



**ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΟΝΙΚΗΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ**

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

*Δημιουργία εφαρμογής Επαυξημένης Πραγματικότητας για
φορητή συσκευή Android*

ΠΕΝΝΑΣ ΑΘΑΝΑΣΙΟΣ

**ΕΠΙΒΛΕΠΩΝ:
ΝΙΚΟΛΑΪΔΗΣ ΝΙΚΟΛΑΟΣ, ΑΝΑΠΛΗΡΩΤΗΣ
ΚΑΘΗΓΗΤΗΣ**

ΟΚΤΩΒΡΙΟΣ 2024

ΘΕΣΣΑΛΟΝΙΚΗ

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ρητά ότι η παρούσα πτυχιακή εργασία, καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στο πλαίσιο αυτής της εργασίας, αποτελεί αποκλειστικά προϊόν προσωπικής μου εργασίας, δεν προσβάλλει κάθε μορφής δικαιώματα διανοητικής ιδιοκτησίας, προσωπικότητας και προσωπικών δεδομένων τρίτων, δεν περιέχει έργα/εισφορές τρίτων για τα οποία απαιτείται άδεια των δημιουργών/δικαιούχων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον και πληρούν τους κανόνες της επιστημονικής παράθεσης. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή/και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή. Αναλαμβάνω πλήρως, ατομικά και προσωπικά, όλες τις νομικές και διοικητικές συνέπειες που δύναται να προκύψουν στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής

ΠΕΡΙΛΗΨΗ

Η Επαυξημένη Πραγματικότητα αποτελεί μία από τις πιο καινοτόμες τεχνολογίες της εποχής καθώς επιτρέπει την αλληλεπίδραση του χρήστη με το φυσικό περιβάλλον, ενσωματώνοντας ψηφιακή πληροφορία. Οι πρώτες δημιουργίες επαυξημένης πραγματικότητας έκαναν την εμφάνισή τους στις αρχές της δεκαετίας του 90' με πρώτο εγχείρημα το Virtual Fixtures σύστημα που αναπτύχθηκε στο εργαστήριο Armstrong της πολεμικής αεροπορίας των ΗΠΑ το 1992. Οι εμπειρίες που παρείχαν τα συστήματα μεικτής πραγματικότητας στους χρήστες ήταν καθηλωτικές και ιδιαίτερα πρωτοπόρες για την εποχή. Στην συνέχεια των χρόνων, οι εμπορικές εμπειρίες επαυξημένης πραγματικότητας εισήχθησαν για πρώτη φορά σε επιχειρήσεις ψυχαγωγίας και παιχνιδιών. Στη σύγχρονη εποχή, οι εφαρμογές της έχουν εισαχθεί στον τομέα της βιομηχανίας, της εκπαίδευσης, της επικοινωνίας όπως και της ψυχαγωγίας[1].

Το Unity αποτελεί μία από τις πιο δημοφιλείς πλατφόρμες ανάπτυξης παιχνιδιών και διαδραστικών εφαρμογών παγκοσμίως. Αναλυτικότερα πρόκειται για μία μηχανή γραφικών πολλαπλών πλατφορμών που αναπτύχθηκε από τη Unity Technologies η οποία κυκλοφόρησε πρώτη φορά τον Ιούνιο του 2005 στο Apple Worldwide Developers Conference ως μηχανή γραφικών Mac OS X. Έκτοτε η μηχανή επεκτάθηκε σταδιακά με σκοπό την υποστήριξη ποικίλων πλατφορμών επιτραπέζιων υπολογιστών, κινητών, κονσολών, επαυξημένης και εικονικής πραγματικότητας[2].

ABSTRACT

Augmented reality is one of the most innovative technologies of the time as it allows the user to interact with the physical environment, integrating digital information. The first AR creations appeared in the early 90s with the Virtual Fixtures system, developed at the Armstrong laboratory of the USA Air Force in 1992. The experiences that mixed reality systems provided to users were immersive and highly pioneering for the time. Over the years, augmented reality commercial experiences were first introduced to entertainment and gaming businesses. In modern ages, its applications have been introduced in the field of industry, education, communication as well as entertainment.

Unity is one of the most famous game engines and interactive applications worldwide. More specifically it is a cross-platform game engine developed by Unity Technologies which was first released in June 2005 at the Apple Worldwide Developers Conference as the Mac OS X graphics engine. Since then the engine has been gradually expanded to support a variety of desktop, mobile and console platforms as also augmented and virtual reality technologies.

[1] Ανακτήθηκε από https://en.wikipedia.org/wiki/Augmented_reality

[2] Ανακτήθηκε από [https://en.wikipedia.org/wiki/Unity_\(game_engine\)](https://en.wikipedia.org/wiki/Unity_(game_engine))

ΕΥΧΑΡΙΣΤΙΕΣ

Στο σημείο αυτό θα ήθελα να ευχαριστήσω τον καθηγητή και επιβλέποντα της εργασίας, κύριο Νικολαΐδη Νικόλαο, για την εμπιστοσύνη που μου έδειξε με την ανάθεση του θέματος αλλά και για τη κατανόηση και βοήθειά του σε όλο αυτό το ταξίδι.

Θα ήθελα επίσης να ευχαριστήσω τους γονείς μου που όλα αυτά τα χρόνια με στήριξαν με οποιοδήποτε τρόπο και δεν έπαψαν ποτέ να πιστεύουν σε μένα. Ελπίζω να τους έκανα έστω και λίγο περήφανους.

Ένα μεγάλο ευχαριστώ στους φίλους μου και στα αδέρφια μου που με ενθάρρυναν ψυχολογικά καθ' όλη τη διάρκεια.

Τέλος θα ήθελα να ευχαριστήσω μέσα από την καρδιά μου τη Σοφία, τη σύντροφό μου, το στήριγμά μου που ήταν πάντα δίπλα μου σε αυτή τη περίοδο ανεχόμενη τα πάντα και βοηθώντας με ψυχολογικά. Δεν σταμάτησε ποτέ να πιστεύει σε μένα δείχνοντάς το μου με κάθε τρόπο.

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΕΡΙΛΗΨΗ	3
ABSTRACT	3
ΕΥΧΑΡΙΣΤΙΕΣ	4
ΠΕΡΙΕΧΟΜΕΝΑ	5
ΠΙΝΑΚΑΣ ΕΙΚΟΝΩΝ	7
ΚΕΦΑΛΑΙΟ 1 : ΕΙΣΑΓΩΓΗ	9
1.1 ΣΤΟΧΟΣ ΤΗΣ ΠΤΥΧΙΑΚΗΣ	9
1.2 ΤΙ ΕΙΝΑΙ ΕΠΑΥΞΗΜΕΝΗ ΠΡΑΓΜΑΤΙΚΟΤΗΤΑ	9
1.3 ΠΕΔΙΟ - ΚΑΤΗΓΟΡΙΕΣ ΕΦΑΡΜΟΓΩΝ	13
1.4 ΕΡΓΑΛΕΙΑ ΠΟΥ ΧΡΗΣΙΜΟΠΟΙΗΘΗΚΑΝ ΚΑΤΑ ΤΗΝ ΑΝΑΠΤΥΞΗ	15
1.4.1 UNITY ENGINE	15
1.4.2 AR FOUNDATION	15
1.4.3 ARCORE / ARKIT	16
1.5 ΡΥΘΜΙΣΕΙΣ ΣΥΣΤΗΜΑΤΟΣ	17
ΚΕΦΑΛΑΙΟ 2 : ΤΟ ΑΡΧΙΚΟ SCENE	18
2.1 LOADING SCENE KAI MAIN APP SCENE LOADER	20
ΚΕΦΑΛΑΙΟ 3 : SPINNER TOP MULTIPLAYER GAME	21
3.1 LOBBY SCENE	21
3.2 PHOTON UNITY NETWORKING (PUN)	22
3.3 PLAYER SELECTION SCENE	24
3.4 GAMEPLAY SCENE	26
3.4.1 AR SESSION KAI AR SESSION ORIGIN	27
3.4.2 ARENA PLACEMENT	28
3.5 ARENA SETUP	29
3.6 BATTLE SCRIPT	29

ΚΕΦΑΛΑΙΟ 4 : ICE HOCKEY MULTIPLAYER GAME.....	32
4.1 ICE HOCKEY LOBBY.....	32
4.2 ICE HOCKEY PLAYER SELECTION.....	33
4.3 ICE HOCKEY GAMEPLAY SCENE.....	34
4.3.1 GAMEPLAY PITCH.....	35
4.3.2 BALL PREFAB.....	37
4.4 GAME SETUP.....	38
4.5 GOAL SETUP.....	40
ΚΕΦΑΛΑΙΟ 5 : POST-IT.....	41
5.1 SCENE PROPERTIES.....	41
5.2 POST-IT BEHAVIOUR.....	44
ΚΕΦΑΛΑΙΟ 6 : ΨΗΦΙΑΚΟ ΚΑΤΟΙΚΙΔΙΟ.....	46
6.1 SCENE PROPERTIES.....	46
6.2 SCRIPTING.....	47
6.3 DIGITAL DOG PREFAB.....	50
6.4 ANIMATION.....	51
6.5 DOG CANVAS.....	52
6.6 PET DOG SCRIPT.....	54
6.7 ΕΝΔΕΙΚΤΙΚΕΣ ΦΩΤΟΓΡΑΦΙΕΣ.....	55
ΕΠΙΛΟΓΟΣ.....	56
ΑΝΑΦΟΡΕΣ.....	57

ΠΙΝΑΚΑΣ ΕΙΚΟΝΩΝ

ΕΙΚΟΝΑ 1 : ΠΑΡΑΔΕΙΓΜΑ ΕΠΑΥΞΗΜΕΝΗΣ ΠΡΑΓΜΑΤΙΚΟΤΗΤΑΣ.....	10
ΕΙΚΟΝΑ 2 : ΠΑΡΑΔΕΙΓΜΑ ΕΦΑΡΜΟΓΗΣ FACE FILTER.....	13
ΕΙΚΟΝΑ 3 : PIZZA HUT ARCADE PAC MAN ΕΦΑΡΜΟΓΗ.....	13
ΕΙΚΟΝΑ 4 : ANGRY BIRDS AR.....	14
ΕΙΚΟΝΑ 5 : POKEMON GO.....	14
ΕΙΚΟΝΑ 6 : UNITY LOGO.....	15
ΕΙΚΟΝΑ 7 : AR FOUNDATION PACKAGE, ARCORE XR PLUGIN PACKAGE.....	16
ΕΙΚΟΝΑ 8 : BUILD SETTINGS.....	17
ΕΙΚΟΝΑ 9 : MAINAPP_SCENE HIERARCHY.....	18
ΕΙΚΟΝΑ 10 : ΠΑΡΑΔΕΙΓΜΑ ΣΥΝΔΕΣΗΣ BUTTON ME SCRIPT.....	19
ΕΙΚΟΝΑ 11 : MAINAPP_LOADING SCENE.....	20
ΕΙΚΟΝΑ 12 : SPINNER TOP LOGIN PHASE.....	21
ΕΙΚΟΝΑ 13 : SPINNER TOP LOBBY PHASE.....	22
ΕΙΚΟΝΑ 14 : PHOTON PUN APP REGISTRATION.....	23
ΕΙΚΟΝΑ 15 : PHOTON VIEW COMPONENT.....	23
ΕΙΚΟΝΑ 16 : SPINNER TOP PLAYER SELECTION UI_SELECTION.....	24
ΕΙΚΟΝΑ 17 : ΔΙΑΦΟΡΑ ΜΕΤΑΞΥ ATTACK – DEFEND.....	24
ΕΙΚΟΝΑ 18 : SPINNER TOP PLAYER SELECTION UI_AFTERSELECTION.....	25
ΕΙΚΟΝΑ 19 : SPINNER TOP GAMEPLAY SCENE.....	26
ΕΙΚΟΝΑ 20 : Η ΘΕΣΗ ΤΗΣ BATTLE ARENA ΠΡΙΝ ΓΙΝΕΙ ΟΡΑΤΗ ΣΤΟΝ ΧΡΗΣΤΗ.....	26
ΕΙΚΟΝΑ 21 : AR SESSION.....	27
ΕΙΚΟΝΑ 22 : AR SESSION ORIGIN.....	28
ΕΙΚΟΝΑ 23 : ΟΘΟΝΗ ΠΡΙΝ ΚΑΙ ΜΕΤΑ ΤΟ PLACE.....	29
ΕΙΚΟΝΑ 24 : INSTANTIATION ΤΟΥ ΠΑΙΚΤΗ ΚΑΙ ΛΗΨΗ ΤΟΥ PHOTONVIEW.....	29
ΕΙΚΟΝΑ 25 : RIGIDBODY COMPONENT.....	30
ΕΙΚΟΝΑ 26 : DEATH PANEL.....	31
ΕΙΚΟΝΑ 27 : ICE HOCKEY LOGIN ICE HOCKEY LOBBY.....	32
ΕΙΚΟΝΑ 28 : ICE HOCKEY PLAYER SELECTION.....	33
ΕΙΚΟΝΑ 29 : ICE HOCKEY GAMEPLAY SCENE HIERARCHY.....	34
ΕΙΚΟΝΑ 30 : Η ΘΕΣΗ ΤΟΥ ΓΗΠΕΔΟΥ ΠΡΙΝ ΓΙΝΕΙ ΟΡΑΤΗ ΣΤΟΝ ΧΡΗΣΤΗ.....	35
ΕΙΚΟΝΑ 31 : ΟΙ ΠΑΡΑΜΕΤΡΟΙ ΓΙΑ ΤΑ PHYSICS ΤΩΝ WALLS.....	36
ΕΙΚΟΝΑ 32 : PARTICLE SYSTEM.....	36
ΕΙΚΟΝΑ 33 : BALL PREFAB.....	37
ΕΙΚΟΝΑ 34 : NODE LIBRARY.....	38
ΕΙΚΟΝΑ 35 : GAME SETUP ON START NODE.....	38
ΕΙΚΟΝΑ 36 : TIMER NODE.....	39
ΕΙΚΟΝΑ 37 : GAME SETUP ON UPDATE NODE.....	39
ΕΙΚΟΝΑ 38 : GOAL SETUP.....	40
ΕΙΚΟΝΑ 39 : GOAL SETUP STOP.....	40
ΕΙΚΟΝΑ 40 : EXPLORE SCENE.....	41
ΕΙΚΟΝΑ 41 : POST-IT AR SESSION ORIGIN.....	42
ΕΙΚΟΝΑ 42 : REFERENCE IMAGE LIBRARY, MARKER.....	42
ΕΙΚΟΝΑ 43 : POST-IT PREFAB.....	43
ΕΙΚΟΝΑ 44 : ΤΑ ΣΤΑΔΙΑ ΕΜΦΑΝΙΣΗΣ ΤΟΥ POST-IT.....	43
ΕΙΚΟΝΑ 45 : Η ΥΛΟΠΟΙΗΣΗ ΤΩΝ 4 BUTTONS ΣΕ C#.....	44
ΕΙΚΟΝΑ 46 : ΤΡΟΠΟΣ ΕΥΡΕΣΗΣ ΚΑΙ ΥΛΟΠΟΙΗΣΗΣ ΤΩΝ ΜΕΤΑΒΛΗΤΩΝ ΤΟΥ POST-IT.....	45
ΕΙΚΟΝΑ 47 : DIGITAL DOG AR SESSION ORIGIN COMPONENTS.....	46
ΕΙΚΟΝΑ 48 : ΟΙ ΣΥΝΑΡΤΗΣΕΙΣ ΤΩΝ BUTTONS.....	47

ΕΙΚΟΝΑ 49 : Η UPDATE ΣΥΝΑΡΤΗΣΗ.....	48
ΕΙΚΟΝΑ 50 : SETALLPLANESDEACTIVE ΣΥΝΑΡΤΗΣΗ.....	48
ΕΙΚΟΝΑ 51 : ΧΡΗΣΗ ΤΩΝ GET SPAWN ΚΑΙ GET PRESSED ΜΕΤΑΒΛΗΤΩΝ.....	49
ΕΙΚΟΝΑ 52 : Η ΔΙΑΔΙΚΑΣΙΑ ΕΜΦΑΝΙΣΗΣ ΤΟΥ ΚΑΤΟΙΚΙΔΙΟΥ.....	49
ΕΙΚΟΝΑ 53 : GERMAN SHEPHERD PREFAB.....	50
ΕΙΚΟΝΑ 54 : TO ANIMATOR CONTROLLER ΤΟΥ ΚΑΤΟΙΚΙΔΙΟΥ.....	51
ΕΙΚΟΝΑ 55 : ΠΑΡΑΔΕΙΓΜΑ TRANSITION.....	51
ΕΙΚΟΝΑ 56 : ANIMATION BUTTON SCRIPT.....	52
ΕΙΚΟΝΑ 57 : ONANIMATIONBUTTONCLICK ΣΥΝΑΡΤΗΣΗ.....	53
ΕΙΚΟΝΑ 58 : Η UPDATE ΣΥΝΑΡΤΗΣΗ ΤΟΥ PET DOG SCRIPT.....	54
ΕΙΚΟΝΑ 59 : ANIXNEYΣΗ PLANE.....	55
ΕΙΚΟΝΑ 60 : ΤΟΠΟΘΕΤΗΣΗ ΚΑΤΟΙΚΙΔΙΟΥ.....	55
ΕΙΚΟΝΑ 61 : SIT DOWN BUTTON.....	55
ΕΙΚΟΝΑ 62 : FEED BUTTON.....	55

ΚΕΦΑΛΑΙΟ 1 : ΕΙΣΑΓΩΓΗ

1.1 ΣΤΟΧΟΣ ΤΗΣ ΠΤΥΧΙΑΚΗΣ

Η παρούσα διπλωματική εργασία στοχεύει στην δημιουργία ενός «επαυξημένου» σπιτιού (augmented house). Μέσω της εν λόγω εφαρμογής ο ένοικος ενός σπιτιού θα αποκτήσει τις εξής δυνατότητες, να αναζητήσει στον πραγματικό κόσμο και να σαρώσει (scan), με τη βοήθεια της κάμερας της κινητής συσκευής του, ένα marker το οποίο θα βρίσκεται σε μία επιφάνεια κι αφού γράψει ένα μήνυμα, αυτό να εμφανίζεται πάνω σε ένα post-it το οποίο θα τοποθετηθεί στο εν λόγω marker. Επίσης ανιχνεύοντας μία επίπεδη επιφάνεια, θα μπορεί να παίξει ένα από τα δύο multiplayer παιχνίδια με ένα φίλο του, είτε βρίσκεται στο ίδιο σπίτι είτε σε άλλο χώρο. Τέλος θα έχει τη δυνατότητα να αλληλεπιδράσει με ένα ψηφιακό κατοικίδιο, δίνοντάς του εντολές, προσφέροντάς του φαγητό και χάδια. Η εφαρμογή απευθύνεται σε χρήστες Android συσκευών μέσω της πλατφόρμας Unity όπου χρησιμοποιήθηκαν τα AR Foundation και AR Core εργαλεία για την υλοποίησή της.

1.2 ΤΙ ΕΙΝΑΙ ΕΠΑΥΞΗΜΕΝΗ ΠΡΑΓΜΑΤΙΚΟΤΗΤΑ

Επαυξημένη Πραγματικότητα ή αλλιώς Augmented Reality είναι μία παραλλαγή της Εικονικής Πραγματικότητας (Virtual Reality)[3]. Η Εικονική Πραγματικότητα τοποθετεί τον χρήστη σε ένα εξ' ολοκλήρου συνθετικό (μη πραγματικό) κόσμο που κατά τη διάρκεια της εμπειρίας, δεν μπορεί να δει τον πραγματικό κόσμο γύρω του. Αντιθέτως, κατά της εμπειρία της Επαυξημένης Πραγματικότητας, ο χρήστης αντικρίζει εικονικά αντικείμενα που συνδυάζονται με τον πραγματικό κόσμο. Επομένως η Επαυξημένη Πραγματικότητα συμπληρώνει τον πραγματικό κόσμο, δεν τον αντικαθιστά. Η εικόνα 1 αποτελεί ένα παράδειγμα αυτού όπου σε ένα χώρο σπιτιού, υπάρχει ένα κατοικίδιο και δύο φυτά τα οποία είναι πραγματικά αντικείμενα. Στο χώρο όμως υπάρχει κι ένα εικονικό λιοντάρι το οποίο είναι σταθερά τοποθετημένο στον χώρο σε τρεις διαστάσεις, σαν να είναι αληθινό. Στέκεται σωστά στην επίπεδη επιφάνεια (τα πόδια του ακουμπάνε ακριβώς στο πάτωμα, δεν υπάρχει κάποια ανωμαλία) και δέχεται τις ακτίνες του πραγματικού ήλιου, δημιουργώντας έτσι σκιές πάνω του αλλά και στον πραγματικό κόσμο.

[3] Ανακτήθηκε από “A Survey of Augmented Reality”, Ronald T. Azuma



ΕΙΚΟΝΑ 1 : ΠΑΡΑΔΕΙΓΜΑ ΕΠΑΥΞΗΜΕΝΗΣ ΠΡΑΓΜΑΤΙΚΟΤΗΤΑΣ[4]

Συνεπώς η Επαυξημένη Πραγματικότητα επιτρέπει την άμεση ή έμμεση θέαση ενός φυσικού, πραγματικού περιβάλλοντος σε πραγματικό χρόνο, του οποίου τα στοιχεία επικαλύπτονται από ψηφιακή πληροφορία δημιουργημένη και αναπαραγόμενη από ηλεκτρονικούς υπολογιστές. Η πληροφορία μπορεί να είναι εξ' ολοκλήρου συνθετική ή μπορεί να είναι αντίγραφα πληροφοριών του πραγματικού κόσμου, αναπαριστώμενα ψηφιακά. Δεν υπάρχει περιορισμός ως προς την έννοια που θα αφορούν οι πληροφορίες, μπορεί να είναι οπτικές πληροφορίες, ακουστικές ή και πληροφορίες σχετικά με την μυρωδιά, τη γεύση ή την επαφή. Η πληροφορία μπορεί να είναι στατική, όπως μία φωτογραφία ή ένα τρισδιάστατο μοντέλο, μπορεί να είναι μία ψηφιακή εγγραφή ήχου ή μπορεί να βασίζεται δυναμικά σε μια προσομοίωση υπολογιστή που εξελίσσεται από δεδομένα αισθητήρων σε πραγματικό χρόνο[5].

