ΠΑΝΕΠΙΣΤΗΜΙΟ
ΙΩΑΝΝΙΝΩΝ

ΣΧΟΛΗ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΣΧΕΔΙΑΣΜΟΣ ΚΑΙ ΑΝΑΠΤΥΞΗ ΕΦΑΡΜΟΓΗΣ ΕΙΚΟΝΙΚΗΣ ΠΡΑΓΜΑΤΙΚΟΤΗΤΑΣ ΜΕ ΧΡΗΣΗ ΤΗΣ ΜΗΧΑΝΗΣ UNITY.

Γεωργίου Παναγιώτης Αθανάσιος

Επιβλέπων: Καθ. Χρυσόστομος Στύλιος

Άρτα, Φεβρουάριος, 2023

DESIGN AND APPLICATION DEVELOPMENT IN VIRTUAL REALITY USING THE UNITY GAME ENGINE.

Εγκρίθηκε από τριμελή εξεταστική επιτροπή

Άρτα, 14/2/2023

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

1. Επιβλέπων καθηγητής
   Χρυσόστομος Στύλιος,

2. Μέλος επιτροπής
   Βασιλική Λιάγκου,

3. Μέλος επιτροπής
   Σπυριδούλα Μαργαρίτη,

Δήλωση μη λογοκλοπής

Δηλώνω υπεύθυνα και γνωρίζοντας τις κυρώσεις του Ν. 2121/1993 περί Πνευματικής Ιδιοκτησίας, ότι η παρούσα πτυχιακή εργασία είναι εξ ολοκλήρου αποτέλεσμα δικής μου ερευνητικής εργασίας, δεν αποτελεί προϊόν αντιγραφής ούτε προέρχεται από ανάθεση σε τρίτους. Όλες οι πηγές που χρησιμοποιήθηκαν (κάθε είδους, μορφής και προέλευσης) για τη συγγραφή της περιλαμβάνονται στη βιβλιογραφία.

Γεωργίου, Παναγιώτης Αθανάσιος.

Υπογραφή

ΕΥΧΑΡΙΣΤΙΕΣ

Με την ολοκλήρωση της πτυχιακής μου εργασίας θα ήθελα να ευχαριστήσω τους ανθρώπους που με στήριξαν καθ' όλη τη διάρκεια των σπουδών μου.

Αρχικά θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου τον κ. Χρυσόστομο Στύλιο για την καθοδήγηση που μου προσέφερε και τον χρόνο που διέθεσε για την ολοκλήρωση της πτυχιακής μου εργασίας.

Στο ίδιο πλαίσιο ευγνωμοσύνης θα ήθελα να ευχαριστήσω τους καθηγητές του Τμήματος για τις γνώσεις που μου προσέφεραν.

Τέλος θα ήθελα να ευχαριστήσω του γονείς μου και τον αδερφό μου για την συμπαράσταση που μου έδιναν κατά την διάρκεια των σπουδών και όχι μόνο.

ΠΕΡΙΛΗΨΗ

Τα τελευταία χρόνια το ενδιαφέρον για την εικονική πραγματικότητα έχει ανθίσει και προσελκύει όλο και περισσότερο ενδιαφέρον γεγονός που οδηγεί στην ραγδαία ανάπτυξη της τεχνολογίας αυτής. Σήμερα η εικονική πραγματικότητα χρησιμοποιείται ως εργαλείο σε διάφορους κλάδους όπως της ψυχαγωγία, της εκπαίδευση, του τουρισμό και της υγείας.

Η παρούσα πτυχιακή εργασία περιέχει βασικές έννοιες της εικονικής πραγματικότητας, την ιστορική εξέλιξη αυτής από τις πρώτες προσπάθειες των πρωτοπόρων μέχρι τις πιο πρόσφατες τεχνολογικές εξελίξεις και τις χρήσεις της σε διάφορους κλάδους.

Τέλος, στο πρακτικό κομμάτι της πτυχιακής εργασία, έχει υλοποιηθεί μια εφαρμογή εικονικής πραγματικότητας που ως θέμα έχει το ιστορικό κάστρο της Άρτας και το μικρό θεατράκι της Αμβρακίας.

Η εφαρμογή αυτή στοχεύει στην χρήση της εικονικής πραγματικότητας ως εκπαιδευτικό εργαλείο μέσω της απλής μετάδοσης πληροφορίων αλλά και πιο διαδραστικά και ψυχαγωγικά μέσω του παιχνιδοποιημένου σεναρίου που θα προσελκύσει το ενδιαφέρων μικρότερων ηλικιών. Ταυτόχρονα με την μετάδοση γνώσης, η εφαρμογή χρησιμοποιείται και ως ένας τρόπος ανάδειξης των ιστορικών μνημείων της Άρτας.

Λέξεις κλειδιά: Εικονική Πραγματικότητα, Παιχνιδοποίηση, Εκπαιδευτικό εργαλείο, μηχανή Unity.

ABSTRACT

In recent years, interest in virtual reality has blossomed and is attracting more and more interest leading to the rapid development of this technology. Today virtual reality is used as a tool in various sectors such as entertainment, education, tourism and health.

This thesis contains basic concepts of virtual reality, its historical development from the early efforts of the pioneers to the most recent technological developments and its uses in various industries.

Finally, in the practical part of the thesis a virtual reality application has been implemented, which is about the historical castle of Arta and the small theatre of Amvrakia.

The aim of this application is to use virtual reality as an educational tool through simple transmission of information but also in a more interactive and entertaining way through the gamified scenario that will attract the interest of younger age groups. At the same time with the transmission of knowledge, the application is also used as a way of highlighting the historical monuments of Arta.

Key words: Virtual Reality, Gamification, Educational tool, Unity engine.

ΠΙΝΑΚΑΣ ΣΥΝΤΟΜΟΓΡΑΦΙΩΝ

HMD..……. Head Mounted Display

VR……..Virtual Reality

AR……..Augmented Reality

VE……Virtual Environment

FOR…….……..…..Field Of Regard

ΕΙΣΑΓΩΓΗ

Τα τελευταία τρία χρόνια με την επέκταση και την επικράτηση της υγειονομικής πανδημίας η παγκόσμια κοινότητα έκρινε αναγκαία την λήψη αυστηρών μέτρων για τον περιορισμό και την εξάλειψη της πανδημίας που έθεσε σε κίνδυνο τις βάσεις του σύγχρονου πολιτισμού. Ως κύριο μέτρο αποτροπής εφαρμόστηκε ο περιορισμός των κοινωνικών επαφών και η μερική κοινωνική απομόνωση του ατόμου έως ότου καταστεί εφικτή η αντιμετώπιση με κατάλληλα φαρμακευτικά σκευάσματα. Η παραπάνω ενέργειες έπληξαν, όπως ήταν φυσικό, το οικοδόμημα της σύγχρονης κοινωνίας, μεγάλες και μικρές επιχειρήσεις, εμπόριο παγκόσμιο και εγχώριο που, αποτελούν κρίκους της οικονομίας και της εργασίας, διαρρήχθηκαν. Με την σειρά τους υπηρεσίες παροχής δημόσιας ωφέλειας, ιδιωτικές, καταστάθηκαν ανενεργές ή συνέχισαν την λειτουργιά με χαρακτηριστικές ελλείψεις. Φυσικά η σχολική και ακαδημαϊκή κοινότητα έχει πληγεί σε μεγάλο βαθμό, γεγονός που δημιούργησε την ανάγκη για την ανάπτυξη και την χρήση εργαλείων-μέσων για την εξ αποστάσεως εργασία και εκπαίδευση, ώστε να καλυφθούν επιτακτικές ανάγκες σε αγαθά και υπηρεσίες.

Ένα πολύ σημαντικό εργαλείο εκπαίδευσης που χρησιμοποιήθηκε για να καλύψει τις ανάγκες τις εκπαίδευσης είναι η εικονικής πραγματικότητας (Virtual Reality). Το εργαλείο αυτό μπορεί να αξιοποιηθεί εφαρμόζοντας τις τεχνικές και τα χαρακτηριστικά ενός παιχνιδιού σε ένα απλό περιβάλλον εισάγοντας τους νέους εκπαιδευόμενους σε σύγχρονες ρεαλιστικές μεθόδους εκμάθησης.

Σκοπός της παρούσας πτυχιακής εργασίας είναι η σχεδίαση και η υλοποίηση μιας εφαρμογής εικονικής πραγματικότητας η οποία περιέχει το ιστορικό κάστρο και το μικρό θεατράκι της πόλης της Άρτας. Στόχος της εφαρμογής είναι η ανάδειξη της ιστορίας των δύο αυτών σημείων μέσω της μετάδοσης πληροφοριών και της ρεαλιστικής απεικόνιση του κάστρου σε ένα διαδραστικό περιβάλλον ώστε ο χρήστης της εφαρμογής να αλληλεπιδρά με το εικονικό «περιβάλλον» με ήχους εικόνες και αντικείμενα και να μαθαίνει για την ιστορία του κάστρου με την χρήση της παιχνιδοποίησης (Gamification).

1. ΒΑΣΙΚΕΣ ΕΝΝΟΙΕΣ.

1.1 Ορισμός Εικονικής Πραγματικότητας.

Η εικονική πραγματικότητα (Virtual Reality ή VR) είναι η χρήση τεχνολογίας υπολογιστών για την δημιουργία ενός προσομοιωμένου περιβάλλοντος τριών διαστάσεων που επιτρέπει στον χρήστη να αλληλεπιδράσει με αυτό. Σε αντίθεση με τις παραδοσιακές διεπαφές, το VR βάζει τον χρήστη σε αυτό το τεχνητό περιβάλλον με την βοήθεια ορισμένων διαδραστικών συσκευών που παρέχουν μια καθηλωτική εμπειρία 360°. Οι συσκευές αυτές στέλνουν και λαμβάνουν πληροφορίες και φοριούνται με την μορφή γυαλιών, γαντιών ή στολών. Στον εικονικό αυτόν κόσμο ο χρήστης έχει την ελευθερία να κάνει το οτιδήποτε, όπως το να συνομιλήσει με άλλα άτομα ή να πετάξει με ένα μαχητικό αεροπλάνο (Gandhi and Patel, 2018).

1.2 Διαφορές Εικονικής - Επαυξημένης Πραγματικότητας.

Οι όροι εικονική πραγματικότητα (VR) και επαυξημένη πραγματικότητα (AR) αρκετά συχνά συγχέονται. Το AR προκαλείται από την τοποθέτηση εικονικών αντικειμένων στο πραγματικό περιβάλλον, ενώ το VR μπορεί να οριστεί ως μια εφαρμογή για την διαμονή και την αλληλεπίδραση ατόμων σε εικονικά δημιουργημένα περιβάλλοντα χρησιμοποιώντας μια μεγάλη ποικιλία εργαλείων. Η διαφορά μεταξύ VR και AR βρίσκεται επίσης στις συσκευές που χρειάζεται η κάθε μια. Το VR απαιτεί μια συσκευή εμβύθισης αλλά το AR μπορεί να εκτελεστεί σε ένα έξυπνο κινητό τηλέφωνο. Επιπροσθέτως, το AR ενισχύει τόσο τον εικονικό όσο και τον πραγματικό κόσμο, ενώ το VR ενισχύει μόνο τη φανταστική πραγματικότητα. Τέλος οι χρήστες AR μπορούν να ελέγχουν την παρουσία τους στον πραγματικό κόσμο, ενώ οι χρήστες VR ελέγχονται από το σύστημα (YILDIRIM, ELBAN and YILDIRIM, 2018).

1.3 Είδη Εμβύθισης Εικονικής Πραγματικότητας.

Αν και είναι δύσκολο να ταξινομηθούν όλα τα συστήματα VR, οι περισσότερες διαμορφώσεις εμπίπτουν σε μια από τις τρεις κύριες κατηγορίες ανάλογα με το επίπεδο εμβύθισης (Immersion) που παρέχουν. Με τον όρο immersion αναφερόμαστε στον βαθμό τον οποίο η προσοχή του χρήστη εστιάζεται στην εργασία η οποία του παρέχεται. Η καθηλωτική εμπειρία δεν εξαρτάται από μια μεμονωμένη μεταβλητή αλλά από ένα σύνολο μεταβλητών όπως το επίπεδο διαδραστικότητας, την ανάλυση

της εικόνας, την στερεοσκοπική όραση, το πεδίο εκτίμησης (Field Of Regard – FOR), το πεδίο ορατότητας (Field Of View – FOV), τον ήχο και τον ρυθμό ανανέωσης της οθόνης. Για παράδειγμα, η στερεοσκοπική προβολή συγκριτικά με την μονοσκοπική προβολή εικονικού περιβάλλοντος (Virtual Environment-VR) βελτιώνει την εμβύθιση του χρήστη. Είναι αξιοσημείωτο πως δεν υπάρχουν αποτελεσματικές παράμετροι από μόνες τους και ο βαθμός βύθισης επιτυγχάνεται με την αλληλεπίδραση πολλών παραγόντων (Miller, 2021) (Andrzejczak et al., 2021).

1.3.1 Χαμηλής εμβύθισης.

Όπως υποδηλώνει το όνομα, τα συστήματα χαμηλής εμβύθισης είναι η λιγότερο καθηλωτική εφαρμογή της τεχνολογίας VR, είδος που είναι τόσο δημοφιλές στην καθημερινή ζωή όπου οι χαμηλές εικονικές εμπειρίες συχνά παραβλέπονται στην κατηγορία VR. Αυτή η τεχνική δημιουργεί έναν κόσμο που παράγεται από υπολογιστή, ενώ ο χρήστης έχει αντίληψη του φυσικού του περιβάλλοντος. Τα βιντεοπαιχνίδια είναι το πιο αντιπροσωπευτικό παράδειγμα αυτού του τύπου εμβύθισης. Το πλεονέκτημα αυτής της τεχνολογίας είναι πως δεν απαιτούνται υψηλά επίπεδα γραφικών ή κάποιο ειδικό υλικό. Περεταίρω, η αλληλεπίδραση γίνεται με τις παραδοσιακές συσκευές εισόδου όπως το πληκτρολόγιο και το ποντίκι. Ως αποτέλεσμα, αυτά τα συστήματα μπορούν να θεωρηθούν η πιο οικονομική επιλογή για ένα ευρύ φάσμα εφαρμογών, ένας παράγοντας που το καθιστά ως την πιο δημοφιλή τεχνική (Costello, 1997).

Εικόνα 1 VR χαμηλής εμβύθισης (Poetker, B., 2019.).

1.3.2 Ημι-εμβύθισης.

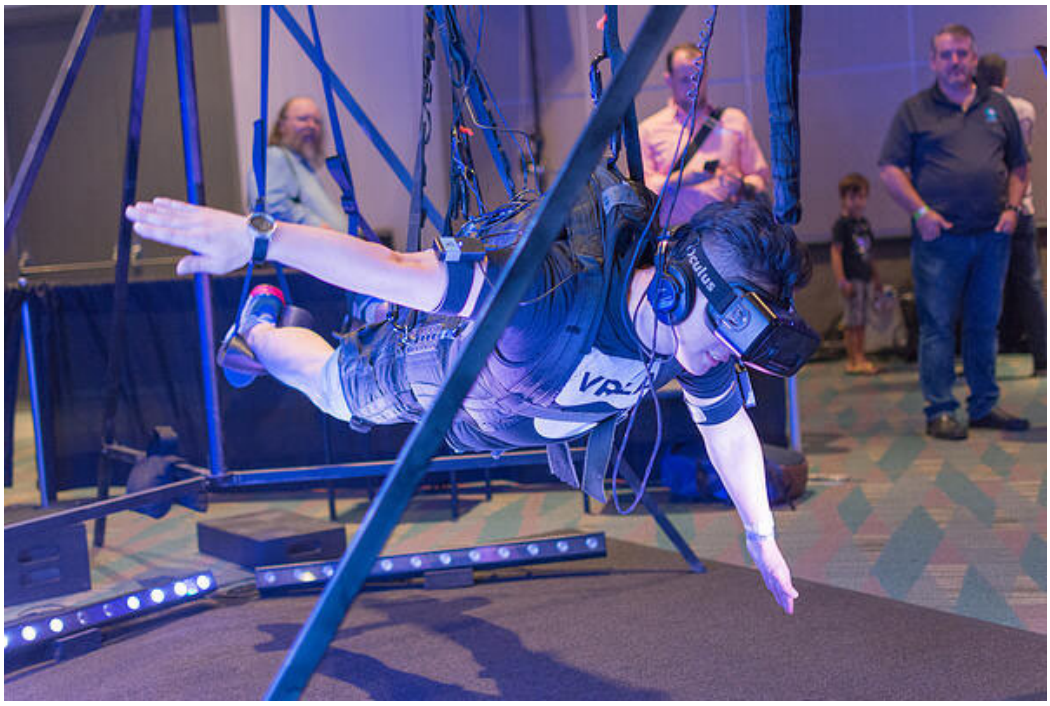
Οι χρήστες μπορούν να βυθιστούν σε ένα μερικώς περιβάλλον μέσω ημι-εμβυθιστικών εμπειριών. Οι χρήστες δεν είναι αποκομμένοι τελείως από το φυσικό τους περιβάλλον, αυτό σημαίνει πως ενώ εστιάζουν στην ψηφιακή εικόνα βρισκόμενοι σε μια διαφορετική πραγματικότητα θα εξακολουθούν να έχουν επαφή με το φυσικό τους περιβάλλον. Μια πιο καθηλωτική εμπειρία επιτυγχάνεται με την χρήση εικόνων με υψηλότερη λεπτομέρεια. Αυτός ο τύπος εικονικής πραγματικότητας χρησιμοποιείται συχνά για εκπαιδευτικούς σκοπούς και βασίζεται σε πολλαπλές οθόνες υψηλής ανάλυσης, ισχυρούς υπολογιστές και προβολείς εικόνας. Η χρήση του ευρέως οπτικού πεδίου, που επιτυγχάνεται με τις πολλαπλές οθόνες, αυξάνει την αίσθηση της εμβύθισης στον χρήστη. Ως εκ τούτου, τα ημιεμβυθιστικά συστήματα παρέχουν υψηλότερο επίπεδο βύθισης καθώς και καλύτερη κατανόηση της κλίμακας. Επιπροσθέτως, παρέχουν γραφικά με σημαντικά υψηλότερη ανάλυση από τα γυαλιά VR ή ένα HMD. Μία από τις πιο γνωστές χρήσεις της ημι-εμβύθισης είναι η εκπαίδευση, για παράδειγμα οι προσομοιωτές πτήσεις με τους οποίους οι χρήστες μπορούν να εκπαιδευτούν σε ασφαλείς συνθήκες (Costello, 1997).

Εικόνα 2 VR ημι-εμβύθισης (Poetker, B., 2019.).

1.3.3 Πλήρης εμβύθισης.

Η πλήρης εμβύθιση VR προσφέρει την πιο άμεση και ρεαλιστική εμπειρία στο εικονικό περιβάλλον με εικόνα και ήχο. Σε αυτόν τον τύπο εμβύθισης ο χρήστης φοράει

ένα HMD. Αυτές οι συσκευές προσφέρουν σχετικά υψηλή ανάλυση με ευρύ φάσμα όρασης. Η οθόνη του HMD χωρίζεται ανάμεσα στα μάτια του χρήστη για να παρέχει στερεοσκοπική εικόνα 3D και η παρακολούθηση εισόδου χρησιμοποιείται για να δημιουργήσει μια καθηλωτική, ζωντανή εμπειρία. Οι δύο οθόνες τοποθετούνται πολύ κοντά στο μάτι (50-70mm), αν και η εικόνα στην οποία εστιάζει ο χρήστης είναι πολύ πιο μακριά λόγο του οπτικού συστήματος HMD. Αυτό το είδος VR χρησιμοποιείται ευρέως για παιχνίδια και άλλες μορφές ψυχαγωγίας, παρόλα αυτά η χρήση τους σε άλλους τομείς, όπως η εκπαίδευση αυξάνεται. Κατά την χρήση της συσκευής ο χρήστης είναι αποκομμένος από τον πραγματικό κόσμο ενισχύοντας παράλληλα τον κόσμο που δημιουργείται από τον υπολογιστή. Πλεονέκτημα αυτής της τεχνολογίας είναι το γεγονός πως προσφέρει στον χρήστη ένα οπτικό πεδίο 360 μοιρών, πράγμα που σημαίνει πως ο χρήστης θα λάβει μια οπτική εικόνα γυρίζοντας το κεφάλι τους προς οποιαδήποτε κατεύθυνση. Ένα σύστημα πλήρης εμβύθισης θα προσφέρει μια αίσθηση που δεν μπορεί να συγκριθεί με καμία άλλη τεχνική εμβύθισης, βέβαια το επίπεδο εμβύθισης εξαρτάται από πολλούς παράγοντες, όπως το οπτικό πεδίο, η ανάλυση της εικόνας, ο ρυθμός ανανέωσης, η αντίθεση και ο φωτισμός της οθόνης. Τέλος τα συστήματα πλήρης εμβύθισης είναι τα πιο απαιτητικά όσον αφορά την ισχύ και την τεχνολογία του υλικού του υπολογιστή, και ως εκ τούτου η πιο ακριβή επιλογή, για την επίτευξη ενός ικανοποιητικού επιπέδου ρεαλισμού.



Εικόνα 3 VR πλήρης εμβύθισης (Poetker, B., 2019.).

2. Η ΙΣΤΟΡΙΑ ΤΗΣ ΕΙΚΟΝΙΚΗΣ ΠΡΑΓΜΑΤΙΚΟΤΗΤΑΣ

2.1 1838 Stereoscopic photos.

Το 1838, μια μελέτη από τον Charles Whetstone (Άγγλος επιστήμονας 1802 - 1875) έδειξε πως κατά την παρακολούθηση δύο δισδιάστατων εικόνων, μια σε κάθε μάτι, ο εγκέφαλος του χρήστη συνδυάζει τις δύο φωτογραφίες σε ένα τρισδιάστατο αντικείμενο. Έτσι, το στερεοσκόπιο, μια διάταξη φακών και δύο φωτογραφίες του ίδιου αντικειμένου που λαμβάνονται από διαφορετικά σημεία συνδυάζονται έτσι ώστε να κάνουν το αντικείμενο να ξεχωρίζει με μια συμπαγή όψη, ο παρατηρητής έχει την αίσθηση του βάθους δημιουργώντας του την ορθή ψευδαίσθηση τριών διαστάσεων. Η μετέπειτα ανάπτυξη του δημοφιλούς στερεοσκοπίου τύπου «View Master» χρησιμοποιήθηκε και για τον εικονικό τουρισμό. Οι αρχές σχεδίασης του στερεοσκοπίου χρησιμοποιούνται ακόμα και σήμερα σε γυαλιά εικονικής πραγματικότητας χαμηλού κόστους που τοποθετούνται στο κεφάλι όπως του δημοφιλούς Google Cardboard. (History Of Virtual Reality - Virtual Reality Society, n.d.)

Εικόνα 4 Charles Whetstone's stereoscope (King's College London, 2016.).

Εικόνα 5 View Master (Spanner, S., 2015.)

Εικόνα 6 Google Cardboard VR Headset (Amos, E., 2015.).

2.2 1929 Link Trainer.

