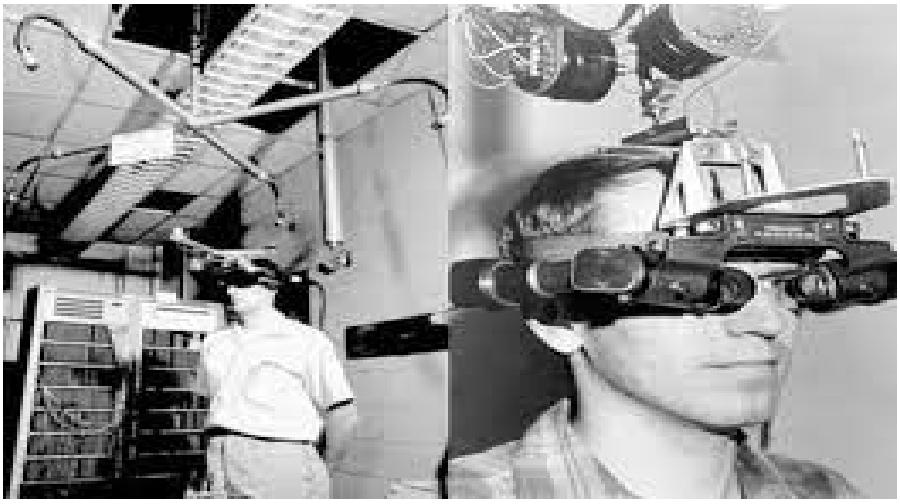


Εικόνα 5. Θεωρία επαυξημένης πραγματικότητας

Πηγή: https://www.researchgate.net/figure/Theoretical-Framework-of-AR-in-Education-and-Physics-Education_fig1_367327827

2.2. Εξέλιξη Επαυξημένης πραγματικότητας

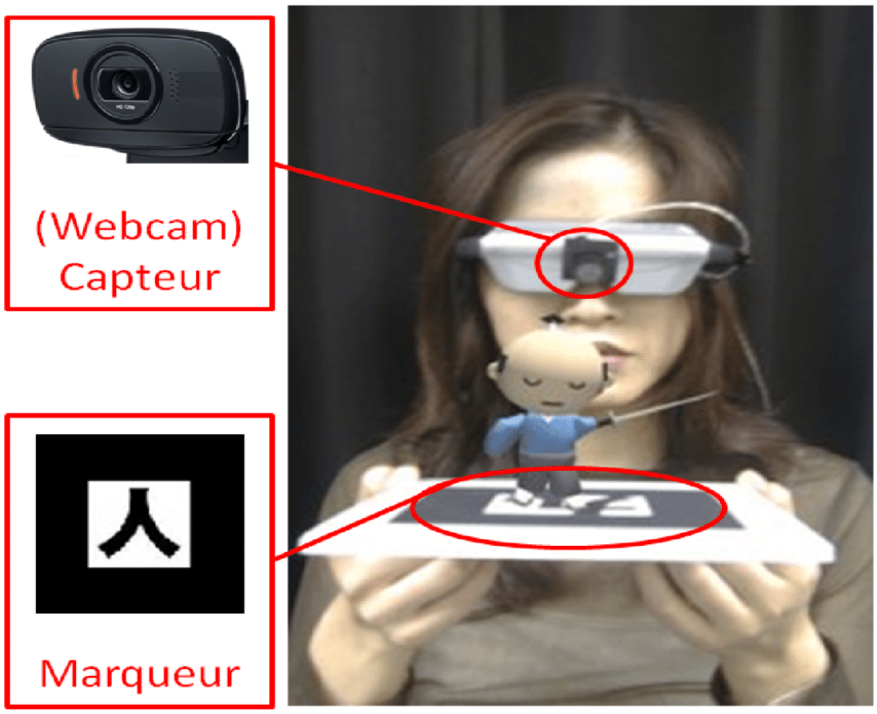
Το 1968, ένας καθηγητής του Χάρβαρντ και επιστήμονας πληροφορικής με το όνομα Ιβάν Σάδερλαντ εφηύρε το πρώτο είδος συσκευής επαυξημένης πραγματικότητας με τον μαθητή του, τον Bob Sproull, το οποίο ονόμασε «Το Ξίφος Του Δαμοκλή» [6]. Το Ξίφος Του Δαμοκλή είχε μια οθόνη προβολής πάνω του που κρεμόταν από την οροφή. Ο χρήστης βίωνε με αυτήν γραφικά ηλεκτρονικού υπολογιστή που τον έκανε να νιώθει σαν να ήταν σε μια εναλλακτική πραγματικότητα. Η συσκευή ήταν πρωτόγονη τόσο από την άποψη της διεπαφής του χρήστη όσο και από το ρεαλισμό, το σύστημα Sutherland εμφάνιζε έξοδο από ένα πρόγραμμα υπολογιστή σε μια στερεοσκοπική οθόνη. Η οπτική του προγράμματος που έδειχνε στον χρήστη θα εξαρτιόταν από τη θέση που έβλεπε ο χρήστης - γι 'αυτό ήταν απαραίτητη η παρακολούθηση των κινήσεων του κεφαλιού.



Εικόνα 6. Εφαρμογή «Το ξίφος του Δαμοκλή»

Πηγή: https://www.dsource.in/course/virtual-reality-introduction/evolution-vr/sword-damocles-head-mounted-display

Το 2000 ο Hirokazu Kato δημιούργησε το ARToolKit [7] μια εφαρμογή ανοιχτού κώδικα. Η εφαρμογή χρησιμοποιεί δυνατότητες παρακολούθησης βίντεο που υπολογίζουν την πραγματική θέση και τον προσανατολισμό της κάμερας σε σχέση με τους τετραγωνικούς φυσικούς δείκτες ή τους φυσικούς δείκτες χαρακτηριστικών σε πραγματικό χρόνο. Μόλις γίνει γνωστή η πραγματική θέση της κάμερας, μπορεί να τοποθετηθεί μια εικονική κάμερα στο ίδιο σημείο και 3D γραφικά σχεδιάζονται ακριβώς πάνω από τον πραγματικό δείκτη.



Εικόνα 7. Εφαρμογή ARToolkit

Πηγή: https://www.researchgate.net/figure/La-technologie-ARToolkit_fig9_281357543

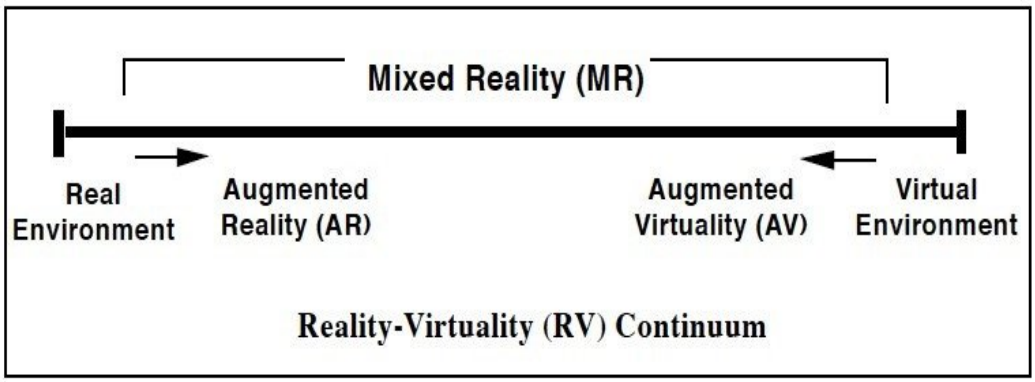
Στις 12 Οκτωβρίου 2016, η Microsoft ανήγγειλε την παγκόσμια κυκλοφορία των HoloLens [8] Τα HoloLens είναι μια απο τις σύγχρονες εφαρμογές επαυξημένης πραγματικότητας βασιζόμενη στο Head Mounted Display δηλαδή προσαρμόζεται στο κεφάλι και συνδέεται με μια ρυθμιζόμενη εσωτερική ζώνη κεφαλή,με την εφαρμογή αυτή ο χρήστης, μπορεί να αντιληφθεί και να εντοπίσει έναν ήχο, σαν να έρχεται από μια εικονική εντολή ή τοποθεσία. Όπως και να λαμβάνει διάφορες πληροφορίες σχετικά με το πραγματικό αντικείμενο που κοιτάζει.

Εικόνα 8. Εφαρμογή Microsoft HoloLens

Πηγή: https://www.technologyrecord.com/article/embracing-the-new-reality-created-by-microsoft-hololens-2

2.3. Μεικτή πραγματικότητα

Πρώτη αναφορά στον όρο Μεικτή Πραγματικότητα ή Mixed Reality (MR) [9] έγινε σε μια επιστημονική εφημερίδα του 1994 από τους Paul Milgram και Fumio Kishino στο άρθρο τους με τίτλο «A Taxonomy of Mixed Reality Visual Displays» [10] ή σε ελεύθερη μετάφραση «Μια κατηγοριοποίηση των οπτικών επιδείξεων της μεικτής πραγματικότητας».



Εικόνα 9. Θεωρία μεικτής πραγματικότητας (MR)

Πηγή: https://www.researchgate.net/figure/Milgram-and-Kishinos-Mixed-Reality-on-the-Reality-Virtuality-Continuum-Milgram-and_fig1_321405854

Σε αυτή τους την αναφορά όρισαν την τεχνολογία αυτή ως το πάντρεμα του φυσικού με τον ψηφιακό κόσμο, ακόμα προσπάθησαν να αναλύσουν και να κατηγοριοποιήσουν τις εφαρμογές της, που τότε περιορίζονταν μονάχα στην οθόνη του ηλεκτρονικού υπολογιστή ως μοναδικό μέσο προβολής της. Σήμερα εντοπίζουμε εφαρμογές της Μεικτής Πραγματικότητας μακράν πέρα της οθόνης, για παράδειγμα περιβαλλοντικές εισόδους, χωρικό ήχο ή ταυτόχρονη μετακίνηση στους δύο κόσμους (δηλαδή του φυσικού και του ψηφιακού).

2.4. Κατηγορίες εφαρμογών επαυξημένης & μεικτής πραγματικότητας

Υπάρχουν πολλοί τομείς της καθημερινής ζωής όπου χρησιμοποιούνται ποικίλες εφαρμογές Επαυξημένης, Εικονικής ή Μεικτής Πραγματικότητας [11], λόγω των διαφορετικών χαρακτηριστικών και δυνατοτήτων που παρέχουν. Πιο συγκεκριμένα, χρησιμοποιούνται ευρέως από διάφορους τομείς και βιομηχανίες [12] όπως:

1) Εκπαίδευση και Κατάρτιση: Η εκπαίδευση είναι ένας από τους σημαντικότερους τομείς που θα επωφεληθεί από τις τεχνολογίας AR και VR. Η Επαυξημένη Πραγματικότητα παρέχει μια εξαιρετική εμπειρία τόσο για τους μαθητές όσο και για τους εκπαιδευτικούς, που μπορούν να επωφεληθούν από αυτή την τεχνολογία για την ανάπτυξη νέων δεξιοτήτων. Μπορεί να χρησιμοποιηθεί για να επιτρέψει στους εκπαιδευόμενους να αλληλεπιδράσουν με ασφάλεια με πράγματα στα οποία δεν θα είχαν πρόσβαση διαφορετικά, ενώ παραμένουν σε ένα οικείο περιβάλλον, είτε πρόκειται για μια τάξη είτε για εκπαίδευση στην εργασία. Επιπλέον, λόγω της δυνατότητας της επαυξημένης πραγματικότητας να επηρεάσει πολλές πτυχές της εκπαίδευσης, οι εκπαιδευτικοί έχουν ήδη αρχίσει να την χρησιμοποιούν σε αίθουσες διδασκαλίας παγκοσμίως και αναμένεται ότι αυτή η τάση θα συνεχιστεί. Στο επαγγελματικό τοπίο, οι εργοδότες μπορούν να χρησιμοποιήσουν επαυξημένες και εικονικές πραγματικότητες για να βελτιώσουν τις ευκαιρίες εισαγωγής και κατάρτισης των εργαζομένων. Ένα AR Headset μπορεί να καθοδηγήσει έναν εργαζόμενο κατά τη διαδικασία εξέτασης ενός εξοπλισμού, διδάσκοντάς τον για κάθε στοιχείο καθώς προχωράει. Περιβάλλοντα εικονικής πραγματικότητας μπορούν να διδάξουν στους υπαλλήλους πώς να αντιμετωπίζουν μια αγχωτική εμπειρία πελάτη ή να ολοκληρώνουν ορισμένες εργασίες.

2) Υγειονομική Περίθαλψη: Ο τομέας της υγειονομικής περίθαλψης μπορεί να μεταμορφωθεί δραστικά με τη βοήθεια της τεχνολογίας AR και της VR. Η Επαυξημένη Πραγματικότητα έχει πλήθος χρήσεων σε αυτόν τον τομέα, συμπεριλαμβανομένης της υποβοήθησης των γιατρών στην εκτέλεση χειρουργικών επεμβάσεων, στην ιατρική εκπαίδευση, στην περίθαλψη, και πολλά άλλα. Μέσω των κατάλληλων εφαρμογών, οι χειρουργοί και οι εξειδικευμένοι ειδικοί μπορούν να εξασκούν πολύπλοκες διαδικασίες χωρίς να διακινδυνεύουν ακριβούς πόρους ή την άνεση των ασθενών. Οι φοιτητές που

πρέπει να μάθουν πώς λειτουργούν οι διαδικασίες μπορούν επίσης να παρακολουθήσουν μέσω AR/VR Headsets έναν χειρουργό να εκτελεί μια χειρουργική επέμβαση σε πραγματικό χρόνο – γεγονός το οποίο τους παρέχει πολύ περισσότερες λεπτομέρειες για την όλη διαδικασία. Η εικονική και επαυξημένη πραγματικότητα στην υγειονομική περίθαλψη έχει επίσης αντίκτυπο στο είδος της υποστήριξης των ασθενών που μπορούν να προσφέρουν οι γιατροί και το νοσηλευτικό προσωπικό. Στην εποχή της τηλεϊατρικής, όπου οι ασθενείς πρέπει να αλληλεπιδρούν με τους ειδικούς από απόσταση, η VR και η AR ξεπερνούν τις τηλεδιασκέψεις. Οι γιατροί μπορούν να δουν τους ασθενείς τους και να διαγνώσουν προβλήματα από απόσταση. Οι νοσηλευτές μπορούν να διδάξουν στους ασθενείς πώς να εκτελούν δραστηριότητες αυτοφροντίδας χρησιμοποιώντας επικαλύψεις και γραφικά επαυξημένης πραγματικότητας.