Συνδυάζοντας πραγματικά και εικονικά αντικείμενα, η Επαυξημένη Πραγματικότητα καθίσταται χρήσιμη καθώς οι πληροφορίες που μεταφέρουν τα αντικείμενα βοηθούν τον άνθρωπο να εκτελεί εργασίες πραγματικού κόσμου. Κάποιες χρήσιμες εφαρμογές είναι οι παρακάτω :

- **Ιατρική** : Οι γιατροί θα μπορούσαν να χρησιμοποιήσουν την επαυξημένη πραγματικότητα ως βοήθημα εκπαίδευσης στην χειρουργική. Μπορεί να είναι δυνατή η συλλογή δεδομένων ενός ασθενούς σε πραγματικό χρόνο χρησιμοποιώντας μη επεμβατικούς αισθητήρες όπως η μαγνητική τομογραφία ή το υπερηχογράφημα. Αυτά τα σύνολα δεδομένων θα μπορούν να αποδίδονται σε πραγματικό χρόνο με θέα τον ασθενή και θα ήταν πολύ χρήσιμο κατά τη διάρκεια μικρών επεμβάσεων, η οποία μειώνει το τραύμα μιας επέμβασης χρησιμοποιώντας μικρές ή και καθόλου τομές.
- **Κατασκευή & Επισκευή** : Άλλη μία κατηγορία όπου η Επαυξημένη Πραγματικότητα θα ήταν χρήσιμη, είναι η συντήρηση κι επισκευή σύνθετων μηχανημάτων. Οι οδηγίες θα ήταν πιο εύκολες και κατανοητές αν αντί για εγχειρίδια χρήσης με κείμενα και εικόνες, εμφανίζονταν ως τρισδιάστατα αντικείμενα ακριβώς πάνω στο πραγματικό εξοπλισμό, δείχνοντας βήμα-βήμα τη διαδικασία που πρέπει να γίνει. Επιπλέον οι οδηγίες θα μπορούσαν να περιέχουν και κίνηση (animation) ώστε να τις καθιστούσαν ακόμα πιο σαφείς.[6]

[4] Ανακτήθηκε από <https://medium.com/6d-ai/how-is-arcore-better-than-arkit-5223e6b3e79d>

[5] Ανακτήθηκε από "Understanding Augmented Reality" Alan B. Craig

[6] Ανακτήθηκε από "A Survey of Augmented Reality", Ronald T. Azuma

- **Εκπαίδευση** : Η Επαυξημένη Πραγματικότητα εμφανίστηκε ως ένα ισχυρό εργαλείο στην εκπαίδευση, φέρνοντας επανάσταση στις παραδοσιακές μεθόδους διδασκαλίας και παρέχοντας καθηλωτικές εμπειρίες μάθησης. Προσφέρει στους μαθητές ένα δυναμικό και διαδραστικό περιβάλλον μάθησης. Τα σχολικά βιβλία μπορούν να μετατραπούν σε εμπειρίες πολυμέσων, με τρισδιάστατα μοντέλα, κινούμενα σχέδια και διαδραστικά στοιχεία επικαλύπτοντας το έντυπο περιεχόμενο. Πολύπλοκες έννοιες, όπως ανατομικές δομές ή ιστορικά ορόσημα, μπορούν να οπτικοποιηθούν σε τρεις διαστάσεις επιτρέποντας στους μαθητές να εξερευνήσουν και να αλληλεπιδράσουν με εικονικά αντικείμενα.
- **Πολεοδομία & Έξυπνες πόλεις** : Η Επαυξημένη Πραγματικότητα παρουσιάζει ενδιαφέρουσες δυνατότητες στα πεδία της πολεοδομίας και των έξυπνων πόλεων. Πλαισιώνοντας ψηφιακές πληροφορίες στον πραγματικό κόσμο, η Επαυξημένη Πραγματικότητα επιτρέπει την απρόσκοπτη ενσωμάτωση και ανάλυση της ψηφιακής νοημοσύνης στο φυσικό περιβάλλον διευκολύνοντας την λήψη αποφάσεων και τη συμμετοχή του κοινού στη κατασκευή και συντήρηση, ενισχύοντας την ευημερία της κοινότητας. Για τους πολεοδόμους, παρέχει ένα δυναμικό εργαλείο για την οπτικοποίηση αλλαγών σε τοπία, υποδομές και στη ροή της κυκλοφορίας και είναι ευκολότερο να αξιολογηθεί το αντίκτυπο των προτεινόμενων εξελίξεων.^[7]

Τα τελευταία χρόνια η τεχνολογία της Επαυξημένης Πραγματικότητας γνωρίζει σημαντική ανάπτυξη κι απευθύνεται κυρίως στις κινητές συσκευές. Με την χρήση της κάμερας ενός κινητού και σε αρκετές περιπτώσεις μέσω του συστήματος GPS δίνεται η δυνατότητα προβολής πληροφοριών, αντικειμένων ή εικονικών προσώπων που δεν ανταποκρίνονται στη πραγματικότητα, αλλά είναι σχεδιασμένα από υπολογιστές μέσω εφαρμογών. Χαρακτηριστικό παράδειγμα αποτελεί η εφαρμογή Unity. Η προαναφερθείσα προβολή δεν περιορίζεται μόνο στα κινητά τηλέφωνα αλλά έχει επεκταθεί σε πληθώρα συσκευών. Αξιόλογες κατηγορίες είναι οι παρακάτω :

Αρχικά μία από τις πιο γνωστές κατηγορίες επαυξημένης πραγματικότητας είναι τα Headsets & Smart Glasses. Είναι σχετικά μεγάλες συσκευές, πιο καθηλωτικές, που παρέχουν ένα ευρύτερο πεδίο προβολής (Field Of View) και πιο προηγμένες λειτουργίες. Συνήθως περιλαμβάνουν ενσωματωμένους αισθητήρες, κάμερες και συχνά αισθητήρες βάθους για τη χαρτογράφηση του περιβάλλοντος και την επικάλυψη 3D ψηφιακού περιεχομένου στον πραγματικό κόσμο. Οι συσκευές αυτές επιτρέπουν λειτουργία hands-free που είναι ιδιαίτερα χρήσιμο σε πεδία όπως η ιατρική, η κατασκευή και η συντήρηση που οι χρήστες χρειάζονται πρόσβαση σε πληροφορίες ενώ χρησιμοποιούν τα χέρια τους για άλλες ασχολίες. Ένα από τα πιο φημισμένα προϊόντα αυτής της κατηγορίας αποτελούν τα Hololens, προϊόν της ευρέως γνωστής εταιρείας Microsoft.^[7]

[7] Ανακτήθηκε από "Augmented Reality: Survey" Carlos E. Mendoza-Ramirez

Εξίσου σημαντική κατηγορία να αναφερθεί είναι τα Projection Based συστήματα. Οι συσκευές της κατηγορίας αυτής προβάλλουν τεχνητό φως σε επίπεδες επιφάνειες κι ανιχνεύουν την ανθρώπινη αλληλεπίδραση με αυτό. Αυτό το είδος τεχνολογίας της επαυξημένης πραγματικότητας δεν προϋποθέτει ο χρήστης να κρατά ή να φορά κάποια συσκευή, πράγμα που το διαφοροποιεί από τα άλλα είδη τεχνολογιών.[8] Κάποιες από τις projection based συσκευές επαυξημένης πραγματικότητας είναι το Celluon Epic το οποίο προβάλλει ένα εικονικό πληκτρολόγιο σε μία επίπεδη επιφάνεια και οι Interactive Projectors (Sony Xperia Touch και Lightform LF1).

Ένα όραμα για το μέλλον αποτελούν τα AR Contact Lenses. Θα λειτουργούν προβάλλοντας ψηφιακές εικόνες ή πληροφορίες στο οπτικό πεδίο του χρήστη απευθείας στον φακό επαφής. Από τεχνικής άποψης οι φακοί επαφής θα πρέπει να έχουν την δυνατότητα προβολής ψηφιακού περιεχομένου στον αμφιβληστροειδή, προσαρμογής οπτικών διακυμάνσεων στην απόσταση και ενδεχομένως να παρακολουθούν ακόμα και δείκτες υγείας. Αυτή η τεχνολογία βρίσκεται σε πρώιμα στάδια ανάπτυξης και παρουσιάζει πολλές τεχνολογικές προκλήσεις που πρέπει να αντιμετωπιστούν προτού γίνει εμπορικά βιώσιμο.[8]

1.3 ΠΕΔΙΟ - ΚΑΤΗΓΟΡΙΕΣ ΕΦΑΡΜΟΓΩΝ

Υπάρχουν διάφορες κατηγορίες επαυξημένης πραγματικότητας εμπειριών. Κάποιες από τις πιο συνηθισμένες είναι οι παρακάτω:

- 1. AR Face Filters :** Οι συγκεκριμένες εφαρμογές ανιχνεύουν πρόσωπα κι εμφανίζουν ψηφιακό περιεχόμενο πάνω σε αυτά. Εξαιρετικά γνωστά είναι τα φίλτρα που χρησιμοποιούνται στα μέσα κοινωνικής δικτύωσης (Snapchat, Instagram, TikTok).[9]



ΕΙΚΟΝΑ 2 : ΠΑΡΑΔΕΙΓΜΑ ΕΦΑΡΜΟΓΗΣ FACE FILTER[10]

- 2. Marker – based AR :** Οι εφαρμογές αυτού του τύπου εντοπίζουν ένα marker (μία συγκεκριμένη εικόνα, pattern ή QR Code) κι εμφανίζουν το συνθετικό περιεχόμενο επάνω σε αυτό. Αυτή η τεχνική χρησιμοποιείται συχνά για σκοπούς marketing (Pizza Hut Pac Man) καθώς και σε επαγγελματικές κάρτες για παροχή πρόσθετων πληροφοριών.[9]



ΕΙΚΟΝΑ 3 : PIZZA HUT ARCADE PAC MAN ΕΦΑΡΜΟΓΗ[11]

[9] Ανακτήθηκε από

<https://learn.unity.com/tutorial/welcome-to-the-course-2?pathwayId=63e3a4c1edbc2a344bfe21d8&missionId=63f63a3bedbc2a663dc6ffde#>

[10] Ανακτήθηκε από <https://www.perfectcorp.com/consumer/blog/selfie-editing/best-face-filter-apps>

[11] Ανακτήθηκε από

<https://blog.pizzahut.com/pizza-hut-serves-up-newstalgia-with-campaign-celebrating-all-that-fans-know-and-love-about-the-pizza-restaurant/>

3. Markerless AR : Σε αντίθεση με τις παραπάνω τεχνολογίες, οι εφαρμογές που χρησιμοποιούν markerless AR δεν ανιχνεύουν κάποιο πρόσωπο ή marker αντιθέτως χρησιμοποιούν πληροφορίες από το περιβάλλον για να εφαρμόσουν τις δυνατότητές τους. Παράδειγμα αυτής αποτελεί το Angry Birds: AR.[12]



EIKONA 4 : ANGRY BIRDS AR[13]

4. Location – based AR : Στην τέταρτη και τελευταία κατηγορία η εφαρμογή χρησιμοποιεί το GPS για να ανιχνεύσει την τοποθεσία του χρήστη και να εμφανίσει ψηφιακές πληροφορίες. Πολύ γνωστό παράδειγμα αυτής της κατηγορίας είναι το Pokemon Go.[12]



EIKONA 5 : POKEMON GO[14]

[12] Ανακτήθηκε από

<https://learn.unity.com/tutorial/welcome-to-the-course-2?pathwayId=63e3a4c1edbc2a344bfe21d8&missionId=63f63a3bedbc2a663dc6ffde#>

[13] Ανακτήθηκε από <https://poplar.studio/blog/why-mixed-reality-headsets-are-the-future-of-shopping/>

[14] Ανακτήθηκε από <https://playswap.gg/blog/pokemon-go-best-pokemon>

1.4 ΕΡΓΑΛΕΙΑ ΠΟΥ ΧΡΗΣΙΜΟΠΟΙΗΘΗΚΑΝ ΚΑΤΑ ΤΗΝ ΑΝΑΠΤΥΞΗ

1.4.1 UNITY ENGINE

Όπως αναφέρθηκε παραπάνω, οι εφαρμογές που χρησιμοποιούν την τεχνολογία της επαυξημένης πραγματικότητας και οι εικονικές πληροφορίες που εμφανίζονται κατά τη διάρκεια της χρήσης τους, μπορούν να δημιουργηθούν μέσω διαφόρων πλατφορμών, μία από τις οποίες είναι η πλατφόρμα Unity Engine.



EIKONA 6 : UNITY LOGO[15]

Σύμφωνα με την ίδια την εταιρία, η μηχανή τους (συγκεκριμένα η real-time 3D development engine) επιτρέπει σε καλλιτέχνες, σχεδιαστές και προγραμματιστές να συνεργάζονται για την δημιουργία καθηλωτικών και διαδραστικών εμπειριών σε δύο διαστάσεις (2D), σε τρεις (3D), Augmented και Virtual Reality (εικονική πραγματικότητα) εφαρμογές ακόμα και Mixed Reality (συνδυασμός AR-VR).^[16] Μερικά από τα πιο γνωστά παιχνίδια που έχουν δημιουργηθεί με τη χρήση της Unity είναι τα Among us, Pokemon Go κ.ά.^[17] Στην παρούσα εργασία χρησιμοποιήθηκε η **2021.3.13f1** έκδοση για την υλοποίηση της.

1.4.2 AR FOUNDATION

Ένα ιδιαίτερα σημαντικό εργαλείο του Unity για την δημιουργία AR εφαρμογών για κινητές συσκευές Android, IOS, AR wearable συσκευές όπως το Hololens είναι το AR Foundation (ΕΙΚΟΝΑ 7). Το συγκεκριμένο package προσφέρει στους προγραμματιστές ένα περιβάλλον που επιτρέπει τη χρήση σημαντικών χαρακτηριστικών από κάθε πλατφόρμα ενώ παράλληλα αξιοποιεί τις δυνατότητες του ίδιου του Unity.^[18]

[15] Ανακτήθηκε από https://en.wikipedia.org/wiki/Unity_%28game_engine%29

[16] Ανακτήθηκε από <https://unity.com/products/unity-engine>

[17] Ανακτήθηκε από <https://www.create-learn.us/blog/top-games-made-with-unity/>

[18] Ανακτήθηκε από “Augmented Reality with Unity AR Foundation” Jonathan Linowes

Συγκεκριμένα κάποιες σημαντικές δυνατότητες που υποστηρίζει το AR Foundation είναι οι ακόλουθες : (1) παρακολουθεί τη θέση και περιστροφή της συσκευής στον πραγματικό χώρο (Device Tracking), ανιχνεύει και παρακολουθεί (2) επίπεδες επιφάνειες (Plane Detection), (3) εικόνες δύο διαστάσεων (Image Tracking), (4) τρισδιάστατα αντικείμενα (Object Tracking), (5) ανθρώπινα πρόσωπα και σώματα (Face and Body Tracking) και (6) feature points (Point Clouds). Επιπλέον επιτρέπει (7) το Raycasting, (8) την ανίχνευση αυθαίρετων σημείων στο φυσικό χώρο (Anchors) και (9) την δημιουργία πλεγμάτων (Meshing).[19] Κατά αυτόν τον τρόπο κρίνεται απαραίτητη η ενσωμάτωση του συγκεκριμένου πακέτου για την ομαλή ανάπτυξη της εφαρμογής. Στην εν λόγω πτυχιακή χρησιμοποιήθηκε η **4.2.10** έκδοση.

1.4.3 ARCORE / ARKIT

Το ARCore (ΕΙΚΟΝΑ 7) αποτελεί ένα εργαλείο της Google που συνεργάζεται με διάφορες development πλατφόρμες, όπως το Unity, για την ανάπτυξη εφαρμογών επαυξημένης πραγματικότητας σε συσκευές android. Το αντίστοιχο εργαλείο για συσκευές IOS είναι το ARKit. Στην παρούσα εφαρμογή η οποία αναπτύχθηκε για συσκευή android χρησιμοποιήθηκε το ARCore. Ειδικότερα στηρίζεται σε τρεις βασικές δυνατότητες προκειμένου να συνδύασει ψηφιακά αντικείμενα με το φυσικό περιβάλλον μέσω της κάμερας της συσκευής.

- Motion Tracking : Επιτρέπει στη συσκευή να ανιχνεύει την θέση της αναλύοντας τον φυσικό κόσμο.[20]
- Κατανόηση περιβάλλοντος (αγγλικη ορολογία) : Δίνει την δυνατότητα στη συσκευή να προσδιορίζει το μέγεθος και τη θέση επιφανειών, τόσο κάθετων όσο και σε οριζόντιων.[20]
- Light estimation : Παρέχει την ικανότητα στην συσκευή να υπολογίσει τα χαρακτηριστικά της φωτεινότητας στο φυσικό χώρο.[20]

AR Foundation Release Unity Technologies Version 4.2.10 - January 23, 2024 Registry Unity <code>com.unity.xr.arfoundation</code> View documentation • View changelog • View licenses A collection of MonoBehaviours and C# utilities for working with AR Subsystems. Includes: • GameObject menu items for creating an AR setup • MonoBehaviours that control AR session lifecycle and create GameObjects from detected, real-world trackable features • Scale handling • Face tracking	ARCore XR Plugin Release Unity Technologies Version 4.2.10 - January 23, 2024 Registry Unity <code>com.unity.xr.arcore</code> View documentation • View changelog • View licenses Provides native Google ARCore integration for use with Unity's multi-platform XR API. Supports the following features: -Efficient Background Rendering -Horizontal Planes -Depth Data -Anchors -Hit Testing -Occlusion
---	--

EIKONA 7 : AR FOUNDATION PACKAGE (ΑΡΙΣΤΕΡΑ), ARCORE XR PLUGIN PACKAGE (ΔΕΞΙΑ)

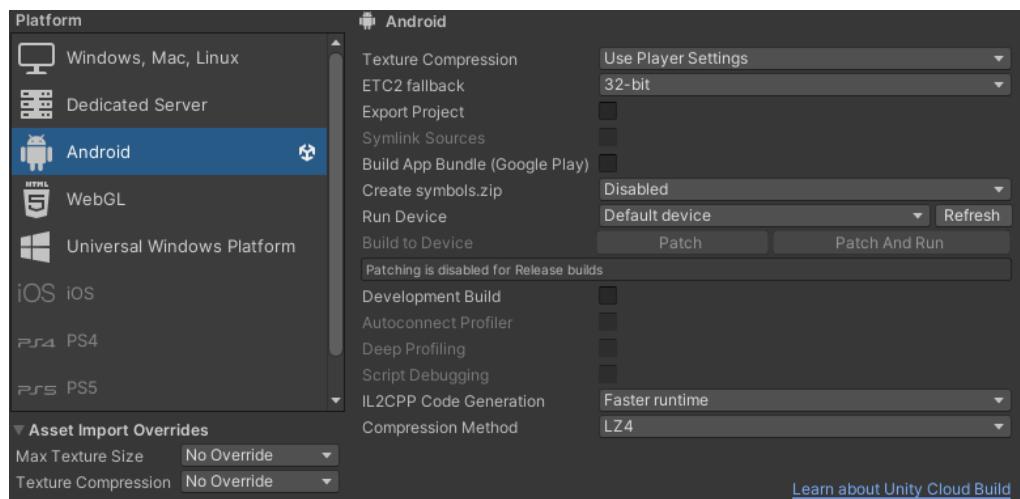
[19] Ανακτήθηκε από <https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@6.0/manual/index.html>

[20] Ανακτήθηκε από <https://developers.google.com/ar/develop>

1.5 ΡΥΘΜΙΣΕΙΣ ΣΥΣΤΗΜΑΤΟΣ

Πριν από την ανάλυση της εφαρμογής είναι σημαντικό να αναφερθούν συνοπτικά οι ρυθμίσεις που χρησιμοποιήθηκαν κατά τη δημιουργία της εφαρμογής (ΕΙΚΟΝΑ 8).

- Platform : Android
- Texture Compression : Use Player Settings
- ETC2 fallback : 32-bit
- IL2Cpp Code Generation : Faster Runtime
- Compression Method LZ4



EIKONA 8 : BUILD SETTINGS

Επιπλέον παρατίθενται ορισμένες σημαντικές ρυθμίσεις από τα “Player Settings”.

Other Settings:

- Rendering : Linear
- Auto Graphics API : uncheck
- Graphics APIs : OpenGLES3
- Multithreaded Rendering : check
- Minimum API Level : Android 7.0 ‘Nougat’ (API level 24)
- Scripting Backend : IL2CPP

ΚΕΦΑΛΑΙΟ 2 : ΤΟ ΑΡΧΙΚΟ SCENE

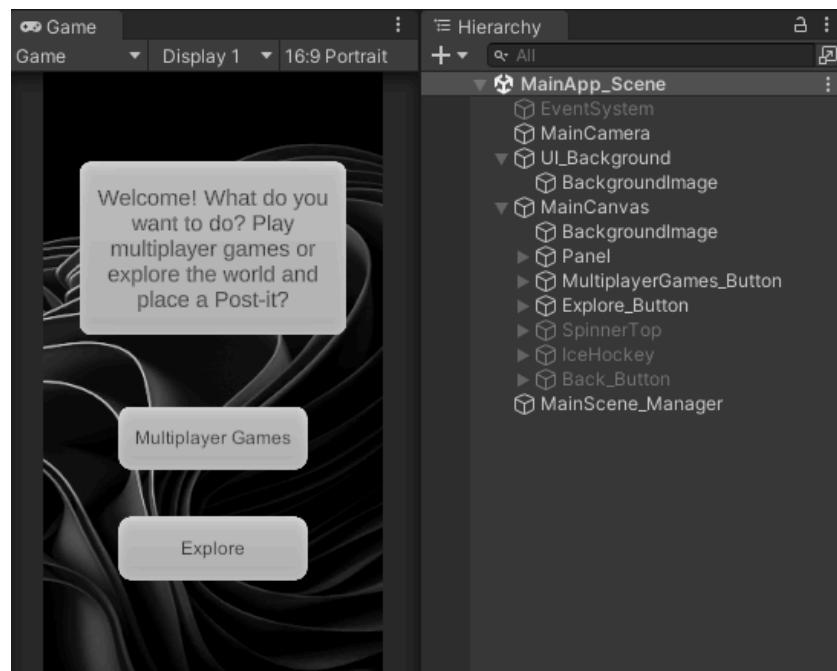
Η εφαρμογή ξεκινάει με το αρχικό scene (MainApp_Scene) όπως φαίνεται στην εικόνα 9 το οποίο αποτελείται από την Main Camera, το Event System το οποίο είναι υπεύθυνο για την διαχείριση των γεγονότων που συμβαίνουν στο εκάστοτε scene, από τον καμβά μας (Main Canvas) ο οποίος περιέχει όλα τα στοιχεία του UI (User Interface) τα οποία είναι απαραίτητα για το συγκεκριμένο scene και από το MainScene_Manager το οποίο περιέχει το Main Scene Manager Script που ελέγχει την συμπεριφορά του scene.

