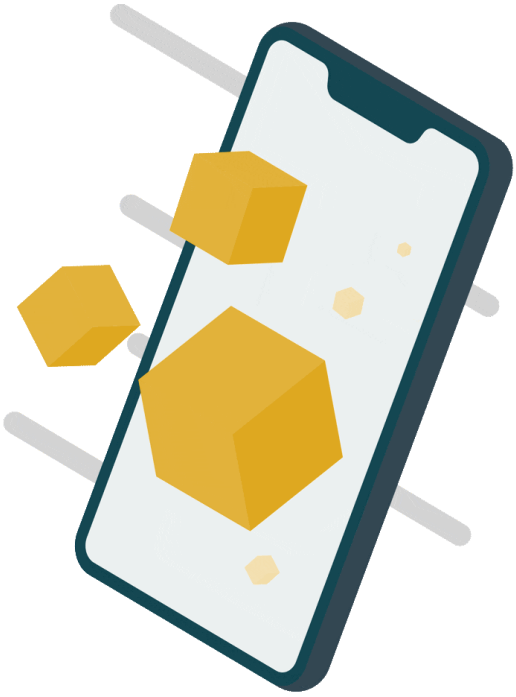
ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Σχεδιασμός και Υλοποίηση Εφαρμογής Online Παιχνιδιού με χρήση Επαυξημένης Πραγματικότητας



ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΣΠΥΡΟΣ ΜΠΑΚΑΛΑΚΟΣ

ΑΜ: 71345405

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: ΕΥΑΓΓΕΛΟΣ ΚΟΣΜΑΤΟΣ ΕΠΙΣΤΗΜΟΝΙΚΟΣ ΣΥΝΕΡΓΑΤΗΣ, ΓΕΩΡΓΙΟΣ ΠΡΕΖΕΡΑΚΟΣ ΚΑΘΗΓΗΤΗΣ ΠΑΔΑ

ΑΘΗΝΑ, ΜΑΡΤΙΟΣ 2022

ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Σχεδιασμός και Υλοποίηση Εφαρμογής Online Παιχνιδιού με χρήση Επαυξημένης Πραγματικότητας

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΣΠΥΡΟΣ ΜΠΑΚΑΛΑΚΟΣ

ΑΜ: 71345405

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: ΕΥΑΓΓΕΛΟΣ ΚΟΣΜΑΤΟΣ ΕΠΙΣΤΗΜΟΝΙΚΟΣ ΣΥΝΕΡΓΑΤΗΣ, ΓΕΩΡΓΙΟΣ ΠΡΕΖΕΡΑΚΟΣ ΚΑΘΗΓΗΤΗΣ ΠΑΔΑ

ΕΞΕΤΑΣΘΗΚΕ ΕΠΙΤΥΧΩΣ ΑΠΟ ΤΗΝ ΚΑΤΩΘΙ ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ:

Α/α	Ονοματεπώνυμο	Βαθμίδα/Ιδιότητα	Ψηφιακή Υπογραφή
1	**Γιώργος Πρεζεράκος**	Καθηγητής	Georgios Prezerakos Digitally signed by Georgios Prezerakos Date: 2022.07.25 10:32:46 +03'00'
2	**Παναγιώτης Γιαννακόπουλος**	Καθηγητής	Panagiotis Giannakopoulos Digitally signed by Panagiotis Giannakopoulos Date: 2022.07.29 08:54:36 +07'00'
3	**Νικόλαος Ζάχαρης**	Καθηγητής	Digitally signed by Nikolaos Zacharis Date: 2022.07.25 11:11:53 +03'00'
4	**Ευάγγελος Κοσμάτος**	Επιστημονικός Συνεργάτης	

ΑΘΗΝΑ, ΙΟΥΛΙΟΣ 2022

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος ΜΠΑΚΑΛΑΚΟΣ ΣΠΥΡΙΔΩΝ του ΚΩΝΣΤΑΝΤΙΝΟΥ, με αριθμό μητρώου 71345405 φοιτητής του Πανεπιστημίου Δυτικής Αττικής της Σχολής ΜΗΧΑΝΙΚΩΝ του Τμήματος ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ, δηλώνω υπεύθυνα ότι:

«Είμαι συγγραφέας αυτής της πτυχιακής/διπλωματικής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.

Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου».

Ο Δηλών

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω τον καθηγητή κ. Ευάγγελο Κοσμάτο για την πολύτιμη καθοδήγηση και βοήθεια που μου παρείχε κατά την διάρκεια εκπόνησης της διπλωματικής εργασίας μου καθώς και για τις γνώσεις που μου παρείχε σε όλα τα χρόνια των σπουδών μου.
Ακόμη, θέλω να ευχαριστήσω τους φίλους και κυρίως την οικογένειά μου για την πολύτιμη υποστήριξη αλλά και βοήθεια που μου προσέφεραν καθόλη την διάρκεια ανάπτυξης της εφαρμογής. Επίσης εκφράζω ένα θερμό ευχαριστώ στους γονείς μου που στάθηκαν δίπλα μου σε όλα τα χρόνια των σπουδών μου.

ΠΕΡΙΛΗΨΗ

Η εν λόγω εργασία αποσκοπεί στην ενασχόληση και εξέλιξη μιας νέας τεχνολογίας όπως είναι η Εικονική Πραγματικότητα (VR) και η Επαυξημένη Πραγματικότητα (AR), τεχνολογίες οι οποίες έχουν κάνει πρόσφατα ένα τεράστιο άλμα και υπόσχονται να αλλάξουν τον τρόπο με τον οποίο αλληλοεπιδρούμε σε πάρα πολλούς τομείς της καθημερινότητά μας.

Στόχος της συγκεκριμένης διπλωματικής είναι η εκπόνηση εισ-βάθους έρευνας των συγκεκριμένων τεχνολογιών. Για τον σκοπό της έρευνας αυτής σχεδιάστηκε και αναπτύχθηκε μια διαδραστική εφαρμογή εικονικού παιχνιδιού καρτών με χρήση τεχνολογίας Επαυξημένης Πραγματικότητας (Augmented Reality – AR) έτσι ώστε να μελετήσουμε την λειτουργία και την επίδραση της τεχνολογίας αυτής στο πεδίο της ανάπτυξης ηλεκτρονικών παιχνιδιών και να ανακαλύψουμε νέους τρόπους αλληλεπίδρασης με τον εικονικό κόσμο.

ABSTRACT

This work encourages the engagement and development with new technologies such as Virtual Reality (VR) and Augmented Reality (AR), technologies that have recently made a huge leap and promise to change the way we interact in many areas of our everyday life.

The aim of this thesis is to conduct in-depth research upon these technologies. In the scope of this research an interactive Augmented Reality (AR) mobile card game was developed so that we can examine the functionality and effect of this technology in game development and discover other perspectives of interacting with the virtual world.

ΠΕΡΙΕΧΟΜΕΝΑ

ΜΕΡΟΣ 1: ΘΕΩΡΙΑ

ΚΕΦΑΛΑΙΟ 1: ΓΡΑΦΙΚΑ ΥΠΟΛΟΓΙΣΤΩΝ

1.1 ΕΙΣΑΓΩΓΗ ΣΤΗΝ ΕΝΝΟΙΑ ΤΩΝ ΓΡΑΦΙΚΩΝ ΥΠΟΛΟΓΙΣΤΗ

Ο πιο κοινά αποδεκτός τρόπος για να ορίσουμε την έννοια των *Γραφικών Υπολογιστών* είναι πως ο όρος αυτός χρησιμοποιείται για να περιγράψει κάθε εργασία με χρήση ηλεκτρονικού υπολογιστή, η οποία αποσκοπεί στην δημιουργία ή την επεξεργασία εικόνων. Εικόνες κάθε τύπου και μορφής μπορούν να αναπαρασταθούν ψηφιακά με χρήση κατάλληλων εργαλείων για την παραγωγή γραφικών σε υπολογιστή. Οι εικόνες αυτές μπορεί να είναι απλά γεωμετρικά σχήματα που συνθέτουν ένα σχέδιο, ψηφιακές ζωγραφιές ή ακόμα και ρεαλιστικά οπτικά εφέ, γραφικές απεικονίσεις τεχνικού ή πληροφοριακού χαρακτήρα, πολύπλοκα και θεαματικά animations κ.α.

Τα γραφικά υπολογιστών είναι ένα πολύ ευρύ επιστημονικό πεδίο, οι αλλαγές και οι εξελίξεις του οποίου σημειώνονται με εξαιρετικά ταχύ ρυθμό. Τα γραφικά χωρίζονται σε εικόνες δύο διαστάσεων (2D) ή τριών διαστάσεων (3D) και μπορεί να είναι εξ' ολοκλήρου φτιαγμένες από τον ηλεκτρονικό υπολογιστή ή να είναι αποτέλεσμα επεξεργασίας άλλων εικόνων. Η διαδικασία απεικόνισης των εκάστοτε γραφικών πάνω σε μια δισδιάστατη συσκευή απεικόνισης όπως είναι η οθόνη γίνεται με τον κατάλληλο χρωματισμό των εικονοστοιχείων (pixels) που συνθέτουν ένα κανονικό πλέγμα στην επιφάνεια της οθόνης και κάθε ένα απο αυτά μπορεί να λάβει μια χρωματική τιμή έτσι ώστε να δημιουργηθεί κάποια εικόνα. Υπεύθυνοι για τον σωστό χρωματισμό των εικονοστοιχείων είναι ειδικοί αλγόριθμοι σχεδίασης όπως ο αλγόριθμος του Bresenham.

1.2 ΙΣΤΟΡΙΚΗ ΑΝΑΔΡΟΜΗ

Πριν όμως επεκταθούμε στις τεχνικές λεπτομέρειες και τα μαθηματικά που παίζουν κυρίαρχο ρόλο στην επιστήμη των γραφικών υπολογιστών, ιδιαίτερο ενδιαφέρον θα έχει να κάνουμε μια ανασκόπηση στην σύντομη ιστορία των γραφικών από την πρώτη τους εμφάνιση έως και σήμερα.

Ο όρος γραφικά υπολογιστή (computer graphics) χρησιμοποιήθηκε για πρώτη φορά στις αρχές της δεκαετίας του 1960 από ερευνητές για να περιγράψουν την εργασία τους με στόχο την δημιουργία διανυσματικών εικόνων σε έναν ψηφιακό υπολογιστή. Πατέρας της επιστήμης των γραφικών θεωρείται ο *Ivan Sutherland* που το 1963 παρουσίασε στο MIT την πρωτοποριακή εργασία του, το σύστημα *Sketchpad*. Το Sketchpad αποτελούσε μια προσπάθεια δημιουργίας μιας αποτελεσματικής αμφίδρομης επικοινωνίας μεταξύ ανθρώπου

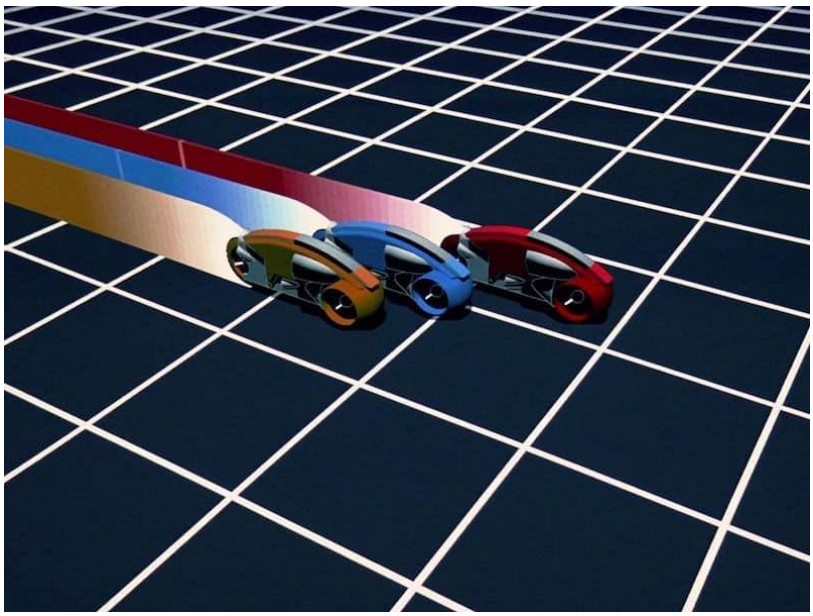
και υπολογιστή και είναι ο πρόγονος πολλών σημερινών εργαλείων σχεδίασης με βοήθεια υπολογιστή (CAD). Ο Sutherland μάλιστα κέρδισε και το βραβείο Turing για την εργασία αυτή.

Εικόνα 1. Ο Ivan Sutherland και το σύστημα Sketchpad, 1963

Το 1967 οι επιφάνειες Coons έφεραν επανάσταση στον τρόπο με τον οποίο γινόταν η γεωμετρική σχεδίαση με υπολογιστή. Οι επιφάνειες Coons επέτρεπαν την κατασκευή επιφανειών απο στοιχειώδη τμήματα που μπορούσαν να συνδεθούν μεταξύ τους παρέχοντας περιορισμούς συνέχειας στα σύνορά τους. Ήταν ο πρόδρομος των επιφανειών Bezier και B-spline που χρησιμοποιούμε σήμερα ευρέως στην γεωμετρική σχεδίαση με υπολογιστή.

Στα μέσα της δεκαετίας του 1970, η εμφάνιση της μνήμης τυχαίας προσπέλαση (RAM) έπαιξε σημαντικό ρόλο στην δημιουργία του καταχωρητή πλαισίου μέσω του οποίου επιτράπηκε ο διαχωρισμός από την δημιουργία μιας εικόνας με την ανανέωση της συσκευής απεικόνισης με αποτέλεσμα την παραγωγή πολύπλοκων συνθετικών εικόνων. Αργότερα, το 1974, η εισαγωγή του καταχωρητή βάθους (z-buffer) ήρθε για να προσθέσει άλλο ένα στοιχείο πολυπλοκότητας στην κατασκευή συνθετικών εικόνων, αυτό του βάθους, όπως υποδηλώνει και το όνομά του. Ο z-buffer συντέλεσε στην εμφάνιση των κρυμμένων επιφανειών μιας εικόνας και παίζει από τότε κυρίαρχο ρόλο σε κάθε μονάδα επεξεργασίας γραφικών (GPU).

Την δεκαετία του 1980 παρατηρείται η μεγάλη μετάβαση των γραφικών υπολογιστών από ερευνητικό επίπεδο σε μια ευρέως διαδεδομένη τεχνολογία. Πλέον πρόσβαση σε εφαρμογές αποτελούμενες από γραφικά παραγόμενα από υπολογιστή δεν έχουν μόνο ερευνητές και ελάχιστοι άνθρωποι του χώρου αλλά η τεχνολογία είναι διαθέσιμη για το ευρύ κοινό από τον κινηματογράφο, την τηλεόραση μέχρι και οικιακές ηλεκτρονικές συσκευές. Το 1980 κυκλοφορεί το TRON, η πρώτη ταινία που περιλάμβανε εκτεταμένη χρήση συνθετικών εικόνων. Ακόμα, το 1987 αναγνωρίζεται επίσημα από το Εθνικό Ίδρυμα Επιστημών των Η.Π.Α η χρηματοδότηση του πεδίου.



Εικόνα 2. Σκηνή απο την ταινία TRON (1982)

Το 1990 τα γραφικά γνώρισαν την μεγαλύτερη ανάπτυξή τους καθώς η ραγδαία εξέλιξη που γνώρισε εκείνη την εποχή η τεχνολογία είχε ως επακόλουθο την κυκλοφορία προϊόντων που θα ενίσχυαν σημαντικά την εφαρμογή των γραφικών και της οπτικοποίησης. Τέτοια συστήματα ήταν το Visualization Data Explorer (1991) της IBM, το OpenDX, το οποίο αποτέλεσε μετεξέλιξη του προηγούμενου, το OpenGL της SGI (1992) και το Direct3D της Microsoft (1995) τα οποία είναι πρότυπα διεπαφής εφαρμογών (APIs). Επίσης, από το 1998 εισήχθησαν μαζικά στην αγορά οι τρισδιάστατοι επιταχυντές γραφικών (GPUs) για κάθε προσωπικό υπολογιστή.

Με την απαρχή της νέας χιλιετίας, οι αυξανόμενες απαιτήσεις σε γραφικά τόσο βιντεοπαιχνιδιών όσο και εφαρμογών λογισμικού συντελούν στην συνεχή εξέλιξη και τον εμπλουτισμό των μονάδων επεξεργασίας γραφικών με νέα χαρακτηριστικά και λειτουργίες. Ο ρυθμός ανάπτυξης των επιταχυντών γραφικών έχει ξεπεράσει πλέον αυτό των επεξεργαστών.

Την δεκαετία που μας πέρασε το CGI ήταν κυρίαρχο σε ταινίες, video, διαφημίσεις, βιντεοπαιχνίδια και τα γραφικά μπορούν να αποδώσουν σχεδόν με επιστημονική ακρίβεια σε πραγματικό χρόνο φωτορεαλιστικά αποτελέσματα ακόμα και σε μη εκπαιδευμένο μάτι. Οι περισσότεροι προσωπικοί υπολογιστές είναι ικανοί για απεικόνιση αλλά και δημιουργία εξαιρετικά προηγμένων τρισδιάστατων γραφικών. Η δυνατότητα παραγωγής γραφικών και οπτικοποίησης είναι πλέον διαθέσιμη σε όλους.

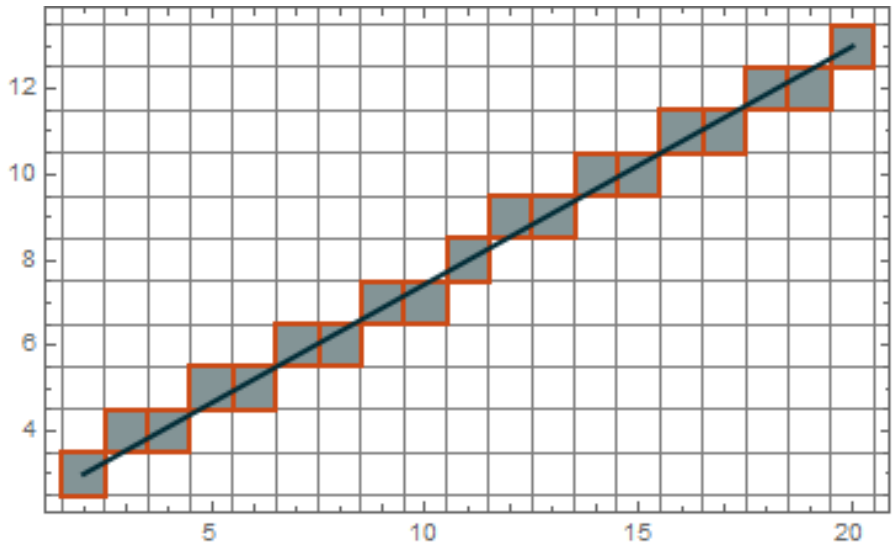
1.3 ΤΑ ΜΑΘΗΜΑΤΙΚΑ ΣΤΗΝ ΣΧΕΔΙΑΣΗ ΓΡΑΦΙΚΩΝ

Μπορούμε να θεωρήσουμε πως τα γραφικά υπολογιστών γεννιούνται από την κατάλληλη μετατροπή μαθηματικών σε κώδικα. Είναι σίγουρο πως η βάση της επιστήμης των γραφικών

είναι τα μαθηματικά και στο κεφάλαιο αυτό θα αναφέρουμε επιγραμματικά τους σημαντικότερους αλγόριθμους για την σχεδίαση ευθειών, κύκλοων, καμπυλών, περικοπής, απόκρυψης κ.α.

Αλγόριθμος Bresenham για ευθύγραμμα τμήματα

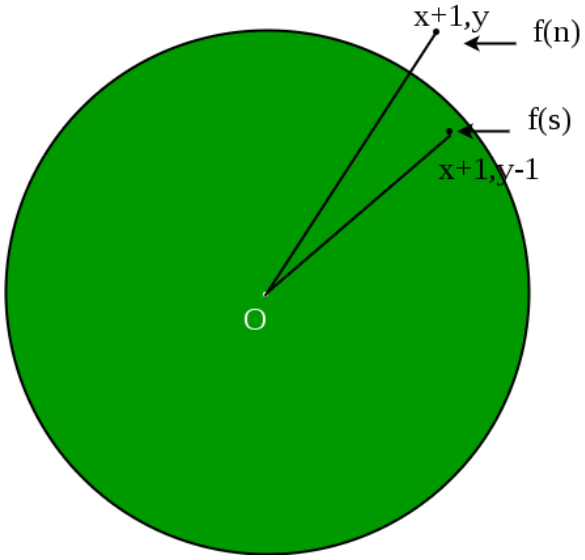
Ο αλγόριθμος του Bresenham για την δημιουργία ευθύγραμμων τμημάτων είναι ένας από τους παλαιότερους και απλούς αλγόριθμους αντιστοίχισης ενός γεωμετρικού αντικειμένου με εικονοστοιχεία της οθόνης. Ο αλγόριθμος αυτός αποφασίζει πιο θα είναι το επόμενο pixel που θα χρωματιστεί για την δημιουργία ευθείας βασιζόμενος σε έναν περιορισμό της επιτρεπόμενης κλίσης της γραμμής στο εύρος $0 \leq m \leq 1$. Έτσι, δουλεύοντας στο πρώτο θετικό οκταμόριο του επιπέδου, το γραμμικό σχέδιο γίνεται θέμα απόφασης μεταξύ δύο δυνατοτήτων σε κάθε βήμα.



Εικόνα 3. Σχεδίαση ευθείας με τον αλγόριθμο Bresenham

Αλγόριθμος Bresenham για κύκλο

Ο Αλγόριθμος υπολογίζει τη θέση των εικονοστοιχείων στο πρώτο οκταμόριο των 45 μοιρών και την επεκτείνει στα άλλα 7 οκταμόρια. Για κάθε εικονοστοιχείο (x, y), ο αλγόριθμος σχεδιάζει ένα εικονοστοιχείο σε καθένα από τα 8 οκταμόρια του κύκλου. Το σημείο (x,y) είναι στο πρώτο οκταμόριο και είναι μέσα στον κύκλο. Το επόμενο σημείο που θα χρωματιστεί το pixel μπορεί να είναι είτε το N(x+1, y) ή το S(x+1, y-1). Ο υπολογισμός γίνεται με την χρήση της παραμέτρου υπολογισμού d. Αν $d \leq 0$, τότε το N(x+1, y) είναι το επόμενο pixel, αλλιώς αν $d > 0$, το S(x+1, y-1) είναι το επόμενο pixel.



Εικόνα 4. Σχεδίαση κύκλου με αλγόριθμο Bresenham.

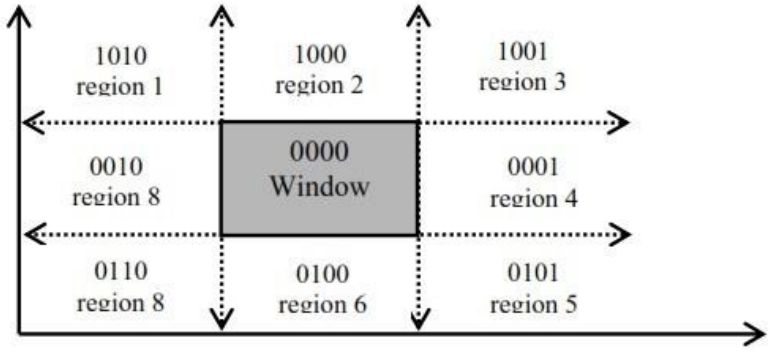
Αλγόριθμος DeCasteljau

Ο αλγόριθμος DeCasteljau χρησιμοποιείται για την εύρεση ενός σημείου πάνω σε μια καμπύλη Bezier. Η θεμελιώδης ιδέα του αλγορίθμου του de Casteljau είναι να επιλέξει ένα σημείο C στο ευθύγραμμο τμήμα AB έτσι ώστε ο λόγος της απόστασης μεταξύ A και C και η απόσταση μεταξύ του A και του B να είναι u. Το διάνυσμα από το A στο B είναι B - A. Δεδομένου ότι το u είναι ένας λόγος στο εύρος 0 και 1, το σημείο C βρίσκεται στο u(B - A). Λαμβάνοντας υπόψη τη θέση του A, το σημείο Γ είναι A + u(B - A) = (1 - u)A + uB. Επομένως, δεδομένου ενός u, (1 - u)A + uB είναι το σημείο C μεταξύ A και B που διαιρεί το AB σε αναλογία u:1-u.

Αλγόριθμος Cohen-Sutherland

Ο αλγόριθμος Cohen-Sutherland είναι ένας τυπικός αλγόριθμος περικοπής ευθύγραμμων τμημάτων σε δυο διαστάσεις. Καταργεί τις γραμμές από ένα δεδομένο σύνολο γραμμών και την ορθογώνια περιοχή ενδιαφέροντος που ανήκει εκτός της περιοχής ενδιαφέροντος και κόβει εκείνες τις γραμμές που βρίσκονται εν μέρει εντός της περιοχής ενδιαφέροντος. Ο αλγόριθμος διαιρεί τον δισδιάστατο χώρο της οθόνης σε 9 περιοχές (οκτώ εξωτερικές περιοχές και μία εντός περιοχής) και στη συνέχεια προσδιορίζει

αποτελεσματικά τις γραμμές και τα τμήματα των γραμμών που είναι ορατά στην κεντρική περιοχή ενδιαφέροντος.



Εικόνα 5. Περικοπή αντικειμένων με τον αλγόριθμο Cohen-Sutherland

Αλγόριθμος του ζωγράφου

Ο painter's algorithm είναι αλγόριθμος περικοπής ο οποίος αξιοποιεί την πληροφορία βάθους ώστε να σχεδιάσει τα αντικείμενα διαδοχικά ξεκινώντας από το μακρινότερο σε σχέση με τον παρατηρητή. Τα στάδια σχεδίασης του αλγορίθμου είναι τα εξής, Ταξινομεί τις διάφορες επιφάνειες που βρίσκονται στην βάση του μειούμενου βάθους τους, ελέγχει για να μετατρέψει τις επιφάνειες που βρίσκονται σε μεγαλύτερο βάθος, κάνει σύγκριση επικαλυπτόμενων επιφανειών ώστε ο χρήστης να αποφασίσει ποια επιφάνεια πρέπει να είναι ορατή. Τέλος εκχωρεί την επιφάνεια αυτή στον buffer ανανέωσης. Η διαδικασία αυτή επαναλαμβάνεται για κάθε επιφάνεια.

1.4 ΕΦΑΡΜΟΓΕΣ

Είναι βέβαιο πως στην σημερινή εποχή τα πεδία δράσης των γραφικών και της οπτικοποίησης είναι πολυπληθή και επικαλυπτόμενα. Σχεδόν κάθε τεχνολογικό επιχείρημα μπορεί να κάνει χρήση των γραφικών με κάποιο τρόπο. Στην συνέχεια θα αναφέρουμε ορισμένους από τους πιο σημαντικούς τομείς της επιστήμης την γραφικών.

Αρχικά, ίσως ο τομέας εφαρμογής των γραφικών που μας έρχεται πιο άμεσα στον νου είναι τα ***βιντεοπαιχνίδια.*** Είναι πλέον η μεγαλύτερη βιομηχανία στο πεδίο των γραφικών και με την αυξανόμενη ζήτηση της αγοράς έχει συμβάλει στην ταχεία άνοδο των επιταχυντών γραφικών.

Έπειτα, τα ***οπτικά εφέ (visual effects)*** που χρησιμοποιούνται στον κινηματογράφο ή και την τηλεόραση είναι αναπόσπαστο κομμάτι της προόδου των γραφικών καθώς υπάρχουν σε

σχεδόν κάθε μεγάλη κινηματογραφική παραγωγή για την απεικόνιση ψηφιακού περιβάλλοντος ή χαρακτήρων.

Τα ***κινούμενα σχέδια (animations)*** εφαρμόζουν την ίδια αρχή με τα visual effect ενώ κάνουν χρήση και 3D μοντέλων ώστε να δημιουργήσουν περιβάλλοντα, αντικείμενα και πρόσωπα χωρίς ωστόσο να επικεντρώνονται στην απεικόνιση της πραγματικότητας.

Ιδιαίτερα χρήσιμη έχει αποδειχτεί η χρήση γραφικών στον τομέα της ***προσομοίωσης*** και της ***επιστημονικής έρευνας***. Μέσω κατάλληλων εργαλείων προσομοίωσης που χρησιμοποιούν εκλεπτυσμένα 3D γραφικά είτε για την καλύτερη έρευνα και μελέτη πειραμάτων σε τομείς όπως η ιατρική, η γεωλογία, οι φυσικές επιστήμες ή η πληροφορική, είτε για την εκπαίδευση εξειδικευμένου προσωπικού μέσω προσομοίωσης κάποιου οργάνου, όπως οι προσομοιωτές πτήσης για πιλότους ή οι προσομοιωτές εκτόξευσης που χρησιμοποιεί η NASA.

Ένα φάσμα των γραφικών υπολογιστών που μερικές φορές παραλείπεται ή υποτιμάται είναι αυτό της ***τέχνης***. Παρόλα αυτά η σύγχρονη τέχνη και η τέχνη με την βοήθεια ηλεκτρονικού υπολογιστή έχει κερδίσει πλέον την αναγνώριση της καλλιτεχνικής κοινότητας. Μέσω των 2D ή 3D γραφικών ο καλλιτέχνης έχει ένα παντοδύναμο εργαλείο και παράλληλα ένα αυτόνομο μέσο καλλιτεχνικής έκφρασης.

Η ***εικονική και επαυξημένη πραγματικότητα*** είναι δύο νέες τεχνολογίες που πηγάζουν από την εξέλιξη των γραφικών και της όρασης υπολογιστή. Οι τεχνολογίες αυτές θα μας απασχολήσουν στην συνέχεια.

ΚΕΦΑΛΑΙΟ 2: ΕΙΚΟΝΙΚΗ ΚΑΙ ΕΠΑΥΞΗΜΕΝΗ ΠΡΑΓΜΑΤΙΚΟΤΗΤΑ

2.1 ΕΙΣΑΓΩΓΗ ΣΤΗΝ ΕΚΤΕΤΑΜΕΝΗ ΠΡΑΓΜΑΤΙΚΟΤΗΤΑ

Με τον όρο Εκτεταμένη Πραγματικότητα (Extended Reality) συμβολίζεται ο συνδυασμός αλληλεπίδρασης του ανθρώπου με γραφικά παραγόμενα από υπολογιστή στον εικονικό κόσμο. Στην δομή της η εκτεταμένη πραγματικότητα (XR) είναι ένα υπερσύνολο Εικονικής Πραγματικότητας (VR), Επαυξημένης Πραγματικότητας (AR) και Μικτής Πραγματικότητας (MR). Η ιδέα της εκτεταμένης πραγματικότητας γεννήθηκε με την εδραίωση της χρήσης Εικονικής και Επαυξημένης πραγματικότητας από προγραμματιστές και εταιρείες τεχνολογίας σε όλο τον κόσμο. Για την κατανόηση όμως της Εκτεταμένης πραγματικότητας πρέπει πρώτα να κατανοήσουμε τις τεχνολογίες που την απαρτίζουν.

Εικονική Πραγματικότητα (Virtual Reality)

Αν και δεν υπάρχει ένας καθολικά αποδεκτός ορισμός για την Εικονική Πραγματικότητα, μιας και έχουν αναπτυχθεί πολλά διαφορετικά πλαίσια εφαρμογής αυτής, μπορούμε να αποδώσουμε την εικονική πραγματικότητα ως την χρήση τεχνολογικών μέσων για την δημιουργία ενός τεχνητού διαδραστικού περιβάλλοντος, το οποίο ο χρήστης μπορεί να αντιλαμβάνεται ως πραγματικό. Για την δημιουργία ενός εικονικού περιβάλλοντος απαιτούνται ένας ηλεκτρονικός υπολογιστής υψηλών επιδόσεων, το κατάλληλο, εξιδανικευμένο λογισμικό για τον προγραμματισμό και τα περιφερειακά για την διάδραση του χρήστη. Κατά την αλληλεπίδραση του ανθρώπου με τον εικονικό κόσμο μπορούν να συμμετέχουν όλες οι αισθήσεις, κυρίως η όραση, η ακοή και η αφή.

Τα χαρακτηριστικά της Εικονικής πραγματικότητας μπορούν να συνοψιστούν στα εξής τρία: την Επικοινωνία-Διάδραση, την έντονη πληροφορία και το κυριότερο, την Εμβύθιση. Εμβύθιση (Immersion) είναι η ψευδαίσθηση που έχει ο χρήστης αναφορικά με την ύπαρξή του μέσα σε ένα εικονικό περιβάλλον και πρακτικά προσεγγίζεται ως ο βαθμός στον οποίο το σύστημα εικονικής πραγματικότητας καταφέρνει να απομονώσει τον χρήστη από το φυσικό του περιβάλλον. Για να επιτευχθεί ένας ικανοποιητικός βαθμός εμβύθισης συνήθως χρησιμοποιούνται ειδικά κράνη, γάντια, 3D γυαλιά ή ολόσωμες φόρμες. Έτσι λοιπόν μπορούμε να αποδώσουμε στην εικονική πραγματικότητα έναν απο τους πρώτους ορισμούς που χρησιμοποιήθηκε το 1989 από τον Jaron Lanier, *“Ένα αλληλεπιδραστικό, τρισδιάστατο περιβάλλον, παραγόμενο από υπολογιστή, στο οποίο μπορεί κάποιος να εμβυθιστεί”*.

Επαυξημένη Πραγματικότητα (Augmented Reality)

Με τον όρο Επαυξημένη Πραγματικότητα αναφερόμαστε σε εφαρμογές που ενσωματώνουν στον πραγματικό κόσμο εικονική πληροφορία παραγμένη απο υπολογιστή. Η θεμελιώδης ιδέα της τεχνολογίας εικονικής πραγματικότητας είναι η προβολή ψηφιακών αντικειμένων και πολυμέσων όπως εικόνων, τρισδιάστατων αντικειμένων, κειμένου, μουσική, βίντεο κ.α με σκοπό της επαύξηση, δηλαδή την βελτίωση σε ποιότητα, ποσότητα ή έκταση του πραγματικού χώρου γύρο από τον χρήστη.

Ο πυρήνας μια εμπειρίας επαυξημένης πραγματικότητας είναι η αλληλεπίδραση του χρήστη στον φυσικό κόσμο με στοιχεία που δεν βρίσκονται πραγματικά εκεί. Αντί για αυτό, είναι εικονικές προσθήκες πληροφορίας που αλλιώς δεν θα μπορούσαμε να δούμε, ακούσουμε ή αγγίξουμε. Από τα παραπάνω λοιπόν προκύπτει πως η Επαυξημένη Πραγματικότητα είναι η τεχνολογία η οποία γνωρίζοντας που βρίσκεται ο χρήστης, προς τα πού κοιτάει, πως είναι ο χώρος στον οποίο βρίσκεται και τι είναι το αντικείμενο με το οποίο αλληλεπιδρά στον πραγματικό κόσμο, επιτρέπει την χωρική και χρονική συσχέτιση πληροφορίας που παράγει ο υπολογιστής και την εμφανίζει σε τρισδιάστατη υπέρθεση με τον φυσικό κόσμο, σε πραγματικό χρόνο.

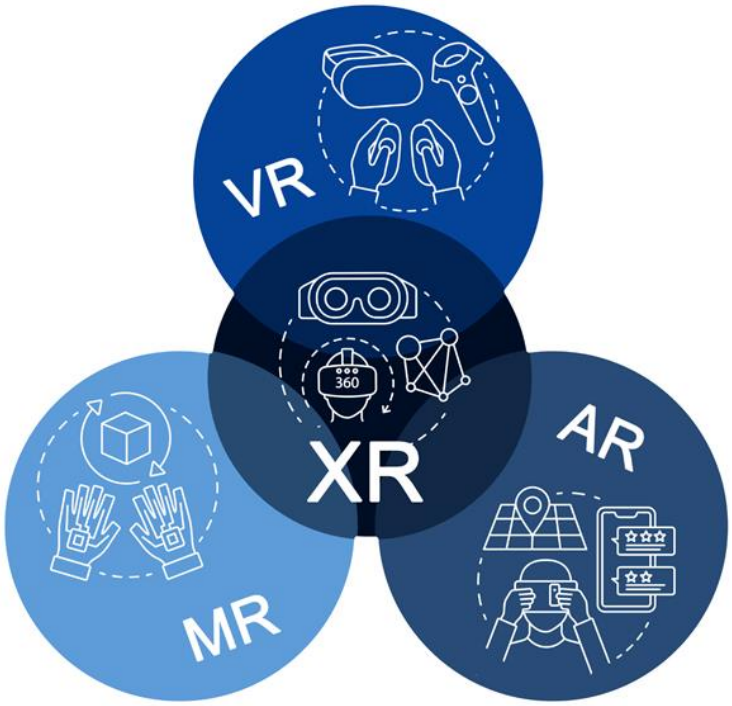
Η επαυξημένη πραγματικότητα αντίθετα με τον ορισμό της εικονικής πραγματικότητας ενσωματώνει την πληροφορία που παράγει ο υπολογιστής στον πραγματικό κόσμο του χρήστη. Τα τρία χαρακτηριστικά που καθορίζουν την επαυξημένη πραγματικότητα είναι ότι α) συνδυάζει το εικονικό και το πραγματικό, β) είναι διαδραστική σε πραγματικό χρόνο, γ) η πληροφορία χωροθετείται σε τρείς διαστάσεις.

Για την ανάπτυξη μια εφαρμογής με τεχνολογία επαυξημένης πραγματικότητας θα πρέπει να ακολουθούνται οι εξής κανόνες: Πρέπει να καθορίζεται η τρέχουσα κατάσταση τόσο του φυσικού όσο και του εικονικού κόσμου. Η πληροφορία πρέπει να εμφανίζεται με χωρική και χρονική συσχέτιση με τον πραγματικό κόσμο έτσι ώστε να επιτρέπει στον χρήστη να αντιληφθεί τα εικονικά στοιχεία ως μέρος του φυσικού κόσμου. Η επαυξημένη πραγματικότητα έχει καταχωρηθεί από πολλούς ως μια από τις πιο υποσχόμενες και ελπιδοφόρες τεχνολογίες των επόμενων ετών.

Μικτή Πραγματικότητα (Mixed Reality)

Ο ορισμός της Μικτής Πραγματικότητας (MR) δίνεται από την έννοια του «reality-virtuality» το οποίο είναι μια αναπαράσταση πραγματικού κόσμου κλάσεων βασισμένη σε υπολογιστικές τεχνικές. Θα μπορούσαμε να πούμε πως η μικτή πραγματικότητα είναι ένα layer ανάμεσα στο φυσικό και το εικονικό περιβάλλον και περιλαμβάνει την επαυξημένη πραγματικότητα. Ο κύριος στόχος της είναι η δημιουργία ενός μεγάλου χώρου με τη συγχώνευση πραγματικών και εικονικών περιβαλλόντων όπου πραγματικά και εικονικά αντικείμενα συνυπάρχουν και αλληλεπιδρούν σε πραγματικό χρόνο. Τέλος, η μικτή πραγματικότητα είναι μια κατηγορία προσομοιωτών που συνδυάζει εικονικά και πραγματικά αντικείμενα για να δημιουργήσει ένα υβρίδιο του εικονικού και του πραγματικού κόσμου.

Η μικτή πραγματικότητα χαρακτηρίζεται από τρείς οντότητες: την εμβύθιση, δηλαδή την ερμηνεία του περιβάλλοντος από τον χρήστη, την πληροφορία, δηλαδή τα εικονικά αντικείμενα που καταχωρούνται στον χώρο του χρήστη και την αλληλεπίδραση που γίνεται με φυσικούς τρόπους όπως ομιλία, ακοή, χειρονομίες κτλ. Οι τεχνολογίες Μικτής Πραγματικότητας προσφέρουν πολλά πλεονεκτήματα σε σχέση με τις προηγούμενες τεχνολογίες (VR, AR).