Ωστόσο, η Επαυξημένη Πραγματικότητα διαφέρει από τον πιο γνωστό "συγγενή" της, την Εικονική Πραγματικότητα (VR), καθώς η τελευταία δημιουργεί έναν τρισδιάστατο κόσμο αποσυνδέοντας πλήρως τον χρήστη από την πραγματικότητα μεταφέροντάς τον μέσω κλειστών VR Headsets, ενώ η πρώτη, η Επαυξημένη Πραγματικότητα, διατηρεί τον χρήστη στην τρέχουσα ύπαρξή του και απλώς την επαυξάνει με εικονικές πληροφορίες. Με έναν απλούστερο τρόπο, η Επαυξημένη Πραγματικότητα φέρνει μια νέα διάσταση στην υγειονομική περίθαλψη και την ιατρική. Υπάρχουν δύο πτυχές στις οποίες η Επαυξημένη Πραγματικότητα είναι μοναδική: οι χρήστες δεν χάνουν την επαφή με την πραγματικότητα και μεταφέρει τις πληροφορίες το συντομότερο δυνατό. Αυτά τα διακριτικά χαρακτηριστικά επιτρέπουν στην Επαυξημένη Πραγματικότητα να γίνει κινητήρια δύναμη στο μέλλον της ιατρικής. Στην περίπτωση της Επαυξημένης Πραγματικότητας, η χρήση της τεχνολογίας στην ιατρική και την υγειονομική περίθαλψη αποτελεί φυσική συνέπεια της τεχνολογικής ανάπτυξης και της έκρηξης των δεδομένων.

Εικόνα 10. Εφαρμογή AR και MR στην υγεία

Πηγή: https://medcitynews.com/2019/09/the-benefits-of-ar-in-healthcare/

3) **Κατασκευαστικός τομέας**: Η επαυξημένη πραγματικότητα είναι ένα κρίσιμο εργαλείο στον κατασκευαστικό κλάδο από το στάδιο του σχεδιασμού μέχρι την πραγματική κατασκευαστική διαδικασία. Η AR και η VR μπορούν να κάνουν τον κλάδο αυτό πιο αποτελεσματικό και παραγωγικό. Οι αρχιτέκτονες, οι μηχανικοί και οι εργολάβοι κατασκευαστικών έργων είναι ενθουσιασμένοι με τις δυνατότητες της Επαυξημένης Πραγματικότητας για σκοπούς ψηφιακής μοντελοποίησης και πολλά άλλα. Πολλά αρχιτεκτονικά εργαλεία βοηθούν στην οπτικοποίηση του χώρου, επιτρέποντας στους σχεδιαστές να δημιουργούν πολύ πιο εύκολα τρισδιάστατα μοντέλα σπιτιών, κτιρίων ή έργων υποδομής, τόσο στην εικονική όσο και στην επαυξημένη πραγματικότητα. Στην κατασκευαστική πλευρά της εξίσωσης, η τεχνολογία AR προσφέρει εφαρμογές που κυμαίνονται από την εκπαίδευση των εργαζομένων σχετικά με την ασφάλεια έως την τεχνολογία καταγραφής και παρακολούθησης της προόδου που συγκρίνει άμεσα τα πραγματικά εργοτάξια με τα εικονικά μοντέλα σε πραγματικό χρόνο για να διασφαλίσει ότι δεν αποκλίνουν.

Εικόνα 11. Εφαρμογή AR και MR στον κατασκευαστικό τομέα

Πηγή: https://archicgi.com/architecture/augmented-reality-apps-architecture/

4) **Κτηματομεσιτική αγορά ακινήτων (Real-Estate)**: Ο τομέας των αγοραπωλησιών ακινήτων (Real Estate) βρίσκεται στη λίστα των κορυφαίων τομέων που είναι πιθανότερο να προσελκύσουν τις μεγαλύτερες επενδύσεις σε AR και VR. Με τη βοήθεια της Επαυξημένης Πραγματικότητας ενισχύεται η εμπειρία περιήγησης του υποψήφιου αγοραστή στο φυσικό χώρο του ακινήτου, ενώ ή Εικονική Πραγματικότητα μπορεί ακόμη και παρέχει σε έναν υποψήφιο αγοραστή μια εικονική περιήγηση σε ένα ακίνητο χωρίς να απαιτείται να το επισκεφθεί αυτοπροσώπως. Επίσης, με τη χρήση εφαρμογών Επαυξημένης

Πραγματικότητας όπως ο κατάλογος AR της IKEA, οι πελάτες που αγοράζουν σπίτια είναι σε θέση να μετρήσουν το εμβαδόν του σπιτιού, να επιλέξουν τα κατάλληλα έπιπλα και οικιακά αντικείμενα χωρίς καν να τα έχουν αγοράσει.

5) **Βιομηχανία Ηλεκτρονικών Παιχνιδιών**: Μια άλλη αγαπημένη χρήση της Επαυξημένης Πραγματικότητας είναι τα ηλεκτρονικά παιχνίδια. Το AR gaming ενσωματώνει το παιχνίδι με το περιβάλλον του χρήστη σε πραγματικό χρόνο. Πριν από μερικά χρόνια, το Pokemon Go ήταν μια τεράστια επιτυχία, με πάνω από 250 εκατομμύρια παίκτες το μήνα να μεταφέρονται σε πραγματικές τοποθεσίες από τη σειρά βιντεοπαιχνιδιών. Αυτό έδειξε τις δυνατότητες της AR στους τομείς των παιχνιδιών και της ψυχαγωγίας. Ιδιαίτερα, η επαυξημένη πραγματικότητα έχει βρει μια φυσική στέγη στον τομέα των παιχνιδιών, όπου η τεχνολογία έχει τροφοδοτήσει αρκετές τεράστιες επιτυχίες παιχνιδιών για κινητά.

6) **Αυτοκινητοβιομηχανία & Μεταφορές**: Η Επαυξημένη Πραγματικότητα στην αυτοκινητοβιομηχανία βοηθά στη βελτίωση της οδηγικής εμπειρίας όσον αφορά τις πληροφορίες για την κυκλοφορία, τη συντομότερη διαδρομή, τα τυφλά σημεία και πολλά άλλα. Χρησιμοποιείται επίσης εκτενώς σε επιδείξεις προϊόντων και εκπαιδευτικές ενότητες, όπως επικαλύψεις εικόνων AR κάτω από το καπό ενός αυτοκινήτου για την εκπαίδευση στην επισκευή του κινητήρα. Για παράδειγμα, η Audi δημιούργησε μια εφαρμογή AR που επιτρέπει στους χρήστες να βλέπουν αυτοκίνητα σχεδόν οπουδήποτε και να δημιουργούν εξατομικευμένες πίστες δοκιμών που δείχνουν πώς μπορεί να λειτουργεί το όχημα. Η διαδικασία αγοράς αυτοκινήτου γίνεται αμέσως πολύ πιο καθηλωτική όταν οι πελάτες έχουν VR και AR για να πειραματιστούν.

7) **Τουριστική βιομηχανία**: Η τεχνολογία AR μπορεί να κάνει τα ταξίδια πιο ευχάριστα. Ξεκινώντας από την επιλογή ενός ξενοδοχείου μέχρι την κράτηση δωματίου και τον προγραμματισμό εκδρομών, η Επαυξημένη Πραγματικότητα βοηθά την τουριστική βιομηχανία με αμέτρητους τρόπους. Αναπτύσσεται για να προσφέρει στους πελάτες αξέχαστες και χωρίς προβλήματα εμπειρίες κατά τον προγραμματισμό των διακοπών τους. Για παράδειγμα, τα ξενοδοχεία μπορούν να προσφέρουν στους πελάτες εικονικές περιηγήσεις στα δωμάτια πριν από την επίσκεψή τους. Οι αεροπορικές εταιρείες θα μπορούσαν να δώσουν στους πελάτες την ευκαιρία να χρησιμοποιήσουν ένα σετ ακουστικών VR που τους κάνει να αισθάνονται ότι βρίσκονται στον προορισμό τους, ενώ βρίσκονται ακόμα στη μέση του ταξιδιού. Επιπλέον, οι λειτουργίες που βασίζονται στην τοποθεσία επιτρέπουν την παροχή μιας καλύτερης εμπειρίας που είναι προσαρμοσμένη στα ενδιαφέροντα και τις απαιτήσεις του χρήστη. Υπάρχουν επίσης μεμονωμένες τοποθεσίες που έχουν πειραματιστεί με την επαυξημένη πραγματικότητα για την καθοδήγηση των επισκεπτών.

8) **Βιομηχανία Παραγωγής, Μεταποίησης και Εφοδιαστική Αλυσίδα**: Η Επαυξημένη Πραγματικότητα στη μεταποιητική βιομηχανία χρησιμοποιείται κυρίως για τη συντήρηση και την υποστήριξη. Για παράδειγμα, η AR βοηθά στην ταχύτερη συναρμολόγηση εξελιγμένων μηχανών με ακρίβεια, γεγονός που συμβάλλει στην εξοικονόμηση κόστους και στις ταχύτερες παραδόσεις. Οι επαγγελματίες της παραγωγής και της εφοδιαστικής μπορούν να χρησιμοποιήσουν ολογραφικές εικόνες για να βελτιώσουν την παραγωγικότητα και την απόδοση. Επίσης, μπορεί να χρησιμοποιηθεί για την εκπαίδευση νέων επαγγελματιών της μεταποίησης πώς να χρησιμοποιούν εξειδικευμένα μηχανήματα με την επικάλυψη οδηγιών μέσω AR Headsets, καθώς εργάζονται. Στον τομέα των logistics και της εφοδιαστικής αλυσίδας η χρήση της Επαυξημένης Πραγματικότητας θα μπορούσε να κάνει θαύματα όταν πρόκειται για την ενίσχυση της αποδοτικότητας των εργαζομένων και στην καλύτερη διαχείριση αποθεμάτων.

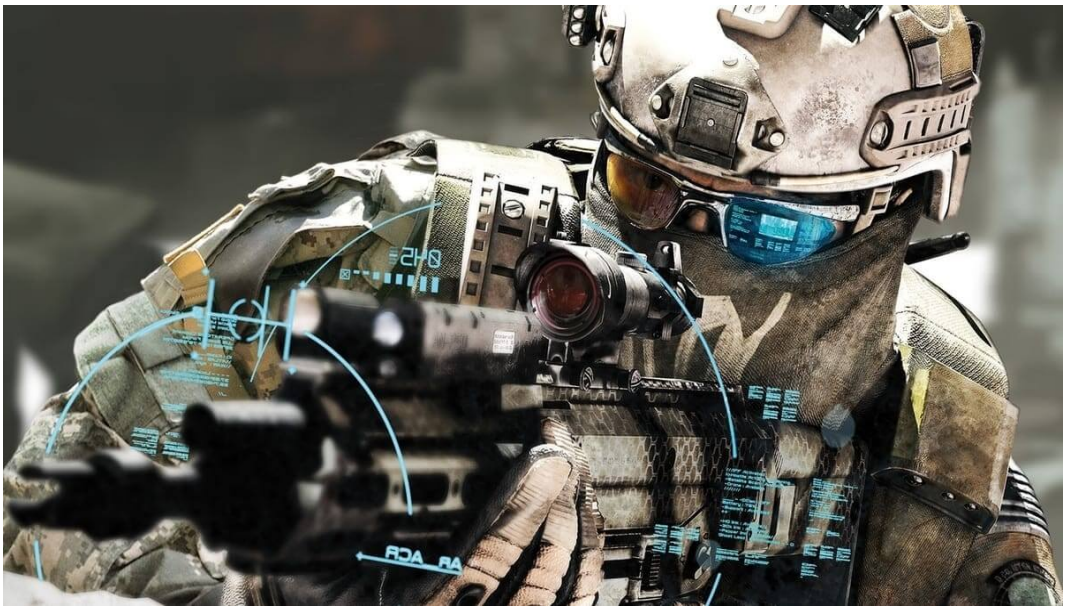
9) **Λιανικό Εμπόριο και Ηλεκτρονικό Εμπόριο**: Ο τομέας του λιανικού εμπορίου προσφέρει μια από τις πιο καινοτόμες και ενδιαφέρουσες εφαρμογές για την Επαυξημένη Πραγματικότητα. Οι λιανοπωλητές, τόσο online όσο και offline, μπορούν να αξιοποιήσουν τις εφαρμογές επαυξημένης πραγματικότητας προς όφελός τους για να αλληλεπιδρούν με τους πελάτες πιο αποτελεσματικά. Και τώρα που η τεχνολογία είναι ευρύτερα διαθέσιμη και σε πιο λογικές τιμές για όλες τις επιχειρήσεις, όλοι μπορούν να τη χρησιμοποιήσουν. Η ζήτηση για αυτές τις λύσεις αυξήθηκε ακόμη περισσότερο από το 2020 με την άφιξη της παγκόσμιας πανδημίας Covid. Με τα καταστήματα να μην μπορούν να ανοίξουν, οι εταιρείες έπρεπε να αναζητήσουν νέους τρόπους για να προσφέρουν νέους σύγχρονους τρόπους αγορών στους καταναλωτές, όπως η μέτρηση αντικειμένων χρησιμοποιώντας τις κάμερες των έξυπνων smartphone ή tablets ή να φανταστούν τα έπιπλα στα δωμάτιό τους, προτού τα αγοράσουν. Βοηθά τους καταναλωτές να βλέπουν και να επιλέγουν τα προϊόντα που είναι διαθέσιμα στο κατάστημα και να αναζητούν γρήγορα πληροφορίες για τα προϊόντα. Η εικονική δοκιμή με τη χρήση επαυξημένης πραγματικότητας ενθαρρύνει πολλούς περισσότερους πελάτες να κάνουν αγορές απ' ό,τι είχαν προγραμματίσει.

   Πολλές διαφορετικές δοκιμαστικές εφαρμογές έχουν αποκτήσει δημοτικότητα, όπως η δοκιμαστική εφαρμογή γυαλιών με βάση την επαυξημένη πραγματικότητα, η δοκιμαστική εφαρμογή μακιγιάζ, η δοκιμαστική εφαρμογή κοσμημάτων, η δοκιμαστική εφαρμογή επίπλων, η δοκιμαστική εφαρμογή ηλεκτρικών συσκευών και η δοκιμαστική εφαρμογή αξεσουάρ αυτοκινήτου, μεταξύ άλλων. Χρησιμοποιείται επίσης σε οδηγίες χρήσης προϊόντων, προωθητικές ενέργειες, μεθόδους προετοιμασίας κ.λπ.

   Επίσης, μπορεί να προσφέρει μια εξατομικευμένη εμπειρία στους καταναλωτές, βελτιώνοντας έτσι την ευκολία εξυπηρέτησής τους. Για παράδειγμα, στη βιομηχανία μόδας η Επαυξημένη Πραγματικότητα μπορεί να χρησιμοποιηθεί ως

εικονικός καθρέφτης που βοηθά τους καταναλωτές να επιλέγουν το επιθυμητό ντύσιμο από το σπίτι και να ψωνίζουν σε περιβάλλον εικονικής πραγματικότητας, με εικονικά δοκιμαστήρια που τους επιτρέπουν να δοκιμάζουν τα ρούχα τους και τα αξεσουάρ της επιλογής τους, για να δουν πώς μπορεί να φαίνονται.