Main Canvas : Πρόκειται για ένα UI object που περιλαμβάνει το background image, ένα panel για την εμφάνιση μηνυμάτων στον χρήστη καθώς και πέντε κουμπιά (buttons).

-Explore_Button : Με το πάτημα του κουμπιού αυτού, ο χρήστης μεταφέρεται στην αντίστοιχη σκηνή.

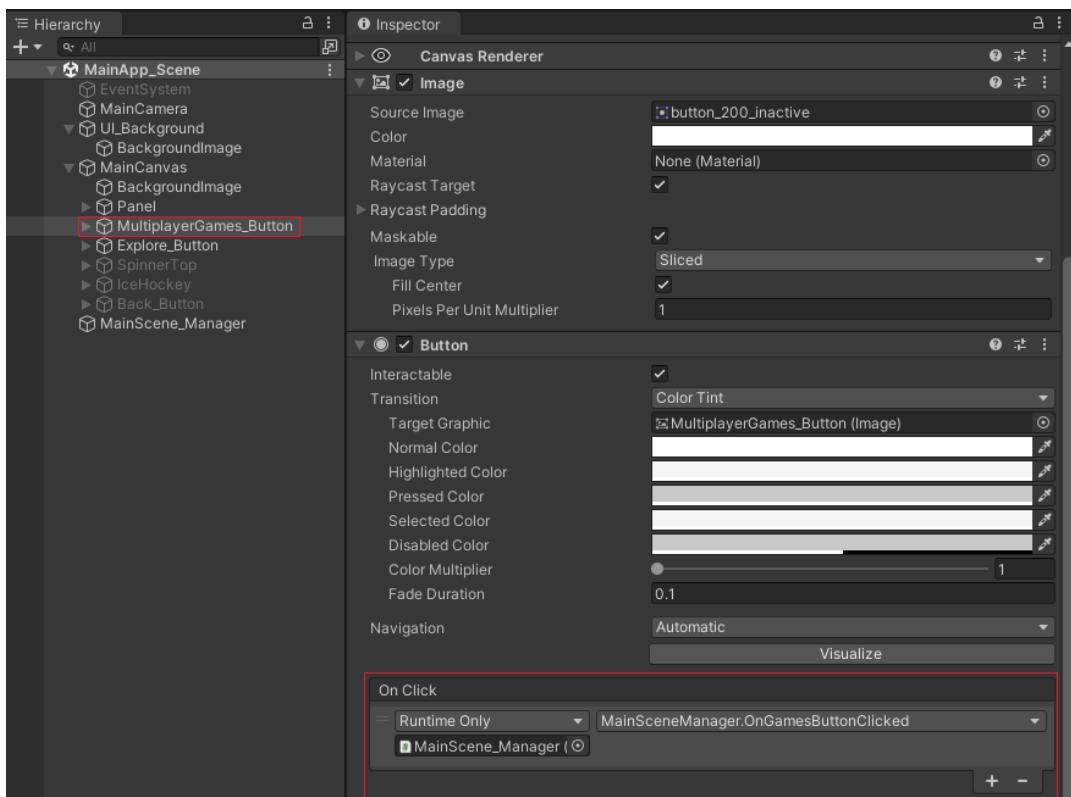
-MultiplayerGames_Button : Με την συγκεκριμένη επιλογή, το κουμπί αυτό απενεργοποιείται “οπτικά”, ομοίως και το Explore_Button, ενώ παράλληλα εμφανίζονται τα **SpinnerTop** και **IceHockey** που κατευθύνουν τον χρήστη στα ανάλογα παιχνίδια. Επιπρόσθετα εμφανίζεται το κουμπί επιστροφής (Back_Button).

-Back_Button : Βρίσκεται στη πάνω αριστερά πλευρά της οθόνης και επιτρέπει στον χρήστη να αναιρέσει αυτήν την ενέργεια επανεμφανίζοντας τα δύο κύρια κουμπιά του scene (MultiplayerGames, Explore).



EIKONA 9 : MAINAPP_SCENE HIERARCHY

Όλες οι λειτουργίες που αφορούν τα κουμπιά και το panel μηνυμάτων ελέγχονται από ένα csharp script το οποίο είναι τοποθετημένο ως component στο MainScene_Manager. Στο εν λόγω script ορίζονται ως public μεταβλητές τα GameObject που χρησιμοποιούνται (buttons, panel, texts) και συντάσσονται οι συναρτήσεις που καθορίζουν τη συμπεριφορά τους. Στη συνέχεια για την ενεργοποίηση της λειτουργίας αυτής εφόσον πατηθεί κάποιο button, μεταβαίνουμε στο κάθε ένα από αυτά και στο component “Button” στην επιλογή “On Click” τοποθετούμε το MainScene_Manager object επιλέγοντας τη συνάρτηση που θα εκτελείται κάθε φορά που το button πατιέται.

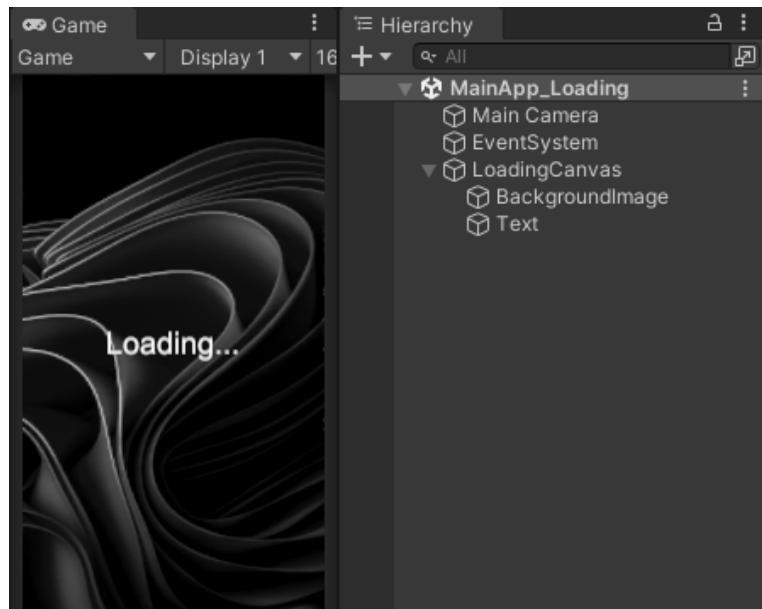


ΕΙΚΟΝΑ 10 : ΠΑΡΑΔΕΙΓΜΑ ΣΥΝΔΕΣΗΣ BUTTON ME SCRIPT

Χαρακτηριστικό παράδειγμα αποτελεί η εικόνα 10, καθώς στο inspector του MultiplayerGames_Button, έχει τοποθετηθεί το MainScene_Manager object όπου λαμβάνει αυτόματα το script που περιέχει κι έχει επιλεχθεί να εκτελεί τη συνάρτηση “OnGamesButtonClicked()”.

2.1 LOADING SCENE KAI MAIN APP SCENE LOADER

Όταν επιλεγεί μία από τις MultiplayerGames ή Explore επιλογές, εμφανίζεται για λίγα δευτερόλεπτα ένα ενδιάμεσο scene, το MainApp_Loading (ΕΙΚΟΝΑ 11) πριν εμφανιστεί το επόμενο scene.



EIKONA 11 : MAINAPP_LOADING SCENE

Η μετάβαση αυτή επιτυγχάνεται μέσω του MainAppSceneLoader script το οποίο παρουσιάζει μία ιδιαιτερότητα στον κώδικά του. Η κλάση αρχικοποιείται με την εξής δήλωση:

```
public class MainAppSceneLoader : Singleton<MainAppSceneLoader>
```

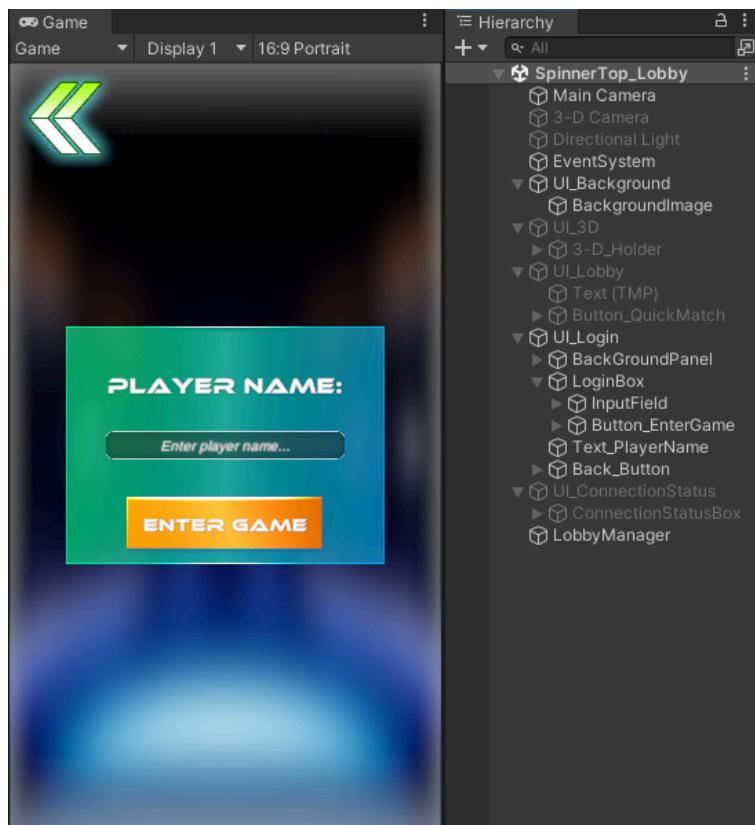
Η χρήση του Singleton εξασφαλίζει ότι θα υπάρχει μόνο ένα instance (αντίγραφο) της συγκεκριμένης κλάσης κατά τη διάρκεια λειτουργίας της εφαρμογής ανεξάρτητα από το πόσες φορές θα προσπαθήσει η εφαρμογή να τη αναδημιουργήσει. Συγκεκριμένα η κλάση του script κληρονομεί από τη γενική κλάση Singleton<T> η οποία φροντίζει να εφαρμόσει τη συγκεκριμένη συμπεριφορά όπως προαναφέρθηκε.[21]

ΚΕΦΑΛΑΙΟ 3 : SPINNER TOP MULTIPLAYER GAME

Ένα από τα δύο multiplayer games που προσφέρονται στην εφαρμογή είναι το SpinnerTop στο οποίο κάθε παίκτης αφού εισάγει το ψευδώνυμό του και επιλέξει μία από τις τέσσερις διαθέσιμες σβούρες, εισέρχεται σε μία πίστα και περιμένει τον αντίπαλο του για να αναμετρηθούν. Το παιχνίδι προορίζεται για δύο παίκτες όπου ο καθένας διαθέτει ταχύτητα περιστροφής (spinspeed) και νικητής είναι εκείνος που θα σταματήσει τη περιστροφή της σβούρας του άλλου. Στη συνέχεια θα αναφερθούν οι λεπτομέρειες σχετικά με το παιχνίδι και την υλοποίησή του. Στο σημείο αυτό πρέπει να αναφερθεί ότι τα γραφικά αντικείμενα (assets) όπως και τα αντικείμενα της επιφάνειας εργασίας (εικόνες, γραμματοσειρές, κουμπιά, πάνελ κτλ) του παιχνιδιού, είναι δημιούργημα της IRONHEAD Games.[22]

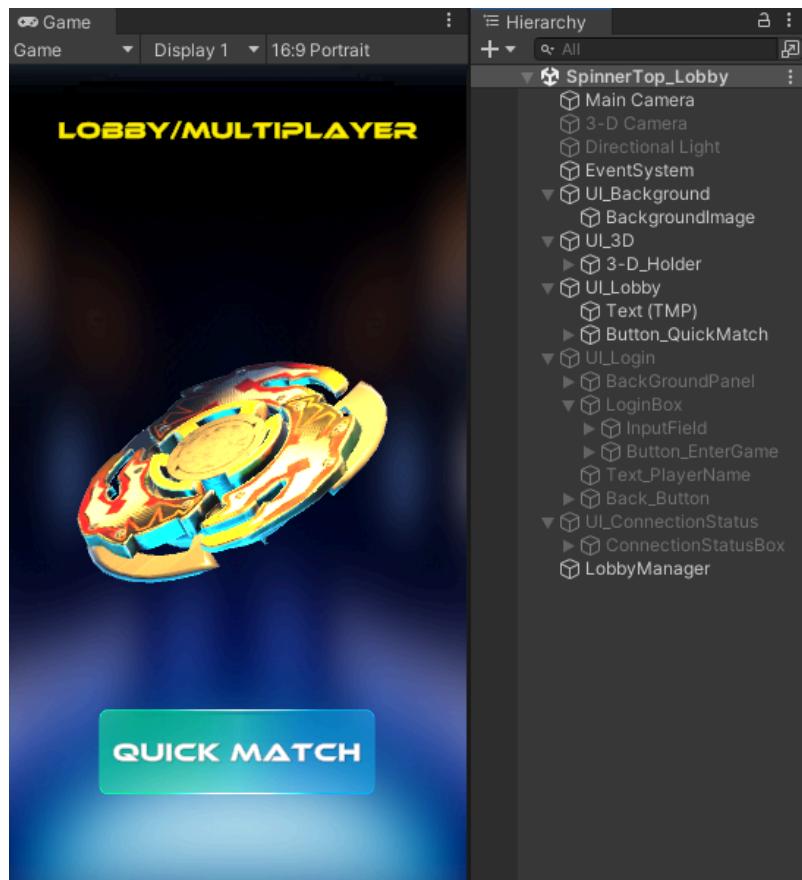
3.1 LOBBY SCENE

Το πρώτο scene που αντικρίζει ο χρήστης όταν επιλέξει το SpinnerTop (μετά το scene_loading) είναι το SpinnerTop_Lobby όπως φαίνεται στην εικόνα 12. Στο scene αυτό υπάρχει το Back_Button το οποίο επιστρέφει τον χρήστη στο MainApp_Scene, καθώς κι ένα panel που περιέχει το input field για την εισαγωγή του ονόματος του χρήστη. Επιπλέον συναντάται το Button_EnterGame που μεταφέρει τον χρήστη στο επόμενο βήμα μόλις αυτός είναι έτοιμος.



EIKONA 12 : SPINNER TOP LOGIN PHASE

Στο επόμενο στάδιο του παιχνιδιού όπως παρατηρείται στην εικόνα 13 εμφανίζεται μια σβούρα που περιστρέφεται σε πραγματικό χρόνο καθώς και το Button_QuickMatch που οδηγεί τον χρήστη στο επόμενο scene που θα κληθεί να επιλέξει τον τύπο και το skin της σβούρας.

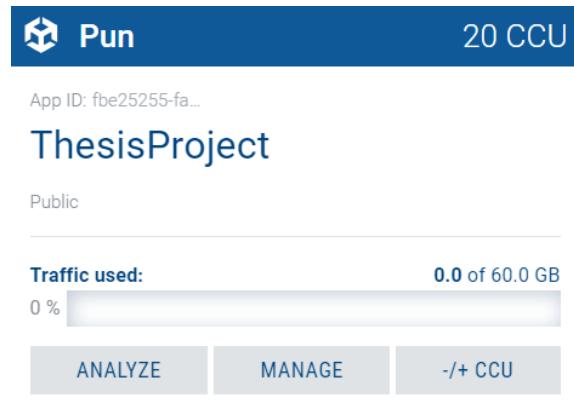


EIKONA 13 : SPINNER TOP LOBBY PHASE

3.2 PHOTON UNITY NETWORKING (PUN)^[23]

Πριν προχωρήσουμε στο επόμενο scene κρίνεται απαραίτητο να αναφερθεί μία σημαντική λεπτομέρεια, το PHOTON. Το Photon Unity Networking είναι ένα επιπρόσθετο (plugin) εργαλείο του Unity, που αναπτύχθηκε από την Exit Games και χρησιμοποιείται για την δημιουργία multiplayer παιχνιδιών. Το photon παρέχει servers (στη συγκεκριμένη περίπτωση χρησιμοποιήθηκε server των είκοσι ατόμων όπως φαίνεται στην εικόνα 14) καθώς και την βιβλιοθήκη photon.run που περιέχει τις απαραίτητες εντολές για την δημιουργία του multiplayer περιβάλλοντος.

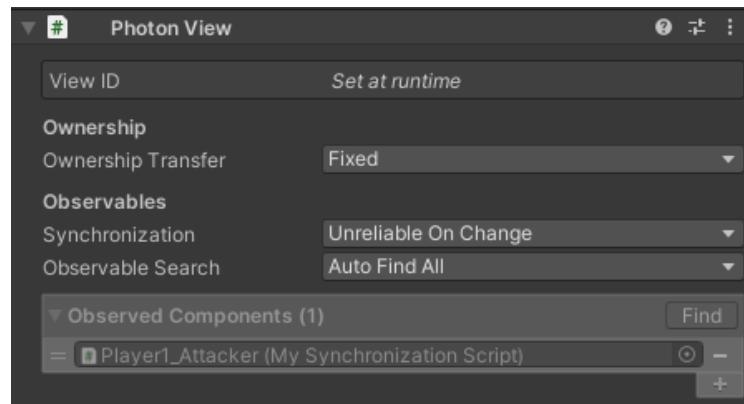
[23] Ανακτήθηκε από <https://doc.photonengine.com/pun/current/getting-started/pun-intro>



ΕΙΚΟΝΑ 14 : PHOTON PUN APP REGISTRATION

Χρησιμοποιούμε την εντολή `PhotonNetwork.LocalPlayer.Nickname()` για την σύνδεση των παικτών ενώ για την δημιουργία του δωματίου χρησιμοποιείται η `PhotonNetwork.JoinOrCreateRoom()`. Έτσι ο πρώτος παίκτης που θα εισέλθει στο παιχνίδι, εισάγει το ψευδώνυμό του, επιλέξει τον τύπο σβούρας που επιθυμεί και προχωρά στο επόμενο βήμα. Στη συνέχεια δημιουργείται ένα «δωμάτιο» όπου θα βρεθεί ο ίδιος και κατ' επέκταση ο αντίπαλός του.

Εφόσον το παιχνίδι είναι multiplayer σημαίνει ότι οι δύο παίκτες θα πρέπει να βλέπουν την ίδια εικόνα από τη συσκευή τους, γεγονός που απαιτεί τον συγχρονισμό των δεδομένων των δύο παικτών. Ο κάθε χρήστης μέσω του Photon στέλνει τα realtime δεδομένα του στον άλλο χρήστη και λαμβάνει του δικά του ως δεδομένα του remote client. Όλα επιτυγχάνονται μέσω ενός component της photon, το Photon View (ΕΙΚΟΝΑ 15) κι ενός script συγχρονισμού (MySynchronizationScript) το οποίο είναι επίσης δημιούργημα της IRONHEAD Games.[24] Σε συνέχεια η κάθε σβούρα έχει ένα photon view component που περιέχει το μοναδικό του id και το script συγχρονισμού κι έτσι τα δεδομένα του ενός μεταφέρονται στον άλλον κι αντίστροφα.

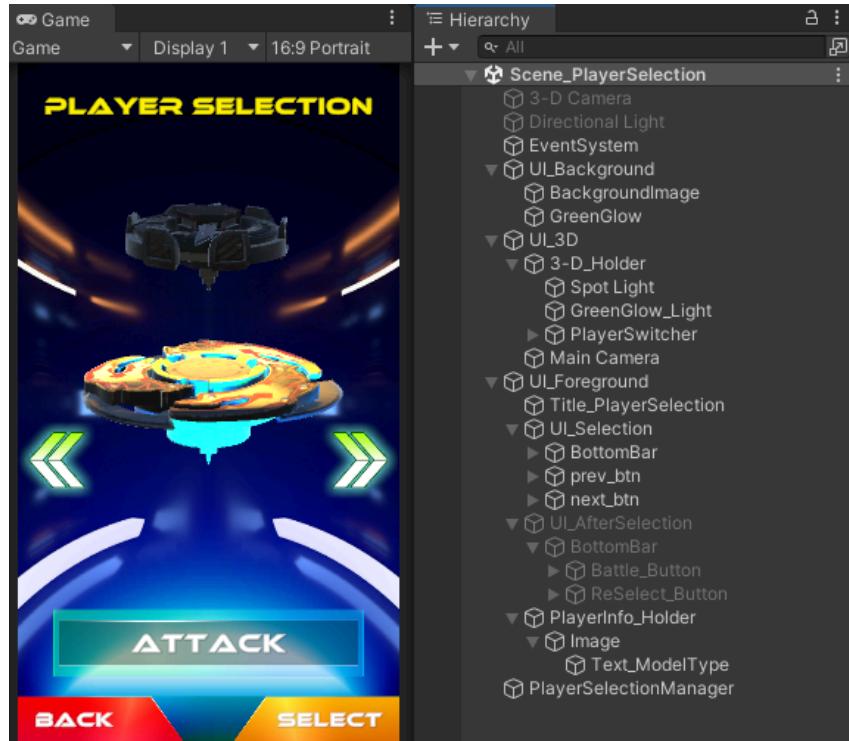


ΕΙΚΟΝΑ 15 : PHOTON VIEW COMPONENT

[24] Ανακτήθηκε από <https://assetstore.unity.com/packages/templates/tutorials/photon-multiplayer-ar-beyblade-ar-157814>

3.3 PLAYER SELECTION SCENE

Μετά την επιλογή του Quick Match button η εφαρμογή μεταφέρεται στο Scene_PlayerSelection όπως φαίνεται στην εικόνα 16 όπου ο παίκτης καλείται να επιλέξει τον τύπο και το χρώμα της σβούρας του.



EIKONA 16 : SPINNER TOP PLAYERSELECTION UI_SELECTION

Υπάρχουν δύο είδη σβούρας, τα attack και defend. Μεταξύ τους οι διαφορές που εντοπίζονται είναι οι εξής (ΕΙΚΟΝΑ 17):

- Οι attack σβούρες προκαλούν περισσότερο «damage» από τις defend.
- Οι defend σβούρες απορροφούν λιγότερο «damage» από τις attack.
- Οι defend σβούρες έχουν μεγαλύτερη ταχύτητα περιστροφής (spinspeed) που σημαίνει ότι απαιτείται περισσότερη ζημιά για να σταματήσει η περιστροφή τους (3600 spinspeed οι attack, 4400 οι defend).