Εικόνα 6.Γραφική απεικόνιση του πεδίου της Εκτεταμένης Πραγματικότητας

2.2 ΙΣΤΟΡΙΚΗ ΑΝΑΔΡΟΜΗ

Αν και οι τεχνολογίες εικονικής και επαυξημένης τεχνολογίας φαντάζουν νέες και χωρίς μεγάλη ιστορία, στην πραγματικότητα η σύλληψη της ιδέας και οι πρώτες προσπάθειες για δημιουργία τέτοιων συστημάτων φτάνουν πάνω από μισό αιώνα πριν.

Το 1956 ο Morton Heilig δημιούργησε το Sensorama. Αυτό ήταν ένα ψυχαγωγικό σύστημα που προσέφερε μια εμπειρία σε ένα συνθετικό περιβάλλον μέσω πολλαπλών αισθήσεων. Περιλάμβανε, εικόνα, ήχο, οσμή, δόνηση και αέρα. Το εν λόγω περιβάλλον δεν προσέφερε καμία διάδραση με τον χρήστη όμως ήταν η πρώτη προσπάθεια αξιοποίησης πολλών αισθήσεων για την εμβύθιση του χρήστη σε μια πλούσια εμπειρία σε ένα εικονικό περιβάλλον.

Το 1965 ο βασικός οραματιστής της εικονικής πραγματικότητας, Ivan Sutherland περιέγραψε την ιδέα ενός μηχανισμού απεικόνισης όπου οι χρήστες θα ήταν σε θέση να διαδράσουνε με αντικείμενα ενός κόσμου που δεν θα ακολουθούσε υποχρεωτικά τους νόμους της φυσικής. Όπως ανέφερε και ο ίδιος, «θα λειτουργούσε σαν καθρέφτης σε μια μαθηματική χώρα των θαυμάτων». Έτσι, το 1968, η Evans and Sutherland Computer Corp. Κατασκεύασε το πρώτο κράνος εικονικής πραγματικότητας. Η συσκευή αυτή είχε δυο ιδιαίτερα χαρακτηριστικά που την έκανε μοναδική την εποχή εκείνη: την στερεοσκοπία και την ανίχνευση κίνησης. Η στερεοσκοπία είναι μια τεχνική η οποία δημιουργεί στον εγκέφαλο του χρήστη τεχνητή αίσθηση του βάθους μέσω της παροχής διαφορετικών, κατάλληλα διαμορφωμένων απόψεων περιβάλλοντος στα δυο μάτια. Η ανίχνευση κίνησης δίνει πληροφορίες στον σύστημα σχετικά με την θέση και τον προσανατολισμό του χρήστη.

Εικόνα 7. Το πρώτο κράνος εικονικής πραγματικότητας, 1968

Η ίδια εταιρεία, το 1973 ανέπτυξε τον πρώτο προσομοιωτή πτήσης βασισμένο σε υπολογιστή, ο οποίος αποτέλεσε την πρώτη εμπορικά επιτυχημένη εφαρμογή εικονικής πραγματικότητας. Το 1977 κατασκευάστηκε από το πανεπιστήμιο του Illinois και το πρώτο γάντι

εικονικής πραγματικότητας, το Sayre Glove. Το γάντι αυτό είχε την δυνατότητα να μετράει το λύγισμα των δακτύλων και να στέλνει τα δεδομένα σε υπολογιστή, ανακατασκευάζοντας κατά συνέπεια την συνολική διάταξη του χεριού. Τα γάντια δεδομένων αποτελούν μέχρι και σήμερα ένα αξιόπιστο μέσο για την φυσική αλληλεπίδραση του χρήστη με τον συνθετικό κόσμο.

Την δεκαετία του ’80 ο κλάδος γνώρισε ταχεία ανάπτυξη κυρίως λόγω της ραγδαίας διάδοσης των υπολογιστών. Το 1988 η NASA ανέπτυξε την συσκευή Convolvotron για την χωροθέτηση του ήχου ώστε ο ήχος να φαίνεται ότι προέρχεται από το σημείο που παράγεται στον εικονικό κόσμο. Ακόμη, το 1989 αποδόθηκε για πρώτη φορά ο όρος «εικονική πραγματικότητα» από την εταιρεία VPL σε όλη την κατηγορία αυτών των συστημάτων.

Το 1990 έκαναν την εμφάνισή τους δημόσιες εγκατάστασης ψυχαγωγικού περιεχομένου μέσω των οποίων το ευρύ κοινό ήταν πλέον ικανό να έρθει σε επαφή με την εικονική πραγματικότητα. Επίσης, το 1992 παρουσιάστηκε το CAVE, ένα σύστημα προβολής κατά το οποίο οι χρήστες εισέρχονται σε ένα δωμάτιο στο οποίο τα εικονικά δεδομένα προβάλλονται στον τοίχο. Επομένως οι χρήστες δεν χρειάζεται να φοράνε κάποια συσκευή όπως κράνος ή γάντια. Μπορούμε να θεωρήσουμε πως αυτό ήταν ο πρόγονος για τις εφαρμογές επαυξημένης πραγματικότητας αν και ο όρος δεν έχει εμφανιστεί ακόμα.

Στην σημερινή εποχή έχουμε αφθονία επιλογών όσων αφορά τα συστήματα αναπαραγωγής εικονικών περιβαλλόντων αλλά και σε λύσεις για την ανάπτυξη τέτοιων συστημάτων. Πρωτοπόροι στην αγορά σε συστήματα εικονικής και επαυξημένης πραγματικότητας είναι η Oculus, η SONY με το PSVR η VALVE με το Index, τα Hololense γυαλιά της Microsoft, το Magic Leap κ.α.

2.3 ΕΦΑΡΜΟΓΕΣ

Σίγουρα οι τεχνολογία της εικονικής και επαυξημένης πραγματικότητας ξεκίνησαν ως ένα μέσο ψυχαγωγίας και διασκέδασης και παράλληλα ένα ερευνητικό πεδίο ως προς την εξέλιξη των γραφικών και την αλληλεπίδραση του χρήστη με τον υπολογιστή. Με την πάροδο του χρόνο όμως οι ορίζοντες αξιοποίησης των τεχνολογιών αυτών άνοιξαν και πλέον η εικονική πραγματικότητα βρίσκει εφαρμογή σε πληθώρα πεδίων όπως η εκπαίδευση, η ιατρική, η επιστημονική έρευνα, οι στρατιωτικές εφαρμογές, η αρχιτεκτονική και η διακόσμηση, η πολεοδομία, η τέχνη και πολλά άλλα.

Μιας και τα βιντεοπαιχνίδια έχουν ιδιαίτερη απήχηση σε παιδιά και έφηβους, αρκετές προσπάθειες έχουν γίνει για την μεταφορά βιντεοπαιχνιδιών στις αίθουσες διδασκαλίας με στόχο ένα πιο διαδραστικό και ενδιαφέρον μάθημα. Πληθώρα εφαρμογών έχουν κατασκευαστεί με υποστήριξη σε όλα σχεδόν τα μαθήματα απο μαθηματικά και φυσική μέχρι μουσική και ιστορία. Πάνω σε αυτό έχει βρει εφαρμογή και η εικονική πραγματικότητα με προσομοιώσεις για την καλύτερη κατανόηση ή την διεξαγωγή εικονικών πειραμάτων σε περιπτώσεις που δεν δύναται να διεξαχθούν στον φυσικό χώρο. Επίσης ιδιαίτερα ενδιαφέρουσα είναι η έννοια της παιχνιδοποίησης στον χώρο της εκπαίδευσης.

Στον κλάδο της ιατρικής υπήρξε σημαντική και ταχεία εξέλιξη τα τελευταία χρόνια καθώς με τεχνολογίες εικονικής και επαυξημένης πραγματικότητας διευκολύνεται και βελτιστοποιείται η εκπαίδευση των γιατρών και των νοσοκόμων μέσα από εφαρμογές αναπαράστασης του ανθρώπινου σώματος. Ένας ακόμα κλάδος της ιατρικής στον οποίο μπορεί να έχει εφαρμογή η εικονική πραγματικότητα είναι η αντιμετώπιση φοβιών. Για παράδειγμα μέσα από χρήση ενός προσομοιωτή κάποιας φοβίας (πχ. Αγοραφοβία, κλειστοφοβία) ο ασθενής μαθαίνει να ελέγχει και να ξεπερνάει τον φόβο του.

Στις στρατιωτικές εφαρμογές ο στόχος χρήσης της εικονικής πραγματικότητας είναι η εκπαίδευση σε συνθήκες που είναι δύσκολο ή ασύμφορο να δοκιμαστούν στον κανονικό κόσμο όπως η εναέριες μάχες ή οι συνθήκες μάχης για τους στρατιώτες. Από την άλλη η χρήση της επαυξημένης πραγματικότητας στοχεύει στην παροχή πληροφοριών στους στρατιώτες σε πραγματικό χρόνο μέσα στο οπτικό πεδίο των χρηστών.

Και στην αρχιτεκτονική ακόμα έχουν δοθεί λύσεις με χρήση τεχνολογιών εικονική πραγματικότητας που μπορούν να απλοποιήσουν το έργο του αρχιτέκτονα και την επικοινωνία του με τον τελικό χρήστη. Με προγράμματα οπτικοποίησης ο χρήστης είναι σε θέση να έχει μια πρώτη άποψη του τελικού προϊόντος και ακόμα και να περιηγηθεί σε αυτό εξερευνώντας κάθε πτυχή του σχεδίου. Από την άλλη ο αρχιτέκτονας και οι μηχανικοί μπορούν πιο έυκολα να εντοπίσουν και να διορθώσουν πιθανές ατέλειες. Ακόμη, με εφαρμογές επαυξημένης πραγματικότητας ο χρήστης μπορεί εικονικά να διαμορφώσει τον χώρο του και εσωτερικά και εξωτερικά πριν καταλήξει στην τελική διακόσμηση.

Είναι σαφές πως η λίστα εφαρμογών εικονικής και επαυξημένης πραγματικότητας δεν τελειώνει εδώ και πως αυτά είναι μερικά μόνο παραδείγματα. Αν και οι προαναφερθείσες είναι δυο διαφορετικές τεχνολογίες, έχουν όμως κοινά χαρακτηριστικά και είναι αναμενόμενο να βρίσκουν εφαρμογή στους ίδιους τομείς, από διαφορετική ίσως σκοπιά ή διαφορετικά αποτελέσματα. Στην συνέχεια της παρούσης εργασίας θα ασχοληθούμε αποκλειστικά με την επαυξημένη πραγματικότητα. Ορισμένες από τις πιο δημοφιλείς εφαρμογές της επαυξημένης πραγματικότητας είναι τα έξυπνα βιβλία, οι έξυπνοι καθρέφτες, η διαδρασική πλοήγηση και τα παιχνίδια γαι φορητές συσκευές. Στο πλαίσιο αυτής της διπλωματικής θα αναπτυχθεί ένα παιχνίδι επαυξημένης πραγματικότητας για κινητή συσκευή.

ΚΕΦΑΛΑΙΟ 3: ΑΝΑΠΤΥΞΗ ΒΙΝΤΕΟΠΑΙΧΝΙΔΙΩΝ ΚΑΙ ΕΦΑΡΜΟΓΩΝ

3.1 ΕΙΣΑΓΩΓΗ

Μιας και αντικείμενο αυτής της διπλωματικής είναι ο σχεδιασμός και η ανάπτυξη ενός βιντεοπαιχνιδιού, θα ήταν σκόπιμο να αναφέρουμε και να μελετήσουμε τις πτυχές αυτής της βιομηχανίας και τους τρόπους με τους οποίους κάποιος προγραμματιστής ή μια ομάδα επιτυγχάνουν την παραγωγή ενός ηλεκτρονικού παιχνιδιού.

Η επιστήμη των ηλεκτρονικών παιχνιδιών έχει γνωρίσει και αυτή ραγδαία ανάπτυξη τα τελευταία χρόνια και έχει ξεφύγει κατά πολύ από την εικόνα την οποία είχε σχηματίσει ο κόσμος για τα ηλεκτρονικά παιχνίδια τις προηγούμενες δεκαετίες. Πλέον η βιομηχανία ηλεκτρονικών παιχνιδιών είναι αντάξια εκείνων του κινηματογράφου και της μουσικής και τις συναγωνίζεται επάξια σε δημοτικότητα αλλά και σε εισπράξεις. Τα βιντεοπαιχνίδια σήμερα δεν ανταποκρίνονται μόνο σε παιδιά αλλά έχουν τεράστια απήχηση στο ενήλικο κοινό και βρίσκονται διαθέσιμα σε κάθε μορφή και για κάθε είδους συσκευή, από υπολογιστή, κονσόλα, κινητά ή συσκευές εικονικής και επαυξημένης πραγματικότητας.

Για την ιστορία, προτού τα βιντεοπαιχνίδια αφομοιωθούν από την κοινωνία και γίνουν μέρος της κουλτούρας μας, έγιναν το επίκεντρο αντιπαράθεσης και ανησυχίας. Ειδικότερα την δεκαετία του 90’ όπου και γνώρισαν ταχεία πρόοδο και μαζική κατανάλωση ξέσπασε αναταραχή με το πρόσχημα πως η νέα αυτή τεχνολογία προκαλεί εθισμό στους χρήστες αποτραβώντας τους από την κοινωνική τους ζωή, κάνοντάς του απαθείς για τον εαυτό τους και τις ασχολίες τους και προκαλώντας βία και επιθετικότητα σε κοινωνικό επίπεδο. Το φαινόμενο αυτό δεν πρέπει να προκαλεί εντύπωση μιας και η ανησυχία και η δυσπιστία απέναντι σε ένα νέο τεχνολογικό επίτευγμα δεν είναι κάτι καινούριο. Οι ίδιες αντιδράσεις έχουν παρατηρηθεί και στο παρελθόν με κάθε νέα τεχνολογία που έρχεται να αλλάξει τον τρόπο που βλέπουμε τον κόσμο και αλληλοεπιδρούμε με αυτό. Τρανταχτά παραδείγματα αποτελεί κατακραυγή κατά των βιβλίων, του τηλεφώνου, της μουσικής και τέλος της τηλεόρασης.

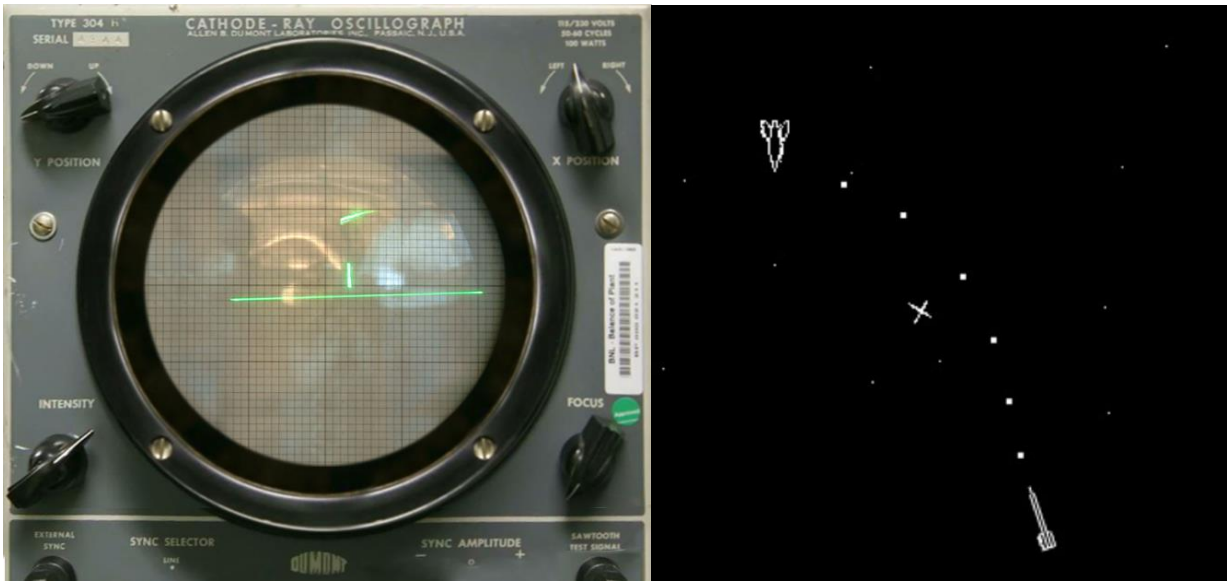
3.2 ΙΣΤΟΡΙΚΗ ΑΝΑΔΡΟΜΗ

Οι Πρόγονοι

Το 1951 ο Martin Bromley ιδρύει την SEGA και δημιουργεί την μοναδική μορφή ηλεκτρονικής ψυχαγωγίας της εποχής, τα φλιπεράκια, τα οποία διαθέτει σε κέντρα ψυχαγωγίας αμερικανικών στρατοπέδων στην Χαβάη. Αυτό αποτέλεσε τον πρόγονο για την δημιουργία των γνωστών καταστημάτων με ηλεκτρονικά παιχνίδια (arcades) που μεσουρανούσαν τις δεκαετίες του 70’ και του 80’.

Το 1958 γεννήθηκε το πρώτο ηλεκτρονικό παιχνίδι για κονσόλα από τον William Higinbotham, το Tennis for two. Ένα παιχνίδι τένις δυο παιχτών που έτρεχε στην οθόνη ενός

παλμογράφου. Έπειτα, το 1962 κυκλοφόρησε το SpaceWar! Από τον Steve Russel. Το SpaceWar! Ήταν το πρώτο παιχνίδι για υπολογιστή και έτρεχε στο μεγάλο υπολογιστικό σύστημα του MIT.



Εικόνα 8. Tennis for two, 1958 (αριστερά). SpaceWar!, 1962 (δεξιά)

Η πρώτη γενιά

Το 1972 ιδρύθηκε η εταιρεία Atari. Η εταιρεία όχι μόνο κυριάρχησε στη βιομηχανία βιντεοπαιχνιδιών για την επόμενη ***δεκαετία***, ανέπτυξε επίσης το Pong, το πρώτο παιχνίδι που έγινε παγκόσμια επιτυχία. Η αρχή παιχνιδιού του Pong μοιάζει πολύ με αυτή του προκατόχου του, του Tennis for Two. Σε αυτό το παιχνίδι, δύο παίκτες προσπαθούν να χτυπήσουν μια μπάλα στο μέγεθος ενός pixel, πάνω από μια γραμμή. Αν και η ιδέα για το παιχνίδι δεν ήταν καινούργια, η Atari ενσωμάτωσε τον υπολογιστή, μαζί με μια οθόνη προβολής, σε ένα κουτί με μια υποδοχή για κέρματα, εφευρίσκοντας έτσι τη μηχανή βιντεοπαιχνιδιών. Για πρώτη φορά, ένα βιντεοπαιχνίδι ήταν διαθέσιμο σε ένα ευρύτερο κοινό με πολύ λίγα χρήματα. Το παιχνίδι Space Invaders (1978) προανήγγειλε την αρχή της χρυσής εποχής των arcades, όπου οι έφηβοι της δεκαετίας του 1980 έπαιζαν το χαρτζιλίκι τους σε μηχανές βιντεοπαιχνιδιών. Πριν από αυτό, η τεχνολογία των υπολογιστών είχε σημειώσει σημαντική πρόοδο, με την ίδρυση της Apple το 1976 και την ανάπτυξη μικροεπεξεργαστών. Αργότερα η Atari κυκλοφόρησε την οικιακή κονσόλα Atari 2600 και περισσότεροι από 30 εκατομμύρια άνθρωποι αγόρασαν την κονσόλα παιχνιδιών, που κυκλοφόρησε το 1977, η οποία δεν περιοριζόταν σε ένα μόνο παιχνίδι, αλλά, χάρη στις εναλλάξιμες κασέτες, πρόσφερε θεωρητικά άπειρο αριθμό παιχνιδιών. Τα παιχνίδια είχαν πολύ απλά γραφικά και αφηγήσεις. Τα επίπεδα, με τα οποία το παιχνίδι γινόταν όλο και πιο δύσκολο, και τα συστήματα πόντων ήταν τυπικά χαρακτηριστικά. Οι παίκτες προσπάθησαν να πετύχουν το υψηλότερο σκορ και να γράψουν το όνομά τους στη συνεχή λίστα κατάταξης. Αυτό σήμαινε ότι τα απλά παιχνίδια παρέμειναν συναρπαστικά για μεγάλο χρονικό διάστημα

Η δεύτερη γενιά

Πολλά κλασικά παιχνίδια που εξακολουθούν να είναι δημοφιλή και σήμερα βγήκαν τη δεκαετία του 1980, το Pac-Man (1980), το Ultima (1980), το Mario Bros (1983), το Tetris (1984) και το SimCity (1989). Στις αρχές της δεκαετίας, όχι μόνο η αγορά ήταν γεμάτη με αμέτρητες νέες κονσόλες, αλλά άρχισαν να εμφανίζονται όλο και πιο φθηνοί και ισχυρότεροι οικιακοί υπολογιστές. Το 1983 η αγορά των κονσολών κατέρρευσε. Πολλές εταιρείες, μεταξύ των οποίων και η πρωτοπόρος στα παιχνίδια Atari, χρεοκόπησαν. Αλλά πριν από αυτό, η Atari κατάφερε να κυκλοφορήσει το παιχνίδι E.T. (1982), που εξακολουθεί να θεωρείται το χειρότερο βιντεοπαιχνίδι όλων των εποχών χάρη στα ακατέργαστα γραφικά και το περίπλοκο τρόπο παιχνιδιού του. Μέσα από τα συντρίμμια της βιομηχανίας αναδύθηκαν οι εταιρείες Commodore, με τον οικιακό υπολογιστή Commodore 64 (1982), και η Nintendo, με την κονσόλα Nintendo Entertainment System, ή για συντομία NES (1985). Με τις πιο εκλεπτυσμένες τεχνολογίες, τα παιχνίδια προωθήθηκαν σε νέες σφαίρες. Το παιχνίδι και τα γραφικά έγιναν πιο καινοτόμα. Τα παιχνίδια απέκτησαν τους δικούς τους χαρακτήρες και πιο περίπλοκες ιστορίες. Τα περισσότερα είδη όπως τα γνωρίζουμε σήμερα έχουν τις ρίζες τους σε αυτήν την περίοδο. Το Commodore 64 επέτρεψε ακόμη και σε χρήστες με υψηλά κίνητρα να προγραμματίσουν τα δικά τους παιχνίδια. Παιδιά και έφηβοι της δεκαετίας του 1980 περνούσαν ώρες μπροστά στους οικιακούς υπολογιστές ή τις κονσόλες τους, και με την κυκλοφορία του Game Boy στα τέλη της δεκαετίας, συνέχισαν ακόμη και μετά το σβήσιμο των φώτων, παίζοντας κάτω από το φως του φακού κάτω από τις κουβέρτες.

Η τρίτη γενιά

Την δεκαετία του 90’ τα βιντεοπαιχνίδια εισήλθαν σε μια νέα διάσταση, κυριολεκτικά, γιατί στο δεύτερο μισό της δεκαετίας τα γραφικά έγιναν τρισδιάστατα. Οι παίκτες μπορούσαν πλέον να κινούνται προς τρεις κατευθύνσεις αντί για δύο. Αυτοί οι κόσμοι του παιχνιδιού έμοιαζαν πιο ρεαλιστικοί και πρόσφεραν πιο σύνθετες δυνατότητες. Το 1994, η Sony κυκλοφόρησε το PlayStation το οποίο σε τεχνικούς και γραφικούς όρους, ήταν ένα κβαντικό άλμα σε σύγκριση με τις υπάρχουσες κονσόλες. Τα στούντιο σχεδιασμού παιχνιδιών ανέπτυξαν πιο καινοτόμες ιδέες, με την ιστορία να χρησιμεύει συχνά ως έμπνευση για το gameplay όπως στο Age of Empires (1997), στο Command & Conquer (1995), στο Tomb Raider (1996) και στο Monkey Island (1990). Όμως, στη δεκαετία του 1990 η βιομηχανία παιχνιδιών έχασε επίσης την αθωότητά της, επειδή, εκτός από τη δράση του παιχνιδιού, η βία έγινε επίσης όλο και περισσότερο θέμα. Ξεκινώντας με το Wolfenstein 3D (1992), ένας παίκτης χρησιμοποιεί ένα όπλο για να σκοτώσει τους αντιπάλους του από την οπτική γωνία του πρώτου προσώπου. Για πρώτη φορά, η κοινωνία αρχίζει να αναρωτιέται εάν η βία στα βιντεοπαιχνίδια οδηγεί σε βία στην πραγματική ζωή.

Η τέταρτη γενιά

Στις αρχές της νέας χιλιετίας, το Διαδίκτυο δεν ήταν ακόμα αρκετά ισχυρό ή σε αρκετά διαδεδομένη χρήση ώστε να επιτρέπει στους χρήστες να παίζουν ο ένας εναντίον του άλλου διαδικτυακά. Έτσι, οι παίκτες οργάνωναν LAN Parties. Οι παίκτες έφερναν τους δικούς τους

υπολογιστές και τους συνέδεαν σε ένα τοπικό δίκτυο για να παίξουν μεταξύ τους. Ιδιαίτερα δημοφιλές σε αυτά τα πάρτι ήταν το μερικές φορές αμφιλεγόμενο παιχνίδι Counter-Strike (2000), στο οποίο οι παίκτες αναλαμβάνουν τον ρόλο τρομοκρατών ή μελών μιας αντιτρομοκρατικής μονάδας και προσπαθούν να εξοντώσουν ο ένας τον άλλον. Στη συνέχεια, όμως, με την εντυπωσιακή αύξηση της χρήσης του Διαδικτύου, τα παιχνίδια έγιναν και διαδικτυακά. Για πρώτη φορά παιχνίδια, όπως το World of Warcraft (2004), παίζονται κυρίως στο διαδίκτυο. Με κάθε έτος της δεκαετίας, η τεχνολογία έκανε επίσης μεγάλα άλματα. Οι ισχυρές τεχνολογίες γραφικών έκαναν τους κόσμους των βιντεοπαιχνιδιών ακόμα πιο ρεαλιστικούς. Χάρη στην τεχνητή νοημοσύνη, οι προσομοιωμένοι αντίπαλοι δεν συμπεριφέρονται πλέον με τον ίδιο τρόπο σε κάθε περίπτωση, αντίθετα αντιδρούν αυτόνομα σε αυτό που συμβαίνει στο παιχνίδι. Παρουσιάστηκαν παιχνίδια ανοιχτού κόσμου, στα οποία οι παίκτες εξερευνούν μόνοι τους φανταστικούς κόσμους και μπορούν να καθορίσουν ελεύθερα την πορεία του παιχνιδιού. Η δεκαετία του 2000 έδωσε πολλά best-sellers, μεταξύ των οποίων τα κυριότερα ήταν τα The Sims (2000), Grand Theft Auto: San Andreas (2004), Halo (2001), World of Warcraft (2004), God of War (2005) και πολλά άλλα.

Η πέμπτη γενιά

Στη δεκαετία που μόλις τελείωσε, τα βιντεοπαιχνίδια έχουν γίνει μια επιχείρηση δισεκατομμυρίων δολαρίων και τα κέρδη είναι υψηλότερα από αυτά της βιομηχανίας ταινιών και μουσικής. Αμέτρητα ανεξάρτητα στούντιο παιχνιδιών έχουν αναπτύξει παιχνίδια για κάθε τύπο πλατφόρμας: υπολογιστές, κονσόλες, tablet και κινητά τηλέφωνα. Ως αποτέλεσμα, ακόμα περισσότεροι άνθρωποι παίζουν βιντεοπαιχνίδια τακτικά. Και οι εποχές που έπαιζαν αυτά τα παιχνίδια μόνο παιδιά και έφηβοι πέρασαν. Οι ηλικιωμένοι ανακαλύπτουν επίσης παιχνίδια δεξιοτεχνίας και γρίφων στα κινητά τους τηλέφωνα. Ένα βιντεοπαιχνίδι υπάρχει πάντα άμεσα διαθέσιμο, πράγμα που αυξάνει τα ποσοστά εθισμού στο gaming κυρίως στα κινητά τηλέφωνα. Για την πλειοψηφία όσων παίζουν, ωστόσο, το gaming είναι ακίνδυνη ψυχαγωγία. Παιχνίδια όπως το Red Dead Redemption 2 (2019) που, χάρη σε εξελιγμένους διαλόγους και συγκινητικές ιστορίες, όχι μόνο προσφέρουν ώρες διασκέδασης στο gaming αλλά και ακολουθούν διαφορετική πορεία ανάλογα με τις αποφάσεις του παίκτη και έτσι μπορούν να παιχτούν ξανά και ξανά σε υπολογιστές, PlayStation και Xbox.

3.3 ΣΧΕΔΙΑΣΜΟΣ ΒΙΝΤΕΟΠΑΙΧΝΙΔΙΩΝ

Ένα βιντεοπαιχνίδι αποτελεί μια μορφή τέχνης, μια διαδικασία με κανόνες στην οποία οι εμπλεκόμενοι (οι παίκτες) συνεργάζονται με σκοπό να εκπληρώσουν έναν στόχο και να φτάσουν στο επιθυμητό αποτέλεσμα. Όπως κάθε παιχνίδι, έτσι και τα ηλεκτρονικά παιχνίδια αποτελούνται από συγκεκριμένα δομικά στοιχεία τα οποία ο δημιουργός πρέπει να λάβει υπ' όψη του κατά τον σχεδιασμό του παιχνιδιού.

Αναφέροντας τα δομικά στοιχεία που βρίσκονται στην καρδιά ενός βιντεοπαιχνιδιού αλλά είναι και καίριας σημασίας στην φάση της παραγωγής του, ξεκινάμε ίσως με το πιο προφανές, την ***συμμετοχή των παικτών***. Χωρίς κάποιον παίκτη ή παίκτες να αλληλοεπιδράσουν με τον εικονικό κόσμο ένα παιχνίδι δεν θα είχε καμία ουσία. Κάθε παιχνίδι έχει κάποιον ***σκοπό***. Οι παίκτες που συμμετέχουν στο παιχνίδι επιδιώκουν να επιτύχουν κάποιο απώτερο στόχο, αυτός μπορεί να είναι η συλλογή των περισσότερων πόντων. Έπειτα, το παιχνίδι παίζεται με βάση κάποιες ***διαδικασίες***, οδηγίες δηλαδή στην ουσία για όλες τις κινήσεις που μπορεί να εκτελέσει ο παίκτης στην προσπάθειά του να φτάσει στον στόχο του. Οι διαδικασίες αυτές αναφέρονται ως gameplay του παιχνιδιού. Οι επιτρεπτές διαδικασίες που μπορεί να κάνει ο παίκτης μέσα στο παιχνίδι καθορίζονται από τους ***κανόνες*** του παιχνιδιού. Οι κανόνες καθορίζουν ποιες κινήσεις γίνονται επιτρεπτές στα όρια του παιχνιδιού και ποιες όχι. Ο παίκτης δεν μπορεί παρά να σεβαστεί τους κανόνες αυτούς. Εν συνεχεία, οι ***πόροι*** ενός παιχνιδιού είναι οποιοδήποτε αντικείμενο μπορεί να χρησιμοποιήσει ο παίκτης κατά την διάρκεια του gameplay και το οποίο θα τον βοηθήσει να πετύχει τον στόχο του. Επίσης, στην διαδρομή του προς τον τελικό σκοπό του παιχνιδιού ο παίκτης θα συναντήσει ***εμπόδια*** τα οποία καλείται να προσπεράσει, είτε λύνοντας κάποιο γρίφο, είτε πολεμώντας, είτε αποφεύγοντάς τα. Όλα τα παραπάνω στοιχεία λαμβάνουν χώρα μέσα σε έναν προκαθορισμένο φανταστικό χώρο ο οποίος αποτελεί τα ***όρια*** του παιχνιδιού. Το πόσο καλά σχεδιασμένος είναι αυτός ο χώρος συμβάλει καταλυτικά στον βαθμό εμβύθισης του παίκτη στον εικονικό κόσμο και στην βελτίωση της συνολικής εμπειρίας. Τέλος, κάθε παιχνίδι πρέπει να καταλήγει σε μια τελική κατάσταση, με ένα ***αποτέλεσμα*** για τους παίκτες. Το αποτέλεσμα δεν είναι πάντα προκαθορισμένο και μπορεί σε πολλές περιπτώσεις ο παίκτης βάση των αποφάσεών του να αλλάξει την έκβασή του.

Καταλήγουμε λοιπόν στο συμπέρασμα ότι ένα βιντεοπαιχνίδι δεν είναι μια αυθαίρετη δημιουργία αλλά ένα σύνολο καλά ορισμένων κανόνων και στοιχείων που συνεργάζονται αρμονικά μεταξύ τους και με τον χρήστη. Εξίσου σημαντικό είναι πως τα παραπάνω δομικά στοιχεία λειτουργούν και ως καθοδήγηση για τον σχεδιαστή κατά την ανάπτυξη ενός νέου παιχνιδιού. Παράλληλα όμως ένα βιντεοπαιχνίδι είναι τέχνη και έκφραση και απαιτεί σωστή τεχνική από την μεριά του προγραμματιστή. Έτσι, ο προγραμματιστής πρέπει να έχει διαίσθηση και κατανόηση του τι ζητάει η αγορά και ποιες είναι οι απαιτήσεις του καταναλωτικού κοινού ώστε το παιχνίδι να είναι ελκυστικό στον χρήστη και ταυτόχρονα εμπορικά επιτυχημένο. Η δημιουργία ηλεκτρονικών παιχνιδιών απαιτεί ένα συνδυασμό φαντασίας, γνώσης, τεχνικής κατάρτισης, αισθητικής, αντίληψης και άλλων δεξιοτήτων.

3.4 ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΒΙΝΤΕΟΠΑΙΧΝΙΔΙΩΝ

Η αρχιτεκτονική ενός παιχνιδιού μπορεί να είναι από αρκετά απλή έως εξαιρετικά πολύπλοκη. Σε παιχνίδια μικρότερης έκτασης ο αυστηρός σχεδιασμός δεν είναι πάντα απαραίτητος, όμως όσο μεγαλώνει η παραγωγή η σωστή οργάνωση των λειτουργιών και των αντικειμένων του παιχνιδιού είναι κρίσιμης σημασίας. Η αρχιτεκτονική ενός παιχνιδιού καθορίζεται από τον σχεδιασμό, την αναπαράσταση και την αλληλεπίδραση των μεταξύ τους λειτουργιών, όπως αυτές παρουσιάζονται παρακάτω.

Από τεχνικής άποψης πρέπει να καθοριστούν εξαρχής οι απαιτήσεις στα γραφικά και την τελική απεικόνιση στο σύστημα προβολής. Τα γραφικά μπορεί να είναι τρισδιάστατα (3D Models) ή δισδιάστατα (Sprites). Για την λεπτομερέστερη απεικόνιση των μοντέλων στην δισδιάστατη οθόνη συχνά χρησιμοποιούνται τεχνικές αύξησης της λεπτομέρειας (Textures), ειδικά εφέ, σκιές και ανακλάσεις. Για την δημιουργία αίσθησης ότι η κίνηση της κάμερα και των αντικειμένων γίνεται ομαλά απαραίτητος είναι ο μεγάλος ρυθμός ανανέωσης της εικόνας. Στα σύγχρονα παιχνίδια ένας αποδεκτός ρυθμός ανανέωσης είναι τα 60 καρέ το δευτερόλεπτο ενώ πολλές φορές αναζητάται από τους χρήστες και μεγαλύτερη απόδοση.

Η κίνηση σε ένα βιντεοπαιχνίδι είναι αυτό που το κάνει να φαίνεται ζωντανό στα μάτια του παίκτη. Οι περισσότερες κινήσεις στο παιχνίδι ελέγχονται από το χρήστη ή από τον ίδιο τον υπολογιστή ενώ μπορεί να υπάρχει κίνηση στο παρασκήνιο όπως αλλαγές στο περιβάλλον. Η κίνηση μπορεί να είναι απλοϊκή σαν να μια απλή μεταφορά ή περιστροφή στον χώρο ή μπορεί να είναι σύνθετη για να περιγράψει πολύπλοκους μετασχηματισμούς όπως η κίνηση ενός ανθρώπινου σώματος. Για να επιτύχουμε σύνθετη ρεαλιστική κίνηση πρέπει να εξομοιώσουμε με ειδικές τεχνικές φυσικές εργασίες όπως η επίδραση του ανέμου στα μαλλιά του χαρακτήρα μας ή τα σωματίδια του καπνού μιας φωτιάς κ.α.

Ο ήχος έρχεται για να συμπληρώσει την εικόνα ως μέσο αντίληψης του περιβάλλοντος και είναι εξίσου σημαντικός στην συνολική εμπειρία. Ο ήχος μπορεί να είναι μουσική υπόκρουση, ομιλία, ηχητικά εφέ ενώ μπορεί να βοηθήσει σημαντικά τον παίκτη στην πορεία του gameplay.

Ένα ακόμη συστατικό της αρχιτεκτονικής ενός βιντεοπαιχνιδιών είναι η διεπαφή με τον χρήστη. Μέσα από το User Interface ο παίκτης είναι είτε σε θέση να λάβει πληροφορίες και μηνύματα σχετικά με το status του χαρακτήρα του ή την έκβαση της ιστορίας και παράλληλων γεγονότων αλλά ακόμα και να αλλάξει ρυθμίσεις, να πειραματιστεί με στοιχεία του εικονικού κόσμου, να τροποποιήσει την εμφάνιση και την συμπεριφορά του χαρακτήρα κ.α. Επίσης, ιδιαίτερα δημοφιλής έχει γίνει τα τελευταία χρόνια η αξιοποίηση του διαδικτύου μέσα από το παιχνίδι αφού πλέον η πλειοψηφία των παιχνιδιών παρέχουν την επιλογή στους παίκτες να παίξουν και συνεργατικά, είτε τοπικά από την ίδια συσκευή είτε απομακρυσμένα ο καθένας από τον προσωπικό του υπολογιστή.

Αυτές είναι κάποιες μόνο από τις οντότητες που απαρτίζουν την λειτουργία ενός παιχνιδιού. Υπάρχουν πολλές ακόμα μονάδες εργασιών που συντελούν στην ολοκληρωμένη ανάπτυξη του λογισμικού όπως η τεχνητή νοημοσύνη, η ανίχνευση συγκρούσεων, η διαχείριση δεδομένων και ο χειρισμός γεγονότων. Η αρχιτεκτονική του παιχνιδιού ορίζει πως θα οριστούν αυτές οι οντότητες, ποιες λειτουργίες θα πραγματοποιούνται και πως θα επικοινωνούν μεταξύ τους.

3.5 ΕΡΓΑΛΕΙΑ ΑΝΑΠΤΥΞΗΣ

Ένα βιντεοπαιχνίδι εκτός από ψυχαγωγικό μέσο είναι και ένα τεχνολογικό προϊόν. Για τον λόγο αυτό απαιτεί εξιδεικευμένες γνώσεις από την μεριά του προγραμματιστή αλλά και χρήση εξιδεικευμένων εργαλείων που κάνουν την υλοποίησή του δυνατή. Η διαδικασία ανάπτυξης του λογισμικού ενός παιχνιδιού συνήθως περιλαμβάνει τρείς κατηγορίες λογισμικού: τον κώδικα του παιχνιδιού, την μηχανή γραφικών και τα βοηθητικά εργαλεία.

Το ισχυρότερο εργαλείο ενός προγραμματιστή είναι η γλώσσα προγραμματισμού που θα χρησιμοποιήσει για να γράψει τον κώδικα του παιχνιδιού. Ο κώδικας είναι το κομμάτι στο οποίο υλοποιούνται όλες οι λειτουργίες του gameplay, η λογική και οι μηχανισμοί του παιχνιδιού. Η πλειοψηφία μεγάλων παραγωγών αναπτύσσονται σήμερα σε C++, μια αντικειμενοστραφή γλώσσα προγραμματισμού σχεδιασμένη για μεγάλη απόδοση και ταχύτητα. Επίσης πάρα πολλά παιχνίδια που τρέχουν στο διαδίκτυο ή σε κινητά έχουν κατασκευαστεί σε Java, επίσης αντικειμενοστραφή γλώσσα προγραμματισμού. Ακόμη, τα τελευταία χρόνια έχει ανέβει σε δημοτικότητα η γλώσσα προγραμματισμού C#, η οποία είναι μια παραλλαγή της C++ που προσφέρει μεγαλύτερη ευελιξία και πληθώρα εργαλείων ειδικά σχεδιασμένων για να διευκολύνουν την διαδικασία παραγωγής των βιντεοπαιχνιδιών.