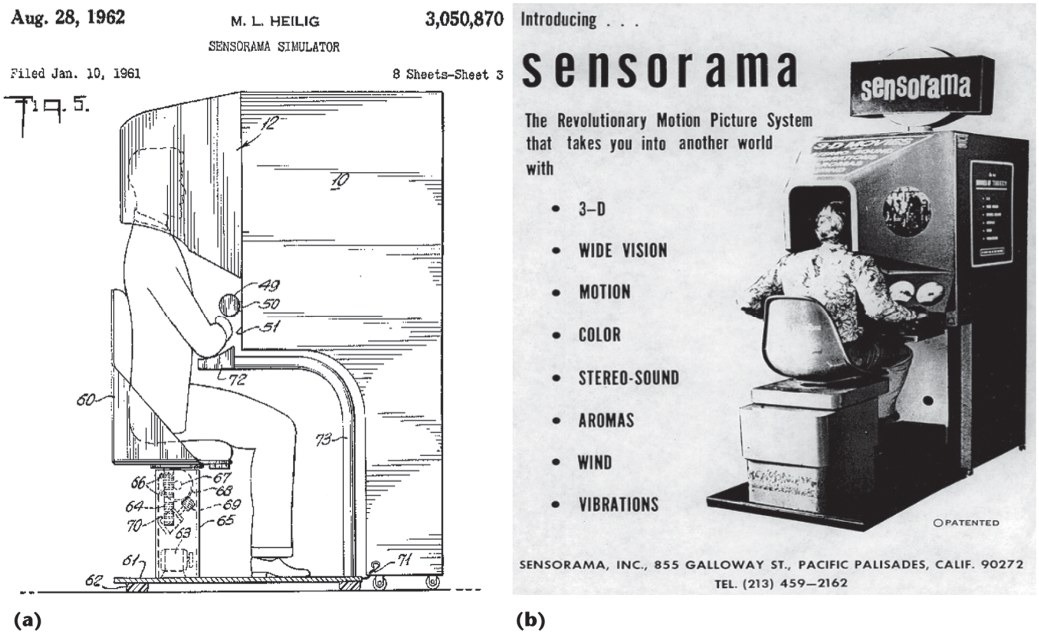
Το Link Trainer, γνωστό και ως «Blue Box» ή «Pilot Trainer», είναι ένας προσομοιωτής πτήσης που κατασκευάστηκε στις αρχές της δεκαετίας του 1930 με βάση την τεχνολογία που εισήγαγε ο Edwin Albert ink έναν χρόνο νωρίτερα. Ορόσημο θα αποτελέσει η επιτυχία όταν ο εκπαιδευτής πιλότων της Link το 1934 θα συμμετάσχει σε άσκηση και θα εκτελέσει με επιτυχία πτήση με χρήση οργάνων μέσω πυκνής νέφωσης που σύμφωνα με την εκτίμηση της πολεμικής Αμερικανικής αεροπορίας ήταν αδύνατη, αποδεικνύοντας την αξία της εκπαίδευσης και μάλιστα σε ασφαλές περιβάλλον. Με την έναρξη του Β'ΠΠ χιλιάδες πιλότοι από διάφορες χώρες της υφηλίου θα εκπαιδευτούν με ασφάλεια και επιτυχία στο «Blue Box» (Εικ. 7-8) που πήρε το όνομα από την χαρακτηριστικά βαμμένη μπλε άτρακτο (History Of Virtual Reality - Virtual Reality Society, n.d. ; Lowood et al., 2021 ; Link Trainer | flight simulator, 2018 ; Link Trainer Flight Simulator, n.d.).

Εικόνα 7 Link Trainer interior (n.d., THE DEVELOPMENT OF INSTRUMENT FLIGHT TRAINERS.).

Εικόνα 8 Link Trainer interior (n.d., THE DEVELOPMENT OF INSTRUMENT FLIGHT TRAINERS.).

2.3 1950 Sensorama.

Την δεκαετία του 1950 ο κινηματογραφιστής Morton Heilig, άρχισε την ανάπτυξη μιας συσκευής συνδυασμού τεχνολογιών με σκοπό την διέγερση αισθήσεων του χρήστη, που αποκαλούσε «Sensorama», ένας σταθερός θάλαμος παρατήρησης με στερεοφωνικό ήχο, γεννήτριες οσμών, ανεμιστήρες για προσομοίωση ανέμου, στερεοσκοπική οθόνη για προβολή έγχρωμου video και μια δονούμενη καρέκλα. Αν και η εφεύρεση δεν είχε την δυνατότητα να αλληλεπιδρά με τον χρήστη-θεατή ήταν μια επαναστατική προσπάθεια να δημιουργήσει μια καθηλωτική εμπειρία αισθήσεων, μια προσπάθεια ικανή ώστε να λάβει ο εμπνευστής της ανεπίσημα τον τίτλο του «Πατέρα της εικονικής πραγματικότητας». Θεωρείται το πρώτο μηχάνημα VR, κατοχυρώθηκε με δίπλωμα ευρεσιτεχνίας το 1962 (Εικ.9) (History Of Virtual Reality - Virtual Reality Society, n.d. ; Corning, 2018).

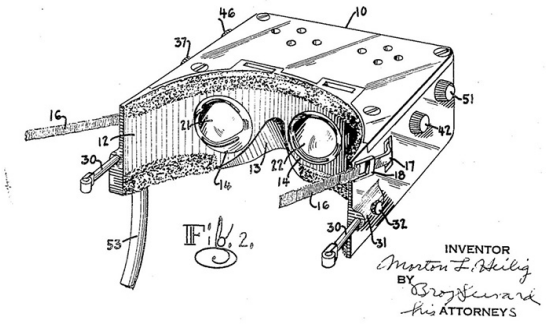


Εικόνα 9 Sensorama Simulator Device (Fattouh, L., 2017.)

2.4 1960 Telesphere Mask.

Μετά την εφεύρεση του «Sensorama» το 1960 από τον ίδιο δημιουργό, ακολούθησε αυτή της «Telesphere Mask» μια συσκευή μη διαδραστικού μέσου παρακολούθησης φιλμ με την διαφορά ότι που ήταν το πρώτο παράδειγμα οθόνης τοποθετημένη στο κεφάλι του θεατή (Head Mounted Display – HMD) που παρείχε στερεοσκοπική 3D ευρυγώνια εικόνα με στερεοφωνικό ήχο (History Of Virtual Reality - Virtual Reality Society, n.d.).

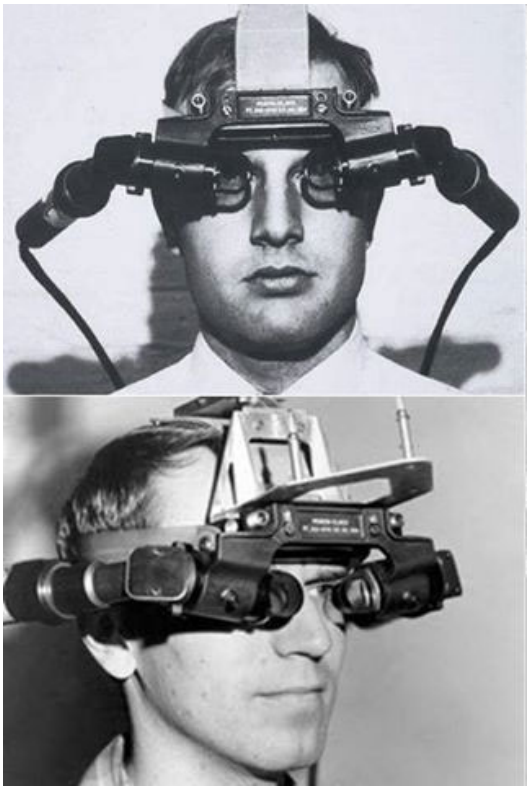
Εικόνα 10 Telesphere Mask (Ismail, A., 2022.)



Εικόνα 11 A Drawing of Telesphere Mask Patent (Ismail, A., 2022.)

2.5 1961 Headsight – First motion tracking HMD.

Το 1961, δυο μηχανικοί της εταιρίας Philco ανέπτυξαν το «Headsight», τον πρόδρομο του HMD όπως το γνωρίζουμε σήμερα. Αυτό περιλαμβάνει μια οθόνη για το κάθε μάτι και ένα μαγνητικό σύστημα παρακολούθησης της κίνησης του χρήστη συνδεδεμένο σε μια κάμερα κλειστού κυκλώματος. Το Headsight δεν αναπτύχθηκε πραγματικά για εφαρμογές εικονικής πραγματικότητας (ο όρος αυτός δεν υπήρχε εκείνη την εποχή) αλλά για να επιτρέπει στο στρατό την παρατήρηση και λήψη αποφάσεων εξ αποστάσεως σε καταστάσεις υψίστης επιχειρησιακής προτεραιότητας. Με αυτήν την εφεύρεση παρά την έλλειψη της ενσωμάτωσης υπολογιστή και της δημιουργίας εικόνας τοποθετήθηκε ο ακρογωνιαίος λίθος για την εξέλιξη της εικονικής πραγματικότητας (History Of Virtual Reality - Virtual Reality Society, n.d. ; Corning, 2018 ; HMD – History and objectives of inventions, 2014).



Εικόνα 12 Headsight Demonstration (2018).

2.6 1965 The ultimate display.

Ο Ivan Sutherland περιέγραψε μια ιδέα το «Ultimate Display» όπου θα μπορούσε να προσομοιώσει την πραγματικότητα σε τέτοιο βαθμό που κανείς δεν θα μπορούσε να διακρίνει τη διαφορά μεταξύ της φυσικής και μη πραγματικότητα. Αυτή η ιδέα περιελάμβανε έναν εικονικό κόσμο που τα στοιχεία του προβάλλονται μέσω ενός Head Mounted Display που αναπαρήγαγε ρεαλιστικά μέσω του επαυξημένου τρισδιάστατου ήχου και της οπτικής ανάδρασης, υλισμικό υπολογιστή για την δημιουργία του εικονικού κόσμου και την διατήρηση του σε πραγματικό χρόνο και την δυνατότητα των χρηστών να αλληλεπιδρούν με ρεαλιστικό τρόπο με τα αντικείμενα του εικονικού κόσμου. Η φιλόδοξη ιδέα συνοψίζεται στον λόγο που συνόδευσε ο ίδιος στην παρουσίαση του: *«Η απόλυτη οθόνη θα ήταν, φυσικά, ένα δωμάτιο μέσα στο οποίο ο υπολογιστής μπορεί να ελέγξει την ύπαρξη της ύλης. Μια καρέκλα που εκτίθεται σε ένα τέτοιο δωμάτιο θα ήταν αρκετά καλή για να καθίσει μέσα. Οι χειροπέδες που εμφανίζονται σε ένα τέτοιο δωμάτιο θα ήταν περιοριστικές και μια σφαίρα που εμφανίζεται σε ένα τέτοιο δωμάτιο θα ήταν θανατηφόρα. Με κατάλληλο προγραμματισμό, μια τέτοια οθόνη θα μπορούσε κυριολεκτικά να είναι η χώρα των*

θαυμάτων στην οποία περπάτησε η Αλίκη.» (History Of Virtual Reality - Virtual Reality Society, n.d.).

"The ultimate display would, of course, be a room within which the computer can control the existence of matter. A chair displayed in such a room would be good enough to sit in. Handcuffs displayed in such a room would be confining, and a bullet displayed in such a room would be fatal. With appropriate programming such a display could literally be the Wonderland into which Alice walked." – ***Ivan Sutherland.***

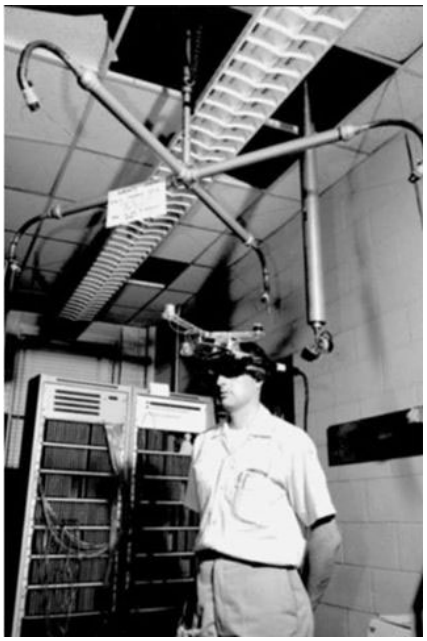
2.7 1966 Furness' Flight Sim.

Στα τέλη του 1960 με την αυξανόμενη πολυπλοκότητα της τεχνολογίας των μαχητικών αεροσκαφών ένας μηχανικός του στρατού ονόματι Tom Furness απέκτησε κίνητρο ώστε να επικεντρωθεί σε λύση προβλημάτων που υπήρχαν όπως η συγκέντρωση και αναπαράσταση κρίσιμων πληροφοριών από αισθητήρες απεικόνισης σε εικονικές οθόνες στις οποίες ο πιλότος έπρεπε να έχει άμεση πρόσβαση ώστε να προβεί τάχιστα σε ενέργειες παράλληλες μεταξύ τους, παράδειγμα να μπορεί να χρησιμοποιήσει το κεφάλι του για να στοχεύει ενώ πυροβολεί, να βλέπει ταυτόχρονα το ύψος, την ταχύτητα, τον ορίζοντα, εχθρικές απειλές και πλήθος άλλων πληροφοριών, ώστε να κάνει τα συστήματα λιγότερο πολύπλοκα στην κατανόηση και στον χειρισμό. Λόγο του περιορισμένου χώρου στο πιλοτήριο ο Tom στράφηκε στην δημιουργία συστημάτων επαυξημένης πραγματικότητας για την εμφάνιση των πληροφοριών σε ένα εικονικό περιβάλλον. Στον Tom Furness αποδίδεται η έναρξη της ανάπτυξης της σύγχρονης τεχνολογίας συστημάτων προσομοιωτών πτήσης (History Of Virtual Reality - Virtual Reality Society, n.d.).

2.8 1968 Sword of Damocles.

Το σπαθί του Δαμοκλή (Sword of Damocles) ήταν το όνομα του μηχανικού συστήματος παρακολούθησης και όχι της οθόνης που τοποθετείται στο κεφάλι, και θεωρείται ευρέως ως το πρώτο σύστημα επαυξημένης πραγματικότητας Head Mounted Display. Δημιουργήθηκε το 1968 από τον επιστήμονα υπολογιστών Ivan Sutherland με την βοήθεια μαθητών του, και πρακτικά ήταν η υλοποίηση της ιδέας Ultimate Display. Η συσκευή ήταν πρωτόγονη τόσο από πλευράς διεπαφής χρήστη όσο και από πλευράς γραφικών που ήταν απλοϊκά δωμάτια wireframe. Το σύστημα εμφάνιζε την έξοδο από

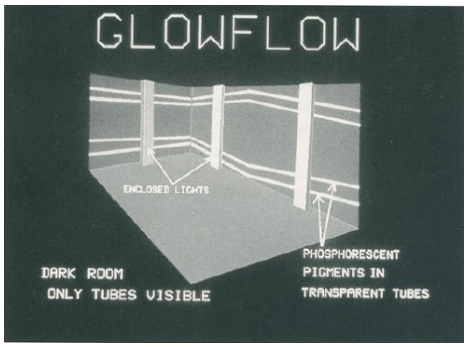
ένα πρόγραμμα υπολογιστή στην στερεοσκοπική οθόνη. Η προοπτική που έδειχνε το λογισμικό ήταν συνάρτηση της θέσης του βλέμματος του χρήστη για αυτό και ήταν απαραίτητη η μετακίνηση του κεφαλιού. Το Head Mounted Display έπρεπε να είναι συνδεδεμένο σε έναν μηχανικό βραχίονα αναρτημένο από την οροφή του εργαστηρίου αφενός μεν λόγω του βάρους αφετέρου δε για την παρακολούθηση των κινήσεων του κεφαλιού μέσο συνδέσεων. Κατά την χρήση της συσκευής ο χρήστης έπρεπε να έχει το κεφάλι του στερεωμένο πάνω της για την εκτέλεση των πειραμάτων. Στο αρχικό στάδιο τα εξαρτήματα που ελέγχονταν δεν ήταν πλήρως ενσωματωμένα μεταξύ τους. Στα τέλη της δεκαετίας άρχισε η ενσωμάτωση των εξαρτημάτων της συσκευής σε ένα ενιαίο σύστημα HMD και με την ολοκλήρωση της ενσωμάτωσης δημιουργήθηκε και τέθηκε σε λειτουργία το πρώτο πλήρως λειτουργικό σύστημα HMD. Η πρώτη προβολή εφαρμογής ήταν ένας κύβος που κρεμόταν στον αέρα μπροστά από τον χρήστη. Το ίδιο το σύστημα αποτελούνταν από έξι υποσυστήματα, ένα διαχωριστικό αποκοπής, πολλαπλασιαστή μήτρας, γεννήτρια διανυσμάτων, ακουστικά, αισθητήρα θέσης κεφαλής και έναν υπολογιστή γενικής χρήσης. Τα προηγούμενα απάρτια θα αποτελέσουν τεχνικά στοιχεία συγκροτήματος της πρώτης μηχανής εικονικής πραγματικότητας όπως τα γνωρίζουμε σήμερα. Το συγκρότημα ήταν εν μέρει διαφανής, έτσι ο χρήστης της συσκευής δεν ήταν τελείως αποκομμένος από το φυσικό του περιβάλλον, αυτή η ημιδιαφάνεια σε συνδυασμό με τα υπόλοιπα χαρακτηριστικά είναι ο λόγος που το σύστημα αναφέρεται πολλές φορές και ως πρόδρομος της τεχνολογίας της επαυξημένης πραγματικότητας (History Of Virtual Reality - Virtual Reality Society, n.d. ; Corning, 2018 ; HMD – History and objectives of inventions, 2014 ; The Sword of Damocles (1968) - The Complete History of VR, n.d.).



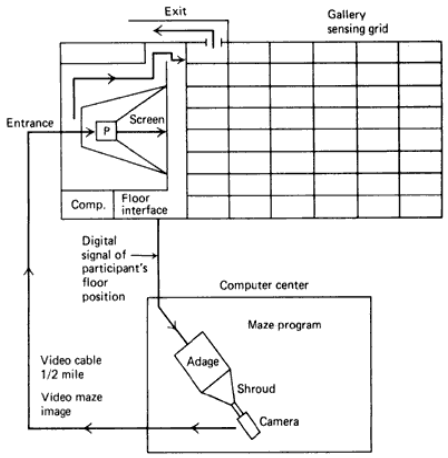
Εικόνα 13 Sword Of Damocles (2018).

2.9 1969 Artificial Reality

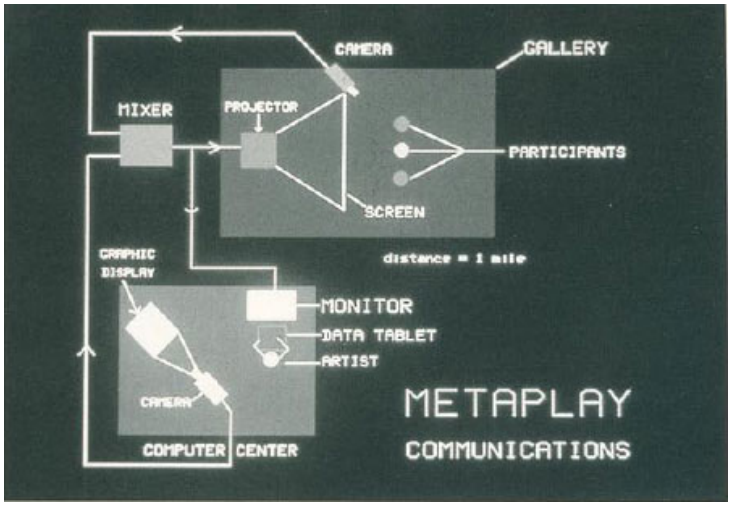
Ξεκινώντας το 1969 ο Myron Krueger από το πανεπιστήμιο του Wisconsin δημιούργησε μια σειρά από έργα σχετικά με την φύση της ανθρώπινης πραγματικότητας σε εικονικά περιβάλλοντα τα οποία αργότερα ονόμασε τεχνητή πραγματικότητα. Τα έργα του με το όνομα GLOWFLOW METAPLAY και PSYCHIC SPACE ήταν πρόοδοι στην έρευνά του που τελικά επέτρεψαν την ανάπτυξη της τεχνολογίας VIDEOPLACE το 1975. Το σύστημα αυτό μπορούσε να αναλύσει και να επεξεργαστεί τις ενέργειες του χρήστη στον πραγματικό κόσμο και να τις μεταφράσει σε αλληλεπιδράσεις με τα εικονικά αντικείμενα με διάφορους προ-προγραμματισμένους τρόπους. Το VIDEOPLACE διέφερε σε πολλές πτυχές από τις προσομοιώσεις εκπαιδευτικού χαρακτήρα και έρευνας. Συγκεκριμένα το σύστημα αντέστρεψε την έμφαση από τον χρήστη που αντιλαμβάνεται τον κόσμο που δημιουργείται από τον υπολογιστή στον υπολογιστή που αντιλαμβάνεται τις ενέργειες σε συνθέσεις αντικειμένων και χώρου στον εικονικό κόσμο. Με την μετατόπιση αυτή της έμφασης ο Krueger διαπίστωσε πως η πιστότητα αναπαράστασης έγινε λιγότερο σημαντική από τις αλληλεπιδράσεις μεταξύ του χρήστη και του εικονικού κόσμου όπως και από την ταχύτητα απόκρισης σε εικόνες ή άλλες μορφές αισθητηριακής εισόδου (History Of Virtual Reality - Virtual Reality Society, n.d. ; Lowood et al., 2021).



Εικόνα 14 Glowflow. n.d..



Εικόνα 15 Psychic Space. n.d.



Εικόνα 16 Metaplay. n.d.

2.10 1972 GE Digital Flight Simulator

Η General Electric παρήγαγε το πρώτα συστήματα παραγωγής εικόνας υπολογιστή (Computer Generated Images – CGI) για το διαστημικό πρόγραμμα. Πρώιμες εκδόσεις αυτών των συστημάτων παρήγαγαν μια εικόνα «εδαφικού επιπέδου» ενώ μεταγενέστερα συστήματα μπορούσαν να δημιουργήσουν εικόνες αντικειμένων τριών διαστάσεων. Η πρόοδος σε αυτόν τον τομέα ήταν ραγδαία και στενά συνδεδεμένη με την εξέλιξη στο υλισμικό του ψηφιακού υπολογιστή σε συνδυασμό με την παράλληλη ανάπτυξη που έλαβε χώρα σε συστήματα παραγωγής εικόνων. Ο προσομοιωτής αυτός διέθετε τρεις οθόνες διατεταγμένες σε διαμόρφωση 180° έτσι ώστε οι οθόνες να περιβάλλουν το προσομοιωμένο εκπαιδευτικό πιλοτήριο για να δώσουν στους εκπαιδευόμενους πιλότους μια αίσθηση αληθινής εμβύθισης (History Of Virtual Reality - Virtual Reality Society, n.d. ; Page, 2004).

2.11 1979 McDonnel-Douglas HMD

Όπως προαναφέρθηκε η αυξημένη πολυπλοκότητα λόγο της ραγδαίας εξέλιξης των μαχητικών αεροσκαφών οδήγησε τον στρατό να πειραματιστεί με Head Mounted Display με σκοπό την επίλυση προβλημάτων. Βασική προϋπόθεση της συσκευής ήταν η προβολή πληροφοριών απευθείας στα μάτια του πιλότου έχοντας στόχο να εξαλειφθούν οι ογκώδεις οθόνες και τα συστήματα προβολής. Ένα από τα πρώτα παραδείγματα είναι το κράνος VITAL του McDonnell Douglas, χρησιμοποιούσε έναν ηλεκτρομαγνητικό ανιχνευτή κίνησης του κεφαλιού για να αντιληφθεί που κοιτούσε ο πιλότος του αεροσκάφους. Δίπλα από τα αυτιά του πιλότου τοποθετήθηκαν διπλοί μονοχρωματικοί σωλήνες καθόδου προβάλλοντας την εικόνα σε διαχωριστές δέσμης μπροστά στα μάτια του πιλότου. Αυτό επιτρέπει στον πιλότο να δει και να χειριστεί τα μηχανικά χειριστήρια στο πιλοτήριο, ενώ έβλεπε την εικόνα του έξω κόσμου που δημιουργήθηκε από υπολογιστή. Ωστόσο τα προβλήματα με τα ογκώδη καλύμματα κεφαλής και ο τρόπος θέασης μέσω των σωλήνων περιόρισαν την αποδοχή αυτών των πρώιμων προσπαθειών (History Of Virtual Reality - Virtual Reality Society, n.d. ; The Complete History of VR – Part 4: Military Purposes, n.d.).

Εικόνα 17 VITAL Helmet n.d.

