



**ΙΟΝΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ**

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Θέμα: Ανάπτυξη 3D παιχνιδιού πρώτου προσώπου Shooter με Unity

Φοιτητής: Αυγέρης Σωτήρης

A.M. : Π2013002

Επιβλέπων Καθηγητής: Φοίβος Μυλωνάς

Ημερομηνία: 01/05/2019

ΠΕΡΙΕΧΟΜΕΝΑ

Κεφάλαιο 1 : Εισαγωγή

Σύντομη αναδρομή στην ιστορία των βιντεοπαιχνιδιών	5
Τι είναι η Unity;.....	7
Ιστορική αναδρομή της μηχανής Unity.....	10
Ρόλοι	11

Κεφάλαιο 2 : User Interface Unity 3D

Scene View	12
Game View.....	13
Project View	15
Hierarchy	16
Inspector.....	17
Console.....	18
Animator	18
Asset Store	19
Tools – Εργαλεία της Unity	20

Κεφάλαιο 3 : Ανάπτυξη του παιχνιδιού

Level Design.....	21
Programming	27
Προσθήκες στο παιχνίδι.....	52
Mini map	53
Main Menu	55

Κεφάλαιο 4 : Ολοκλήρωση και Walkthrough παιχνιδιού

Χτίσιμο παιχνιδιού	57
Άδειες και δικαιώματα	59
Walkthrough screenshots	60

Κεφάλαιο 5 : Συμπεράσματα και βιβλιογραφία

Συμπεράσματα	66
Βιβλιογραφία.....	67

Ευχαριστίες

Η παρούσα εργασία πραγματοποιήθηκε σε αρκετά μεγάλο χρονικό διάστημα, πάνω σε ένα θέμα όχι και τόσο γνωστό αλλά πολύ ενδιαφέρον. Για την υποστήριξη της προσπάθειας μου αυτής θα ήθελα να ευχαριστήσω τον επιβλέπων Καθηγητή μου. Επίσης όλους τους διαδικτυακούς προγραμματιστές και τα Forum τα οποία επισκέφτηκα κατά τη διάρκεια της υλοποίησης της εργασίας.

Περίληψη

Η παρούσα πτυχιακή εργασία ασχολείται με την ανάπτυξη βιντεοπαιχνιδιού με χρήση της μηχανής Unity3D. Στη βιομηχανία των παιχνιδιών, με τις Game Engine υπάρχει πλέον η δυνατότητα οι νέοι προγραμματιστές να εργάζονται και να εξοικειώνονται πολύ πιο εύκολα από ποτέ, πάνω σε ένα απαιτητικό τομέα της εργασίας. Η παρακάτω πτυχιακή είχε ως θέμα την υλοποίηση ενός παιχνιδιού πρώτου προσώπου όπου θα περιείχε τα παρακάτω:

- 1) Τουλάχιστον 3 μπάρες πληροφοριών στο player UI: ενέργεια (Energy), Υγεία (Health) και Πυρομαχικά (Ammunition).
- 2) Τουλάχιστον 5 διαφορετικά είδη όπλων με τα οποία μπορεί να αλληλεπιδράσει ο παίκτης (π.χ. μαχαίρι, τόξο, πιστόλι, οπλοπολυβόλο, laser gun).
- 3) Ικανός αριθμός πυρομαχικών για κάθε είδος όπλου θα βρίσκονται διάσπαρτα στην πίστα και θα εξαφανίζονται μόλις «τα πάρει» ο παίκτης και θα επανεμφανίζονται στο ίδιο σημείο μετά από 25
- 4) Εξωτερικοί "φυσικοί" χώροι (λ.χ., ένα δάσος με δένδρα ποικίλων μεγεθών και ειδών, μία λίμνη, μία έρημος, κλπ)
- 5) Ένα μικρό «χωριό» με τουλάχιστον 10 κτίρια/φρούρια διαφόρων μεγεθών, το οποίο να έχει τουλάχιστον δύο (νοητές) εισόδους.
- 6) Μία μετρίου μεγέθους αληθοφανής φωτιά με καπνό στο κέντρο του «χωριού».

7) Προσθέτοντας δέκα περιστρεφόμενα αντικείμενα ενέργειας (energy packs) και δέκα περιστρεφόμενα αντικείμενα υγείας (healthpacks) και της επιλογής σας κάπου στο «χωριό».