Η μηχανή παιχνιδιού (ή μηχανή γραφικών) είναι το πρόγραμμα μέσα από το οποίο γίνεται ο σχεδιασμός του παιχνιδιού, η εναλλαγή των σκηνών, η επικοινωνία με τον κώδικα και η παραγωγή του τελικού προϊόντος. Βασικές λειτουργίες της μηχανής γραφικών είναι η απεικόνιση των γραφικών, η αναπαραγωγή ήχου, η κίνηση, η επικοινωνία με τον χρήστη και το δίκτυο κ.α. Η

πιο δημοφιλής και ευρέως χρησιμοποιούμενη μηχανή γραφικών είναι η Unreal Engine. Μια πιο εύκολη στην χρήση και πιο ευέλικτη επιλογή, καθώς υποστηρίζει ανάπτυξη σε κάθε διαθέσιμη πλατφόρμα αλλά και υλικό για όλους τους τομείς της βιομηχανίας, είναι η μηχανή γραφικών Unity. Ιδιαίτερα αγαπητή στα ανεξάρτητα studios και προγραμματιστές. Αξιοσημείωτες αναφορές είναι επίσης η CryEngine, η Godot και το GameMaker.

Οποιοδήποτε λογισμικό μπορεί να χρησιμοποιηθεί για την περαιτέρω βοήθεια ή διευκόλυνση της ομάδας ανάπτυξης στην δημιουργία του βιντεοπαιχνιδιού συγκαταλέγεται στα βοηθητικά εργαλεία. Τέτοια μπορεί να είναι σχεδιαστικά προγράμματα, προγράμματα μοντελοποίησης, editors, επεκτάσεις και βιβλιοθήκες. Στα βιντεοπαιχνίδια η επικοινωνία με τον επεξεργαστή γραφικών γίνεται μέσω μια εξειδικευμένης βιβλιοθήκης εντολών για γραφικά. Οι κυριότερες βιβλιοθήκες γραφικών στην αγορά είναι η OpenGL και η DirectX. Κατά την δημιουργία ενός παιχνιδιού απαιτούνται ειδικές βοηθητικές βιβλιοθήκες ανάπτυξης λογισμικού που δίνουν ένα σύνολο από εργαλεία τα οποία παρέχουν στην εφαρμογή πρόσβαση σε κάποια τεχνολογία. Στην συνέχεια θα ασχοληθούμε με τεχνολογίες επαυξημένης πραγματικότητας (AR). Οι πιο γνωστές βιβλιοθήκες ανάπτυξης λογισμικού επαυξημένης πραγματικότητας είναι η Vuforia της Qualcomm, το ARCore για Android και το ARKit για iOS της Google, το Wikitude, το EasyAR και άλλα.

ΜΕΡΟΣ 2: ΥΛΟΠΟΙΗΣΗ ΕΦΑΡΜΟΓΗΣ ΕΠΑΥΞΗΜΕΝΗΣ ΠΡΑΓΜΑΤΙΚΟΤΗΤΑΣ

ΚΕΦΑΛΑΙΟ 1: ΟΡΓΑΝΩΣΗ ΚΑΙ ΤΕΧΝΟΛΟΓΙΕΣ

1.1 Η ΜΗΧΑΝΗ ΓΡΑΦΙΚΩΝ UNITY

Η Unity είναι μια μηχανή γραφικών σχεδιασμένη να δουλεύει σε όλες σχεδόν τις υπολογιστικές πλατφόρμες. Κυκλοφόρησε πρώτη φορά τον Ιούνιο του 2005 από την Unity Technologies στο Διεθνές Συνέδριο Προγραμματιστών της Apple ως αποκλειστική μηχανή παιχνιδιών για MacOS συστήματα. Έκτοτε η δημοτικότητά της ανέβηκε και η χρήση της επεκτάθηκε σε ποικιλία άλλων συστημάτων όπως οι προσωπικοί υπολογιστές, οι κινητές συσκευές, οι κονσόλες παιχνιδιών και τα συστήματα εικονικής πραγματικότητας.

Πέρα από την παραγωγή βιντεοπαιχνιδιών και λογισμικών προσομοίωσης, η Unity έχει εδραιωθεί ως εργαλείο και σε βιομηχανίες όπως ο κινηματογράφος, η αυτοκινητοβιομηχανία, η αρχιτεκτονική, η μηχανολογία, η ρομποτική κ.α. Την τελευταία δεκαετία η Unity έγινε ιδιαίτερα δημοφιλής στις ομάδες ανάπτυξης λογισμικού με την πλατφόρμα να υποστηρίζει αυτή την στιγμή ανάπτυξη λογισμικού για πλήθος συστημάτων. Ανάμεσά τους τα Windows, macOS, Linux, Android, iOS, WebGL, PlayStation, XBOX, Nintendo Switch, Oculus VR, Hololense, SteamVR, ARCore, και η λίστα συνεχίζεται.

Παρακάτω παρουσιάζεται μια επισκόπηση του γραφικού περιβάλλοντος (editor) της Unity game engine καθώς και όλα τα εργαλεία και παράθυρα του με τις αντίστοιχες λειτουργίες.

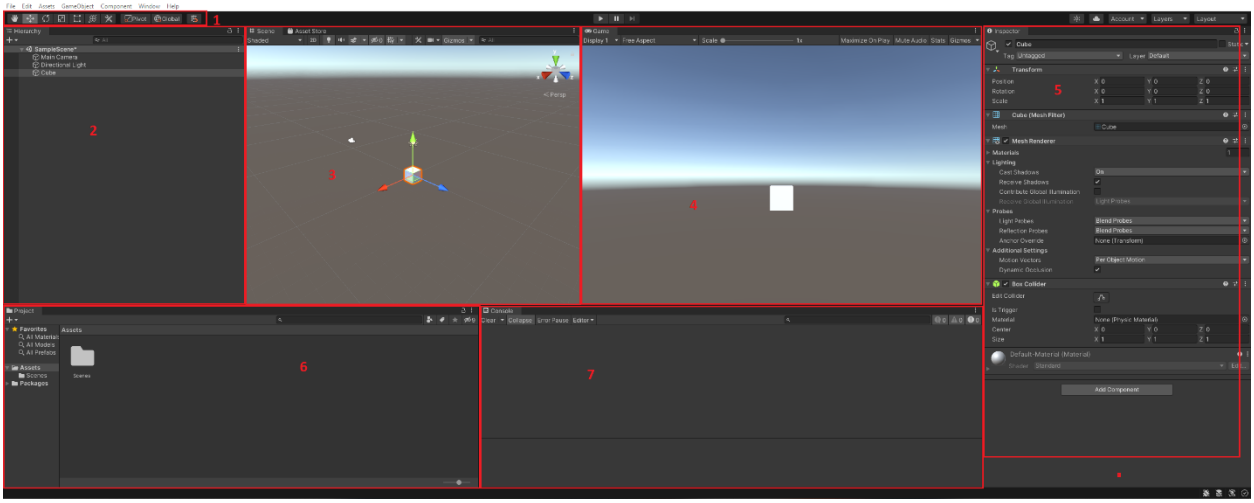


Figure 1. Unity Editor

- **(1)** Η ***γραμμή εργαλείων*** παρέχει πρόσβαση στις πιο βασικές λειτουργίες. Στα αριστερά βρίσκονται τα βασικά εργαλεία για τον χειρισμό της προβολής σκηνής και των GameObjects μέσα σε αυτήν. Στο κέντρο βρίσκονται τα χειριστήρια αναπαραγωγής, και παύσης. Τα κουμπιά στα δεξιά δίνουν πρόσβαση στο Unity λογαριασμό του χρήστη και σε ορισμένες υπηρεσίες του Unity Cloud.
- **(2)** Το ***παράθυρο ιεραρχίας*** προσφέρει μια ιεραρχική αναπαράσταση κειμένου για κάθε GameObject στη σκηνή. Κάθε στοιχείο στη σκηνή καταχωρείται στην ιεραρχία, επομένως τα δύο παράθυρα συνδέονται εγγενώς. Η ιεραρχία αποκαλύπτει τη δομή του τρόπου με τον οποίο τα GameObjects συνδέονται μεταξύ τους.
- **(3)** Η ***προβολή σκηνής*** σάς επιτρέπει την οπτική πλοήγηση πάνω στα αντικείμενα του παιχνιδιού και την επεξεργασία της σκηνής. Η προβολή σκηνής μπορεί να γίνεται είτε σε 3D ή σε 2D, ανάλογα με την φύση του έργου.
- **(4)** ***Η προβολή Παιχνιδιού*** προσομοιώνει πώς θα μοιάζει το τελικό παιχνίδι που αποδόθηκε μέσω της κάμερας της σκηνής. Με το κουμπί Αναπαραγωγή, ξεκινά η προσομοίωση.
- **(5)** Το ***παράθυρο επιθεώρησης*** προβάλει όλες τις ιδιότητες του τρέχοντος επιλεγμένου αντικειμένου. Επειδή διαφορετικοί τύποι GameObjects έχουν διαφορετικές ιδιότητες, η διάταξη και τα περιεχόμενα του παραθύρου Inspector αλλάζουν κάθε φορά που επιλέγετε ένα διαφορετικό GameObject.
- **(6)** Το ***παράθυρο Project*** εμφανίζει μια βιβλιοθήκη με όλα τα στοιχεία που είναι διαθέσιμα για χρήση στο project. Όταν εισάγονται νέα στοιχεία στο project (τα οποία ονομάζουμε Assets), εμφανίζονται εδώ.
- **(7)** Το ***παράθυρο κονσόλας*** παρέχει ειδοποιήσεις για διάφορες διαδικασίες που εκτελούνται από την Unity.

1.2 ΤΟ ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΟ ΠΕΡΙΒΑΛΛΟΝ MICROSOFT VISUAL STUDIO

Η μηχανή γραφικών Unity έρχεται πλέον με προ-εγκατεστημένο editor για την συγγραφή του κώδικα, το Visual Studio της Microsoft. Το Visual Studio είναι ένα IDE, ένα ολοκληρωμένο, ενσωματωμένο περιβάλλον ανάπτυξης δηλαδή που περιλαμβάνει πανίσχυρα εργαλεία για τον προγραμματιστή όπως το InteliSense για την αυτόματη συμπλήρωση, τα εξελιγμένα εργαλεία για debugging του κώδικα και το Code Navigation.

Η γλώσσα προγραμματισμού που χρησιμοποιεί η Unity είναι η C#, μια αντικειμενοστραφής γλώσσα προγραμματισμού βασισμένη στο .NET πρωτόκολλο. Κάθε project που αναπτύσσεται στην Unity είναι άμεσα συνδεδεμένο με το Visual Studio. Έτσι, όταν πατήσουμε διπλό κλικ σε κάποιο C# Script από τον Unity Editor θα ανοίξει αυτόματα το αντίστοιχο project στο Visual Studio με τον κώδικα της εφαρμογής. Αν, για παράδειγμα, προσπαθήσουμε να ανοίξουμε ένα script που ανήκει σε διαφορετικό project, τότε θα ανοίξει καινούριο παράθυρο στο Visual Studio το οποίο θα επικοινωνεί μόνο με το συγκεκριμένο project.

1.3 ΤΟ Google ARFoundation ΚΑΙ ARCore

Το ARCore είναι η πλατφόρμα της Google για τη δημιουργία εφαρμογών και εμπειριών επαυξημένης πραγματικότητας. Χρησιμοποιώντας διαφορετικά APIs, το ARCore επιτρέπει στην συσκευή να αντιλαμβάνεται το περιβάλλον γύρω της, να κατανοεί τον κόσμο και να αλληλοεπιδρά με πληροφορίες.

Το ARCore παρέχει μια ποικιλία εργαλείων για την κατανόηση αντικειμένων στον πραγματικό κόσμο. Αυτά τα εργαλεία περιλαμβάνουν την περιβαλλοντική κατανόηση (environmental understanding), η οποία επιτρέπει στις συσκευές να ανιχνεύουν οριζόντιες και κάθετες επιφάνειες και επίπεδα. Περιλαμβάνουν επίσης παρακολούθηση κίνησης (motion tracking), η οποία επιτρέπει στα τηλέφωνα να κατανοούν και να παρακολουθούν τις θέσεις τους σε σχέση με τον κόσμο.

Το ARCore χρησιμοποιεί τρεις βασικές δυνατότητες για την ενσωμάτωση εικονικού περιεχομένου με τον πραγματικό κόσμο και την προβολή στην οθόνη της συσκευής

- Η παρακολούθηση κίνησης επιτρέπει στο τηλέφωνο να κατανοεί και να παρακολουθεί τη θέση του σε σχέση με τον κόσμο.
- Η περιβαλλοντική κατανόηση επιτρέπει στο τηλέφωνο να ανιχνεύει το μέγεθος και τη θέση όλων των τύπων επιφανειών: οριζόντιες, κάθετες και υπό γωνία επιφάνειες όπως το έδαφος, ένα τραπεζάκι του καφέ ή οι τοίχοι.
- Η εκτίμηση φωτός (light estimation) επιτρέπει στο τηλέφωνο να εκτιμήσει τις τρέχουσες συνθήκες φωτισμού του περιβάλλοντος.

Το ARFoundation είναι ένα framework με εφαρμογή ανάμεσα σε διάφορες πλατφόρμες ανάπτυξης το οποίο επιτρέπει την δημιουργία μιας εφαρμογής επαυξημένης πραγματικότητας από οποιοδήποτε υπολογιστικό σύστημα είτε για Android είτε για iOS λειτουργικά ταυτόχρονα και με ευκολία.

1.4 ΤΟ ANDROID STUDIO

Το λογισμικό που αναπτύσσουμε είναι μια εφαρμογή επαυξημένης πραγματικότητας για λειτουργικό Android. Η Unity δεν έχει από μόνη της την δυνατότητα ανάπτυξης λογισμικού android, για τον λόγο αυτό συνεργάζεται με το Android Studio ώστε να αντλήσει τους πόρους που είναι απαραίτητοι για το «χτίσιμο» της εφαρμογής. Αν και η ανάπτυξη της εφαρμογής γίνεται εξ' ολοκλήρου μέσα από την Unity και το Android Studio έχει καμία ανάμειξη σε αυτήν, το τελευταίο πρέπει να είναι εγκατεστημένο στο σύστημα και συνδεδεμένο με την Unity.

Επιπλέον, κατά την εγκατάσταση της έκδοσης Unity που θα χρησιμοποιηθεί στο project πρέπει μαζί να εγκατασταθεί το εργαλείο Android Build Support. Επίσης απαραίτητα για την δημιουργία και εκτέλεση του προγράμματος είναι τα Android Software Development Kit (SDK) και Native Development Kit (NDK). Από προεπιλογή η Unity συμπεριλαμβάνει το Java Development Kit βασισμένο στο OpenJDK. Τα εργαλεία αυτά πρέπει να είναι σωστά καθορισμένα στο περιβάλλον της Unity.

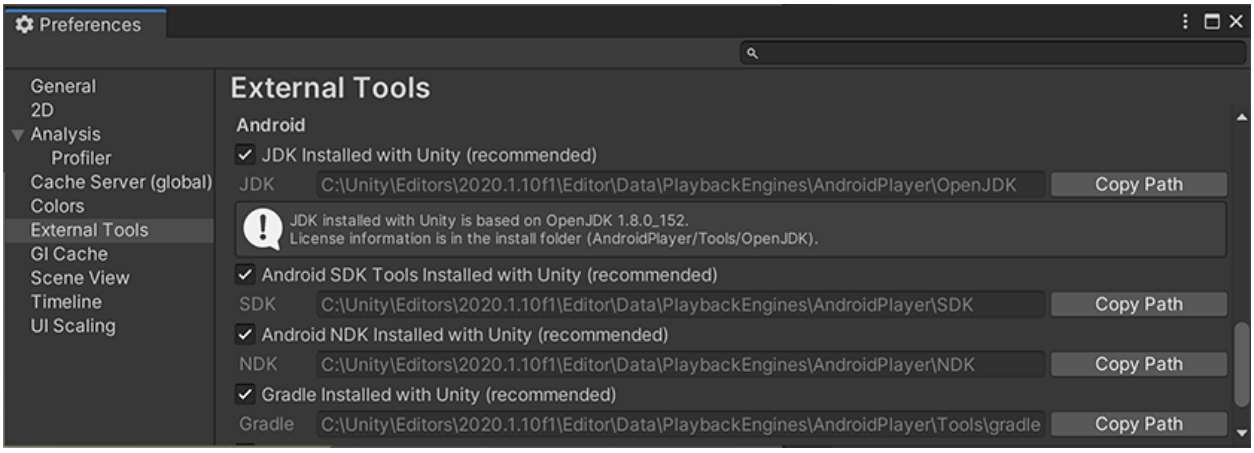


Figure 2. Ρυθμίσεις για Android

1.5 ΤΟ PHOTON ENGINE

Η εφαρμογή που θα αναπτύξουμε παρακάτω εκτός από την τεχνολογία επαυξημένης πραγματικότητας θα έχει την μορφή παιχνιδιού πολλών χρηστών (multiplayer game). Για αυτό τον λόγο χρειαζόμαστε μια λύση στο επίπεδο του δικτύου και της επικοινωνίας. Το ρόλο αυτού του διαμεσολαβητή παίζει το Photon Engine, ένα πρόσθετο εργαλείο (plug-in) υπεύθυνο για την υλοποίηση της δικτυακής επικοινωνίας όλων των χρηστών και την διαχείριση της πληροφορίας που διαμοιράζεται.

Το Photon είναι ένα αρκετά διαδεδομένο εργαλείο στην κατηγορία του και χρησιμοποιείται από πολλούς προγραμματιστές παιχνιδιών ως μια ανεξάρτητη λύση. Το σύνολο των εργαλείων και των υπηρεσιών που διαθέτει το καθιστούν αρκετά ικανό στην διαχείριση επικοινωνίας ανάμεσα σε client και server. Μερικές από τις υπηρεσίες που προσφέρει είναι η δημιουργία τοπικού server από την μεριά ενός χρήστη, η σύνδεση και αποσύνδεση χρηστών από τον κύριο server του παιχνιδιού, η δημιουργία παιχνιδιού πολλών παικτών σε ένα κοινό δωμάτιο στο οποίο οι χρήστες αλληλοεπιδρούν, η εύρεση από άλλους παίκτες ήδη ενεργών παιχνιδιών για να συμμετάσχουν, η διαχείριση των γεγονότων κατά την διάρκεια της συνεδρίας κ.α.

ΚΕΦΑΛΑΙΟ 2: ΣΧΕΔΙΑΣΜΟΣ

2.1 ΣΥΝΟΛΙΚΗ ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΠΑΙΧΝΙΔΙΟΥ

Η δομή της εφαρμογής όπως έχει αναφερθεί χωρίζεται σε δύο στρώματα. Αυτό της επαυξημένης πραγματικότητας (AR) και αυτό των πολλών ταυτόχρονων χρηστών (Multiplayer). Το περιβάλλον ανάπτυξης που θα χρησιμοποιηθεί είναι η Unity Engine, η τεχνολογία επαυξημένης πραγματικότητας είναι το ARCore και το ARFoundation Framework, την επικοινωνία μεταξύ των παικτών θα διαχειρίζεται το Photon Engine ενώ όλες οι παραπάνω λειτουργίες του παιχνιδιού θα είναι γραμμένες σε γλώσσα C#.

Αρχικά στο επίπεδο της επικοινωνίας χρησιμοποιείται το UDP πρωτόκολλο λόγω της υψηλής απόδοσης και της ταχύτητας μετάδοσης των πακέτων που προσφέρει. Το UDP πρωτόκολλο είναι αναγκαίο σε υπηρεσίες που επικοινωνούν σε πραγματικό χρόνο όπως ένα διαδικτυακό βιντεοπαιχνίδι μιας και είναι πιο αποδοτικό από άποψη άμεσης διαχείρισης της πληροφορίας, έχοντας ως αποτέλεσμα μικρότερη καθυστέρηση (lag) και μεγαλύτερο bandwidth. Παρακάτω απεικονίζεται μια τυπική επικοινωνία μέσω UDP.

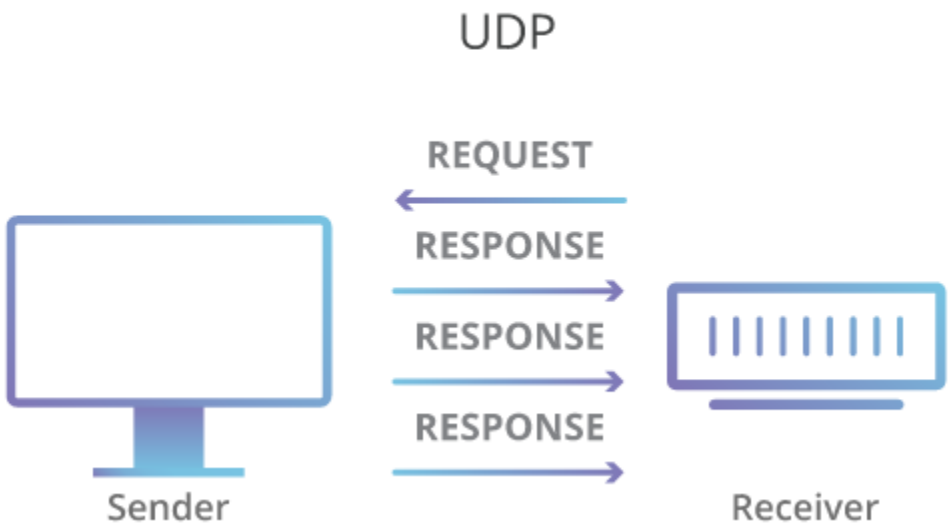


Figure 3. UDP πρωτόκολλο

Το παιχνίδι έχει αρχιτεκτονική Server – Client με ένα από τους παίκτες (clients) να αναλαμβάνει τον ρόλο του host. Την δουλειά του Server την κάνει το Photon Cloud, ένας server στο υπολογιστικό νέφος.

Από πλευράς επαυξημένης πραγματικότητας η εφαρμογή κάνει χρήση των τεχνολογιών αναγνώρισης του χώρου, ανίχνευσης κίνησης, αναγνώρισης εικόνων και προβολής εικονικών αντικειμένων στον πραγματικό κόσμο. Οι παίκτες έχουν στην κατοχή τους κάποιες κάρτες οι οποίες

αναγνωρίζονται από το ARCore και προβάλλουν τον αντίστοιχο χαρακτήρα στον χώρο της οθόνης. Εφόσον οι χρήστες είναι συνδεδεμένοι σε ένα κοινό εικονικό περιβάλλον, τότε μπορούν να αλληλοεπιδράσουν με του χαρακτήρες αυτούς και να τους δουν να ζωντανεύουν μπροστά τους.

2.2 ΠΡΟΕΤΟΙΜΑΣΙΑ ΧΩΡΟΥ ΕΡΓΑΣΙΑΣ

Προτού μπορέσει να ξεκινήσει η φάση της ανάπτυξης, πρέπει να οριστούν καλά τα προγράμματα που θα χρησιμοποιηθούν, τα εργαλεία, οι υπηρεσίες, οι βιβλιοθήκες, τα plug-ins, πρέπει να εγκατασταθούν στον υπολογιστή, να ρυθμιστούν κατάλληλα ώστε να συνεργάζονται σωστά και να τεσταριστούν.

Η μηχανή γραφικών της επιλογής μας είναι η Unity 2020.3.0f1 έκδοση που κυκλοφόρησε τον Μάρτιο του 2021. Κατά την εγκατάσταση της Unity πρέπει να συμπεριλάβουμε το πακέτο υποστήριξης για ανάπτυξη σε λειτουργικό Android και να το προσθέσουμε στην καρτέλα “external tools” της Unity όπως φαίνεται στο figure2. Επίσης για να είναι προ-εγκατεστημένο το Android Studio έκδοση 4.1.3. στην Συνέχεια από την καρτέλα Window → Package Manager που βρίσκεται στην γραμμή εργαλείων του Unity editor, βρίσκουμε και κάνουμε download και install τα απαραίτητα πακέτα για ανάπτυξη σε επαυξημένη πραγματικότητα, στην περίπτωσή μας τα ARCore και ARFoundaiton καθώς και τα απαραίτητα πρόσθετα πακέτα για XR development. Ακόμη κατεβάζουμε Photon Engine πακέτο από το Unity Asset Store και το κάνουμε import στο project μας.

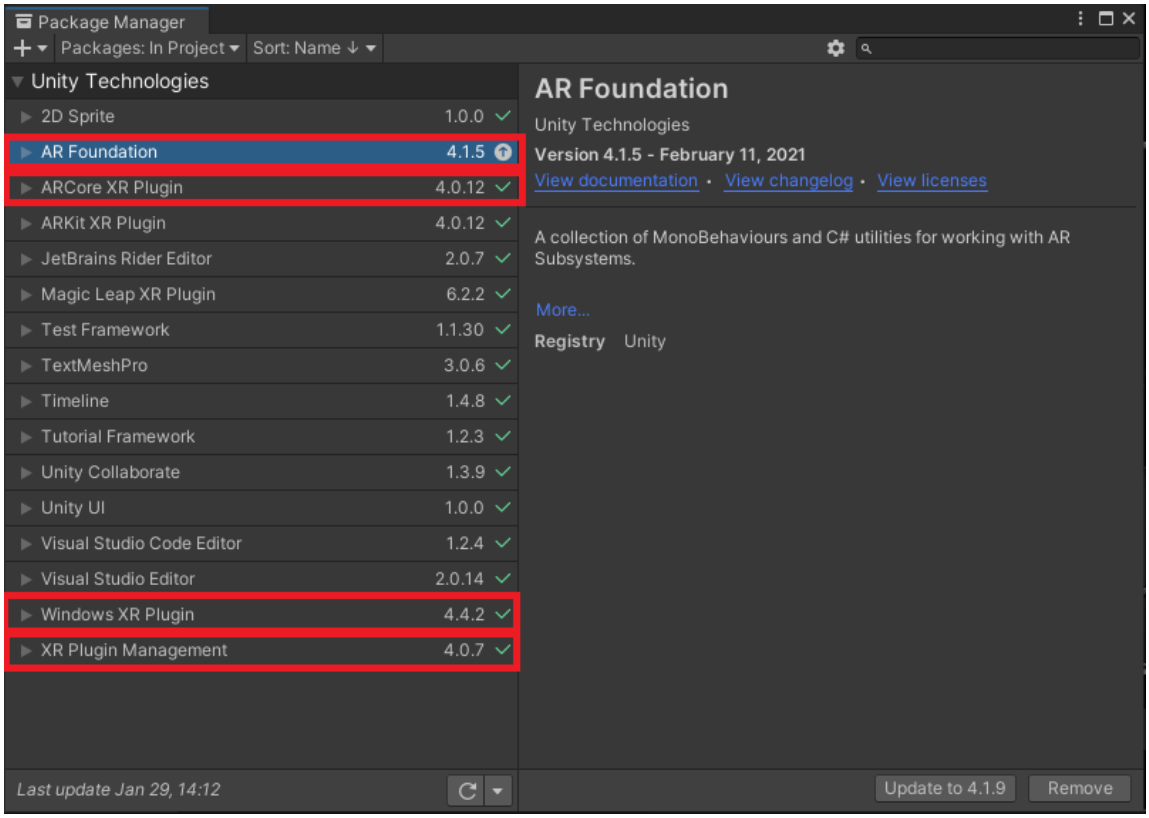


Figure 4. Plugins for AR Development

Όπως φαίνεται στην παραπάνω εικόνα χρησιμοποιούμε το ARFoundation 4.1.5 framework, το ARCore 4.0.12 plugin, το Windows XR Plugin 4.4.2 και το XR Plugin Management 4.0.7.

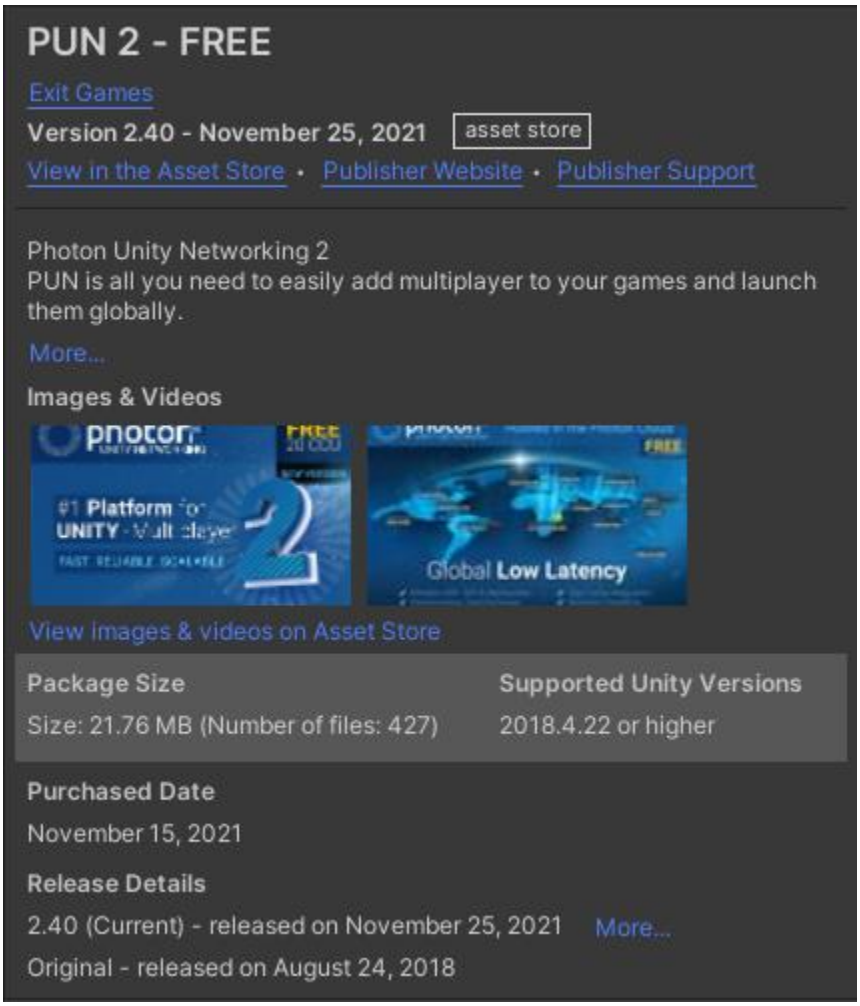


Figure 5. Photon PUN 2

Το PUN 2 είναι το Unity πακέτο για την επικοινωνία του παιχνιδιού με τους Servers του Photon Engine. Χρησιμοποιούμε την έκδοση 2.40.

Τέλος, είναι απαραίτητες ορισμένες ρυθμίσεις ώστε η Unity να είναι σε θέση να ανάπτυξη λογισμικό σε Android αλλά και για AR. Πρώτα πρέπει το project να μετατραπεί σε πλατφόρμα ανάπτυξης Android. Από την καρτέλα File → Build Settings → Platform, επιλέγουμε “Android” και πατάμε “Switch Platform”.

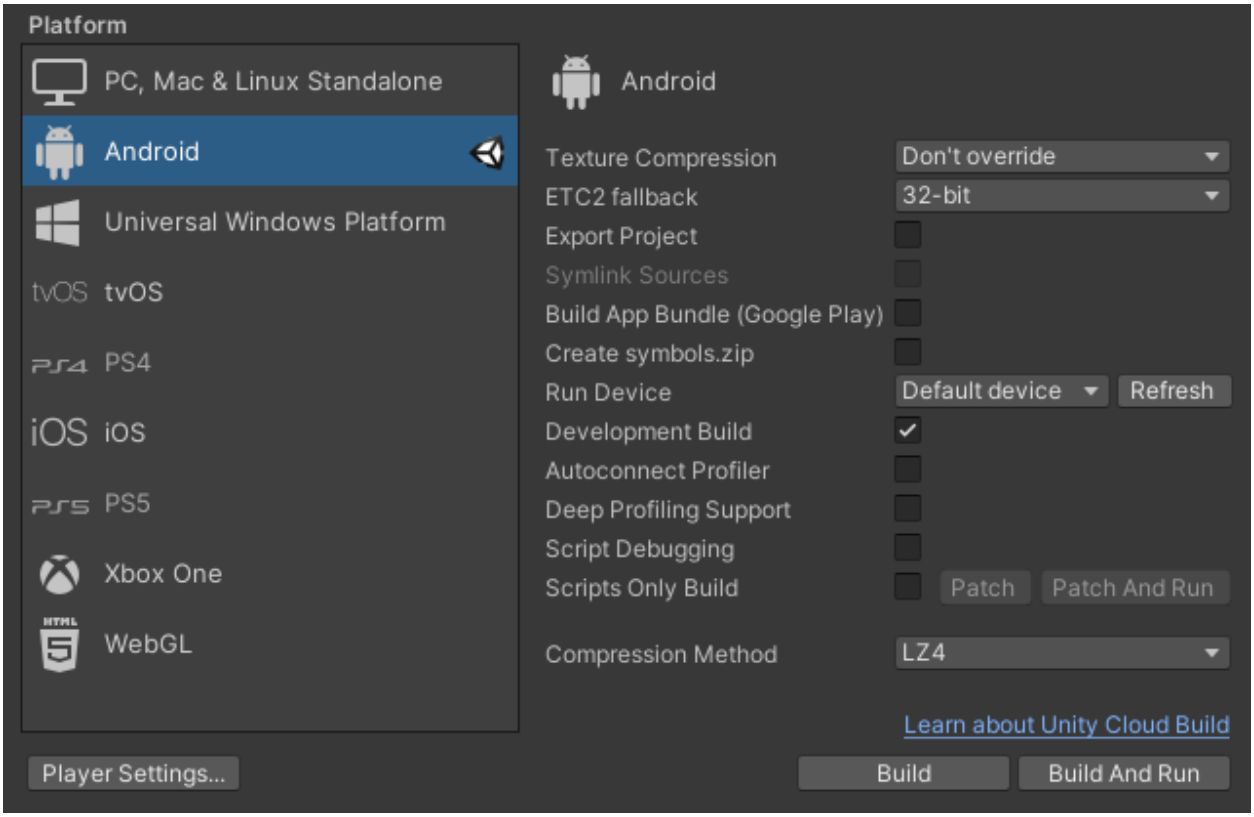


Figure 6. Build Platform Selection

Έπειτα, στο Player Settings εισάγουμε το όνομα του προγραμματιστή ή του οργανισμού, το όνομα της εφαρμογής, την έκδοσή της και στην καρτέλα Other Settings ξε-τικάρουμε την επιλογή Multithreaded Rendering, θέτουμε ως Minimum API Level: Android 7 'Nougat' και Target API: Automatic.

Figure 7. Player Settings

Ως τελευταίο βήμα πρέπει να ορίσουμε στην καρτέλα XR Plug-in Management, το ARCore ως Plug-in provider και από την καρτέλα του ARCore να επιλέξουμε για requirement: required.

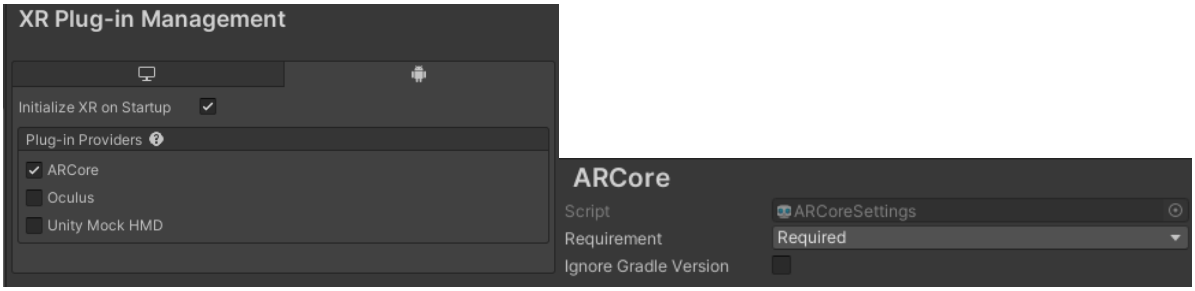


Figure 8. XR Settings

2.3 ΔΟΜΗ ΕΦΑΡΜΟΓΗΣ ΕΠΑΥΞΗΜΕΝΗΣ ΠΡΑΓΜΑΤΙΚΟΤΗΤΑΣ

Θεμελιώδεις έννοιες της επαυξημένης πραγματικότητας είναι η παρακολούθηση κίνησης, η κατανόηση του χώρου, η αντίληψη βάθους, η εκτίμηση φωτισμού, η διεπαφή με τον χρήστη, τα προσανατολισμένα σημεία (για την τοποθέτηση αντικειμένων), επαυξημένες εικόνες κα αντικείμενα. Για την εκπλήρωση των παραπάνω προδιαγραφών το ARFoundation προσφέρει τις εξής λειτουργίες:

- ***Device Tracking:*** Παρακολούθηση της θέσης και του προσανατολισμού της συσκευής στον φυσικό χώρο.
- ***Raycast:*** Χρησιμοποιείται για τον προσδιορισμό του σημείου που θα εμφανίζεται το εικονικό περιεχόμενο, όπου μια ακτίνα τέμνεται με ένα χαρακτηριστικό του πραγματικού κόσμου που ανιχνεύεται και παρακολουθείται από τη συσκευή.
- ***Plane detection:*** Εντοπίζει το μέγεθος και τη θέση των οριζόντιων και κάθετων επιφανειών στον χώρο. Αυτές οι επιφάνειες ονομάζονται «planes».
- ***Reference points:*** Παρακολουθεί και καταγράφει την θέση των planes στον χώρο.
- ***Point cloud detection:*** Εντοπίζει την θέση της κάμερας στον πραγματικό κόσμο μέσω διακριτών σημείων.
- ***Gestures:*** Αναγνωρίζει τις χειρονομίες ως συμβάντα εισόδου πάνω σε ανθρώπινα χέρια.
- ***Face tracking:*** Αντίστοιχα αναγνωρίζει και ακολουθεί την κίνηση των χαρακτηριστικών του προσώπου.
- ***2D image tracking:*** Ανίχνευση συγκεκριμένων δισδιάστατων εικόνων στον χώρο.
- ***3D object tracking:*** Ανίχνευση συγκεκριμένων τρισδιάστατων αντικειμένων στον χώρο**.**
- ***Environment probes:*** Ανιχνεύει πληροφορίες φωτισμού και χρώματος σε συγκεκριμένες περιοχές του περιβάλλοντος, γεγονός που βοηθά το τρισδιάστατο περιεχόμενο να συνδυάζεται άψογα με το περιβάλλον.
- ***Meshing:*** Δημιουργία τρίγωνων πλεγμάτων που αντιστοιχούν στον φυσικό χώρο, επεκτείνοντας την ικανότητα αλληλεπίδρασης με αναπαραστάσεις του φυσικού περιβάλλοντος.
- ***2D and 3D body tracking:*** Παρέχει δισδιάστατη ή τρισδιάστατη αναπαράσταση του ανθρώπινου σώματος που αναγνωρίζεται στο πλαίσιο της κάμερας.
- ***Human segmentation:*** Βοηθάει στην ρεαλιστική απεικόνιση ανθρώπινων φιγούρων πάνω στα τρισδιάστατα αντικείμενα.

- ***Occlusion***: Εφαρμόζει την αίσθηση της απόσταση σε αντικείμενα του φυσικού κόσμου από το τρισδιάστατο περιεχόμενο, το οποίο επιτυγχάνει μια ρεαλιστική ανάμειξη φυσικών και εικονικών αντικειμένων.
- ***Participant tracking:*** Παρακολούθηση της θέσης και του προσανατολισμού άλλων συσκευών σε μια κοινή εμπειρία επαυξημένης πραγματικότητας.