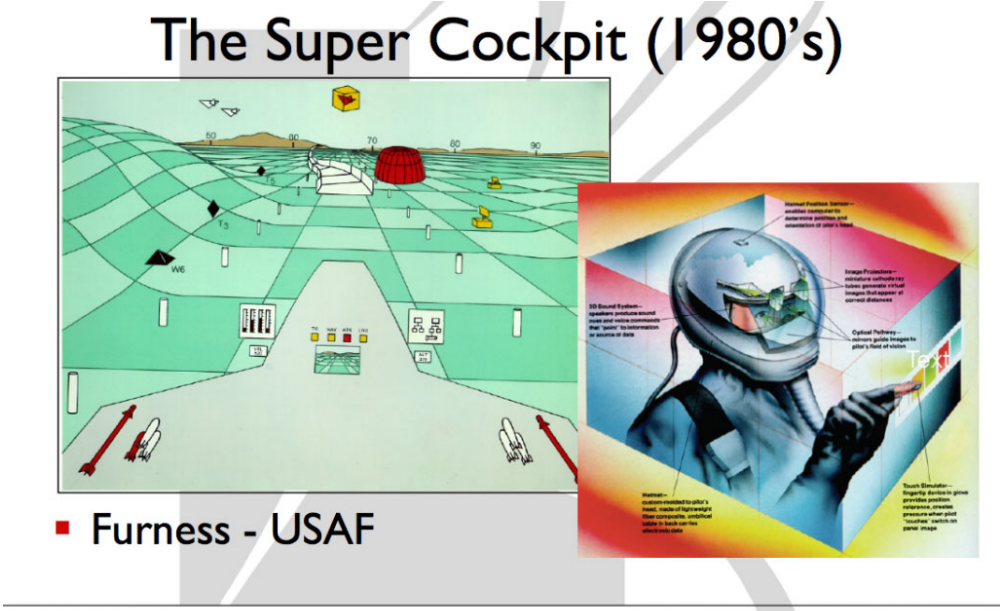
2.12 1982 Sayre Gloves.

Το 1977 οι Daniel J. Sandin και Thomas Defanti στο εργαστήριο Electronic Visualization Laboratory δημιούργησαν το πρώτο ενσύρματο γάντι ή γάντι δεδομένων ονόματι Sayre Glove, βασίστηκε στην ιδέα ενός συνεργάτη τους στο εργαστήριο, του Richard Sayre και η κύρια χρήση των γαντιών αυτών ήταν κυρίως για τον χειρισμό ρυθμιστικών.. Η κατασκευή ήταν ελαφριά και χαμηλού κόστους. Σκοπός ήταν η παρακολούθηση των κινήσεων των χεριών και η παροχή μιας αποτελεσματικής μεθόδου για πολυδιάστατο έλεγχο. Οι συσκευές αυτές χρησιμοποιούν αισθητήρες φωτός με εύκαμπτους σωλήνες, στο ένα άκρο υπάρχει η πηγή φωτός ενώ στην άλλη πλευρά ένα φωτοκύτταρο. Καθώς τα δάχτυλα ήταν λυγισμένα η ποσότητα φωτός που έφτανε στα φωτοκύτταρα ποικίλλει, παρέχοντας έτσι ένα μέτρο της κάμψης των δακτύλων (History Of Virtual Reality - Virtual Reality Society, n.d. ; Norman, n.d. ;DeFanti and Sandin, 1977).

Εικόνα 18 Sayre Glove 1977.

2.13 1986 Super Cockpit.

Από το 1986 έως το 1989, ο Furness διηύθυνε με επιτυχία το πρόγραμμα Super Cockpit της πολεμικής αεροπορία. Η βασική ιδέα του έργου ήταν ότι η ικανότητα των πιλότων να χειρίζονται χωρικές πληροφορίες εξαρτάται από το ότι αυτά τα δεδομένα απεικονίζονται με τρόπο που εκμεταλλεύεται τους φυσικούς αντιληπτικούς μηχανισμούς του ανθρώπου. Εφαρμόζοντας το HMD σε αυτόν τον στόχο, ο Furness σχεδίασε ένα σύστημα που πρόβαλλε πληροφορίες όπως τρισδιάστατους χάρτες που δημιουργούνται από υπολογιστή, εικόνες υπερύθρων και ραντάρ αεροηλεκτρονικής σε έναν καθηλωτικό, τρισδιάστατο εικονικό χώρο που μπορούσε ο πιλότος να δει και να ακούσει σε πραγματικό χρόνο. Το σύστημα παρακολούθησης του κράνους, τα χειριστήρια που ενεργοποιούνται με φωνή και οι αισθητήρες επέτρεψαν στον πιλότο να ελέγχει το αεροσκάφος με χειρονομίες εκφωνήσεις και κινήσεις του βλέμματος μετατρέποντας τη βύθιση σε έναν εικονικό χώρο γεμάτο δεδομένα σε τρόπους ελέγχου. Αυτή η διεπαφή μείωσε την πολυπλοκότητα και τον αριθμό των χειριστηρίων στο πιλοτήριο (History Of Virtual Reality - Virtual Reality Society, n.d. ; #245: 50 years of VR with Tom Furness: The Super Cockpit, Virtual Retinal Display, HIT Lab, & Virtual World Society | Voices of VR Podcast, 2015).



Furness, T. A. (1986, September). The super cockpit and its human factors challenges. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* (Vol. 30, No. 1, pp. 48-52). SAGE Publications.

Εικόνα 19 1986. The Super Cockpit and its Human Factors Challenges.

2.14 1987 Virtual Reality, η δημιουργία του ονόματος.

Το 1985 οι πρωτοπόροι της εικονικής πραγματικότητας Jaron Lanier και thomas Zimmerman ίδρυσαν το εργαστήριο οπτικού προγραμματισμού (Virtual Programming Lab - VPL). Ακόμη και μετά από όλη αυτή την εξέλιξη στην εικονική πραγματικότητα, δεν υπάρχει ακόμα ένας όρος που να περιέγραφε αυτό το πεδίο. Όλα αυτά άλλαξαν το 1987 όταν ο Jaron Lanier επινόηση τον όρο εικονική πραγματικότητα. Μέσω της έρευνας της εταιρείας ο Jaron ανέπτυξε μια σειρά από εξοπλισμό εικονικής πραγματικότητα συμπεριλαμβανομένου του dataglove και της οθόνης EyePhone που τοποθετείται στο κεφάλι. Αξιοσημείωτο είναι το γεγονός πως ήταν η πρώτη εταιρεία VR που πουλά HMD και γάντια δεδομένων όρος που προέρχεται από το προϊόν της εταιρείας το Data Glove (History Of Virtual Reality - Virtual Reality Society, n.d. ; VPL Research Jaron Lanier - Virtual Reality Society, n.d. ; Sorene, 2014).

2.15 1989 NASA Gets Into VR.

Μία από τις πρώτες συσκευές VR κατασκευάστηκε από τον Οργανισμό εξερεύνησης του διαστήματος (NASA). Αναπτύχθηκε και δημιουργήθηκε με το χέρι από μια ομάδα επιστημόνων και μηχανικών σε ένα ερευνητικό κέντρο της NASA στην Καλιφόρνια σε συνδυασμό με την Crystal River Engineering που βοήθησε στην

κατασκευή εξαρτήματος ήχου. Οι Ερευνητές της NASA επιθυμούσαν για αρκετό καιρό έναν χαμηλού κόστους προσομοιωτή εικονικής πραγματικότητας που θα μπορούσε να χρησιμοποιηθεί ως μέσο εκπαίδευσης για τους υποψήφιους αστροναύτες με στόχο να αλληλεπιδρούν με τα συστήματα των διαστημικών σκαφών χωρίς την ανάγκη κατασκευής ενός ολοκληρωμένου προσομοιωτή με τεράστιο κόστος. Το αποτέλεσμα ήταν παρόμοιο με αυτό των σύγχρονων συσκευών εικονικής πραγματικότητας, διαθέτει μια αυτόνομη οθόνη συνδεδεμένη με ένα σύνολο φακών που εστιάζουν την εικόνα μέσω μια θύρας προβολής στο πίσω μέρος την συσκευής. Η συσκευή με το όνομα σταθμός εργασίας περιβάλλοντος εικονικής πραγματικότητας (Virtual Interface Environment Workstation - VIEW) σε αντίθεση με προηγούμενες συσκευές ψευδό VR, το VIEW έχει ενσωματωμένα γραφήματα υπολογιστή και αισθητήρες κίνησης για να μιμηθεί όσο το δυνατόν πιο ρεαλιστικά το περιβάλλον που θα μπορούσε να χρησιμοποιήσουν οι εκπαιδευόμενοι αστροναύτες και οι πιλότοι μαχητικών αεροσκαφών για να αντιμετωπίσουν τις εξαιρετικά περίπλοκες μηχανές (History Of Virtual Reality - Virtual Reality Society, n.d. ; Kirk, 2021).

Εικόνα 20 NASA'S VIEW VR (CONSTANTINE, F., n.d.).

2.16 1991 Virtuality Group Arcade Machines.

Ένα πειστικό παράδειγμα της χρήσης της εικονικής πραγματικότητας ως μέσο ψυχαγωγίας είναι αυτό της επιχείρησης W Industries, σχηματίστηκε από τον Jonathan Waldern, έναν από τους πρώτους πρωτοπόρους - ερευνητές της εικονικής πραγματικότητας στα μέσα του 1980 όποτε και έφτιαξε τα αρχικά πρωτότυπα των οποίων η σύνθεση βασίστηκε σε τεχνολογία μοντέρνα αλλά ταυτόχρονα ξεπερασμένη λόγω της πλαστικής κατασκευής της οθόνης που κάλυπτε το κάθε μάτι για να δίνει την ψευδαίσθηση της 3D εικόνας. Παρόλο την αύξηση ενδιαφέροντος για τους εικονικούς του χώρους αυτό που δεν κατάφερε να συγκεντρώσει ήταν χρήματα διότι οι εταιρίες ήταν ακόμα διστακτικές στο να επενδύσουν σε μια τέτοια τεχνολογία λόγο του κόστους. Εξαιτίας αυτού ο Waldern δυσκολεύτηκε αρκετά οικονομικά μέχρι που κέρδισε το βραβείο για την «Καλύτερη αναδυόμενη τεχνολογία». Με το χρηματικό έπαθλο του βραβείου και με μια πιο ακόμα πιο σημαντική χρηματική επένδυση από μια μεγάλη εταιρεία αναψυχής και με την έκρηξη ενδιαφέροντος "τεχνολογικής έκρηξης" που δημιουργήθηκε από εταιρείες όπως η VPL, ήταν η κατάλληλη στιγμή να δημιουργηθεί το Virtuality Group. Με το κόστος της κάθε μονάδας να είναι απαγορευτικό, επίκεντρο ήταν η επιχειρηματική και εμπορική εφαρμογή. Αναπόφευκτα ωστόσο το ενδιαφέρον για την εικονική πραγματικότητα ήταν τόσο μεγάλο που δεν άργησε να περάσει το Virtuality στην αγορά των καταστημάτων ψυχαγωγίας το 1991 με την κυκλοφορία νέων μοντέλων ελαφρώς φθηνότερων από τα προηγούμενα. Πιθανότατα η επιτυχία που γνώρισε το Virtuality ώθησε και άλλες μεγάλες εταιρείες τεχνολογίας στο να προσπαθήσουν να κερδίσουν χρήματα και στον ανταγωνισμό εξοπλισμού εικονικής πραγματικότητας αλλά αυτό που ήταν ξεκάθαρο ήταν πως η εικονική πραγματικότητα ήταν το απίστευτα επικερδές μέλλον της τεχνολογίας. Παρόλα αυτά, με την αλματώδης τεχνολογική εξέλιξη των υπολογιστών στις αρχές της δεκαετίας του '90 που άφησε το νέο Virtuality να υπολειτουργεί συγκριτικά με άλλες κονσόλες που μάλιστα ήταν φθηνότερες και συνδυαστικά με το γεγονός ότι το κοινό ήθελε πολλά περισσότερα από απλά τρισδιάστατα παιχνίδια δεκαετίας ήταν κάποιοι από τους λόγους που οδήγησαν στο τέλος του Virtuality. Μολονότι τις προσπάθειες να κρατηθεί ζωντανό το Virtuality μέσα από πολλές συνεργασίες με άλλες εταιρείες, έχοντας χάσει τόσο το τεχνολογικό πλεονέκτημα όσο και την ελκυστικότητα του σαν τεχνολογία του μέλλοντος, το 1997 κήρυξε πτώχευση

(History Of Virtual Reality - Virtual Reality Society, n.d. ; Virtuality – A New Reality of Promise, Two Decades Too Soon - Virtual Reality Society, n.d.).

Εικόνα 21 Virtuality. 2022.

2.17 1993 SEGA announce new VR glasses.

Το 1993 και ενώ πολλές εταιρείες προσπαθούσαν να εκμεταλλευτούν τα οικονομικά οφέλη από την εμπορευματοποίηση υπηρεσιών της εικονικής πραγματικότητας η SEGA, μια εταιρία ανάπτυξης και διανομής ηλεκτρονικών παιχνιδιών, κυκλοφόρησε το Genesis, μια συσκευή τύπου κονσόλας που αν και δεν ξεπερνούσε ποιοτικά σε σχέση με αυτή του Virtuality και άλλων μηχανημάτων και παρά τα εμφανή μειονεκτήματα που υπήρχαν τόσο στο αυξημένο βάρος όσο και στην άνεση του χειρισμού αυτές οι μηχανές στην πραγματικότητα ανταποκρίθηκαν πολύ καλά στην υπόσχεση της εικονικής πραγματικότητας στοχεύοντας στην προσφορά της κονσόλας σε πολύ προσιτή για την αγορά τιμή. Έχοντας κατακτήσει ευρέως την αγορά σε περιοδικά βιντεοπαιχνιδιών και ηλεκτρονικών η SEGA επρόκειτο να έχει μια μεγάλη κυκλοφορία προϊόντος. Παρόλα αυτά η SEGA λάμβανε πολλές αρνητικές κριτικές από ομάδες ειδικών κριτικών εκθέτοντας ως πρόβλημα τις συχνές παρενέργειες εμφάνισης του αισθήματος της ναυτίας, θέμα που αντιμετωπίζετε ακόμα και σήμερα τα σύγχρονα συστήματα VR. Με αναφορές να κάνουν λόγο πως οι συσκευές αυτές αρρωσταίνουν τα παιδιά προκαλώντας ναυτία και πονοκέφαλο η εταιρεία οδηγήθηκε στο να ακυρώσει το έργο απορρίπτοντας τους ισχυρισμούς, υποστηρίζοντας ωστόσο πως λόγος αυτής της απόφασης είναι ότι το SEGA VR-1 ήταν

τόσο ρεαλιστικό που θα μπορούσε να προκαλέσει αποπροσανατολισμό και σύγχυση στους χρήστες λόγω της υπερβολικής δραστήριας κίνησης (History Of Virtual Reality - Virtual Reality Society, n.d.; Wiltz, 2019; Sega VR - Sega Retro, 2022).

Εικόνα 22 Sega VR. 2022.

2.18 2012 The Oculus Kickstarter.

Το 2012 μια νέα καμπάνια συγκέντρωσης χρημάτων έκανε θραύση στο διαδίκτυο. Με την παρουσίαση της ιδέας και την πρώτη σειρά δοκιμών της συσκευής Oculus rift και των χειριστηρίων Oculus Touch ο Palmer Luckey κατάφερε να συγκεντρώσει 10 φορές τα χρήματα από τον αρχικό στόχο της εκστρατείας. Τα κεφάλαια αυτά επέτρεψαν στην Oculus να προωθήσει 2 μοντέλα προ παραγωγής με σκοπό την χρήση από τους προγραμματιστές. Από τα πρώτα δύο χρόνια της ίδρυσης της εταιρείας υπήρχε ένας ενθουσιασμός από τους προγραμματιστές και τους λάτρεις της εικονικής πραγματικότητας διότι η τεχνολογία γινόταν κάτι περισσότερο από μια συναρπαστική δυνατότητα, γινόταν πραγματικότητα. Με την κυκλοφορία του πρώτου τεχνικού συγκροτήματος το 2013 οι πρώτοι χρήστες μπορούσαν να βιώσουν καθηλωτικό τρόπο χειρισμού της προβολής απλά μετακινώντας το κεφάλι τους. Η συσκευή απαιτούσε τη σύνδεση της σε έναν υπολογιστή ή ένα Mac που θα μπορούσε να τροφοδοτεί την εμπειρία της εικονικής πραγματικότητας. Μέχρι το 2014, η προσθήκη μιας επιπλέον κάμερας για παρακολούθηση προσέφερε μια ακόμα πιο καθηλωτική εμπειρία στην εικονική πραγματικότητα χωρίς την ανάγκη επιπλέον υλικού. Η ταχεία εξέλιξη της τεχνολογίας και των δυνατοτήτων στον χώρο της εικονικής πραγματικότητας ενέπνευσε το Facebook στο να εξαγοράσει την Oculus. Το 2015 το μέλλον της εικονικής πραγματικής φαινόταν λαμπρό με πρόσθετες εταιρείες

να παίρνουν μέρος στον ανταγωνισμό όπως η HTC με το HTC VIVE, δυστυχώς όμως για την HTC η Oculus είχε την χρηματοδότηση του Facebook σε συνδυασμό με την εμπειρία των προγραμματιστών τους. Το 2016 το Oculus Rift ήταν το πρώτο ευρέως γνωστό HMD που έγινε διαθέσιμο στους καταναλωτές σε προσιτή τιμή, και αδιαμφισβήτητα ήταν ο καταλύτης για βελτιώσεις στην αγορά της εικονικής πραγματικότητας. Από την ίδια εταιρεία ακολούθησαν τα μοντέλα Oculus Go, η πρώτη απόπειρα της Facebook να δημιουργήσει μια αυτόνομη συσκευή που θα προσέφερε την εμπειρία της εικονικής πραγματικότητας, το Oculus Quest που παρομοίως με το Rift ήταν αναγκαία η σύνδεση της συσκευής με υπολογιστή με αξιωσημείωτη διαφορά το κόστος που ήταν σχεδόν το μισό κάνοντας την συσκευή ακόμα πιο προσιτή στο κοινό και το Oculus Quest 2 το πιο σύγχρονο αυτόνομο μοντέλο της Oculus που προσφέρει καλύτερες επιδόσεις σε όλους τους τομείς σε σχέση με το προγενέστερο μοντέλο (History Of Virtual Reality - Virtual Reality Society, n.d. ; (Oculus: Complete Guide — History, Products, Founding, and more, 2022)).

Εικόνα 23 Oculus. n.d.

3. ΜΗΧΑΝΕΣ ΠΑΙΧΝΙΔΙΩΝ (GAME ENGINES).

Η μηχανή παιχνιδιών είναι ένα εξειδικευμένο λογισμικό που δίνει την δυνατότητα στους χρήστες να αναπτύξουν ένα τελικό προϊόν γρήγορα και εύκολα. Η πλειονότητα των παιχνιδιών σήμερα έχουν δημιουργηθεί με την χρήση κάποιου game engine. Ο όρος μηχανή χρησιμοποιείται καθώς τα λογισμικά αυτά είναι υπεύθυνα εξ ολοκλήρου για την τροφοδοσία του περιβάλλοντος στα παιχνίδια (MARTIN, 2020).

Η χρήση ενός game engine δίνει την δυνατότητα της προσαρμογής του περιβάλλοντος του παιχνιδιού με όποιον τρόπο επιθυμεί ο χρήστης. Αυτό περιλαμβάνει τα πάντα, από την τοποθέτηση των αντικειμένων στο χώρο του παιχνιδιού, την φυσική του παιχνιδιού, τον φωτισμό, τον ήχο και οτιδήποτε άλλο απαιτείται για την δημιουργία ενός παιχνιδιού (MARTIN, 2020).

3.1 Αρχιτεκτονική Game Engine.

Essential Manager	Reasons
Render Manager Status: Essential	1. Game must *show on-screen* any essential error messages and information. 2. Almost all games feature graphics.
Resource Manager Status: Essential	1. The resource manager processes resources, and the render manager depends on resources. The render manager is essential, and therefore the resource manager is essential. 2. The game must keep a list of all resources for the purposes of memory management and tidy resource unloading.
Scene Manager Status: Essential	1. The scene manager is required to organize the contents of a scene. Without this manager, resources cannot be placed in meaningful arrangements. 2. Scene managers are essential for monitoring game logic.
Input Manager Status: Essential	1. A game depends in part on the input received from the user, and an engine processes user input through the input manager. Therefore, the input manager is an essential manager. 2. User input is necessary to terminate the execution of a game once the gamer has finished playing.
Error Manager Status: Essential	1. Developers cannot be sure their engine is free of all errors through standard debugging techniques. Therefore, an error manager is required to handle errors as they occur. 2. The error manager might need to notify users of errors.

Εικόνα 24 Απαραίτητοι διαχειριστές (Thorn, 2011).

3.1.1 Διαχειριστής Πόρων.

Ο διαχειριστής πόρων διασφαλίζει ότι τα μέσα και τα δεδομένα συμπεριφοράς διαχειρίζονται κατάλληλα και αποτελεσματικά. Τα δεδομένα πολυμέσων αναφέρονται σε όλα τα ψηφιακά μέσα που καθορίζουν την εμφάνιση και τον ήχο ενός παιχνιδιού. Τα δεδομένα συμπεριφοράς καθορίζουν πώς πρέπει να συμπεριφέρεται ένα παιχνίδι κατά την εκτέλεση. Ο διαχειριστής πόρων είναι υπεύθυνος για:

- τον προσδιορισμό και διάκριση μεταξύ όλων των πόρων.
- την φόρτωση και την εκφόρτωση των πόρων από και προς τα αρχεία και από και προς τη μνήμη.
- την διασφάλιση ότι δεν υπάρχουν περισσότερες από μία περιπτώσεις κάθε πόρου στη μνήμη ανά πάσα στιγμή.

3.1.2 Διαχειριστής Απόδοσης.

Εφόσον ο διαχειριστής πόρων φορτώσει τα γραφικά στην μνήμη, ο διαχειριστής απόδοσης είναι υπεύθυνος για την εμφάνιση των γραφικών στην οθόνη. Για να συμβεί αυτό, μια διεργασία προσπελαύνει τον πόρο γραφικών στη μνήμη και τον σχεδιάζει στην οθόνη με τη βοήθεια του υλικού γραφικών. Ο διαχειριστής απόδοσης:

- επικοινωνεί μεταξύ της μηχανής και του υλικού γραφικών.
- αποδίδει πόρους εικόνας από τη μνήμη στην οθόνη.
- ρυθμίζει την ανάλυση.

3.1.3 Διαχειριστής Εισόδου.

Οι διαχειριστές εισόδου είναι απαραίτητοι σε όλα τα παιχνίδια και τις πλατφόρμες, για να επιτρέπουν στους παίκτες να αλληλεπιδρούν με το παιχνίδι, οπότε προκύπτει η ανάγκη για έναν διαχειριστή εισόδου. Σκοπός του διαχειριστή εισόδου είναι να:

- διαβάζει την είσοδο του χρήστη κατά την εκτέλεση από όλες τις συσκευές που δέχεται το παιχνίδι.
- να κωδικοποιεί την είσοδο σε μορφή ανεξάρτητη από τη συσκευή

3.1.4 Διαχειριστής Ήχου.

Ό,τι ισχύει για τον διαχειριστή απόδοσης ισχύει και για τον διαχειριστή ήχου όσον αφορά τους πόρους ήχου. Ως ήχος νοούνται τόσο η μουσική όσο και ο ήχος. Ο διαχειριστής ήχου θα πρέπει να είναι σε θέση να:

- επικοινωνεί μεταξύ της μηχανής και του υλικού ήχου
- ρυθμίζει τα επίπεδα έντασης ήχου

● εφαρμόζει εφέ στον ήχο, όπως εφέ εξασθένισης και ηχούς.

3.1.5 Διαχείριση Σφαλμάτων.

Ένας προγραμματιστής δεν μπορεί ποτέ να είναι απόλυτα σίγουρος ότι το πρόγραμμά του είναι απαλλαγμένο από σφάλματα. Οι προγραμματιστές χρειάζονται έναν τρόπο να εντοπίζουν και να χειρίζονται σφάλματα ή bugs που μπορεί να εμφανιστούν στα παιχνίδια τους. Ο διαχειριστής σφαλμάτων έχει ως σκοπό:

- την ανίχνευση εξαιρέσεων σε χρόνο εκτέλεσης σε ένα παιχνίδι
- το χειρισμό αυτών των εξαιρέσεων και την αποφυγή ξαφνικών καταρρεύσεων του συστήματος
- την καταγραφή των περιστατικών σφαλμάτων σε μια ευανάγνωστη για τον άνθρωπο αναφορά αρχείου

3.1.6 Διαχείριση Σκηνών.