10) **Στρατιωτικός τομέας**: Οι τεχνολογίες τόσο της επαυξημένης όσο και της εικονικής πραγματικότητας, μπορεί να χρησιμοποιηθούν για εκπαιδευτικούς σκοπούς, καθώς και για πραγματικές επιχειρήσεις στο πεδίο της μάχης. Η μεικτή πραγματικότητα μπορεί να χρησιμοποιηθεί για την εκπαίδευση στρατιωτών στη χρήση όπλων, τακτικών και διαδικασιών χωρίς να τους θέτει σε κίνδυνο. Αυτό συμβάλλει στη μείωση των τραυματισμών, ενώ παράλληλα αυξάνει την αποτελεσματικότητα, επιτρέποντας στους στρατιώτες να αποκτήσουν τις απαραίτητες δεξιότητες με τον δικό τους ρυθμό. Στην πραγματικότητα, τα στρατεύματα του αμερικανικού στρατού χρησιμοποιούν AR headsets κατά τη διάρκεια ασκήσεων εκπαίδευσης μάχης τουλάχιστον από το 2017 - και οι συσκευές αυτές αναμένεται να γίνουν ακόμη πιο διαδεδομένες τα επόμενα χρόνια.



Εικόνα 12. Εφαρμογή AR και MR στον στρατό

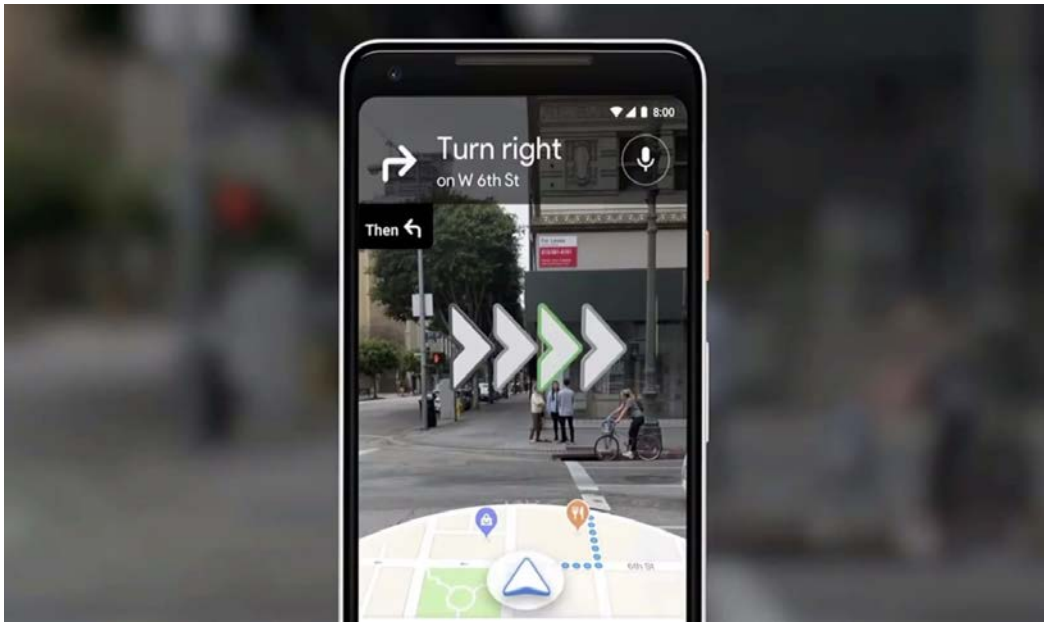
Πηγή: https://www.jasoren.com/augmented-reality-military/

2.5. Διαδεδομένες εφαρμογές επαυξημένης και μεικτής πραγματικότητας για τουριστική περιήγηση

2.5.1. Google Maps Augmented Reality

Η συγκεκριμένη εφαρμογή δίνει την δυνατότητα στον χρήστη να μεταφερθεί στην τοποθεσία που θέλει χρησιμοποιώντας τον χάρτη της google και με την

βοήθεια τρισδιάστατων γραφικών αλλά και φωνητικών εντολών κάνει την περιήγησή του πιο ευχάριστη. Μπορείτε να την βρείτε στο διαδίκτυο αλλά και στο playstore για android συσκευές, όπως και στο app store για συσκευές με λειτουργικό σύστημα IOS



Εικόνα 13. Google maps AR

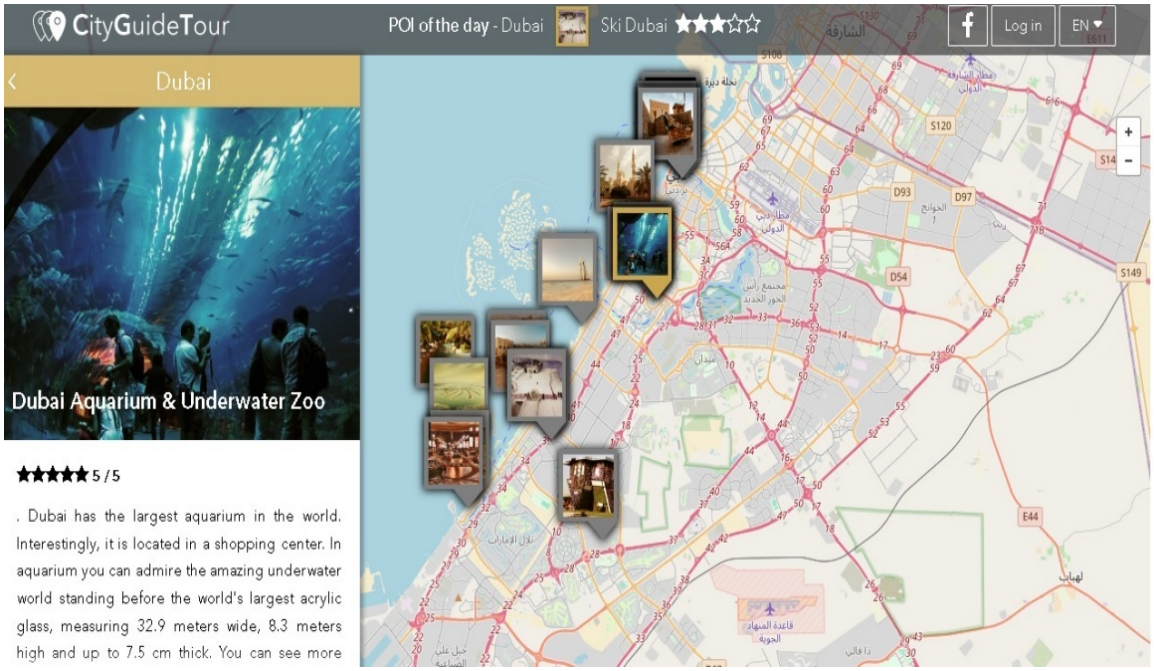
Πηγή: https://www.cnbc.com/2019/03/15/google-maps-ar-first-look-at-augmented-reality-directions.html

Εικόνα 14. Google maps AR

Πηγή: https://edition.cnn.com/2019/02/11/tech/google-maps-ar/index.html

2.5.2. City Guide Tour

Διαδικτυακή εφαρμογή που μέσω χάρτη εμφανίζει είκονες πάνω σε αυτόν για τις συγκεκριμένες τοποθεσίες και πατώντας πάνω σε αυτές τις εικόνες εμφανίζονται είκονες και πληροφορίες για το συγκεκριμένο σημείο.Η συγκεκριμένη εφαρμογές περιέχει πολλές χώρες και πόλεις για τουριστική περιήγηση.



Εικόνα 15. City Guide Tour

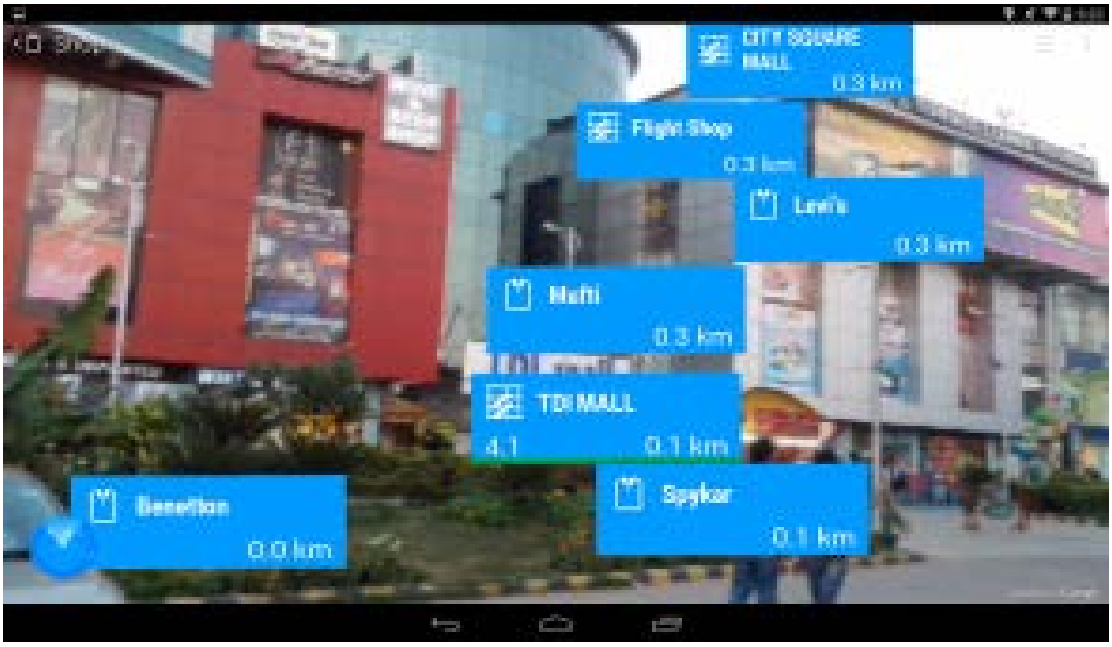
Πηγή: https://www.cityguidetour.com/dubai/dubai-aquarium-underwater-zoo

2.5.3. World around me

Μία εφαρμογή για συσκευές android και IOS στην οποία ο χρήστης δίνει στην εφαρμογή την προτίμησή του για σημεία όπως εστιατόρια,βεζινάδικα,μουσεία ή αξιοθέτα κλπ. Και έχοντας ανοιχτεί την κάμερα του και κάνοντας περιήγηση σε οποιοδήποτε μέρος του εμφανίζεται στην οθόνη το πόσο κοντά βρίσκεται σε μέρη των προτιμήσεών του.

Εικόνα 16. World around me logo

Πηγή: https://appadvice.com/app/wam-pro-world-around-me/638846539

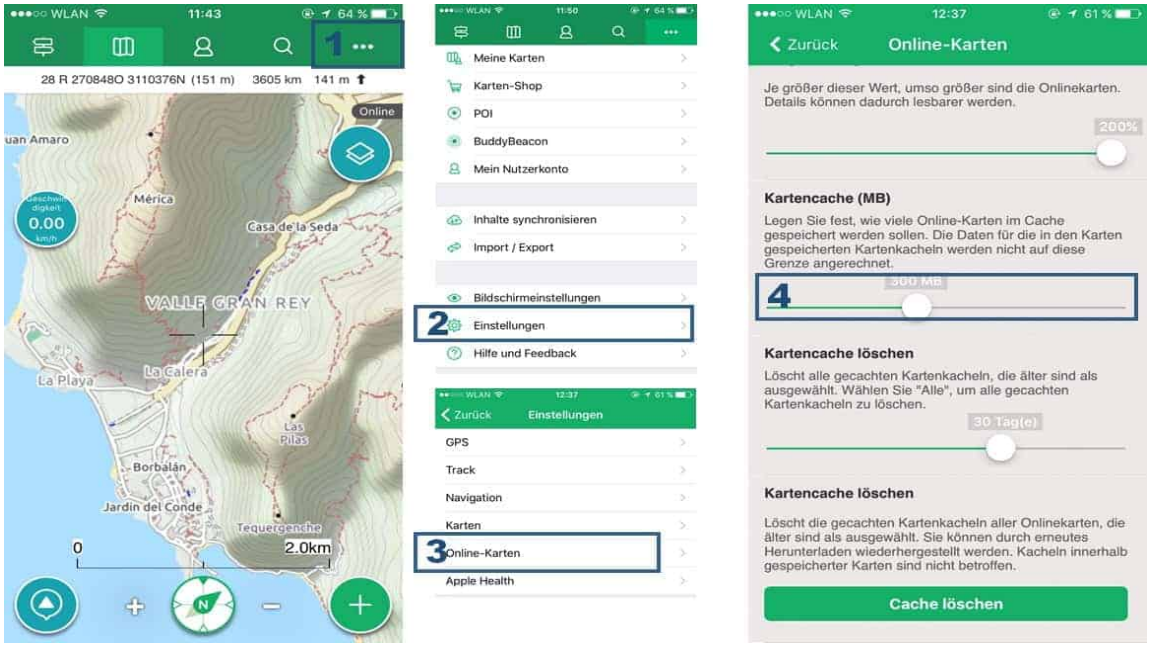


Εικόνα 17. World around me

Πηγή: https://world-around-me.en.aptoide.com/app

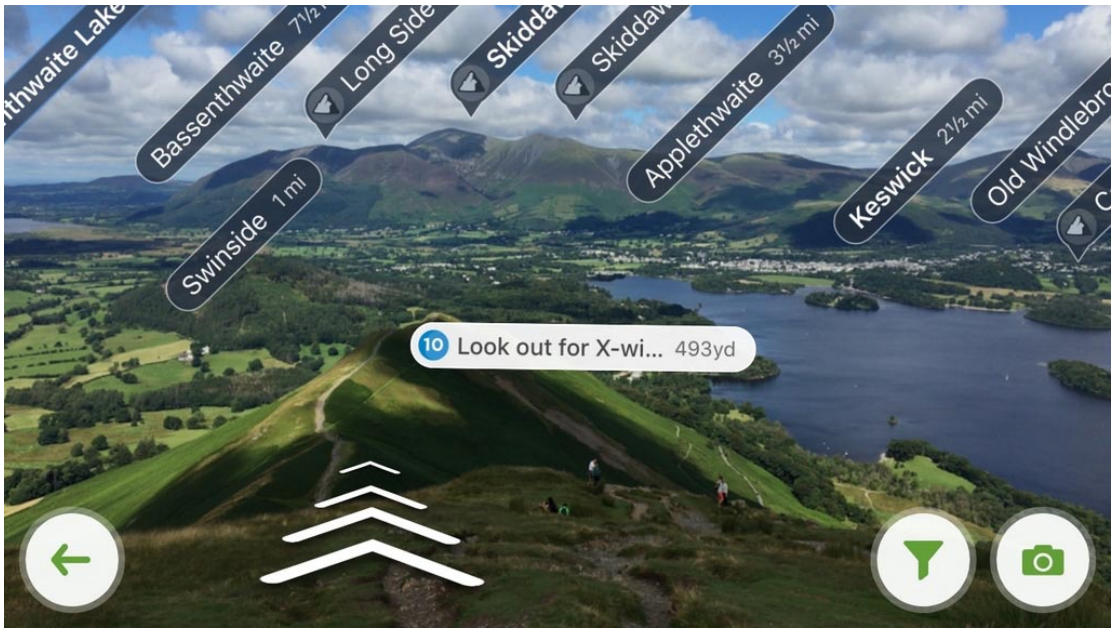
2.5.4. Viewranger

Εφαρμογή που χρησιμοποιεί χάρτη και δίνει οδηγίες στον χρήστη για οποιδήποτε σημείο στον κόσμο και εμφανίζει βέλοι πλοήγησης επαυξημένης πραγματικότητας αλλά και μέροι που βρίσκονται κοντά στον χρήστη και το πόσα χιλιόμετρα απέχει απο αυτά.