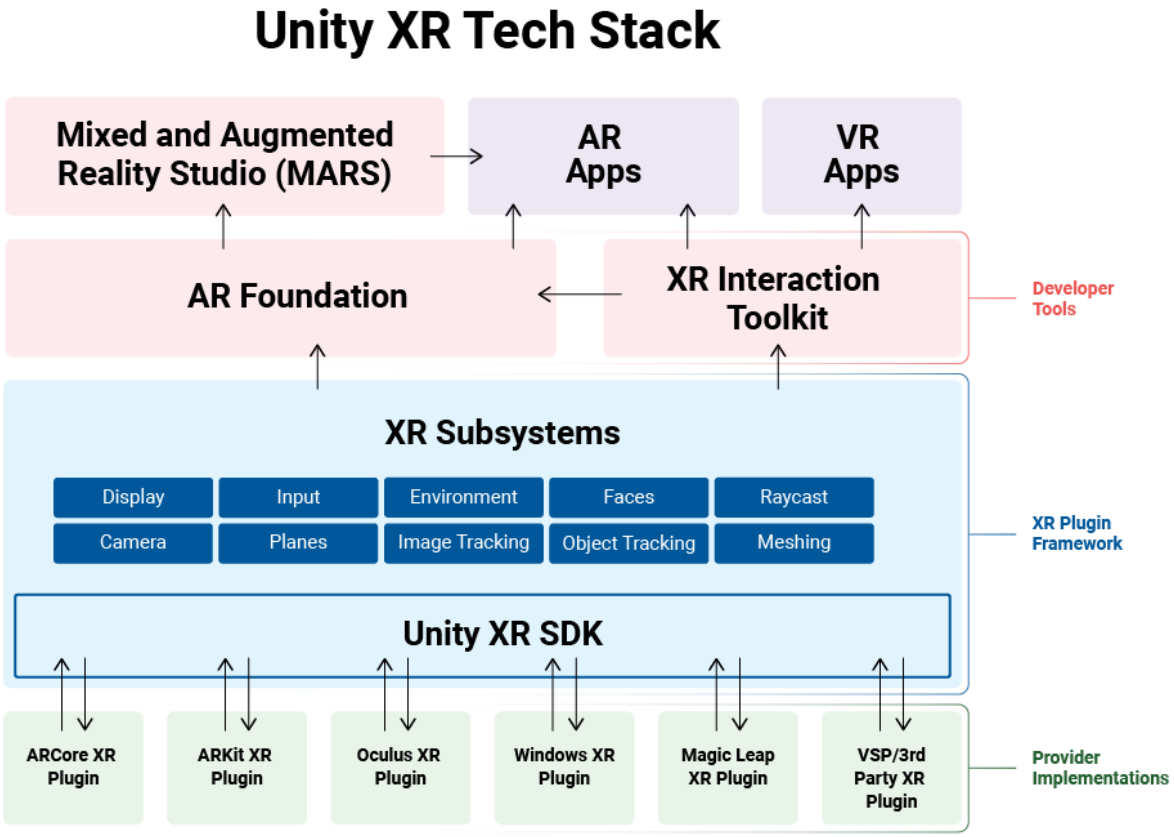


Figure 9. Unity XR Plug-in Framework

2.4 ΔΟΜΗ ΕΦΑΡΜΟΓΗΣ ΠΟΛΛΩΝ ΠΑΙΚΤΩΝ (MULTIPLAYER)

Τα παιχνίδια πολλών παιχτών (Multiplayer Games) έχουν αποκτήσει τεράστια δημοτικότητα με την άνοδο του διαδικτύου και πλέον όλα σχεδόν τα παιχνίδια προσφέρουν την δυνατότητα στον χρήστη να παίξει στο internet συνεργατικά μαζί με άλλους παίχτες. Οι χρήστες μπορούν να συνδέονται σε έναν απομακρυσμένο server που διαμοιράζει όλες τις πληροφορίες και δυνατότητες στους συνδεδεμένους χρήστες ενώ συχνά υπάρχει και η δυνατότητα κάποιος χρήστης να λειτουργεί ως server. Το Photon Engine δίνει αυτή την δυνατότητα στον χρήστη να δημιουργήσει αν το θέλει τον προσωπικό του server.

Τα multiplayer παιχνίδια διακρίνονται από μικρά παιχνίδια δύο παικτών σε πιο σύνθετα, των 6, 10, 20 μέχρι και εκατοντάδες ταυτόχρονα συνδεδεμένους παίκτες ενώ σε ορισμένες

κατηγορίες μπορούν να φτάσουν και χιλιάδες χρήστες ανάλογα με την έκταση του παιχνιδιού. Είναι λοιπόν πολύ σημαντική η σωστή επιλογή της αρχιτεκτονικής και η διαχείριση των πόρων ενός τέτοιου παιχνιδιού ώστε να μην υπάρχει ασυνέχεια στην σύνδεση, καθυστέρηση στην επικοινωνία και τον έλεγχο των γεγονότων και γενικά να μην δυσχεραίνεται η εμπειρία του χρήστη. Οι τρείς παράμετροι που επηρεάζουν περισσότερο την ποιότητα της δικτυακής επικοινωνίας σε ένα παιχνίδι πολλών χρηστών είναι το πλήθος των αντικειμένων που βρίσκονται στο οπτικό πεδίο της οθόνης, το μέσο μέγεθος των αντικειμένων αυτών και ο ρυθμός ανανέωσης του παιχνιδιού. Είναι λογικό πως αν το πλήθος και το μέγεθος των αντικειμένων που θα πρέπει να σχεδιαστούν από την υπολογιστική μονάδα σε συνδυασμό με γρήγορες εναλλαγές των καρέ (frames), αυτό θα συντελέσει σε μείωση της απόδοσης του παιχνιδιού.

Το παιχνίδι μας είναι ένα παιχνίδι καρτών επαυξημένης πραγματικότητας το οποίο θα παίζεται ανάμεσα σε δύο παίκτες και θα φιλοξενείται, μέχρι να αποχωρήσουν και οι δύο παίκτες από την συνεδρία, στο Photon Cloud. Το Photon Cloud παρέχει παγκόσμια συνδεσιμότητα με servers σε όλο τον κόσμο για μείωση της απώλειας ταχύτητας. Οι Ευρωπαϊκοί servers στους οποίους θα συνδεθούμε βρίσκονται στο Amsterdam. Η διαδικασία σύνδεσης των παικτών με τον server περνάει από τρία στάδια, αρχικά η σύνδεση των παικτών γίνεται στον Name Server που παρέχει μια λίστα με όλες τις διαθέσιμες περιοχές. Στην συνέχεια, αν δεν έχει οριστεί κατά προτίμηση, το Photon συνδέει τον χρήστη με τον πιο κοντινό γεωγραφικά σε αυτό Master Server. Από εκεί οι παίκτες θα συνδεθούν με ένα Game Server ο οποίος κρατάει όλες τις πληροφορίες, τα γεγονότα και αντικείμενα του παιχνιδιού και τα διαμοιράζει στους συνδεδεμένους χρήστες. Ο προγραμματιστής έχει πρόσβαση στην κατάσταση του δικτύου κάθε στιγμή μέσω της χρήσης Callbacks.

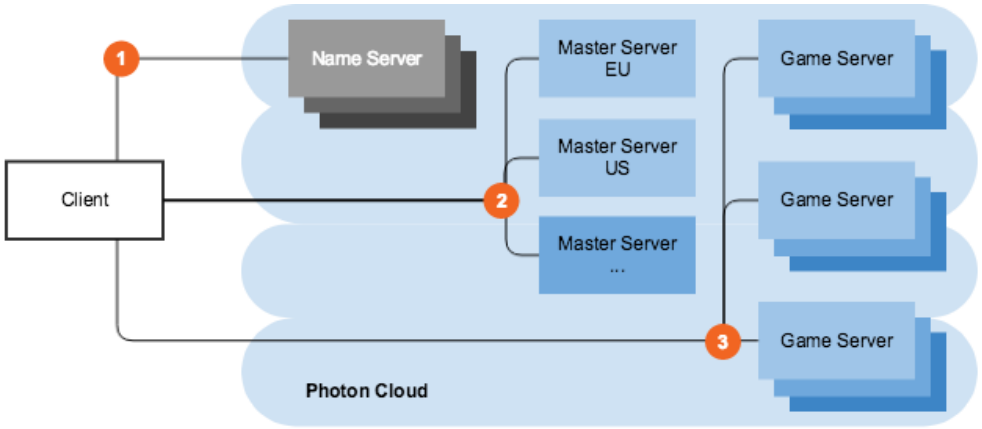


Figure 10. Σύνδεση στο Photon Cloud

ΚΕΦΑΛΙΑΟ 3: ΥΛΟΠΟΙΗΣΗ

3.1 ΕΙΣΑΓΩΓΗ

Η δημιουργία παιχνιδιών επαυξημένης πραγματικότητας πολλών παικτών είναι ένα εγχείρημα που πρωτοξεκίνησε με μικρά βήματα από τις αρχές του 2000, χωρίς όμως να γνωρίζει πολύ επιτυχία δεδομένης των περιορισμένων δυνατοτήτων που είχε τότε η τεχνολογία. Μια δεκαετία αργότερα, ο τομέας της επαυξημένης πραγματικότητας άρχισε να εξελίσσεται γοργά με αποτέλεσμα να εμφανιστούν τα πρώτα multiplayer AR παιχνίδια που φαίνεται να υπόσχονται ένα ελπιδοφόρο μέλλον για την εξέλιξη AR Mobile gaming. Αποκορύφωμα ήταν η κυκλοφορία, το 2016, του παιχνιδιού PokemonGO που βασίζεται στην ιαπωνική σειρά «Pokemon» και καλεί τους παίκτες να βγουν από το σπίτι και να ψάξουν στον πραγματικό κόσμο μικρά τέρατα ώστε να τα εκπαιδεύσουν και να κάνουν μάχες με άλλους εκπαιδευτές, ζωντανεύοντας έτσι τον εικονικό κόσμο μια από τις πιο αγαπημένες σειρές κινουμένων σχεδίων των παιδικών μας χρόνων.

Έμπνευση για το παιχνίδι που αναπτύσσουμε εδώ αποτελεί εν μέρη το PokemonGo καθώς και ένα ακόμα ιαπωνικό cartoon, το «YU-GI-OH». Στο YU-GI-OH οι πρωταγωνιστές έπαιρναν μέρος σε πρωταθλήματα καρτών, στα οποία χρησιμοποιούσαν ολογραφικά μηχανήματα για να προβάλουν το τέρας που βρισκόταν στην κάρτα που έπαιζε ο παίκτης, στην αρένα. Έτσι, στην παρούσα εργασία γίνεται η προσπάθεια βασιζόμενοι στην τεχνολογία του PokemonGO να δημιουργηθεί ένα εικονικό περιβάλλον στον πραγματικό κόσμο όπου ο παίκτης μπορεί να δει τους χαρακτήρες από το σύνολο των καρτών του σε τρισδιάστατη προβολή και να μονομαχήσει με άλλους παίκτες.

3.2 ΠΟΡΟΙ ΠΑΙΧΝΙΔΙΟΥ

Αφού έχει ολοκληρωθεί η εγκατάσταση όλων των απαραίτητων εργαλείων και η προετοιμασία του χώρου εργασίας όπως αυτή αναλύθηκε στο κεφάλαιο 2.2, σειρά έχει η δημιουργία ή η προσθήκη όλων των πόρων που θα χρησιμοποιηθούν κατά την διάρκεια του παιγνιδιού. Οι πόροι αυτοί αποτελούνται από τα 2D και 3D αντικείμενα, τις φωτογραφίες, τα UI στοιχεία, τα animations, τους ήχους κ.α. Για τον σκοπό της εργασίας χρησιμοποιούνται έτοιμα στοιχεία παιχνιδιού (game elements) που προσφέρονται δωρεάν στον επίσημο ηλεκτρονικό κατάστημα πόρων της Unity, το Unity Asset Store. Όλα τα στοιχεία παιχνιδιού που εκμεταλλεύεται ο προγραμματιστής για την δημιουργία του περιβάλλοντος ονομάζονται assets και αποθηκεύονται στον αντίστοιχο φάκελο μέσα στον Unity Editor. Ξεχωριστά εργαλεία όπως το Photoshop, Blender, Maya, 3DS MAX, Adobe Animate μπορούν να χρησιμοποιηθούν για την δημιουργία νέων μοναδικών assets. Παρακάτω ρίχνουμε μια ματιά στα πιο κύρια assets που θα κάθε project χρειάζεται για την παραγωγή ενός παιχνιδιού.

Figure 11. Project Assets

- ***(1) Scenes***: Ο φάκελος «Scenes» περιέχει όλες τις σκηνές που εναλλάσσονται στο παιχνίδι. Η κάθε σκηνή έχει το δικό της UI και τα δικά της αντικείμενα.
- ***(2) Scripts***: Στον φάκελο «Scripts» αποθηκεύονται τα διάφορα αρχεία C# κώδικα τα οποία υλοποιούν την λειτουργία του παιχνιδιού. Τα scripts ενσωματώνονται πάνω στην σκηνή ή στο εκάστοτε αντικείμενο που πρέπει να επηρεάσουν βάση της λογικής του παιχνιδιού. Κάνοντας διπλό κλικ σε κάποιο script ανοίγει το αρχείο κώδικα στο Visual Studio.
- ***(3) Prefabs***: Τα Prefabs στην Unity είναι διαμορφωμένα GameObjects που αποθηκεύονται με όλα τα στοιχεία, τις τιμές ιδιοτήτων και τα GameObjects παιδιά τους ως επαναχρησιμοποιήσιμο στοιχείο.
- ***(4) Resources***: Ο φάκελος «Resources» εξυπηρετεί ως σημείο αναφοράς για την κλάση "Resources". Όσα αντικείμενα βρίσκονται μέσα στον φάκελο αυτό μπορούν να κληθούν ανά πάσα στιγμή μέσα από τον κώδικα στην σκηνή του παιχνιδιού.
- ***(5) Fonts***: Οι γραμματοσειρές των κειμένων που εμφανίζονται στην οθόνη.
- ***(6) Sprites***: Τα Sprites είναι αντικείμενα 2D γραφικών. Τα Sprites είναι ουσιαστικά τυπικές υφές, που προσφέρουν όμως ειδικές τεχνικές για το συνδυασμό και τη διαχείριση των textures για αποτελεσματικότητα και ευκολία κατά την ανάπτυξη.
- ***(7) Materials***: Ένα material περιέχει μια αναφορά στον Shader ενός αντικειμένου. Εάν αυτό το αντικείμενο Shader ορίζει ιδιότητες υλικού, τότε το υλικό μπορεί επίσης να περιέχει δεδομένα όπως χρώματα ή αναφορές σε υφές. Τα materials εφαρμόζονται πάνω σε κάποιο αντικείμενο για να ορίσουν την εμφάνισή τους.
- ***(8) XR***: Στον φάκελο «XR» αποθηκεύονται όλα τα εργαλεία και οι πόροι του συστήματος για την επαυξημένη πραγματικότητα.
- ***(9) Photon***: Αντίστοιχα ο φάκελος «Photon» διατηρεί όλες τις πληροφορίες για το multiplayer κομμάτι της εφαρμογής. Δημιουργείται αυτόματα με την εγκατάσταση του πακέτου.

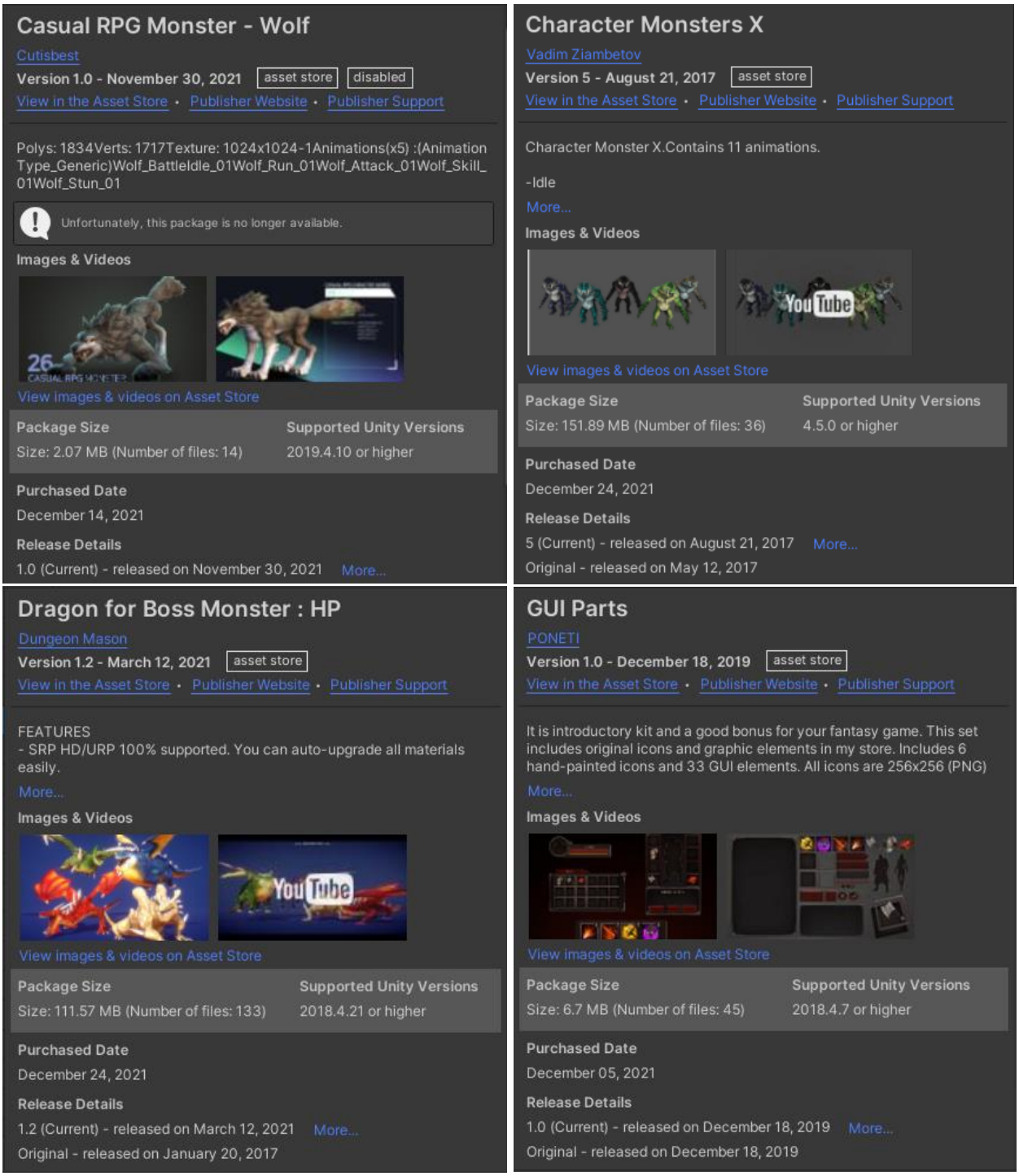


Figure 12. Game Assets

Στις παραπάνω φωτογραφίες φαίνονται κάποια game elements που δανειζόμαστε από το Asset Store για ενσωμάτωση στο project. Αυτά είναι κάποιοι 3D χαρακτήρες που έχουν το ρόλο των τεράτων στις κάρτες και στοιχεία για τον σχεδιασμό της διεπαφής χρήστη όπως κουμπιά, πλαίσια και δισδιάστατα γραφικά για τον εμπλουτισμό της σκηνής.

3.3 ΥΛΟΠΟΙΗΣΗ ΤΟΥ MULTIPLAYER

3.3.1. Μια εισαγωγή στην δημιουργία παιχνιδιών πολλών παικτών (Multiplayer Games)

Η δημιουργία ενός περιβάλλοντος στο οποίο μπορούν να βρίσκονται ταυτόχρονα πολλοί παράλληλα συνδεδεμένοι χρήστες και να αλληλοεπιδρούν με τον κόσμο αλλά και μεταξύ τους είναι από τα κύρια θέματα της παρούσας εργασίας. Φυσικά, για τον λόγο αυτό καλούμαστε να βρούμε μια λύση που θα μας επιτρέψει να υλοποιήσουμε όλες τις απαραίτητες διαδικασίες που καθιστούν ένα παιχνίδι, παιχνίδι πολλαπλών παικτών. Κάποια από τα συστατικά που απαρτίζουν ένα τέτοιο ηλεκτρονικό παιχνίδι είναι η επικοινωνία μεταξύ απομακρυσμένων συσκευών, η διαχείριση των πόρων πάνω από το δίκτυο, ο συγχρονισμός των κινήσεων των παικτών σε όλα τα διαφορετικά στιγμιότυπα του παιχνιδιού για τους διαφορετικούς παίκτες, η αλληλεπίδραση των στοιχείων του παιχνιδιού στην τοπική συσκευή με την απομακρυσμένη (πχ. Επίθεση στον χαρακτήρα του αντιπάλου) κ.α. Η δημιουργία τέτοιων μεθόδων και συστημάτων είναι ιδιαίτερα δύσκολη και απαιτητική, κυρίως για ανεξάρτητους προγραμματιστές ή μικρές ομάδες ανάπτυξης. Εδώ έρχεται το Photon Engine για να ρίξει λίγο φως στο σκοτάδι. Το Photon Engine είναι μια ολοκληρωμένη λύση που προσφέρει μεγάλη ποικιλία εργαλείων ανάπτυξης multiplayer παιχνιδιών τόσο για την Unity όσο και άλλες μηχανές γραφικών. Έτσι, η υλοποίηση ενός παιχνιδιού πολλών παικτών γίνεται αρκετά ευκολότερη απ’ ότι έχουμε συνηθίσει.

3.3.2. Ενσωμάτωση Photon Unity Networking (PUN) 2 πακέτου στο Unity Project

Μπορούμε να βρούμε το PUN 2 πακέτο στο Unity Asset Store και να το προσθέσουμε από εκεί στην συλλογή μας. Το Unity Asset Store είναι το επίσημο marketplace της Unity το οποίο είναι συνδεδεμένο με τον λογαριασμό μας και οτιδήποτε επιλέξουμε ή αγοράσουμε προστίθεται αυτόματα στον Unity editor που χρησιμοποιούμε. Το Photon Engine διαθέτει πολλές διαφορετικές εκδόσεις του στα πακέτα του, οι περισσότερες από τις οποίες είναι επί πληρωμή και χρησιμοποιούνται κυρίως για έργα που προορίζονται για εμπορική χρήση. Για το σκοπό της εργασίας χρησιμοποιήθηκε η δωρεάν έκδοση του PUN 2, η οποία προσφέρει όλα όσα χρειαζόμαστε με μόνο μειονέκτημα ότι έχει περιορισμένη εμβέλεια όσον αφορά τον αριθμό των παικτών ανά συνεδρία.

Πριν ξεκινήσουμε την δημιουργία του παιχνιδιού μας, πρέπει να συνδέσουμε το project με το Photon Cloud. Το Photon Cloud είναι μια υπηρεσία νέφους στην οποία πολλοί υπολογιστές τρέχουν ταυτόχρονα τον server που είναι υπεύθυνος για τις multiplayer υπηρεσίες. Αμέσως μόλις προσθέσουμε το PUN 2 Unity Package ο editor θα μας ζητήσει να συμπληρώσουμε ένα πεδίο με το μοναδικό AppID που πρέπει να παράγουμε από την σελίδα του Photon Engine.

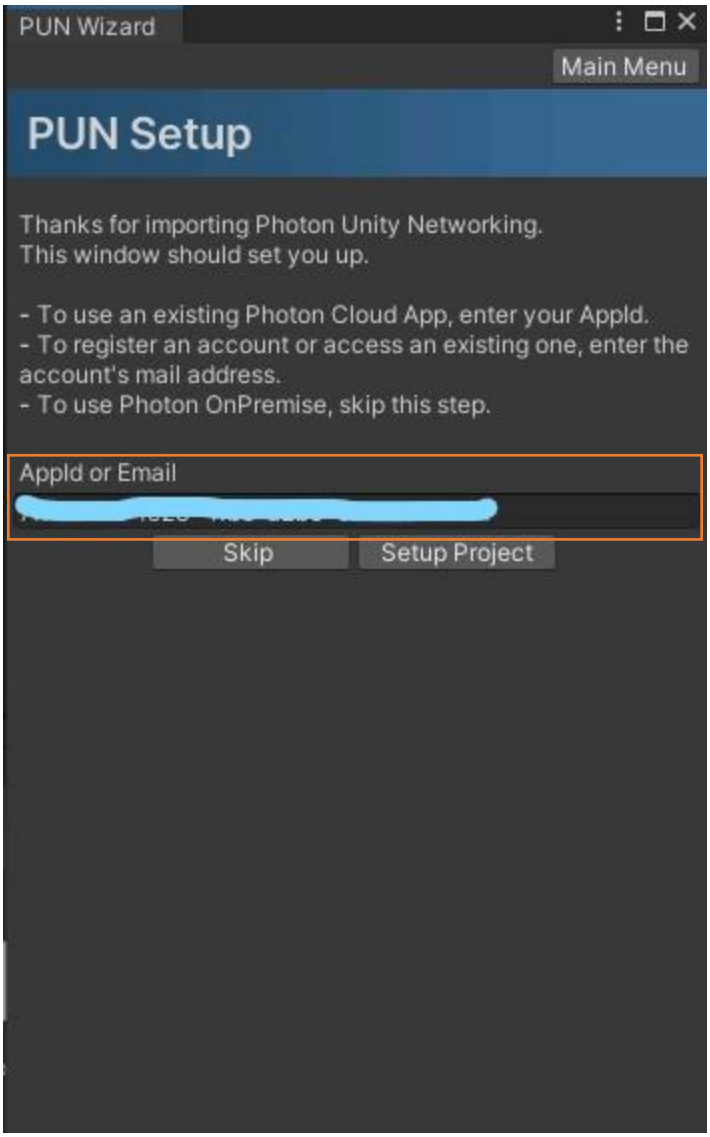


Figure 13.PUN Wizard

Μπαίνοντας λοιπόν στην ιστοσελίδα του Photon Engine, θα πρέπει να δημιουργήσουμε έναν λογαριασμό εφόσον δεν έχουμε ήδη. Αφού κάνουμε είσοδο με τα στοιχεία μας θα βρεθούμε στην κεντρική σελίδα του Dashboard στο οποίο βρίσκονται παρόν όλα τα project που έχουμε δημιουργήσει μέχρι τότε. Για να ξεκινήσουμε ένα καινούριο project πατάμε στο κουμπί "CREATE A NEW APP" το οποίο θα μας προτρέψει να επιλέξουμε το πακέτο που θα χρησιμοποιήσουμε για την υλοποίηση του multiplayer, να δώσουμε ένα όνομα στην εφαρμογή μας και προαιρετικά μια περιγραφή.

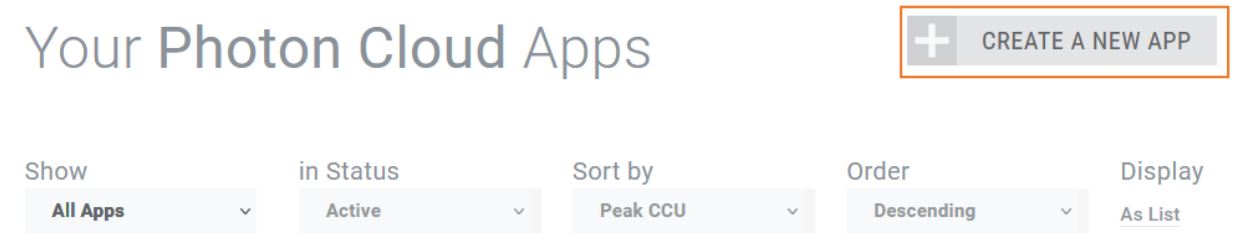


Figure 14.Photon Dashboard - Create new App

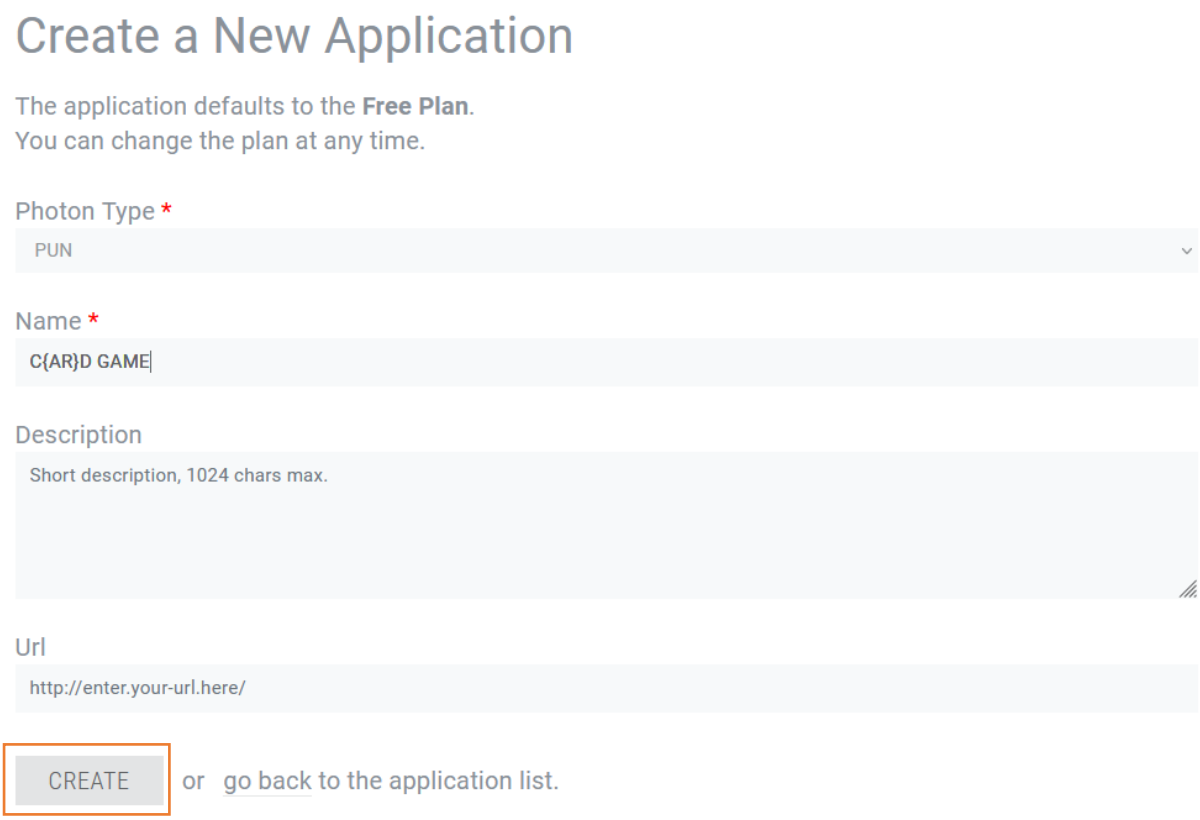


Figure 15. Photon Dashboard - Application Setup

Αφού πατήσουμε το κουμπί "CREATE" θα δημιουργηθεί η αντίστοιχη καρτέλα με τις πληροφορίες για την εφαρμογή που μόλις δημιουργήσαμε καθώς και το μοναδικό AppId της εφαρμογής.

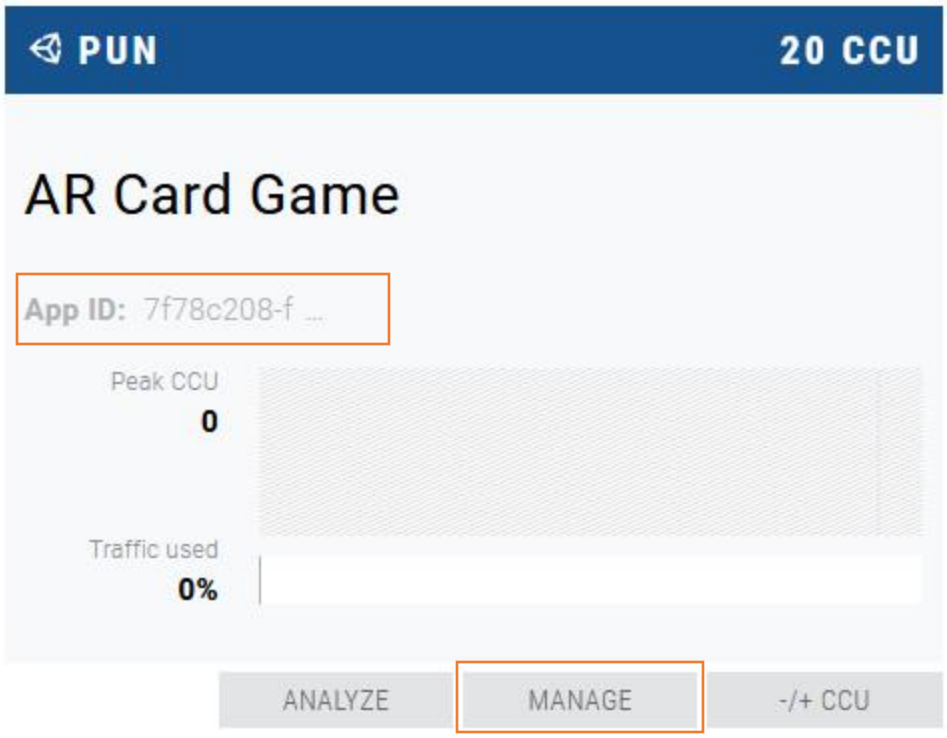


Figure 16. Photon Dashboard - Application Card

Όπως φαίνεται στο πάνω δεξιά μέρος της καρτέλας, η εφαρμογή μας μπορεί να υποστηρίξει έως και 20 CCU (Concurrent Users). Αυτό σημαίνει πως επειδή χρησιμοποιούμε την δωρεάν έκδοση, μόνο μέχρι 20 παίκτες μπορούν να βρίσκονται ταυτόχρονα στο ίδιο δωμάτιο. Επίσης κάτω από το όνομα αναγράφεται το αναγνωριστικό App ID στο οποίο αν πατήσουμε πάνω εμφανίζεται ολόκληρο ώστε να μπορούμε να το αντιγράψουμε στο αντίστοιχο πεδίο του PUN Wizard όπως φαίνεται στο figure 13 και να συνδέσουμε το Unity Project μας με το Photon Cloud.

Επιλέγοντας το κουμπί “MANAGE” μπορούμε να εποπτεύσουμε τις ιδιότητες που παρέχονται στην συγκεκριμένη εφαρμογή που δημιουργήσαμε. Συγκεκριμένα αυτό που μας ενδιαφέρει είναι το “Regions Whitelist”. Όταν συνδεόμαστε μέσα από την εφαρμογή σε έναν server, θα συνδεθούμε με τον πιο κοντινό σε απόσταση server. Το Photon Cloud παρέχει servers σε πολλές περιοχές ανά τον κόσμο, εδώ μπορούμε να φιλτράρουμε από ποια περιοχή server θέλουμε να συνδέεται η εφαρμογή ώστε να εξασφαλίσουμε την λιγότερη απώλεια σε ταχύτητα επικοινωνίας και lag. Για τον λόγο αυτό, μιας και είμαστε στην Ελλάδα, θα επιλέξουμε να συνδεόμαστε με τον Ευρωπαϊκό server (EU) που βρίσκεται στο Amsterdam.

Figure 17. Χάρτης των servers στο Photon Cloud

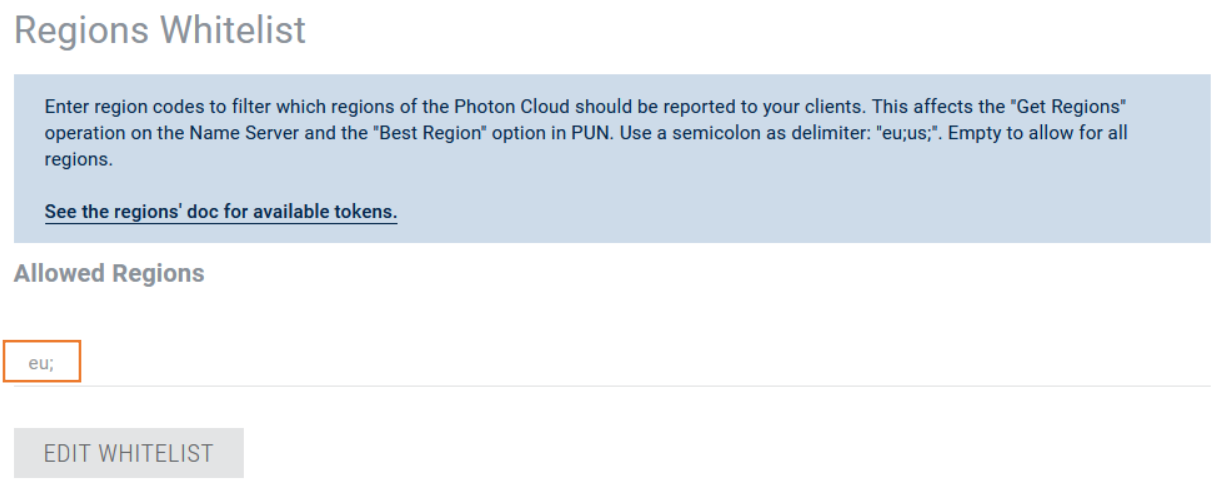


Figure 18. Photon Dashboard - Regions Whitelist

Αφού ολοκληρωθούν οι παραπάνω ενέργειες, το project είναι πλέον συνδεδεμένο με το Photon Cloud και οι multiplayer λειτουργίες μπορούν να υλοποιηθούν.

3.3.3. Σύνδεση στον Photon Server με το username του παίκτη

Πρώτα απ' όλα θα πρέπει να δημιουργήσουμε ένα γραφικό περιβάλλον για τον χρήστη που θα εξυπηρετεί ως η αρχική οθόνη του παιχνιδιού την οποία θα αντικρίζει με το ανοίξει την εφαρμογή. Στην οθόνη αυτή παρουσιάζονται οι κύριες επιλογές για τον

χρήστη όσον αφορά την συνέχεια στο παιχνίδι και τις ρυθμίσεις αυτού. Επιλέγοντας ο χρήστης το κουμπί "Play", θα προχωρήσει σε μια δεύτερη οθόνη όπου του ζητείται να πληκτρολογήσει ένα όνομα που θα αντιπροσωπεύει το όνομά του κατά την διάρκεια του παιχνιδιού και το οποίο θα είναι ορατό στους υπόλοιπους παίκτες που βρίσκονται συνδεδεμένοι στην ίδια συνεδρία παιχνιδιού.