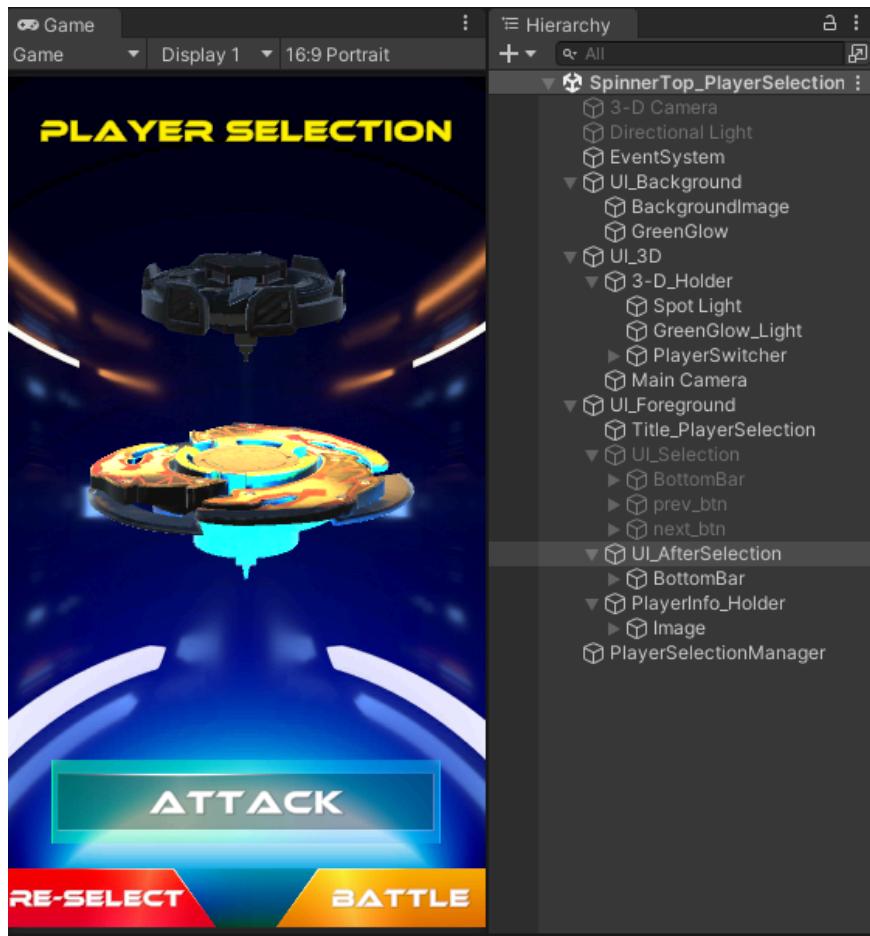
```
[Header("Player Type Damage Coefficients")]
public float doDamage_Coefficient_Attacker = 10f; //do more damage than defender- ADVANTAGE
public float getDamaged_Coefficient_Attacker = 1.2f;// gets more damage - DISADVANTAGE

public float doDamage_Coefficient_Defender = 0.75f; //do less damage- DISADVANTAGE
public float getDamaged_Coefficient_Defender = 0.2f; //gets less damage - ADVANTAGE
```

EIKONA 17 : ΔΙΑΦΟΡΑ ΜΕΤΑΞΥ ATTACK – DEFEND

Στο scene εμφανίζονται οι διαθέσιμες σβούρες οι οποίες περιστρέφονται. Πιο συγκεκριμένα παρατηρούνται δύο βέλη για την αλλαγή των σβουρών, ένα panel που αναγράφει τον τύπο της σβούρας και τα “Back” και “Select” buttons. Το “Back” επιστρέφει τον χρήστη στο Scene_Lobby ενώ το “Select” κλειδώνει τη σβούρα που έχει επιλέξει ο χρήστης.

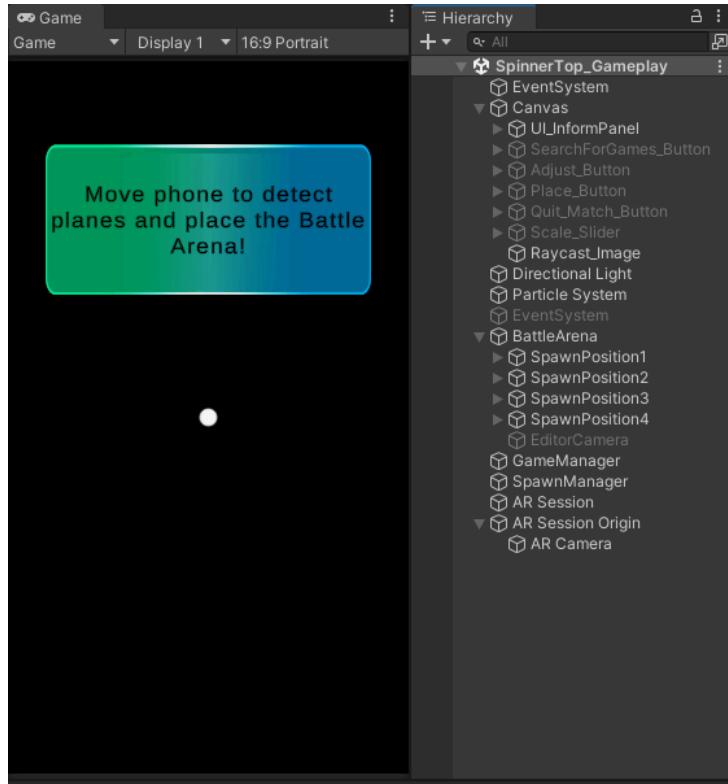
Ακολούθως αφού κλειδωθεί η επιλογή της σβούρας, τα “Re-Select” και “Battle” buttons αντικαθιστούν τα προηγούμενα (ΕΙΚΟΝΑ 18). Η εφαρμογή παρέχει τη δυνατότητα να αλλάξει η σβούρα μέσω του “Re-Select” ή να οδηγηθεί προς τη «μάχη» με το “Battle”.



EIKONA 18 : SPINNER TOP PLAYER SELECTION UI_AFTERSELECTION

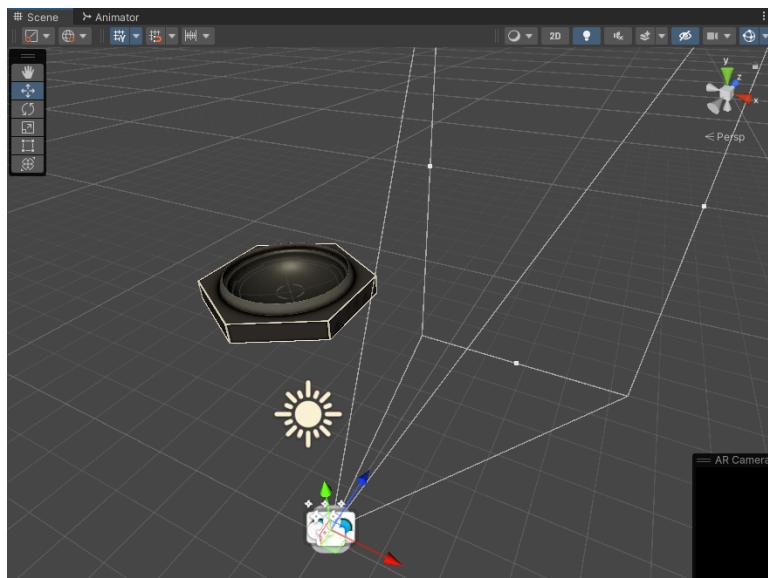
3.4 GAMEPLAY SCENE

Στη φάση αυτή η εφαρμογή μεταβαίνει στη χρήση της κάμερας της συσκευής όπου ο χρήστης βλέπει τον φυσικό χώρο μέσω αυτής (ΕΙΚΟΝΑ 19).



ΕΙΚΟΝΑ 19 : SPINNER TOP GAMEPLAY SCENE

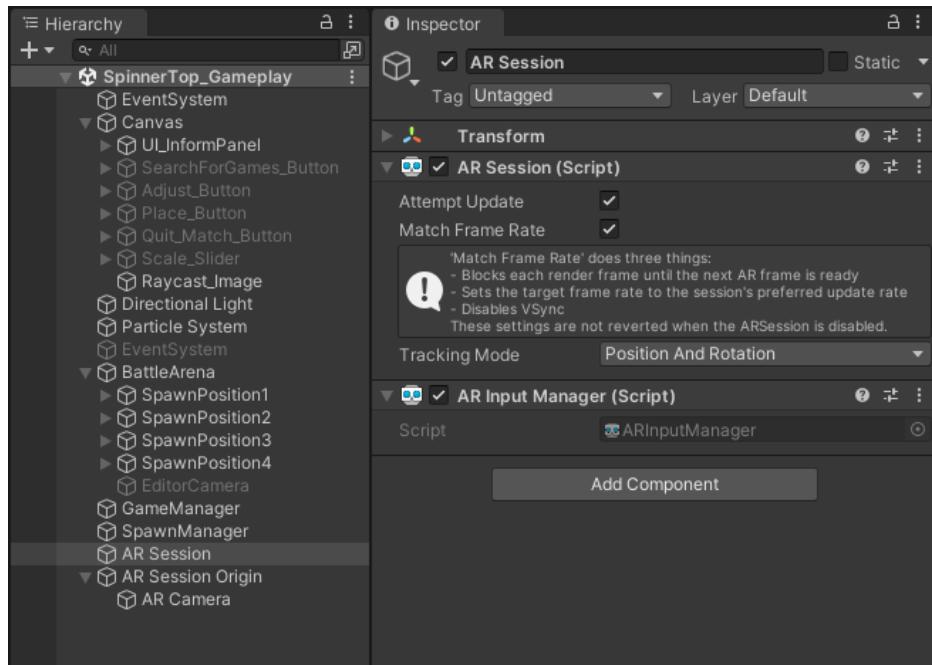
Αρχικά στην οθόνη εμφανίζεται ένας καμβάς (canvas) που περιλαμβάνει ένα πάνελ πληροφοριών (UI_InformPanel) και μία raycast εικόνα (η λευκή τελεία στο κέντρο της οθόνης) η οποία βοηθά στην τοποθέτηση της Battle Arena. Η αρένα είναι αρχικά αόρατη όπως φαίνεται στην εικόνα 20 και γίνεται ορατή μόλις η εφαρμογή ανιχνεύσει μία επίπεδη επιφάνεια (plane).



ΕΙΚΟΝΑ 20 : Η ΘΕΣΗ ΤΗΣ BATTLE ARENA ΠΡΙΝ ΓΙΝΕΙ ΟΡΑΤΗ ΣΤΟΝ ΧΡΗΣΤΗ

3.4.1 AR SESSION KAI AR SESSION ORIGIN

Για να υλοποιήση της λειτουργίας της επαυξημένης πραγματικότητας απαιτείται η χρήση συγκεκριμένων objects στο scene. Το πρώτο είναι το **AR Session** (ΕΙΚΟΝΑ 21) το οποίο διαχειρίζεται τον «κύκλο ζωής» της AR εμπειρίας ενεργοποιώντας κι απενεργοποιώντας το AR σύστημα. Περιέχει το AR Input Manager component το οποίο επιτρέπει την ανίχνευση του χώρου σε πραγματικό χρόνο.[25]



EIKONA 21 : AR SESSION

Το δεύτερο είναι το **AR SESSION ORIGIN** το οποίο περιλαμβάνει όλα τα απαραίτητα components για την τοποθέτηση της arena στο χώρο όπως φαίνεται στην εικόνα 22. Το AR Session Origin είναι ο «γονιός» της εγκατάστασης της επαυξημένης πραγματικότητας και έχει ως public παράμετρο την κάμερα που χρησιμοποιείται από την εφαρμογή.[26] Το AR Plane Manager είναι το εργαλείο που ανιχνεύει και διαχειρίζεται τις επιφάνειες (planes). Έχει ως public παράμετρο το AR Default Plane το οποίο καθορίζει τον τρόπο εμφάνισης των επιφανειών στην εφαρμογή (χρώμα, διαφάνεια κτλ) καθώς και το είδος που είναι επιθυμητό να ανιχνευθούν. Στην προκειμένη περίπτωση αναζητούνται επίπεδες οριζόντιες επιφάνειες για την τοποθέτηση της arena οπότε επιλέγεται Horizontal detection.[27]

[25] Ανακτήθηκε από

<https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.2/api/UnityEngine.XR.ARFoundation.ARSession.html>

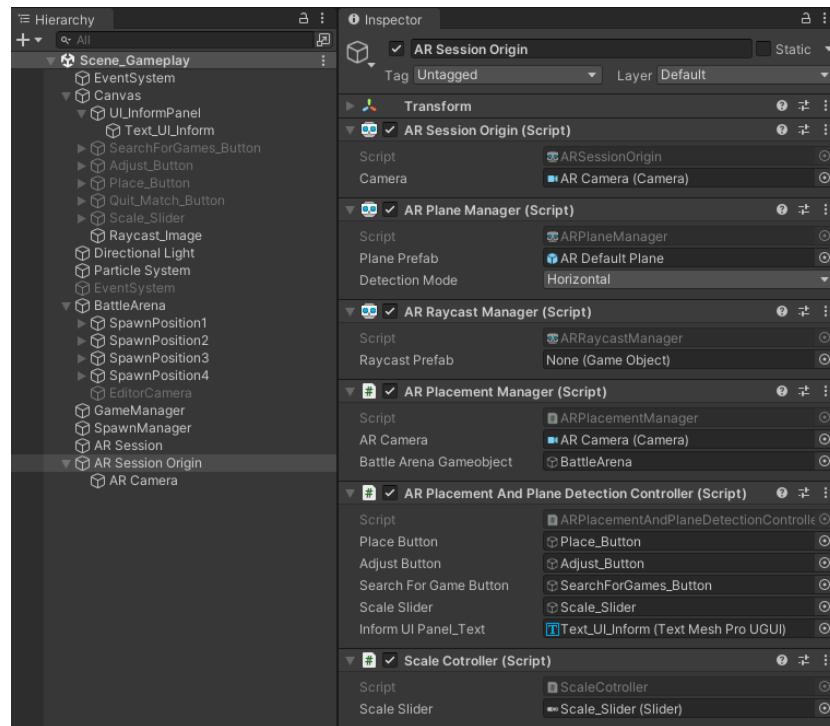
[26] Ανακτήθηκε από

<https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.2/api/UnityEngine.XR.ARFoundation.ARSessionOrigin.html>

[27] Ανακτήθηκε από

<https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.2/api/UnityEngine.XR.ARFoundation.ARPlaneManager.html>

Επιπλέον στο session origin έχουν προστεθεί διάφορα scripts για την διαχείριση τοποθέτησης της arena (AR Placement Manager), τον έλεγχο της συμπεριφοράς του UI πριν και μετά την τοποθέτηση της arena (AR Placement and PlaneDetection Controller) και τέλος ένα script (Scale Controller) που δίνει την δυνατότητα στον χρήστη να προσαρμόσει το μέγεθος της arena ώστε να ταιριάζει καλύτερα στον διαθέσιμο χώρο.



EIKONA 22 : AR SESSION ORIGIN

3.4.2 ARENA PLACEMENT

Αφού η εφαρμογή ανιχνεύσει μία επίπεδη επιφάνεια (plane), η arena τοποθετείται αυτόματα πάνω της χωρίς όμως να κλειδώνει η θέση της. Ύστερα ο χρήστης έχει την δυνατότητα να τοποθετήσει την arena σε οποιοδήποτε σημείο του ανιχνευθέντος plane. Στην οθόνη εντοπίζεται μία μπάρα μεγέθυνσης (scale slider) το οποίο προσαρμόζει το μέγεθος της arena ώστε να χωρά σε διαφορετικές επιφάνειες καθώς και το "Place" button. Έπειτα πατώντας το συγκεκριμένο κουμπί, η arena κλειδώνει στη θέση που έχει επιλέξει ο χρήστης. Μόλις τοποθετηθεί η arena, το "Place" button απενεργοποιείται κι εμφανίζεται το "Adjust". Συγκεκριμένα δίνει στον χρήστη την ευκαιρία να αλλάξει ξανά τη θέση της arena πριν ξεκινήσει το παιχνίδι. Ταυτόχρονα εμφανίζεται το "Search For Games" button το οποίο όταν πατηθεί τοποθετεί τη σβούρα του παίκτη στην arena αναμένοντας την άφιξη του αντιπάλου. Παράδειγμα αυτών αποτελεί η εικόνα 23.



ΕΙΚΟΝΑ 23 : ΟΘΟΝΗ ΠΡΙΝ (ΑΡΙΣΤΕΡΑ) ΚΑΙ ΜΕΤΑ (ΔΕΞΙΑ) ΤΟ PLACE

3.5 ARENA SETUP

Αναφορικά με την arena γίνονται αντιληπτά τέσσερα Spawn Positions τα οποία χρησιμοποιούνται στο Spawn Manager object και το Spawn Manager script που ελέγχει τη δημιουργία (Instantiate) των παικτών μέσα στην arena. Ειδικότερα το script λαμβάνει τις πληροφορίες του κάθε παίκτη, δηλαδή τον τύπο της σβούρας, ένα από τα τέσσερα spawn positions μέσω τυχαίας συνάρτησης (Random.Range(0, spawnPositions.Length - 1)) και τις συντεταγμένες περιστροφής (Quaternion rotation) της σβούρας. Στη συνέχεια δημιουργεί τη σβούρα μέσα στην arena μέσω της συνάρτησης Instantiate() και λαμβάνει το photonview component για τον συγχρονισμό των δεδομένων όπως αναφέρθηκε προηγουμένως (ΕΙΚΟΝΑ 24).

```
GameObject playerGameObject = Instantiate(playerPrefabs[(int)playerSelectionNumber], instantiatePosition, Quaternion.identity);
PhotonView _photonView = playerGameObject.GetComponent<PhotonView>();
```

ΕΙΚΟΝΑ 24 : INSTANTIATION ΤΟΥ ΠΑΙΚΤΗ ΚΑΙ ΛΗΨΗ ΤΟΥ PHOTONVIEW

3.6 BATTLE SCRIPT

Η λειτουργία του παιχνιδιού βασίζεται στο **Battle Script**, το οποίο είναι ενσωματωμένο σε κάθε σβούρα και περιέχει τις απαραίτητες οδηγίες για τη συμπεριφορά της κατά τη διάρκεια του παιχνιδιού.

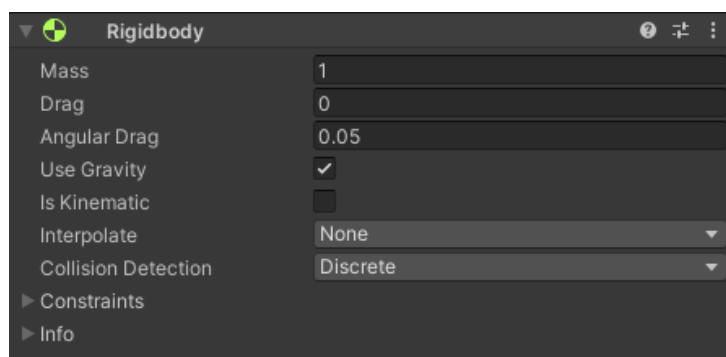
-Μεταβλητές :

Αρχικά υπάρχει σαν μεταβλητή το **spinner script** το οποίο είναι επίσης ενσωματωμένο στη σβούρα, καθορίζει την ταχύτητα περιστροφής της κι ενημερώνεται διαρκώς καθώς επηρεάζεται από τις ενέργειες στο παιχνίδι.

Επίσης λαμβάνει το **Rigidbody**^[28] component της σβούρας το οποίο επιτρέπει τη φυσική συμπεριφορά στη σβούρα, όπως αλλαγές στο position και στο rotation της αν δεχτεί χτυπήματα από άλλη σβούρα ή από το τοίχωμα της arena (ΕΙΚΟΝΑ 25). Έχει τις εξής ιδιότητες:

[28] Ανακτήθηκε από <https://docs.unity3d.com/2021.3/Documentation/Manual/class-Rigidbody.html>

- Mass : Ορίζει την μάζα του αντικειμένου (της σβούρας) σε κιλά.[29]
- Drag : Καθορίζει τον ρυθμό της μείωσης της γραμμικής ταχύτητας για να προσομοιώσει το σύρσιμο (drag) της σβούρας, την αντίσταση του αέρα και την τριβή (friction).[29]
- Angular Drag : Προσδιορίζει το ρυθμό μείωσης της περιστροφικής ταχύτητας.[29]
- Gravity : Ορίζει αν το αντικείμενο επηρεάζεται από την βαρύτητα.[29]
- Is Kinematic : Όταν είναι ενεργοποιημένο, το αντικείμενο δεν επηρεάζεται από φυσική δύναμη αλλά μόνο μέσω κώδικα (Transform).[29]
- Interpolate : Επηρεάζει την ομαλότητα της κίνησης του αντικειμένου. Έχει επιλεχθεί το none.[29]
- Collision Detection : Ελέγχει τη συμπεριφορά του αντικειμένου όταν έρχεται σε επαφή (collision) με άλλα αντικείμενα.[29]
- Constraints : Επιτρέπει στον προγραμματιστή να «παγώσει» συντεταγμένες του position ή του rotation.[29] Στη συγκεκριμένη περίπτωση έχουν παγώσει το rotation και στις τρεις συντεταγμένες (x, y, z) καθώς και περιστροφή της σβούρας καθορίζεται από script.



EIKONA 25 : RIGIDBODY COMPONENT

To Battle script περιέχει επίσης τις παρακάτω μεταβλητές. Εν πρώτοις η αρχική κι η τρέχουσα ταχύτητα της σβούρας ώστε να μπορεί να την ανανεώνει. Δύο boolean μεταβλητές που καθορίζουν αν είναι τύπου attack ή defend, μία μεταβλητή float που είναι το κοινό damage που προκαλούν οι σβούρες η οποία μεταβάλλεται ανάλογα με τον τύπο τους. Τέλος υπάρχει κι ένα panel object (Death Panel) σαν μεταβλητή όπου εμφανίζεται στον παίκτη που έχει εξοντωθεί μαζί με ένα χρονόμετρο των έξι δευτερολέπτων που καθορίζει τον χρόνο αναμονής για την επανεμφάνισή του.