Εικόνα 18. Viewranger

Πηγή: https://www.wanderndeluxe.de/en/download-instructions-for-the-viewranger-app-map-material-for-offline-use/



Εικόνα 19. Viewranger

Πηγή: https://gearinstitute.com/how-viewranger-skyline-uses-virtual-reality-to-make-navigation-easier/

2.5.5. Smartify

Πολλοί αναφέρουν ότι η συγκεκριμένη εφαρμογή είναι για τα έργα τέχνης, όπως η εφαρμογή "shazam" για την αναγνώριση τραγουδών. Η συγκεκριμένη εφαρμογή δίνει πληροφορίες στον χρήστη για ένα συγκεκριμένο έργο τέχνης που ο χρήστης έχει φωτογραφήσει μέσω της εφαρμογής.

Εικόνα 20. Smartify

Πηγή: https://advisor.museumsandheritage.com/news/getting-to-know-smartify-partner-of-the-museums-heritage-autumn-series/

Εικόνα 21. Smartify

Πηγή: https://advisor.museumsandheritage.com/news/getting-to-know-smartify-partner-of-the-museums-heritage-autumn-series/

2.6. Λογισμικά ανάπτυξης εφαρμογής επαυξημένης και μεικτής πραγματικότητας

2.6.1. Microsoft Visual Studio

Το Microsoft Visual Studio [13] είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) από τη Microsoft. Χρησιμοποιείται για την ανάπτυξη προγραμμάτων ηλεκτρονικών υπολογιστών, καθώς και ιστοσελίδων, εφαρμογών ιστού, υπηρεσιών ιστού και εφαρμογών για κινητά. Το Visual Studio χρησιμοποιεί πλατφόρμες ανάπτυξης λογισμικού της Microsoft όπως τα Windows API, τα Windows Forms, το Windows Presentation Foundation, το Windows Store και το Microsoft Silverlight. Μπορεί να παράγει τόσο εγγενή κώδικα όσο και διαχειριζόμενο κώδικα. Το Visual Studio περιλαμβάνει ένα πρόγραμμα επεξεργασίας κώδικα το οποίο υποστηρίζει το IntelliSense (το στοιχείο ολοκλήρωσης κώδικα) καθώς και το refactoring κώδικα. Το ολοκληρωμένο πρόγραμμα εντοπισμού σφαλμάτων λειτουργεί τόσο ως πρόγραμμα εντοπισμού σφαλμάτων σε επίπεδο πηγής όσο και ως εργαλείο εντοπισμού σφαλμάτων σε επίπεδο μηχανής.

Άλλα ενσωματωμένα εργαλεία περιλαμβάνουν έναν προφίλ κώδικα, σχεδιαστή μορφών για την κατασκευή εφαρμογών GUI, σχεδιαστής ιστοσελίδων, σχεδιαστής τάξεων και σχεδιαστής σχήματος βάσης δεδομένων. Αποδέχεται plugins που βελτιώνουν τη λειτουργικότητα σχεδόν σε όλα τα επίπεδα - συμπεριλαμβανομένης της προσθήκης υποστήριξης για συστήματα ελέγχου πηγής (όπως το Subversion και το Git) και την προσθήκη νέων εργαλείων όπως editors και visual designers για συγκεκριμένες γλώσσες ή ομάδες εργαλείων. Το Visual Studio υποστηρίζει 36 διαφορετικές γλώσσες προγραμματισμού και επιτρέπει στον Code Editor και στον Debugger να υποστηρίζει (σε διαφορετικούς βαθμούς) σχεδόν οποιαδήποτε γλώσσα προγραμματισμού, υπό την προϋπόθεση ότι υπάρχει μια συγκεκριμένη γλώσσα. Οι ενσωματωμένες γλώσσες περιλαμβάνουν την C, C++, C++/CLI, Visual Basic .NET, C#, F#, JavaScript, TypeScript, XML, XSLT, HTML και CSS. Υποστήριξη για άλλες γλώσσες όπως Python, Ruby, Node.js και M μεταξύ άλλων είναι διαθέσιμη μέσω plugins. Η Java (και η J #) υποστηρίχθηκαν στο παρελθόν.

Εικόνα 22. Λογότυπο Microsoft Visual Studio

Πηγή: https://www.liblogo.com/lib/visual-studio-logo.html

2.6.2. Mapbox

Το Mapbox είναι μια πλατφόρμα προγραμματιστών που χρησιμοποιείται σε διάφορες βιομηχανίες για τη δημιουργία προσαρμοσμένων εφαρμογών που επιλύουν προβλήματα με χάρτες, δεδομένα και χωρική ανάλυση.

Για να χρησιμοποιηθεί οποιοδήποτε από τα εργαλεία του Mapbox [14], τα API ή τα SDK, χρειάζεται ένα Mapbox access token. Το Mapbox χρησιμοποιεί τα access token για τη συσχέτιση αιτημάτων API με το λογαριασμό του χρήστη. Ακόμη έχει την δυνατότητα δημιουργίας τεσσάρων (4) βασικών ειδών χαρτών:

1) **Mapbox Streets**: Περιλαμβάνει δρόμους, κτίρια, διοικητικές περιοχές, δεδομένα στεριάς και θαλασσών, με βάση το OpenStreetMap, ενημερωμένο τόσο συχνά όσο κάθε πέντε λεπτά,
2) **Mapbox Terrain**: Περιλαμβάνει στοιχεία για την κάλυψη του εδάφους και ένα παγκόσμιο σύνολο δεδομένων με πλήρεις πληροφορίες σχετικά με τα περιγράμματα, τα στοιχεία του λόφου και τα δεδομένα ανύψωσης,
3) **Mapbox Satellite**: Περιλαμβάνει παγκόσμιες δορυφορικές εικόνες από μια σειρά πηγών, επεξεργασμένες και συνεπτυγμένες από το Mapbox και
4) **Mapbox Traffic**: Περιλαμβάνει ενημερωμένες πληροφορίες σχετικά με την κυκλοφοριακή συμφόρηση πάνω από τις οδούς Mapbox.

Εικόνα 23. Λογότυπο Mapbox

Πηγή: https://www.liblogo.com/lib/visual-studio-logo.html

2.6.3. Unity Game Engine

Η μηχανή Unity Game Engine (https://unity.com/) χρησιμοποιείται για τη δημιουργία 2D και 3D εφαρμογών χάρη στα εργαλεία SDKs και APIs που διαθέτει, τα οποία παρέχουν στον προγραμματιστή ευκολία στη δημιουργία μίας εφαρμογής. Μπορεί να δημιουργήσει το οτιδήποτε για κινητά IOS και Android, φορητούς και επιτραπέζιους υπολογιστές και tablets. Πολλοί προγραμματιστές το προτιμούν λόγω του API της γλώσσας C# που διαθέτει μέσω του Visual studio. Επίσης, υποστηρίζει και τη γλώσσα Javascript σε διαφορετικό IDE (MonoDevelop) για όσους επιθυμούν μία ενναλακτική εκτός του Visual Studio [15].

Μερικά από τα κυριότερα εργαλεία που διαθέτει είναι τα εξής:

- **Asset store:** Το Unity Asset Store περιέχει μια βιβλιοθήκη με δωρεάν και εμπορικά στοιχεία που δημιουργούν οι Unity Technologies και τα μέλη της κοινότητας. Διατίθεται μεγάλη ποικιλία στοιχείων, όπως υφές, μοντέλα, 2D και 3D γραφικά, κινούμενα σχέδια, ολόκληρα παραδείγματα έργων, διάφορα scripts και επεκτάσεις του Editor. Ξεκινώντας από το Unity 2020.1,

το ειδικό παράθυρο του Asset Store δεν φιλοξενείται πλέον μέσα στον Unity Editor. Ωστόσο, στον ιστότοπο του Asset Store https://assetstore.unity.com/ μπορείτε να αναζητήσετε πακέτα που έχετε αγοράσει και κατεβάσει από το Asset Store και να τα εισάγετε και να τα κατεβάσετε απευθείας στο παράθυρο Package Manager.

- **Cloud Code:** Οι δυνατότητες που παρέχει είναι πολλές. Μέσω Cloud Code μπορεί να δημιουργηθεί μία εφαρμογή πολλαπλών χρηστών, η οποία μπορεί να λειτουργήσει μέσω server ή να δημιουργηθεί μία εφαρμογή με άλλους συνεργάτες αποθηκεύοντας για παράδειγμα μια εργασία στον χώρο αυτόν.
- **Editor:** Μπορούν να δημιουργηθούν εργαλεία και scripts με υποστηριζόμενα APIs τα οποία μπορούν να χρησιμοποιηθούν στην πλατφόρμα της Unity.
- **MARS:** Το Unity Mars βοηθά στην επίλυση των δύσκολων προβλημάτων της δημιουργίας εφαρμογών εφαρμογών επαυξημένης πραγματικότητας (AR) για δυναμικά φυσικά περιβάλλοντα, με καλύτερες ροές εργασίας, μείωση του χρόνου ανάπτυξης και εργαλεία συγγραφής σε απλή γλώσσα και πολλαπλές πλατφόρμες με υποστήριξη για iOS, Android και HoloLens.

Εικόνα 24. Λογότυπο Unity Game Engine

Πηγή: https://commons.wikimedia.org/wiki/File:Unity_Technologies_logo.svg

2.6.4. AR Foundation

Το AR Foundation [16] επιτρέπει τη δημιουργία εφαρμογών επαυξημένης πραγματικότητας (AR) πολλαπλών πλατφορμών με το Unity. Σε ένα έργο AR Foundation, επιλέγετε ποια χαρακτηριστικά AR θα ενεργοποιήσετε προσθέτοντας τα αντίστοιχα στοιχεία διαχειριστή στη σκηνή σας. Όταν κατασκευάζετε και εκτελείτε την εφαρμογή σε μια συσκευή, το AR Foundation ενεργοποιεί αυτά τα χαρακτηριστικά χρησιμοποιώντας το εγγενές AR SDK της πλατφόρμας, ώστε να μπορείτε να δημιουργήσετε μία φορά και να την αναπτύξετε στις κορυφαίες πλατφόρμες AR του κόσμου. Το πακέτο AR Foundation περιέχει διεπαφές για χαρακτηριστικά AR, αλλά δεν υλοποιεί κανένα χαρακτηριστικό.

Το πακέτο AR Foundation διαθέτει πολλά managers για διάφορες λειτουργίες χρησιμοποιώντας το native AR SDK. Μερικές από αυτές είναι οι εξής:

- **plane detection:** Αναλύει τον χώρο και ανιχνεύει τις επιφάνειες για διάφορους λόγους, όπως για παράδειγμα, την τοποθέτηση ενός αντικειμένου.
- **Track image:** Με την ανίχνευση εικόνας μπορούμε να τοποθετήσουμε ένα εικονικό αντικείμενο αντί για ένα πραγματικό. Οποιαδήποτε εικόνα έχει τραβηχτεί με μία φωτογραφική μηχανή ή ένα κινητό και έχει τοποθετηθεί στην βιβλιοθήκη του AR track image manager μπορεί να ανιχνευθεί και μόλις συμβεί αυτό να ενεργοποιηθεί οποιαδήποτε λειτουργία θελήσει ο προγραμματιστής.
- **Raycast:** Με την λειτουργία του raycast ο προγραμματιστής μπορεί πάλι να κάνει ότι ενέργεια θελήσει, αλλά αυτήν την φορά πατώντας πάνω στο αντικείμενο το οποίο έχει αναφερθεί στο raycast manager.
- **Body and face tracking**: Ανίχνευση προσώπου και σώματος.
- **Anchors:** Ανιχνεύει ένα συγκεκριμένο σημείο και το μαρκάρει για οποιαδήποτε χρήση.
- **Meshes:** Δημιουργία διαφόρων χρωμάτων και σχεδίων που βρίσκουμε στο περιβάλλον.

Εικόνα 25. Unity AR Foundation

Πηγή: https://blog.unity.com/news/how-ar-foundation-and-mars-work-together-to-enable-interactive-multiplatform-ar-experiences

Για να χρησιμοποιήσετε το AR Foundation σε μια πλατφόρμα-στόχο, χρειάζεστε επίσης ένα ξεχωριστό πακέτο πρόσθετου παρόχου για την εν λόγω πλατφόρμα. Η Unity Game Engine υποστηρίζει επίσημα τα ακόλουθα πρόσθετα παρόχου για το AR Foundation:

i) Google ARCore XR Plug-in στο Android,

ii) Apple ARKit XR Plug-in στο iOS και

iii) OpenXR Plug-in στο HoloLens 2

3. Ανάπτυξη εφαρμογήςΚύκλος ζωής ανάπτυξης λογισμικού

Για την ανάπτυξη μιας εφαρμογής για κινητά, πρέπει να έχουμε καλή κατανόηση του κύκλου ζωής ανάπτυξης λογισμικού (Software Development LifeCycle - SDLC). Είναι μια διαδικασία που ορίζει τα στάδια της ανάπτυξης λογισμικού, από την αρχική ιδέα έως την τελική ανάπτυξη. Ο κύκλος SDLC περιλαμβάνει τα ακόλουθα στάδια:

- **Δημιουργία ιδεών**: Αυτό είναι το στάδιο όπου μας έρχεται η ιδέα για την εφαρμογή μας.
- **Συλλογή απαιτήσεων**: Αυτό είναι το στάδιο όπου καθορίζουμε τις απαιτήσεις για την εφαρμογή μας.
- **Σχεδιασμός**: Αυτό είναι το στάδιο όπου σχεδιάζουμε τη διεπαφή χρήστη (UI) και την εμπειρία χρήστη (UX) για την εφαρμογή μας.
- **Ανάπτυξη**: Αυτό είναι το στάδιο όπου αναπτύσσουμε τον κώδικα της εφαρμογής.
- **Δοκιμές**: Αυτό είναι το στάδιο όπου δοκιμάζουμε την εφαρμογή για σφάλματα και λάθη.
- **Ανάπτυξη**: Αυτό είναι το στάδιο κατά το οποίο ξεκινάμε την εφαρμογή στο κατάστημα εφαρμογών.
- **Συντήρηση**: Αυτό είναι το στάδιο όπου συντηρούμε την εφαρμογή μετά την κυκλοφορία της.