Figure 19. Αρχική οθόνη εφαρμογής

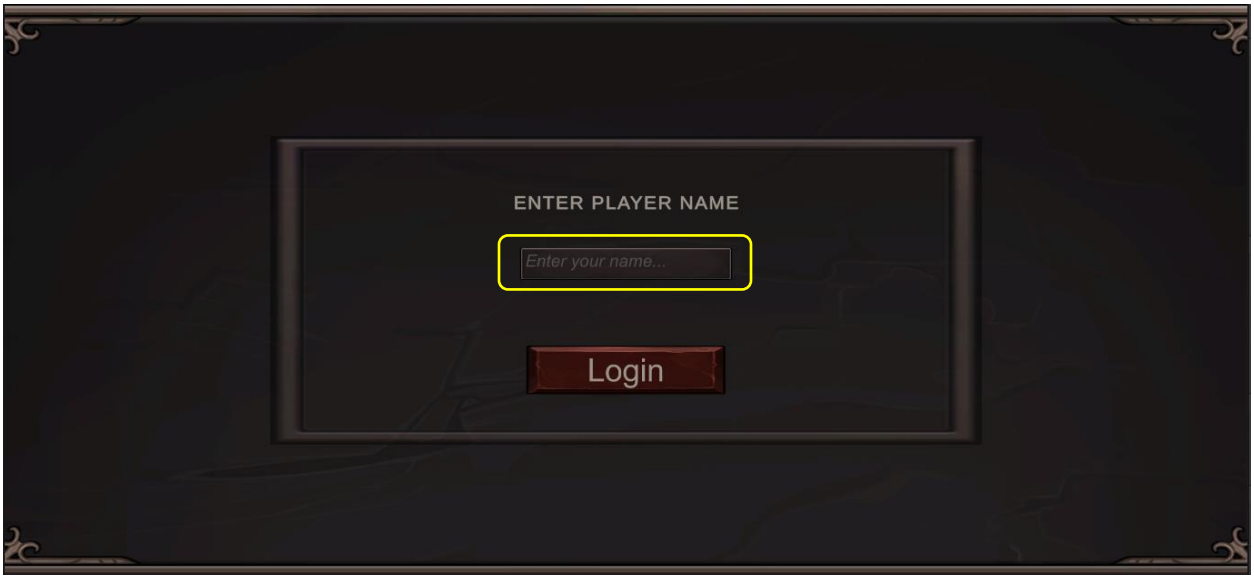


Figure 20. Login Panel

Για τον έλεγχο και την διαχείριση όλων των στοιχείων του γραφικού περιβάλλοντος όπως είναι τα κουμπιά, τα πεδία εισαγωγής κειμένου, οι εικόνες κτλ. αλλά

και την εκτέλεση των ενεργειών του UI όταν αλληλοεπιδρά με αυτό ένας χρήστης θα συντάξουμε ένα C# Script που θα είναι υπεύθυνο να αντιστοιχίσει τις επιλογές του χρήστη με την οπτική απεικόνιση στην οθόνη. Αυτό το script είναι το "PlayManager.cs". Όλα τα script με τον C# κώδικα που θα χρησιμοποιήσουμε βρίσκονται μέσα στον φάκελο "Scripts", στον Unity Editor. Όπως φαίνεται από την ονομασία των μεθόδων, η οποία είναι αρκετά περιγραφική για λόγους καλύτερης κατανόησης της λειτουργίας που επιτελούν, τα παρακάτω κομμάτια κώδικα διαχειρίζονται τις ενέργειες που τρέχουν στο περιβάλλον της Unity όταν πατηθεί το κουμπί "Play".

```
0 references
public void OnStartButtonClicked()
{
    if (PhotonNetwork.IsConnected)
    {
        menuCanvas.SetActive(true);
        loginCanvas.SetActive(false);
        connectionCanvas.SetActive(false);
    }
    else
    {
        menuCanvas.SetActive(false);
        loginCanvas.SetActive(true);
        connectionCanvas.SetActive(false);
    }
}
```

Το PUN 2 δίνει στον προγραμματιστή την ευκολία να παρακολουθεί και να καταγράφει το status της εφαρμογής στο δίκτυο ή του παίκτη στον server. Έτσι μπορεί να ελέγξει αν σε μια δεδομένη στιγμή ο χρήστης έχει συνδεθεί σε κάποιο δημόσιο δωμάτιο ή όχι και να πράξει αντίστοιχα. Η δυνατότητα αυτή παρέχεται από το Photon Engine μέσω της χρήσης Callbacks. Ο προγραμματιστής έχει πρόσβαση στις Callback συναρτήσεις εφόσον η κλάση που τις καλεί, κληρονομεί από το κατάλληλο interface στο οποίο υπάρχει η υλοποίησή του. Πρέπει λοιπόν η κλάση "PlayManager" να κληρονομήσει την "MonoBehaviourPunCallbacks" διεπαφή.

```
Unity Script | 0 references
public class PlayManager : MonoBehaviourPunCallbacks
```

Εφόσον έχουμε πρόσβαση, μπορούμε να υλοποιήσουμε δύο Callback μεθόδους, στην προκειμένη περίπτωση, για να ελέγξουμε αν ο παίκτης είναι σε πρώτη φάση συνδεδεμένος στο internet και έπειτα αν έχει συνδεθεί με τον κύριο server στο Photon Cloud. Από εκεί και έπειτα υλοποιούμε όποια εργασία θέλουμε μέσα στις Callbacks. Υπενθυμίζεται ότι ο server που γίνεται η σύνδεση θα είναι πάντα αυτός που έχει

προεπιλεγεί από το Photon Dashboard κατά την αρχικοποίηση, σε διαφορετική περίπτωση θα είναι πάντα αυτός που βρίσκεται πιο κοντά γεωγραφικά με τον χρήστη που ζητάει να συνδεθεί.

```
#region Photon Callback Methods
8 references
public override void OnConnected()
{
    Debug.Log("We Connected to the Internet!");
}

14 references
public override void OnConnectedToMaster()
{
    Debug.Log(PhotonNetwork.LocalPlayer.NickName + " is connected to Photon Server");
    SceneManager.LoadScene("Lobby_Scene");
}
#endregion
```

3.3.4. Ενημέρωση του χρήστη για τα στάδιο της σύνδεσης στο γραφικό περιβάλλον

Ένας άλλος τρόπος να καταγράψουμε την κατάσταση της σύνδεσης του χρήστη στον server είναι με την μέθοδο “PhotonNetwork.NetworkClientState” η οποία μας δίνει σε κάθε δεδομένη στιγμή τα στάδια που περνάει ο client του παιχνιδιού μέχρι να συνδεθεί με τον server.

```
#region Unity Methods
// Update is called once per frame
Unity Message | 0 references
void Update()
{
    if (showConnectionStatus)
    {
        connectionStatusText.text = "Connection Status: " + PhotonNetwork.NetworkClientState;
    }
}
#endregion
```

Επιγραμματικά, τα states που μπορεί να βρεθεί ο client του παιχνιδιού κατά την διαδικασία της σύνδεσης είναι τα παρακάτω:

◆ ClientState

enum ClientState strong

State values for a client, which handles switching Photon server types, some operations, etc.

Enumerator	
PeerCreated	Peer is created but not used yet.
Authenticating	Transition state while connecting to a server. On the Photon Cloud this sends the AppId and AuthenticationValues (UserID).
Authenticated	Not Used.
JoiningLobby	The client sent an OpJoinLobby and if this was done on the Master Server, it will result in. Depending on the lobby, it gets room listings.
JoinedLobby	The client is in a lobby, connected to the MasterServer. Depending on the lobby, it gets room listings.
DisconnectingFromMasterServer	Transition from MasterServer to GameServer.
ConnectingToGameServer	Transition to GameServer (client authenticates and joins/creates a room).
ConnectedToGameServer	Connected to GameServer (going to auth and join game).
Joining	Transition state while joining or creating a room on GameServer.
Joined	The client entered a room. The CurrentRoom and Players are known and you can now raise events.
Leaving	Transition state when leaving a room.
DisconnectingFromGameServer	Transition from GameServer to MasterServer (after leaving a room/game).
ConnectingToMasterServer	Connecting to MasterServer (includes sending authentication values).
Disconnecting	The client disconnects (from any server). This leads to state Disconnected.
Disconnected	The client is no longer connected (to any server). Connect to MasterServer to go on.
ConnectedToMasterServer	Connected to MasterServer. You might use matchmaking or join a lobby now.
ConnectingToNameServer	Client connects to the NameServer. This process includes low level connecting and setting up encryption. When done, state becomes ConnectedToNameServer.
ConnectedToNameServer	Client is connected to the NameServer and established encryption already. You should call OpGetRegions or ConnectToRegionMaster.
DisconnectingFromNameServer	Clients disconnects (specifically) from the NameServer (usually to connect to the MasterServer).
ConnectWithFallbackProtocol	Client was unable to connect to Name Server and will attempt to connect with an alternative network protocol (TCP).

Figure 21. Network Client States

Ουσιαστικά, με το που εισάγει ο χρήστης το username του και πατήσει το κουμπί “Login”, ο client θα προσπαθήσει να συνδεθεί με τον server. Για να μπορεί ο χρήστης να έχει την εποπτεία της σύνδεσής του, έχει δημιουργηθεί ένα νέο παράθυρο το οποίο ανοίγει με το πάτημα του κουμπιού “Login” στο οποίο τυπώνονται οι διάφορες φάσεις από τις οποίες περνάει ο client μέχρι την σύνδεση. Την στιγμή που η σύνδεση έχει ολοκληρωθεί, ο παίκτης μεταφέρεται στο Lobby Page.

Figure 22. Status Info Panel

3.3.5. Singleton Pattern Implementation και επιλογή σκηνής

Γενικά η αλλαγή σκηνών είναι ένα φαινόμενο που παρουσιάζεται πολύ συχνά κατά την διάρκεια ενός παιχνιδιού όσο αυτό προχωράει. Στο συγκεκριμένο παιχνίδι καλούμαστε να αλλάζουμε σκηνές μετά την σύνδεση του παίκτη με τον server όπου φορτώνεται η σκηνή του “Lobby”, έπειτα η επιλογή να ξεκινήσει ένα παιχνίδι με τον αντίπαλο όπου με την σειρά του θα φορτώσει την σκηνή του κυρίως παιχνιδιού και τέλος εφόσον αναδειχτεί νικητής και χαμένος η σκηνή γυρνάει πίσω στην σκηνή του Lobby. Θα θέλαμε λοιπόν πολλές φορές ανάμεσα στην εναλλαγή σκηνών να φορτώνεται μια ενδιάμεση σκηνή που ενημερώνει τον χρήστη ότι η επόμενη σκηνή βρίσκεται στην διαδικασία φόρτωσης.

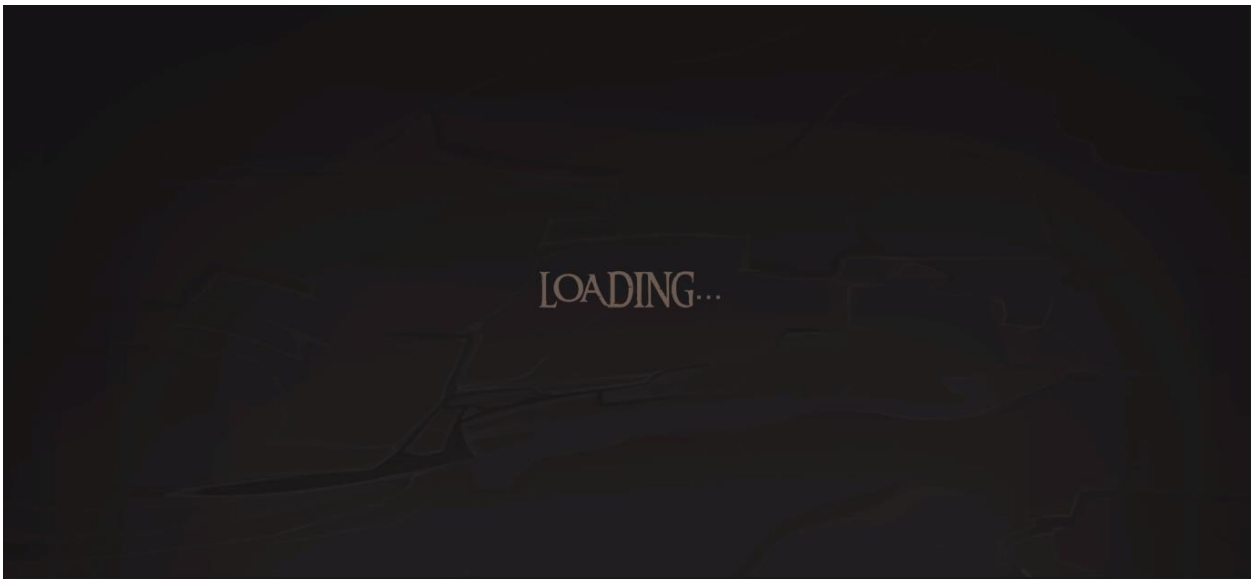


Figure 23. Loading Page

Χρειαζόμαστε λοιπόν έναν εύκολο και έξυπνο τρόπο να χρησιμοποιούμε την ίδια σκηνή πολλές φορές καλώντας την από διαφορετικά κάθε φορά scripts. Εδώ αναλαμβάνει δράση η υλοποίηση Singleton. Το Singleton pattern στην Unity είναι μια globally προσβάσιμη κλάση η οποία μπορεί να υπάρχει σε κάθε σκηνή μια μόνο φορά, λέμε δηλαδή ότι ένα μόνο instance της κλάσης δημιουργείται στην σκηνή που την καλεί. Η ιδέα πίσω από αυτή την υλοποίηση είναι ότι κάθε άλλο script μπορεί να έχει πρόσβαση στο Singleton script και άρα πολλά σημαντικά κομμάτια του παιχνιδιού που αυτό περιλαμβάνει. Στην περίπτωσή μας την συνάρτηση που είναι υπεύθυνη να φορτώσει την ενδιάμεση σκηνή "Loading". Αρκεί λοιπόν να φτιάξουμε ένα νέο script που θα ονομάσουμε "SceneLoader.cs" το οποίο θα κληρονομεί από την κλάση Singleton.

```
Unity Script | 2 references
public class SceneLoader : Singleton<SceneLoader>
{
    private string sceneToBeLoaded;
    1 reference
```

Μέσα στο παρόν script υλοποιούνται δύο κο-ρουτίνες (Coroutine), μια για την φόρτωση της ενδιάμεσης σκηνής (Loading Scene) και μια για την τελική σκηνή που θέλουμε να φορτωθεί στο παιχνίδι, η οποία διαφέρει από script σε script. Οι κο-ρουτίνες είναι συναρτήσεις τύπου IEnumerator που έχει την δυνατότητα να καθυστερήσει την εκτέλεσή της έως ότου έχει τελειώσει μια άλλη διαδικασία. Στην περίπτωσή μας θέλουμε να φορτώνεται η κύρια σκηνή αφού έχει τελειώσει η φόρτωση της ενδιάμεσης σκηνής. Υπεύθυνη για την φόρτωση των σκηνών του παιχνιδιού είναι η μέθοδος SceneManager.LoadScene της βιβλιοθήκης SceneManagement της Unity. Οι δύο κο-ρουτίνες φαίνονται παρακάτω.

```
1 reference
IEnumerator InitializeSceneLoading()
{
    yield return SceneManager.LoadSceneAsync("Loading_Scene");

    StartCoroutine(LoadActualScene());
}

1 reference
IEnumerator LoadActualScene()
{
    var asyncSceneLoading = SceneManager.LoadSceneAsync(sceneToBeLoaded);
    asyncSceneLoading.allowSceneActivation = false;

    while (!asyncSceneLoading.isDone)
    {
        if(asyncSceneLoading.progress >= 0.9f)
        {
            asyncSceneLoading.allowSceneActivation = true;
        }
        yield return null;
    }
}
```

3.3.6. Δημιουργία και συμμετοχή σε δωμάτιο (Room Creation and Join)

Από την στιγμή που είμαστε βέβαιοι ότι ο παίκτης έχει συνδεθεί επιτυχώς στο δίκτυο και στον Photon Server και έχει γίνει αποτελεσματικά η εναλλαγή των σκηνών, ο παίκτης πρέπει να βρίσκεται πλέον από την Login σκηνή στο Lobby, τον προθάλαμο δηλαδή του παιχνιδιού, στον οποίο οι παίκτες μπορούν να ζητήσουν να συνδεθούν σε ένα δωμάτιο ή αν δεν υπάρχει κάποιο δωμάτιο να το δημιουργήσουν και να συνδεθούν σε αυτό. Με τον όρο «δωμάτιο» εννοούμε έναν εικονικό χώρο στον server που είναι συνδεδεμένοι χρήστες, στον οποίο μπορούν να αλληλοεπιδρούν μεταξύ τους και να επικοινωνούν. Για να είναι ορατός ένας παίκτης σε έναν άλλο πρέπει οπωσδήποτε να συνυπάρχουν σε έναν κοινό χώρο, το δωμάτιο. Ένα παιχνίδι πολλών παικτών μπορεί να διαθέτει πολλά δωμάτια στα οποία παίκτες μπορούν να συνδεθούν. Κάθε δωμάτιο θα αναπαράγει διαφορετικά στιγμιότυπα του ίδιου παιχνιδιού για τους χρήστες που το χρησιμοποιούν.

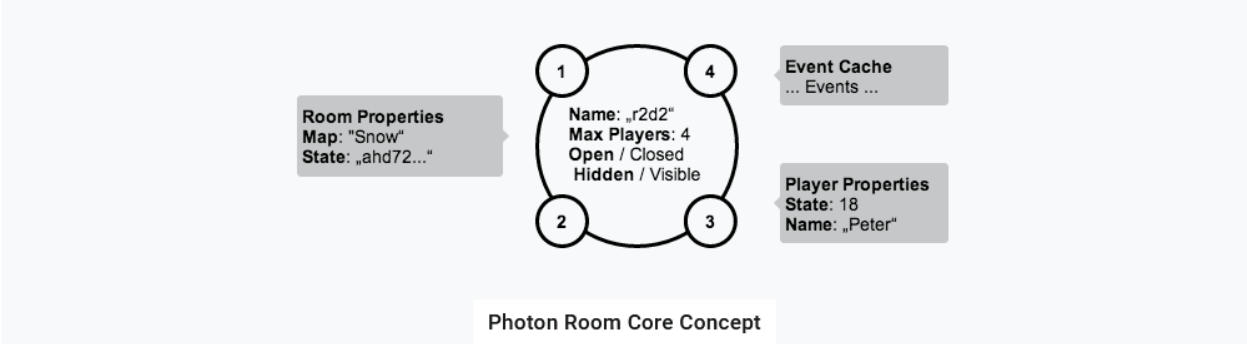


Figure 24.Αρχιτεκτονική ενός δωματίου στον Photon Server

Παρακάτω απεικονίζεται η Lobby σκηνή με τα UI elements:

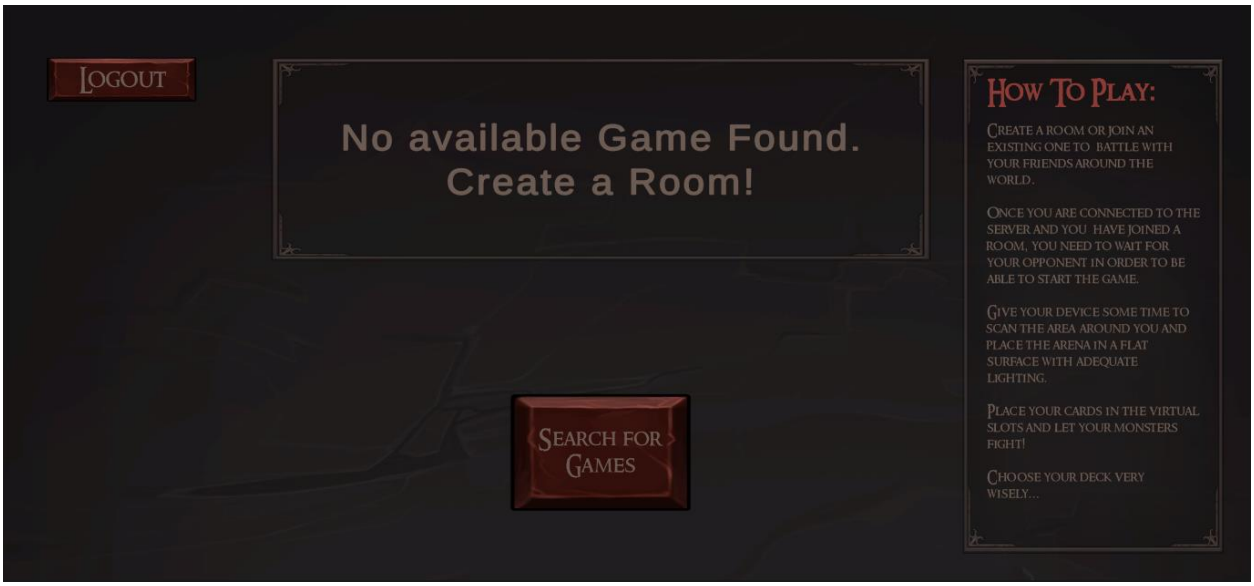


Figure 25. Lobby Screen

Στο κέντρο της σκηνής υπάρχει ένα παράθυρο με κείμενο που ενημερώνει τον παίκτη για την κατάσταση του δωματίου που πρόκειται να εισέλθει. Αν υπάρχει ήδη δημιουργημένο δωμάτιο από κάποιον άλλο παίκτη, τότε πατώντας το κουμπί από κάτω θα συμμετέχει αυτόματα στο δωμάτιο εκείνο, αλλιώς αν δεν υπάρχει κάποιο έτοιμο δωμάτιο, με το πάτημα του ίδιου κουμπιού θα δημιουργήσει ένα δωμάτιο και θα περιμένει μέχρι κάποιος άλλος παίκτης να μπει. Αυτός που δημιουργεί το δωμάτιο λέμε ότι είναι ο ιδιοκτήτης (owner) του δωματίου. Στα αριστερά πάνω υπάρχει επίσης ένα κουμπί με το οποίο γίνεται έξοδος από τον server και ο παίκτης μεταφέρεται στην αρχική σκηνή. Τέλος στα δεξιά υπάρχει ένα πάνελ με οδηγίες προς τον παίκτη αναφορικά με το τι πρόκειται να συναντήσει κατά την διάρκεια του παιχνιδιού. Φυσικά όλες αυτές τις διαδικασίες χρειάζεται να τις εποπτεύει κάποιο script που αναλαμβάνει την δημιουργία των δωματίων, την συμμετοχή του χρήστη σε αυτά, την διαχείριση εξαιρέσεων σε περίπτωση που κάτι δεν πάει καλά με την λειτουργία των δωματίων, την λειτουργικότητα των κουμπιών, την ενημέρωση του UI κ.ο.κ. Το script αυτό έχει ονομαστεί “LobbyManager.cs”.

Όπως καταλαβαίνει κανείς από την περιγραφική ονομασία των συναρτήσεων, τα παρακάτω κομμάτια κώδικα εκτελούν τις ενέργειες για την συμμετοχή σε δωμάτιο, την δημιουργία δωματίου και την περίπτωση που δεν βρεθεί ελεύθερο δωμάτιο αντίστοιχα. Οι μέθοδοι αυτές είναι τύπου Callback που κληρονομούνται από την κλάση MonobehaviourPunCallback και στις οποίες έχουμε πρόσβαση μέσω του keyword override.

```
0 references
public void JoinRandomRoom()
{
    infoText.text = "Searching for available rooms...";
    PhotonNetwork.JoinRandomRoom();
}
```

```
void CreateAndJoinRoom()
{
    string randomRoomName = "Room" + Random.Range(0, 10000);

    RoomOptions roomOptions = new RoomOptions();
    roomOptions.MaxPlayers = 2;

    PhotonNetwork.CreateRoom(randomRoomName, roomOptions);
}
```

```
#region Photon Callback Methods
14 references
public override void OnJoinRandomFailed(short returnCode, string message)
{
    Debug.Log(message);
    CreateAndJoinRoom();
}
```

Ο τρόπος που χρησιμοποιούμε για να δημιουργήσουμε το δωμάτιο είναι αρκετά απλός. Απλά δίνουμε ένα τυχαίο όνομα στο δωμάτιο που δημιουργείται σε κάθε συνεδρία και ορίζουμε κάποιες ρυθμίσεις όπως τον μέγιστο αριθμό παικτών που επιτρέπεται να εισέλθουν στο δωμάτιο. Στην δική μας περίπτωση ορίζουμε αυτό τον αριθμό με “2” μιας και το παιχνίδι προορίζεται για δυο παίκτες. Φυσικά θα μπορούσαμε να δημιουργούμε περισσότερα δωμάτια όπου οι χρήστες διαλέγουν σε ποιο θέλουν να μπουν και να κάνουμε τις ρυθμίσεις πιο σύνθετες.

3.3.7. Έναρξη παιχνιδιού

Αφού έχει βρεθεί ή φτιαχτεί κάποιο δωμάτιο και εφόσον ο παίκτης έχει ενημερωθεί κατάλληλα από την εφαρμογή, καλείτε να επιβεβαιώσει ότι είναι έτοιμος να αντιμετωπίσει τον αντίπαλο παίκτη.

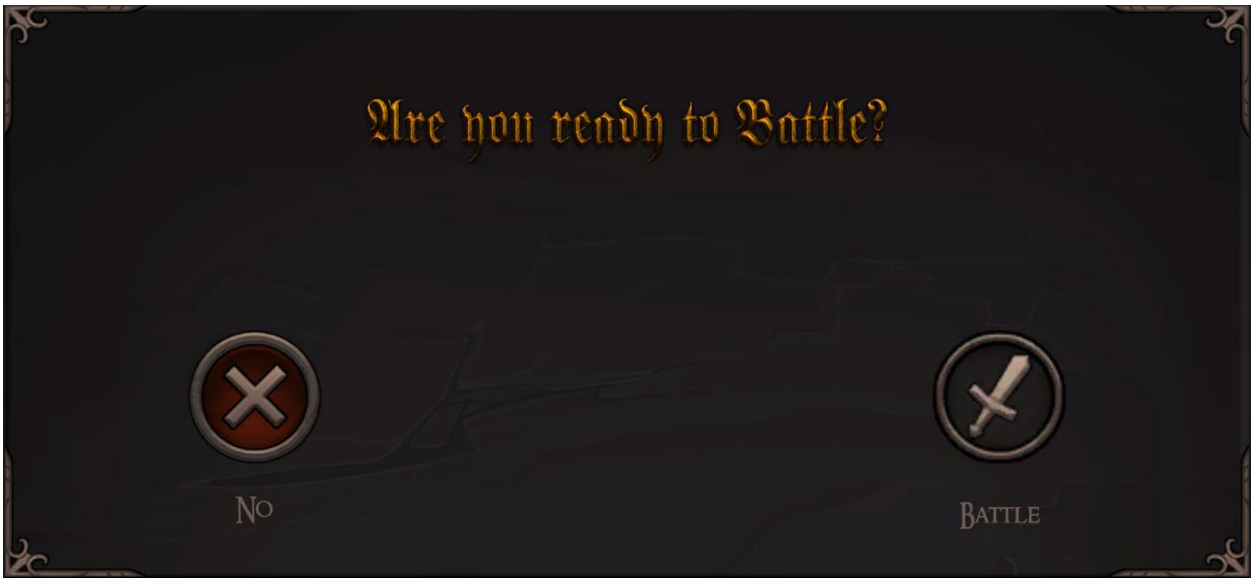


Figure 26. Begin Match Prompt Panel

Αυτό είναι ένα πολύ απλό παράθυρο στο οποίο ο χρήστης είτε επιβεβαιώνει ότι είναι έτοιμος είτε γυρίζει πίσω στο lobby και μπορεί να ψάξει για δωμάτιο από την αρχή όποια στιγμή θελήσει. Με την επιλογή του "BATTLE" φορτώνεται η σκηνή του παιχνιδιού.

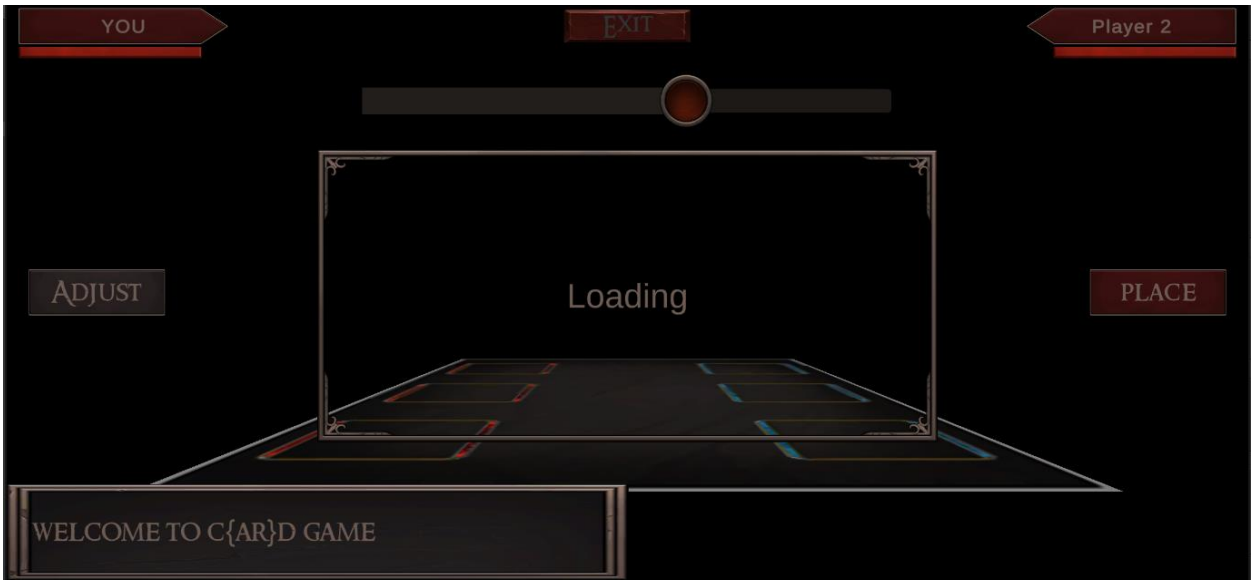


Figure 27. Gameplay Scene

Προτού μπορέσουν οι δύο παίκτες να παίξουν τις κάρτες τους και να πολεμήσουν πρέπει να είναι σίγουροι ότι βρίσκονται στο ίδιο δωμάτιο. Για τον λόγο αυτό το βρίσκεται εκεί ένα παράθυρο που ενημερώνει τον παίκτη που δημιούργησε το δωμάτιο αν δεύτερος

παίκτης έχει μπει σε αυτό. Σε περίπτωση που ο δεύτερος παίκτης εισέλθει στο δωμάτιο το μήνυμα θα τον ενημερώνει ότι το παιχνίδι είναι έτοιμο να ξεκινήσει και μόνο τότε θα κλείνει αυτό το παράθυρο.

Ο τρόπος με τον οποίο μπορεί ν εφαρμογή να γνωρίζει ποιος παίκτης μπήκε πρώτος και ποιος ακολούθησε είναι πάλι με χρήση Callbacks που μας επιτρέπουν να μετράμε πόσοι παίκτες υπάρχουν σε μια δεδομένη στιγμή στον δωμάτιο που συμμετέχουμε. Έτσι, αν την στιγμή που ένας παίκτης μπαίνει στο δωμάτιο ο αριθμός των παικτών σε αυτό το δωμάτιο είναι «1», τότε καταλαβαίνουμε ότι είναι ο μοναδικός παίκτης και άρα ο ιδιοκτήτης του δωματίου. Σε αντίθετη περίπτωση οι παίκτες που ακολουθούν είναι guests.

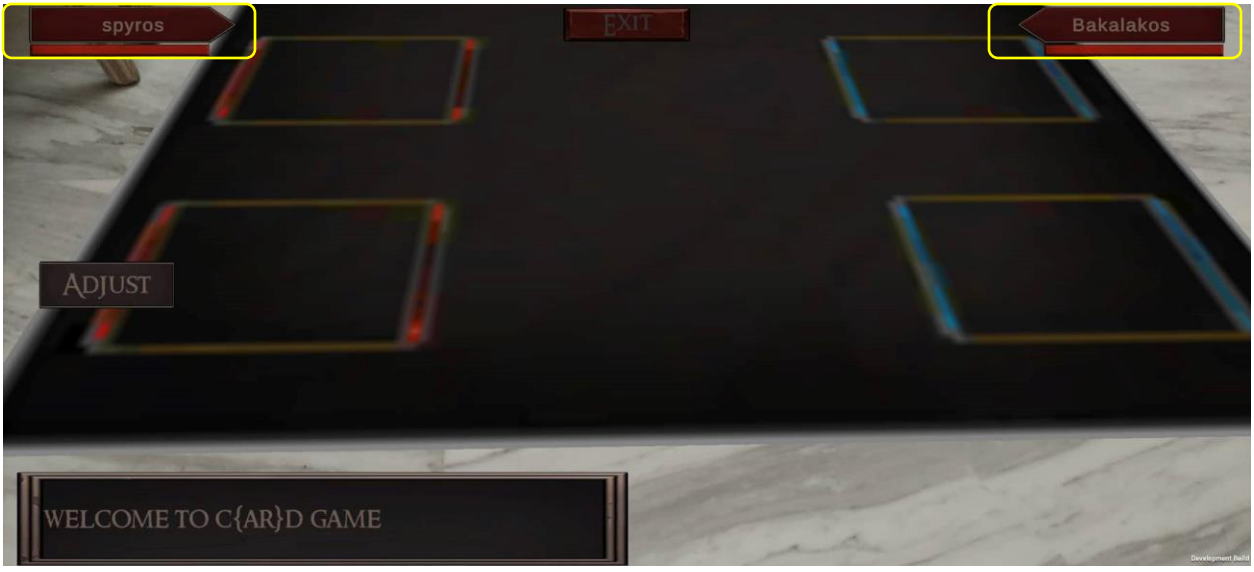
```
void Start()
{
    PlayerNameText.text = PhotonNetwork.LocalPlayer.NickName;

    if (PhotonNetwork.CurrentRoom.PlayerCount == 1)
    {
        infoText.text = "Joined to " + PhotonNetwork.CurrentRoom.Name + ". Waiting for other player...";
    }
    else
    {
        infoText.text = "Joined to " + PhotonNetwork.CurrentRoom.Name;
        StartCoroutine(DeactivatePanel(informationPanel, arInformationPanel, 2.0f));
    }

    PhotonNetwork.NetworkingClient.EventReceived += OnEvent;
}
```

Ακόμη, μπορούμε να έχουμε πρόσβαση στα στοιχεία των παικτών που μπαίνουν σε ένα δωμάτιο με την Callback συνάρτηση “OnPlayerEnteredRoom”.

```
13 references
public override void OnPlayerEnteredRoom(Player newPlayer)
{
    if (newPlayer != null)
    {
        EnemyPlayerNameText.text = newPlayer.NickName;
    }
    else
    {
        EnemyPlayerNameText.text = " ";
    }

    infoText.text = newPlayer.NickName + " joined to " + PhotonNetwork.CurrentRoom.Name + " Player count " + PhotonNetwork.CurrentRoom.PlayerCount;
    StartCoroutine(DeactivatePanel(informationPanel, arInformationPanel, 2.0f));
    //ActivatePlacement();
}
```

Αντίστοιχα, με την είσοδο των παικτών στο δωμάτιο και την φόρτωση της σκηνής παιχνιδιού πρέπει για κάθε παίκτη να τοποθετούνται στα αντίστοιχα πεδία πάνω δεξιά και πάνω αριστερά τα ονόματα των εκάστοτε παικτών. Στο αριστερό πεδίο αναγράφεται πάντα το όνομα του παίκτη που χειρίζεται την συσκευή και δεξιά το όνομα του αντιπάλου του. Ο διαχωρισμός αυτός γίνεται σε ένα script που ονομάζουμε “PlayerSetup.cs” και το οποίο ελέγχει αν για κάθε στιγμιότυπο (instance) του παιχνιδιού, ο παίκτης που το χειρίζεται είναι ο τοπικό χρήστης ή ο απομακρυσμένος. Ανάλογα με την απάντηση καταχωρούνται τα ονόματά τους στα κατάλληλα πεδία.

```
Unity Script | 0 references
public class PlayerSetup : MonoBehaviourPun
{
    public TextMeshProUGUI PlayerNameText;

    // Start is called before the first frame update
    Unity Message | 0 references
    void Start()
    {
        SetPlayerName();
    }

    1 reference
    void SetPlayerName()
    {
        if(PlayerNameText != null)
        {
            if (photonView.IsMine)
            {
                PlayerNameText.text = PhotonNetwork.LocalPlayer.NickName;
            }
            else
            {
                PlayerNameText.text = photonView.Owner.NickName;
            }
        }
    }
}
```



3.3.8. Εμφάνιση χαρακτήρων πάνω από το δίκτυο

Από την στιγμή που μιλάμε για Online παιχνίδι, ξέρουμε πως οτιδήποτε κάνει ένας παίκτης στο δικό του στιγμιότυπο του παιχνιδιού πρέπει να είναι ορατό και στο στιγμιότυπο του απομακρυσμένου χρήστη. Κάθε ενέργεια που εκτελεί ο ένας παίκτης πρέπει να ενημερώνεται στην σκηνή του άλλου και να έχει αντίκτυπο στα στοιχεία του

παιχνιδιού του. Πρώτα απ' όλα όταν παιχτεί μια κάρτα και «καλέσει» έναν χαρακτήρα, αυτός ο χαρακτήρας πρέπει να εμφανιστεί στο περιβάλλον παιχνιδιού όλων των συνδεδεμένων παικτών.

Για την διαχείριση των πόρων παιχνιδιού πάνω από το δίκτυο έχει δημιουργηθεί το "SpawnManager.cs" script στο οποίο υπάρχει η ιδιωτική μέθοδος "SpawnPlayer()" με την οποία γίνεται η εμφάνιση των χαρακτήρων σε όλα τα στιγμιότυπα που τρέχουν για όλους του παίκτες.

```
1 reference
private void SpawnPlayer()
{
    if (photonView.IsMine)
    {
        Vector3 instantiateLocalPosition = spawnPositions[0].position;
        monster = PhotonNetwork.Instantiate("Wolf", instantiateLocalPosition, Quaternion.Euler(0f, 90f, 0f));
    }
    else
    {
        Vector3 instantiateEnemyPosition = spawnPositions[3].position;
        monster = PhotonNetwork.Instantiate("Wolf", instantiateEnemyPosition, Quaternion.Euler(0f, -90, 0f));
    }
}
```

Για να μπορεί ένα αντικείμενο, όπως ο χαρακτήρας από μια κάρτα να εμφανιστεί πάνω από το δίκτυο σε όλους τους παίκτες πρέπει να έχει ένα συστατικό που ονομάζεται Photon View. Το Photon View είναι ένα εξάρτημα που δίνει στο αντικείμενο που συνδέεται δικτυακές ιδιότητες. Με αυτόν τον τρόπο ελέγχοντας για κάθε στιγμιότυπο του παιχνιδιού αν το Photon View Component του κάθε χαρακτήρα ανήκει στο τοπικό παιχνίδι ή στο απομακρυσμένο, μπορούμε να αποφασίσουμε αν ο χαρακτήρας αυτό είναι δικός μας ή του αντιπάλου και να κάνουμε τι αντίστοιχες ενέργειες. Για την εμφάνιση του χαρακτήρα χρειάζεται να δώσουμε ως παράμετρο το όνομα του αντικειμένου που θέλουμε να εμφανίσουμε, την θέση που θέλουμε να βρίσκεται στον χώρο και την περιστροφή του σε σχέση με τον άξονά του. Κάτι που πρέπει να προσέξει ο προγραμματιστής είναι πως ο χαρακτήρας που καλείται με το όνομά του στην συνάρτηση πρέπει να βρίσκεται μέσα σε φάκελο που ονομάζεται "Resources" για να έχει πρόσβαση από την Unity.

Figure 28. Οι χαρακτήρες των δύο παικτών σε αντιστοιχία με το Username τους

3.4 ΣΥΓΧΡΟΝΙΣΜΟΣ ΠΑΙΚΤΩΝ

3.4.1. Βασικές έννοιες συγχρονισμού

Τα multiplayer παιχνίδια έχουν να κάνουν με την ενημέρωση των άλλων παικτών και τη διατήρηση της ίδιας κατάστασης μεταξύ τους. Μέσα στο παιχνίδι ο παίκτης πρέπει να ξέρει ποιοι είναι οι άλλοι παίκτες, τι κάνουν, πού βρίσκονται και πώς φαίνεται ο κόσμος του παιχνιδιού τους. Όπως και προηγουμένως, με την εισαγωγή του Photon View component σε ένα αντικείμενο, αυτόματα το αντικείμενο αποκτά επίγνωση του δικτύου και μπορεί να ενημερώνει τους υπόλοιπου χρήστες για την αλλαγή της θέσης του, της περιστροφής του και άλλων χαρακτηριστικών που ίσως διαθέτει. Το Photon View μπορεί να παρακολουθεί οποιαδήποτε ιδιότητα του αντικειμένου που χρειάζεται να συγχρονιστεί στο δίκτυο, αρκεί να κληρονομεί από την κλάση IPunObservable. Επιλέγουμε στο πεδίο “Synchronization” την επιλογή “Unreliable On Change” ώστε να λαμβάνονται ενημερώσεις με την σειρά που έρχονται και αν το Photon view λάβει την ίδια πληροφορία με την τελευταία θα διακόψει την αποστολή ενημερώσεων προς τους άλλους χρήστες. Αυτό σημαίνει πως κάποιες πληροφορίες μπορεί να χαθούν, όμως η ροή θα συνεχιστεί κανονικά μόλις υπάρξει κάποια αλλαγή στην κατάσταση του περιβάλλοντος παιχνιδιού του χρήστη που στέλνει την ενημέρωση.