-Συναρτήσεις :

- Awake() : Αρχικοποιεί την αρχική και την τρέχουσα ταχύτητα περιστροφής της σβούρας.
- CheckPlayerType() : Ελέγχει τον τύπο σβούρας κι αν είναι τύπου defend αλλάζει την ταχύτητα.
- OnCollisionEnter() : Αν σβούρα έρθει σε επαφή με την άλλη επηρεάζεται η θέση της. Επίσης ορίζονται οι damage μεταβλητές που θα επηρεάσουν την ταχύτητα των σβουρών και τέλος καλείται η συνάρτηση DoDamage με RPC (Remote Procedure Calls) όπου είναι μία μέθοδος της Photon για την συμπεριφορά σύγκρουσης δύο remote client.
- DoDamage() : Υπολογίζει τη ζημιά που θα δεχτούν οι σβούρες κι ελέγχει αν η ταχύτητα είναι μικρότερη από ένα ποσό (100) κι αν είναι καλεί την Die συνάρτηση.
- Die() : Απενεργοποιεί το joystick του παικτη ώστε να μην μπορεί να κουνήσει την σβούρα και ξεκινάει τη διαδικασία του respawn καλώντας την συνάρτηση ReSpawn.
- ReSpawn() : Ενεργοποιεί το Death Panel (ΕΙΚΟΝΑ 26) κι αφού τελειώσουν τα οκτώ δευτερόλεπτα, το απενεργοποιεί, ενεργοποιεί το joystick και καλεί την ReBorn συνάρτηση πάλι με τη μέθοδο RPC.
- ReBorn() : Ορίζει την ταχύτητα της σβούρας στην αρχική τιμή και μπορεί να ξεκινήσει η μάχη εκ νέου.



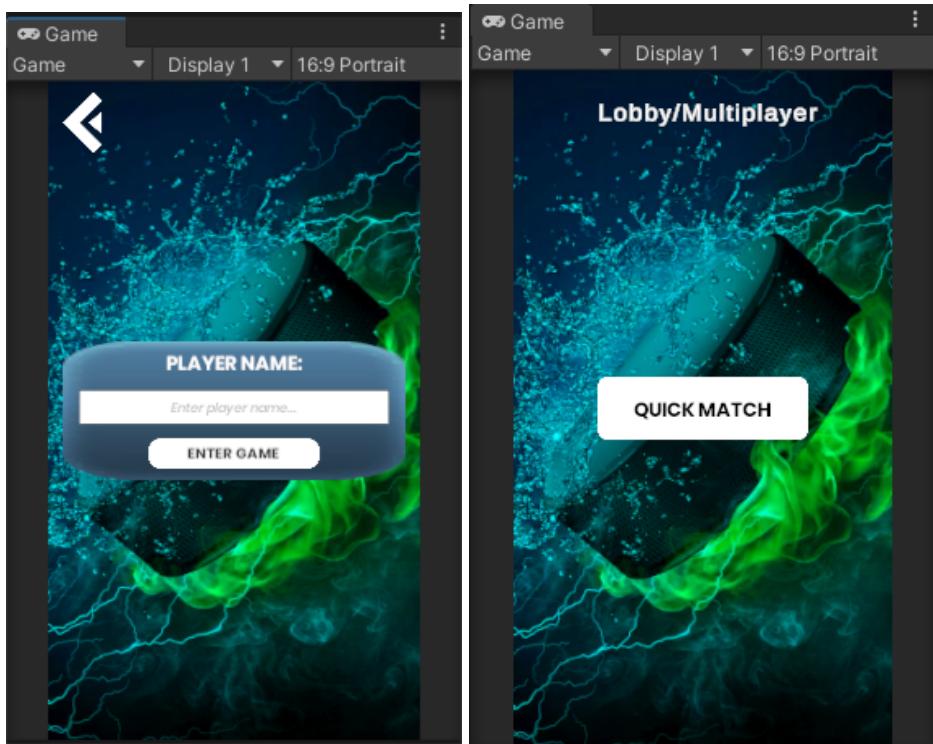
EIKONA 26 : DEATH PANEL

ΚΕΦΑΛΑΙΟ 4 : ICE HOCKEY MULTIPLAYER GAME

Το δεύτερο παιχνίδι της εφαρμογής είναι το κλασσικό arcade Ice Hockey το οποίο περιλαμβάνει δύο παίκτες, μία μπάλα και ένα γήπεδο. Στόχος του κάθε παίκτη είναι να σημειώσει περισσότερα τέρματα από τον αντίπαλο μέσα σε τρία λεπτά. Από προγραμματιστικής άποψης, δεν έχει πολλές διαφορές με το Spinner Top. Ας το αναλύσουμε εκτενέστερα.

4.1 ICE HOCKEY LOBBY

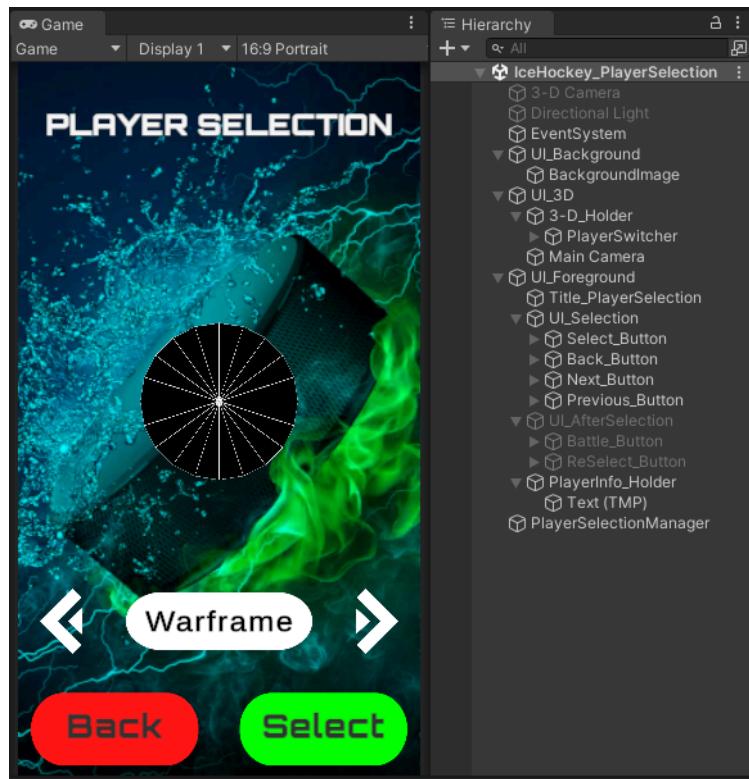
Όπως στο SpinnerTop έτσι και σε αυτό υπάρχει ένα background σχετικό με το παιχνίδι, ένα πάνελ με ένα input field για την εισαγωγή του ονόματος του χρήστη, ένα “Enter Game” button και τέλος το “Back” button (στο επάνω αριστερά μέρος της οθόνης) το οποίο επιστρέφει τον χρήστη στη κύρια σκηνή της εφαρμογής (MainApp_Scene). Μετά την εισαγωγή του ονόματος και την επιλογή του “Enter Game”, εμφανίζεται το “Quick Match” button που οδηγεί τον παίκτη στο Player Selection του παιχνιδιού (ΕΙΚΟΝΑ 27).



ΕΙΚΟΝΑ 27 : ICE HOCKEY LOGIN (ΑΡΙΣΤΕΡΑ) ICE HOCKEY LOBBY (ΔΕΞΙΑ)

4.2 ICE HOCKEY PLAYER SELECTION

Σε αυτό το scene (ΕΙΚΟΝΑ 28) ο χρήστης επιλέγει το skin που θα φέρει ο παίκτης του. Τα skins δεν προσφέρουν ιδιαίτερα πλεονεκτήματα πέρα από την αισθητική διαφοροποίηση. Υπάρχουν πέντε skin, το Warframe, το Lava, το Brick, το Metal και το Plasma, υλικά (materials) από το PBR Materials Sampler Pack της Integrity Software & Games^[30] και το Stylized Lava Materials του Rob Iuο^[31]. Υπάρχει επίσης το “Select” button το οποίο επιβεβαιώνει την επιλογή του skin καθώς και το “Back” που επιστρέφει τον χρήστη στο Lobby. Μετά την επιλογή του skin, εμφανίζονται τα “ReSelect” και “Score” buttons όπου το πρώτο δίνει την δυνατότητα στον παίκτη να επαναλάβει την επιλογή skin ενώ το δεύτερο τον προωθεί στο επόμενο scene.



EIKONA 28 : ICE HOCKEY PLAYER SELECTION

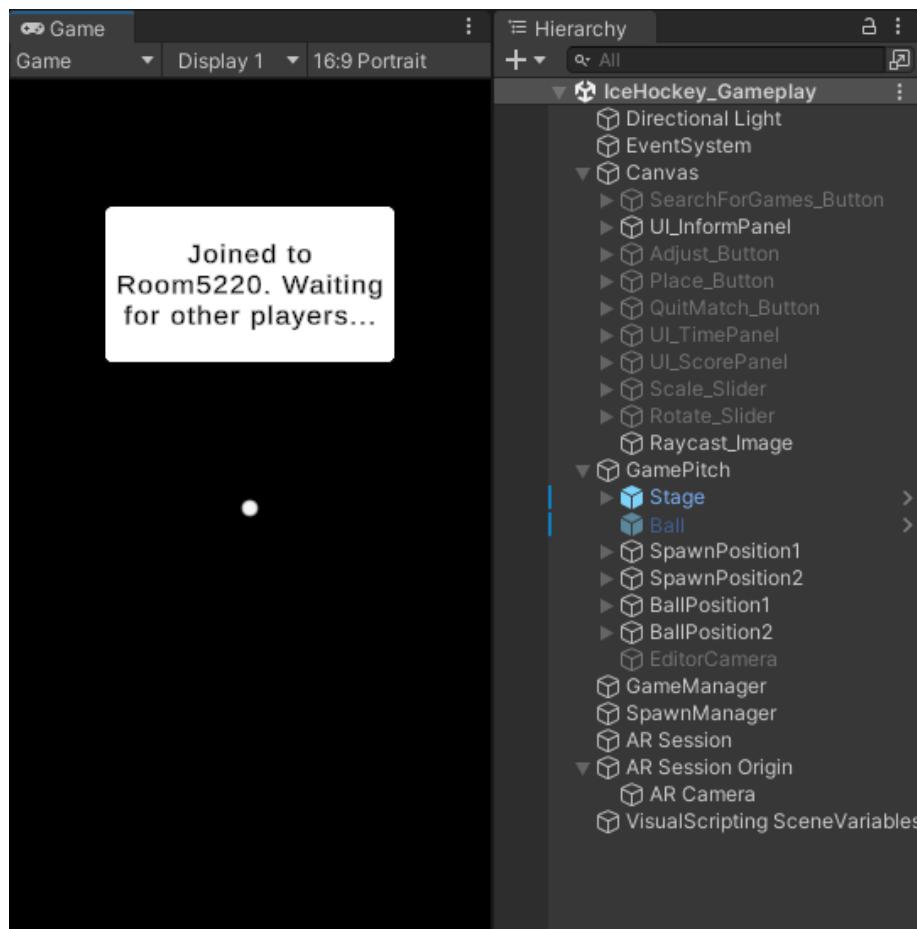
[30] Ανακτήθηκε από <https://assetstore.unity.com/packages/2d/textures-materials/pbr-materials-sampler-pack-40112>

[31] Ανακτήθηκε από <https://assetstore.unity.com/packages/2d/textures-materials/stylized-lava-materials-180943>

4.3 ICE HOCKEY GAMEPLAY SCENE

Στην κύρια σκηνή του παιχνιδιού (ΕΙΚΟΝΑ 29), όπως και στο Spinner Top, περιλαμβάνονται τα AR Session και AR Session Origin. Στο Session Origin ενσωματώνονται τα εξής : AR Session Origin, AR Plane Manager, AR Raycast Manager, τα scripts για την ανίχνευση των planes, την ενεργοποίηση και απενεργοποίηση των UI objects. Επιπρόσθετα περιλαμβάνεται ένα Scale Controller script και σε αντίθεση με το Spinner Top, υπάρχει ένα Rotate Controller όπου ενδείκνυται να το χρησιμοποιήσει ο δεύτερος παίκτης προκειμένου να ρυθμίσει την οπτική γωνία του για καλύτερη εμπειρία.

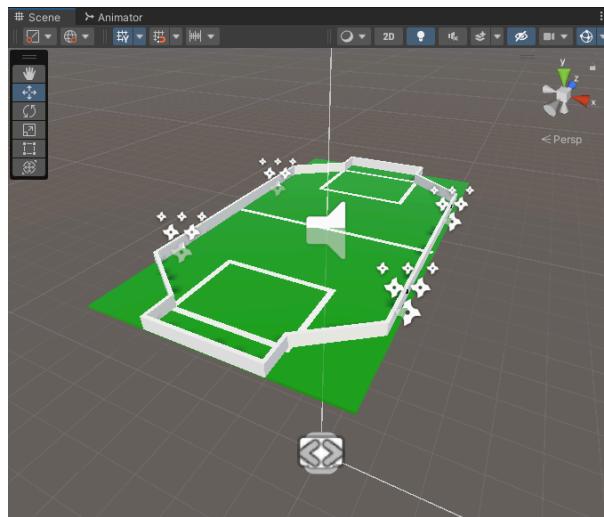
Επιπλέον υπάρχει ένα Canvas object που περιέχει όλα τα UI components δηλαδή το panel με τις πληροφορίες, καθώς και τα “Place”, “Adjust”, “Search For Games” και “Quit Match” buttons που έχουν την ίδια ακριβώς λειτουργία με το Spinner Top. Το UI περιλαμβάνει επίσης τα Scale και Rotate sliders και ένα scoreboard panel που δείχνει το χρόνο (timer) και το score του παιχνιδιού.



EIKONA 29 : ICE HOCKEY GAMEPLAY SCENE HIERARCHY

4.3.1 GAME PITCH

Το γήπεδο του Ice Hockey αποτελεί μία απλή κατασκευή που δημιουργήθηκε εξ ολοκλήρου στο Unity με 3D objects. Το γήπεδο, όπως και η arena στο Spinner Top, πριν τοποθετηθεί στο plane που θα ανιχνευθεί βρίσκεται πάνω από την κάμερα του χρήστη χωρίς να είναι εμφανής όπως απεικονίζεται στην εικόνα 30.

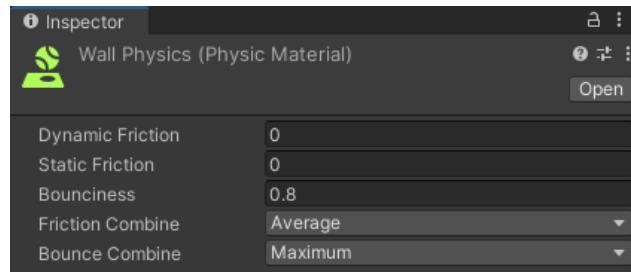


ΕΙΚΟΝΑ 30 : Η ΘΕΣΗ ΤΟΥ ΓΗΠΕΔΟΥ ΠΡΙΝ ΓΙΝΕΙ ΟΡΑΤΗ ΣΤΟΝ ΧΡΗΣΤΗ

Τα 3D objects τα οποία χρησιμοποιούνται για τους τοίχους του γηπέδου περιέχουν collider για να επιτρέπει την επαφή και σύγκρουση με την μπάλα και τους παίκτες. Στο κέντρο και στις εστίες του γηπέδου υπάρχει ένας τοίχος ο οποίος είναι τοποθετημένος λίγο πιο ψηλά από την πράσινη επιφάνεια και δεν είναι εμφανής. Συγκεκριμένα ο τοίχος αυτός διαθέτει ενεργοποιημένο collider ώστε να αποτρέπει τη διέλευση του παίκτη στην μεριά του αντιπάλου επιτρέποντας το μόνο στη μπάλα. Επιπλέον μέσα στις δύο εστίες υπάρχουν δύο πολύ λεπτές επιφάνειες πράσινου χρώματος πάνω από την ήδη υπάρχουσα επιφάνεια που χρησιμεύουν στην λειτουργία του γκολ. Επίσης στις παραμέτρους του collider υπάρχουν ενσωματωμένα physics materials τα οποία αναλόγως τις τιμές έχουν και την ανάλογη συμπεριφορά στην επαφή με τα άλλα αντικείμενα. Οι παράμετροι αυτών των materials περιλαμβάνουν τα παρακάτω (ΕΙΚΟΝΑ 31) :

- **Dynamic Friction** : Η συμπεριφορά τριβής όταν το αντικείμενο κινείται. Παίρνει τιμές από 0 έως 1. Αν είναι 0 δεν υπάρχει τριβή (σαν να κινείται πάνω σε πάγο) κι αν είναι 1 η τριβή είναι πολύ υψηλή.[32]
- **Static Friction** : Καθορίζει τη τριβή όταν το αντικείμενο ακίνητο.[32]
- **Bounciness** : Ορίζει πόσο αναπηδά το αντικείμενο. Αν η τιμή είναι 0 το αντικείμενο δεν αναπηδά κι αν είναι 1 θα αναπηδήσει χωρίς να χάσει ενέργεια από την σύγκρουση με το άλλο αντικείμενο.[32]
- **Friction Combine** : Ορίζει τη τιμή τριβής που προκύπτει από τη σύγκρουση των δύο αντικειμένων.[32]
- **Bounce Combine** : Προσδιορίζει τη τιμή αναπήδησης από τα δύο αντικείμενα που συγκρούονται.[32]

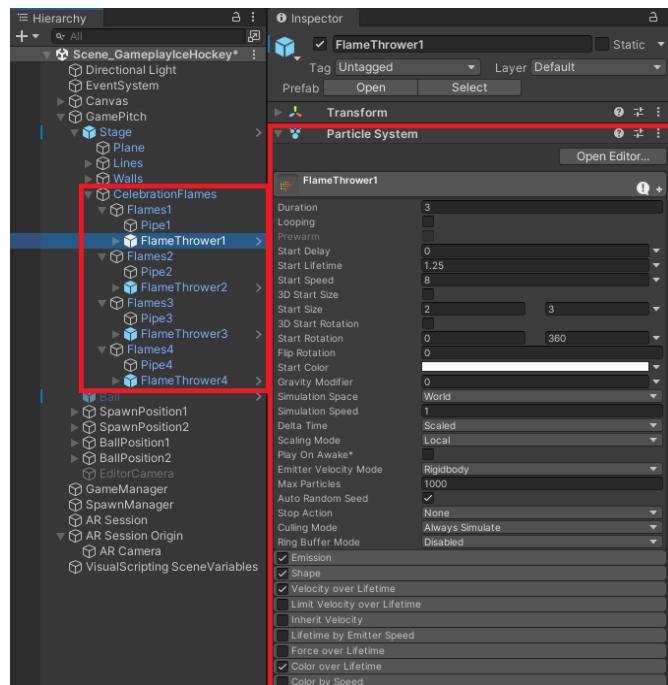
[32] Ανακτήθηκε από <https://docs.unity3d.com/Manual/class-PhysicsMaterial.html>



ΕΙΚΟΝΑ 31 : ΟΙ ΠΑΡΑΜΕΤΡΟΙ ΓΙΑ ΤΑ PHYSICS ΤΩΝ WALLS

Στο γήπεδο υπάρχουν δύο Spawn Positions, μία για κάθε εστία. Επεξηγηματικά στο script για το spawn των παικτών, δεν εφαρμόζεται η random.range() αλλά ο πρώτος παίκτης τοποθετείται στο πρώτο Spawn position και ο δεύτερος στο άλλο. Επιπλέον υπάρχουν και δύο θέσεις για την μπάλα (Ball Positions) καθεμία τοποθετημένη στο κέντρο του γηπέδου, στις δύο πλευρές.

Επιπλέον υπάρχουν και δύο flamethrower αντικείμενα στη κάθε μεγάλη πλευρά του γηπέδου που έχουν ενσωματωμένα particle systems (ΕΙΚΟΝΑ 32). Το Particle system προσομοιώνει ρευστές οντότητες όπως υγρά, σύννεφα και φλόγες. Στην περίπτωσή μας προσομοιώνονται φλόγες οι οποίες ενεργοποιούνται όταν κάποιος παίκτης σκοράρει συνοδευόμενες από έναν ήχο. Αυτά που χρησιμοποιήθηκαν στην εφαρμογή είναι το Particle Pack, απότοκο της Unity Technologies.[33]

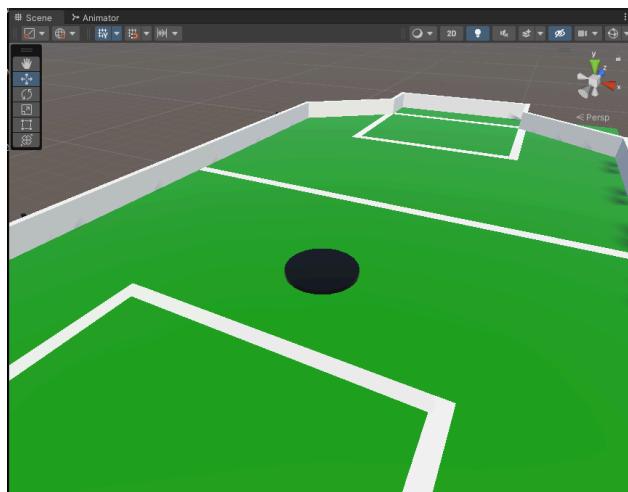


ΕΙΚΟΝΑ 32 : PARTICLE SYSTEM

[33] Ανακτήθηκε από <https://assetstore.unity.com/packages/vfx/particles/particle-pack-127325>

4.3.2 BALL PREFAB

Στη μία από τις δύο πλευρές του γηπέδου υπάρχει η μπάλα (ΕΙΚΟΝΑ 33) η οποία είναι αρχικά αόρατη κι ενεργοποιείται μόλις μπουν κι οι δύο παίκτες μέσα στο γήπεδο. Περιέχει ένα RigidBody και ένα Mesh Collider όπως και οι παίκτες, ένα Audio Source για να παίζει τον ήχο όταν μπαίνει γκολ, ένα Photon View κι ένα Synchronization Script για να γίνεται σωστά ο συγχρονισμός και να βλέπουν και οι δύο παίκτες την μπάλα στο ίδιο σημείο και τέλος ένα script (Ball Behaviour) που είναι η συμπεριφορά της μπάλας.