3.2. Ορισμός Ιδέας εφαρμογής

Για να ορίσουμε την ιδέα της εφαρμογής μας, μπορεούμε να ακολουθήσουμε τα εξής βήματα:

- Προσδιορίζουμε το πρόβλημα ή την ανάγκη: Πρέπει να προσδιορίσουμε το πρόβλημα ή την ανάγκη που θα λύσει η εφαρμογή μας.
- Καταιγισμός ιδεών για λύσεις: Μπορούμε να κάνουμε καταιγισμό ιδεών για την επίλυση του προβλήματος ή της ανάγκης.
- Αξιολόγηση λύσεων: Μπορούμε να αξιολογήσουμε τις λύσεις με βάση τη σκοπιμότητά τους, τα δυνητικά έσοδα και τη ζήτηση της αγοράς.
- Επιλέγουμε την καλύτερη λύση: Μπορούμε να επιλέξουμε την καλύτερη λύση που ανταποκρίνεται στα κριτήριά μας.

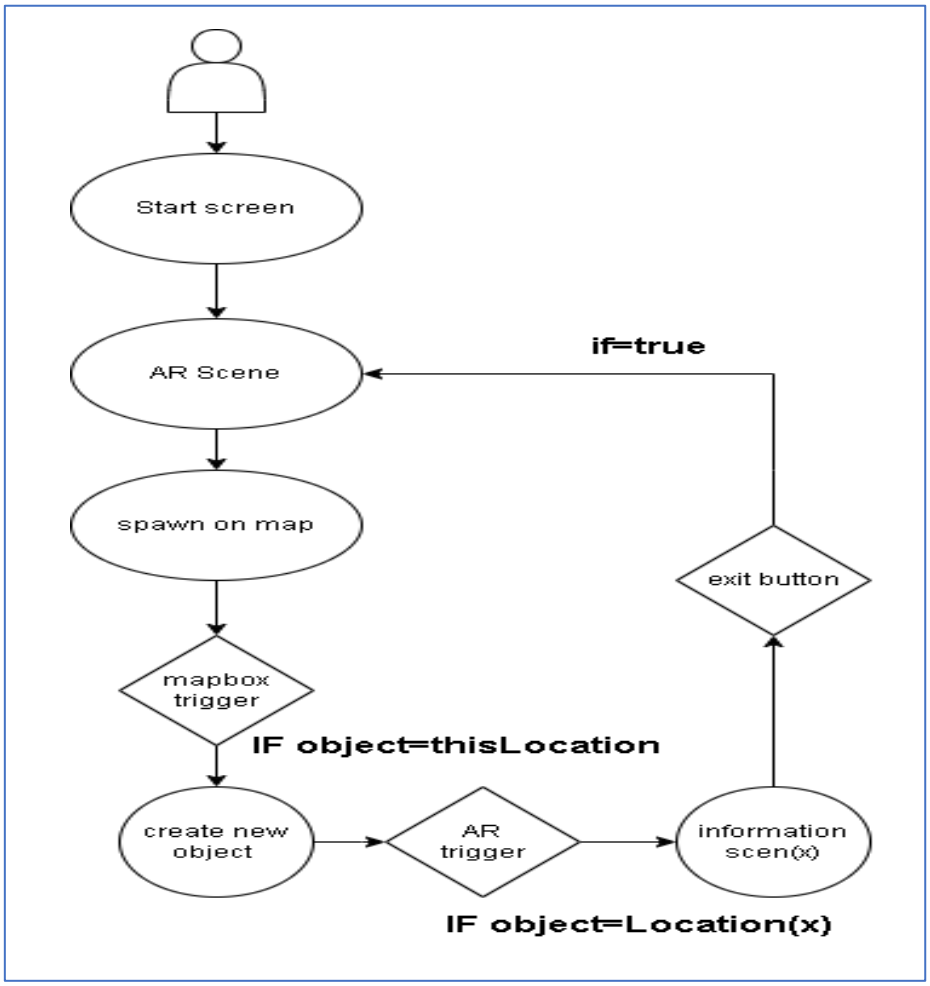
3.3. Εργαλεία εφαρμογής

Τέλος πρέπει να βρεθούν τα εργαλεία που μας ταιριάζουν για την δημιουργία της εφαρμογής μας. Στη συγκεκριμένη εφαρμογή χρησιμοποιήθηκαν τα εργαλεία :

- **Unity**: Για την δημιουργία της εφαρμογής, όπως και το ενσωματωμένο εργαλείο Unity AR Foundation για την υποστήριξη δυνατοτήτων επαυξημένης πραγματικότητας μέσω της κάμερας και την οθόνης της έξυπνης κινητής συσκευής.
- **Mapbox**: Για την δημιουργία του χάρτη τα API directions για την πλοήγηση όπως και πολλά scripts για την εύρεση συγκεκριμένης τοποθεσίας με συντεταγμένες και τοποθέτηση διαφόρων αντικειμένων στις συγκεκριμένες τοποθεσίες για αλληλεπίδραση.
- **Visual Studio**: Για τη συγγραφή των scripts με την υποστηριζόμενη γλώσσα C# που περιέχει.

3.4. Περιγραφή εφαρμογής και διάγραμμα ροής Kastoria3D

Η συγκεκριμένη εφαρμογή δημιουργήθηκε με σκοπό ενίσχυση της εμπειρίας ξενάγησης του χρήστη στα αξιοθέατα και τα μουσεία της πόλης της Καστοριάς, μέσω χρήσης επαυξημένης και μεικτής πραγματικότητας σε έξυπνες κινητές συσκευές Android. Η ολοκληρωμένη δομή και το διάγραμμα ροής αποτυπώνεται στην παρακάτω εικόνα:



Εικόνα 26. Διάγραμμα εφαρμογής Kastoria3D

Ο χρήστης επιλέγει το σημείο ενδιαφέροντος (μνημείο-αξιοθέατο) στο οποίο επιθυμεί να περιηγηθεί έχοντας ανοιχτή την κάμερα του και βλέποντας μία μίξη πραγματικού και εικονικού κόσμου. Με τη βοήθεια των χαρτών πλοήγησης και των οδηγιών που διαθέτει η εφαρμογή, ο χρήστης καθοδηγείται από το σημείο που βρίσκεται μέχρι το σημείο ενδιαφέροντος. Κατά την περιήγηση εάν ο χρήστης βρεθεί κοντά σε μία καφετέρια, φούρνο, βυζαντινά μνημεία, πιάτσα ταξί, αστικά λεωφορεία, κλπ. εμφανίζεται πάλι ένα αντικείμενο και τον ενημερώνει με ένα κείμενο ότι βρίσκεται στην συγκεκριμένη τοποθεσία. Κατά την διάρκεια της περιήγησης μας εμφανίζεται κείμενο με διάφορες ιστορίες που αφορούν την πόλη της Καστοριάς. Όταν ο χρήστης πλησιάσει κοντά στο σημείο ενδιαφέροντος εμφανίζεται στην οθόνη ένα αντικείμενο και ένα κείμενο ότι βρίσκεται κοντά στο σημείο και του ζητά να πλησιάσει ακόμα περισσότερο. Μόλις πλησιάσει αρκετά γίνεται αλλαγή σκηνής και εμφανίζονται στην οθόνη πληροφορίες για το συγκεκριμένο σημείο. Τέλος, πατώντας το κουμπί «Έξοδος» μεταφέρεται στην αρχική οθόνη της εφαρμογής, προκειμένου να συνεχιστεί η πλοήγηση.

Τα σημαντικά σημεία ενδιαφέροντος που έχουν προστεθεί είναι:

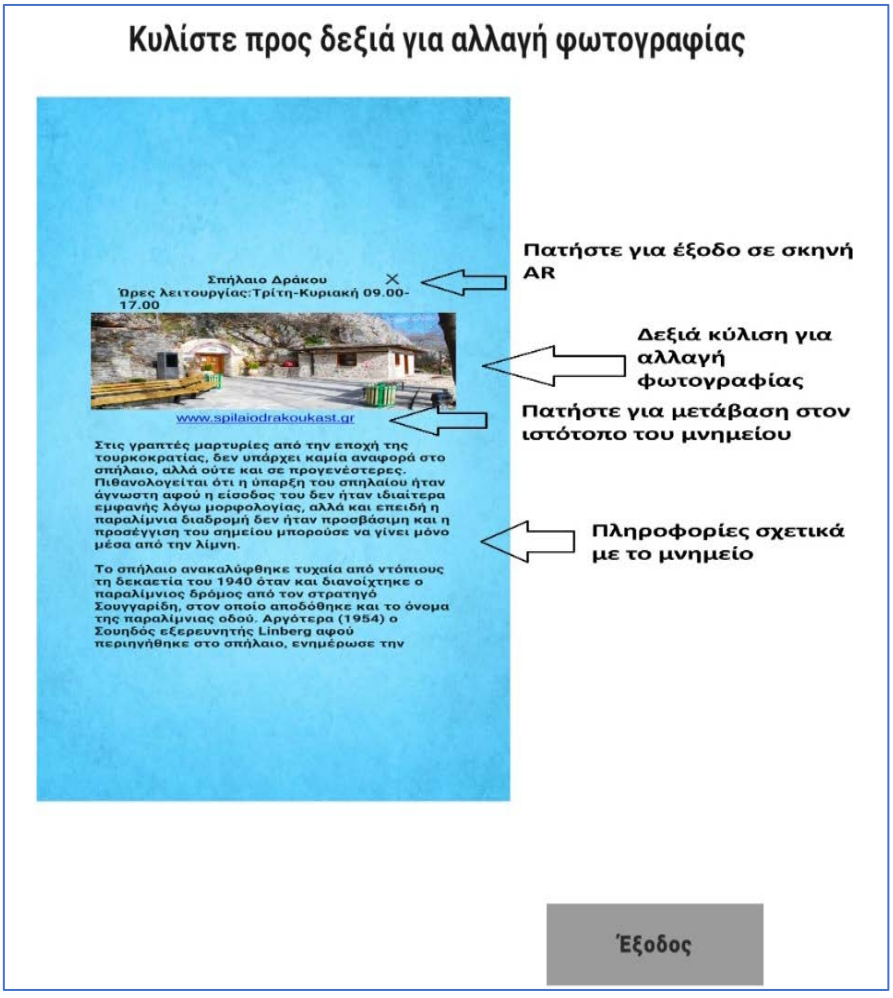
- Σπήλαιο δράκου
- Ενυδρείο
- Μουσείο Μακεδονικού Αγώνα
- Ενδυματολογικό Μουσείο
- Λαογραφικό Μουσείο
- Δεληνάνειο Μουσείο
- Μουσείο κέρινων ομοιωμάτων
- Προφήτης Ηλίας
- Λιμναίος Οικισμός

Επίσης προστέθηκαν ως σημείο αναφοράς ότι βρίσκεστε κοντά σε καφετέρια, φούρνο, πιάτσα ταξί, στάσεις αστικής και υπεραστικής συγκοινωνίας (ΚΤΕΛ).

3.4.1. Αρχική σκηνή

Στην αρχική σκηνή παρατηρούμε δυο κουμπιά, ένα για την εκκίνηση της εφαρμογής κι ένα για τις πληροφορίες για την εφαρμογή. Πατώντας στο κουμπί των πληροφοριών απενεργοποιείται ο πρώτος καμβάς και ενεργοποιείται ένας άλλος, ο οποίος περιέχει εικόνες σχετικά με οδηγίες και πληροφορίες χρήσης της εφαρμογής, οι οποίες αλλάζουν με κύλιση προς τα αριστερά με τον κώδικα SwipeLeftImageChange και ένα κουμπί Έξοδος για να μεταφερθούμε πάλι στην αρχική οθόνη.

Εικόνα 27. Εικόνα αρχικής σκηνής



Εικόνα 28. Εικόνα πληροφοριών αρχικής σκηνής

3.4.2. Σκηνή AR και MR

Αν πατήσουμε το κουμπί εκκίνησης της αρχικής σκηνής μεταφερόμαστε στην κύρια σκηνή επαυξημένης και μεικτής πραγματικότητας με τον κώδικα ChangeScene Ενώ παρατηρούμε τον πραγματικό κόσμο με την κάμερα μας, εμφανίζεται και ένας μικρός χάρτης μέσω του mapbox αλλά και ένα αντικείμενο (βοηθός) μπροστά στην κάμερα που περπατάει μαζί με τον χρήστη με την βοήθεια του κώδικα Raycast.

Επίσης υπάρχει και ένα κουμπί με πληροφορίες πατώντας το εμφανίζεται στην οθόνη ένα μενού για τα επιλεγμένα σημεία και δύο κουμπιά με οδηγίες ένα για οδήγηση και ένα για πεζούς πατώντας ποιό από τα δύο θέλουμε μας εμφανίζεται στον minimap η γραμμή πλοήγησης προς το σημείο μέσω των DirectionsFactory και του DirectionsFactoryFoot αν πατήσουμε ένα κουμπί για οδηγίες τότε αυτό αλλάζει σε έξοδος το οποίο αν το πατήσουμε κλείνει τις συγκεκριμένες οδηγίες.

Τέλος μέσω του κώδικα SpawnOnMap έχω τοποθετήσει αόρατα αντικείμενα για να κάνουν trigger με το playertarget του mapbox και όταν γίνει το trigger με τον κώδικα TriggerActivation να ανοίγει ένας καμβάς αλλά και να δημιουργείτε ένα αντικείμενο (πινέζα) μπροστά στην κάμερα με των κώδικα SpawnOnMap2

Εικόνα 29. Σκηνή AR & MR

Αν φτάσουμε ακόμα πιο κοντά στο σημείο ενδιαφέροντος γίνεται trigger πάλι με ένα αόρατο αντικείμενο, το οποίο μας αλλάζει σκηνή με την βοήθεια του κώδικα TriggerDetection. Κατά τη διάρκεια της διαδρομής ο βοηθός μας αναφέρει ιστορίες

για την πόλη της Καστοριάς χρησιμοποιώντας μία εικόνα που μετακινείται μαζί με την κάμερα AR με τον κώδικα raycast2 η οποία περιέχει διάφορα κείμενα τα οποία ενεργοποιούνται τυχαία, ανά κάποια δευτερόλεπτα, μέσω του κώδικα ImageText και του κώδικα TyppingEffect για να αλλάζει σελίδα το κείμενο ανά κάποια δευτερόλεπτα.