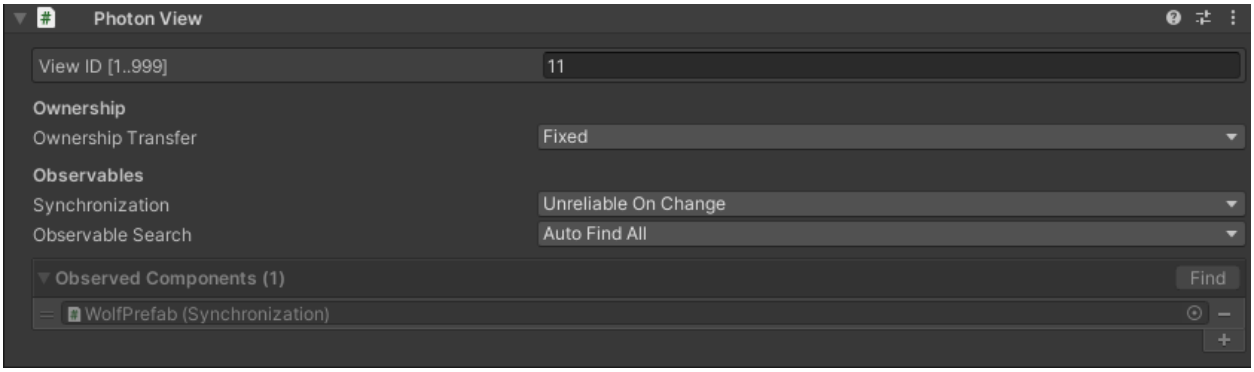


Figure 29. Photon View Component

3.4.2. C# κώδικας συγχρονισμού

Για να παρέχουμε στο Photon View πρόσβαση να εποπτεύει κάποιο script πρέπει να υλοποιήσουμε την διεπαφή (interface) IPunObservable μέσα σε αυτό το script. Μια διεπαφή περιέχει ορισμούς για μια ομάδα σχετικών λειτουργιών που πρέπει να εφαρμοστούν σε μια μη αφηρημένη (abstract) κλάση ή μια δομή (struct). Μια διεπαφή μπορεί να ορίζει στατικές μεθόδους, οι οποίες πρέπει να έχουν υλοποίηση. Η διεπαφή IPunObservable καθορίζει τη μέθοδο OnPhotonSerializeView για να διευκολύνει τη σωστή εφαρμογή της σε παρατηρήσιμα σενάρια. Στην περίπτωση μας θέλουμε να παρατηρούμε την μετατίποση και την περιστροφή του αντικειμένου που ενημερώνει την κατάστασή του και το παιχνίδι να εξελίσσεται ανάλογα με το αν αυτό το αντικείμενο ανήκει στον ιδιοκτήτη του στιγμιότυπου του παιχνιδιού ή οχι. Οι μέθοδοι που πραγματοποιούν αυτόν τον έλεγχο είναι οι “stream.IsWriting” και “stream.IsReading”. Η ιδιότητα IsWriting θα ισχύει εάν αυτός ο παίκτης είναι ο κάτοχος του PhotonView και επομένως του αντικειμένου παιχνιδιού. Τα δεδομένα στη ροή αποστέλλονται μέσω του διακομιστή στους άλλους παίκτες στο δωμάτιο. Στην πλευρά του απομακρυσμένου χρήστη, το IsWriting είναι ψευδές και τα δεδομένα πρέπει να διαβαστούν.

```
public void OnPhotonSerializeView(PhotonStream stream, PhotonMessageInfo info)
{
    if (stream.IsWriting)
    {
        stream.SendNext(rigidbody.position - playBoard.transform.position);
        stream.SendNext(rigidbody.rotation);

        if (synchronizeVelocity)
        {
            stream.SendNext(rigidbody.velocity);
        }

        if (synchronizeAngularVelocity)
        {
            stream.SendNext(rigidbody.angularVelocity);
        }
    }
    else
    {
        networkedPosition = (Vector3)stream.ReceiveNext() + playBoard.transform.position;
        networkedRotation = (Quaternion)stream.ReceiveNext();
```

3.4.3. Απόσβεση απώλειας (Lag Compensation)

Όταν πρόκειται για multiplayer παιχνίδι όπου συμμετέχουν πολλοί απομακρυσμένοι χρήστες, είναι σίγουρο πως κάποιοι θα έχουν γρηγορότερη σύνδεση στο internet από ότι άλλοι. Για τον λόγο αυτό ένα μήνυμα ή μια αλλαγή κατάστασης που στέλνει ένας παίκτης σε άλλον μπορεί να αργήσει να ληφθεί από τον παίκτη με αργή σύνδεση, έτσι η κατάσταση στο περιβάλλον του δεύτερου παίκτη θα έχει αλλάξει μέχρι να λάβει την ενημέρωση και αυτό πολλές φορές μπορεί να δημιουργεί αρνητικό αντίκτυπο στην εμπειρία των παικτών στο παιχνίδι. Όταν ένας χρήστης έχει χαμηλή ταχύτητα σύνδεσης λέμε ότι έχει μεγάλο ping. Το ping είναι η μονάδα μέτρησης της απώλειας ταχύτητας στο δίκτυο και μετριέται σε miliseconds(ms). Μια συνεδρία που φτάνει από 150ms και πάνω κάνει πολύ δύσκολη την σωστή αντίδραση του παίκτη στα συνεχώς εναλλασσόμενα γεγονότα της σκηνής.

Ευτυχώς για τους προγραμματιστές, το Photon Unity Networking παρέχει έναν αυτοματοποιημένο τρόπο ώστε να μπορούμε να μετρήσουμε την ανταπόκριση του server και την ακριβή στιγμή που η πληροφορία στέλνεται από τον χρήστη. Αφαιρώντας λοιπόν την στιγμή αποστολής του μηνύματος από την ακριβή ώρα του server, η οποία είναι κοινή για όλους, παίρνουμε την καθυστέρηση στο δίκτυο. Από εκεί και πέρα είναι εύκολο να προσαρμόσουμε την θέση και την ταχύτητα του αντικειμένου υπολογίζοντάς τες σε συνάρτηση με την καθυστέρηση (lag).

```
if (isTeleportationEnabled)
{
    if(Vector3.Distance(rigidbody.position, networkedPosition) > teleportIfDistanceGreaterThan)
    {
        rigidbody.position = networkedPosition;
    }
}

if(synchronizeVelocity || synchronizeAngularVelocity)
{
    float lag = Mathf.Abs((float)(PhotonNetwork.Time - info.SentServerTime));

    if (synchronizeVelocity)
    {
        rigidbody.velocity = (Vector3)stream.ReceiveNext();
        networkedPosition += rigidbody.velocity * lag;
        distance = Vector3.Distance(rigidbody.position, networkedPosition);
    }

    if (synchronizeAngularVelocity)
    {
        rigidbody.angularVelocity = (Vector3)stream.ReceiveNext();
        networkedRotation = Quaternion.Euler(rigidbody.angularVelocity * lag) * networkedRotation;
        angle = Quaternion.Angle(rigidbody.rotation, networkedRotation);
    }
}
```

3.4.4. Βελτιστοποίηση εφαρμογής για Android

Μιας και η εφαρμογή αυτή προορίζεται για Android φορητές συσκευές υπάρχουν κάποια πράγματα που πρέπει να προσέξουμε. Η εφαρμογή αυτή είναι μια εφαρμογή επαυξημένης πραγματικότητας, κάτι που επιβαρύνει αρκετά την συσκευή και υπάρχει κίνδυνος υπερθέρμανση ή κατασπατάλησης της μπαταρίας. Πρέπει λοιπόν να ρυθμίσουμε την εφαρμογή κατάλληλα ώστε να τρέχει ομαλά χωρίς να καταναλώνει αυξημένους πόρους. Πρώτα απ' όλα απενεργοποιούμε τον φωτισμό πραγματικού χρόνου. Ο ρεαλιστικός φωτισμός στα ηλεκτρονικά παιχνίδια είναι μια αρκετά βαριά διαδικασία που μπορεί να δυσχεραίνει την απόδοση. Έπειτα, στις ρυθμίσεις ποιότητας θα φροντίσουμε η εφαρμογή να τρέχει στα μεσαίας ποιότητας γραφικά. Τα 3D μοντέλα και τα textures που χρησιμοποιούμε δεν είναι φωτορεαλιστικά ούτε περιέχουν μεγάλο όγκο πληροφορίας ώστε να έχουν απεικόνιση που προσομοιάζει το πραγματικό, επομένως δεν υπάρχει ανάγκη για γραφικά υψηλότατων επιδόσεων. Τέλος απενεργοποιούμε κάποιες λειτουργίες που απορροφούν μεγάλη υπολογιστική ισχύ και μπαταρία όπως το antialiasing, τις αντανακλάσεις, σκιάσεις κ.α.

Figure 30. Project Player Settings

3.5 ΠΡΟΣΘΗΚΗ ΕΠΑΥΞΗΜΕΝΗΣ ΠΡΑΓΜΑΤΙΚΟΤΗΤΑΣ (AUGMENTED REALITY)

3.5.1. Υλοποίηση επαυξημένης πραγματικότητας στην σκηνή παιχνιδιού

Έχοντας τελειώσει με την ανάπτυξη όσον αφορά το δίκτυο, την επικοινωνία και το multiplayer κομμάτι της εφαρμογής έφτασε η ώρα να προσθέσουμε χαρακτήρα επαυξημένης πραγματικότητας στο παιχνίδι μας. Όπως έχει αναφερθεί παραπάνω η υλοποίηση για τα χαρακτηριστικά επαυξημένης πραγματικότητας γίνεται χάρη στο ARFoundation και το ARCore για την ανάπτυξη σε Android συσκευές. Το ARFoundation επιτρέπει στον δημιουργό να δουλεύει ταυτόχρονα με πολλές πλατφόρμες επαυξημένης

πραγματικότητας μέσα στην Unity ώστε η παραγωγή του έργου να είναι ταυτόχρονη για πολλαπλές πλατφόρμες. Πχ η ίδια εφαρμογή αναπτύσσεται παράλληλα και για Android μέσα από την πλατφόρμα ARCore και για Apple μέσα από την ARKit πλατφόρμα. Αυτά τα πακέτα είναι που προσφέρουν όλες τις λειτουργίες εικονικής πραγματικότητας που είναι απαραίτητες για την δημιουργία μιας τέτοιας εφαρμογής.

Υπάρχουν κάποια απαραίτητα στοιχεία που πρέπει να εισάγουμε στην σκηνή παιχνιδιού ώστε να προσάγουμε σε αυτής χαρακτηριστικά AR εφαρμογής. Με την προσθήκη των πακέτων που έχουν αναφερθεί σε προηγούμενο κεφάλαιο από την καρτέλα Package Manger, δημιουργείται ένας φάκελος με όνομα "XR" στο dropdown menu της καρτέλας "GameObject". Εκεί επιλέγουμε τα components που χρειάζονται για την υλοποίηση της επαυξημένης πραγματικότητας.

- **AR Session**: Μια AR σκηνή θα πρέπει να περιλαμβάνει οπωσδήποτε ένα AR Session. Το AR Session ελέγχει τον κύκλο ζωής μιας εμπειρίας επαυξημένης πραγματικότητας, ενεργοποιώντας και απενεργοποιώντας τα AR χαρακτηριστικά στην πλατφόρμα.
- **AR Session Origin**: Ο σκοπός του AR Session Origin είναι να μετατρέψει τα ανιχνεύσιμα χαρακτηριστικά (όπως επίπεδες επιφάνειες και σημεία χαρακτηριστικών) στην τελική τους θέση, προσανατολισμό και κλίμακα στη σκηνή της Unity. Το AR Session Origin εκτελεί τον κατάλληλο μετασχηματισμό των δεδομένων του χώρου όπως τον αντιλαμβάνεται η συσκευή σε κλίμακα σχετική με τον εικονικό χώρο μέσα στην Unity.
- **AR Camera Manager**: Το ARCameraManager ενεργοποιεί λειτουργίες της κάμερας, όπως υφές που αντιπροσωπεύουν τη ροή βίντεο και ελέγχει τις λειτουργίες εκτίμησης φωτός.
- **Tracked Pose Driver**: Η κάμερα θα πρέπει επίσης να έχει ένα στοιχείο Tracked Pose Driver, το οποίο θα προσαρμόζει την τοπική θέση και την περιστροφή της κάμερας σύμφωνα με τις πληροφορίες παρακολούθησης της συσκευής. Αυτή η ρύθμιση επιτρέπει στον τοπικό χώρο της κάμερας να ταιριάζει με τον "χώρο συνεδρίας" σε λειτουργία επαυξημένης πραγματικότητας.
- **AR Input Manager**: Αυτό το στοιχείο είναι απαραίτητο για την ενεργοποίηση της παρακολούθησης περιβάλλοντος κόσμου. Χωρίς αυτό, το Tracked Pose Driver που αναφέρεται παραπάνω δεν θα μπορεί να έχει άποψη της συσκευής.
- **AR Raycast Manager**: Το raycasting επιτρέπει στον προγραμματιστή να προσδιορίσει πού τέμνεται μια ακτίνα (που ορίζεται από την αρχή και την κατεύθυνσή της) με μια ανιχνεύσιμη επιφάνεια.

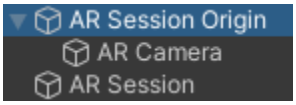


Figure 31. ARSession & ARSessionOrigin Components

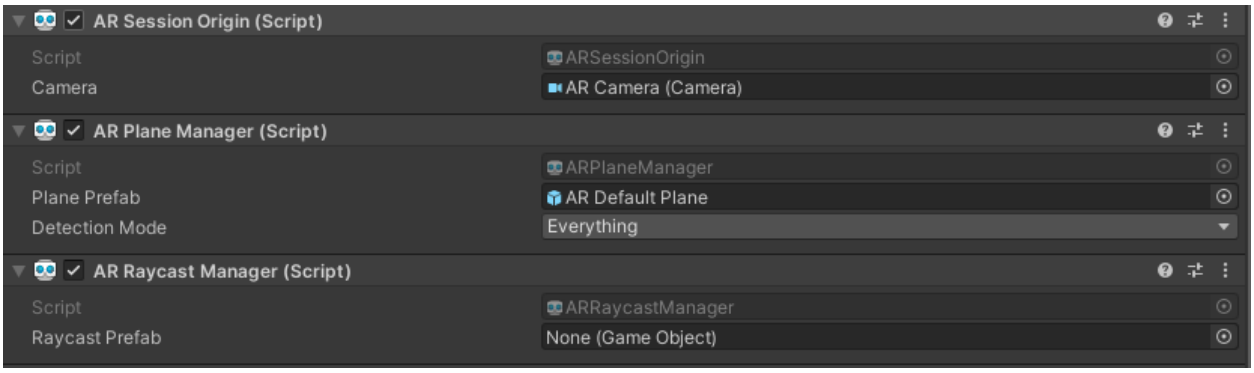


Figure 32. AR Session Origin Component

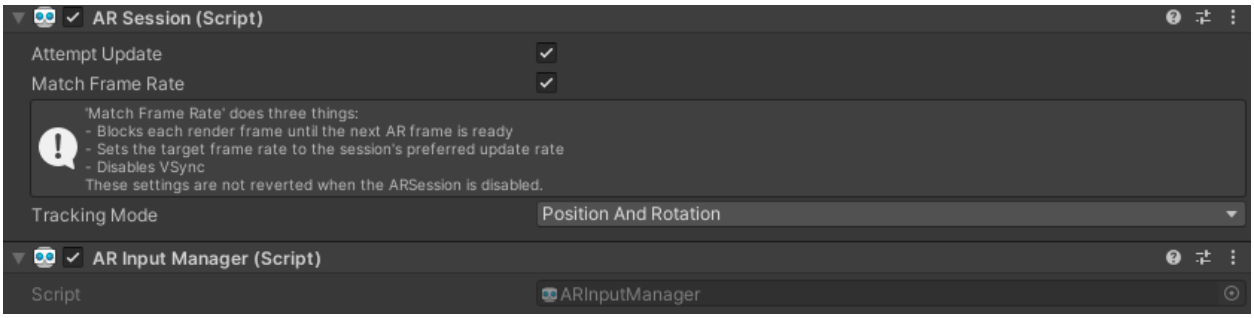


Figure 33. AR Session Component

3.5.2. Εντοπισμός επιφανειών

Στο ARFoundation ένα "trackable" μπορεί να είναι οτιδήποτε μπορεί να ανιχνευτεί και να παρατηρηθεί στον πραγματικό κόσμο. Μπορεί να είναι πρόσωπα, εικόνες, επιφάνειες ή ακόμα και τρισδιάστατα αντικείμενα. Ο εντοπισμός και η παρακολούθηση αυτών των ανιχνεύσιμων στοιχείων στο περιβάλλον γίνεται μέσω των αντίστοιχων managers που αποτελούν δομικό στοιχείο του AR Session Origin για να τοποθετήσουν τα εντοπισμένα ανιχνεύσιμα στοιχεία στην κατάλληλη θέση μέσα την σκηνή επαυξημένης πραγματικότητας.

Στο παρόν στάδιο ανάπτυξης της εφαρμογής χρειάζεται να διαχειριστούμε την ανίχνευση οριζόντιων και κάθετων επίπεδων επιφανειών στον περιβάλλοντα χώρο, θα πραγματοποιήσουμε δηλαδή plane detection. Για αυτό το λόγο προσθέτουμε στο AR Session Origin το AR Plane Manager component όπως φαίνεται παραπάνω. Με αυτή την διαδικασία, όποτε η συσκευή αναγνωρίσει κάποια επίπεδη επιφάνεια στον χώρο θα προβάλλει ένα εικονικό περίβλημα τοποθετημένο παράλληλα στην επιφάνεια δημιουργώντας ένα περίγραμμα, σηματοδοτώντας την περιοχή ανάγνωση επιφανειών από την εφαρμογή.

Οι επιφάνειες αυτές εξυπηρετούν ώστε πάνω σε αυτές να τοποθετηθεί η εικονική αρένα παιχνιδιού στην οποία θα ζωντανεύουν οι χαρακτήρες αλλά και να έχει επίγνωση ο χρήστης του χώρου που διατίθεται για την ανεμπόδιστη κίνηση των στοιχείων του

παιχνιδιού. Με την αναγνώριση μιας επίπεδης επιφάνειας από την συσκευή, δίνεται η επιλογή στον παίκτη να βάλει εκεί την αρένα και μόνο τότε μπορεί να ξεκινήσει η μάχη. Σε περίπτωση που η αρένα δεν έχει τοποθετηθεί σωστά σε σταθερό σημείο του χώρου υπάρχει κίνδυνος να επηρεαστούν η θέση, η κίνηση και η κλίμακα των στοιχείων κατά την διάρκεια του παιχνιδιού.

Figure 34. Plane Detection on Flat Surface

Βλέπουμε ότι η συσκευή έχει εντοπίσει το επίπεδο δάπεδο και έχει εφαρμόσει σε αυτό μια κίτρινη ημιδιαφανή επιφάνεια στην οποία πάνω έχει τοποθετηθεί η αρένα. Επειδή τόσο το plane όσο και η αρένα είναι και τα δύο επίπεδες επιφάνειες σχεδόν ίδιου πάχους παρατηρούμε ότι την στιγμή της τοποθέτησης της αρένας τα δύο αντικείμενα επικαλύπτονται. Μόλις ο χρήστης τοποθετήσει την αρένα, η κίτρινη επιφάνεια εξαφανίζεται και η αρένα μένει παραμένει σταθερή σε αυτό το σημείο ανεξάρτητα με την κίνηση του χρήστη στον κόσμο.

3.5.3. Τοποθέτηση αρένας στον χώρο

Η γενικότερη ιδέα είναι πως η AR σκηνή ξεκινάει με την εικόνα από την κάμερα της συσκευής η οποία ξεκινάει άμεσα την αναζήτηση επιφανειών. Μόλις βρεθεί επιφάνεια εμφανίζεται στην μέση της οθόνης η αρένα μάχης με όλες της ιδιότητες επαυξημένης πραγματικότητας και ο χρήστης πρέπει να αποφασίσει σε ποιο σημείο να την τοποθετήσει. Πρέπει όμως το αντικείμενο να είναι ευέλικτο, να προσαρμόζεται στην αλλαγή της οπτικής της κάμερα και όχι να ακολουθεί την κατεύθυνση της συσκευής παραμένοντας στο κέντρο της σκηνής. Η διαδικασία αυτή διευκολύνεται με την βοήθεια ενός άλλου στοιχείου στο οποίο αναφερθήκαμε παραπάνω, το AR Raycast Manager. Ο τρόπος που λειτουργεί το συγκεκριμένο component είναι να στέλνει μια ακτίνα (ray) από το κέντρο της σκηνής σε κάθε ανιχνεύσιμο στοιχείο το οποίο αναγνωρίζει. Όταν στέλνεται μια ακτίνα προς την

ανιχνευθείσα επιφάνεια θα τέμνεται με αυτή και ως αποτέλεσμα θα επιστρέφει πίσω ένα αντικείμενο τύπου “hit”, που σημαίνει ότι βρέθηκε αντικείμενο στην πορεία της ακτίνας. Τα δεδομένα αυτού του αντικειμένου όπως η θέση του και η περιστροφή του γίνονται ως εκτούτου γνωστά στο στοιχείο που διαχειρίζεται το αντικείμενο που έστειλε την ακτίνα, στην περίπτωσή μας το AR Session Origin που είναι υπεύθυνο για την τοποθέτηση της αρένας, και άρα μπορούν να χρησιμοποιηθούν ως συντεταγμένες για την τοποθέτηση της αρένας σε οριζόντια επιφάνεια στον πραγματικό χώρο. Όλα αυτά βέβαια πρέπει να ελέγχονται από ένα C# script το οποίο παίρνει ως παραμέτρους την κάμερα επαυξημένης πραγματικότητας του AR Session Origin και τον αντικείμενο που πρέπει να τοποθετηθεί. Το script αυτό ονομάζεται “ARPlacementManager.cs”.

```
Unity Script | 2 references
public class ARPlacementManager : MonoBehaviour
{
    ARRaycastManager arRaycastManager;
    static List<ARRaycastHit> raycastHits = new List<ARRaycastHit>();

    public Camera arCamera;
    public GameObject board;

    Unity Message | 0 references
    private void Awake()
    {
        arRaycastManager = GetComponent<ARRaycastManager>();
    }

    // Update is called once per frame
    Unity Message | 0 references
    void Update()
    {
        Vector3 screenCenter = new Vector3(Screen.width / 2, Screen.height / 2);
        Ray ray = arCamera.ScreenPointToRay(screenCenter); //The ray comes to the projected planes from the center of the screen

        //A ray is sent and intersects with the detected plane
        if(arRaycastManager.Raycast(ray, raycastHits, TrackableType.PlaneWithinPolygon))
        {
            //Get the pose of the hit so the object can be placed in the real space position
            Pose hitPose = raycastHits[0].pose;

            Vector3 positionToBePlaced = hitPose.position;

            board.transform.position = positionToBePlaced;
        }
    }
}
```

Παρατηρούμε όμως ότι ακόμα και μετά την τοποθέτηση της αρένας στον χώρο, η εφαρμογή συνεχίζει να ανιχνεύει τις επίπεδες επιφάνειες. Χρειαζόμαστε ένας τρόπο ώστε ο προγραμματιστής να μπορεί να σταματάει ορισμένες λειτουργίες όταν αυτές εξυπηρετήσουν τον σκοπό τους. Επίσης, αφού έχει γίνει η επιλογή της επιθυμητής θέσης από τους παίκτες δεν υπάρχει πλέον ανάγκη για την αποστολή ακτινών (raycasting) από την εφαρμογή. Ο κώδικας που επιτελεί τις διαδικασίες αυτές φιλοξενείται στο “ARPacementControler.cs” script. Την δυνατότητα χειρισμού των εν λόγω λειτουργιών δίνουν στον χρήστη τα κουμπιά “PLACE” και “ADJUST” τα οποία αντίστοιχα αποθηκεύουν την στιγμιαία θέση της αρένας και την τοποθετούν στο σημείο απενεργοποιώντας τις λειτουργίες “raycast” και “plane detection” ή τις ενεργοποιούν ξανά για εκ νέου επιλογή θέσης της αρένας από τον χρήστη. Το script αυτό πρέπει επίσης να προστεθεί μαζί με το “ARPlacementManager” στο AR Session Origin. Επιπρόσθετα, το script αυτό είναι υπεύθυνο για την ανανέωση του ενημερωτικού κειμένου στο UI της εφαρμογής σχετικά με την αλλαγή κατάστασης που περιβάλλοντα χώρου. Όταν είναι όλα

έτοιμα να ξεκινήσει το παιχνίδι, το panel αυτό απενεργοποιείται. Η τακτική ενημέρωση των χρηστών για οποιαδήποτε εξέλιξη συμβαίνει μέσα στο παιχνίδι κάνει αυτομάτως την εμπειρία πιο ξεκούραστη και αποτρέπει από οποιαδήποτε λανθασμένη κίνηση.

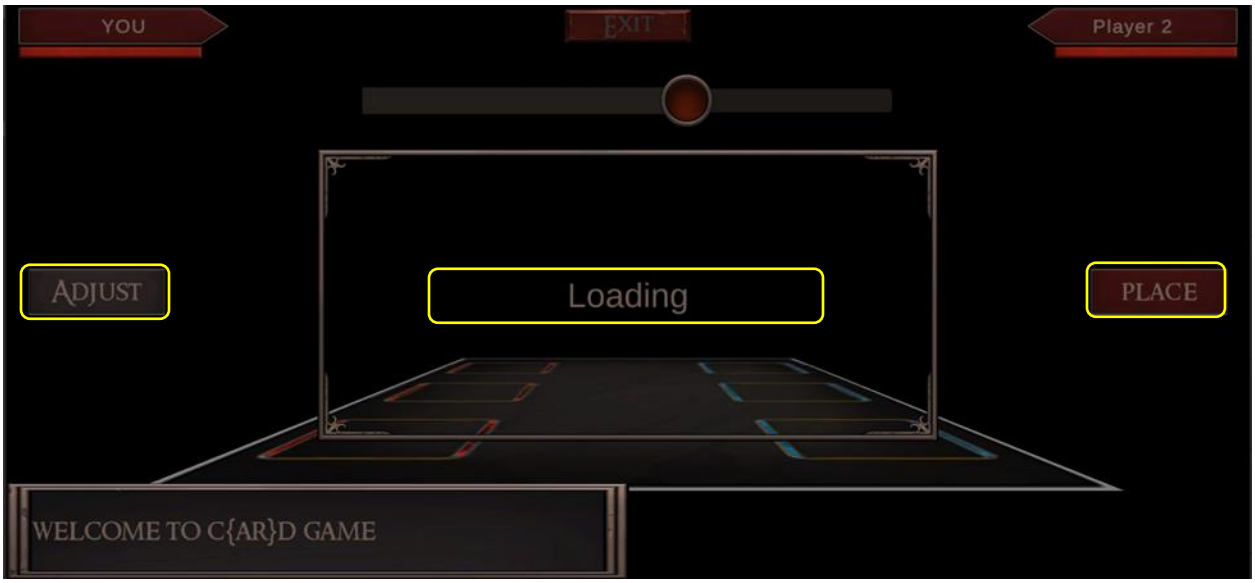


Figure 35. Adjust and Place Buttons & Gameplay Info Panel

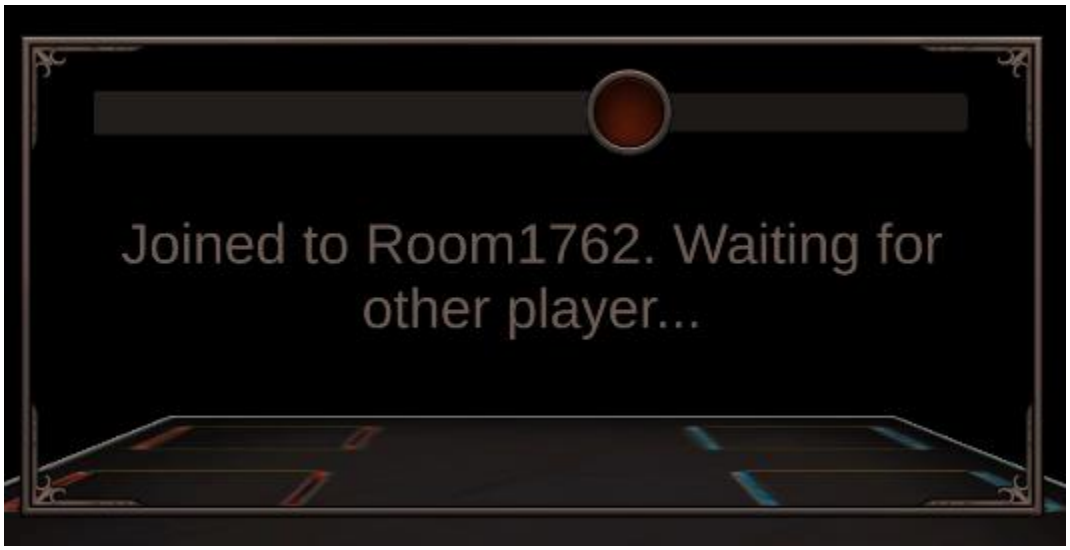
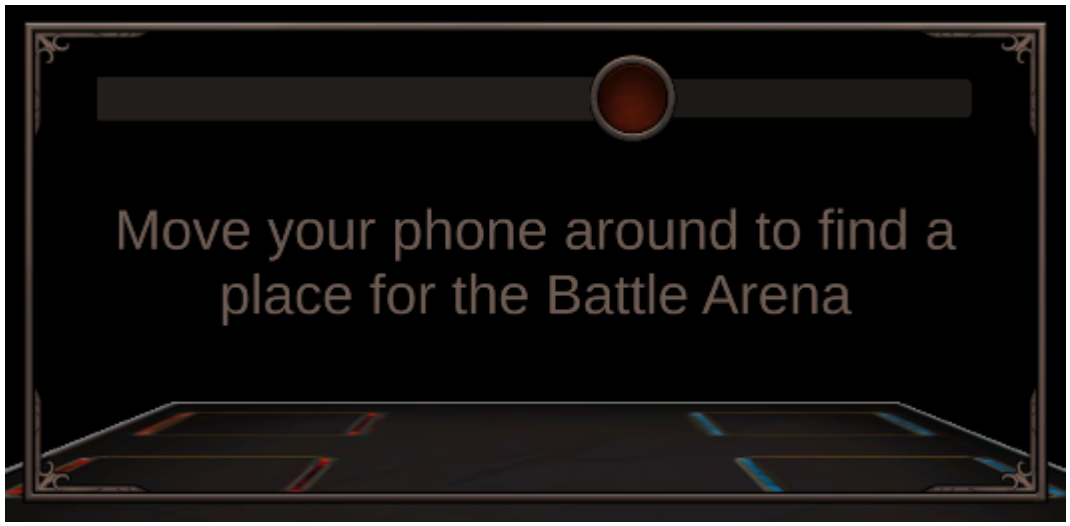


Figure 36. Message on info panel after creating room

Figure 37. Message on info panel after second player has joined



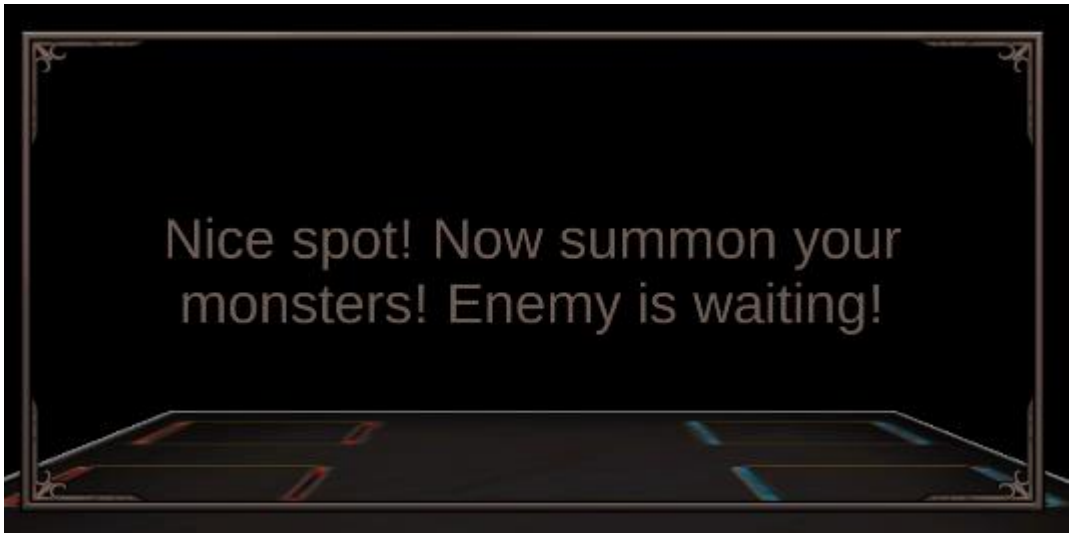


Figure 38. Messages on info panel regarding the arena placement

```
0 references
public void DisablePlaneDetection()
{
    arPlaneManager.enabled = false;
    arPlacementManager.enabled = false;
    CastPlanes(false);

    placeButton.SetActive(false);
    adjustButton.SetActive(true);
    scaleSlider.SetActive(false);

    arInfoText.text = "Nice spot! Now summon your monsters! Enemy is waiting!";
    StartCoroutine(DeactivateARPanel(arInformationPanel, 2.0f));
}

0 references
public void EnablePlaneDetection()
{
    arPlaneManager.enabled = true;
    arPlacementManager.enabled = true;
    CastPlanes(true);

    placeButton.SetActive(true);
    adjustButton.SetActive(false);
    scaleSlider.SetActive(true);

    arInformationPanel.SetActive(true);
    arInfoText.text = "Very well...find a better place then. HURRY!";
}

2 references
private void CastPlanes(bool value)
{
    foreach (var plane in arPlaneManager.trackables)
    {
        plane.gameObject.SetActive(value);
    }
}
```

3.5.4. Κλίμακα στην AR σκηνή

Η αλλαγή κλίμακας είναι ένα πάρα πολύ σημαντικό συστατικό που δεν πρέπει να λείπει από μια AR εφαρμογή. Ας πούμε ότι θέλουμε να τοποθετήσουμε την αρένα μάχης μας σε ένα σημείο που είναι πολύ μικρό όπως ένα τραπεζάκι, ή σε ένα πολύ μεγάλο χώρο όπως το έδαφος σε ένα πάρκο. Το αντικείμενο που εμφανίζεται στην οθόνη έχει πάντα τις αρχικές του διαστάσεις, θα θέλαμε όμως να μπορούσαμε να διαμορφώσουμε αυτές τις διαστάσεις σε περίπτωση που αυτό είναι πολύ μεγάλο για τον εκάστοτε χώρο που διαθέτουμε ή τόσο μικρό που με τα βίας φαίνεται, χωρίς όμως να αλλοιωθούν οι φυσικές του ιδιότητές όπως η μάζα.

Το σίγουρο είναι πως δεν είναι εφικτό να αλλάξει μόνο το μέγεθος της αρένας γιατί αυτό προϋποθέτει να αλλάξει η κλίμακα σε όλα τα αντικείμενα που αλληλοεπιδρούν με αυτή κατά την διάρκεια όλης της συνεδρίας. Ευτυχώς υπάρχει ένας πιο εύκολος τρόπος να γίνει αυτό και αυτός είναι να αλλάξει το μέγεθος του AR Session Origin component. Αυτό θα έχει ως αποτέλεσμα την αλλαγή κλίμακας ολόκληρης της σκηνής κάνοντας έτσι την εμπειρία αρκετά πιο αποκριτική. Για την αλλαγή κλίμακας του AR Session Origin απλά αλλάζουμε την τιμή στο πεδίο μετασχηματισμού του, αυτό έχει ως αποτέλεσμα την κλιμάκωση όλων των δεδομένων που προέρχονται από τη συσκευή, συμπεριλαμβανομένης της θέσης της κάμερας και τυχόν εντοπισμένων ανιχνεύσιμων στοιχείων. Αυτό που πρέπει να προσέξει εδώ ο προγραμματιστής είναι πως το μέγεθος του AR Session Origin είναι αντιστρόφως ανάλογο του μεγέθους του περιεχομένου επαυξημένης πραγματικότητας. Έτσι, η αλλαγή της κλίμακας κατά 10 θα κάνει το αντικείμενο δέκα φορές μικρότερο ενώ κατά 0.1, δέκα φορές μεγαλύτερο. Με αυτόν τον απλό τρόπο ο οποίος περιγράφεται στο “ScaleController.cs” script τα φυσικά στοιχεία των αντικειμένων δεν θα επηρεαστούν καθόλου. Τέλος, η δυνατότητα αλλαγής της κλίμακας της σκηνής παρέχεται από το slider πάνω από το panel ενημέρωσης χρήστη, το οποίο απενεργοποιείται και αυτό με την τελική τοποθέτηση της αρένας.

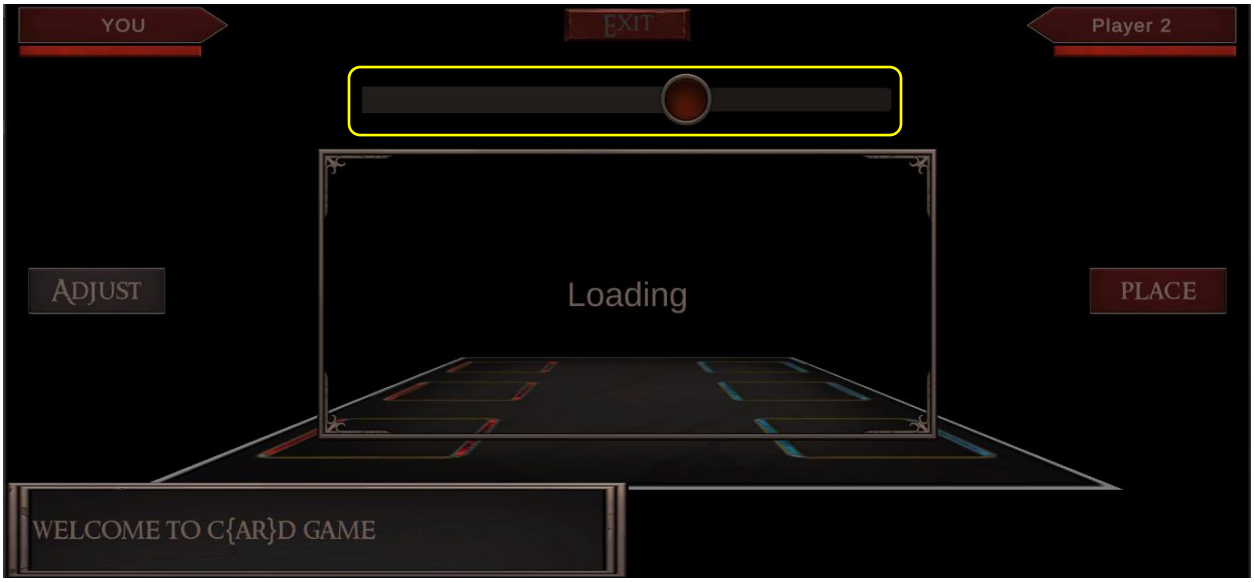


Figure 39. Scale Slider

```
public class ScaleController : MonoBehaviour
{
    ARSessionOrigin arSessionOrigin;

    [Header("UI")]
    public Slider scaleSlider;

    Unity Message | 0 references
    private void Awake()
    {
        arSessionOrigin = GetComponent<ARSessionOrigin>();
    }

    // Start is called before the first frame update
    Unity Message | 0 references
    void Start()
    {
        scaleSlider.onValueChanged.AddListener(OnSliderValueChanged);
    }

    1 reference
    public void OnSliderValueChanged(float value)
    {
        if(scaleSlider != null)
        {
            arSessionOrigin.transform.localScale = Vector3.one / value;
        }
    }
```

3.5.5. Συγχρονισμός θέσης

Ένα εμπόδιο που πρέπει να ξεπεραστεί όταν μιλάμε για multiplayer AR παιχνίδια είναι το πρόβλημα της θέσης της σκηνής στον πραγματικό κόσμο. Για παράδειγμα, όταν η εφαρμογή τρέχει σε έναν στατικό εικονικό κόσμο παραγόμενο από γραφικά δεν υπάρχει κανένα πρόβλημα μιας και η θέση των στοιχείων στην οθόνη είναι η ίδια για όλους. Όταν όμως όλα αυτά τα εικονικά αντικείμενα μεταφέρονται στον πραγματικό κόσμο, οι συνθήκες αλλάζουν. Κάθε σκηνή επαυξημένης πραγματικότητας έχει μια θέση και έναν προσανατολισμό στον χώρο με το σημείο (0,0,0) να είναι η αρχική θέση της συσκευής όταν ξεκινάει η εφαρμογή. Έτσι, αν οποιοσδήποτε από τους παίκτες κινηθεί στον χώρο, ας πούμε, 2 μέτρα μπροστά, η νέα θέση της σκηνής θα είναι (0,0,2) κατά τους άξονες x,y,z, αντίστοιχα. Αυτό δημιουργεί τεράστιο πρόβλημα όταν θέλουμε να στείλουμε τις συντεταγμένες για την εμφάνιση των χαρακτήρων μας στον απομακρυσμένο χρήστη. Στην περίπτωση του τοπικού χρήστη ο χαρακτήρας θα βρίσκεται στην σωστή θέση, όμως για τον απομακρυσμένο χρήστη θα υπάρχει μια απόκλιση δύο μέτρων αφού αυτός δεν έχει ανανεώσει τις συντεταγμένες τις σκηνής του.