Αυτός ο διαχειριστής είναι ένας μεσάζων μεταξύ όλων των διαχειριστών που καθορίζει τον τρόπο με τον οποίο όλοι οι διαχειριστές συνδέονται μεταξύ τους και χρησιμοποιούνται έτσι ώστε να δώσουν νόημα στον παίκτη. Ο διαχειριστής σκηνής είναι υπεύθυνος για πολλά πράγματα, μερικά από τα οποία περιλαμβάνουν:

- την επικοινωνία μεταξύ πολλών διαχειριστών
- την παρακολούθηση των γεγονότων
- την απαρίθμηση των ενεργών αντικειμένων στο παιχνίδι και τη δυνατότητα πρόσβασης και τροποποίησής τους από τον προγραμματιστή

3.2 Είδη Game Engine.

Υπάρχουν δύο είδη μηχανών που κάποιος μπορεί να χρησιμοποιήσει. Το πρώτο είδος είναι οι καθιερωμένες μηχανές όπως η Unity και η Unreal στις οποίες ο χρήστης λαμβάνει μια άδεια χρήσης, δωρεάν ή επί πληρωμή ανάλογα με τις δυνατότητες που χρειάζεται και τις ανάγκες του. Το δεύτερο είδος είναι μηχανές δημιουργημένες από το μηδέν. Η επιλογή μεταξύ των δύο αυτών ειδών έχει τα θετικά και τα αρνητικά (Maxwell, 2021 ; MARTIN, 2020).

Πλεονεκτήματα χρήσης ανεπτυγμένης μηχανής :

- Εξοικονόμηση πολύ χρόνου.
- Δεν απαιτείται ικανότητα πολύπλοκου προγραμματισμού.
- Δοκιμασμένο λογισμικό και εργαλεία.
- Διαθέσιμη βοήθεια.
- Εύκολη και γρήγορη χρήση περιουσίας τρίτων.

Μειονεκτήματα χρήσης ανεπτυγμένης μηχανής :

- Κόστος χρήσης εκ των προτέρων είτε σε δικαιώματα.
- Πιθανή έλλειψη κάποιας λειτουργικότητας.
- Συμμόρφωση με τους κανόνες της μηχανής.

Πλεονεκτήματα χρήσης μηχανής από το μηδέν:

- Η ύπαρξη μόνο απαραίτητου κώδικα αυξάνει την απόδοση.
- Δεν υπάρχει χρηματικό κόστος για την χρήση.
- Ελευθερία στον έλεγχο.
- Απόλυτη γνώση της λειτουργίας της μηχανής.

Μειονεκτήματα χρήσης μηχανής από το μηδέν:

- Χρονοβόρο στην δημιουργία.
- Απαιτητικό στην μάθηση για την δημιουργία.
- Περιορισμένοι εκπαιδευτικοί πόροι.
- Περιορισμένη βοήθεια σε τυχόν σφάλματα και προβλήματα.

3.3 Unity.

Το Unity είναι μια μηχανή παιχνιδιών τριών/δύο διαστάσεων και ένα ισχυρό ολοκληρωμένο περιβάλλον ανάπτυξης (Integrated Development Environment - IDE) πολλαπλών πλατφόρμων για προγραμματιστές. Αυτό σημαίνει πως το Unity έχει την δυνατότητα να παρέχει πολλές σημαντικές λειτουργίες που κάνουν τα παιχνίδια να λειτουργούν. Αυτό περιλαμβάνει πράγματα όπως η φυσική, η τρισδιάστατη απόδοση αντικειμένων και η ανίχνευση σύγκρουσης. Για τους προγραμματιστές αυτό σημαίνει

πως δεν χρειάζεται να ξεκινήσουν από το μηδέν κατά την ανάπτυξη ενός νέου παιχνιδιού (Sinicki, 2021).

Το IDE είναι μια διεπαφή που επιτρέπει στον χρήστη να έχει πρόσβαση σε όλα τα εργαλεία που χρειάζεται για ανάπτυξη σε ένα μέρος. Το λογισμικό Unity διαθέτει ένα φιλικό προς το χρήστη οπτικό πρόγραμμα επεξεργασίας που επιτρέπει στους δημιουργούς να σύρουν και να εναποθέσουν με ευκολία στοιχεία σε σκηνές και στην συνέχεια να προσαρμόσουν τις ιδιότητες τους (Sinicki, 2021).

Το Unity παρέχει επίσης μια σειρά από άλλες λειτουργίες όπως τη δυνατότητα πλοήγησης στους φακέλους του έργου και δημιουργίας κινούμενων εικόνων μέσω ενός εργαλείου χρονοδιαγράμματος (Sinicki, 2021).

Μια ακόμα μεγάλη δυνατότητα του Unity είναι ότι διαθέτει επίσης ένα κατάστημα εργαλείων όπου οι προγραμματιστές μπορούν να ανεβάσουν τις δημιουργίες τους και να τις κάνουν διαθέσιμες στην κοινότητα είτε δωρεάν είτε επί πληρωμή. Με την μεγάλη ποικιλία εργαλείων τα οποία είναι διαθέσιμα στο κατάστημα αυτό σημαίνει πως ο χρήστης μπορεί να γλιτώσει χρόνο τον οποίο μπορεί να αφιερώσει στην ανάπτυξη της εφαρμογής του (Sinicki, 2021).

Όσον αφορά το προγραμματιστικό κομμάτι η κωδικοποίηση μπορεί να γίνει μέσο ενός editor της επιλογής του χρήστη αντί για τον προεπιλεγμένο editor Visual Studio της Microsoft (Sinicki, 2021).

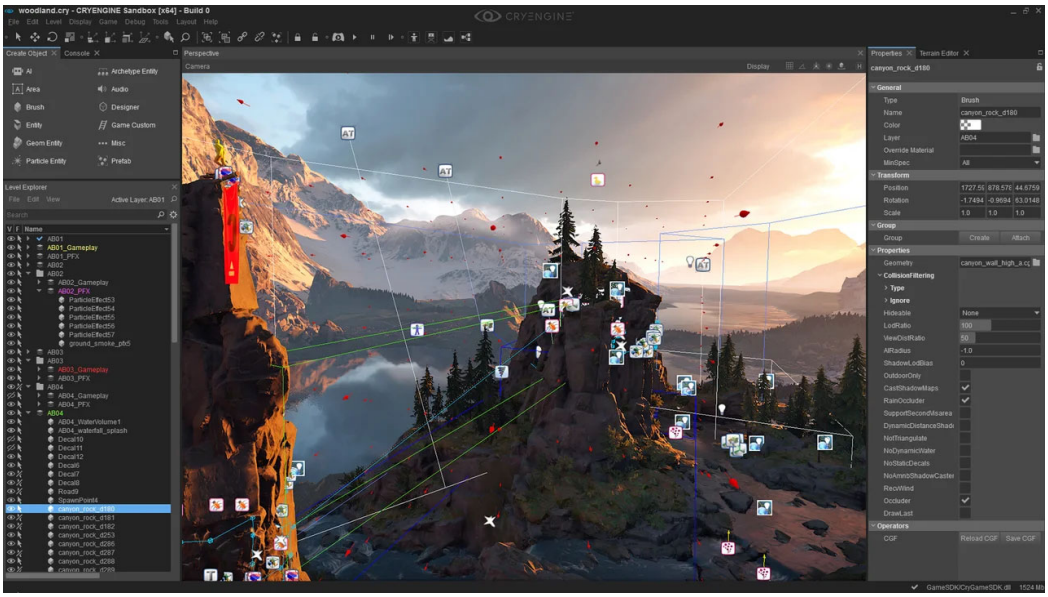
Η Unity έχει καταστήσει σαφές ότι η γλώσσα η C# θεωρείται η κανονική γλώσσα για την ανάπτυξη Unity. Η C# είναι μια ισχυρή γλώσσα που χρησιμοποιείται ευρέως στην βιομηχανία του προγραμματισμό και είναι εύκολη στην εκμάθηση (BUCKLEY, 2021).

Εικόνα 25 Unity. n.d. 2022.

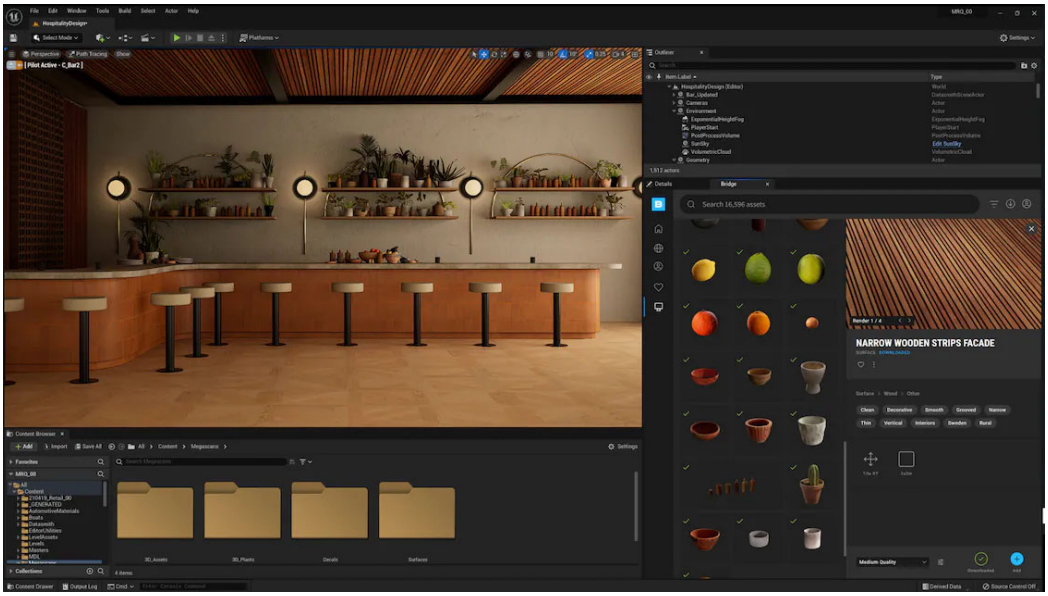
Φυσικά η μηχανή παιχνιδιών Unity είναι μία από τις διαθέσιμες μηχανές για ανάπτυξη. Κάποιες μηχανές από τις οποίες η Unity αντιμετωπίζει ανταγωνισμό είναι η Unreal Engine και η Cryengine. Οι δυνατότητες του Unity δεν είναι τόσο καλές στα High-end γραφικά σε σύγκριση με την Unreal ή την Cryengine. Είναι όμως αξιοσημείωτο πως έχουν γίνει προσπάθειες με πρόσφατες ενημερώσεις για να καλυφθεί αυτό το κενό που έχει η μηχανή Unity (Sinicki, 2021).

Το πλεονέκτημα όμως που έχει η μηχανή Unity συγκριτικά με τις άλλες δύο είναι το γεγονός πως είναι πολύ φιλικότερες στους νεοφερμένους χρήστες. Όσον αφορά την δημιουργία 2D παιχνιδιών το Unity έχει το πλεονέκτημα συγκριτικά με την διάσημη Unreal μηχανή και χρησιμοποιείται τόσο από indie όσο και από πιο μεγάλες εταιρείες. Ένα από τα σημαντικά πλεονεκτήματα που προσφέρει είναι το γεγονός πως με τον τρόπο με τον οποίο διαχειρίζεται τα γραφικά το Unity κάνει εύκολο το porting σε συσκευές με υλικό χαμηλότερων δυνατοτήτων (Sinicki, 2021).

Επίσης επειδή η μηχανή Unity είναι Cross Platform είναι εξίσου εύκολη η δημιουργία εφαρμογών σε iOS, PC και κονσόλες. Τέλος η μηχανή Unity προσφέρει εξαιρετική υποστήριξη VR για τους προγραμματιστές που θέλουν να ασχοληθούν με την ανάπτυξη VR εφαρμογών για συσκευές όπως το Oculus (Sinicki, 2021).



Εικόνα 26 CryEngine 2022.



Εικόνα 27 Unreal Engine 2022.

4. ΧΡΗΣΕΙΣ ΤΗΣ ΕΙΚΟΝΙΚΗΣ ΠΡΑΓΜΑΤΙΚΟΤΗΤΑΣ.

Η έκταση της εικονικής πραγματικότητας αφορά πολλούς τομείς της καθημερινότητας μας τόσο των επιστημών όσο και της συνηθισμένης μας ζωής. Παρακάτω θα μιλήσουμε ενδεικτικά για κάποιους τομείς στους οποίους γίνεται χρήση της εικονικής πραγματικότητας.

4.1 Ιατρική.

Η ιατρική περίθαλψη όπως και οι περισσότεροι τομείς, απαιτούν μια ανανέωση τόσο στον τρόπο εκπαίδευσης των μελλοντικών ιατρών όσο και στην φροντίδα των ασθενών. Η εικονική πραγματικότητα δοκιμάζεται, αναπτύσσεται και χρησιμοποιείται ενεργά στην ιατρική σε όλο τον κόσμο και επιλύει πολλά κρίσιμα ζητήματα. Οι φοιτητές ιατρικής και οι νέοι εργαζόμενοι του τομέα της υγείας επωφελούνται από την χρήση της εικονικής πραγματικότητας βελτιώνοντας την παραγωγικότητά τους στην θεωρητική μάθηση. Οι μαθητές μπορούν να διερευνήσουν ιατρικά μοντέλα 3D σε βάθος, με τρόπο που τα φυσικά μοντέλα δεν μπορούν να αναπαράξουν (Abdelmaged, 2021).

Το VR λογισμικό που δίνει έμφαση στην ρεαλιστική ιατρική κατάρτιση βοηθά τους σπουδαστές μια ιατρικής σχολής και τους εργαζόμενους του ίδιο τομέα στο να αποκτήσουν πρακτική εμπειρία σε περιβάλλον χωρίς κινδύνους. Αυτός ο τύπος εικονικής πραγματικότητας επιτρέπει στην προσομοίωση σεναρίων που διαφορετικά θα ήταν δύσκολο, επικίνδυνο ή ακριβό στο να αναπτυχθεί στην πραγματική ζωή. Στον προσομοιωμένο αυτόν κόσμο ο χρήστης γίνεται μέρος ενός σεναρίου στο οποίο αναμένεται να φέρει εις πέρας συγκεκριμένες εργασίες όπως την διεξαγωγή μια ιατρικής επέμβασης ή την διαχείριση ιατρικού εξοπλισμού. Έτσι υπάρχει η δυνατότητας της αξιολόγησης της απόδοσης του χρήστη με προκαθορισμένα κριτήρια (Abdelmaged, 2021).

Εικόνα 28 Surgical simulation training. n.d.

Η θεραπεία έκθεσης είναι μια τεχνική θεραπείας για διάφορες μορφές ψυχικών ασθενειών. Η θεραπεία έκθεσης περιλαμβάνει την σταδιακή έκθεση του ασθενούς στην πηγή άγχους αρχίζοντας από μία ήπια εκδοχή του ερεθίσματος, για παράδειγμα, στην εικονική πραγματικότητα ένας ασθενής με υψοφοβία μπορεί να έρθει αντιμέτωπος με τον φόβο του σε ασφαλείς συνθήκες και αριθμός των συνεδριών και της διάρκειας της θεραπείας εξαρτάται από την πρόοδο του ασθενούς (Exposure Therapy: Types, How It’s Done, and More, 2021).

Όσον αφορά τις ιατρικές εφαρμογές η εικονική πραγματικότητα βρίσκεται ακόμα σε αρχικό στάδιο. Στο μέλλον πιθανότατα να χρησιμοποιηθεί ευρέως για την βελτίωση της ασφάλειας και της αποτελεσματικότητας των χειρουργικών επεμβάσεων καθώς και για την καλύτερη κατανόηση του ανθρώπινου σώματος (Exposure Therapy: Types, How It’s Done, and More, 2021).

Εικόνα 29 Virtual reality therapy helps people overcome fear of heights. 2018.

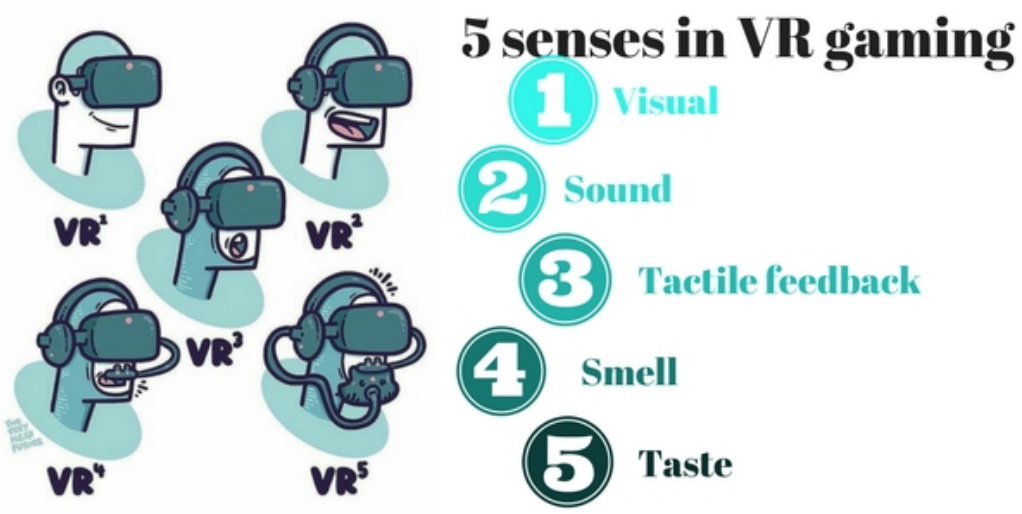
4.2 Ψυχαγωγία.

Τα τελευταία χρόνια το VR παρουσιάζει σημαντικό ρυθμό ανάπτυξης στην ψυχαγωγία και ιδιαίτερα στα βιντεοπαιχνίδια. Αυτό συμβαίνει διότι η εικονική πραγματικότητα προσφέρει στους καταναλωτές κάτι διαφορετικό από αυτό που προσφέρουν οι συμβατικές μορφές ψυχαγωγίας σε τέτοιο βαθμό που δεν μπορούν να συγκριθούν με το VR. Οι χρήστες βιώνουν μια πλήρη καθηλωτική εμπειρία αισθήσεων με θέαση 360°. Όχι μόνο παρακολουθούν το παιχνίδι αλλά έχουν την αίσθηση ότι είναι μέσα σε αυτό (Abdelmaged, 2021).

Το VR προσφέρει καθηλωτική εμπειρία διεγείροντας και τις 5 βασικές αισθήσεις του χρήστη γεγονός που προϋποθέτει την χρήση περεταίρω εξοπλισμού.

1. Όραση: Ο χρήστης μέσω του VR Headset και των ειδικών φακών που υπάρχουν σε αυτό τοποθετείται στο τριών διαστάσεων εικονικό περιβάλλον έχοντας οπτικό πεδίο 360°.
2. Ακοή: Κατά την αλληλεπίδραση του χρήστη στο εικονικό περιβάλλον παράγονται διαφορετικοί ήχοι για να δημιουργήσουν μια καθηλωτική εμπειρία για τον χρήστη. Για την δημιουργία ενός ρεαλιστικού περιβάλλοντος, το σύστημα λαμβάνει υπόψη την απόσταση του αντικειμένου από τον χρήστη, και την κατεύθυνση του χρήστη στο VE.

3. Αφή: Περιλαμβάνει κυρίως δονήσεις ποικίλης έντασης και συχνότητας για μικρές ή μεγάλες περιόδους σύμφωνα με τις ανάγκες του παιχνιδιού. Για παράδειγμα σε ένα παιχνίδι σκοποβολής, ο παίκτης μπορεί να βιώσει έναν τύπο δόνησης στα χέρια όταν πυροβολεί με ένα πιστόλι και άλλον με μία καραμπίνα. Πέρα του βασικού εξοπλισμού των περισσοτέρων VR, οι χρήστες μπορούν με ειδικές στολές να αισθανθούν περισσότερα πράγματα, όπως για παράδειγμα την αίσθηση χτυπήματος από σφαίρα
4. Όσφρηση: Η αισθητηριακή διέγερση της όσφρησης είναι άλλος ένας τρόπος με τον οποίο η εμπειρία του χρήστη μπορεί να επαυξηθεί. Ωστόσο δεδομένου ότι οι συσκευές VR δεν έχουν άρωμα χρησιμοποιούνται άλλα μέσα όπως η δημοφιλή μάσκα της εταιρίας Feelreal που κυκλοφορεί στην αγορά. Αυτή η συσκευή προσφέρει στον χρήστη συνδυασμούς αρωμάτων που προσφέρει ζωντάνια, ρεαλισμό και μια καθηλωτική εμπειρία κατά την χρήση.
5. Γεύση: Η γεύση είναι άλλη μια αίσθηση η οποία μπορεί να διεγερθεί κατά την χρήση του VR. Αν και φαντάζει δύσκολο σαν πράξη επιστήμονες από διάφορα πανεπιστήμια στον κόσμο εργάζονται πάνω στην τεχνολογία της θερμικής γεύσης. Η αίσθηση της γεύσης μέσω αυτής της τεχνολογίας είναι δυνατή χάρης σε ένα κουτί που περιέχει τόσο ηλεκτρικές όσο και θερμικές μονάδες ελέγχου. Οι ηλεκτρικές μονάδες ελέγχου είναι υπεύθυνες για το ξινό, το αλμυρό και το πικρό ενώ οι θερμικές για το γλυκό και το πικάντικο (Abdelmaged, 2021 ; Ninad, 2019 ; The sense of taste in virtual reality, 2017).



Εικόνα 30 Senses in VR gaming. Polycarpou, C., 2018.

4.3 Εκπαίδευση.

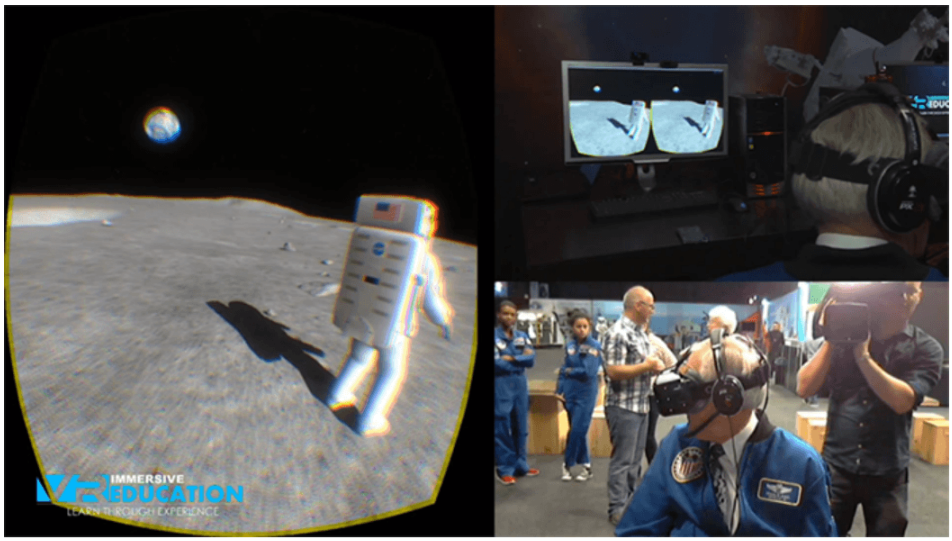
Αν και η γνώση έχει γίνει πιο προσιτή σε περισσότερους ανθρώπους, η τρέχουσα προσέγγιση στην εκπαίδευση έχει δύο σημαντικά προβλήματα:

- Βασίζεται στην απαρχαιωμένη μορφή της απομνημόνευσης γεγονότων. Οι συμβατικές μέθοδοι διδασκαλίας έχουν σχεδιαστεί για να παρέχουν τεράστιες ποσότητες πληροφοριών. Το να έχει κάποιος πρόσβαση και να καταναλώνει πολλές πληροφορίες δεν σημαίνει απαραίτητα πως μορφώνεται το άτομο.
- Οι πολλές πληροφορίες σε μικρό χρονικό διάστημα μπορούν εύκολα να κορέσουν τους μαθητές. Ως αποτέλεσμα οι μαθητές κουράζονται, αποδεσμεύονται από δέκτες πληροφοριών και αναρωτιούνται γιατί μαθαίνουν για το θέμα αυτό εξαρχής (Babich, 2019).

Η εικονική πραγματικότητα μπορεί να χρησιμοποιηθεί για να ενισχύσει τη μάθηση και τη συμμετοχή των μαθητών. Η χρήση του VR στην εκπαίδευση μπορεί να μεταμορφώσει τον τρόπο με τον οποίο η γνώση φτάνει στους μαθητές. Στο εικονικό αυτό περιβάλλον οι μαθητές όχι μόνο βλέπουν τον κόσμο, πραγματικό ή φανταστικό, αλλά μπορούν και να αλληλεπιδρούν με αυτόν (Babich, 2019).