3.4.3. Σκηνή πληροφοριών

Αν γίνει το trigger που αναφέραμε προηγουμένως τότε κατευθυνόμαστε στην σκηνή των πληροφοριών εκεί βλέπουμε έναν καμβά με ένα κείμενο με την ονομασία του σημείου όπως και το ωράριο λειτουργίας. Επίσης βλέπουμε και μία εικόνα η οποία κάνοντας κύλιση προς τα αριστερά αλλάζει, αυτό πραγματοποιείται με τον κώδικα SwipeLeftImageChange

Επίσης βλέπουμε και ένα κείμενο που περιέχει πληροφορίες για το συγκεκριμένο σημείο το οποίο μετακινείται προς τα πάνω και προς τα κάτω με τον κώδικα TextScrolling

Τέλος υπάρχει και ένας σύνδεσμος με τον ιστότοπο του κάθε σημείου που δημιουργήθηκε με τον κώδικα HyperLink και ένα κουμπί έξοδος(Χ) το οποίο μας μεταφέρει και πάλι στην σκηνή επαυξημένης και μεικτής πραγματικότητας με τον κώδικα XButton .

Εικόνα 30. Σκηνή πληροφοριών

Συμπεράσματα

Οι τεχνολογίες επαυξημένης και μεικτής πραγματικότητας έχουν έρθει στο προσκήνιο και θα είναι ένα βασικό κομμάτι στις μελλοντικές εφαρμογές λόγο των δυνατοτήτων που παρέχουν. Στην τουριστική περιήγηση τέτοιου είδους εφαρμογές κάνουν την διαφορά για την αύξηση της επισκεψιμότητας μίας πόλης.

Η πλατφόρμα Unity Game Engine είναι αρκετά δημοφιλής για τη δημιουργία τέτοιων εφαρμογών. Με τα εργαλεία που διαθέτει παρέχει την ευελιξία κι ευκολία να διαχειριστείς κατάλληλα το project που δημιουργείς και διαθέτει πολύ καλές ομάδες επικοινωνίας και forums χρηστών, που μπορούν να βοηθήσουν για κάθε πρόβλημα που προκύπτει κατά την ανάπτυξη ενός project. Σε συνδυασμό με το εργαλείο AR Foundation προσφέρει ποικίλες δυνατότητες ανάπτυξης εφαρμογών επαυξημένης και μεικτής πραγματικότητας. Το εργαλείο Mapbox είναι το κατάλληλο δωρεάν λογισμικό για τη δημιουργία χαρτών αλλά και για μετατροπή συντεταγμένων του πραγματικού κόσμου σε εικονικό μέσω των API που διαθέτει.

Τέλος, μετά τη δημιουργία της εφαρμογής Kastoria3D προκύπτει το συμπέρασμα ότι η δημιουργία τέτοιου είδους εφαρμογών επαυξημένης και μεικτής πραγματικότητας, είναι ένας ενδιαφέρον τρόπος εκμάθησης πολλών νέων και καινοτόμων λειτουργιών για το καλύτερο δυνατό αποτέλεσμα.

Προτάσεις για μελλοντικές Βελτιώσεις

Μερικές βελτιώσεις που μπορεί να γίνουν στην εφαρμογή είναι οι εξής:

- Η επέκταση της εφαρμογής και η προσθήκη περισσότερων σημείων ενδιαφέροντος.
- Η δημιουργία web έκδοσης της εφαρμογής, καθώς και έκδοσης για κινητές συσκευές με λειτουργικό σύστημα iOS.
- Βελτίωση πληροφοριών (με περιεχόμενο σε ξένες γλώσσες) καθώς και προσθήκη δυνατότητας εικονικής ξενάγησης στα μουσεία και στα αξιοθέατα.
- Η δημιουργία της εφαρμογής και για άλλες πόλεις.

Βιβλιογραφία

[1] P. Georgiadis and C. Kay, "Τι είναι το Android;," 01 July 2020. [Online]. Available: https://www.doctorandroid.gr/p/iphone.html. [Accessed 01 February 2024].

[2] «Android Architecture,» 13 June 2011. [Ηλεκτρονικό]. Available: https://elinux.org/Android_Architecture. [Πρόσβαση 28 February 2024].

[3] «Android SDK and it's Components,» 18 February 2021. [Ηλεκτρονικό]. Available: https://www.geeksforgeeks.org/android-sdk-and-its-components/. [Πρόσβαση 28 February 2024].

[4] «API Framework,» 31 August 2023. [Ηλεκτρονικό]. Available: https://appmaster.io/glossary/api-framework. [Πρόσβαση 03 March 2024].

[5] «What is Augmented Reality?,» 31 August 2023. [Ηλεκτρονικό]. Available: https://fi.edu/en/what-is-augmented-reality. [Πρόσβαση 01 March 2024].

[6] J. Werner, «Catchup With Ivan Sutherland - Inventor Of The First AR Headset,» 23 February 2024. [Ηλεκτρονικό]. [Πρόσβαση 03 March 2024].

[7] H. Kato και M. Billinghurst, «Marker Tracking and HMD Calibration for a Video-based Augmented Reality Conferencing System,» σε *Proceedings 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR'99)*, San Francisco, CA, USA,, 1999.

[8] «Microsoft HoloLens,» 16 October 2016. [Ηλεκτρονικό]. Available: https://learn.microsoft.com/en-us/hololens/. [Πρόσβαση 01 March 2024].

[9] «What is Mixed Reality?,» 25 January 2023. [Ηλεκτρονικό]. Available: https://learn.microsoft.com/en-us/windows/mixed-reality/discover/mixed-reality. [Πρόσβαση 01 March 2024].

[10] P. Milgram και K. Fumio, «A Taxonomy of Mixed Reality Visual Displays,» *IEICE Transactions on Information and Systems,* τόμ. 77, αρ. 12, pp. pp. 1321-1329, 25 December 1994.

[11] «Augmented Reality (AR) in Healthcare,» Swevens Immersive Studio, 19 March 2019. [Ηλεκτρονικό]. Available: https://medium.com/swevens/augmented-reality-ar-in-healthcare-3c12bdf86a8e. [Πρόσβαση 01 March 2024].

[12] «Augmented Reality in Healthcare: Use Cases, Examples and Trends,» Fingent.com, 01 June 2021. [Ηλεκτρονικό]. Available: https://www.fingent.com/blog/augmented-reality-in-healthcare-use-cases-examples-and-trends/. [Πρόσβαση 31 March 2024].

[13] «Welcome to the Visual Studio 2017 IDE,» learn.microsoft.com, 02 October 2022. [Ηλεκτρονικό]. Available: https://learn.microsoft.com/en-us/previous-versions/visualstudio/visual-studio-2017/get-started/visual-studio-ide?view=vs-2017.

[Πρόσβαση 31 March 2024].

[14] «Mapbox Mobile Maps SDKs,» Mapbox.com, 02 October 2023. [Ηλεκτρονικό]. Available: https://www.mapbox.com/mobile-maps-sdk. [Πρόσβαση 01 April 2024].

[15] J. Petty, «What is Unity 3D and what is it used for?,» Conceptartempire.com, 01 March 2019. [Ηλεκτρονικό]. Available: https://conceptartempire.com/what-is-unity/. [Πρόσβαση 01 April 2024].

[16] «AR Foundation,» Unity Technologies, 18 October 2023. [Ηλεκτρονικό]. Available: https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@5.0/manual/index.html. [Πρόσβαση 01 April 2024].

[17] «Τι Είναι το API και Πώς Λειτουργεί; (Οδηγός 2023),» Bigblue Academy, 31 July 2022. [Ηλεκτρονικό]. Available: https://bigblue.academy/gr/ti-einai-to-api. [Πρόσβαση 04 April 2024].

Παράρτημα Α. Κώδικας Εφαρμογής

Raycast.cs

Κώδικας για να ακολουθεί και να μετακινείται ανάλογα το αντικείμενο με την κάμερα

```
using UnityEngine;

public class raycast : MonoBehaviour
{
    public Camera ARCamera; // Reference to your AR camera
    public float distanceFromCamera = 1f; // Distance from the camera
    public float yOffset = -0.5f; // Offset in the vertical direction
    public float rotationSpeed = 1f; // Speed of rotation
    public float movementThreshold = 0.01f; // Threshold for camera movement
detection

    private Animator animator; // Reference to the Animator component
    private int isWalkingParameterHash; // Hash of the "IsWalking" parameter for
efficiency
    private Vector3 lastCameraPosition; // Last recorded position of the camera
    private Vector3 lastCameraRotation; // Last recorded rotation of the camera

    void Start()
    {
        // Get the Animator component attached to this GameObject
        animator = GetComponent<Animator>();
        if (animator == null)
        {
            Debug.LogError("Animator component not found on the GameObject.");
        }

        // Calculate the hash of the "IsWalking" parameter for efficiency
        isWalkingParameterHash = Animator.StringToHash("IsWalking");

        // Initialize the last recorded position and rotation of the camera
        if (ARCamera != null)
        {
            lastCameraPosition = ARCamera.transform.position;
            lastCameraRotation = ARCamera.transform.rotation.eulerAngles;
        }
    }

    void Update()
    {
        // Check if the AR camera reference is set
        if (ARCamera != null)
        {
            // Calculate the position offset along the camera's forward vector
            Vector3 offset = ARCamera.transform.forward * distanceFromCamera;

            // Calculate the final position with the offset
            Vector3 finalPosition = ARCamera.transform.position + offset +
Vector3.up * yOffset;

            // Set the position of this GameObject
            transform.position = finalPosition;

            // Calculate the rotation difference between the current and last
camera rotation
            Vector3 deltaRotation = ARCamera.transform.rotation.eulerAngles -
lastCameraRotation;

            // Apply rotation to the object based on the camera rotation
```

```
            transform.Rotate(Vector3.up, deltaRotation.y * rotationSpeed);

            // Update the last recorded rotation of the camera
            lastCameraRotation = ARCamera.transform.rotation.eulerAngles;

            // Check if the camera is moving
            bool isCameraMoving = CheckIfCameraIsMoving();

            // Activate or deactivate the animator based on camera movement
            if (animator != null)
            {
                animator.enabled = isCameraMoving;
                animator.SetBool(isWalkingParameterHash, isCameraMoving);
            }
        }
        else
        {
            Debug.LogWarning("AR camera reference is not set.");
        }
    }

    bool CheckIfCameraIsMoving()
    {
        // Get the current position of the AR camera
        Vector3 currentCameraPosition = ARCamera.transform.position;

        // Check if the camera position has changed since the last frame
        if (Vector3.Distance(currentCameraPosition, lastCameraPosition) >
movementThreshold)
        {
            // Camera is moving
            lastCameraPosition = currentCameraPosition; // Update the last camera
position
            return true;
        }
        else
        {
            // Camera is not moving
            return false;
        }
    }
}
```

XButton.cs

Κώδικας αλλαγής σκηνής με το κουμπί έξοδος

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class XButton : MonoBehaviour
{
    public string sceneName; // Name of the scene to load

    // Method to be called when the button is pressed
    public void ChangeScene()
    {
        // Load the scene with the specified name
        SceneManager.LoadScene(sceneName);
    }
}
```

MapCameraController.cs

Κώδικας για να παραμένει ο minimap μέσα στο όρια της AR κάμερας

```
using UnityEngine;

public class MapCameraController : MonoBehaviour
{
    public Transform playerTransform; // Reference to the player's transform
    public Vector3 offset; // Offset from the player's position

    private void LateUpdate()
    {
        // Calculate target position for the camera (player's position plus offset)
        Vector3 targetPosition = playerTransform.position + offset;

        // Update the camera's position to match the target position
        transform.position = targetPosition;
    }
}
```

TriggerDetection.cs

Κώδικας για trigger αντικειμένου κάμερας με αντικείμενο που γίνεται spawn σε συγκεκριμένη τοποθεσία και αλλαγή σκηνής για άνοιγμα σκηνής πληροφοριών

```
using UnityEngine;
using UnityEngine.SceneManagement;
using System.Collections;

public class TriggerDetection : MonoBehaviour
{
    public string sceneName; // Name of the scene to load
    public float cooldownDuration = 180f; // Cooldown duration in seconds (3
minutes)

   private static bool hasTriggered = false; // Flag to control trigger activation

    private void OnTriggerEnter(Collider other)
    {
        // Check if the collider is tagged as "Player2" (or replace with desired
tag) and the trigger is enabled
        if (other.CompareTag("Player") && !hasTriggered)
        {
            Debug.Log("Player entered trigger zone.");

            // Load the scene with the specified name
            SceneManager.LoadScene(sceneName);

            hasTriggered = true;

        }
        else
        {
            StartCoroutine(TriggerAfterDelay(120f)); // Change the delay time as
needed
        }
    }

    private void OnTriggerExit(Collider other)
    {
```

```
        // Check if the collider is tagged as "Player2" (or replace with desired
tag)
        if (other.CompareTag("Player"))
        {
            Debug.Log("Player exited trigger zone.");
        }
    }

    // Coroutine for trigger cooldown
    private IEnumerator TriggerAfterDelay(float delay)
    {
        yield return new WaitForSeconds(delay);

        // Trigger the second event here
        SceneManager.LoadScene(sceneName);
    }
}
```

TriggerActivation.cs

Κώδικας trigger αντικειμένου playertarget του χάρτη με αντικείμενο που γίνεται spawn στον χάρτη για να ενεργοποιηθεί αντικείμενο που περιέχει τον κώδικα που κανει spawn αντικείμενο σε συγκεκριμενή τοποθεσία στην κάμερα AR και ενός καμβά.