Πρέπει λοιπόν να υπάρχει ένας τρόπος ώστε να απαλειφθεί αυτή η διαφορά απόστασης μεταξύ των παικτών ανεξάρτητα από την μετατόπισή τους στον χώρο. Αυτό

που θέλουμε να πετύχουμε στην πραγματικότητα είναι να ενημερώσουμε του υπόλοιπους παίκτες για την αλλαγή στην τοπική σκηνή ώστε να προσαρμόσουν την σκηνή του κατάλληλα, αυτόματα φυσικά, μέσω του κώδικα και όχι με άμεση ανάμειξη του χρήστη. Ακριβώς αυτή την διαδικασία προσφέρει το Photon Engine μέσω μεθόδων που ονομάζονται Raise Events. Τα Raise Events δίνουν στον δημιουργό του παιχνιδιού το δικαίωμα ουσιαστικά να δημιουργήσει δικά του γεγονότα μέσα στο παιχνίδι, τα οποία στέλνονται στους υπόλοιπους παίκτες μέσα από το δίκτυο. Το γεγονός (event) που μας απασχολεί είναι η μετατόπιση της σκηνής. Για να υπολογίσουμε το νούμερο αυτό αρκεί να δημιουργήσουμε ένα γεγονός στο οποίο στέλνουμε την θέση του χαρακτήρα που καλούμε στην αρένα πάνω από το δίκτυο αφαιρώντας από αυτή την όποια τυχών αλλαγή στην θέση της ίδιας της αρένας. Επειδή για τον απομακρυσμένο χρήστη δεν έχει καμία σημασία που θα έχουμε τοποθετήσει την αρένα στην τοπική σκηνή του παιχνιδιού μας.

```
GameObject wolfGameObject = Instantiate(wolfPrefab, instantiateLocalPosition, Quaternion.Euler(0f, 90f, 0f));
PhotonView _photonView = wolfGameObject.GetComponent<PhotonView>();

if (PhotonNetwork.AllocateViewID(_photonView))
{
    object[] data = new object[]
    {
        wolfGameObject.transform.position - playBoard.transform.position, wolfGameObject.transform.rotation, _photonView.ViewID, wolfPrefab
    };

    RaiseEventOptions raiseEventOptions = new RaiseEventOptions
    {
        Receivers = ReceiverGroup.Others,
        CachingOption = EventCaching.AddToRoomCache
    };

    SendOptions sendOptions = new SendOptions
    {
        Reliability = true
    };

    //Raise Events
    PhotonNetwork.RaiseEvent((byte)RaiseEventCodes.CharacterSpawnEventCode, data, raiseEventOptions, sendOptions);
}
else
{
    Debug.Log("Failed to allocate viewID");
    Destroy(wolfGameObject);
}
```

Για να στείλουμε ένα Raise Event στους υπόλοιπους παίκτες πρέπει να έχουμε ορίσει σε αυτό έναν μοναδικό κωδικό γεγονότος που δημιουργούμε, τα δεδομένα που θέλουμε να περάσουμε στον απομακρυσμένο χρήστη, επιλογές για το ποιοι θέλουμε να είναι οι παραλήπτες και πόσο αξιόπιστο θέλουμε το γεγονός να είναι, αυτό σημαίνει αν μας ενδιαφέρει να χαθούν δεδομένα ή όχι κατά την μεταφορά τους στο δίκτυο. Τώρα το μόνο που μένει είναι οι απομακρυσμένοι χρήστες να παραλάβουν τα δεδομένα που στάλθηκαν από την Raise Event μέθοδο. Αυτό επιτυγχάνεται με την κλήση της OnEvent, Callback μεθόδου από την πλευρά του δέκτη, η οποία ενεργοποιείται κάθε φορά που δέχεται ένα γεγονός από κάποιον στο ίδιο δωμάτιο. Παρατηρούμε πως από τα δεδομένα που λαμβάνονται δεν αφαιρείται η θέση της αρένας καθώς τώρα μιλάμε για την σκηνή του χρήστη – παραλήπτη του γεγονότος από την οποία δεν επιθυμούμε να αλλάξει η θέση της αρένας μάχης.

```
2 references
void OnEvent(EventData photonEvent)
{
    if(photonEvent.Code == (byte)RaiseEventCodes.CharacterSpawnEventCode)
    {
        object[] data = (object[])photonEvent.CustomData;
        Vector3 receivedPosition = (Vector3)data[0];
        Quaternion receivedRotation = (Quaternion)data[1];
        GameObject recevedPrefab = (GameObject)data[3];

        GameObject character = Instantiate(recevedPrefab, receivedPosition + playBoard.transform.position, receivedRotation);
        PhotonView _photonView = character.GetComponent<PhotonView>();
        _photonView.ViewID = (int)data[2];
    }
}
```

3.6 ΜΗΧΑΝΙΣΜΟΙ ΤΟΥ ΠΑΙΧΝΙΔΙΟΥ

3.6.1 Προεπισκόπηση της μάχης

Το παιχνίδι έχει πάρει πλέον μορφή επαυξημένης πραγματικότητας ενώ παράλληλα διαθέτει και δικτυακή υποστήριξη. Το τελευταίο συστατικό είναι η πραγματοποίηση της μάχης και της αλληλεπίδρασης των παικτών με τους μηχανισμούς του παιχνιδιού (gameplay). Η λογική του του παιχνιδιού είναι βασισμένη σε ένα turn-based σύστημα. Αυτό σημαίνει πρακτικά πως κάθε παίκτης έχει την σειρά του για να παίξει όπως πχ και στο σκάκι ή στα περισσότερα παιχνίδια καρτών. Όταν είναι ο γύρος του ενός παίκτη, ο άλλος δεν μπορεί να κάνει κάποια κίνηση.

Στον γύρο του, κάθε παίκτης μπορεί να καλέσει έναν χαρακτήρα στην αρένα. Κάθε χαρακτήρας αντιπροσωπεύεται από μια κάρτα που ο παίκτης έχει στην κατοχή του. Αυτές οι κάρτες έχουν φυσική υπόσταση, δεν είναι δηλαδή ψηφιακά αντικείμενα στην σκηνή, και η εφαρμογή έχει την ιδιότητα να τις διαβάζει και να αναγνωρίζει τον εκάστοτε χαρακτήρα που κάθε κάρτα προσομοιάζει (Image recognition). Με το που διαβαστεί η κάρτα, ο παίκτης έχει την επιλογή να την εμφανίσει στην αρένα. Τότε έχει ολοκληρώσει την κίνησή του για αυτό τον γύρο και είναι η σειρά του αντιπάλου να παίξει την δική του κάρτα.

Όλοι οι χαρακτήρες έχουν τα δικά τους μοναδικά χαρακτηριστικά τα οποία είναι το όνομά τους, το επίπεδο δυναμικής, πόσο ζημιά κάνει στον αντίπαλο χαρακτήρα, πόσο από την ζωή του μπορεί να ανανεώσει σε έναν γύρο και πόση ζωή έχει, τα οποία αποθηκεύονται σε ένα script ονόματι “Stats.cs”. Κάθε χαρακτήρας μπορεί σε κάθε γύρο να εκτελέσει ανάμεσα σε δύο ενέργειες, να επιτεθεί στον αντίπαλο ή να ανανεώσει την ζωή του ώστε να μην είναι τόσο ευάλωτος στην επίθεση του αντιπάλου. Το παιχνίδι τελειώνει μόλις ένας από τους δύο παίκτες μηδενίσει το υπόλοιπο ζωής που του έχει απομείνει στην μπάρα κάτω από το όνομά του. Ο παίκτης που θα τερματίσει την παρτίδα με υπόλοιπο ζωής μεγαλύτερο του μηδενός αναδεικνύεται νικητής του αγώνα ενώ ο άλλος είναι ο χαμένος.

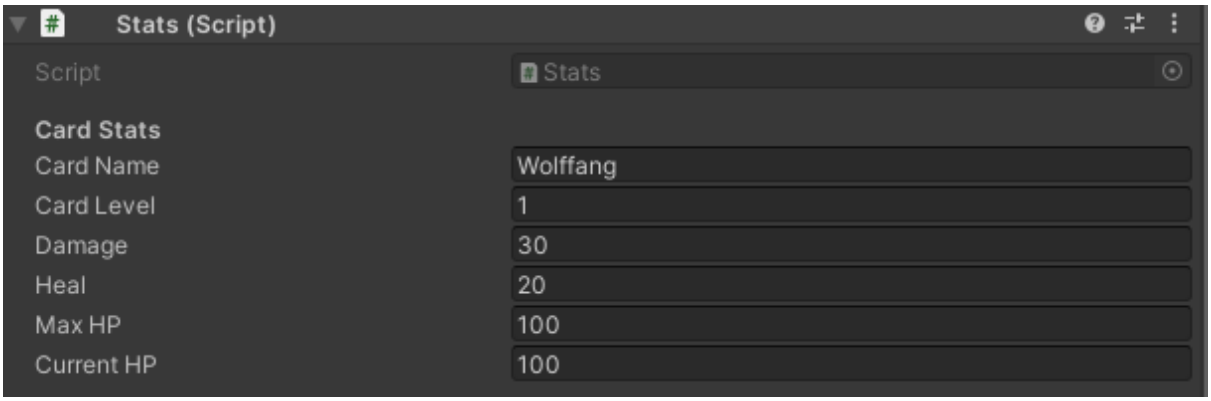


Figure 40. Character's Stats

3.6.2 Παρακολούθηση εικόνων και αντικειμένων (Image Tracking & Object Tracking)

Η παρακολούθηση εικόνων είναι ο τρόπος με τον οποίο η εφαρμογή επαυξημένης πραγματικότητας μπορεί να αναγνωρίζει μια δισδιάστατη εικόνα και να ακολουθεί την θέση και την κατεύθυνσή της στον χώρο. Την ανίχνευση των εικόνων την αναλαμβάνει το AR Tracked Image Manager, το οποίο προστίθεται στο AR Session Origin. Το στοιχείο αυτό προσθέτει τρισδιάστατα αντικείμενα στον χώρο για κάθε εικόνα που ανιχνεύει. Για την αντιστοίχιση αυτή χρειάζεται να γνωρίζει την βιβλιοθήκη με τις εικόνες που θα κάνει την αναζήτηση, τον μέγιστο αριθμό εικόνων που του ζητείται να αναγνωρίσει και το αντικείμενο παιχνιδιού που θα εμφανίσει πάνω στην κάρτα, δηλαδή τον χαρακτήρα που της αντιστοιχεί.

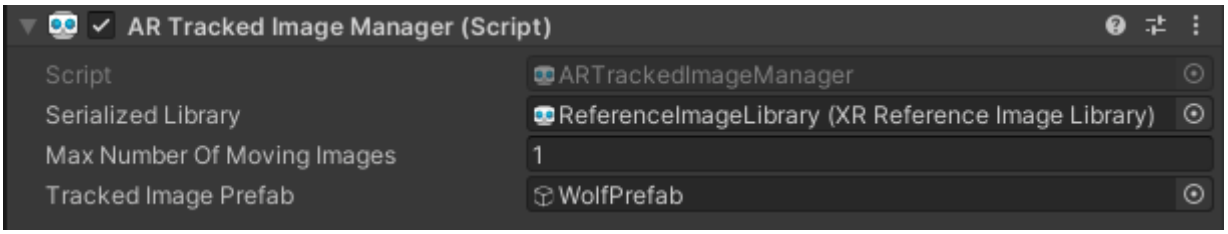


Figure 41. ARTrackedImageManager Component

Όμως, πριν μια εικόνα μπορέσει να αναγνωριστεί, πρέπει το AR Tracked Image Manager να ψάξει για αυτή την εικόνα μέσα σε ένα προκαθορισμένο σύνολο εικόνων που λειτουργούν ως σημεία αναφοράς. Αυτό το σετ εικόνων ονομάζεται βιβλιοθήκη εικόνων αναφοράς (Reference Image Library) και όλες οι εικόνες που πρόκειται να χρησιμοποιηθούν ως ανιχνεύσιμα στοιχεία την AR σκηνή πρέπει να βρίσκονται εκεί μέσα. Στην βιβλιοθήκη αυτή πρέπει να περιλαμβάνονται μαζί με την εικόνα, το όνομά της το οποίο μπορεί να είναι χρήσιμο για την ταύτιση της εικόνας με το αντικείμενο, το φυσικό

μέγεθος που πρέπει να περιμένει η συσκευή να έχει η εικόνα στον πραγματικό κόσμο και μια αναφορά στην υφή (texture) της εικόνας.

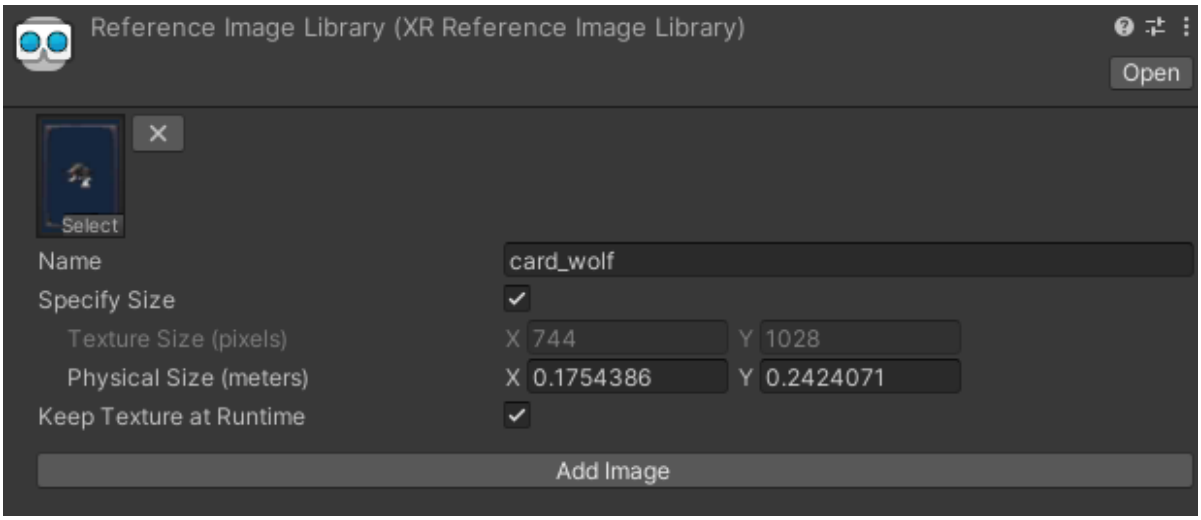


Figure 42. Βιβλιοθήκη Εικόνων – καρτών

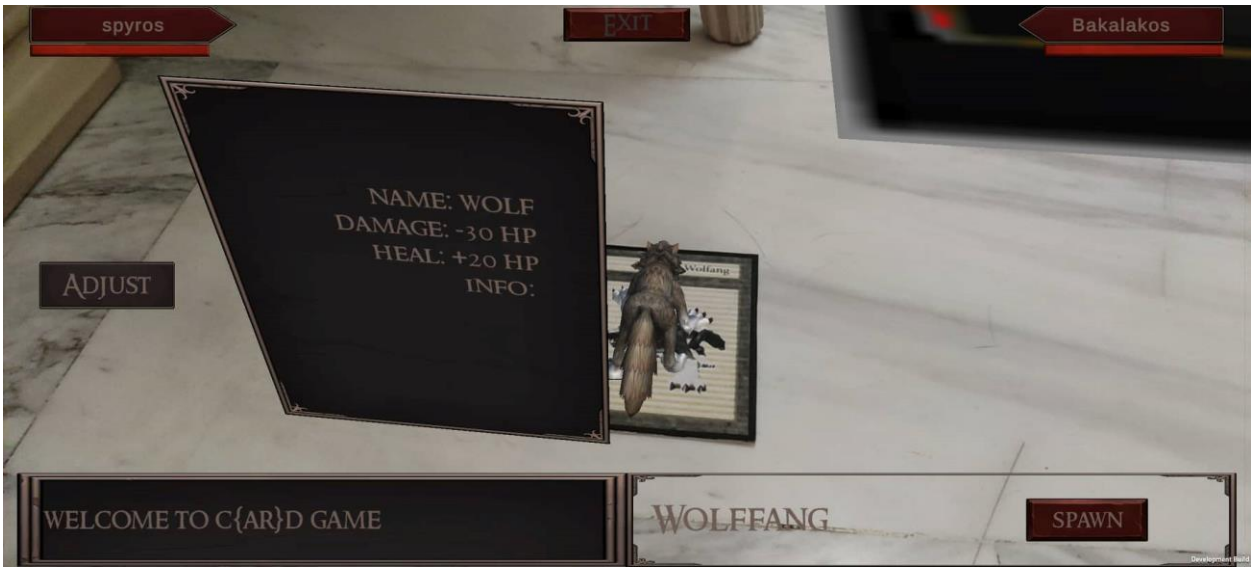


Figure 43. Image recognition on character card

Στην συνέχεια, χρειαζόμαστε κατά την αναγνώριση εικόνας, μαζί με τον χαρακτήρα να εμφανίζεται στην οθόνη του χρήστη και ένα πεδίο με τις πληροφορίες και τις ενέργειες που είναι διαθέσιμες για αυτή την κάρτα και είναι μοναδικές για κάθε μια. Για τον λόγο αυτό θα δημιουργηθεί ένα script το οποίο παρακολουθεί το αντικείμενο – χαρακτήρα και αναγνωρίζοντάς το ξέρει ότι πρέπει να εμφανίσει στην οθόνη το κατάλληλο πάνελ που αντιστοιχεί στις μοναδικές ιδιότητες του χαρακτήρα. Το script ονομάζεται "ARTrackedObjectManager.cs".

Figure 44. Character's Action Panel

```
private void Awake()
{
    arTrackedImageManager = GetComponent<ARTrackedImageManager>();
}

private void Start()
{
    actionCanvas.enabled = false;
}

private void OnEnable()
{
    arTrackedImageManager.trackedImagesChanged += OnTrackedImagesChanged;
}

private void OnDisable()
{
    arTrackedImageManager.trackedImagesChanged -= OnTrackedImagesChanged;
}

void OnTrackedImagesChanged(ARTrackedImagesChangedEventArgs eventArgs)
{
    foreach(ARTrackedImage trackedImage in eventArgs.added)
    {
        actionCanvas.enabled = true;
    }

    foreach(ARTrackedImage trackedImage in eventArgs.updated)
    {
        actionCanvas.enabled = true;
    }

    foreach(ARTrackedImage trackedImage in eventArgs.removed)
    {
        actionCanvas.enabled = false;
    }
}
```

Τέλος, το παιχνίδι αποτελείται από πολλές και διαφορετικές κάρτες οι οποίες πρέπει να αντιστοιχούν στους σωστούς χαρακτήρες. Έτσι, αντίστοιχα με την βιβλιοθήκη εικόνων που φτιάξαμε προηγουμένως, πρέπει να δημιουργηθεί και μια βιβλιοθήκη ή μια λίστα που θα περιλαμβάνει όλα τα αντικείμενα παιχνιδιού που αποτελούν τους χαρακτήρες. Με την βοήθεια ενός νέου script, του "MultipleImageTrackedManager.cs", γίνεται πολλαπλή αναγνώριση των καρτών. Η συσχέτιση της εικόνας με το αντικείμενο γίνεται βάση του ονόματος, οπότε είναι καίριας σημασίας να φροντίσουμε ώστε το image και το GameObject να έχουν ακριβώς το ίδιο όνομα. Ο τρόπος που δουλεύει το script είναι ο εξής. Με το που αναγνωριστεί μια κάρτα, ο κώδικας έχει πρόσβαση στα στοιχεία της, μαζί και το όνομά που έχουμε δώσει. Έπειτα κάνει αναζήτηση στην λίστα αντικειμένων για αντικείμενο με το ίδιο όνομα. Μόλις το αντικείμενο αυτό βρεθεί εμφανίζεται στην οθόνη στην σωστή θέση και περιστροφή πάνω στην κάρτα. Το script αυτό πρέπει να προστεθεί, όπως και τα προηγούμενα, στο ARSessionOriginComponent.

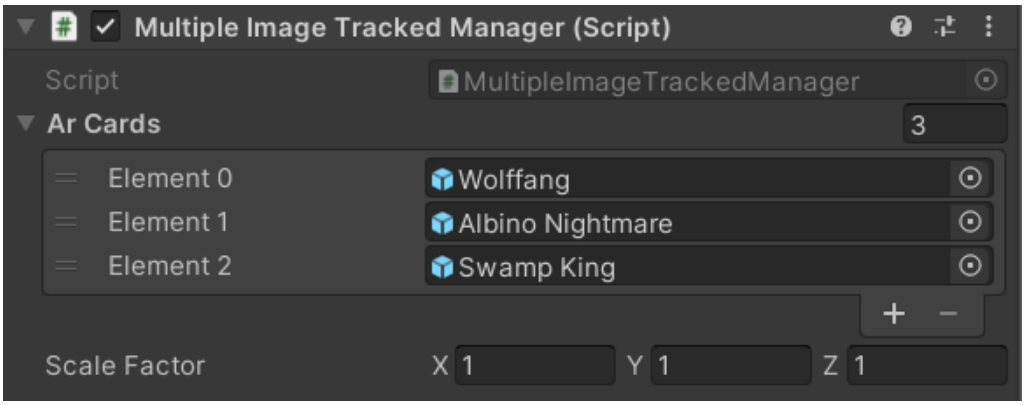


Figure 45. Dictionary αντικειμένων καρτών

```
1 reference
void AssignGameObject(string name, Vector3 newPosition)
{
    if (arCards != null)
    {
        GameObject arObject = arCharacters[name];
        arObject.SetActive(true);
        arObject.transform.position = newPosition;
        arObject.transform.localScale = scaleFactor;

        foreach(GameObject go in arCharacters.Values)
        {
            Debug.Log($"game object in dictionary: {go.name}");
            if(go.name != name)
            {
                go.SetActive(false);
            }
        }
    }
}
```

3.6.3 Let’s Play!

Επιτέλους όλα είναι έτοιμα και μπορούμε να ξεκινήσουμε να δημιουργούμε κάποιους αληθινούς μηχανισμούς μάχης. Κατά την πορεία όλης αυτής της δημιουργίας έχουμε στο μυαλό μας ότι αυτό είναι ένα multiplayer παιχνίδι, άρα πως μπορούμε να εξασφαλίσουμε ότι μια ενέργεια που κάνει ο ένας παίκτης θα φτάσει τελικά στην οθόνη του δεύτερου παίκτη πάνω από το δίκτυο; Το Photon Engine έχει προβλέψει φυσικά για κάτι τέτοιο και για ακόμα μια φορά το Photon View Component θα δώσει την λύση. Όπως ξέρουμε ήδη το Photon View είναι το στοιχείο που μας επιτρέπει να έχουμε πρόσβαση σε δεδομένα ιδιοκτησίας των αντικειμένων παιχνιδιού. Άρα, σε κάθε στιγμή μπορούμε να ελέγξουμε αν το στιγμιότυπο παιχνιδιού ανήκει στον παίκτη που κάνει την κίνηση, πχ: μια επίθεση ή όχι. Φυσικά δεν αρκεί μόνο αυτό καθώς χρειαζόμαστε και έναν τρόπο να περνάμε την πληροφορία από την μια πλευρά της σύνδεσης στην άλλη. Για παράδειγμα, μπορεί να πρέπει να ανανεωθεί η θέση ενός χαρακτήρα και στις δύο συσκευές. Η ανανέωση αυτή της κατάστασης του απομακρυσμένου χρήστη μετά από εντολή του τοπικού χρήστη γίνεται μέσω Remote Procedure Calls (RPCs).

Οι Remote Procedure Calls είναι μέθοδοι που καλούνται στο παιχνίδι όλων των απομακρυσμένων παικτών στο δωμάτιο από τον τοπικό χρήστη. Ουσιαστικά αυτό που κάνουν αυτές οι μέθοδοι είναι να εκτελούν την ίδια συνάρτηση για όλους τους συνδεδεμένους παίκτες ή μόνο τον χρήστη εκείνον που είναι στόχος. Αυτό εξαρτάται από τον προγραμματιστή. Για να ενεργοποιηθούν ιδιότητες για μια μέθοδο να καλείται σε στιγμιότυπο παιχνιδιού που δεν ανήκει στον παίκτη που την κάλεσε πρέπει να “μαρκαριστεί” με την ετικέτα “[PunRPC]” πάνω από την δήλωση της συνάρτησης. Αντίστοιχα, στο σημείο όπου καλείται η συνάρτηση χρειάζεται να καλεστεί μέσω ενός Photon View Component που θα καλεί αυτή την μέθοδο απομακρυσμένης διαδικασίας. Για την εκτέλεσή της το Photon View παίρνει σαν παράμετρο το όνομα της συνάρτησης που καλείτε σε τύπο string, τους παίκτες-στόχους που προορίζεται να τρέξει η συνάρτηση και προαιρετικά όποια παράμετρο υπάρχει σαν παράμετρος στην συνάρτηση που καλείται (π.χ: κάποια τιμή που επιστρέφει).

```
public void OnAttackButtonClicked()
{
    if (state != BattleState.PLAYERTURN)
     return;

    if (!pv.IsMine)
    {
        pv.RequestOwnership();
    }

    StartCoroutine(WaitForOwnershipTransfer());
}

IEnumerator WaitForOwnershipTransfer()
{
    HUDText.text = "...";

    yield return new WaitForSeconds(2f);

    pv.RPC("PlayerAttack", RpcTarget.AllBuffered);
}

[PunRPC]
IEnumerator PlayerAttack()
{
    if (pv.IsMine)
    {
        HUDText.text = "You attacked";
        RemoteHealthbar.fillAmount -= wolfStats.damage / 100;

        state = BattleState.ENEMYTURN;
    }
    else
    {
        HUDText.text = "Enemy is attacking " + wolfStats.cardName;

        yield return new WaitForSeconds(2f);

        wolfStats.currentHP -= wolfStats.damage;
        Localhealthbar.fillAmount -= wolfStats.damage / 100;
        HUDText.text = "You have been attacked! HP: " + wolfStats.currentHP;

        state = BattleState.PLAYERTURN;
    }
```

Η παραπάνω εικόνα απεικονίζει ένα κομμάτι κώδικα από το "BattleScript.cs" script το οποίο διαχειρίζεται όλες τις ενέργειες των παικτών στην σκηνή της μάχης. Στο στιγμιότυπο φαίνεται η επίθεση ενός παίκτη αφού έχει παίξει την κάρτα του στην αρένα. Πατώντας το κουμπί της επίθεσης, θα στείλει στον απομακρυσμένο παίκτη την δράση αυτή και εφόσον ο χαρακτήρας του ενός είναι πιο δυνατός από τον άλλο, θα υπερισχύσει ο δυνατότερος και ο δεύτερος παίκτης θα χάσει ένα κομμάτι από την ζωή του. Το BattleScript.cs έχει πρόσβαση στο Stats.cs ώστε να γνωρίζει τα χαρακτηριστικά της κάθε κάρτας. Αυτό είναι απαραίτητο για τον υπολογισμό των πόντων κατά την επίθεση ή την ανανέωση ζωής ή ακόμα και την ενημέρωση προς τον παίκτη για την κατάσταση των χαρακτήρων που έχει στην αρένα.

Figure 46. Battle info panel when player is attacking

Figure 47. Battle info panel when player is healing

Figure 48. Battle info panel displaying player's health

3.6.4 Τέλος Παιχνιδιού

Ο νικητής αποφασίζεται από το σύστημα, βασιζόμενο στο ποιος παίκτης κατάφερε με τις επιθέσεις του να μηδενίσει τους πόντους του αντιπάλου του πρώτος. Οι παίκτες ξεκινάνε το παιχνίδι με τον ίδιο αριθμό πόντων ζωής και η μείωση της μπάρας ζωής εξαρτάται από το πόσο δυνατός είναι ο χαρακτήρας που κάνει την επίθεση στον αντίπαλό του. Εφόσον κάποιος από τους δύο παίκτες φτάσει τους μηδέν πόντους, το παιχνίδι τερματίζεται και στην οθόνη του παίκτη με την μηδενική ζωή εμφανίζεται ένα μήνυμα που τον ενημερώνει ότι έχασε την μάχη. Στην αντίθετη πλευρά του παιχνιδιού, για τον παίκτη που έχει περισσευούμενους πόντους ζωής, το μήνυμα του λέει ότι αναδείχτηκε ο νικητής του παιχνιδιού.

Στην συνέχεια η σκηνή αυτή καταστρέφεται, και οι παίκτες οδηγούνται στην σκηνή τερματισμού. Στην σκηνή αυτή γίνεται η σύνοψη της μάχης μεταξύ των δύο παικτών οι οποίοι ρωτιούνται παρακάτω αν επιθυμούν να παίξουν ξανά. Η επανάληψη της παρτίδας προϋποθέτει την εκ νέου σύνδεση των χρηστών στο δωμάτιο, η οποία γίνεται αυτόματα μόλις επιλέξουν το κουμπί επιβεβαίωσης. Τότε φορτώνεται ξανά η AR σκηνή του παιχνιδιού και η μάχη ξεκινάει από την αρχή. Αν οι παίκτες αποφασίσουν να τερματίσουν την συνεδρία, απορρίπτοντας την επανάληψη της παρτίδας, αποχωρούν από το δωμάτιο και επιστρέφουν στην αρχική οθόνη.

Figure 49. End Match Screen

3.6.5 Προσθήκη οπτικοακουστικών εφέ και animations

Εννοείται πως πολύ μεγάλο κομμάτι στον σχεδιασμό ενός video game είναι η σωστή αξιοποίηση όλων των πόρων του. Σε αυτούς τους πόρους εκτός φυσικά από τις λειτουργίες που απαρτίζουν την ίδια την εκτέλεση του παιχνιδιού ανήκουν εξίσου η οπτική αναπαράσταση των γεγονότων που συμβαίνουν στο περιβάλλον, όπως κάποια αντίδραση των χαρακτήρων την ώρα που επιτίθονται ή τους επιτίθεται ο αντίπαλος ή κάποια οπτικά εφέ στην σκηνή που σηματοδοτούν συγκεκριμένα γεγονότα. Επίσης σημαντικό ρόλο στην αφήγηση του παιχνιδιού παίζει η μουσική και τα ηχητικά εφέ.

Για τον σκοπό αυτό, θα ήταν δόκιμο να προστεθούν στην σκηνή παιχνιδιού επαυξημένης πραγματικότητας ορισμένα animations στους χαρακτήρες, διαφορετικά για την φάση που κάνουν επίθεση, στην στιγμή που πεθαίνουν ή την διάρκεια που είναι σε αναμονή. Ακόμη, ενδιαφέρον δίνει η μουσική επένδυση στο παρασκήνιο αλλά και ξεχωριστοί ήχοι που θα προκαλούνται από γεγονότα όπως το χτύπημα ενός χαρακτήρα. Με αυτόν το τρόπο το παιχνίδι παύει να είναι μονότονο και προκαλεί στον χρήστη ένα παραπάνω ενδιαφέρον καθώς πυροδοτεί περισσότερες από μια αισθήσεις.

Ο animator controller επιτρέπει στον δημιουργό να διαχειριστεί και να διατηρήσει σετ από animations των χαρακτήρων ή άλλων αντικειμένων στην σκηνή. Μέσα στον controllet βρίσκονται αναφορές από τα διαθέσιμα animation για το κάθε αντικείμενο και ο προγραμματιστής μπορεί να δημιουργήσει καταστάσεις κατά τις οποίες πυροδοτείται ή εναλλάσσεται ένα animation. Για την διαχείριση των animations χρησιμοποιείται μια μηχανή καταστάσεων (state machine).

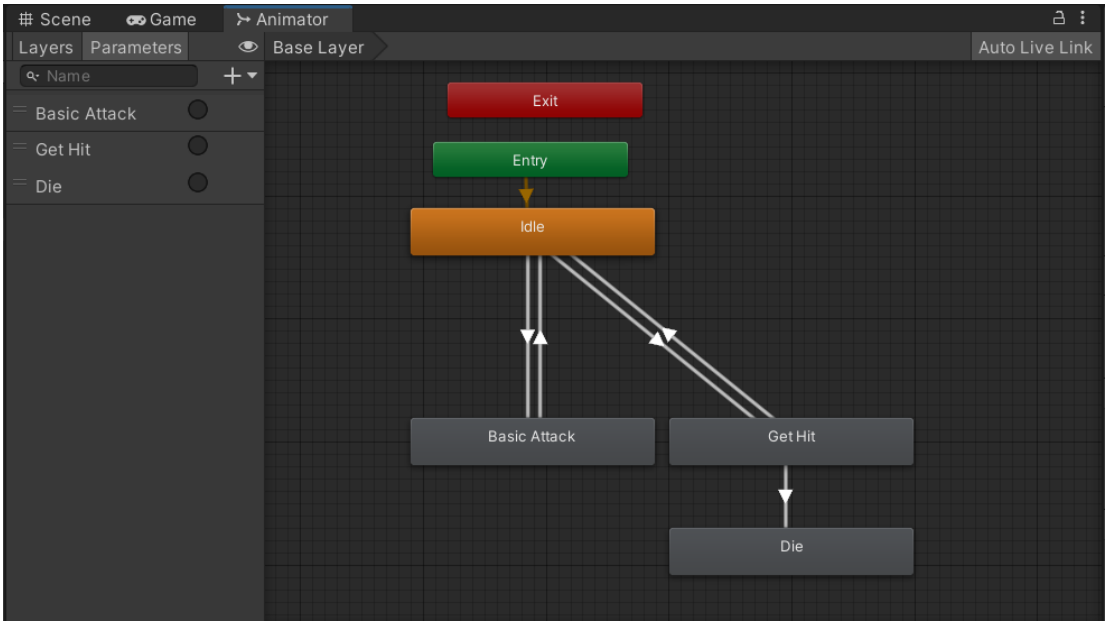


Figure 50. Animator Controller Window

Για την προσθήκη ήχων στην σκηνή είναι απαραίτητα δύο components. Το πρώτο είναι ένας Audio Listener, ο οποίος εφαρμόζεται πάνω στην κύρια κάμερα της σκηνής και παίζει τον ρόλο του μικροφώνου. Ηχογραφεί τους ήχους του περιβάλλοντος και τους αναπαράγει στα ηχεία του χρήστη. Χωρίς Audio Listener η αναπαραγωγή ήχου είναι αδύνατη. Το δεύτερο component είναι ένα Audio Source, η πηγή ήχου δηλαδή που εφαρμόζεται πάνω στο αντικείμενο το οποίο παράγει τον ήχο. Στο παιχνίδι μας έχει εφαρμοστεί μια μουσική παρασκηνίου η οποία αλλάζει ανάλογα την σκηνή αλλά και την κατάσταση του παίκτη, ηχητικά εφέ τα οποία ενεργοποιούνται κατά την επίθεση τον χαρακτήρων και ήχοι που λειτουργούν ως feedback στο πάτημα κουμπιών.

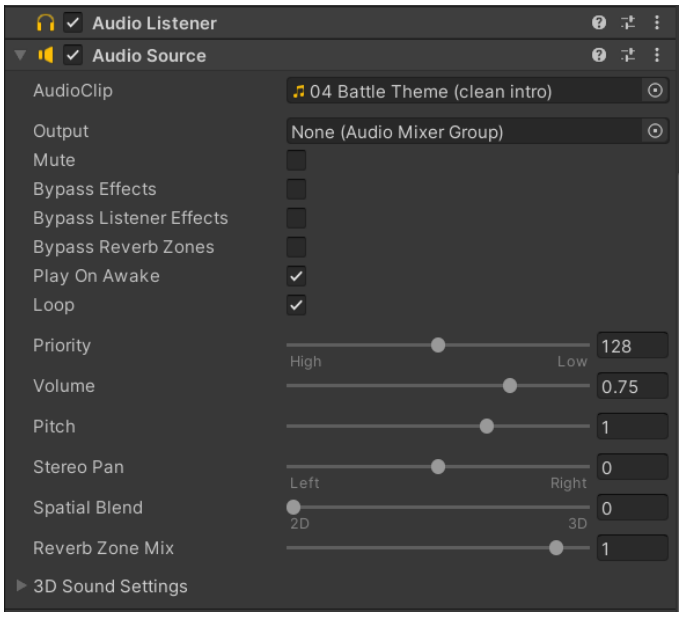


Figure 51. Audio Listener & Audio Source Components

3.7 ΣΕΝΑΡΙΑ ΧΡΗΣΗΣ

Στο τελευταίο κομμάτι της διπλωματικής εργασίας παρουσιάζονται κάποια πραγματικά σενάρια χρήσης της εφαρμογής online παιχνιδιού επαυξημένης πραγματικότητας. Κατά την χρήση, δύο παίκτες έχουν συνδεθεί στον ίδιο server και δωμάτιο, ο καθένας από την συσκευή του και στον δικό του χώρο, και παίζουν μία παρτίδα από το «C{AR}D GAME». Ακολουθούν φωτογραφίες από την ροή του παιχνιδιού.

Figure 52. Αρχική οθόνη παιχνιδιού με επιλογές και ρυθμίσεις

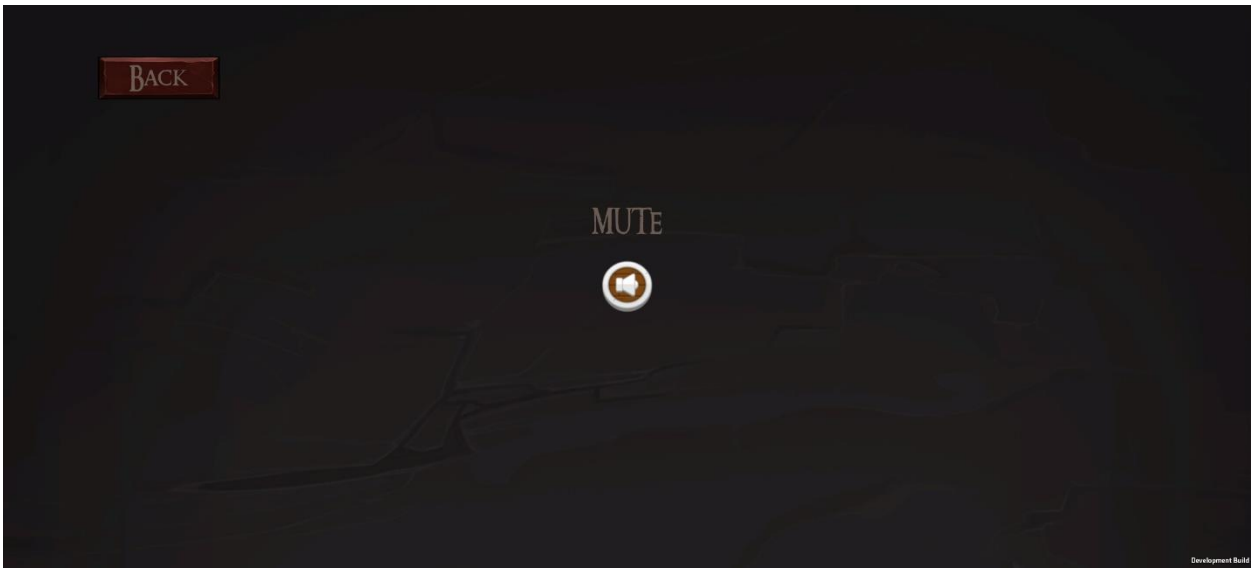


Figure 53. Πατώντας το κουμπί "Settings"

Από την οθόνη “Settings” μπορεί να απενεργοποιηθεί ο ήχος του παιχνιδιού.