Παρακάτω θα αναφερθούμε σε κάποια χαρακτηριστικά που κάνουν την εικονική πραγματικότητα ένα ισχυρό εργαλείο στην εκπαίδευση.

- Καλύτερη αίσθηση του τόπου: Όταν οι μαθητές διαβάζουν για κάτι νιώθουν την ανάγκη να το βιώσουν οι ίδιοι. Με την εικονική πραγματικότητα οι μαθητές δεν χρειάζεται απλά να το φαντάζονται αλλά μπορούν και να το ζήσουν.
- Οπτική μάθηση: Πολλοί άνθρωποι μαθαίνουν οπτικά. Το VR είναι ένα εργαλείο ιδιαίτερα χρήσιμο για αυτήν την κατηγορία μαθητών. Οι μαθητές αντί να απομνημονεύουν θεωρητικές γνώσεις μπορούν να δούνε τα πράγματα για τα οποία μαθαίνουν.
- Από την θεωρία στην πράξη: Είναι ευρέως γνωστό πως οι μαθητές μαθαίνουν καλύτερα εφαρμόζοντας στην πράξη το δίδαγμα, παρά το γεγονός αυτό η εκπαίδευση επικεντρώνεται στο θεωρητικό κομμάτι αντί για το πρακτικό (Babich, 2019).



Εικόνα 31 Astronaut Career exploration. Babich, N., 2019.

4.4 Τουρισμός.

Ο τουρισμός είναι ένας από τους τομείς που επλήγη σοβαρά από την υγειονομική πανδημία εξαιτίας των περιοριστικών μέτρων που αναγκάστηκαν να λάβουν οι χώρες (Gegung, 2021).

Κατά την διάρκεια των περιοριστικών μέτρων οι άνθρωποι δεν είχαν την δυνατότητα να ταξιδεύουν σε άλλες χώρες και συνεπώς δεν είχαν την δυνατότητα να επισκεφθούν αξιοθέατα και μουσεία (Gegung, 2021).

Με το ερώτημα του πότε οι τουριστικές δραστηριότητες θα επανέλθουν στο φυσιολογικό οι τουριστικές επιχειρήσεις αναζητούσαν μια εναλλακτική λύση. Η χρήση της εικονικής πραγματικότητας στον τουρισμό προσφέρει την δυνατότητα της επίσκεψης μουσείων και αξιοθέατων απομακρυσμένα ξεπερνώντας το εμπόδιο των περιοριστικών μέτρων (Gegung, 2021).

Επιπροσθέτως το VR θα μπορούσε να εμπλουτίσει την εμπειρία των χρηστών, κάτι τέτοιο θα ήταν δυνατό κάνοντας το εικονικό περιβάλλον να είναι σχετικό με τον χώρο που επισκέπτονται. Για παράδειγμα αντί ο χρήστης να βλέπει τον Παρθενώνα στην σημερινή του μορφή θα μπορούσε να περιηγηθεί σε έναν χώρο που θα ήταν πολύ κοντά οπτικά στην εποχή όπου χτίστηκε (Gegung, 2021).

Μια ακόμα σημαντική δυνατότητα που προσφέρει η εικονική πραγματικότητα είναι η απεικόνιση τοποθεσιών και περιοχών που η επίσκεψη τους δεν είναι δυνατή στο κοινό ή η πρόσβαση απαγορεύεται (Gegung, 2021).

Εικόνα 32 Use of VR in tourism. HASSASSIAN, A., 2019.

5. ΣΧΕΔΙΑΣΗ ΚΑΙ ΥΛΟΠΟΙΗΣΗ ΕΦΑΡΜΟΓΗΣ.

Στα πλαίσια της πτυχιακής εργασίας σχεδιάστηκε και υλοποιήθηκε μια εφαρμογή εκπαιδευτικού περιεχομένου για δύο ιστορικά σημεία της Άρτας, το κάστρο της Άρτας και το θεατράκι (μικρό θέατρο Αμβρακίας) σε εικονική πραγματικότητα. Η υλοποίηση της εφαρμογής έγινε σε Unity. Η μηχανή αυτή επιλέχθηκε επειδή είναι ιδανική για αρχάριους, εύκολη στη χρήση και έχει καλύτερη συμβατότητα με φορητές συσκευές όπως το Oculus Quest 2. Εκτός από την καλύτερη συμβατότητα, η συντριπτική πλειοψηφία των προγραμματιστών χρησιμοποιεί την Unity για την ανάπτυξη εφαρμογών VR, γεγονός που διευκολύνει την εξεύρεση λύσεων σε προβλήματα και σφάλματα. Τέλος το Unity παρέχει μια τεράστιο πληθώρα εργαλείων και επεκτάσεων που καθιστούν τον σχεδιασμό και την υλοποίηση γρήγορη και εύκολη.

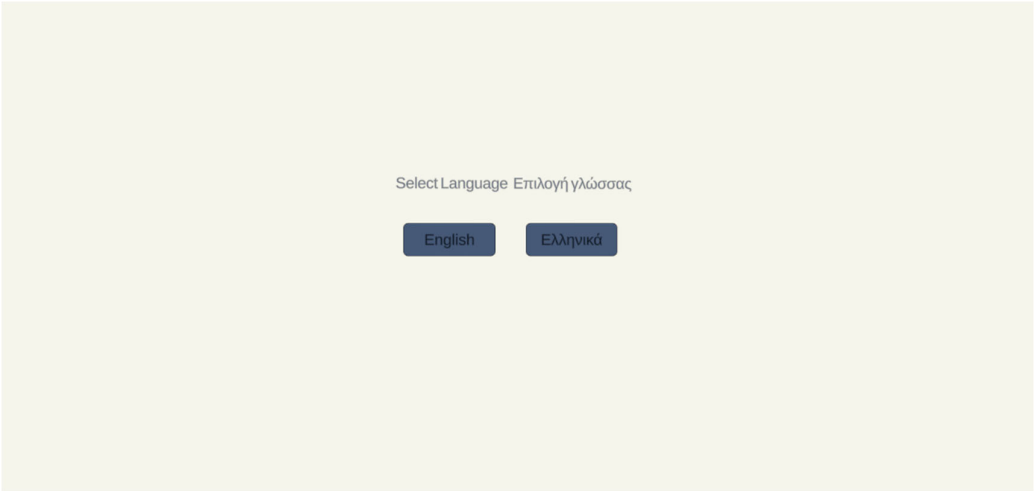
Πρώτο βήμα για την υλοποίηση της εφαρμογής ήταν η συλλογή φωτογραφικού υλικού 360° και πληροφοριών για τα σημεία ενδιαφέροντος τόσο του κάστρου όσο και του μικρού θεάτρου της Άρτας. Με την συλλογή των φωτογραφιών ακολούθησε η επεξεργασία τους με την χρήση λογισμικού επεξεργασίας εικόνων. Στην συνέχεια οι επεξεργασμένες φωτογραφίες προστέθηκαν στο Unity όπου στην περίπτωση του κάστρου μετατράπηκαν σε σφαίρες στις οποίες ο χρήστης βρίσκεται μέσα.

Οι σφαίρες αυτές είναι μια τεχνική η οποία προσφέρει στον χρήστη την ψευδαίσθηση του βάθους. Η τοποθέτηση των σφαιρών στο εικονικό περιβάλλον του Unity έγινε με βάση την πραγματική τους θέση με την βοήθεια ενός χάρτη. Η τοποθέτηση των σφαιρών με βάση την πραγματική τους θέση προσδίδει στην εμπειρία ρεαλισμό, με τους ήχους να ακούγονται πιο δυνατά ή πιο χαμηλά ανάλογα με την απόσταση που έχει ο χρήστης από την πηγή του ήχου.

Εκτός από την χρήστη της μηχανής παιχνιδιών Unity, χρησιμοποιήθηκε το Oculus Quest 2 για την ανάπτυξη της εφαρμογής και την δοκιμή της. Οι συνεχείς δοκιμές της εφαρμογής για την απασφαλμάτωση και την βελτίωση της εφαρμογής ήταν ζωτικής σημασίας. Η δοκιμή της εφαρμογής από άπειρους χρήστες ήταν εξίσου σπουδαία καθώς η απρόβλεπτη χρήση της ήταν σημαντική για την βελτίωση της εμπειρίας του χρήστη.

Επιλογή Γλώσσας.

Κατά την εκκίνηση της εφαρμογής ο χρήστης επιλέγει την γλώσσα που επιθυμεί πατώντας το αντίστοιχο κουμπί. Η επιλογή αυτή καθορίζει την γλώσσα των ηχητικών και των κειμένων της εφαρμογής.



Εικόνα 33 Επιλογή γλώσσας της εφαρμογής.

Προγραμματιστικά για την επιλογή γλώσσας, όπως φαίνεται και στο Παράρτημα 1, γίνεται χρήση του EventSystem το οποίο είναι υπεύθυνο για τον χειρισμό συμβάντων (Event Handler). Κατά το πάτημα ενός εκ των δύο κουμπιών αποθηκεύεται σε μια στατική μεταβλητή η τιμή του η οποία χρησιμοποιείται καθ' όλη την διάρκεια της εφαρμογής. Επιπροσθέτως, κατά το πάτημα του κουμπιού καλείται και ο SceneManager που είναι υπεύθυνος για την διαχείριση των σκηνών (scenes) με σκοπό την αλλαγή της ενεργής σκηνής από την επιλογή γλώσσας στο κύριο μενού.

Κύριο Μενού.

Στην συνέχεια ο χρήστης οδηγείται στο κεντρικό μενού της εφαρμογής. Η εφαρμογή χωρίζεται σε τρία μέρη, την περιήγηση του κάστρου, την gamified έκδοση του κάστρου και το θεατράκι. Η gamified έκδοση του κάστρου έγινε με σκοπό την προσέλκυση του ενδιαφέροντος ατόμων μικρής ηλικίας, έχοντας στόχο την μετάδοση γνώσης με ευχάριστο και διασκεδαστικό τρόπο.

Εικόνα 34 Το κεντρικό μενού της εφαρμογής.

Η εμφάνιση των κειμένων στην επιθυμητή γλώσσα του χρήση επιτυγχάνεται με έναν απλό έλεγχο της στατικής μεταβλητής Language από την οποία καθορίζεται η γλώσσα των κουμπιών και οι εικόνες που θα είναι ενεργές στην σκηνή (Παράρτημα 2).

5.1 Κάστρο της Άρτας.

Το κάστρο της Άρτας χωρίζεται σε 2 σενάρια, το σενάριο της περιήγησης και της παιχνιδοποιημένης (gamified) έκδοσης. Το κάθε σενάριο αποτελείται από ορισμένα σημεία ενδιαφέροντος που παρουσιάζονται παρακάτω:

Περιήγηση

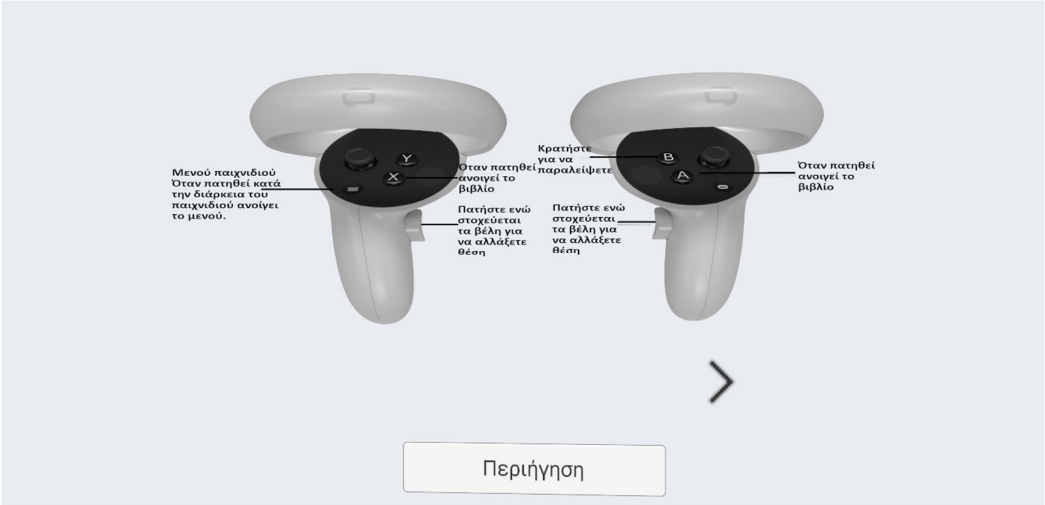
- Κύρια πύλη
- Χαραγμένη επιγραφή
- Το παλάτι των δεσποτών
- Υπόσκαφο κτίριο
- Βόρεια πύλη
- Ξενία
- Η εκκλησία των αγίων πάντων
- Η Ακρόπολη του κάστρου
- Η φυλακή
- Το οικόσημο

Gamified έκδοση

- Κύρια πύλη
- Το παλάτι των δεσποτών
- Βόρεια πύλη
- Η εκκλησία των αγίων πάντων
- Ξενία
- Η Ακρόπολη του κάστρου

5.1.1 Οδηγίες Χειρισμού.

Ανάλογα με την επιλογή σεναρίου ο χρήστης οδηγείται στο αντίστοιχο tutorial, μια σκηνή στην οποία προβάλλονται στον χρήστη οδηγίες για τον τρόπο λειτουργίας τον χειριστηρίων.



Εικόνα 35 Οδηγίες χρήσεως χειριστηρίων.

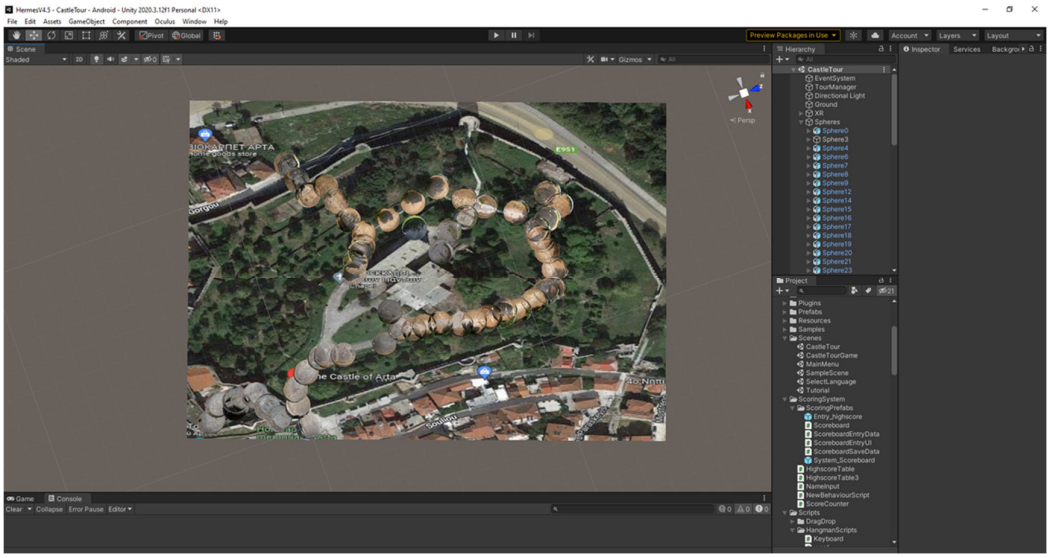
Στο tutorial του παιχνιδιού εμφανίζονται επιπρόσθετες οδηγίες για τα παιχνίδια του σεναρίου.

Εικόνα 36 Οδηγίες παιχνιδιών.

Όσον αφορά τον μηχανισμό εμφάνισης των οδηγιών (Παράρτημα 3) στην αρχή απενεργοποιεί όλες τις φωτογραφίες με τις οδηγίες πέραν της πρώτης. Στον κώδικα υπάρχει επίσης ένας έλεγχος για την τρέχουσα θέση των οδηγιών ώστε να εμφανίζει ή όχι το κουμπί της προηγούμενης και της επόμενης οδηγίας. Τέλος, με το πάτημα των κουμπιών γίνεται και η αντίστροφη περιστροφή του άξονα Z για την εμφάνιση της επόμενης οδηγίας και την εξαφάνιση της προηγούμενης.

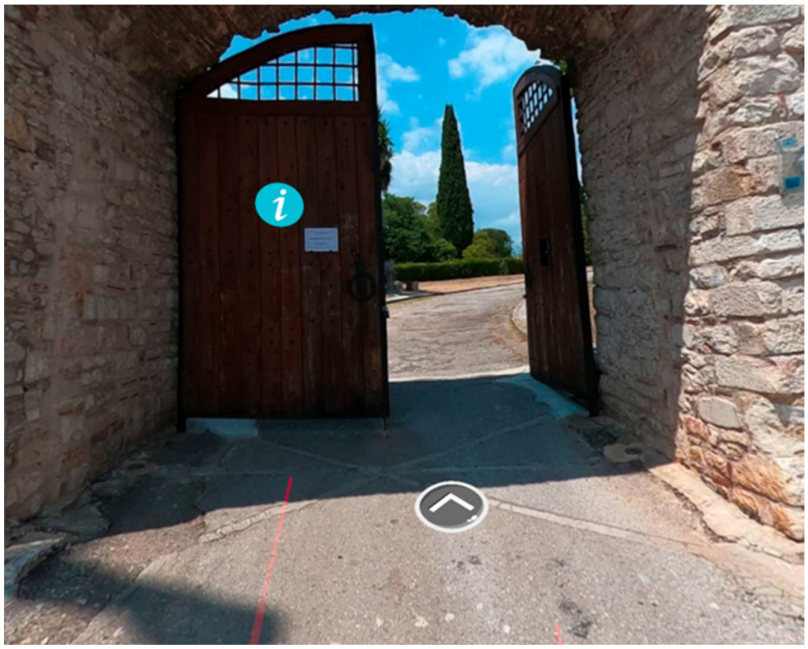
5.1.2 Έκδοση Περιήγησης.

Στο σενάριο της περιήγησης, το περιβάλλον αποτελείται από συνολικά πενήντα οκτώ σημεία του κάστρου. Τα σημεία αυτά πρόκειται για φωτογραφίες 360° που έχουν δεχθεί επεξεργασία και έχουν μετατραπεί σε σφαίρες στο περιβάλλον του Unity.



Εικόνα 37 Η τοποθέτηση των σφαιρών στο περιβάλλον unity.

Οι σφαίρες είναι σημεία στα οποία ο χρήστης μπορεί να περιηγηθεί με το πάτημα ενός κουμπιού στην επόμενη διαθέσιμη σφαίρα της επιλογής του.



Εικόνα 38 Η πρώτη σφαίρα της περιήγησης.

Η μετακίνηση μεταξύ των σφαιρών, τόσο στα σενάρια του κάστρου της Άρτας όσο και στο θεατράκι, γίνεται με τον κώδικα του παραρτήματος τέσσερα. Κατά του πάτημα του κουμπιού της κατεύθυνσης στην οποία θέλει ο χρήστης να μετακινηθεί η τρέχουσα σφαίρα απενεργοποιείται και ενεργοποιείται αυτή που επέλεξε ο χρήστης. Ταυτόχρονα με την μετακίνηση του χρήστη η εικόνα που βλέπει αλλάζει σε μαύρο που

σταδιακά γίνεται διάφανο με σκοπό την αποφυγή ανεπιθύμητων αισθημάτων όπως το αίσθημα της ναυτίας.

Σε ορισμένες σφαίρες δίνεται η επιλογή στον χρήστη να μάθει πληροφορίες για το σημείο ενδιαφέροντος όπως φαίνεται στην εικόνα 39. Τα κείμενα συνοδεύονται από ηχητική αφήγηση που ξεκινάει αυτόματα κατά το πάτημα του αντίστοιχου κουμπιού.

Εικόνα 39 Πληροφορίες για την κεντρική πύλη του κάστρου.

5.1.3 Gamified Έκδοση.

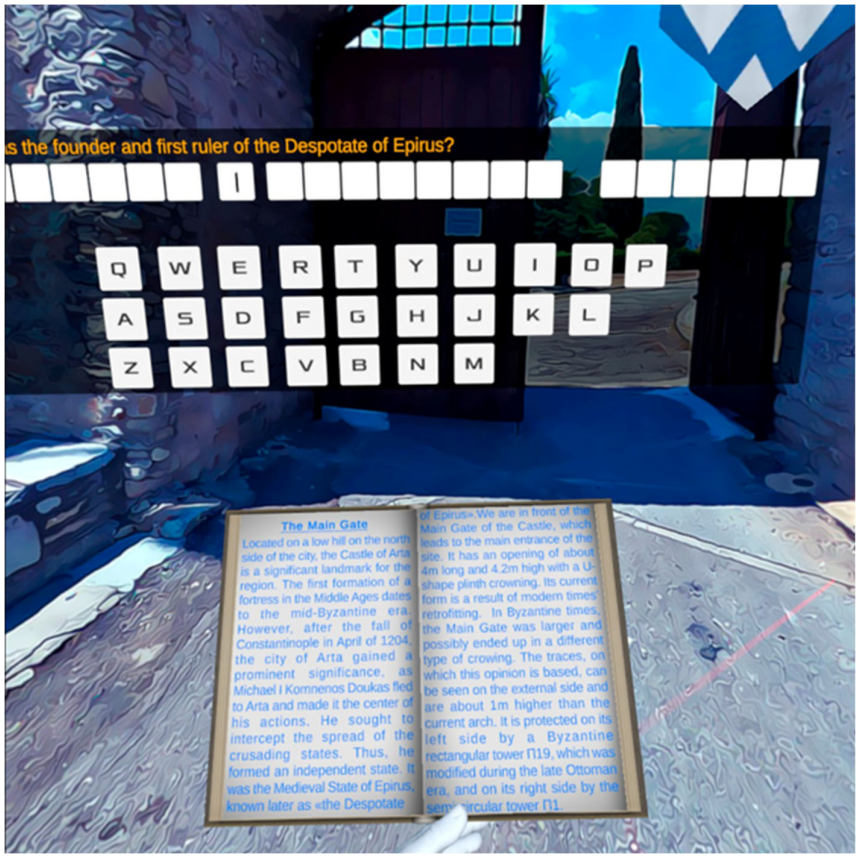
Στην gamified έκδοση οι σφαίρες είναι μόλις έξι και είναι επεξεργασμένες έτσι ώστε να μοιάζουν περισσότερο με παιχνίδι. Σε αυτό το σενάριο ο χρήστης δεν έχει τόσο μεγάλη ελευθερία στην περιήγηση. Για παράδειγμα για να μεταφερθεί στην επόμενη σφαίρα θα πρέπει πρώτα να ολοκληρώσει με επιτυχία το παιχνίδι της τρέχουσας σφαίρας.

Παρόμοια με το προηγούμενο σενάριο ο χρήστης ξεκινάει από την κεντρική πύλη στην οποία μαθαίνει πληροφορίες από τον αφηγητή για το σημείο ενδιαφέροντος. Ταυτόχρονα με την αφήγηση εμφανίζεται ένας διάλογος που ακολουθεί το οπτικό πεδίο του χρήστη.

Εικόνα 40 Πρώτη σφαίρα στο gamified σενάριο.

Με την ολοκλήρωση της αφήγησης υπάρχει μια ηχητική ειδοποίηση που ενημερώνει τον χρήστη πως έχει ξεκλειδωθεί μια νέα σελίδα στο βιβλίο του. Την ίδια στιγμή γίνεται ορατό και το παιχνίδι που πρέπει να ολοκληρώσει επιτυχώς. Το παιχνίδι είναι σχετικό με το κείμενο που έχει προηγηθεί και το βιβλίο που περιέχει το κείμενο βοηθά τον χρήστη στο να απαντήσει την ερώτηση.

Με σκοπό την επίτευξη μεγαλύτερου βαθμού εμβύθισης έχουν τοποθετηθεί στις σφαίρες διάφορα ηχητικά εφέ όπως ήχοι αλόγων και πουλιών. Η τοποθέτηση των σφαιρών σε ανάλογη με αυτή της πραγματική τους τοποθεσίας δημιουργεί την ψευδαίσθηση του βάθους καθώς οι ήχοι ακούγονται σταδιακά περισσότερο ή λιγότερο ανάλογα με την τοποθεσία του χρήστη



Εικόνα 41 Το παιχνίδι της πρώτης σφαίρας, κεντρική πύλη, και το βιβλίο του χρήστη.