```
using UnityEngine;

public class TriggerActivation : MonoBehaviour
{
    public string objectName; // Name of the game object to activate

    private void OnTriggerEnter(Collider other)
    {
        if (other.CompareTag("Player")) // Assuming the trigger is meant for the
player
        {
            ActivateObject();
        }
    }

    private void OnTriggerExit(Collider other)
    {
        if (other.CompareTag("Player")) // Assuming the trigger is meant for the
player
        {
            DeactivateObject();
        }
    }

    private void ActivateObject()
    {
        string parentObjectName = "shmantika"; // Specify the name of the parent
object here

        GameObject parentObj = GameObject.Find(parentObjectName); // Find the
parent object by name

        if (parentObj != null)
        {
            Transform objToActivate = parentObj.transform.Find(objectName); // Find
the object by name within the parent

            if (objToActivate != null)
            {
```

```
                objToActivate.gameObject.SetActive(true); // Activate the object
                Debug.Log("Activated object: " + objToActivate.name);
            }
            else
            {
                Debug.LogWarning("Object not found with the name: " + objectName);
            }
        }
        else
        {
            Debug.LogWarning("Parent object not found with the name: " +
parentObjectName);
        }
    }

    private void DeactivateObject()
    {
        string parentObjectName = "shmantika"; // Specify the name of the parent
object here

        GameObject parentObj = GameObject.Find(parentObjectName); // Find the
parent object by name

        if (parentObj != null)
        {
            Transform objToDeactivate = parentObj.transform.Find(objectName); //
Find the object by name within the parent

            if (objToDeactivate != null)
            {
                objToDeactivate.gameObject.SetActive(false); // Deactivate the
object
                Debug.Log("Deactivated object: " + objToDeactivate.name);
            }
            else
            {
                Debug.LogWarning("Object not found with the name: " + objectName);
            }
        }
        else
        {
            Debug.LogWarning("Parent object not found with the name: " +
parentObjectName);
        }
    }
}
```

SpawnOnMap2.cs

Κώδικας που κάνει spawn αντικείμενο μπροστά στην κάμερα AR το οποίο καταστρέφεται μετά απο 15 δευτερόλεπτα

```
using UnityEngine;
using Mapbox.Utils;
using Mapbox.Unity.Map;
using Mapbox.Unity.Utilities;
using System.Collections.Generic;
using UnityEngine.XR.ARFoundation;

namespace Mapbox.Examples
{
    public class SpawnOnMap2 : MonoBehaviour
    {
```

```
        [SerializeField]
        AbstractMap _map;

        [SerializeField]
        [Geocode]
        string[] _locationStrings;
        Vector2d[] _locations;

        [SerializeField]
        GameObject _objectPrefab;

        [SerializeField]
        float _spawnScale = 1f;

        [SerializeField]
        float _spawnDistance = 5f; // Distance from the AR camera to spawn the
object

        [SerializeField]
        float _leftOffset = -1f; // Offset to the left side

        [SerializeField]
        float _rightOffset = 1f; // Offset to the right side

        [SerializeField]
        float _downOffset = -1f; // Offset downwards

        GameObject _spawnedObject;
        bool _objectSpawned = false; // Flag to track if object has been spawned

        ARSessionOrigin _arSessionOrigin;

        void Start()
        {
            _arSessionOrigin = FindObjectOfType<ARSessionOrigin>();

            if (!_objectSpawned)
            {
                Initialize();
            }
        }

     void Initialize()
{
    _locations = new Vector2d[_locationStrings.Length];

    // Get the AR camera transform
    Transform arCameraTransform = _arSessionOrigin.camera.transform;

    // Calculate spawn position in front of the AR camera
    Vector3 spawnPosition = arCameraTransform.position + arCameraTransform.forward
* _spawnDistance;

    // Adjust spawn position based on left, right, and down offsets
    Vector3 leftOffsetVector = arCameraTransform.right * _leftOffset;
    Vector3 rightOffsetVector = arCameraTransform.right * _rightOffset;
    Vector3 downOffsetVector = Vector3.down * _downOffset;

    // Calculate final spawn position
    Vector3 finalSpawnPosition = spawnPosition + leftOffsetVector +
rightOffsetVector + downOffsetVector;

    // Create a new game object to act as the anchor
    GameObject anchor = new GameObject("Anchor");
    anchor.transform.position = finalSpawnPosition;
```

```
    // Instantiate the object directly without parenting it to the anchor
    _spawnedObject = Instantiate(_objectPrefab, finalSpawnPosition,
Quaternion.identity);

    // Set the scale
    _spawnedObject.transform.localScale = new Vector3(_spawnScale, _spawnScale,
_spawnScale);

    // Set the layer
    _spawnedObject.layer = LayerMask.NameToLayer("ar");

    _objectSpawned = true;


            // Destroy the spawned object after 15 seconds
            Destroy(_spawnedObject, 15f);
        }

        void Update()
        {
            if (_objectSpawned)
            {
                // Move object left
                if (Input.GetKeyDown(KeyCode.LeftArrow))
                {
                    MoveObject(-1f);
                }
                // Move object right
                else if (Input.GetKeyDown(KeyCode.RightArrow))
                {
                    MoveObject(1f);
                }
            }
        }

        void MoveObject(float offset)
        {
            if (_spawnedObject != null)
            {
                _spawnedObject.transform.position += Vector3.right * offset;
            }
        }

        void OnEnable()
        {
            // Check if object has not been spawned yet, then spawn it
            if (!_objectSpawned)
            {
                Initialize();
            }
        }

        void OnDisable()
        {
            // Destroy spawned object when the game object is disabled
            if (_spawnedObject != null)
            {
                Destroy(_spawnedObject);
                _objectSpawned = false;
            }
        }
    }
}
```

ImageClickHandler.cs

Κώδικας για μεγένθυση και σμύκρινσεις εικόνας στο πάτημα

```
using UnityEngine;
using UnityEngine.UI;

public class ImageClickHandler : MonoBehaviour
{
    private bool isEnlarged = false;
    private Vector2 defaultSize;
    private int defaultSiblingIndex;
    private Vector2 defaultPosition;

    [SerializeField]
    private float enlargeFactor = 2.0f;
    [SerializeField]
    private float transitionDuration = 0.5f;

    private RectTransform rectTransform;
    private Canvas canvas;

    private void Start()
    {
        rectTransform = GetComponent<RectTransform>();
        defaultSize = rectTransform.sizeDelta;
        defaultSiblingIndex = transform.GetSiblingIndex();
        defaultPosition = rectTransform.anchoredPosition;

        canvas = GetComponentInParent<Canvas>();
    }

    public void ToggleEnlarge()
    {
        if (isEnlarged)
        {
            StartCoroutine(TransitionToDefaultSize());
        }
        else
        {
            StartCoroutine(TransitionToEnlargedSize());
        }
    }

    private System.Collections.IEnumerator TransitionToDefaultSize()
    {
        float time = 0.0f;
        Vector2 initialSize = rectTransform.sizeDelta;
        Vector2 targetSize = defaultSize;

        // Store the current sibling index
        int currentSiblingIndex = transform.GetSiblingIndex();

        while (time < transitionDuration)
        {
            time += Time.deltaTime;
            float t = Mathf.Clamp01(time / transitionDuration);
            rectTransform.sizeDelta = Vector2.Lerp(initialSize, targetSize, t);
            yield return null;
        }

        rectTransform.sizeDelta = defaultSize;
        rectTransform.anchoredPosition = defaultPosition;

        // Restore the sibling index
        transform.SetSiblingIndex(currentSiblingIndex);
```

```
        isEnlarged = false;
    }

    private System.Collections.IEnumerator TransitionToEnlargedSize()
    {
        float time = 0.0f;
        Vector2 initialSize = rectTransform.sizeDelta;
        Vector2 targetSize = defaultSize * enlargeFactor;

        // Set the sibling index to the highest value
        transform.SetAsLastSibling();

        while (time < transitionDuration)
        {
            time += Time.deltaTime;
            float t = Mathf.Clamp01(time / transitionDuration);
            rectTransform.sizeDelta = Vector2.Lerp(initialSize, targetSize, t);

            // Center the image on the canvas
            Vector2 canvasCenter =
canvas.GetComponent<RectTransform>().rect.center;
            Vector2 targetPosition = canvasCenter;
            rectTransform.anchoredPosition = Vector2.Lerp(defaultPosition,
targetPosition, t);

            yield return null;
        }

        rectTransform.sizeDelta = targetSize;
        isEnlarged = true;
    }
}
```

HyperLinkText.cs

Κώδικας για δημιουργεία συνδέσμου του text component του unity

```
using UnityEngine;
using TMPro;
using UnityEngine.EventSystems;
using System.Text.RegularExpressions;

public class HyperLinkText : MonoBehaviour, IPointerClickHandler
{
    private TextMeshProUGUI textComponent;
    private string originalText;

    void Start()
    {
        textComponent = GetComponent<TextMeshProUGUI>();
        originalText = textComponent.text;
        UpdateText();
    }

    void UpdateText()
    {
        textComponent.text = ParseHyperlinks(originalText);
    }

    string ParseHyperlinks(string input)
    {
```

```
        return Regex.Replace(input, @"\[([^]]+)\]\((.*?)\)",
"<color=blue><u>$1</u></color>");
    }

    public void OnPointerClick(PointerEventData eventData)
    {
        string clickedText =
eventData.pointerPress.GetComponent<TextMeshProUGUI>().text;
        Match match = Regex.Match(originalText, @"\[([^]]+)\]\((.*?)\)");

        while (match.Success)
        {
            if (clickedText.Contains(match.Groups[1].Value))
            {
                string url = match.Groups[2].Value;
                Application.OpenURL(url);
                break;
            }
            match = match.NextMatch();
        }
    }
}
```

ChangeScene.cs

Κώδικας αλλαγής σκηνής μέσω button

```
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;

public class ChangeScene : MonoBehaviour
{
    [SerializeField]
    private string sceneName; // The name of the scene to load

    void Start()
    {
        // Get the Button component attached to the GameObject
        Button button = GetComponent<Button>();

        // Check if a Button component is attached
        if (button != null)
        {
            // Add a listener to the button's onClick event
            button.onClick.AddListener(LoadScene);
        }
        else
        {
            Debug.LogWarning("ChangeScene script requires a Button component
attached to the GameObject.");
        }
    }

    void LoadScene()
    {
        // Check if the scene name is not empty
        if (!string.IsNullOrEmpty(sceneName))
        {
            // Load the scene
            SceneManager.LoadScene(sceneName);
        }
        else
```

```
        {
            Debug.LogWarning("Scene name is not set in the ChangeScene script.");
        }
    }
}
```

SpawnOnMap.cs

Κώδικας του mapbox για δημιουργία αντικειμένων σε συγκεκριμένη τοποθεσία στον χάρτη με μία καθυστέρηση κατα το άνοιγμα

```
namespace Mapbox.Examples
{
    using UnityEngine;
    using Mapbox.Utils;
    using Mapbox.Unity.Map;
    using Mapbox.Unity.MeshGeneration.Factories;
    using Mapbox.Unity.Utilities;
    using System.Collections.Generic;
    using UnityEngine.SceneManagement;

    public class SpawnOnMap : MonoBehaviour
    {
        [SerializeField]
        AbstractMap _map;

        [SerializeField]
        [Geocode]
        string[] _locationStrings;
        Vector2d[] _locations;

        [SerializeField]
        float _spawnScale = 100f;

        [SerializeField]
        GameObject _markerPrefab;

        List<GameObject> _spawnedObjects;
        private float startWait = 8f;


        void Start()
        {

            Invoke("Initialize", startWait);
        }


        void Initialize()
        {
            _locations = new Vector2d[_locationStrings.Length];
            _spawnedObjects = new List<GameObject>();
            for (int i = 0; i < _locationStrings.Length; i++)
            {
                var locationString = _locationStrings[i];
                _locations[i] = Conversions.StringToLatLon(locationString);
                var instance = Instantiate(_markerPrefab);
                instance.transform.localPosition =
_map.GeoToWorldPosition(_locations[i], true);
                instance.transform.localScale = new Vector3(_spawnScale,
_spawnScale, _spawnScale);
                _spawnedObjects.Add(instance);
            }
```

```
        }

        private void OnUpdate()
        {
            int count = _spawnedObjects.Count;
            for (int i = 0; i < count; i++)
            {
                var spawnedObject = _spawnedObjects[i];
                var location = _locations[i];
                spawnedObject.transform.localPosition =
_map.GeoToWorldPosition(location, true);
                spawnedObject.transform.localScale = new Vector3(_spawnScale,
_spawnScale, _spawnScale);
            }
        }
    }
    }
```

DirectionsFactory.cs

Κώδικας του mapbox για την δημιουργία γραμμής πλοήγησης για οδήγηση στον minimap ως δεύτερη παράμετρο βάζουμε τις συντεταγμένες της τοποθεσίας που θέλουμε

```
namespace Mapbox.Unity.MeshGeneration.Factories
{
using UnityEngine;
using Mapbox.Directions;
using System.Collections.Generic;
using System.Linq;
using Mapbox.Unity.Map;
using Mapbox.Unity.MeshGeneration.Modifiers;
using Mapbox.Unity.MeshGeneration.Data;
using Mapbox.Utils;
using Mapbox.Unity.Utilities;
using System.Collections;


    public class DirectionsFactory : MonoBehaviour
    {
        [SerializeField]
        AbstractMap _map;

        [SerializeField]
        MeshModifier[] MeshModifiers;

        [SerializeField]
        Material _material;

        [SerializeField]
        Transform _playerTransform;

        [SerializeField]
        Vector2d _secondWaypoint;

        private List<Vector3> _cachedWaypoints;

        [SerializeField]
        [Range(1, 10)]
        private float UpdateFrequency = 2;

        private Directions _directions;
        private int _counter;
```

```
        private GameObject _directionsGO;
        private GameObject _directionsGO_AR; // GameObject for AR layer
        private bool _recalculateNext;

        private void Awake()
        {
            if (_map == null)
            {
                _map = FindObjectOfType<AbstractMap>();
            }
            _directions = MapboxAccess.Instance.Directions;
            _map.OnInitialized += Query;
            _map.OnUpdated += Query;
            _cachedWaypoints = new List<Vector3>();
            _cachedWaypoints.Add(_playerTransform.position);
            _recalculateNext = false;

            foreach (var modifier in MeshModifiers)
            {
                modifier.Initialize();
            }
        }

        private IEnumerator StartAfterInitialization()
        {
            yield return null;
            Query();
            StartCoroutine(QueryTimer());
        }

        private void OnDisable()
        {
            _map.OnInitialized -= Query;
            _map.OnUpdated -= Query;
        }

        private void Query()
        {
            var wp = new Vector2d[] {
_playerTransform.GetGeoPosition(_map.CenterMercator, _map.WorldRelativeScale),
_secondWaypoint };
            var _directionResource = new DirectionResource(wp,
RoutingProfile.Driving);
            _directionResource.Steps = true;
            _directions.Query(_directionResource, HandleDirectionsResponse);
        }

        private IEnumerator QueryTimer()
        {
            while (true)
            {
                yield return new WaitForSeconds(UpdateFrequency);
                for (int i = 0; i < _cachedWaypoints.Count; i++)
                {
                    if (_cachedWaypoints[i] != _playerTransform.position)
                    {
                        _recalculateNext = true;
                        _cachedWaypoints[i] = _playerTransform.position;
                    }
                }

                if (_recalculateNext)
                {
                    Query();
                    _recalculateNext = false;
```

```
                }
            }
        }

        private void HandleDirectionsResponse(DirectionsResponse response)
        {
            if (response == null || response.Routes == null ||
response.Routes.Count < 1)
            {
                return;
            }

            var meshData = new MeshData();
            var dat = new List<Vector3>();
            foreach (var point in response.Routes[0].Geometry)
            {
                dat.Add(Conversions.GeoToWorldPosition(point.x, point.y,
_map.CenterMercator, _map.WorldRelativeScale).ToVector3xz());
            }

            var feat = new VectorFeatureUnity();
            feat.Points.Add(dat);

            foreach (var mod in MeshModifiers)
            {
                if (mod != null && mod.Active)
                {
                    mod.Run(feat, meshData, _map.WorldRelativeScale);
                }
            }