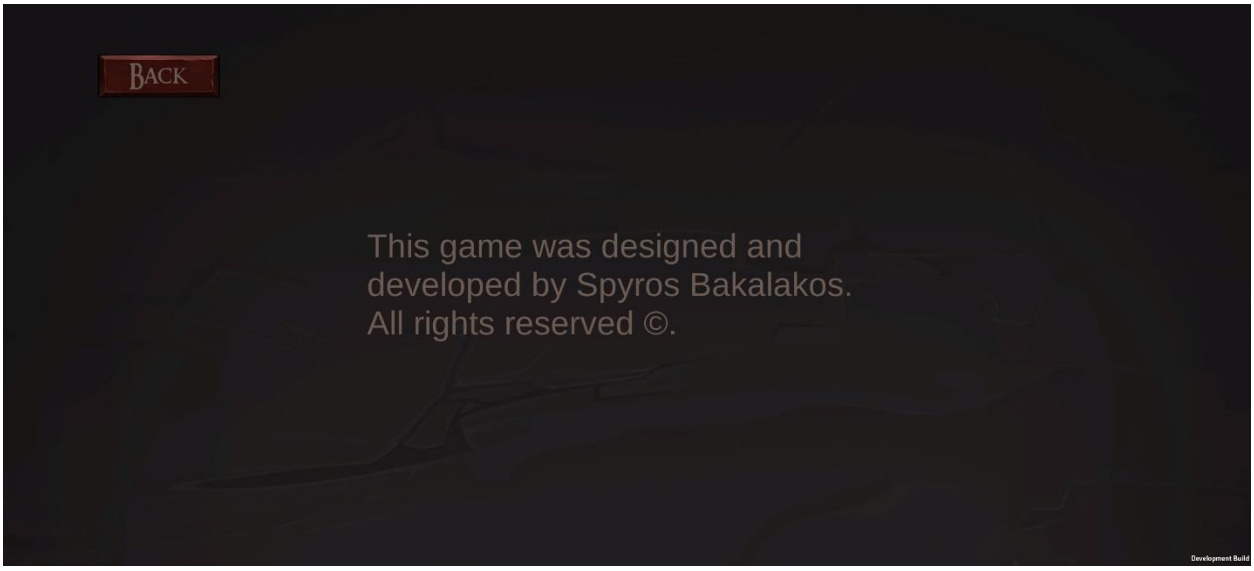


Figure 54. Πατώντας το κουμπί "Info"

Στην οθόνη “Info” βρίσκονται κάποιες πληροφορίες για τον δημιουργό της εφαρμογής και τα πνευματικά δικαιώματα.

Figure 55. Πατώντας το κουμπί "Quit"

Από την οθόνη “Quit” μπορεί να γίνει έξοδος από την εφαρμογή.

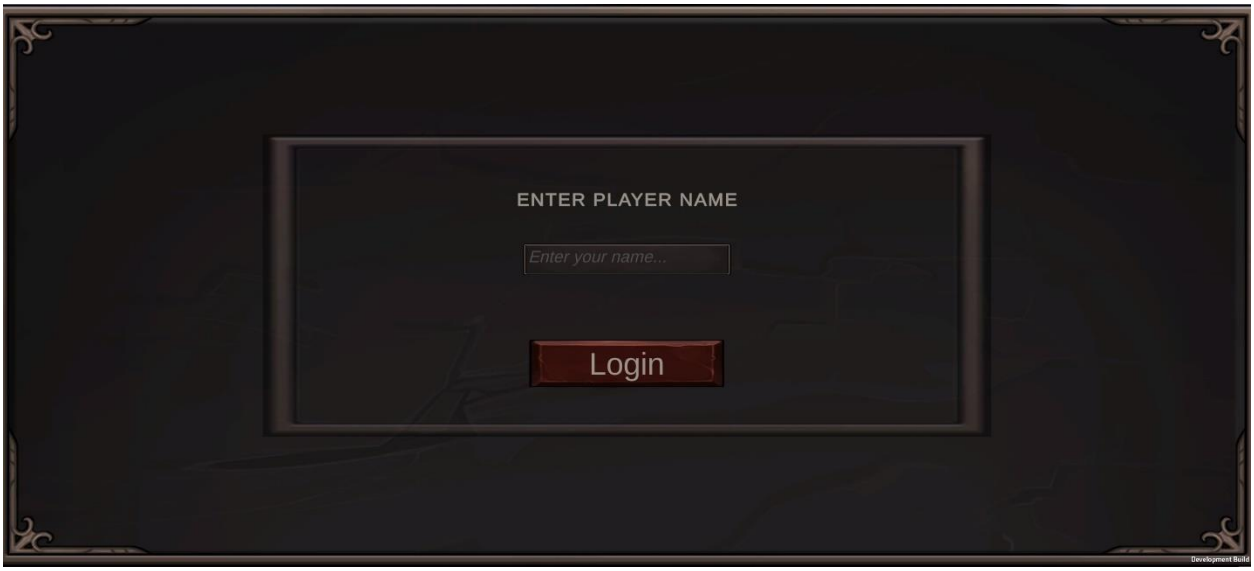


Figure 56. Πατώντας το κουμπί "Play"

Το κουμπί “Play” προτρέπει τον χρήστη να εισάγει ένα όνομα με το οποίο θα συνδεθεί στον Photon Server.

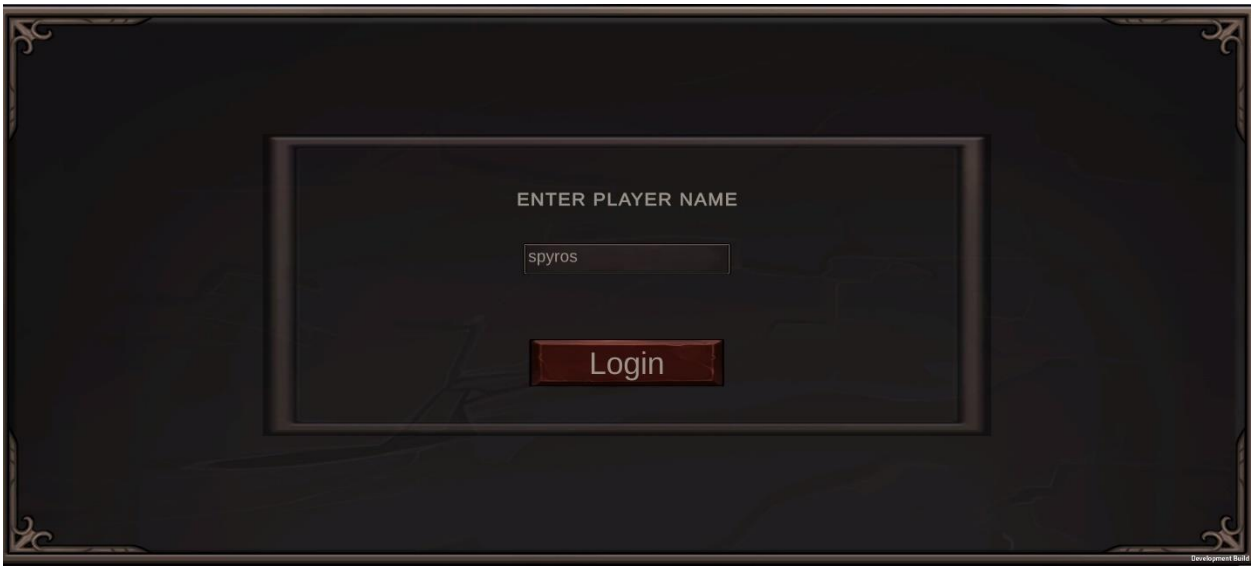


Figure 57. Εισάγοντας το όνομα παίκτη στο πεδίο "Username"

Με το κουμπί “Login” γίνεται η σύνδεση στον server. Αν το πεδίο ονόματος είναι κενό δεν μπορεί να πραγματοποιηθεί σύνδεση.

Figure 58. Επιτυχής σύνδεση χρήστη στον server

Figure 59. Ανεπιτυχής σύνδεση χρήστη στον server

Κατά την διάρκεια της προσπάθειας σύνδεσης ενημερώνεται ο χρήστης για την εξέλιξη της διαδικασίας.

Figure 60. Lobby Scene - μετά από επιτυχημένη σύνδεση στον server πατώντας το κουμπί "Login"

Αυτή η οθόνη είναι ο «προθάλαμος» του παιχνιδιού στον οποίο ο χρήστης μπορεί να δημιουργήσει ή να συμμετέχει σε ένα δωμάτιο.

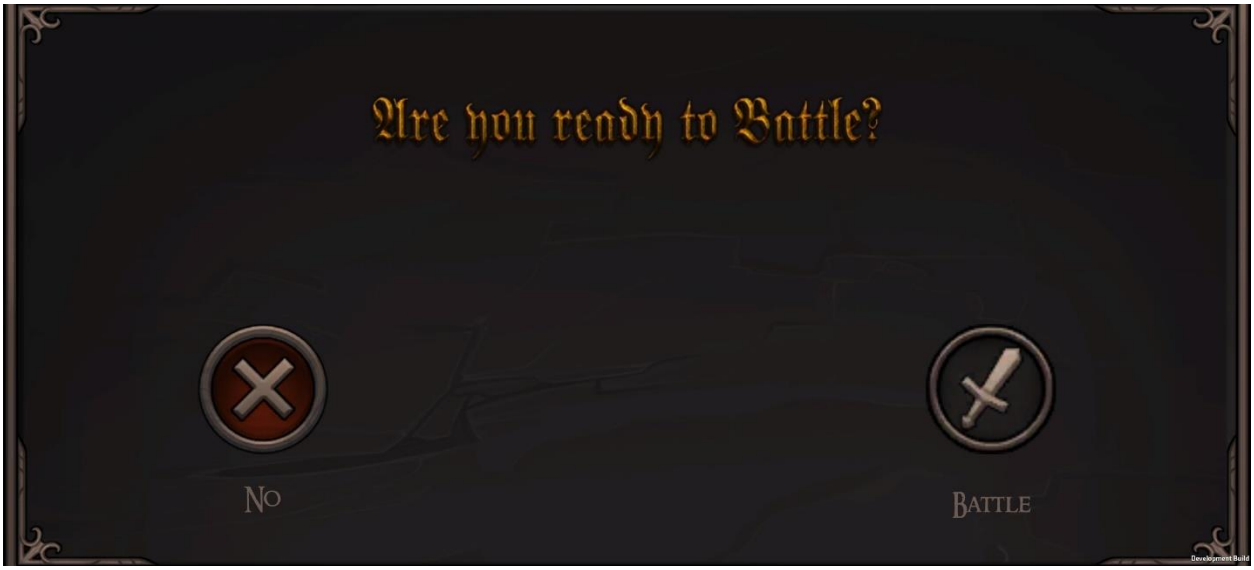


Figure 61. Πατώντας το κουμπί "Search for Games"

Αφού συνδεθεί επιτυχώς σε δωμάτιο, ο παίκτης αποφασίζει αν θα ξεκινήσει το παιχνίδι ή θα γυρίσει στη αρχική οθόνη.

Figure 62. Φόρτωση σκηνής επαυξημένης πραγματικότητας

Όταν ο παίκτης επιλέξει να προχωρήσει στην σκηνή μάχης ενημερώνεται για το δωμάτιο που δημιουργήθηκε και πως πριν προχωρήσει πρέπει να περιμένει την είσοδο δεύτερου παίκτη.

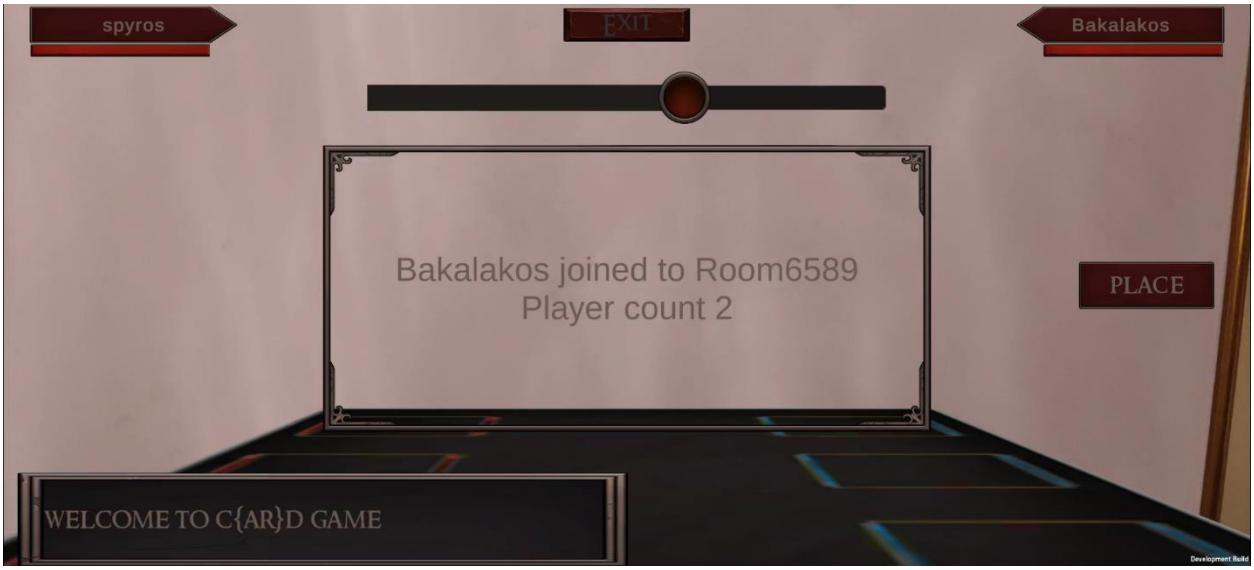


Figure 63. Είσοδος δεύτερου παίκτη στο δωμάτιο

Μόλις εισέλθει και ο αντίπαλος παίκτης, η εφαρμογή ενημερώνει τον κάτοχο του δωματίου για αυτή την ενέργεια, προβάλλοντας το όνομά του και τον αριθμό των παικτών στο δωμάτιο.

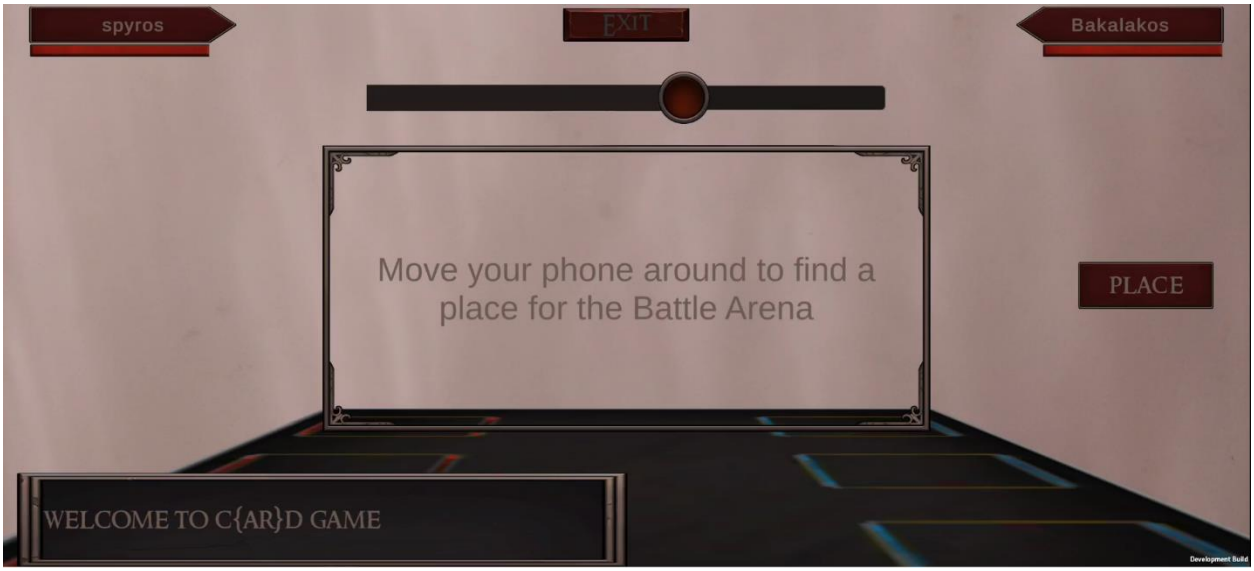


Figure 64. Τοποθέτηση αρένας

Μετά την είσοδο και των δύο παικτών, οι παίκτες καλούνται να βρούν ένα σημείο στον χώρο για να τοποθετήσουν την αρένα και να παλέψουν.

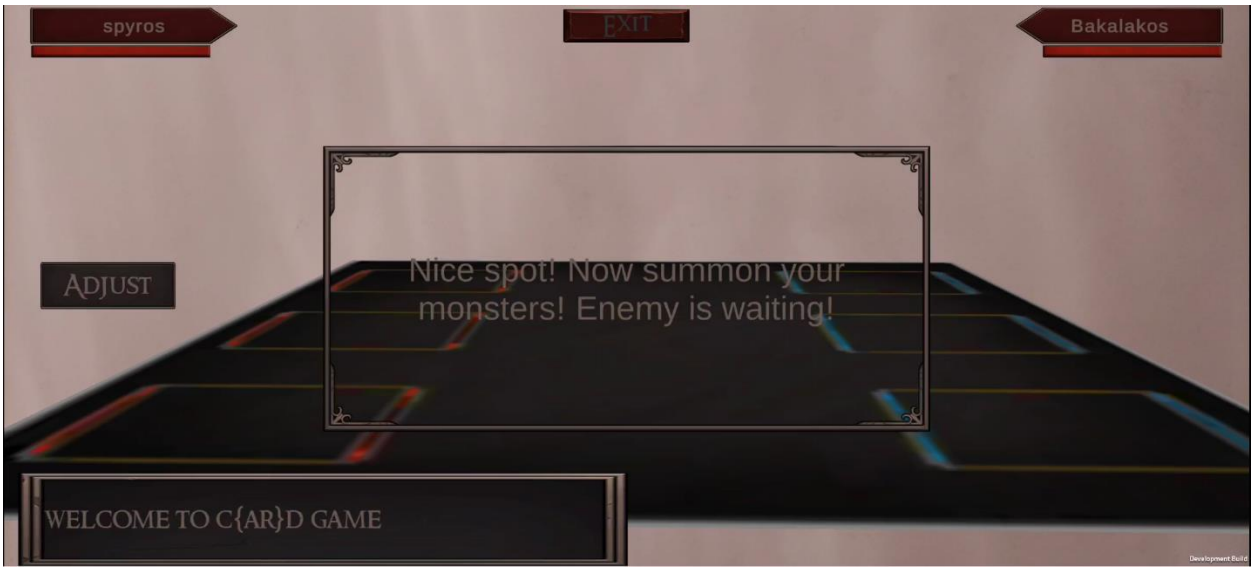


Figure 65. Τελευταίο μήνυμα προς τους παίκτες

Με το πάτημα του κουμπιού “PLACE”, η αρένα ακινητοποιείται στον χώρο. Με το κουμπί “ADJUST” μπορεί να πάρει καινούρια θέση.

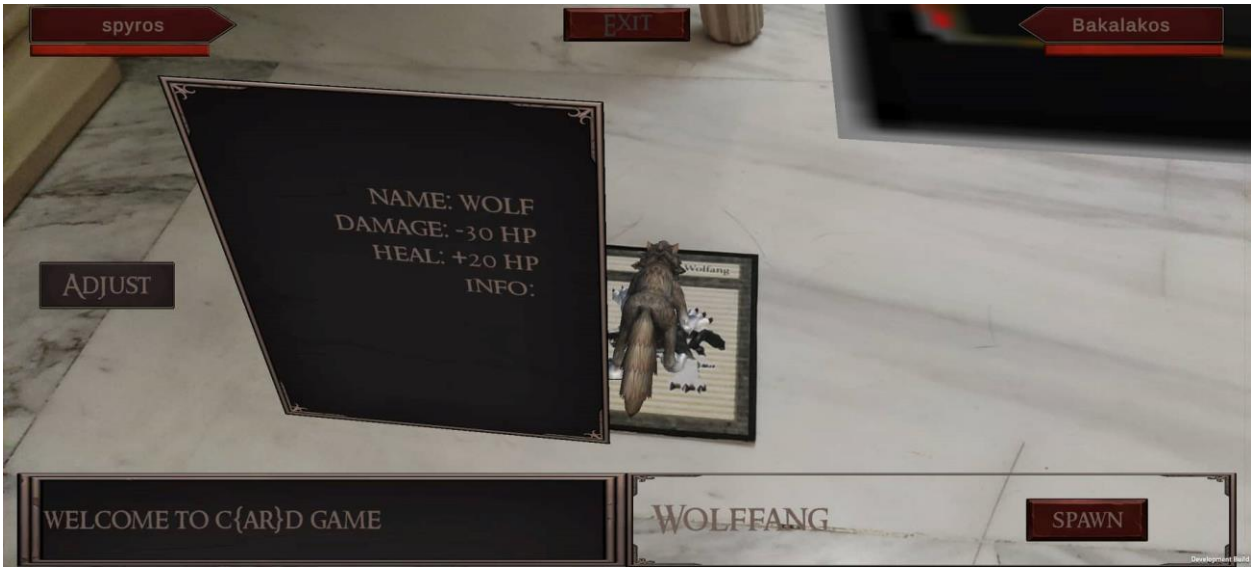


Figure 66. Αναγνώριση κάρτας

Εφόσον η αρένα μάχης είναι έτοιμη, οι δύο παίκτες μπορούν να «διαβάσουν» με το κινητό τους την κάρτα τους και να εμφανίσουν τις πληροφορίες για τον χαρακτήρα της κάρτας.

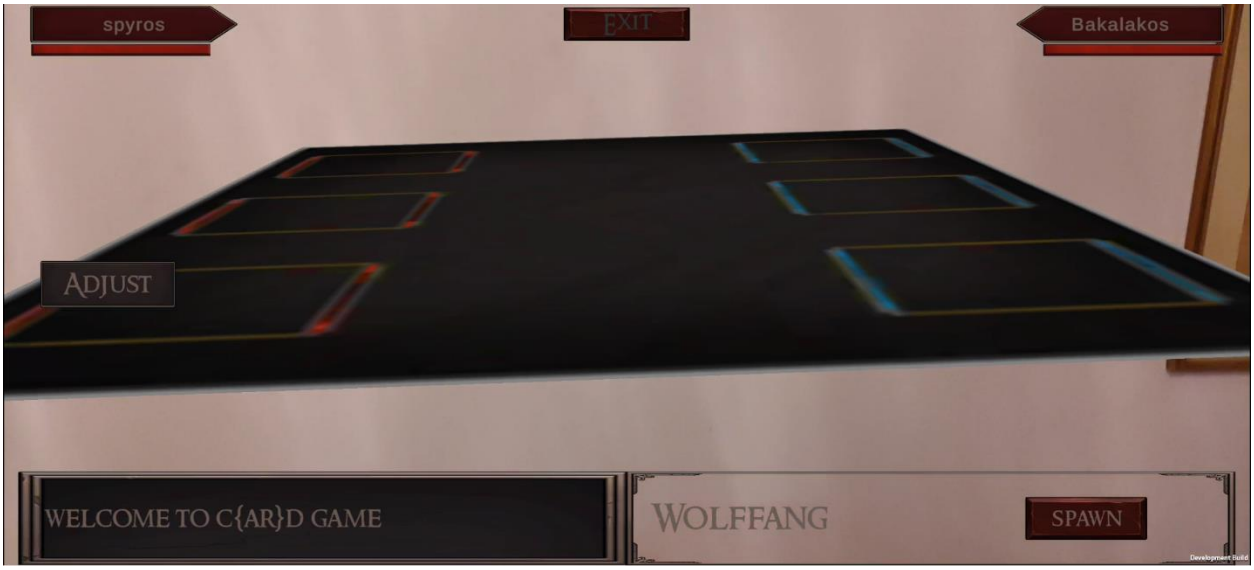


Figure 67. Εμφάνιση χαρακτήρα

Ταυτόχρονα με την αναγνώριση της κάρτας γίνετα ορατό στην οθόνη ένα παράθυρο κάτω δεξιά, το οποίο περιέχει όλα τα στοιχεία και τις δυνατότητες του εκάστοτε χαρακτήρα. Αρχικά αυτά είναι το όνομα του χαρακτήρα και ένα κουμπί με το οποίο οι παίκτες μπορούν να «καλέσουν» τον χαρακτήρα αυτό για μάχη.

Figure 68. Ο χαρακτήρας του τοπικού παίκτη (local player)

Πατώντας το κουμπί “SPAWN” εμφανίζουμε τον χαρακτήρα που διαβάστηκε απο την κάρτα και είμαστε έτοιμοι για μάχη. Με το κάλεσμα του χαρακτήρα στην οθόνη απενεργοποιείται το κουμπί “SPAWN” και ενεργοποιούνται τα δύο κουμπιά επίθεσης “ATTACK” και ενδυνάμωσης “HEAL”. Δηλαδή οι δύο βασικές δυνατότητες κάθε χαρακτήρα.

Figure 69. Ο χαρακτήρας του απομακρυσμένου παίκτη (remote player)

Αφού αντίστοιχα και ο αντίπαλος παίκτης αναγνωρίσει την κάρτα και καλέσει τον χαρακτήρα στην αρένα, ο χαρακτήρας θα εμφανιστεί στην κατάλληλη θέση που δεσμέυεται για τον απομακρυσμένο παίκτη στο στιγμιότυπο παιχνιδιού του τοπικού παίκτη. Αντίστοιχη εικόνα θα έχει και ο απομακτυσμένος παίκτης στο δικό του στιγμιότυπο παιχνιδιού.

Figure 70. Επίθεση στον αντίπαλο

Ο παίκτης που είναι ο ιδιοκτήτης του δωματίου ξεκινάει πρώτος τον γύρο. Για αυτό και τα κουμπιά είναι ενεργοποιημένα στην οθόνη του. Μόλις ο παίκτης κάνει κάποια ενέργιεα, πχ επίθεση, ενημερώνεται το κέιμενο στο battle info panel, ο παίκτης – στόχος (remote player) χάνει ένα κομμάτι απο την ζωή του, ανάλογο με την δύναμη του χαρακτήρα που επιτέθηκε και ο γύρος τελειώνει. Μετά το τέλος του γύρου είναι σειρά του άλλου παίκτη να παίξει, για αυτό τα κουμπιά του τοπικού παίκτη απενεργοποιούνται ενώ του απομακρυσμένου παίκτη είναι ενεργά.

Figure 71. Επίθεση απο τον αντίπαλο

Όταν ο αντίπαλος κάνει επίθεση στον τοπικό παίκτη, το UI ενημερώνεται κατάλληλα.

Figure 72. Ανανέωση γραφικού περιβάλλοντος

Μετά την επίθεση απο τον αντίπαλο φαίνεται η μείωση της μπάρας ζωής του παίκτη, αναγράφονται οι εναπομείναντες πόντοι ζωής και τα κουμπιά γίνονται πάλι ενεργά μιας και είναι ξανά ο γύρος του τοπικού παίκτη.

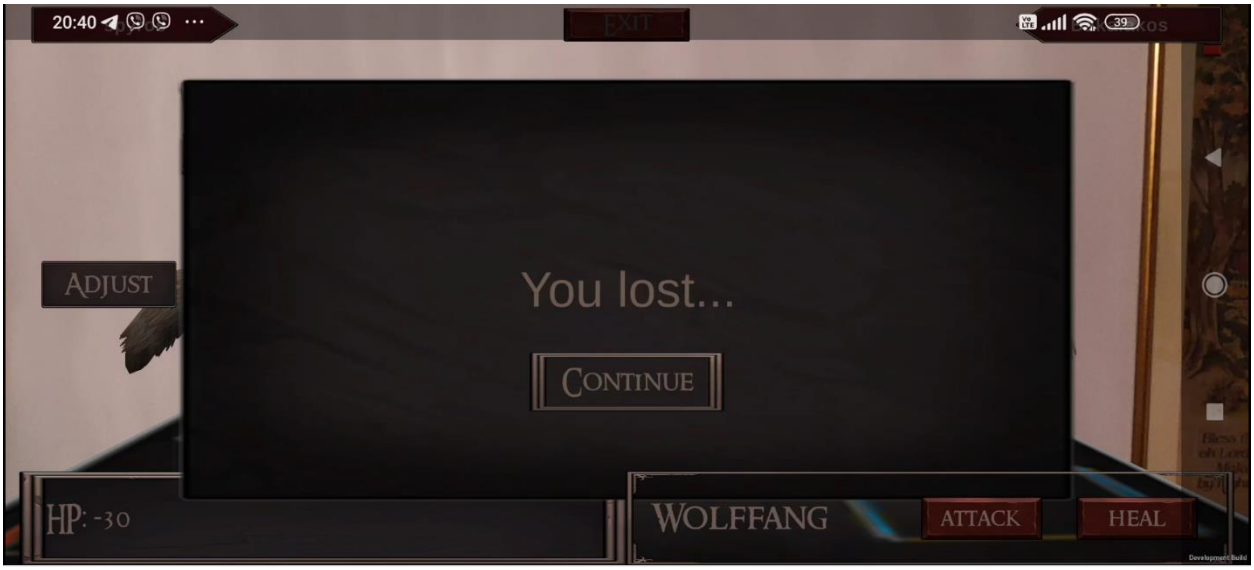


Figure 73. Τέλος παιχνιδιού

Όταν κάποιος απο τους δύο παίκτες χάσει όλους τους πόντους ζωής του, η παρτίδα τερματίζεται. Η παραπάνω εικόνα δείχνει πως ο τοπικός παίκτης αναδείχτηκε χαμένος του παιχνιδιού. Στην αντίστοιχη οθόνη του απομακρυσμένου παίκτη θα αναγράφεται η φράση “YOU WIN!!!”.

Figure 74. Τελική οθόνη απομακρυσμένου παίκτη

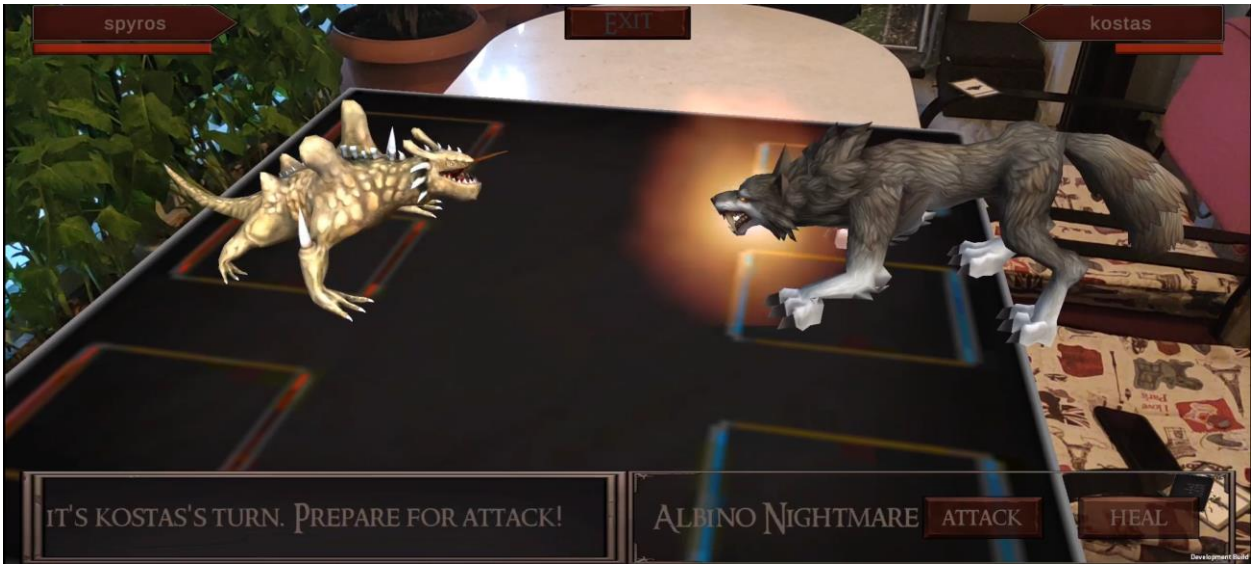
Figure 75. Πατώντας το κουμπί "Continue"

Στο τέλος οι παίκτες θα ερωτηθούν αν θέλουν να τερματίσουν την συνεδρία ή επιθυμούν να παίξουν νέο παιχνίδι από την αρχή. Με τον τερματισμό της συνεδρίας ο παίκτης αποσυνδέεται από το δωμάτιο και τον server και επιστρέφει στην αρχική σκηνή της εφαρμογής. Αν επιλέξουν νέα παρτίδα, επαναφορτώνεται η σκηνή μάχης επαυξημένης πραγματικότητας.

Ακολουθούν μερικές ακόμα ενδεικτικές φωτογραφίες μάχης,


EXIT
kostas
NAME: ALBINO NIGHTMARE
DAMAGE: -40 HP
HEAL: +5 HP
INFO:
YOUR MOVE...
SPAWN


spyros
EXIT
kostas
ALBINO NIGHTMARE IS ATTACKING
ALBINO NIGHTMARE
ATTACK
HEAL


spyros
EXIT
kostas
IT'S KOSTAS'S TURN. PREPARE FOR ATTACK!
ALBINO NIGHTMARE
ATTACK
HEAL


spyros
EXIT
kostas
YOU HAVE BEEN ATTACKED! HP: 70
ALBINO NIGHTMARE
ATTACK

ΚΕΦΑΛΙΑΟ 4: ΜΕΛΛΟΝΤΙΚΗ ΕΞΕΛΙΞΗ

Η παρούσα εργασία αποτελεί ένα μόνο δείγμα των δυνατοτήτων και της δυναμικής που προσφέρει τόσο η μηχανή γραφικών Unity όσο και η τεχνολογία της Επαυξημένης πραγματικότητας. Στην πραγματικότητα υπάρχουν αμέτρητες ακόμα δυνατότητες στην παραγωγή βιντεοπαιχνιδιών και στην δημιουργία γραφικών.

Με μια πρώτη σκέψη υπάρχουν πολλά περιθώρια περαιτέρω ενίσχυσης του εν λόγω παιχνιδιού. Αφ’ ενός μπορούν να προστεθούν περισσότερα χαρακτηριστικά στο τρόπο παιξίματος της παρτίδας, κάνοντας το gameplay πιο απαιτητικό και πολύπλοκο για τους παίκτες, αφ’ ετέρου μπορεί να δοθεί έμφαση στην ποικιλία των στοιχείων δημιουργώντας περισσότερους χαρακτήρες με διαφορετικές κινήσεις και ιδιότητες. Επίσης, αν το επιθυμεί ο προγραμματιστής, είναι στο χέρι του να πειραματιστεί με την δημιουργία δικών του χαρακτήρων και animation. Υπάρχουν πολλά εργαλεία στην αγορά τα οποία είναι δωρεάν και ενδείκνυνται για εκμάθηση νέων καλλιτεχνών. Ακόμα, όσον αφορά την επαυξημένη πραγματικότητα μια ωραία ιδέα θα ήταν η εναλλαγή του περιβάλλοντος γύρω από τον χρήστη με δυναμικό τρόπο μέσω επαύξησης του πραγματικού κόσμου, στοιχείο που θα αύξανε την αίσθηση εμβύθισης στον χρήστη.

Ο χώρος της βιομηχανίας του gaming γνωρίζει εδώ και χρόνια σημαντική άνοδο και θα συνεχίσει την ανοδική πορεία, ειδικά με την είσοδο νέων καινοτόμων τεχνολογιών στον χώρο. Η επαυξημένη και η εικονική πραγματικότητα είναι τεχνολογίες οι οποίες κερδίζουν μέρα με την μέρα τον σεβασμό και των δημιουργών αλλά και της κοινότητας και για αυτό αξίζει κανείς να ασχοληθεί και να επενδύσει σε αυτές.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Θεοχάρης, Θ., et al. *Γραφικά Και Οπτικοποίηση, Αρχές Και Αλγόριθμοι*. Vol. 3, ΕΚΔΟΣΕΙΣ ΣΥΜΜΕΤΡΙΑ, 2015.

[2] Marschner, Steve, and Peter Shirley. *Fundamentals of Computer Graphics*. 5th ed., A K Peters/CRC Press, 2021.

[3] J. Eck, David. *Introduction to Computer Graphics*. Version 1.3.1, Geneva, NY, Department of Mathematics and Computer Science, 2021, https://doi.org/10.1145/1103900.1103904.

[4] Λέπουρας, Γεώργιος, et al. *Ανάπτυξη συστημάτων εικονικής πραγματικότητας*. ΑΘΗΝΑ, Kallipos, 2015.

[5] ΜΟΥΣΤΑΚΑΣ, ΚΩΣΤΑΝΤΙΝΟΣ, et al. *ΓΡΑΦΙΚΑ ΚΑΙ ΕΙΚΟΝΙΚΗ ΠΡΑΓΜΑΤΙΚΟΤΗΤΑ*. ΑΘΗΝΑ, Kallipos, 2015.

[6] Mei, Y., Nie, Q., Wang, F., -, al, Afandi, B., Kustiawan, I., Herman -, N. D., Chen, Y., Wang, Q., Chen, H., Song, X., Tang, H., & Tian, M. (2019). *An overview of augmented reality technology*. 22082. https://doi.org/10.1088/1742-6596/1237/2/022082

[7] Rokhsaritalemi, S., Sadeghi-Niaraki, A., & Choi, S.-M. (n.d.). *A Review on Mixed Reality: Current Trends, Challenges and Prospects*. https://doi.org/10.3390/app10020636

[8] ΒΟΣΙΝΑΚΗΣ, ΣΠΥΡΙΔΩΝ. *Εικονικοί Κόσμοι*. Αθήνα, kallipos, 2015.

[9] Αναγνώστου, Κώστας. *ΒΙΝΤΕΟΠΑΙΧΝΙΔΙΑ, Βιομηχανία και Ανάπτυξη*. Αθήνα, Εκδόσεις Κλειδάριθμος, 2009.

[10] "The History of Video Games." *Swiss National Museum - Swiss History Blog*, 11 Nov. 2021, blog.nationalmuseum.ch/en/2020/01/the-history-of-video-games.

[11] Unity Technologies. "Unity - Manual: Unity User Manual 2020.3 (LTS)." *Unity.Com*, docs.unity3d.com/Manual. Accessed 28 Jan. 2022.

[12] Microsoft. "Developer Tools, Technical Documentation and Coding Examples." *Microsoft Docs*, docs.microsoft.com/en-us. Accessed 28 Jan. 2022.

[13] Google. "Overview of ARCore and Supported Development Environments |." *Google Developers*, developers.google.com/ar/develop. Accessed 28 Jan. 2022.

[14] Photon. "Realtime Intro | Photon Engine." *Photon*, doc.photonengine.com/en-us/realtime/current/getting-started/realtime-intro. Accessed 28 Jan. 2022.

[15] School of Computer Science Carnegie Mellon University. "A Distributed Architecture for Interactive Multiplayer Games." *Cs.Cmu.Edu*, 2005, pp. 1–4, www.cs.cmu.edu/~ashu/papers/cmu-cs-05-112.pdf.

[16] Das, P., Zhu, ou, McLaughlin, L., Bilgrami, Z., Milanaik, R. L., Cruz-Neira, C., Fernández Marín, M., & Portalés Ricart, C. (n.d.). *Multimodal Technologies and Interaction Review Augmented Reality Video Games: New Possibilities and Implications for Children and Adolescents*. https://doi.org/10.3390/mti1020008

[17] ZackIken, et al. "Singletons in Unity (Done Right)." *Game Dev Beginner*, 23 Dec. 2021, https://gamedevbeginner.com/singletons-in-unity-the-right-way/#unity_singleton.

[18] BillWagner. "Interfaces - Define Behavior for Multiple Types." *Microsoft Docs*, https://docs.microsoft.com/en-us/dotnet/csharp/fundamentals/types/interfaces.

[19] "About Ar Foundation: Ar Foundation: 3.0.1." *AR Foundation | 3.0.1*, https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@3.0/manual/index.html.