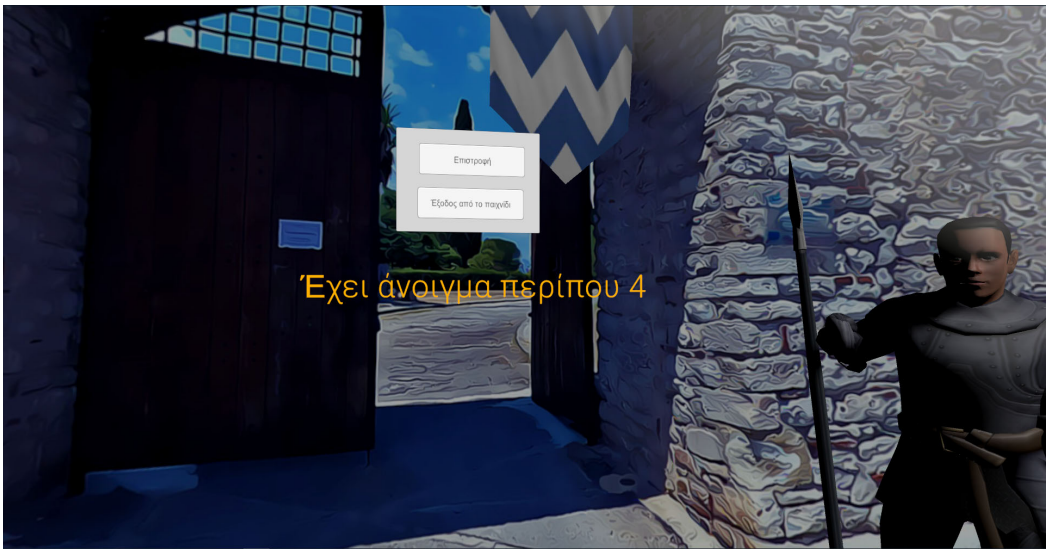
Το βιβλίο του χρήστη το οποίο περιέχει τα κείμενα, πρόκειται για ένα έτοιμο asset το οποίο με τις κατάλληλες παραμετροποιήσεις εμφανίζει τις επιθυμητές σελίδες στις επιθυμητές σφαίρες στην προεπιλεγμένη γλώσσα.

Ο μηχανισμός παράλειψης είναι ένα χαρακτηριστικό του gamified σεναρίου που ενεργοποιείται με το παρατεταμένο πάτημα του κουμπιού «JoystickButton1» για πεπερασμένο χρόνο. Εφόσον περάσει ο χρόνος, το κείμενο και η αφήγηση αυτού παραλείπεται, ενεργοποιείται το παιχνίδι της τρέχουσας σφαίρας και εμφανίζεται οι σελίδες του βιβλίου.

Εικόνα 42 Τρίτη σφαίρα, λειτουργία παράλειψης.

Στο ίδιο script με το μηχανισμό παράλειψης γίνεται και ο έλεγχος για το πάτημα του πλήκτρου μενού. Κατά το πάτημα αυτού του κουμπιού εμφανίζεται ένα μενού στον χρήστη με την επιλογή της επιστροφής στο κεντρικό και μενού και την συνέχιση του παιχνιδιού. Καθ' όλη την διάρκεια που το μενού είναι ενεργοποιημένο, οποιαδήποτε λειτουργία της εφαρμογής, όπως τα ηχητικά, βρίσκονται σε παύση.

Για να γίνεται πιο αισθητό το πάγωμα του χρόνου από το μενού, κατά την ενεργοποίηση του, ο χρήστης περιβάλλεται από μία ημιδιαφανή γκρίζα σφαίρα η οποία έχει αντιστραφεί για να περικλείει τον χρήστη με την χρήση κώδικα του Παράρτημα 6.



Εικόνα 43 Εμφάνιση μενού επιλογών.

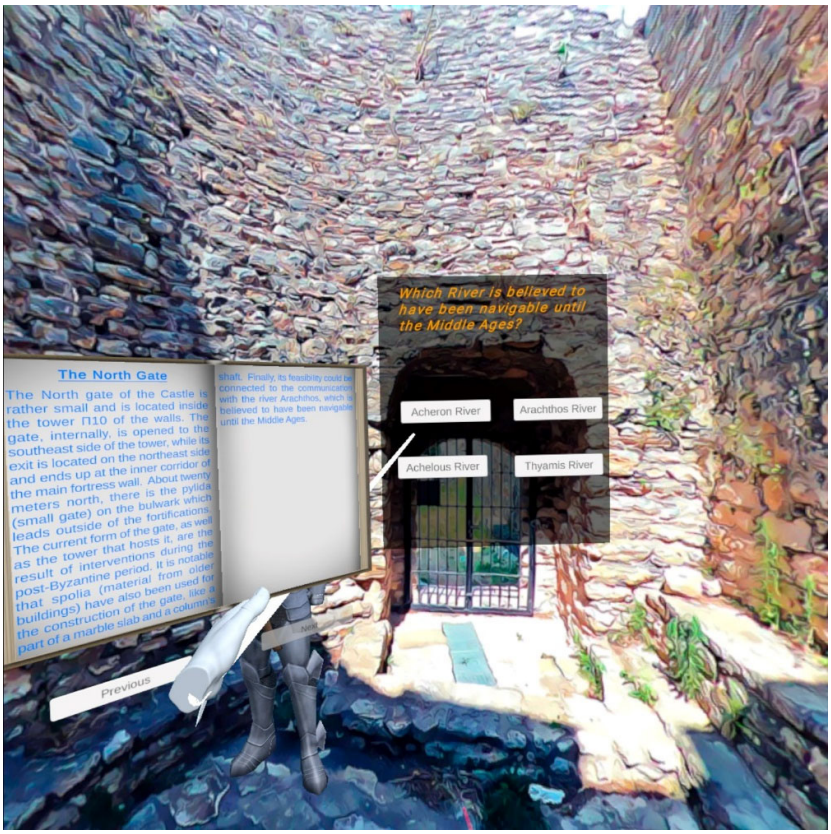
Στο παιχνίδι του παζλ (Παράρτημα 5) γίνεται η διάσπαση της εικόνας σε blocks, έπειτα επιλέγεται τυχαία ένα μπλοκ το οποίο μετατρέπεται σε κενό χώρο και στο οποίο μπορούν να μετακινηθούν τα υπόλοιπα μπλοκ εάν είναι δυνατή η κίνηση. Στον ίδιο κώδικα υπάρχει ο έλεγχος της ολοκλήρωσης του παζλ όπου το παιχνίδι σταματάει και η μετακίνηση στην επόμενη σφαίρα καθίσταται δυνατή εάν όλα τα block βρίσκονται στο σωστό σημείο.

Εικόνα 44 Εμφάνιση παιχνιδιού, Ξενία.

Στην σφαίρα που ακολουθεί, στο τέλος της αφήγησης δεν εμφανίζεται κάποιο παιχνίδι, δίνεται όμως η δυνατότητα στον χρήστη πατώντας τους ιππότες που υπάρχουν στην σφαίρα να μάθει περισσότερες πληροφορίες για τις οικογένειες τις οποίες αντιπροσωπεύουν.

Εικόνα 45 Παλάτι των δεσποτών.

Στην βόρεια πύλη με την ολοκλήρωση της αφήγησης εμφανίζεται στον χρήστη ένα παιχνίδι πολλαπλής επιλογής που υλοποιήθηκε με δυνατότητες που δίνει το ίδιο το Unity και συνεπώς δεν ήταν αναγκαία η συγγραφή κάποιου κώδικα.



Εικόνα 46 Βόρεια πύλη, παιχνίδι πολλαπλή επιλογής.

Όπως με το παιχνίδι της βόρεια πύλης, έτσι και στο εκκλησάκι των αγίων πάντων το παιχνίδι υλοποιήθηκε χρησιμοποιώντας panels και φωτογραφίες τις οποίες ο χρήστης πρέπει να μετακινήσει και να αποθέσει τις φωτογραφίες σε χρονολογική σειρά. Για να επιβεβαιώσει την ορθή τοποθέτηση των φωτογραφιών, ο χρήστης θα πρέπει να πατήσει το πράσινο tick που βρίσκεται αριστερά του παιχνιδιού. Κατά το πάτημα του κουμπιού καλείται ο κώδικας ο οποίος ελέγχει τις συνθήκες και στην περίπτωση που είναι σωστά τοποθετημένες οι φωτογραφίες το παιχνίδι εξαφανίζεται και γίνεται δυνατή η μεταφορά στην επόμενη σφαίρα.

Εικόνα 47 Παιχνίδι drag and drop, εκκλησάκι αγίων πάντων.

Τέλος, στην ακρόπολη του κάστρου υπάρχει ένα παιχνίδι αντιστοίχισης στο οποίο ο χρήστης πρέπει να αντιστοιχίσει τις φωτογραφίες με το όνομα του σημείου. Προγραμματιστικά,(Παράρτημα 7 και Παράρτημα 8) κάθε φορά που συνδέεται κάποιο σημείο γίνεται έλεγχος εάν το όνομα της φωτογραφίας είναι ίδιο με αυτό του ονόματος του σημείου. Όταν όλα τα σημεία αντιστοιχηθούν σωστά τότε το παιχνίδι τελειώνει.

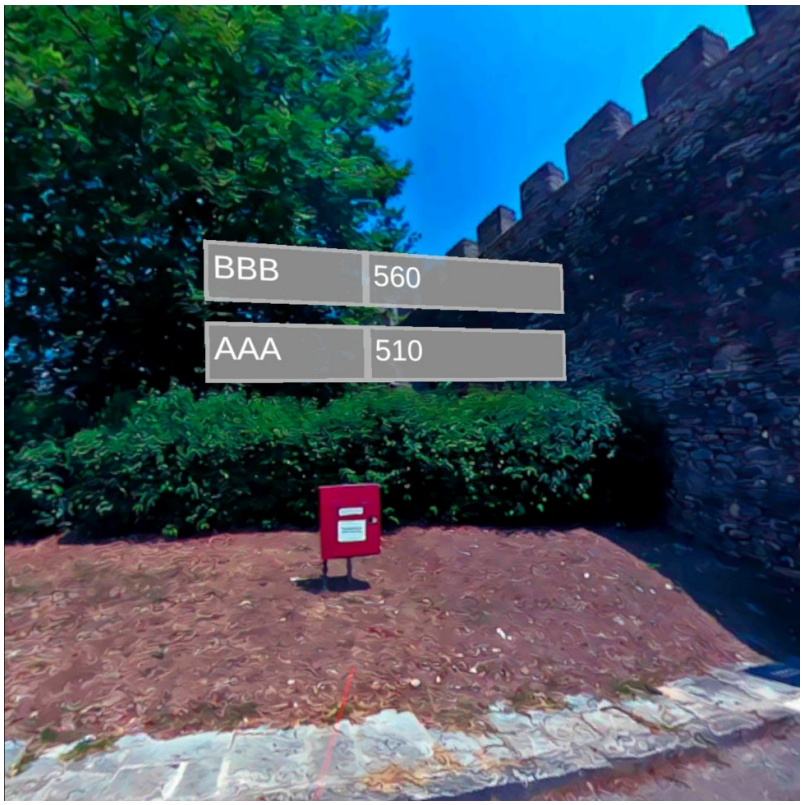
Εικόνα 48 Παιχνίδι Αντιστοίχισης.

Στο gamified σενάριο υπάρχει ένα σύστημα πόντων, με κάθε παιχνίδι να βαθμολογείται λαμβάνοντας υπόψη είτε τον χρόνο που χρειάστηκε για την ολοκλήρωση του π.χ. στο παζλ είτε από τα λάθη που έκανε ο χρήστης π.χ. στην κρεμάλα. Με την ολοκλήρωση και του τελευταίου παιχνιδιού εμφανίζεται σχετικό μήνυμα στο χρήστη, ένα πληκτρολόγιο για την εισαγωγή του ονόματος του και ένας πίνακας με τις πέντε (5) υψηλότερες βαθμολογίες, εάν υπάρχουν.

Για το πληκτρολόγιο εισαγωγής επαναχρησιμοποιήθηκε το ίδιο asset που χρησιμοποιήθηκε και στην κρεμάλα με αρκετές παραμετροποιήσεις όσον αφορά την λογική του. Σε αυτό προστέθηκαν λειτουργίες όπως η διαγραφή (Del) του τελευταίου γράμματος και το κουμπί εισαγωγής (Enter) για την προσθήκη και του ονόματος της βαθμολογίας του χρήστη στον πίνακα με τις υψηλότερα βαθμολογίες. Ο κώδικας του πληκτρολογίου αυτού υπάρχει στο Παράρτημα 9 και Παράρτημα 10.

Εικόνα 49 Πληκτρολόγιο Εισαγωγής Ονόματος.

Οι βαθμολογίες χρηστών αποθηκεύονται σε αρχείο Json και χρησιμοποιείται τόσο για την εγγραφή όσο και για την ανάγνωση των τιμών. Σε περίπτωση που το αρχείο δεν προϋπάρχει τότε ο κώδικας (Παράρτημα 11) δημιουργεί ένα τέτοιο αρχείο.

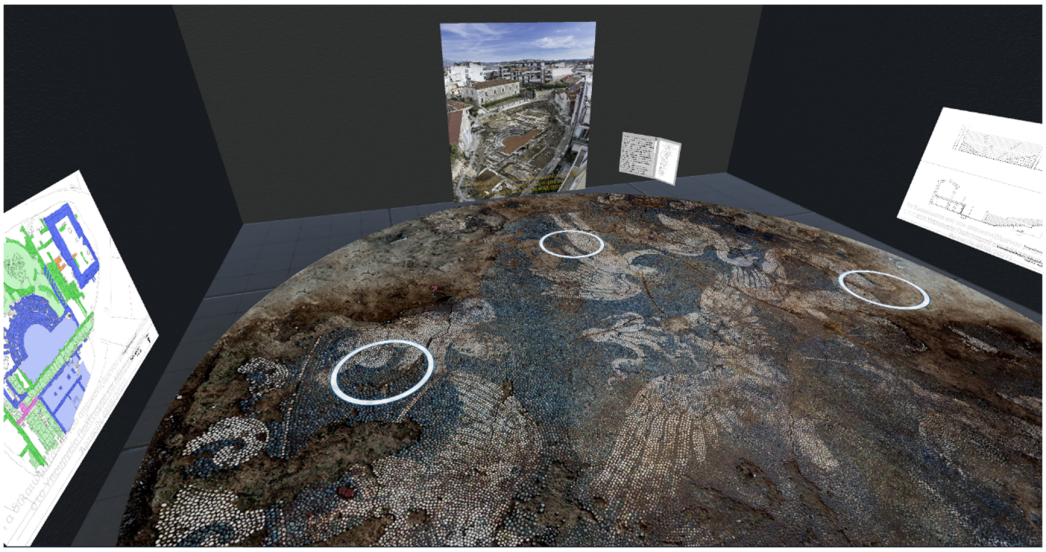


Εικόνα 50 Πίνακα Υψηλών Βαθμολογιών.

5.2 Θεατράκι της Άρτας.

Το θεατράκι της Άρτας αποτελείται από μια μόνο έκδοση που εμφανίζεται στην γλώσσα που επέλεξε ο χρήστης. Το σενάριο αυτό λαμβάνει χώρα σε έναν χώρο που θυμίζει μουσείο. Στον χώρο αυτό ο χρήστης μπορεί να μετακινηθεί ανάμεσα σε τρία σημεία. Στον αρχικό χώρο, ο χρήστης έχει δεξιά του ένα πάνελ με πληροφορίες και φωτογραφίες ενώ μπροστά του μια εικόνα με έξι σημεία με τα οποία μπορεί να αλληλεπιδράσει.

Η μεταφορά μεταξύ των σημείων χρησιμοποιεί τον ίδιο κώδικα με αυτόν του κάστρου (Παράρτημα 4).



Εικόνα 51 Αρχική θέση στο θεατράκι.

Τα έξι σημεία με τα οποία ο χρήστης μπορεί να αλληλεπιδράσει είναι:

- Το δημόσιο κτίριο.
- Λουτρά-Ψηφιδωτά.
- Οδός.
- Προσκήνιο.
- Πηγάδι.
- Ορχήστρα

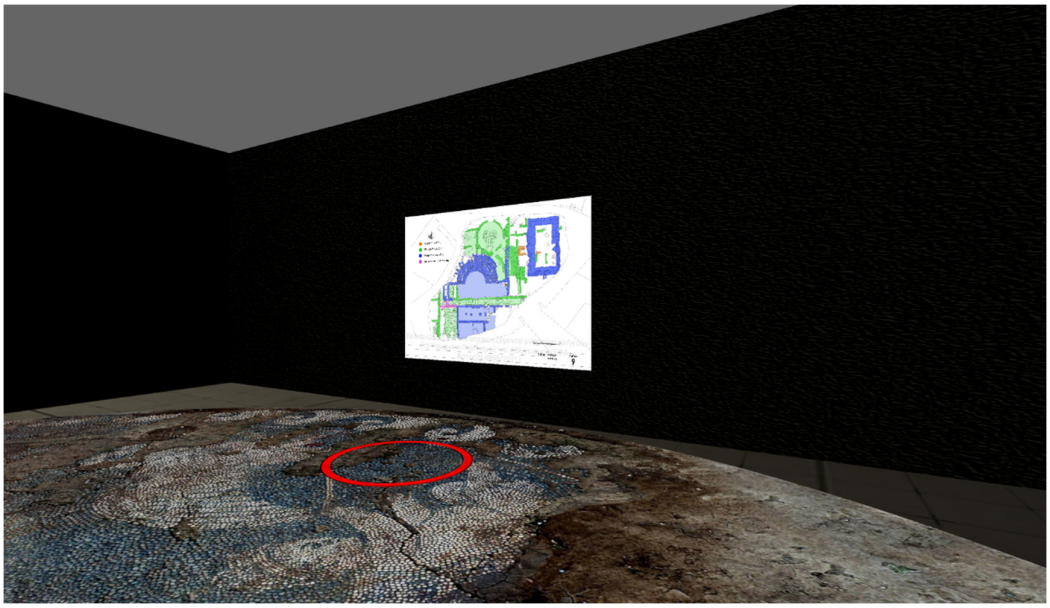
Ανάλογα με το σημείο ενδιαφέροντος που επιλέγει ο χρήστης, το χρώμα του πάνω στην φωτογραφία γίνεται πιο φωτεινό και στο πάνελ που βρίσκεται δεξιά του χρήστη εμφανίζονται οι ανάλογες πληροφορίες και εικόνες του σημείου.

Η εναλλαγή μεταξύ των φωτογραφιών επιτυγχάνεται με τον κώδικα του Παράρτημα 12. Στον κώδικα αυτό υπάρχει ένας μετρητής ο οποίος μεταβάλλεται και εμφανίζει την σωστή φωτογραφία ανάλογα με τις ενέργειες του χρήστη.

Με την επιλογή σημείου εμφανίζεται επίσης ένα κουμπί επιστροφής το οποίο στο πάτημα του επαναφέρει την εικόνα και το κείμενο στην αρχική του κατάσταση, δηλαδή στις γενικές πληροφορίες του μικρού θεάτρου.

Εικόνα 52 Επιλογή σημείου ψηφιδωτών.

Στον χώρο του μουσείου, ο χρήστης έχει την δυνατότητα να μετακινηθεί και σε δύο ακόμα σημεία στα οποία ο χρήστης τηλεμεταφέρεται με το πάτημα του κύκλου.



Εικόνα 53 Επιλογή μεταφοράς του χρήστη.

ΕΠΙΛΟΓΟΣ

Εν κατακλείδι, η σημερινή τεχνολογία εικονικής πραγματικότητας βασίζεται σε ιδέες που χρονολογούνται από το 1838, από το πρώτο στερεοσκόπιο στο VIEW- master και από το Sensorama του Heilig στο Ultimate Display του Sutherland όπου και τοποθετήθηκε ο ακρογωνιαίος λίθος για την εξέλιξη της εικονικής πραγματικότητας.

Η έρευνα και ο πειραματισμός από τον στρατό και την NASA με σκοπό την επίλυση προβλημάτων, όπως των δαπανηρών και επικίνδυνων εκπαιδεύσεων, ακόμα και οι ριψοκίνδυνες επενδύσεις όπως αυτή του Virtuality με σκοπό το κέρδος βοήθησαν στην ραγδαία ανάπτυξη της εικονικής πραγματικότητας.

Στην σημερινή εποχή, αν και βρίσκεται ακόμα σε πρώιμα στάδια, η εικονική πραγματικότητα πρόκειται για ένα εργαλείο που αξιοποιείται τόσο στην διασκέδαση και την ψυχαγωγία όσο και στους τομείς της εκπαίδευσης, του τουρισμού και της ιατρικής.

Ο σύγχρονος εξοπλισμός εικονικής πραγματικότητας είναι πλέον χαμηλού κόστους και υψηλής ποιότητας οι υπηρεσίες του είναι εύκολα προσβάσιμες χάρις στους πρωτοπόρους εφευρέτες των τελευταίων έξι δεκαετιών που μόχθησαν και μέσα από τις αποτυχίες, απογοητεύσεις αλλά με επιμονή και υπομονή έδειξαν τον δρόμο της επιτυχίας. Ο δρόμος έφτασε στο τέρμα του; *Η πρόβλεψη που μπορώ να κάνω με τη μεγαλύτερη βεβαιότητα είναι ότι οι πιο εντυπωσιακές ανακαλύψεις θα είναι αυτές τις οποίες σήμερα δεν είμαστε αρκετά σοφοί για να τις προβλέψουμε (Καρλ Σαγκάν, 1934-1996, Αμερικανός αστρονόμος).*

ΒΙΒΛΙΟΓΡΑΦΙΑ

(Spanner, S., 2015. *Classic 3D Photo Viewer The View-Master Brought Back To Life Using VR*. [image] Available at: <https://www.pocketgamer.com/view-master/classic-3d-photo-viewer-the-view-master-brought-back-to-life-using-vr/> [Accessed 30 August 2022].

1977. *Sayre Glove*. [image] Available at: <https://www.evl.uic.edu/research/2162> [Accessed 30 August 2022].

1986. *The Super Cockpit and its Human Factors Challenges*. [image] Available at: <https://journals.sagepub.com/doi/10.1177/154193128603000112> [Accessed 30 August 2022].

2016. *Charles Wheatstone: The Father Of 3D And Virtual Reality Technology*. [image] Available at: <https://www.kcl.ac.uk/charles-wheatstone-the-father-of-3d-and-virtual-reality-technology-2> [Accessed 30 August 2022].

2018. *Headsight Demonstration*. [image] Available at: <https://arvrjourney.com/a-short-history-of-virtual-reality-for-busy-people-9b088722f3a9> [Accessed 30 August 2022].

2018. *Sword Of Damocles*. [image] Available at: <https://arvrjourney.com/a-short-history-of-virtual-reality-for-busy-people-9b088722f3a9> [Accessed 30 August 2022].

2018. *Virtual reality therapy helps people overcome fear of heights*. [image] Available at: <https://www.indiatvnews.com/lifestyle/health-study-virtual-reality-therapy-helps-people-overcome-fear-of-heights-452379> [Accessed 30 August 2022].

2022. *Sega VR*. [image] Available at: <https://segaretro.org/Sega_VR> [Accessed 30 August 2022].

2022. *Unreal Engine 5*. [image] Available at: <https://www.unrealengine.com/en-US/unreal-engine-5> [Accessed 30 August 2022].

2022. *Virtuality*. [image] Available at: <https://www.vrs.org.uk/dr-jonathan-walden-virtuality-new-reality-promise-two-decades-soon/> [Accessed 30 August 2022].

Abdelmaged, M., 2021. Implementation of Virtual Reality in Healthcare, Entertainment, Tourism, Education, and Retail Sectors. [online] (110491), p.9. Available at: <https://mpra.ub.uni-muenchen.de/110491/> [Accessed 22 August 2022].

Amos, E., 2015. *A Google Cardboard VR Headset*. [image] Available at: <https://en.wikipedia.org/wiki/Google_Cardboard> [Accessed 30 August 2022].

Andrzejczak, J., Kozłowicz, W., Szrajber, R. and Wojciechowski, A., 2021. Factors affecting the sense of scale in immersive, realistic virtual reality space. Computational Science – ICCS 2021, pp.3–16.