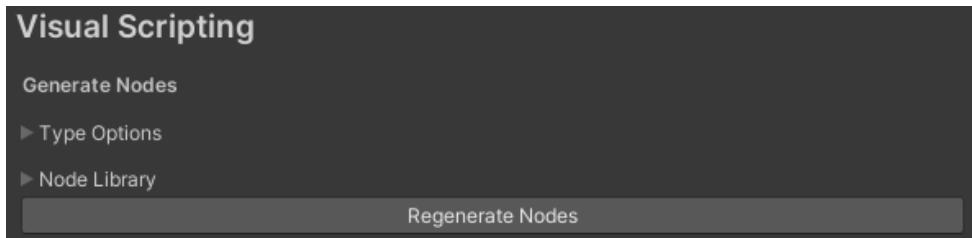


ΕΙΚΟΝΑ 33 : BALL PREFAB

Επεξηγηματικά το Ball Behaviour script ελέγχει αν η μπάλα έχει έρθει σε επαφή με μία από τις δύο επιφάνειες των εστιών, με κάποιον παίκτη ή κάποιον τοίχο. Αν έρθει σε επαφή με παίκτη ή κάποιον τοίχο ακούγεται ένας ήχος (ding) κι αν έρθει σε επαφή με κάποια από τις επιφάνειες, αναλόγως ποιου παίκτη είναι, αυξάνει το σκορ του αλλουνού. Στη συνέχεια απενεργοποιείται η κίνηση της μπάλας και ξεκινάει η διαδικασία επανατοποθέτησής της στη μεριά αυτού που δέχτηκε το γκολ. Τέλος υπάρχει μία boolean μεταβλητή wasGoal που την καθιστά αληθή και χρησιμοποιείται από το visual script “Goal Setup”.

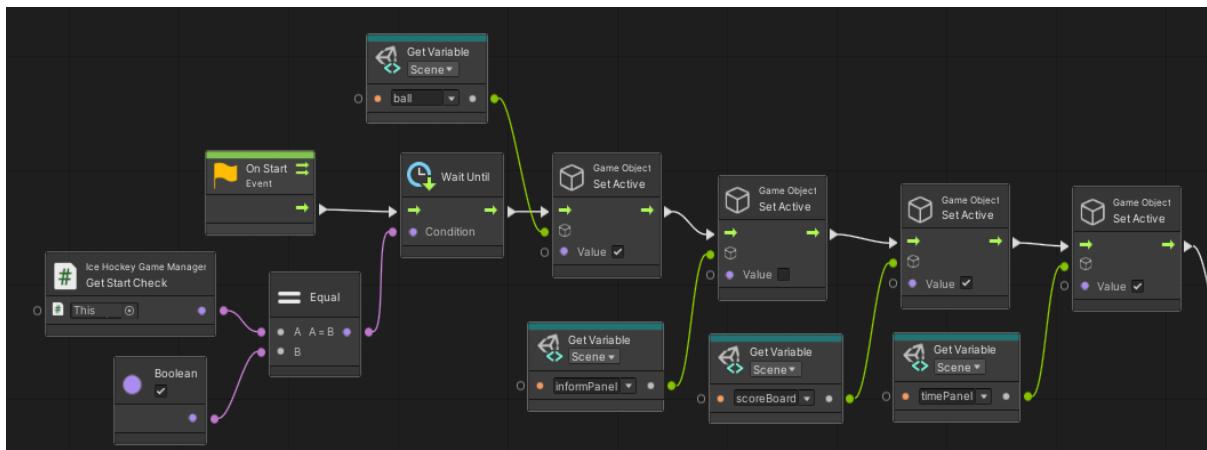
4.4 GAME SETUP

Για τη διάταξη του παιχνιδιού και του γκολ, εκτός από κώδικα σε C#, χρησιμοποιήθηκε οπτικός προγραμματισμός (Visual Scripting). Αποτελεί μία τεχνική προγραμματισμού όπου ο προγραμματιστής δεν γράφει απευθείας κώδικα, αλλά αντ' αυτού χρησιμοποιεί κόμβους (nodes). Τα nodes είναι γραφικά στοιχεία τα οποία αναπαριστούν συναρτήσεις, μεταβλητές, λογικά σύμβολα κ.α. και συνδέονται μεταξύ τους μέσω ακμών. Με τον τρόπο αυτό προσφέρουν μία οπτική αναπαράσταση της ροής του προγράμματος αντί για γραμμές κώδικα.^[34] Για την χρήση του visual scripting είναι απαραίτητο να ενεργοποιηθεί στα “Project Settings” ενώ ταυτόχρονα προσφέρεται η δυνατότητα προσθήκης νέων κόμβων μέσω του “Node Library” όπως φαίνεται στην εικόνα 34.



EIKONA 34 : NODE LIBRARY

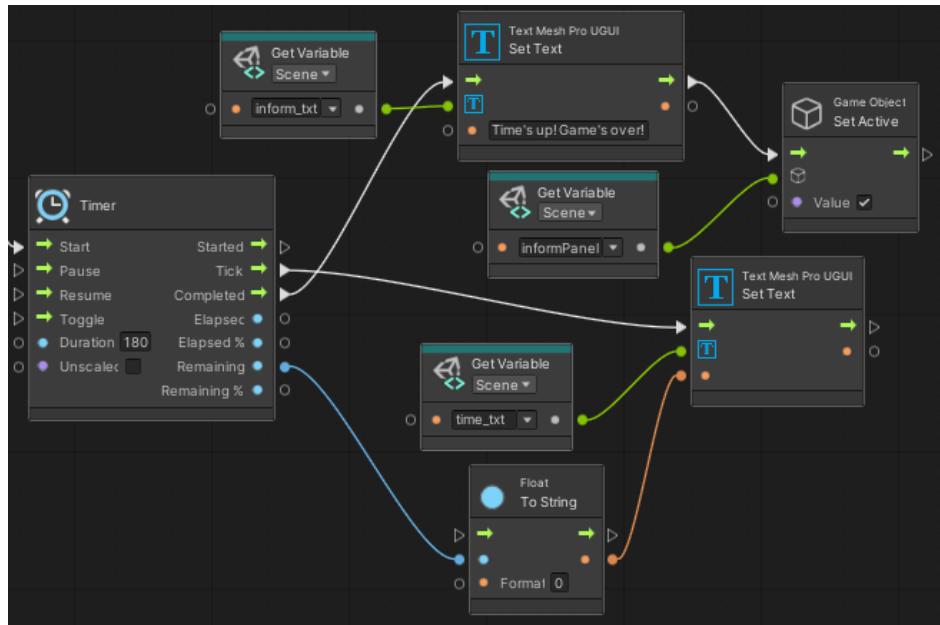
Στο συγκεκριμένο σχεδιασμό χρησιμοποιήθηκε κώδικας για την διαδικασία εμφάνισης των παικτών στο γήπεδο. Μόλις εμφανιστούν και οι δύο παίκτες η boolean μεταβλητή StartCheck τίθεται αληθής. Από το σημείο αυτό κι έπειτα η υλοποίηση του παιχνιδιού πραγματοποιείται με τη χρήση visual scripting.



EIKONA 35 : GAME SETUP ON START NODE

[34] Ανακτήθηκε από <https://docs.unity3d.com/Packages/com.unity.visualscripting@1.9/manual/index.html>

Η εικόνα 35 απεικονίζει τμήμα του γεγονότος “On Start”. Χρησιμοποιείται ο Wait Until κόμβος μέχρι να λάβει αληθή τιμή η StartCheck μεταβλητή που αναφέρθηκε προηγουμένως. Μόλις αυτό συμβεί, ενεργοποιείται η μπάλα, το scoreboard και το timer ενώ απενεργοποιείται το info panel.



ΕΙΚΟΝΑ 36 : TIMER NODE

Ακριβώς μετά ακολουθεί ο κόμβος “Timer”, γεγονός που τεκμηριώνεται στην εικόνα 36. Στο Duration ορίζεται η επιθυμητή διάρκεια του αγώνα η οποία για τους σκοπούς του παιχνιδιού καθορίστηκε στα τρία λεπτά. Το Tick flow (το flow αντιπροσωπεύεται από τα πράσινα βέλη) εκτελείται κάθε δευτερόλεπτο ανανεώνοντας το χρονομετρητή στον πίνακα του σκορ.

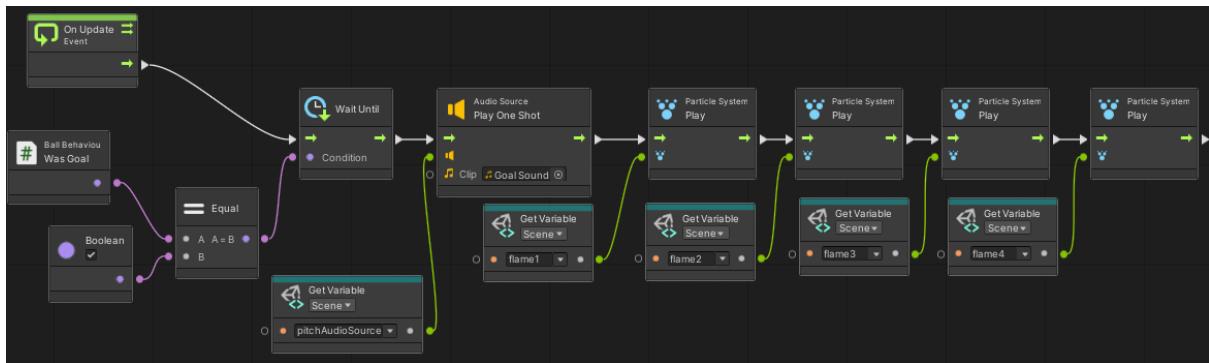


ΕΙΚΟΝΑ 37 : GAME SETUP ON UPDATE NODE

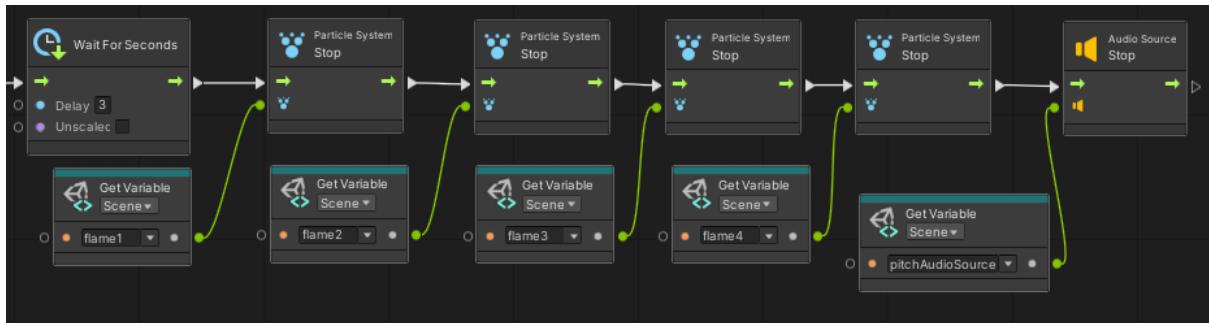
Στον Update κόμβο η μόνο λειτουργία είναι η ανανέωση του σκορ στην οθόνη όπως απεικονίζεται στην εικόνα 37. Το script δέχεται τα score1 και score2 που αντιπροσωπεύουν τα γκολ των δύο παικτών, τα μετατρέπει σε string και τα εμφανίζει στο UI.

4.5 GOAL SETUP

Όταν η μεταβλητή wasGoal του “Ball Behaviour” script γίνει αληθής, ενεργοποιείται ο ήχος του γκολ και ενεργοποιούνται τα particle system (οι φλόγες) στις άκρες του γηπέδου. Παραμένουν ενεργές για τρία δευτερόλεπτα και στη συνέχεια απενεργοποιούνται αυτόματα, ολοκληρώνοντας το οπτικό εφέ της επίτευξης γκολ. Τα γεγονότα αυτά απαθανατίζονται στην εικόνα 38 και 39.



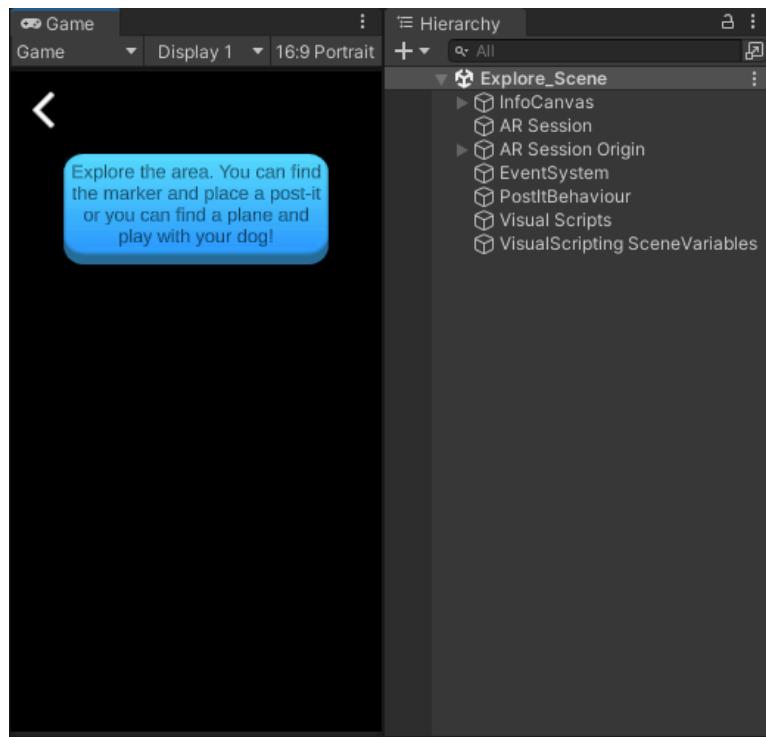
EIKONA 38 : GOAL SETUP



EIKONA 39 : GOAL SETUP STOP

ΚΕΦΑΛΑΙΟ 5 : POST - IT

Στην explore επιλογή της εφαρμογής, ο χρήστης μεταφέρεται στην κάμερα όπου το Information Panel τον ενθαρύνει να εξερευνήσει τον χώρο. Όταν ανιχνεύσει το marker (ΕΙΚΟΝΑ 42), ο χρήστης έχει την δυνατότητα να γράψει ένα σύντομο μήνυμα με σκοπό να εμφανιστεί στο Post-it (ΕΙΚΟΝΑ 43) το οποίο θα τοποθετηθεί στην οθόνη του πάνω στο marker. Το Post-it prefab αποτελεί έργο του Sten Ulfsson από το Office Supplies Low Poly πακέτο[35]. Σε αυτό το κεφάλαιο θα αναλύσουμε την υλοποίηση του Post-it.

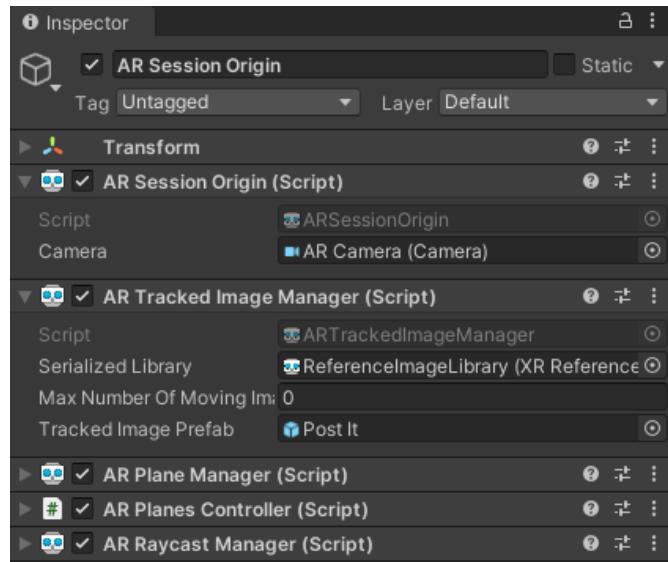


ΕΙΚΟΝΑ 40 : EXPLORE SCENE

5.1 SCENE PROPERTIES

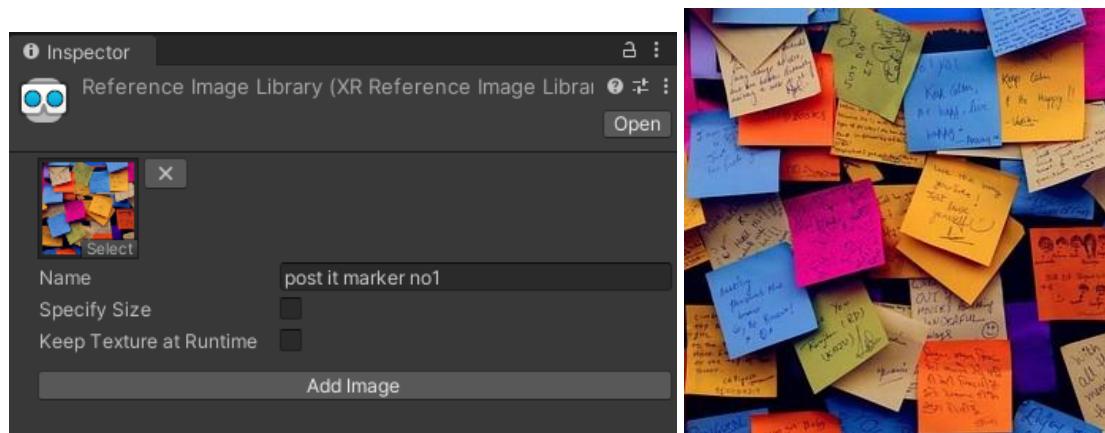
Διαθέτουμε AR Session και AR Session Origin με το δεύτερο να περιέχει σαν components τα AR Session Origin και AR Tracked Image Manager το οποίο χρησιμοποιεί το XRImageTrackingSubsystem για να αναγνωρίζει και να ανιχνεύει 2D εικόνες στο φυσικό περιβάλλον όπως απεικονίζεται στην εικόνα 41.

[35] Ανακτήθηκε από <https://assetstore.unity.com/packages/3d/props/office-supplies-low-poly-105519>



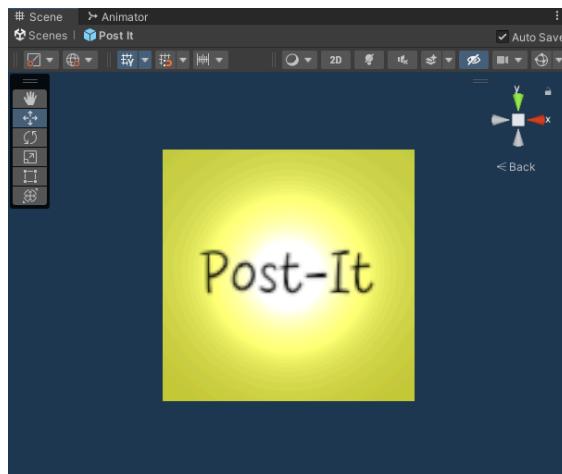
ΕΙΚΟΝΑ 41 : POST-IT AR SESSION ORIGIN

- Serialized Library : Θέτει τις εικόνες που μπορεί η εφαρμογή να ανιχνεύσει και το Reference Image Library είναι η βιβλιοθήκη που περιέχει όλες αυτές τις εικόνες. Στη περίπτωσή μας έχουμε μία μόνο εικόνα όπως φαίνεται στην εικόνα 42.[36]
- Max Number Of Moving Images : Δίνει την δυνατότητα στην εφαρμογή να ανιχνεύει κινούμενες εικόνες.[36]
- Tracked Image Prefab : Αναφέρεται στο αντικείμενο που θα δημιουργηθεί και θα εμφανιστεί στον χρήστη όταν ανιχνεύσει το marker. Στη παρούσα περίπτωση είναι το post-it.[36]



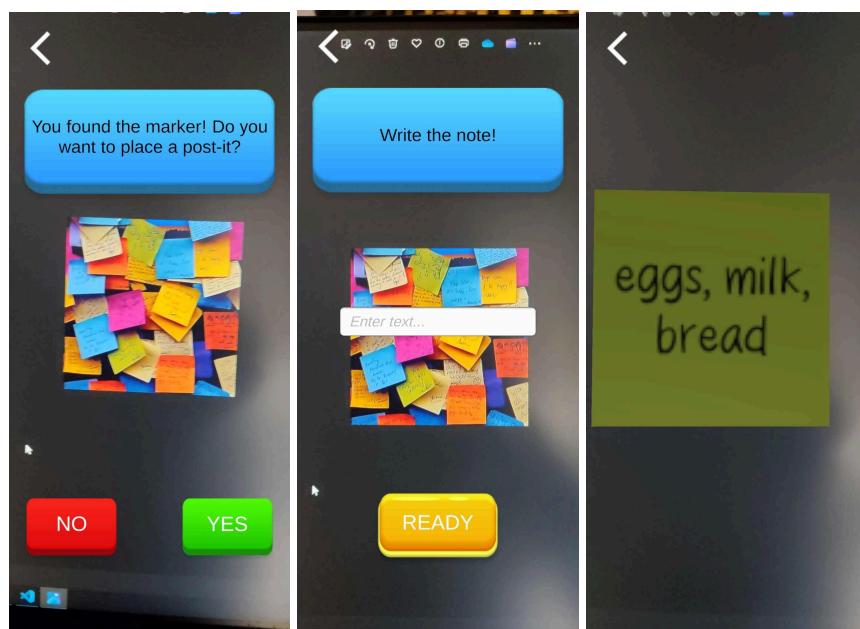
ΕΙΚΟΝΑ 42 : REFERENCE IMAGE LIBRARY (ΑΡΙΣΤΕΡΑ), MARKER (ΔΕΞΙΑ)

[36] Ανακτήθηκε από <https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.2/api/UnityEngine.XR.ARFoundation.ARTrackedImageManager.html>



ΕΙΚΟΝΑ 43 : POST-IT PREFAB

Όταν ο χρήστης εντοπίσει το marker στον χώρο και πλησιάσει τη συσκευή του σε αυτό, στο info panel θα εμφανιστεί το μήνυμα “Do you want to place a post-it?”, ταυτόχρονα θα εμφανιστούν και δύο κουμπιά, “Yes” και “No”. Αν επιλεχθεί το “No”, επανέλθει στην σκηνή ώστε να συνεχίσει να εξερευνεί και να μπορέσει να τοποθετήσει ένα post-it αργότερα. Αν πατηθεί το “Yes”, εμφανίζει ένα πεδίο εισαγωγής (Input Field) που ο χρήστης πρέπει να γράψει το μήνυμα που θα εμφανιστεί πάνω στο post-it. Επίσης εμφανίζεται κι ένα “Ready” κουμπί που όταν πατηθεί θα απενεργοποιηθούν όλα τα παραπάνω αντικείμενα και θα εμφανιστεί το Post-it πάνω στο marker με το μήνυμα που έγραψε προηγουμένως ο χρήστης. Ένα παράδειγμα των βημάτων αυτών απεικονίζονται στην εικόνα 44.