            // Create GameObjects for each line
            if (_directionsGO == null)
            {
                _directionsGO = CreateGameObject(meshData,
"direction_waypoint_entity", LayerMask.NameToLayer("Default"));
            }
            else if (_directionsGO_AR == null)
            {
                _directionsGO_AR = CreateGameObject(meshData,
"direction_waypoint_entity_AR", LayerMask.NameToLayer("ar"));
            }
            else
            {
                // Delete the second object with layer "ar"
                Destroy(_directionsGO_AR);
                _directionsGO_AR = null;
            }
        }

        private GameObject CreateGameObject(MeshData data, string name, int layer)
        {
            GameObject go = new GameObject(name);
            go.transform.parent = transform;
            go.layer = layer;

            var mesh = go.AddComponent<MeshFilter>().mesh;
            mesh.subMeshCount = data.Triangles.Count;

            mesh.SetVertices(data.Vertices);
            _counter = data.Triangles.Count;
            for (int i = 0; i < _counter; i++)
            {
                var triangle = data.Triangles[i];
                mesh.SetTriangles(triangle, i);
            }
```

```
            _counter = data.UV.Count;
            for (int i = 0; i < _counter; i++)
            {
                var uv = data.UV[i];
                mesh.SetUVs(i, uv);
            }

            mesh.RecalculateNormals();
            go.AddComponent<MeshRenderer>().material = _material;

            // Adjust the local position for the "ar" line
            if (layer == LayerMask.NameToLayer("ar"))
            {
                go.transform.localPosition += new Vector3(0, -6, 0);
            }

            return go;
        }


        public void Start()
        {
            StartCoroutine(StartAfterInitialization());
        }
    }
}
```

DirectionsFactoryFoot.cs

Κώδικας του mapbox για την δημιουργία γραμμής πλοήγησης για πεζούς στον minimap ως δεύτερη παράμετρο βάζουμε τις συντεταγμένες της τοποθεσίας που θέλουμε

```
namespace Mapbox.Unity.MeshGeneration.Factories
{
    using UnityEngine;
    using Mapbox.Directions;
    using System.Collections.Generic;
    using System.Linq;
    using Mapbox.Unity.Map;
    using Mapbox.Unity.MeshGeneration.Modifiers;
    using Mapbox.Unity.MeshGeneration.Data;
    using Mapbox.Utils;
    using Mapbox.Unity.Utilities;
    using System.Collections;

    public class DirectiosFactoryFoot : MonoBehaviour
    {
        [SerializeField]
        AbstractMap _map;

        [SerializeField]
        MeshModifier[] MeshModifiers;

        [SerializeField]
        Material _material;

        [SerializeField]
        Transform _playerTransform;

        [SerializeField]
        Vector2d _secondWaypoint;
```

```
        private List<Vector3> _cachedWaypoints;

        [SerializeField]
        [Range(1, 10)]
        private float UpdateFrequency = 2;

        private Directions _directions;
        private int _counter;

        private GameObject _directionsGO;
        private bool _recalculateNext;

        private void Awake()
        {
            if (_map == null)
            {
                _map = FindObjectOfType<AbstractMap>();
            }
            _directions = MapboxAccess.Instance.Directions;
            _map.OnInitialized += Query;
            _map.OnUpdated += Query;
            _cachedWaypoints = new List<Vector3>();
            _cachedWaypoints.Add(_playerTransform.position);
            _recalculateNext = false;

            foreach (var modifier in MeshModifiers)
            {
                modifier.Initialize();
            }
        }

        private IEnumerator StartAfterInitialization()
        {
            yield return null;
            Query();
            StartCoroutine(QueryTimer());
        }

        private void OnDisable()
        {
            _map.OnInitialized -= Query;
            _map.OnUpdated -= Query;
        }

        private void Query()
        {
            var wp = new Vector2d[] {
_playerTransform.GetGeoPosition(_map.CenterMercator, _map.WorldRelativeScale),
_secondWaypoint };
            var _directionResource = new DirectionResource(wp,
RoutingProfile.Walking); // Change RoutingProfile to Walking
            _directionResource.Steps = true;
            _directions.Query(_directionResource, HandleDirectionsResponse);
        }

        private IEnumerator QueryTimer()
        {
            while (true)
            {
                yield return new WaitForSeconds(UpdateFrequency);
                for (int i = 0; i < _cachedWaypoints.Count; i++)
                {
                    if (_cachedWaypoints[i] != _playerTransform.position)
                    {
                        _recalculateNext = true;
                        _cachedWaypoints[i] = _playerTransform.position;
```

```
                    }
                }

                if (_recalculateNext)
                {
                    Query();
                    _recalculateNext = false;
                }
            }
        }

        private void HandleDirectionsResponse(DirectionsResponse response)
        {
            if (response == null || response.Routes == null ||
response.Routes.Count < 1)
            {
                return;
            }

            var meshData = new MeshData();
            var dat = new List<Vector3>();
            foreach (var point in response.Routes[0].Geometry)
            {
                dat.Add(Conversions.GeoToWorldPosition(point.x, point.y,
_map.CenterMercator, _map.WorldRelativeScale).ToVector3xz());
            }

            var feat = new VectorFeatureUnity();
            feat.Points.Add(dat);

            foreach (var mod in MeshModifiers)
            {
                if (mod != null && mod.Active)
                {
                    mod.Run(feat, meshData, _map.WorldRelativeScale);
                }
            }

            // Create GameObject for the line
            if (_directionsGO == null)
            {
                _directionsGO = CreateGameObject(meshData,
"direction_waypoint_entity", LayerMask.NameToLayer("Default"));
            }
        }

        private GameObject CreateGameObject(MeshData data, string name, int layer)
        {
            GameObject go = new GameObject(name);
            go.transform.parent = transform;
            go.layer = layer;

            var mesh = go.AddComponent<MeshFilter>().mesh;
            mesh.subMeshCount = data.Triangles.Count;

            mesh.SetVertices(data.Vertices);
            _counter = data.Triangles.Count;
            for (int i = 0; i < _counter; i++)
            {
                var triangle = data.Triangles[i];
                mesh.SetTriangles(triangle, i);
            }

            _counter = data.UV.Count;
            for (int i = 0; i < _counter; i++)
            {
```

```
                var uv = data.UV[i];
                mesh.SetUVs(i, uv);
            }

            mesh.RecalculateNormals();
            go.AddComponent<MeshRenderer>().material = _material;

            return go;
        }

        public void Start()
        {
            StartCoroutine(StartAfterInitialization());
        }
    }
}
```

raycast2.cs

Κώδικας μετακίνησης εικόνας σύμφωνα με την AR κάμερα

```
using UnityEngine;

public class raycast2 : MonoBehaviour
{
    public Camera arCamera; // Reference to your AR camera
    public float distanceFromCamera = 1f; // Distance from the camera
    public float xOffset = -55f; // Offset in the horizontal direction
    public float yOffset = -0.5f; // Offset in the vertical direction
    public float rotationSpeed = 1f; // Speed of rotation

    void Start()
    {
        // Check if the AR camera reference is set
        if (arCamera != null)
        {
            // Calculate the position offset along the camera's forward vector
            Vector3 offset = arCamera.transform.forward * distanceFromCamera;

            // Calculate the final position with both horizontal and vertical
offsets
            Vector3 finalPosition = arCamera.transform.position + offset +
arCamera.transform.right * xOffset + Vector3.up * yOffset;

            // Set the position of the parent GameObject
            transform.parent.position = finalPosition;

            // Calculate the rotation difference between the current and last
camera rotation
            Quaternion targetRotation =
Quaternion.LookRotation(transform.parent.position - arCamera.transform.position);
            targetRotation.x = 0; // Keep the rotation only on the y-axis

            // Apply rotation to the object based on the camera rotation
            transform.parent.rotation = Quaternion.Slerp(transform.parent.rotation,
targetRotation, rotationSpeed * Time.deltaTime);
        }
        else
        {
            Debug.LogWarning("AR camera reference is not set.");
        }
    }
}
```

SwipeLeftImageChange.cs

Κώδικας αλλαγής εικόνας με κύλιση προς τα αριστερά

```
using UnityEngine;
using UnityEngine.UI;

public class SwipeLeftImageChange : MonoBehaviour
{
    public RawImage[] images; // Array of RawImages to cycle through

    private int currentIndex = 0; // Index of the current image

    private Vector2 touchStartPos;
    private bool isSwiping = false;

    void Start()
    {
        // Ensure only the first image is active initially
        images[currentIndex].gameObject.SetActive(true);
        for (int i = 1; i < images.Length; i++)
        {
            images[i].gameObject.SetActive(false);
        }
    }

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

            // Detect swipe left gesture
            switch (touch.phase)
            {
                case TouchPhase.Began:
                    touchStartPos = touch.position;
                    isSwiping = true;
                    break;

                case TouchPhase.Ended:
                    if (isSwiping)
                    {
                        float swipeDistance = touch.position.x - touchStartPos.x;
                        if (Mathf.Abs(swipeDistance) > Screen.width * 0.2f) //
Adjust the threshold for what constitutes a swipe left
                        {
                            if (swipeDistance < 0)
                            {
                                ChangeImages();
                            }
                        }
                        isSwiping = false;
                    }
                    break;
            }
        }
        // Check for mouse input if no touch input
        else if (Input.GetMouseButtonDown(0))
        {
            touchStartPos = Input.mousePosition;
            isSwiping = true;
        }
```

```
        else if (Input.GetMouseButtonUp(0) && isSwiping)
        {
            float swipeDistance = Input.mousePosition.x - touchStartPos.x;
            if (Mathf.Abs(swipeDistance) > Screen.width * 0.2f) // Adjust the
threshold for what constitutes a swipe left
            {
                if (swipeDistance < 0)
                {
                    ChangeImages();
                }
            }
            isSwiping = false;
        }
    }

    void ChangeImages()
    {
        // Deactivate current image
        images[currentIndex].gameObject.SetActive(false);

        // Increment index and wrap around if necessary
        currentIndex = (currentIndex + 1) % images.Length;

        // Activate new image
        images[currentIndex].gameObject.SetActive(true);
    }
}
```

TextScrolling.cs

Κώδικας μετακίνησεις κειμένου πάνω και κάτω

```
using UnityEngine;
using TMPro;

public class TextScrolling : MonoBehaviour
{
    public RectTransform textRectTransform;
    public float dragSpeed = 1f;
    public float minY = -100f; // Minimum Y position for scrolling
    public float maxY = 100f; // Maximum Y position for scrolling

    private Vector3 lastMousePosition;

    private void Update()
    {
        if (Input.GetMouseButtonDown(0))
        {
            lastMousePosition = Input.mousePosition;
        }
        else if (Input.GetMouseButton(0))
        {
            Vector3 deltaMouse = Input.mousePosition - lastMousePosition;
            float deltaY = deltaMouse.y * dragSpeed * Time.deltaTime;

            // Calculate the new Y position after scrolling
            float newY = Mathf.Clamp(textRectTransform.anchoredPosition.y + deltaY,
minY, maxY);

            // Update the Y position of the text
            textRectTransform.anchoredPosition = new
Vector2(textRectTransform.anchoredPosition.x, newY);
```

```
            lastMousePosition = Input.mousePosition;
        }
    }
}
```

TyppingEffect.cs

Κώδικας αλλαγής σελίδας κειμένου μετά από κάποια δευτερόλεπτα

```
using UnityEngine;
using TMPro;

public class TyppingEffect : MonoBehaviour
{
    private TMP_Text textComponent; // Use TMP_Text instead of Text
    public string[] pages;
    private int currentPageIndex = 0;
    private float timer = 0f;
    public float changeInterval = 5f;

    void Start()
    {
        // Get the TextMeshPro component attached to this GameObject
        textComponent = GetComponent<TMP_Text>();

        // Ensure there is a TextMeshPro component
        if (textComponent == null)
        {
            Debug.LogError("TextMeshPro component not found.");
            enabled = false; // Disable this script if TextMeshPro component is not
found
            return;
        }

        // Start displaying the initial page
        UpdateText();
    }

    void Update()
    {
        // Increment timer
        timer += Time.deltaTime;

        // Check if it's time to change the page
        if (timer >= changeInterval)
        {
            // Reset the timer
            timer = 0f;

            // Increment the page index, loop back to the beginning if necessary
            currentPageIndex = (currentPageIndex + 1) % pages.Length;

            // Update the text component with the new page
            UpdateText();
        }
    }

    // Update the text component with the current page content
    void UpdateText()
    {
        textComponent.text = pages[currentPageIndex];
    }
}
```

ImageText.cs

Κώδικας ενεργοποίησης εικόνας και ενός τυχαίου κειμένου μετά απο δευτερόλεπτα τα οποία τοποθετούμε και απενεργοποίηση αυτών μετά απο κάποια δευτερολέπτα τα οποία και πάλι τοποθετούμε

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

public class ImageText : MonoBehaviour
{
    public RawImage rawImage;
    public TextMeshProUGUI[] texts;
    public float activationDelay = 30f;
    public float[] displayDurations; // Array to hold display durations for each
text
    public float reactivationDelay = 10f;

    private List<int> activatedIndices = new List<int>();

    void Start()
    {
        // Disable all texts initially
        foreach (TextMeshProUGUI text in texts)
        {
            text.gameObject.SetActive(false);
        }

        // Start the coroutine to activate the image and random text after the
delay
        StartCoroutine(ActivateImageAndText());
    }

    IEnumerator ActivateImageAndText()
    {
        // Wait for activation delay
        yield return new WaitForSeconds(activationDelay);

        // Activate the raw image
        rawImage.gameObject.SetActive(true);

        // Choose a random text that hasn't been activated before
        int randomIndex = GetRandomUnusedIndex();
        if (randomIndex != -1)
        {
            TextMeshProUGUI randomText = texts[randomIndex];
            // Activate the random text
            randomText.gameObject.SetActive(true);
            // Set the text of the TextMeshProUGUI
            randomText.text = "Your random text here";

            // Add the index to the list of activated indices
            activatedIndices.Add(randomIndex);

            // Get the display duration for this text
            float textDisplayDuration = displayDurations[randomIndex];

            // Wait for the display duration of this text
            yield return new WaitForSeconds(textDisplayDuration);
```

```
            // Deactivate the raw image and random text after the text's display
duration
            rawImage.gameObject.SetActive(false);
            randomText.gameObject.SetActive(false);
            // Remove the index from the list of activated indices
            activatedIndices.Remove(randomIndex);
        }

        // Wait for reactivation delay
        yield return new WaitForSeconds(reactivationDelay);

        // Restart the coroutine to repeat the process
        StartCoroutine(ActivateImageAndText());
    }

    // Function to get a random index that hasn't been used before
    int GetRandomUnusedIndex()
    {
        List<int> unusedIndices = new List<int>();

        // Populate the list of unused indices
        for (int i = 0; i < texts.Length; i++)
        {
            if (!activatedIndices.Contains(i))
            {
                unusedIndices.Add(i);
            }
        }

        // Return a random index from the list of unused indices
        if (unusedIndices.Count > 0)
        {
            return unusedIndices[Random.Range(0, unusedIndices.Count)];
        }
        else
        {
            // If all texts have been activated, return -1
            return -1;
        }
    }
}
```