Babich, N., 2019. *How VR Education Will Change How We Learn & Teach | Adobe XD Ideas*. [online] Ideas. Available at: <https://xd.adobe.com/ideas/principles/emerging-technology/virtual-reality-will-change-learn-teach/> [Accessed 22 August 2022].

BUCKLEY, I., 2021. *The 5 Unity Game Development Languages: Which Should You Learn?*. [online] MUO. Available at: <https://www.makeuseof.com/tag/unity-game-development-languages/> [Accessed 22 August 2022].

CONSTANTINE, F., n.d. NASA'S VIEW VR. [image] Available at: <http://briteliteimmersive.com/blog/remembering-nasas-view-vr> [Accessed 30 August 2022].

Corning, A., 2018. *Blast From The Past: Virtual Reality | Radiant Vision Systems*. [online] Radiant Vision Systems. Available at: <https://www.radiantvisionsystems.com/blog/blast-past-virtual-reality> [Accessed 31 May 2022].

Costello, P., 1997. Health and safety issues associated with virtual reality. Loughborough: Advisory Group on Computer Graphics.

DeFanti, T. and Sandin, D., 1977. *Final Project Report*. [online] Chicago. Available at: <https://www.evl.uic.edu/pubs/2296> [Accessed 6 June 2022].

Encyclopedia Britannica. 2018. *Link Trainer | Flight Simulator*. [online] Available at: <https://www.britannica.com/technology/Link-Trainer> [Accessed 31 May 2022].

Fattouh, L., 2017. *Sensorama Simulator Device*. [image] Available at: <https://www.researchgate.net/figure/Sensorama-simulator-device-15_fig5_321183923> [Accessed 30 August 2022].

Gandhi, R. and Patel, D., 2018. Virtual Reality – Opportunities and Challenges. International Research Journal of Engineering and Technology (IRJET), 5(1).

Gegung, E., 2021. International Tourism and The COVID-19 Pandemic: The Use of Virtual Reality to Increase Tourism Destination Sustainability and How Users Perceive The Authenticity of VR Experiences. [online] 15(1), p.7. Available at: <http://ejournal.kemenparekraf.go.id/index.php/jki/article/view/194> [Accessed 22 August 2022].

glassdevelopment. 2014. *HMD – History And Objectives Of Inventions*. [online] Available at: <https://glassdevelopment.wordpress.com/2014/04/17/hmd-history-and-objectives-of-inventions/> [Accessed 3 June 2022].

HASSASSIAN, A., 2019. *Use of VR in tourism*. [image] Available at: <https://360stories.com/blog/a-boom-of-virtual-reality-in-the-travel-industry> [Accessed 31 August 2022].

Healthline. 2021. *Exposure Therapy: Types, How It'S Done, And More*. [online] Available at: <https://www.healthline.com/health/exposure-therapy> [Accessed 22 August 2022].

History Computer. 2022. *Oculus: Complete Guide — History, Products, Founding, And More*. [online] Available at: <https://history-computer.com/oculus-history/> [Accessed 6 June 2022].

Ismail, A., 2022. *A Drawing from Telesphere Mask Patent*. [image] Available at: <https://www.dsource.in/course/virtual-reality-introduction/evolution-vr/telesphere-mask> [Accessed 30 August 2022].

Ismail, A., 2022. *Telesphere Mask Demonstration*. [image] Available at: <https://www.dsource.in/course/virtual-reality-introduction/evolution-vr/telesphere-mask> [Accessed 30 August 2022].

Kirk, B., 2021. *NASA VIEW Headset Defined VR Decades Before Oculus*. [online] autoevolution. Available at: <https://www.autoevolution.com/news/nasa-view-headset-defined-vr-decades-before-oculus-probably-can-t-play-half-life-alyx-177844.html> [Accessed 6 June 2022].

Little, B., 2021. *6 World War II Innovations That Changed Everyday Life*. [online] HISTORY. Available at: <https://www.history.com/news/world-war-ii-innovations> [Accessed 31 May 2022].

Lowood, H., Das, D., Gaur, A., Hosch, W., Lotha, G., Rodriguez, E. and Tikkanen, A., 2021. Education and training. In: *Britannica*. Britannica.

Lowood, H., Das, D., Gaur, A., Hosch, W., Lotha, G., Rodriguez, E. and Tikkanen, A., 2021. Entertainment. In: *Britannica*. Britannica.

MARTIN, J., 2020. *What Is A Game Engine? | University Of Silicon Valley*. [online] Usv.edu. Available at: <https://usv.edu/blog/what-is-a-game-engine/> [Accessed 22 August 2022].

Maxwell, W., 2021. *Should You Make Your Own Game Engine - What You Need To Consider - CG Obsession*. [online] CG Obsession. Available at: <https://cgobsession.com/should-you-make-your-own-game-engine/> [Accessed 22 August 2022].

Miller, A., 2021. VR resolution, field of view and the science of the human eye. [online] AR Insider. Available at: <https://arinsider.co/2021/12/09/vr-resolution-field-of-view-and-the-science-of-the-human-eye/> [Accessed 6 Feb. 2023].

n.d. *Cryengine*. [image] Available at: <https://www.crytek.com/cryengine> [Accessed 30 August 2022].

n.d. *Glowflow*. [image] Available at: <https://aboutmyronkrueger.weebly.com/glowflow.html> [Accessed 30 August 2022].

n.d. *Link Trainer exterior THE DEVELOPMENT OF INSTRUMENT FLIGHT TRAINERS*. [image] Available at: <http://www.canadianflight.org/content/link-trainer> [Accessed 30 August 2022].

n.d. *Link Trainer interior THE DEVELOPMENT OF INSTRUMENT FLIGHT TRAINERS*. [image] Available at: <http://www.canadianflight.org/content/link-trainer> [Accessed 30 August 2022].

n.d. *Metaplay*. [image] Available at: <https://aboutmyronkrueger.weebly.com/metaplay.html> [Accessed 30 August 2022].

n.d. *Oculus*. [image] Available at: <https://www.kickstarter.com/projects/1523379957/oculus-rift-step-into-the-game?lang=it> [Accessed 30 August 2022].

n.d. *Psychic Space*. [image] Available at: <https://aboutmyronkrueger.weebly.com/psychic-space.html> [Accessed 30 August 2022].

n.d. *Surgical simulation training*. [image] Available at: <https://jasoren.com/using-virtual-reality-for-surgical-simulation-trainings/> [Accessed 30 August 2022].

n.d. *Unity*. [image] Available at: <https://unity.com/case-study/concrete-software#keeping-players-and-developers-happy> [Accessed 30 August 2022].

n.d. *VITAL Helmet*. [image] Available at: <https://www.virtual-reality-shop.co.uk/the-complete-history-of-vr-part-4/> [Accessed 30 August 2022].

Naval Air Station Fort Lauderdale Museum. n.d. *Link Trainer Flight Simulator*. [online] Available at: <https://www.nasflmuseum.com/link-trainer.html> [Accessed 31 May 2022].

Ninad, J., 2019. Impact of Virtual Reality on Gaming. *International Research Journal of Engineering and Technology*, [online] p.4. Available at:

<https://www.irjet.net/archives/V6/i12/IRJET-V6I12352.pdf> [Accessed 22 August 2022].

Norman, J., n.d. *Daniel J. Sandlin Invents The Sayre Glove*. [online] HistoryofInformation.com. Available at: <https://www.historyofinformation.com/detail.php?id=3625> [Accessed 6 June 2022].

Page, R., 2004. *Brief History Of Flight Simulation*. [ebook] LUCAS HEIGHTS NSW 2234. Available at: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.132.5428&rep=rep1&type=pdf> [Accessed 3 June 2022].

Poetker, B., 2019. What Is Virtual Reality? (+3 Types of VR Experiences). [image] Available at: <https://learn.g2.com/virtual-reality> [Accessed 30 August 2022].)

Polycarpou, C., 2018. *Senses in VR gaming*. [image] Available at: <https://www.vi-mm.eu/2018/03/08/virtual-reality-in-gaming/> [Accessed 30 August 2022].

Sega Retro. 2022. *Sega VR - Sega Retro*. [online] Available at: <https://segaretro.org/Sega_VR> [Accessed 6 June 2022].

Sinicki, A., 2021. *What Is Unity? Everything You Need To Know*. [online] Android Authority. Available at: <https://www.androidauthority.com/what-is-unity-1131558/> [Accessed 22 August 2022].

Sorene, P., 2014. *Jaron Lanier's Eyephone: Head And Glove Virtual Reality In The 1980S - Flashbak*. [online] Flashbak. Available at: <https://flashbak.com/jaron-laniers-eyephone-head-and-glove-virtual-reality-in-the-1980s-26180/> [Accessed 6 June 2022].

teslasuit. 2022. *The History Of Virtual Reality: Ultimate Guide. Part 1*. [online] Available at: <https://teslasuit.io/history-of-virtual-reality-ultimate-guide/> [Accessed 6 June 2022].

The Ghost Howls. 2017. *The Sense Of Taste In Virtual Reality*. [online] Available at: <https://skarredghost.com/2017/07/07/sense-taste-virtual-reality/> [Accessed 22 August 2022].

The VR Shop. n.d. *The Complete History Of VR – Part 4: Military Purposes*. [online] Available at: <https://www.virtual-reality-shop.co.uk/the-complete-history-of-vr-part-4/> [Accessed 3 June 2022].

The VR Shop. n.d. *The Sword Of Damocles (1968) - The Complete History Of VR*. [online] Available at: <https://www.virtual-reality-shop.co.uk/the-sword-of-damocles-1968/> [Accessed 3 June 2022].

Virtual Reality Society. n.d. *History Of Virtual Reality - Virtual Reality Society*. [online] Available at: <https://www.vrs.org.uk/virtual-reality/history.html> [Accessed 31 May 2022].

Virtual Reality Society. n.d. Virtuality – A New Reality Of Promise, Two Decades Too Soon - Virtual Reality Society. [online] Available at: <https://www.vrs.org.uk/dr-jonathan-walden-virtuality-new-reality-promise-two-decades-soon/> [Accessed 6 June 2022].

Virtual Reality Society. n.d. *VPL Research Jaron Lanier - Virtual Reality Society*. [online] Available at: <https://www.vrs.org.uk/virtual-reality-profiles/vpl-research.html> [Accessed 6 June 2022].

Voicesofvr.com. 2015. *#245: 50 Years Of VR With Tom Furness: The Super Cockpit, Virtual Retinal Display, HIT Lab, & Virtual World Society | Voices Of VR Podcast*. [online] Available at: <https://voicesofvr.com/245-50-years-of-vr-with-tom-furness-the-super-cockpit-virtual-retinal-display-hit-lab-virtual-world-society/> [Accessed 6 June 2022].

Wiltz, C., 2019. *The Story Of Sega VR: Sega's Failed Virtual Reality Headset*. [online] designnews.com. Available at: <https://www.designnews.com/electronics-test/story-sega-vr-segas-failed-virtual-reality-headset> [Accessed 6 June 2022].

YILDIRIM, G., ELBAN, M. and YILDIRIM, S., 2018. Analysis of Use of Virtual Reality Technologies in History Education: A Case Study. *Asian Journal of Education and Training*, 4(2), pp.62-69.

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΠΑΡΑΡΤΗΜΑΤΑ

Παράρτημα 1

ChangeScene

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.EventSystems;
using UnityEngine.XR;
using TMPro;
using UnityEngine.UI;


public class ChangeScene : MonoBehaviour
{
    public GameObject[] mainMenu;
    public static string Language;//Greek or English
    public static bool Paused=false;
    [SerializeField] int index;

    public float downTime, upTime, pressTime = 0;
    public float countDown = 4.0f;
    public bool ready = false;


    public GameObject[] Button;
    public static int counter = 0;

    public static bool Skipped=false;
    public BarProgress other;

    public GameObject pageUnlockedText;
    void Start()
    {
        Skipped = false;
        if (Language == "Greek")
        {
            index = 1;
        }
        else
        {
            index = 0;
        }

    }
    void Update()
    {
        InputCheck();

        if (Input.GetKeyDown(KeyCode.JoystickButton1) && ready == false)
        {
            downTime = Time.time;
            pressTime = downTime + countDown;
            ready = true;
        }

        if (Input.GetKeyUp(KeyCode.JoystickButton1))
```

```
        {
            ready = false;
        }

        if (Time.time >= pressTime && ready == true)
        {
            ready = false;
            Debug.Log("done");

            if (counter!=1 && Skipped==false)
            {
                Button[counter].GetComponent<Button>().onClick.Invoke();
                StartCoroutine(UnlockedTimer());
                other.currentAmount = 0;
                other.LoadingBar.GetComponent<Image>().fillAmount =
other.currentAmount / 100;
                other.progressBar.SetActive(false);

                Skipped = true;
            }

        }

    }

    IEnumerator UnlockedTimer()
    {
        float count = 0;
        pageUnlockedText.SetActive(true);
        while (count < 4)
        {
            count += Time.deltaTime;
            yield return null;
        }
        pageUnlockedText.SetActive(false);
    }

    public void Pause()
    {

        Time.timeScale = 0f;
        Paused = true;

    }

    public void ExitApp()
    {
        Application.Quit();
    }

    public void Resume()
    {

        Time.timeScale = 1f;
        mainMenu[index].SetActive(false);
        Paused = false;

    }

    public void LoadScene(string sceneName)
    {
        SceneManager.LoadScene(sceneName);
    }
```

```
    public void LanguageChosen()
    {
        Language = EventSystem.current.currentSelectedGameObject.name;
        Debug.Log(Language);
    }

    public void MainMenuOption()
    {
        mainMenu[index].SetActive(true);
    }

    public void InputCheck()
    {
        List<InputDevice> leftHandedControllers = new List<InputDevice>();
        InputDeviceCharacteristics desiredCharacteristics =
InputDeviceCharacteristics.HeldInHand | InputDeviceCharacteristics.Left |
InputDeviceCharacteristics.Controller;
        InputDevices.GetDevicesWithCharacteristics(desiredCharacteristics,
leftHandedControllers);

        foreach (var device in leftHandedControllers)
        {
            Debug.Log(string.Format("Device name '{0}' has characteristics
'{1}'", device.name, device.characteristics.ToString()));
            //Debug.Log("1");
            List<InputFeatureUsage> inputFeatures = new
List<InputFeatureUsage>();
            if (device.TryGetFeatureUsages(inputFeatures))
            {
                //Debug.Log("2");
                foreach (InputFeatureUsage feature in inputFeatures)
                {
                    if (feature.type == typeof(bool))
                    {
                        //Debug.Log("3");
                        bool featureValue;
                        if (device.TryGetFeatureValue(feature.As<bool>(), out
featureValue))
                        {
                            //Debug.Log("4");
                            Debug.Log(featureValue);

                            if ((feature.name == "Menu" || feature.name ==
"Start" || feature.name == "Select" || feature.name == "MenuButton" ||
feature.name == "Home" || feature.name == "start" || feature.name == "select"
|| feature.name == "menuButton" || feature.name == "MenuButton" || feature.name
== "home" || feature.name == "startButton" || feature.name == "StartButton") &&
featureValue)
                            {
                                Debug.Log(string.Format("Bool feature {0}'s
value is {1}", feature.name, featureValue.ToString()));
                                Debug.Log("Catching");

                                MainMenuOption();
                                Pause();
                            }


                        }
                    }
                }
            }
```

```
        }
    }
}
```

Παράρτημα 2

LanguageTextByLanguage

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using TMPro;
public class LanguageTextByLanguage : MonoBehaviour
{
    [SerializeField] Button button1;
    [SerializeField] Button button2;
    [SerializeField] Button button3;
    [SerializeField] Button button4;
    [SerializeField] GameObject KicLabGreek;
    [SerializeField] GameObject KicLabEnglish;

    void Start()
    {


        if (ChangeScene.Language=="Greek")
        {
            button1.GetComponentInChildren<TextMeshProUGUI>().text = " Έναρξη
Παιχνιδιού ";
            button2.GetComponentInChildren<TextMeshProUGUI>().text = " Έναρξη
Περιήγησης Κάστρου ";
            button3.GetComponentInChildren<TextMeshProUGUI>().text = " Έξοδος
";
            button4.GetComponentInChildren<TextMeshProUGUI>().text = " Έναρξη
Περιήγησης Θεάτρου ";
            KicLabGreek.SetActive(true);
        }
        else if (ChangeScene.Language == "English")
        {
            button1.GetComponentInChildren<TextMeshProUGUI>().text = " Start
Game ";
            button2.GetComponentInChildren<TextMeshProUGUI>().text = " Start
Castle Tour ";
            button3.GetComponentInChildren<TextMeshProUGUI>().text = " Exit ";
            button4.GetComponentInChildren<TextMeshProUGUI>().text = " Start
Theater Tour ";

            KicLabEnglish.SetActive(true);
        }

    }
}
```

Παράρτημα 3

RotatePanel

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

public class RotatePanel : MonoBehaviour
{
    public Vector3 rotationAngle = new Vector3(0, 0, 90);
    [SerializeField] float rotationDuration = 1f;


    [SerializeField] Image[] images;
    [SerializeField] int index = 0;
    [SerializeField] RectTransform pivotPoint;
    [SerializeField] Button rightButton;
    [SerializeField] Button leftButton;
    public GameObject GoToCastle;

    void Start()
    {
        foreach (Image image in images)
        {
            image.gameObject.SetActive(false);
        }
        leftButton.gameObject.SetActive(false);
        images[0].gameObject.SetActive(true);

    }

    void Update()
    {
        LastIndex();

        if (index != 0)
        {
            leftButton.gameObject.SetActive(true);
        }
        if (index != images.Length - 1 )
        {
            rightButton.gameObject.SetActive(true);
        }
    }

    public void RotateRight()
    {

        if (index < images.Length - 1)
        {
            index++;
            images[index].gameObject.SetActive(true);
            if (images.Last() == images[index])
            {
                rightButton.gameObject.SetActive(false);
            }
        }

        int lastIndex = index - 1;
        StartCoroutine(AnimateRotation(pivotPoint.rotation,
Quaternion.Euler(rotationAngle) * pivotPoint.rotation, lastIndex));

    }

    public void LastIndex()
```

```
    {
        if(index==images.Length-1)
        {
            if(GoToCastle!=null)
            {
                GoToCastle.SetActive(true);
            }
        }
    }


    public void RotateLeft()
    {
        if (index > 0)
        {
            index--;
            images[index].gameObject.SetActive(true);
            if (images.First() == images[index])
            {
                leftButton.gameObject.SetActive(false);
            }
        }
        Quaternion toRotation = Quaternion.Euler(-rotationAngle) *
pivotPoint.rotation;
        int lastIndex = index + 1;
        StartCoroutine(AnimateRotation(pivotPoint.rotation, toRotation,
lastIndex));
    }

    private IEnumerator AnimateRotation(Quaternion fromRotation, Quaternion
toRotation, int previousImageIndex)
    {
        fromRotation.z = Mathf.RoundToInt(fromRotation.z);
        float time = 0f;
        rightButton.interactable = false;
        leftButton.interactable = false;
        while (time < rotationDuration)
        {
            pivotPoint.rotation = Quaternion.Lerp(fromRotation, toRotation,
time / rotationDuration);
            time += Time.deltaTime;
            yield return null;
        }
        pivotPoint.rotation = toRotation;
        images[previousImageIndex].gameObject.SetActive(false);
        rightButton.interactable = true;
        leftButton.interactable = true;
    }
}
```

Παράρτημα 4

ChangeSphere

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

```
public class ChangeSphere : MonoBehaviour
{
    public TourManager tourManager;
    public int activeSphere;
    public int sphereToLoad;
    public GameObject XR;
    public float x_position;
    public float z_position;
    public Image image;

    //start, όπου image = Find gameobject get children
   // public ActivatingPressToSkip other;
    public void Load()
    {
        if (SceneManager.GetActiveScene() ==
SceneManager.GetSceneByName("CastleTourGame"))
        {
            StartCoroutine(FadingIn());
            tourManager.LoadSphere(activeSphere, sphereToLoad, image);
            ChangeScene.counter++;
            ChangeScene.Skipped = false;
        }
        else
        {
            StartCoroutine(FadingIn());
            tourManager.LoadSphere(activeSphere, sphereToLoad, image);
            //Debug.Log("it runs");

        }

    }

    IEnumerator FadingIn()
    {
        image.color = new Color32(0, 0, 0, 255);
        if (SceneManager.GetActiveScene() ==
SceneManager.GetSceneByName("CastleTourGame"))
            XR.transform.position = new Vector3(x_position,
XR.transform.position.y, z_position);
        else
            XR.transform.position = new Vector3(x_position + 3.5f,
XR.transform.position.y, z_position - 1.5f);
        yield return new WaitForSeconds(0.5f);

    }

}
```

Παράρτημα 5

Puzzle

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

public class Puzzle : MonoBehaviour
{
```

```
public Texture2D image;

public int blockPerLine = 3;
public int shuffleLength = 5;
public float defaultMoveDuration = .2f;
public float shuffleMoveDuaration = .1f;
private bool start = true;
[SerializeField] private GameObject nextPage;
[SerializeField] private GameObject nextPageGRE;
[SerializeField] private GameObject arrow;
[SerializeField] private GameObject text;

[SerializeField] private GameObject textUnlock;

[SerializeField] private AudioSource textUnlockAudio;
enum PuzzleState
{
    Solved,
    Shuffling,
    InPlay
};

PuzzleState state;

Block[,] blocks;
Block emptyBlock;
Queue<Block> inputs;
bool blockIsMoving;
int shuffleMovesRemaining;
Vector2Int previousShuffleOffset;

private float timerValue;
private float MaxtimerValue=20.0f;
private bool timerFlag = false;
int temp1 = 0;
private void Start()
{
    //CreatePuzzle();
    timerValue = 0;
}

private void Update()
{
    if (timerFlag == false)//an den exei luthei to puzzle
    {
        timerValue += Time.deltaTime;
    }

    if(start)
    {
        start = false;
        StartShuffle();
        //text.gameObject.SetActive(false);
    }
}
//changed name for CreatPuzzle()
public void OnEnable()
{
    start = true;
    blocks = new Block[blockPerLine, blockPerLine];
    Texture2D[,] imageSlices = ImageSlicer.GetSlices(image, blockPerLine);
    Vector2 posVector = new Vector2(2, 1);
    for (int y = 0; y < blockPerLine; y++)
```

```
        {
            for (int x = 0; x < blockPerLine; x++)
            {
                GameObject blockObject =
GameObject.CreatePrimitive(PrimitiveType.Quad);
                //blockObject.AddComponent<XRSimpleInteractable>();
                blockObject.transform.position = -posVector * (blockPerLine -
1) * 0.2f + new Vector2(x, y);
                blockObject.transform.parent = transform;


                Block block = blockObject.AddComponent<Block>();
                block.OnBlockPressed += PlayerMoveBlockInput;
                block.OnFinishedMoving += OnBlockFinishedMoving;
                block.Init(new Vector2Int(x, y), imageSlices[x,y]);
                blocks[x, y] = block;
                if(y == 0 && x == blockPerLine - 1)
                {
                    emptyBlock = block;
                }
            }
        }
        gameObject.transform.localScale = gameObject.transform.localScale / 2;
        inputs = new Queue<Block>();

    }

    public void OnDisable()
    {
        foreach(Block block in blocks)
        {
            Destroy(block.gameObject);
        }
    }

    void PlayerMoveBlockInput(Block blockToMove)
    {
        if (state == PuzzleState.InPlay)
        {
            inputs.Enqueue(blockToMove);
            MakeNextMove();
        }
    }

    void MakeNextMove()
    {
        while (inputs.Count > 0 && !blockIsMoving)
        {
            MoveBlock(inputs.Dequeue(), defaultMoveDuration);

        }
    }

    void MoveBlock(Block blockToMove, float duration)
    {
        if ((blockToMove.coord - emptyBlock.coord).sqrMagnitude == 1)
        {
            blocks[blockToMove.coord.x, blockToMove.coord.y] = emptyBlock;
            blocks[emptyBlock.coord.x, emptyBlock.coord.y] = blockToMove;

            Vector2Int targetCoord = emptyBlock.coord;
            emptyBlock.coord = blockToMove.coord;
            blockToMove.coord = targetCoord;
```

```
            Vector2 targetPosition = emptyBlock.transform.position;
            emptyBlock.transform.position = blockToMove.transform.position;
            blockToMove.MoveToPosition(targetPosition, duration);
            blockIsMoving = true;
        }
    }

    void OnBlockFinishedMoving()
    {
        blockIsMoving = false;
        CheckIfSolved();
        if (state == PuzzleState.InPlay)
        {
            MakeNextMove();
        }
        else if (state == PuzzleState.Shuffling)
        {
            if (shuffleMovesRemaining > 0)
            {
                MakeNextShuffleMove();
            }
            else
            {
                state = PuzzleState.InPlay;
            }
        }
    }

    public void StartShuffle()
    {
        state = PuzzleState.Shuffling;
        shuffleMovesRemaining = shuffleLength;
        emptyBlock.gameObject.SetActive(false);
        MakeNextShuffleMove();
    }

    void MakeNextShuffleMove()
    {
        Vector2Int[] offsets = { new Vector2Int(1, 0), new Vector2Int(-1, 0),
new Vector2Int(0, 1), new Vector2Int(0, -1) };

        int randomIndex = Random.Range(0, offsets.Length);
        for(int i =0; i < offsets.Length; i++)
        {
            Vector2Int offset = offsets[(randomIndex + i) % offsets.Length];
            if (offset != previousShuffleOffset * -1)
            {
                Vector2Int moveBlockCoord = emptyBlock.coord + offset;

                if (moveBlockCoord.x >= 0 && moveBlockCoord.x < blockPerLine &&
moveBlockCoord.y >= 0 && moveBlockCoord.y < blockPerLine)
                {
                    MoveBlock(blocks[moveBlockCoord.x, moveBlockCoord.y],
shuffleMoveDuaration);
                    shuffleMovesRemaining--;
                    previousShuffleOffset = offset;
                    break;
                }
            }
        }
```

```
    }

    void CheckIfSolved()
    {
        foreach(Block block in blocks)
        {
            if(!block.IsAtStartingCoord())
            {

                return;
            }
        }

        timerFlag = true;
        ScoreCounter.score += 100;
        temp1 = (int)(timerValue / MaxtimerValue);
        if (temp1 * 10 >= 50)
        {
            ScoreCounter.score -= 50;
        }
        else
        {
            ScoreCounter.score -= temp1 * 10;
        }
        Debug.Log(ScoreCounter.score);



        state = PuzzleState.Solved;

        arrow.SetActive(true);


        textUnlockAudio.Play();
        emptyBlock.gameObject.SetActive(true);
    }


    IEnumerator UnlockedTimer()
    {
        float count = 0;
        text.gameObject.SetActive(false);
        textUnlock.SetActive(true);
        while (count < 4)
        {
            count += Time.deltaTime;
            yield return null;
        }
        textUnlock.SetActive(false);
    }}
```