ΕΙΚΟΝΑ 44 : ΤΑ ΣΤΑΔΙΑ ΕΜΦΑΝΙΣΗΣ ΤΟΥ POST-IT

5.2 POST - IT BEHAVIOUR

Για την υλοποίηση του Post-it έχει δημιουργηθεί ένα script (PostItBehaviour). Έχει σαν μεταβλητές όλα τα buttons που προαναφέρθηκαν, τον καμβά (Canvas) με το info panel και το input field object. Επίσης δηλώνονται σαν μεταβλητές, το prefab του post-it (postItPrefab), το canvas του post-it (postItCanvas) και το text του post-it (postItText).

Έχουν υλοποιηθεί τέσσερις συναρτήσεις (ΕΙΚΟΝΑ 45) για την συμπεριφορά των buttons {OnYesButtonPressed(), OnNoButtonPressed(), OnReadyButtonPressed(), OnBackButtonPressed()} σύμφωνα με αυτά που αναφέρθηκαν παραπάνω.

```
public void OnYesButtonPressed()
{
    yesButton.SetActive(false);
    noButton.SetActive(false);
    readyButton.SetActive(true);
    InputFieldGameObject.SetActive(true);
    panelText.text = "Write the note!";
}

public void OnNoButtonPressed()
{
    MainAppSceneLoader.Instance.LoadScene("PostIt_Scene");
}

public void OnReadyButtonPressed()
{
    postItText.text = postItInputField.text;
    backButton.SetActive(true);
    infoPanel.SetActive(false);
    InputFieldGameObject.SetActive(false);
    readyButton.SetActive(false);
    postItPrefab.SetActive(true);
}

public void OnBackButtonPressed()
{
    MainAppSceneLoader.Instance.LoadScene("MainApp_Scene");
}
```

ΕΙΚΟΝΑ 45 : Η ΥΛΟΠΟΙΗΣΗ ΤΩΝ 4 BUTTONS ΣΕ C#

Εν πρώτης οι μεταβλητές postItPrefab, postItCanvas και postItText δεν είναι αρχικοποιημένες διότι το post-it δεν υπάρχει κάπου στο scene σαν object ώστε να είναι δυνατή η δήλωσή τους. Οπότε στην Update() συνάρτηση του script αναφέρεται ότι αν το postItPrefab είναι null (δεν έχει οριστεί) τότε να ψάξει αντικείμενο με tag “PostIt” (ΕΙΚΟΝΑ 46).

Έπειτα σε μία άλλη if αν το postItPrefab δεν είναι null (αν έχει δηλαδή βρεθεί το post-it), ορίζονται το postItCanvas με τον καμβά του post-it και το postItText με το “παιδί” του συγκεκριμένου καμβά που είναι το αντικείμενο text.

To post-it θα βρεθεί μόνο όταν ο χρήστης της εφαρμογής ανιχνεύσει το marker. Από προεπιλογή, όταν η εφαρμογή ανιχνεύσει το marker, το post-it εμφανίζεται. Στην εν λόγω εφαρμογή, βασική προϋπόθεση αποτελεί η εμφάνισή της αφού ο χρήστης εισάγει το μήνυμα. Συνεπώς όταν ανιχνευθεί το marker και η μεταβλητή postItPrefab οριστεί με το post-it, απενεργοποιείται (postItPrefab.SetActive(false)). Τέλος υπάρχει και μία boolean μεταβλητή η οποία όταν οριστούν για πρώτη φορά οι μεταβλητές, γίνεται αληθής και το περιεχόμενο της if δεν ξαναεκτελείται. Παρακάτω στην εικόνα 46 υπάρχει ο κώδικας που υλοποιεί αυτά που αναφέρθηκαν.

```

private GameObject postItPrefab;
private GameObject postItCanvas;
private Text postItText;
private bool found = false;

// Update is called once per frame
void Update()
{
    if(postItPrefab == null)
    {
        postItPrefab = GameObject.FindGameObjectWithTag("PostIt");
    }

    if(postItPrefab != null && found == false)
    {
        postItCanvas = GameObject.FindGameObjectWithTag("PostItCanvas");
        postItText = postItCanvas.GetComponentInChildren<Text>();
        postItPrefab.SetActive(false);
        panelText.text = "You found the marker! Do you want to place a post-it?";
        yesButton.SetActive(true);
        noButton.SetActive(true);
        found = true;
    }
}

```

ΕΙΚΟΝΑ 46 : ΤΡΟΠΟΣ ΕΥΡΕΣΗΣ ΚΑΙ ΥΛΟΠΟΙΗΣΗΣ ΤΩΝ ΜΕΤΑΒΛΗΤΩΝ ΤΟΥ POST-IT

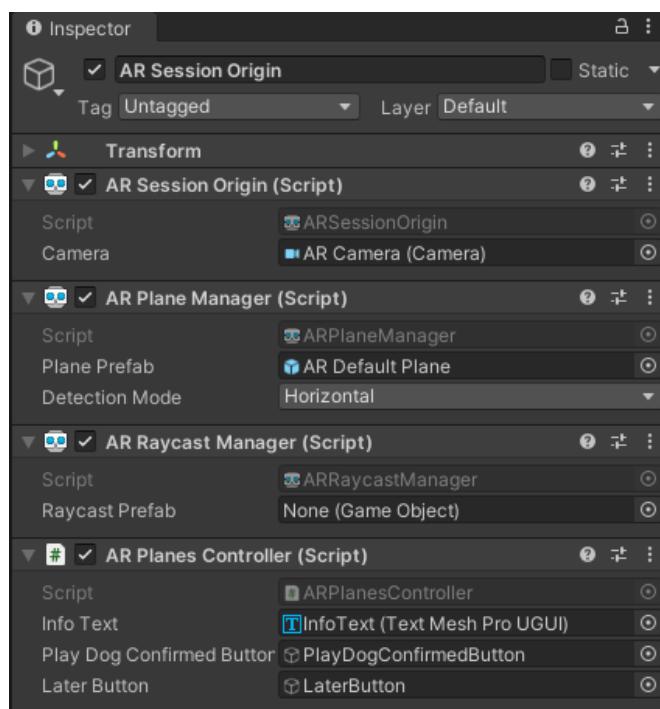
ΚΕΦΑΛΑΙΟ 6 : ΨΗΦΙΑΚΟ ΚΑΤΟΙΚΙΔΙΟ

Εκτός από το post-it στην explore επιλογή της εφαρμογής, ο χρήστης μπορεί να τοποθετήσει και να αλληλεπιδράσει με ένα ψηφιακό κατοικίδιο. Αφού ανιχνεύσει ένα plane, η εφαρμογή θα εμφανίσει ένα μήνυμα ότι μπορεί να τοποθετήσει το κατοικίδιο. Έπειτα ο χρήστης μπορεί να το ταΐσει, να το προστάξει να καθίσει και να το χαιδέψει.

6.1 SCENE PROPERTIES

Υπάρχουν τα απαραίτητα, όπως και στις προηγούμενες περιπτώσεις, AR Session και AR Session Origin όπου το δεύτερο έχει τα εξής components (ΕΙΚΟΝΑ 47):

- AR Session Origin
- AR Plane Manager που είναι υπεύθυνο για την ανίχνευση, συμπεριφορά κι εμφάνιση των planes
- AR Raycast Manager που βοηθάει στην τοποθέτηση του κατοικιδίου στο plane μέσω της επαφής του χρήστη
- AR Planes Controller είναι script που έχω δημιουργήσει για την συμπεριφορά του UI



ΕΙΚΟΝΑ 47 : DIGITAL DOG AR SESSION ORIGIN COMPONENTS

6.2 SCRIPTING

Έχει δημιουργηθεί ένα script για την συμπεριφορά των UI, το ARPlanesController. Δηλώνονται ως μεταβλητές τα infoText για την αλλαγή των κειμένων στο panel, το playDogConfirmed button, το later button, δύο boolean μεταβλητές pressed και spawned οι οποίες αρχικοποιούνται ψευδής και μία arPlaneManager μεταβλητή.

Στη συνάρτηση Awake() με την εντολή :

```
arPlaneManager = GetComponent<ARPLaneManager>();
```

η οποία λαμβάνει το ARPlaneManager component του AR Session Origin και με αυτόν τον τρόπο λαμβάνονται πληροφορίες για την ανίχνευση των planes όπως για παράδειγμα την ποσότητα και την ταυτότητα του καθενός.

Επίσης υπάρχει και μία συνάρτηση για την λειτουργία του κάθε button. Η OnConfirmedButtonClicked() εκτελείται όταν ο χρήστης πατήσει το “YES!” button στην οθόνη του που σημαίνει ότι θέλει να τοποθετήσει και να αλληλεπιδράσει με το ψηφιακό του κατοικίδιο. Απενεργοποιούνται τα δύο buttons και το ARPlaneManager (με την εντολή arPlaneManager.enabled = false) ώστε η εφαρμογή να μην ανιχνεύσει επιπλέον planes και η pressed μεταβλητή γίνεται αληθής (εξηγείται παρακάτω η χρησιμότητά της).

Η OnLaterButtonClicked(); εκτελείται όταν χρήστης δεν θέλει να αλληλεπιδράσει με το ψηφιακό κατοικίδιο και πατάει το button “Later”. Σε αυτήν την περίπτωση κάνει reload το scene ώστε να ξεκινήσει η εξερεύνηση του χώρου από την αρχή.

```
public void OnConfirmedButtonClicked()
{
    arPlaneManager.enabled = false;

    playDogConfirmedButton.SetActive(false);
    laterButton.SetActive(false);
    pressed = true;
}

public void OnLaterButtonClicked()
{
    MainAppSceneLoader.Instance.LoadScene("PostIt_Scene");
}
```

EIKONA 48 : ΟΙ ΣΥΝΑΡΤΗΣΕΙΣ ΤΩΝ BUTTONS

Στην Update() συνάρτηση (ΕΙΚΟΝΑ 49) υπάρχουν δύο if. Στην πρώτη ελέγχεται μέσω του ARPlaneManager αν έχουν ανιχνευθεί planes κι αν η συνθήκη είναι αληθής εμφανίζεται στο panel ότι βρέθηκε επιφάνεια και ρωτάει τον χρήστη αν θέλει να τοποθετήσει το ψηφιακό κατοικίδιο ενεργοποιώντας τα δύο buttons που αναφέρθηκαν προηγουμένως. Η δεύτερη if ελέγχει αν η spawned μεταβλητή είναι αληθής κι αν η συνθήκη αληθεύει τότε καλεί την συνάρτηση SetAllPlanesDeactive() όπως φαίνεται στην εικόνα 50 η οποία απενεργοποιεί όλα τα planes που έχουν ανιχνευθεί. Η spawned μεταβλητή γίνεται αληθής όταν το κατοικίδιο εμφανιστεί στην οθόνη.

```
void Update()
{
    if(arPlaneManager.trackables.count > 0)
    {
        infoText.text = "You found a plane. Do you want to play with your dog?";
        playDogConfirmedButton.SetActive(true);
        laterButton.SetActive(true);
    }

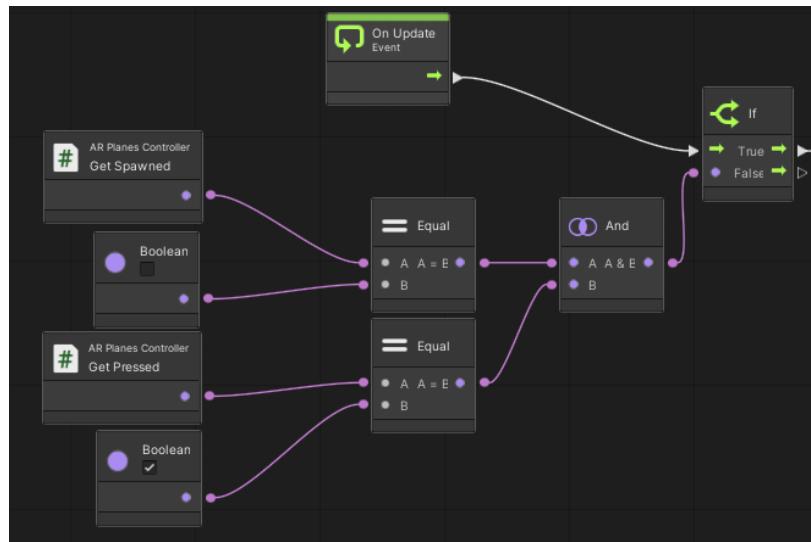
    if(spawned)
    {
        SetAllPlanesDeactive();
    }
}
```

ΕΙΚΟΝΑ 49 : Η UPDATE ΣΥΝΑΡΤΗΣΗ

```
private void SetAllPlanesDeactive()
{
    foreach(var plane in arPlaneManager.trackables)
    {
        plane.gameObject.SetActive(false);
    }
}
```

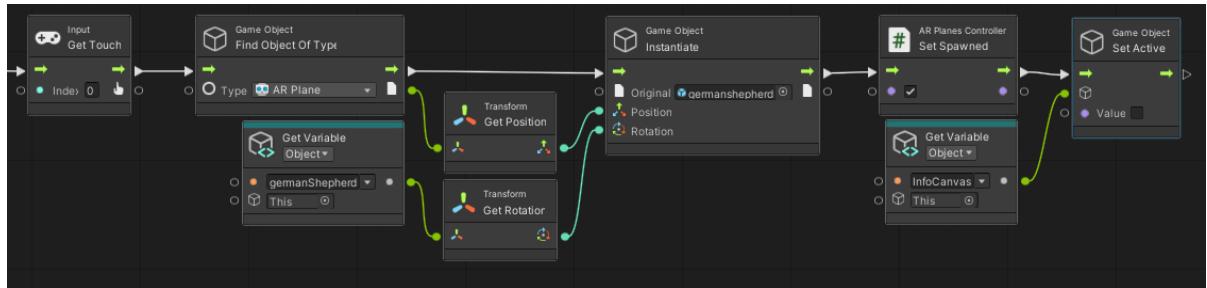
ΕΙΚΟΝΑ 50 : SETALLPLANESDEACTIVE ΣΥΝΑΡΤΗΣΗ

Για το instantiation του κατοικιδίου χρησιμοποιείται visual scripting με το TapToPlace script που περιέχει ένα OnUpdate κόμβο κι αμέσως μετά ένα if όπως απεικονίζεται στην εικόνα 51. Εκμεταλλεύονται τα Get Spawned και Get Pressed nodes στην συνθήκη ελέγχοντας αν η πρώτη είναι ψευδής, αν δηλαδή το κατοικίδιο έχει εμφανιστεί στην οθόνη του χρήστη και η δεύτερη αν είναι αληθής, δηλαδή αν έχει πατηθεί το “YES!” button. Στο αρχικό instantiation του script, η A συνθήκη (Get Spawned == false) είναι αληθής και η B συνθήκη (Get Pressed == true) είναι ψευδής. Οι συνθήκες συνδέονται με το λογικό κόμβο AND οπότε η τελική συνθήκη είναι ψευδής. Όταν πατηθεί το “YES!” button η συνθήκη B γίνεται αληθής με αποτέλεσμα η AND συνθήκη να είναι κι αυτή αληθής και το script προχωράει στον επόμενο κόμβο.



ΕΙΚΟΝΑ 51 : ΧΡΗΣΗ ΤΩΝ GET SPAWN ΚΑΙ GET PRESSED ΜΕΤΑΒΛΗΤΩΝ

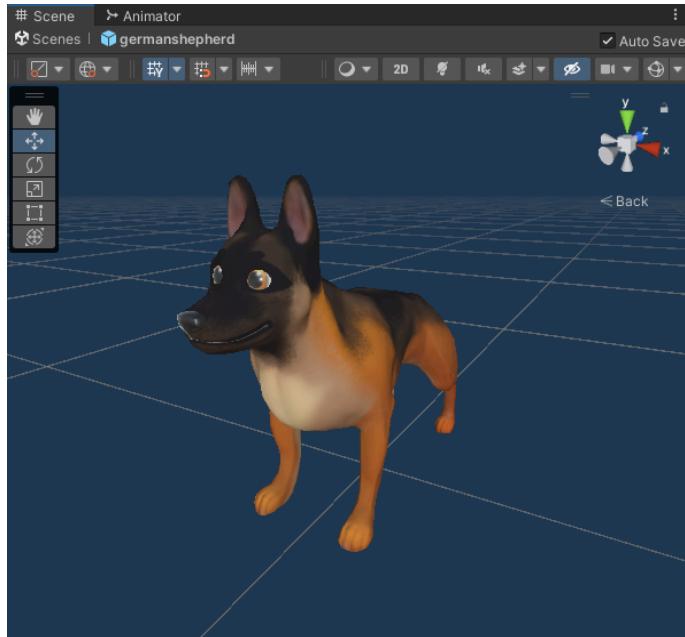
Στη συνέχεια υπάρχει το Get Touch node που αντιλαμβάνεται την επαφή του χρήστη με την οθόνη (στην προκειμένη περίπτωση το πάτημα του “YES!” button) κι αφού εντοπίσει object του τύπου AR Plane (Find Object Of Type node) τοποθετεί (instantiate) το κατοικίδιο στην τοποθεσία της επιφάνειας (μέσω του Get Position κόμβου). Τέλος αλλάζει την spawned μεταβλητή σε αληθή ώστε να μην εκτελεστεί ξανα η ίδια διαδικασία κι απενεργοποιεί τον κανβά με τις πληροφορίες και τα buttons όπως φαίνεται στην εικόνα 52.



ΕΙΚΟΝΑ 52 : Η ΔΙΑΔΙΚΑΣΙΑ ΕΜΦΑΝΙΣΗΣ ΤΟΥ ΚΑΤΟΙΚΙΔΙΟΥ

6.3 DIGITAL DOG PREFAB

Όταν ο χρήστης επιλέξει να τοποθετήσει και να αλληλεπιδράσει με το ψηφιακό κατοικίδιο θα αντικρίσει μία προσομοίωση ενός γερμανικού ποιμενικού (ΕΙΚΟΝΑ 53), δημιούργημα της Publisher.[37]



EIKONA 53 : GERMAN SHEPHERD PREFAB

Αποτελείται από τα φυσικά μέλη του (κεφάλι, πόδια, ουρά) και προστέθηκαν ένα αντικείμενο τροφής, το Udon prefab (δημιούργημα της Mumifier Studio από το Food Pack).[38] που εμφανίζεται μπροστά στο σκυλί όταν ο χρήστης επιλέξει να το ταΐσει και ένας κανβάς (DogCanvas) που περιέχει τα buttons για τις κινήσεις του κατοικίδιου.

Επιπλέον έχουν τοποθετηθεί τα εξής components στο prefab :

- **Audio Source** : Κατά τη διάρκεια αλληλεπίδρασης με το σκυλί ακούγονται τρεις ήχοι. Ο πρώτος ακούγεται όταν εμφανιστεί για πρώτη φορά στην οθόνη, ο δεύτερος όταν τρώει κι ο τελευταίος όταν ο χρήστης το χαιδεύει. Το Audio Source component είναι υπεύθυνο για αυτούς του ήχους.
- **Animator** : Χρειάζεται για να προσδιορίσουμε κινήσεις στο σκυλί οι οποίες εξηγούνται παρακάτω.[39]
- **Demo Controller** : Κώδικας που υλοποιεί τα animations σε buttons. Επίσης είναι υπεύθυνο και για συμπεριφορά των buttons σχετικά με την εμφάνισή τους.
- **Pet Dog** : Υλοποιεί το άγγιγμα του χρήστη στο κατοικίδιο.

[37] Ανακτήθηκε από

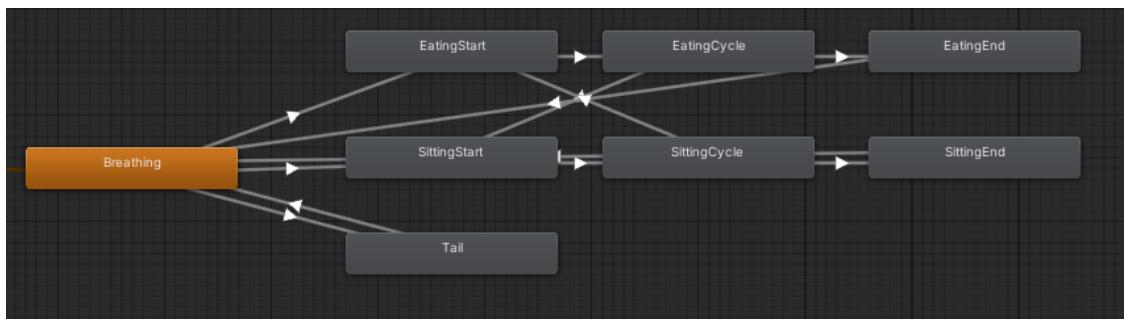
<https://assetstore.unity.com/packages/3d/characters/animals/mammals/3d-stylized-animated-dogs-kit-284699>

[38] Ανακτήθηκε από <https://assetstore.unity.com/packages/3d/props/food/food-pack-free-demo-225294>

[39] Ανακτήθηκε από <https://docs.unity3d.com/Manual/class-Animator.html>

6.4 ANIMATION [40]

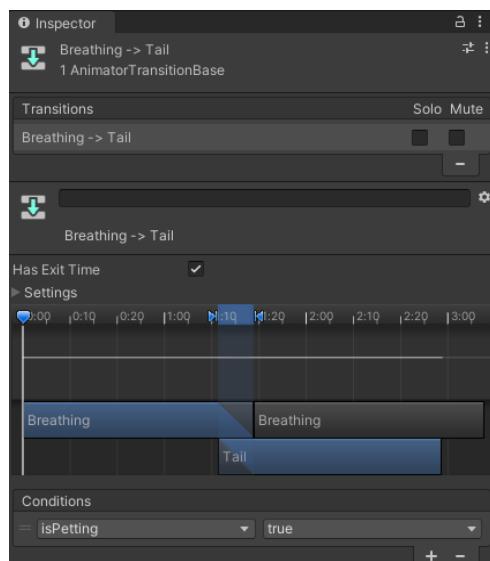
Το animation σύστημα του Unity βασίζεται σε **Animation Clips**[41], τα οποία περιέχουν πληροφορία για το πως μπορούν αντικείμενα να προσαρμόσουν την θέση τους (position), τη περιστροφή τους (rotation) ή άλλες ιδιότητές τους με τη πάροδο του χρόνου. Κάθε clip μπορεί να θεωρηθεί σαν μία απλή γραμμική καταγραφή και όλα μαζί δομούνται σε ένα flowchart-like σύστημα που ονομάζεται **Animator Controller** (ΕΙΚΟΝΑ 54). Η υλοποίηση των animations ανήκει στην Publisher.[42]



ΕΙΚΟΝΑ 54 : TO ANIMATOR CONTROLLER ΤΟΥ ΚΑΤΟΙΚΙΔΙΟΥ

Λειτουργεί σαν ένα “State Machine” το οποίο ανιχνεύει και προβάλλει ποιο clip πρέπει να παίζει τη δεδομένη στιγμή και πότε οι κινήσεις θα αλλάξουν ή θα αναμειχθούν μεταξύ τους. Στο παρόν controller έχουμε το σταθερό clip Breathing, δηλαδή αυτό που εκτελείται συνεχόμενα εκτός κι αν ο χρήστης επιλέξει κάποια κίνηση. Έπειτα το κάθισμα και το τάισμα του σκύλου αποτελούνται από κύκλο τριών κινήσεων. Υπάρχει η δυνατότητα ο χρήστης να μεταβεί από τον ένα κύκλο στον άλλο παραδείγματος χάρη όταν ο σκύλος κάθεται εκτελείται το Sitting Cycle clip, μπορεί ο χρήστης να το τάσσει και να μεταβεί στο Eating Start clip.

Οι ακμές μεταξύ των clips ονομάζονται **transitions** και καθορίζουν τις συνθήκες οι οποίες όταν εκπληρωθούν, το animator controller αλλάζει από το ένα state στο άλλο. Υπάρχει η **animationID** παράμετρος η οποία είναι τύπου int και χρησιμοποιείται στο Eating και στο Sitting κύκλο. Για να μεταβεί το controller από το Breathing state στο Tail state όπως φαίνεται στην εικόνα 55, υπάρχει μία boolean παράμετρος, η **isPetting** η οποία πρέπει να είναι αληθής.



ΕΙΚΟΝΑ 55 : ΠΑΡΑΔΕΙΓΜΑ TRANSITION

[40] Ανακτήθηκε από <https://docs.unity3d.com/Manual/AnimationSection.html>

[41] Ανακτήθηκε από <https://docs.unity3d.com/Manual/AnimationClips.html>

[42] Ανακτήθηκε από

<https://assetstore.unity.com/packages/3d/characters/animals/mammals/3d-stylized-animated-dogs-kit-284699>

6.5 DOG CANVAS

Ο καμβάς του κατοικιδίου περιέχει ένα panel για τις πληροφορίες που εμφανίζονται στον χρήστη και πέντε buttons, τέσσερα που σχετίζονται με τα animations κι ένα exit button το οποίο οδηγεί τον χρήστη στο αρχικό scene της εφαρμογής (MainApp scene). Τα buttons για τα animations του σκύλου είναι τα εξής:

- Sit Down button : Κάθεται
- Stand Up button : Σηκώνεται
- Feed button : Εμφανίζεται το φαγητό και ξεκινάει να τρώει
- Finish button : Σταματάει να τρώει και το φαγητό εξαφανίζεται

Η υλοποίηση των buttons έχει πραγματοποιηθεί με δύο scripts, το AnimationButton (ΕΙΚΟΝΑ 56) και το DemoController. Το πρώτο τοποθετείται σε object τύπου button, στην προκειμένη περίπτωση τα buttons που αναφέρθηκαν παραπάνω. Υπάρχει μία μεταβλητή _animationID τύπου int, έχει δημιουργηθεί ένα event για το κλικάρισμα του button και τέλος στην Awake() συνάρτηση αποκτά το button component.

```
public class AnimationButton : MonoBehaviour
{
    [SerializeField] public int _animationID;

    public event Action<int> Click;

    private void Awake()
    {
        GetComponent<Button>().onClick.AddListener(()=>{Click?.Invoke(_animationID);});
    }
}
```

EIKONA 56 : ANIMATION BUTTON SCRIPT

To DemoCotroller script δημιουργεί μια λίστα με τα buttons και μία για το animator (δέχεται σαν όρισμα το σκυλί και λαμβάνει αυτόματα το animator component του). Έχει σαν μεταβλητές το φαγητό, τα buttons του κανβά, την μεταβλητή dog AudioSource που παίρνει σαν όρισμα το audio source component του κατοικιδίου και το eating clip που είναι ο ήχος που θα ακούγεται όταν το σκυλί θα τρώει. Οι συναρτήσεις που υλοποιούνται στο script είναι οι εξής :

-Start() : Αποκτά το Audio Source του component του κατοικιδίου και για κάθε button που έχει οριστεί δημιουργεί την OnAnimationButtonClick() συνάρτηση.

-OnAnimationButtonClick() : Θέτει το animationID που είναι παράμετρος στο animator με την τιμή που λαμβάνει η συνάρτηση. Αναλόγως την τιμή του animationID, εκτελείται και το συγκεκριμένο transition. Επίσης υπάρχουν if συνθήκες για τη συμπεριφορά του UI (ποια buttons ενεργοποιούνται και ποια απενεργοποιούνται) όπως φαίνεται στην παρακάτω εικόνα 57.

```

animator.SetInteger("AnimationID",id);
if(id == 1)
{
    food.SetActive(false);
    StopEatingSound();
    sitDownButton.SetActive(false);
    standUpButton.SetActive(true);
    finishButton.SetActive(false);
    feedButton.SetActive(true);
}
else if(id == 2)
{
    sitDownButton.SetActive(true);
    standUpButton.SetActive(false);
}
else if(id == 3)
{
    food.SetActive(true);
    PlayEatingSound();
    sitDownButton.SetActive(true);
    standUpButton.SetActive(false);
    finishButton.SetActive(true);
    feedButton.SetActive(false);
}
else if(id == 4)
{
    food.SetActive(false);
    StopEatingSound();
    sitDownButton.SetActive(true);
    standUpButton.SetActive(false);
    finishButton.SetActive(false);
    feedButton.SetActive(true);
}

```

ΕΙΚΟΝΑ 57 : ONANIMATIONBUTTONCLICK ΣΥΝΑΡΤΗΣΗ

-PlayEatingSound() & StopEatingSound() : Όταν το σκυλί τρώει ακούγεται ένας ήχος σχετικός με αυτή τη διαδικασία. Οι συγκεκριμένες συναρτήσεις ενεργοποιούν κι απενεργοποιούν αυτόν τον ήχο.

6.6 PET DOG SCRIPT

Στο script αυτό υλοποιείται η επαφή (continuous touch) του χρήστη πάνω στο κεφάλι του κατοικίδιου και το αποτέλεσμα αυτής. Μεταβλητές του script αποτελούν το dog AudioSource, το pettingClip, το animator και μία boolean μεταβλητή isPetting που αρχικοποιείται ως ψευδής.

Η Start() συνάρτηση λαμβάνει τα AudioSource και Animator components του σκύλου και η Update() υλοποιεί με την μέθοδο case την επαφή του χρήστη με το κατοικίδιο. Όταν ο χρήστης “χαϊδεύει” τον σκύλο εκτελείται η PlayPettingSound() συνάρτηση που ενεργοποιεί έναν ήχο κι όταν δεν υπάρχει επαφή η StopPettingSound() συνάρτηση τον απενεργοποιεί.

```
void Update()
{
    if (Input.touchCount > 0)
    {
        Touch touch = Input.GetTouch(0);
        Ray ray = Camera.main.ScreenPointToRay(touch.position);
        RaycastHit hit;

        // Use raycast to see if the user touches the dog
        switch (touch.phase)
        {
            case TouchPhase.Began:
                if (Physics.Raycast(ray, out hit) && hit.transform.CompareTag("Dog"))
                {
                    isPetting = true;
                    animator.SetBool("isPetting", true);
                    PlayPettingSound();
                }
                break;

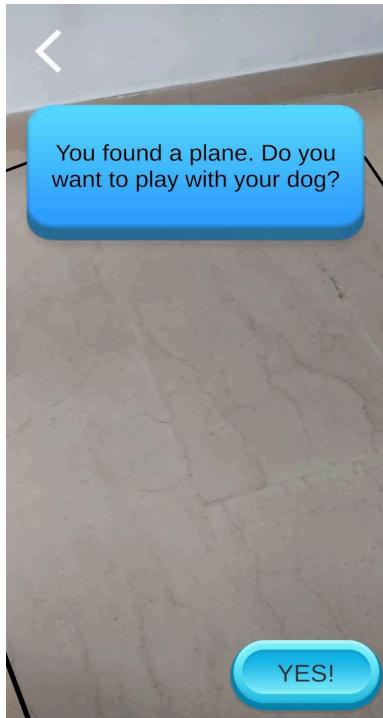
            case TouchPhase.Moved:
            case TouchPhase.Stationary:
                if (isPetting)
                {
                    // The touch continuous
                }
                break;

            case TouchPhase.Ended:
                if (isPetting)
                {
                    isPetting = false;
                    animator.SetBool("isPetting", false);
                    StopPettingSound();
                }
                break;
        }
    }
}
```

ΕΙΚΟΝΑ 58 : Η UPDATE ΣΥΝΑΡΤΗΣΗ ΤΟΥ PET DOG SCRIPT

6.7 ΕΝΔΕΙΚΤΙΚΕΣ ΦΩΤΟΓΡΑΦΙΕΣ

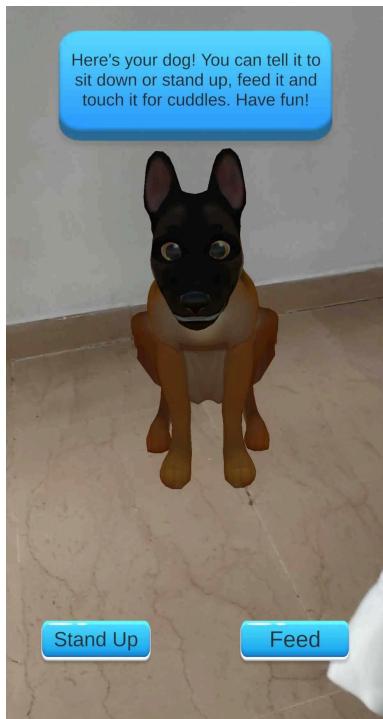
Τέλος παρατίθενται εικόνες σχετικές με την ανίχνευση, τοποθέτηση και την αλληλεπίδραση του χρήστη με το ψηφιακό κατοικίδιο.



EIKONA 59 : ΑΝΙΧΝΕΥΣΗ PLANE



EIKONA 60 : ΤΟΠΟΘΕΤΗΣΗ ΚΑΤΟΙΚΙΔΙΟΥ



EIKONA 61 : SIT DOWN BUTTON



EIKONA 62 : FEED BUTTON

ΕΠΙΛΟΓΟΣ

Στην εργασία αυτή υλοποιήθηκε μία εφαρμογή επαυξημένης πραγματικότητας για χρήστη σε Android συσκευές. Η δημιουργία αυτού του οπτικού υλικού δίνει την δυνατότητα στον χρήστη να περιηγηθεί στο σπίτι και να ψυχαγωγηθεί μέσα από τα Multiplayer Games, το Post-it και την αλληλεπίδραση με το ψηφιακό κατοικίδιο. Πιο συγκεκριμένα η εκπόνηση αυτής της εφαρμογής στηρίχθηκε στην πλατφόρμα της Unity όπου χρησιμοποιήθηκαν τα AR Foundation και ARCore εργαλεία. Η εκπόνηση αυτή στοχεύει στην κατανόηση του ρόλου καθώς και τις έννοιες της επαυξημένης πραγματικότητας μέσα από ορισμούς και εικόνες που δόθηκαν στα πλαίσια της ανάπτυξης της εφαρμογής. Δόθηκαν εξηγήσεις σχετικά με την εν λόγω τεχνολογία και πως γίνεται η χρήση των εργαλείων της με σκοπό την κατάκτηση του αποτελέσματος. Τέλος η εφαρμογή αποτελεί παράδειγμα ενσωμάτωσης της επαυξημένης πραγματικότητας στην καθημερινή ζωή των ανθρώπων καθώς και η παράλληλη διευκόλυνσή της.

Το Unity αποτελεί ένα από τα πιο ισχυρά και δημοφιλή εργαλεία ανάπτυξης προσφέροντας μία φιλική προς τον χρήστη πλατφόρμα. Επιπλέον δίνεται η δυνατότητα δημιουργίας εφαρμογών επαυξημένης πραγματικότητας τα οποία μπορούν στη συνέχεια να χρησιμοποιηθούν σε πληθώρα πλατφορμών.

Ανακεφαλαιώνοντας γίνεται αντιληπτό πως η δημιουργία μιας εφαρμογής επαυξημένης πραγματικότητας αποτελεί ένα πολυδιάστατο εγχείρημα το οποίο απαιτεί την χρήση της σύγχρονης τεχνολογίας σε συνδυασμό με σχεδιαστικά εργαλεία. Επιπρόσθετα, εύκολα παρατηρείται πως η επαυξημένη πραγματικότητα εξελίσσεται ραγδαία τα τελευταία χρόνια και κάνει την είσοδό της σε πολλούς τομείς της βιομηχανίας αφήνοντας πίσω τα μακρινά χρόνια που περιορίζονταν στο τομέα της πολεμικής αεροπορίας. Καταληκτικά, στο μέλλον η επαυξημένη πραγματικότητα μπορεί να κατορθώσει να χρησιμοποιείται σε ακόμα περισσότερες εφαρμογές και να αποτελεί ένα αναπόσπαστο κομμάτι της καθημερινότητας των ανθρώπων.

ΑΝΑΦΟΡΕΣ

ΑΛΦΑΒΗΤΙΚΑ

- 3D Stylized Animated Dogs Kit | Characters | Unity Asset Store. (χ.χ.). Ανακτήθηκε 19 Οκτώβριος 2024, από <https://assetstore.unity.com/packages/3d/characters/animals/mammals/3d-stylized-animated-dogs-kit-284699>
- 8 Best Free Filter Apps Like Snapchat for iPhone and Android | PERFECT. (χ.χ.). Ανακτήθηκε 19 Οκτώβριος 2024, από <https://www.perfectcorp.com/consumer/blog/selfie-editing/best-face-filter-apps>
- 10 Top Games Made with Unity: Unity Game Programming. (χ.χ.). Ανακτήθηκε 19 Οκτώβριος 2024, από <https://www.create-learn.us/blog/top-games-made-with-unity/>
- 10 Top Games Made with Unity: Unity Game Programming. (2024, Αύγουστος 20). Kids' Coding Corner | Create & Learn. <https://www.create-learn.us/blog/top-games-made-with-unity/>
- About Visual Scripting | Visual Scripting | 1.9.4. (χ.χ.). Ανακτήθηκε 19 Οκτώβριος 2024, από <https://docs.unity3d.com/Packages/com.unity.visualscripting@1.9/manual/index.html>
- AR Foundation | AR Foundation | 6.0.3. (χ.χ.). Ανακτήθηκε 19 Οκτώβριος 2024, από <https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@6.0/manual/index.html>
- Augmented reality. (2024). Στο Wikipedia. https://en.wikipedia.org/w/index.php?title=Augmented_reality&oldid=1250972408
- Azuma, R. T. (χ.χ.). A Survey of Augmented Reality. <https://www.hitl.washington.edu/people/tfurness/courses/inde543/READINGS-03/OTHER/Azumapaper.pdf>
- Chung, C. (2011). Pro Objective-C Design Patterns for iOS. Apress. https://link.springer.com/chapter/10.1007/978-1-4302-3331-2_7
- Class ARPlaneManager | AR Foundation | 4.2.10. (χ.χ.). Ανακτήθηκε 19 Οκτώβριος 2024, από <https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.2/api/UnityEngine.XR.ARFoundation.ARPlaneManager.html>
- Class ARSession | AR Foundation | 4.2.10. (χ.χ.). Ανακτήθηκε 19 Οκτώβριος 2024, από <https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.2/api/UnityEngine.XR.ARFoundation.ARSession.html>
- Class ARSessionOrigin | AR Foundation | 4.2.10. (χ.χ.). Ανακτήθηκε 19 Οκτώβριος 2024, από <https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.2/api/UnityEngine.XR.ARFoundation.ARSessionOrigin.html>
- Class ARTtrackedImageManager | AR Foundation | 4.2.10. (χ.χ.). Ανακτήθηκε 19 Οκτώβριος 2024, από <https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.2/api/UnityEngine.XR.ARFoundation.ARTrackedImageManager.html>

Ferrandez, C. (2023, Φεβρουάριος 28). Why mixed reality headsets are the future of shopping. Poplar Studio.

<https://poplar.studio/blog/why-mixed-reality-headsets-are-the-future-of-shopping/>

Food Pack | Free Demo | 3D Food | Unity Asset Store. (χ.χ.). Ανακτήθηκε 19 Οκτώβριος 2024, από

<https://assetstore.unity.com/packages/3d/props/food/food-pack-free-demo-225294>

How is ARCore better than ARKit?. In some ways, but not others | by Matt Miesnieks | 6D.ai | Medium. (χ.χ.). Ανακτήθηκε 19 Οκτώβριος 2024, από

<https://medium.com/6d-ai/how-is-arcore-better-than-arkit-5223e6b3e79d>

Hut Life – Official Pizza Hut Blog. (2021, Μάρτιος 16). Hut Life – Pizza Hut Brand Blog.

<https://blog.pizzahut.com/pizza-hut-serves-up-new-nostalgia-with-campaign-celebrating-all-that-fans-know-and-love-about-the-pizza-restaurant/>

Linowes, J. (2021). Augmented Reality with Unity AR Foundation: A practical guide to cross-platform AR development with Unity 2020 and later versions. Packt Publishing Ltd.

https://books.google.gr/books?hl=el&lr=&id=iBk-EAAAQBAJ&oi=fnd&pg=PP1&dq=augmented+reality+app+in+unity&ots=azf_FGbvdU&sig=PUVtwg2zL15Khr64Ys55SghyUWY&redir_esc=y#v=onepage&q=augmented%20reality%20app%20in%20unity&f=false

Mendoza-Ramírez, C. E., Tudon-Martinez, J. C., Félix-Herrán, L. C., Lozoya-Santos, J. de J., & Vargas-Martínez, A. (2023). Augmented Reality: Survey. *Applied Sciences*, 13(18), Article 18. <https://www.mdpi.com/2076-3417/13/18/10491>

Office Supplies Low Poly | 3D Props | Unity Asset Store. (χ.χ.). Ανακτήθηκε 19 Οκτώβριος 2024, από <https://assetstore.unity.com/packages/3d/props/office-supplies-low-poly-105519>

Overview of ARCore and supported development environments. (χ.χ.). Google for Developers. Ανακτήθηκε 19 Οκτώβριος 2024, από <https://developers.google.com/ar/develop>

Particle Pack | VFX Particles | Unity Asset Store. (χ.χ.). Ανακτήθηκε 19 Οκτώβριος 2024, από <https://assetstore.unity.com/packages/vfx/particles/particle-pack-127325>

PBR Materials Sampler Pack | 2D Textures & Materials | Unity Asset Store. (χ.χ.). Ανακτήθηκε 19 Οκτώβριος 2024, από

<https://assetstore.unity.com/packages/2d/textures-materials/pbr-materials-sampler-pack-40112>

Photon- Multiplayer AR (Beyblade AR) | Tutorials | Unity Asset Store. (χ.χ.). Ανακτήθηκε 19 Οκτώβριος 2024, από

<https://assetstore.unity.com/packages/templates/tutorials/photon-multiplayer-ar-beyblade-ar-157814>

PlaySwap: ALL-IN-ONE MARKETPLACE FOR GAMERS. (χ.χ.). Ανακτήθηκε 19 Οκτώβριος 2024, από <https://playswap.gg/blog/pokemon-go-best-pokemon>

Pun 2 Introduction | Photon Engine. (χ.χ.). Ανακτήθηκε 19 Οκτώβριος 2024, από <https://doc.photonengine.com/pun/current/getting-started/pun-intro>

Real-Time 3D Development Platform & Editor. (χ.χ.). Unity. Ανακτήθηκε 19 Οκτώβριος 2024, από <https://unity.com/products/unity-engine>

Stylized Lava materials | 2D Textures & Materials | Unity Asset Store. (χ.χ.). Ανακτήθηκε 19 Οκτώβριος 2024, από

<https://assetstore.unity.com/packages/2d/textures-materials/stylized-lava-materials-180943>

Technologies, U. (χ.χ.-a). Unity - Manual: Animation. Ανακτήθηκε 19 Οκτώβριος 2024, από <https://docs.unity3d.com/6000.0/Documentation/Manual/AnimationSection.html>

Technologies, U. (χ.χ.-b). Unity - Manual: Animation clips. Ανακτήθηκε 19 Οκτώβριος 2024, από <https://docs.unity3d.com/6000.0/Documentation/Manual/AnimationClips.html>

Technologies, U. (χ.χ.-c). Unity - Manual: Animator component. Ανακτήθηκε 19 Οκτώβριος 2024, από <https://docs.unity3d.com/6000.0/Documentation/Manual/class-Animator.html>

Technologies, U. (χ.χ.-d). Unity - Manual: Physics Material asset reference. Ανακτήθηκε 19 Οκτώβριος 2024, από

<https://docs.unity3d.com/6000.0/Documentation/Manual/class-PhysicsMaterial.html>

Technologies, U. (χ.χ.-e). Unity - Manual: Rigidbody component reference. Ανακτήθηκε 19 Οκτώβριος 2024, από

<https://docs.unity3d.com/2021.3/Documentation/Manual/class-Rigidbody.html>

Unity (game engine). (2024). Στο Wikipedia.

[https://en.wikipedia.org/w/index.php?title=Unity_\(game_engine\)&oldid=1251727878](https://en.wikipedia.org/w/index.php?title=Unity_(game_engine)&oldid=1251727878)

Unity (game engine)—Wikipedia. (χ.χ.). Ανακτήθηκε 19 Οκτώβριος 2024, από

https://en.wikipedia.org/wiki/Unity_%28game_engine%29

Welcome to AR Development. (χ.χ.). Unity Learn. Ανακτήθηκε 19 Οκτώβριος 2024, από

<https://learn.unity.com/tutorial/welcome-to-the-course-2>