Παράρτημα 6

FlipMesh

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System.Linq;
public class FlipMesh : MonoBehaviour
{
```

```
    public GameObject CubeMenu;

    void Start()
    {
        Mesh mesh = GetComponent<MeshFilter>().mesh;
        mesh.triangles = mesh.triangles.Reverse().ToArray();
        CubeMenu.transform.position = new
Vector3(Camera.main.transform.position.x, Camera.main.transform.position.y,
Camera.main.transform.position.z);
    }

    private void Update()
    {
        CubeMenu.transform.position = new
Vector3(Camera.main.transform.position.x, Camera.main.transform.position.y,
Camera.main.transform.position.z);
    }

}
```

Παράρτημα 7

MatchLogic

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


public class MatchLogic : MonoBehaviour
{
    static MatchLogic Instance;

    public int maxPoints = 4;
    static public int connectedLines = 0;
    public Text pointsText;
    private int points = 0;

    public float shakeAmount = 2f;
    public GameObject canvas;
    public GameObject paneleng;
    public GameObject panelgre;
    public GameObject panel;
    private float shakeTime = 0.0f;
    private Vector3 initialPosition;
    private bool isScreenShaking = false;

    private AudioSource poof;
    private AudioSource triumph;
    public GameObject ending;

    public GameObject SubmitButtonObject;
    public bool SubmitButton = false;
    public static int PointsCounter=0;

    public GameObject Scoreboard;

    private void Awake()
```

```
    {
        poof = GameObject.Find("PoofCartoon
CTE02_04.1_preview").GetComponent<AudioSource>();
        triumph = GameObject.Find("Triumph").GetComponent<AudioSource>();
    }
    private void Start()
    {
        PointsCounter = 0;
        Instance = this;
        if(ChangeScene.Language=="Greek")
        {
            panel = panelgre;
        }
        else
        {
            panel = paneleng;
        }
    }

    void UpdatePointsText()
    {
        pointsText.text = points + "/" + maxPoints;

        if (points == maxPoints)
        {
            if(PointsCounter==4)
            {
                ScoreCounter.score += 50;
                Debug.Log("+50 Me thn prwth");
                Debug.Log(ScoreCounter.score);
            }
            else
            {
                ScoreCounter.score -= (PointsCounter - 4) * 5;
                Debug.Log("-5 * tis lathos prospatheies");
                Debug.Log(ScoreCounter.score);
            }
            ScoreCounter.score += 20;
            Debug.Log("+20 Standard");
            Debug.Log(ScoreCounter.score);
            panel.SetActive(false);


            StartCoroutine(EndingScene());
            Debug.Log(connectedLines);
        }
    }

    public static void AddPoint()
    {
        PointsCounter += 1;
        AddPoints(1);

    }

    public static void RemovePoint()
    {
        AddPoints(-1);
    }

    private static void AddPoints(int points)
    {
        Instance.points += points;
```

```
        Instance.UpdatePointsText();
    }



    void Update()
    {
        if (shakeTime > 0)
        {
            canvas.transform.position = UnityEngine.Random.insideUnitSphere *
shakeAmount*0.01f + initialPosition;
            shakeTime -= Time.deltaTime;
        }
        else if (isScreenShaking)
        {
            isScreenShaking = false;
            shakeTime = 0.0f;
            canvas.transform.position = initialPosition;

        }

    }

    public void ScreenShakeForTime(float time)
    {
        initialPosition = canvas.transform.position;
        shakeTime = time;
        isScreenShaking = true;
    }


    IEnumerator EndingScene()
    {
        float count = 0.0f;
        while (count < 2)
        {
            count += Time.deltaTime;
            yield return null;
        }
        poof.Play();
        ending.SetActive(true);
        triumph.Play();
        Scoreboard.SetActive(true);


        yield return null;
    }
}
```

Παράρτημα 8

MatchItem

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

```
public class MatchItem : MonoBehaviour, IPointerDownHandler, IDragHandler,
IPointerEnterHandler, IPointerUpHandler
{
    static MatchItem hoverItem;


    public string itemName;

    public GameObject line;
    private LineRenderer lineRenderer;
    bool isMatched = false;
    private Vector3 mousePos;
    private Vector3 startPos;

    void Start()
    {
        lineRenderer = line.GetComponent<LineRenderer>();
        lineRenderer.positionCount = 2;
    }


    public void OnPointerDown(PointerEventData eventData)
    {
        if (isMatched)
        {
            MatchLogic.RemovePoint();
            isMatched = false;
        }
            startPos = eventData.pointerCurrentRaycast.worldPosition;
            mousePos = eventData.pointerCurrentRaycast.worldPosition;
            lineRenderer.SetPosition(0, new Vector3(startPos.x, startPos.y,
startPos.z));
            lineRenderer.SetPosition(1, new Vector3(mousePos.x, mousePos.y, -
mousePos.z));


    }

    public void OnDrag(PointerEventData eventData)
    {
        Debug.Log(eventData.position);
        mousePos = eventData.pointerCurrentRaycast.worldPosition;
        lineRenderer.SetPosition(1, new Vector3(mousePos.x, mousePos.y, 0f));
    }

    public void OnPointerUp(PointerEventData eventData)
    {
        if (!this.Equals(hoverItem) && itemName.Equals(hoverItem.itemName))
        {
            // UpdateLine(hoverItem.transform.position);
            lineRend using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.EventSystems;
using UnityEngine.UI;

public class MatchItem : MonoBehaviour, IPointerDownHandler, IDragHandler,
IPointerEnterHandler, IPointerUpHandler
{
    static MatchItem hoverItem;


    public string itemName;
```

```
    public GameObject line;
    private LineRenderer lineRenderer;
    bool isMatched = false;
    private Vector3 mousePos;
    private Vector3 startPos;

    void Start()
    {
        lineRenderer = line.GetComponent<LineRenderer>();
        lineRenderer.positionCount = 2;
    }


    public void OnPointerDown(PointerEventData eventData)
    {
        if (isMatched)
        {
            MatchLogic.RemovePoint();
            isMatched = false;
        }
            startPos = eventData.pointerCurrentRaycast.worldPosition;
            mousePos = eventData.pointerCurrentRaycast.worldPosition;
            lineRenderer.SetPosition(0, new Vector3(startPos.x, startPos.y,
startPos.z));
            lineRenderer.SetPosition(1, new Vector3(mousePos.x, mousePos.y, -
mousePos.z));


    }

    public void OnDrag(PointerEventData eventData)
    {
        Debug.Log(eventData.position);
        mousePos = eventData.pointerCurrentRaycast.worldPosition;
        lineRenderer.SetPosition(1, new Vector3(mousePos.x, mousePos.y, 0f));
    }

    public void OnPointerUp(PointerEventData eventData)
    {
        if (!this.Equals(hoverItem) && itemName.Equals(hoverItem.itemName))
        {
            // UpdateLine(hoverItem.transform.position);
            lineRenderer.SetPosition(1, new
Vector3(hoverItem.gameObject.transform.position.x,
hoverItem.gameObject.transform.position.y,
hoverItem.gameObject.transform.position.z));
            //Destroy(hoverItem);
            //Destroy(this);
            MatchLogic.AddPoint();
            isMatched = true;
            MatchLogic.connectedLines += 1;
        }
        else if (hoverItem != null)
        {
            //lineRenderer.SetPosition(0, new Vector3(0f,0f, 0f));
            lineRenderer.SetPosition(1, new
Vector3(hoverItem.gameObject.transform.position.x,
hoverItem.gameObject.transform.position.y));
            MatchLogic.connectedLines += 1;
            MatchLogic.PointsCounter++;
        }
```

```
        else
        {
            MatchLogic.connectedLines -= 1;
            //MatchLogic.PointsCounter++;
        }
    }

    public void OnPointerEnter(PointerEventData eventData)
    {

        hoverItem = this;
    }




}
erer.SetPosition(1, new Vector3(hoverItem.gameObject.transform.position.x,
hoverItem.gameObject.transform.position.y,
hoverItem.gameObject.transform.position.z));
            //Destroy(hoverItem);
            //Destroy(this);
            MatchLogic.AddPoint();
            isMatched = true;
            MatchLogic.connectedLines += 1;
        }
        else if (hoverItem != null)
        {
            //lineRenderer.SetPosition(0, new Vector3(0f,0f, 0f));
            lineRenderer.SetPosition(1, new
Vector3(hoverItem.gameObject.transform.position.x,
hoverItem.gameObject.transform.position.y));
            MatchLogic.connectedLines += 1;
            MatchLogic.PointsCounter++;
        }

        else
        {
            MatchLogic.connectedLines -= 1;
            //MatchLogic.PointsCounter++;
        }
    }

    public void OnPointerEnter(PointerEventData eventData)
    {

        hoverItem = this;
    }

}
```

Παράρτημα 9

Keyboard

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

```
public class Keyboard : MonoBehaviour
{
    public word word;
    public NameInput nameInput;
    public void  pressedLetter()
    {
        if (word != null)
        {

word.setLetter(gameObject.transform.GetChild(0).GetComponent<Text>().text.ToUpp
er());
            gameObject.GetComponent<Button>().interactable = false;
        }
        if (nameInput != null)
        {

nameInput.setLetter(gameObject.transform.GetChild(0).GetComponent<Text>().text.
ToUpper());
        }
    }

}
```

Παράρτημα 10

NameInput

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using TMPro;
using UnityEngine.UI;
using Nem.Scoreboards;
public class NameInput : MonoBehaviour
{
    public static string PlayersName="";
    public GameObject InputReader;
    public string letters = ".";

    public AudioSource audioClip;
    [SerializeField]
    private string letter = "";

    public TextMeshProUGUI wrongWords;
    public TextMeshProUGUI[] cells;

    //public HighscoreTable HighscoreTable;

    public GameObject NameKeyboard;

    public Scoreboard otherScript;

    private void Start()
    {
        if (ChangeScene.Language == "Greek")
        {
            //letter = ".";
        }
        //Debug.Log(letter);
        InputReader.SetActive(true);
    }
```

```
    // Update is called once per frame
    void Update()
    {
        int counter = 0;
        foreach (var cell in cells)
        {
            if (cell.text != "")
                counter++;
        }
        if (counter == letters.Length)
        {
            transform.parent.gameObject.SetActive(false);
            transform.parent.GetChild(1).gameObject.SetActive(false);
            transform.parent.GetChild(2).gameObject.SetActive(false);
            audioClip.Play();
        }
        if (letter != "")
        {
            //Debug.Log(letter);
            if (letters.Contains(letter))
            {
                //Debug.Log("OK");
                for (int i = 0; i < letters.Length; i++)
                {
                    if (letter.ToUpper() == letters[i].ToString())
                    {
                        cells[i].text = letter;
                    }
                }
            }
            else
            {
                wrongWords.text += letter;
            }
            letter = "";
        }
    }

    public void set using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using TMPro;
using UnityEngine.UI;
using Nem.Scoreboards;
public class NameInput : MonoBehaviour
{
    public static string PlayersName="";
    public GameObject InputReader;
    public string letters = ".";

    public AudioSource audioClip;
    [SerializeField]
    private string letter = "";

    public TextMeshProUGUI wrongWords;
    public TextMeshProUGUI[] cells;

    //public HighscoreTable HighscoreTable;

    public GameObject NameKeyboard;

    public Scoreboard otherScript;
```

```
    private void Start()
    {
        if (ChangeScene.Language == "Greek")
        {
            //letter = ".";
        }
        //Debug.Log(letter);
        InputReader.SetActive(true);
    }
    // Update is called once per frame
    void Update()
    {
        int counter = 0;
        foreach (var cell in cells)
        {
            if (cell.text != "")
                counter++;
        }
        if (counter == letters.Length)
        {
            transform.parent.gameObject.SetActive(false);
            transform.parent.GetChild(1).gameObject.SetActive(false);
            transform.parent.GetChild(2).gameObject.SetActive(false);
            audioClip.Play();
        }
        if (letter != "")
        {
            //Debug.Log(letter);
            if (letters.Contains(letter))
            {
                //Debug.Log("OK");
                for (int i = 0; i < letters.Length; i++)
                {
                    if (letter.ToUpper() == letters[i].ToString())
                    {
                        cells[i].text = letter;
                    }
                }
            }
            else
            {
                wrongWords.text += letter;
            }
            letter = "";
        }
    }

    public void setLetter(string s)
    {
        letter = s;
    }

    public void deleteLetter()
    {
        if(wrongWords.text!="")
            wrongWords.text=wrongWords.text.Remove(wrongWords.text.Length - 1,
1);
        Debug.Log("DeletePressed");
    }

    public void enterLetter()
    {
        PlayersName = wrongWords.text;
```

```
        NameKeyboard.SetActive(false);
        otherScript.AddEntry();
    }
}

Letter(string s)
    {
        letter = s;
    }

    public void deleteLetter()
    {
        if(wrongWords.text!="")
            wrongWords.text=wrongWords.text.Remove(wrongWords.text.Length - 1,
1);
        Debug.Log("DeletePressed");
    }

    public void enterLetter()
    {
        PlayersName = wrongWords.text;
        NameKeyboard.SetActive(false);
        otherScript.AddEntry();
    }
}
```

Παράρτημα 11

Scoreboard

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

namespace Nem.Scoreboards
{

    public class Scoreboard : MonoBehaviour
    {
        [SerializeField] private int maxScoreboardEntries = 5;
        [SerializeField] private Transform highscoresHolderTransform = null;
        [SerializeField] private GameObject scoreboardEntryObject = null;

        [Header("test")]
        [SerializeField] ScoreboardEntryData testEntryData = new
ScoreboardEntryData();

        private string SavePath =>
$"{Application.persistentDataPath}/highscores.json";

        private void Start()
        {
            ScoreboardSaveData savedScores = GetSavedScores();


            UpdateUI(savedScores);

            SaveScores(savedScores);
        }
```

```
        [ContextMenu("Add Test Entry")]
        public void AddTestEntry()
        {
            //testEntryData.entryName
            testEntryData.entryScore = -1;
            AddEntry(testEntryData);
        }

        public void AddEntry()
        {
            testEntryData.entryName = NameInput.PlayersName; //static name
            testEntryData.entryScore = ScoreCounter.score; //static score
            //checking
            if (NameInput.PlayersName == "" || NameInput.PlayersName== null)
                NameInput.PlayersName = "Unknown";
            if (ScoreCounter.score <0)
                ScoreCounter.score = 0;
            AddEntry(testEntryData);
        }

        public void AddEntry(ScoreboardEntryData scoreboardEntryData)
        {
            ScoreboardSaveData savedScores = GetSavedScores();

            bool scoreAdded = false;

            for (int i=0; i<savedScores.highscores.Count; i++)
            {
                if(scoreboardEntryData.entryScore >
savedScores.highscores[i].entryScore)
                {
                    savedScores.highscores.Insert(i, scoreboardEntryData);
                    scoreAdded = true;
                    break;
                }
            }
            if(!scoreAdded && savedScores.highscores.Count <
maxScoreboardEntries)
            {
                savedScores.highscores.Add(scoreboardEntryData);
            }

            if(savedScores.highscores.Count>maxScoreboardEntries)
            {
                savedScores.highscores.RemoveRange(maxScoreboardEntries,
savedScores.highscores.Count - maxScoreboardEntries);

            }

            UpdateUI(savedScores);

            SaveScores(savedScores);
        }
        private void UpdateUI(ScoreboardSaveData savedScores)
        {
            foreach(Transform child in highscoresHolderTransform)
            {
                Destroy(child.gameObject);

            }

            foreach (ScoreboardEntryData highscore in savedScores.highscores)
```

```
            {
                Instantiate(scoreboardEntryObject,
highscoresHolderTransform).GetComponent<ScoreboardEntryUI>().Initialise(highsco
re);

            }
        }
        private ScoreboardSaveData GetSavedScores()
        {
            if (!File.Exists(SavePath))
            {
                File.Create(SavePath).Dispose();
                return new ScoreboardSaveData();
            }

            using (StreamReader stream = new StreamReader(SavePath))
            {
                string json = stream.ReadToEnd();

                return JsonUtility.FromJson<ScoreboardSaveData>(json);
            }
        }

            private voi using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System.IO;

namespace Nem.Scoreboards
{

    public class Scoreboard : MonoBehaviour
    {
        [SerializeField] private int maxScoreboardEntries = 5;
        [SerializeField] private Transform highscoresHolderTransform = null;
        [SerializeField] private GameObject scoreboardEntryObject = null;

        [Header("test")]
        [SerializeField] ScoreboardEntryData testEntryData = new
ScoreboardEntryData();

        private string SavePath =>
$"{Application.persistentDataPath}/highscores.json";

        private void Start()
        {
            ScoreboardSaveData savedScores = GetSavedScores();


            UpdateUI(savedScores);

            SaveScores(savedScores);
        }

        [ContextMenu("Add Test Entry")]
        public void AddTestEntry()
        {
            //testEntryData.entryName
            testEntryData.entryScore = -1;
            AddEntry(testEntryData);
        }

        public void AddEntry()
```

```
        {
            testEntryData.entryName = NameInput.PlayersName; //static name
            testEntryData.entryScore = ScoreCounter.score; //static score
            //checking
            if (NameInput.PlayersName == "" || NameInput.PlayersName== null)
                NameInput.PlayersName = "Unknown";
            if (ScoreCounter.score <0)
                ScoreCounter.score = 0;
            AddEntry(testEntryData);
        }

        public void AddEntry(ScoreboardEntryData scoreboardEntryData)
        {
            ScoreboardSaveData savedScores = GetSavedScores();

            bool scoreAdded = false;

            for (int i=0; i<savedScores.highscores.Count; i++)
            {
                if(scoreboardEntryData.entryScore >
savedScores.highscores[i].entryScore)
                {
                    savedScores.highscores.Insert(i, scoreboardEntryData);
                    scoreAdded = true;
                    break;
                }
            }
            if(!scoreAdded && savedScores.highscores.Count <
maxScoreboardEntries)
            {
                savedScores.highscores.Add(scoreboardEntryData);
            }

            if(savedScores.highscores.Count>maxScoreboardEntries)
            {
                savedScores.highscores.RemoveRange(maxScoreboardEntries,
savedScores.highscores.Count - maxScoreboardEntries);

            }

            UpdateUI(savedScores);

            SaveScores(savedScores);
        }
        private void UpdateUI(ScoreboardSaveData savedScores)
        {
            foreach(Transform child in highscoresHolderTransform)
            {
                Destroy(child.gameObject);

            }

            foreach (ScoreboardEntryData highscore in savedScores.highscores)
            {
                Instantiate(scoreboardEntryObject,
highscoresHolderTransform).GetComponent<ScoreboardEntryUI>().Initialise(highsco
re);

            }
        }
        private ScoreboardSaveData GetSavedScores()
        {
            if (!File.Exists(SavePath))
```

```
            {
                File.Create(SavePath).Dispose();
                return new ScoreboardSaveData();
            }

            using (StreamReader stream = new StreamReader(SavePath))
            {
                string json = stream.ReadToEnd();

                return JsonUtility.FromJson<ScoreboardSaveData>(json);
            }
        }

            private void SaveScores(ScoreboardSaveData scoreboardSaveData)
            {
                using(StreamWriter stream = new StreamWriter(SavePath))
                {
                    string json = JsonUtility.ToJson(scoreboardSaveData, true);

                stream.Write(json);
                }
            }

    }
}

d SaveScores(ScoreboardSaveData scoreboardSaveData)
            {
                using(StreamWriter stream = new StreamWriter(SavePath))
                {
                    string json = JsonUtility.ToJson(scoreboardSaveData, true);

                stream.Write(json);
                }
            }

    }
}
```

Παράρτημα 12

SwitchPhotos

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

public class SwitchPhotos : MonoBehaviour
{
    public GameObject[] background;
    public GameObject[] backgroundGRE;
    public int maxIndex;
    public int index;

    void Awake()
    {
        index = 0;
        maxIndex = maxIndex - 1;
```

```
        if (index == 0)
        {
            background[0].gameObject.SetActive(true);

            backgroundGRE[0].gameObject.SetActive(true);
        }
    }


    void Update()
    {
        if (index >= maxIndex)
            index = maxIndex;

        if (index < 0)
            index = 0;

    }

    public void Next()
    {
        if(index<maxIndex)
            index += 1;

        for (int i = 0; i < background.Length; i++)
        {
            background[i].gameObject.SetActive(false);
            background[index].gameObject.SetActive(true);

            backgroundGRE[i].gameObject.SetActive(false);
            backgroundGRE[index].gameObject.SetActive(true);

        }
        Debug.Log(index);
    }

    public void Previous()
    {
        if (index> using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class SwitchPhotos : MonoBehaviour
{
    public GameObject[] background;
    public GameObject[] backgroundGRE;
    public int maxIndex;
    public int index;

    void Awake()
    {
        index = 0;
        maxIndex = maxIndex - 1;

        if (index == 0)
        {
            background[0].gameObject.SetActive(true);

            backgroundGRE[0].gameObject.SetActive(true);
        }
    }
```

```
    void Update()
    {
        if (index >= maxIndex)
            index = maxIndex;

        if (index < 0)
            index = 0;

    }

    public void Next()
    {
        if(index<maxIndex)
            index += 1;

        for (int i = 0; i < background.Length; i++)
        {
            background[i].gameObject.SetActive(false);
            background[index].gameObject.SetActive(true);

            backgroundGRE[i].gameObject.SetActive(false);
            backgroundGRE[index].gameObject.SetActive(true);

        }
        Debug.Log(index);
    }

    public void Previous()
    {
        if (index>0)
            index -= 1;

        for (int i = 0; i < background.Length; i++)
        {
            background[i].gameObject.SetActive(false);
            background[index].gameObject.SetActive(true);

            backgroundGRE[i].gameObject.SetActive(false);
            backgroundGRE[index].gameObject.SetActive(true);
        }
        Debug.Log(index);
    }
}
```