8) Η επαφή του παίκτη με energy packs θα έχει ως αποτέλεσμα την εξαφάνισή τους και την αύξηση κατά 20% της Ενέργειας του παίκτη, ενώ η επαφή του με τα health packs**θα έχει ως αποτέλεσμα την εξαφάνισή τους και την αύξηση κατά 10% της Υγείας του.

9) Τουλάχιστον 5 bots τεσσάρων διαφορετικών ειδών που θα επιτίθενται στον παίκτη και συνακόλουθα θα μειώνεται η ενέργειά του ανάλογα τη σοβαρότητα του χτυπήματος που δέχθηκε.

10) Λειτουργία fatigue της ενέργειας για όσο διάστημα ο παίκτης περπατάει ή τρέχει στον εικονικό κόσμο με εύλογο depreciation rate για κάθε περίπτωση.

11) Ο παίκτης να είναι σε θέση να σκοτώσει όλα τα bots χρησιμοποιώντας τα 5 διαθέσιμα όπλα, η δε ζημία που θα προκαλείται στο bot ως το αποτέλεσμα της αλληλεπίδρασής του με καθένα από τα 5 είδη όπλων να είναι διαφορετική, προοδευτική και ανάλογη του είδους του όπλου.

Τα παραπάνω αποτελούν το αρχικό πλάνο όσο αναφορά το Game design του παιχνιδιού, παρόλα αυτά όμως, όπως συμβαίνει και στην πραγματική βιομηχανία του gaming development κάποια από tasks αλλάξαν και προστέθηκαν μερικά ακόμη σε συνεννόηση πάντα με τον επιτηρητή καθηγητή μου.

Κεφάλαιο 1: Εισαγωγή

1.1 Σύντομη αναδρομή στην ιστορία των βιντεοπαιχνιδιών

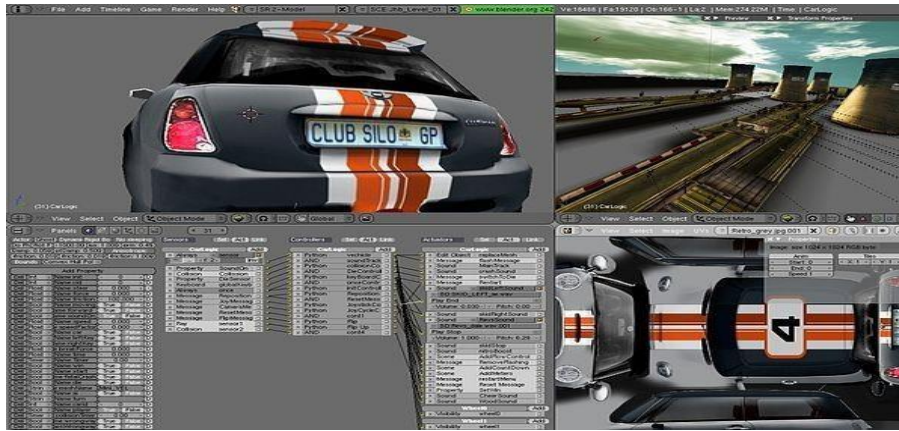
Η ιστορία των games ξεκινά από πολύ παλιά 1970-1980. Στις μέρες μας το μεγαλύτερο ποσοστό των εταιρειών έχει την δική της μηχανή δημιουργίας παιχνιδιών, υπάρχουν όμως μηχανές παιχνιδιών που διατίθενται δωρεάν για νέους developers όπως για παράδειγμα η Unity, η Unreal engine και η Cry engine. Στις μέρες μας είναι πιο εύκολο από ποτέ για έναν αρχάριο προγραμματιστή να αναπτύξει το δικό του παιχνίδι καθώς στο διαδίκτυο υπάρχουν πληθώρα ενημερωτικών βίντεο και φόρουμ ερωτοαπαντήσεων πάνω στο gaming development.

Οι μηχανές παιχνιδιών κάνουν την εμφάνισή τους την δεκαετία των 90's με το ξεκίνημα των παιχνιδιών τριών διαστάσεων που σηματοδότησε και επίσημα την απογείωση της gaming βιομηχανίας. Στα μέσα αυτής της δεκαετίας καθιερώθηκε και ο όρος Game Engine με τα πασίγνωστα παιχνίδια "Doom" και "Quake". Μετά την επιτυχία των δύο παραπάνω παιχνιδιών οι developers είχαν την δυνατότητα να αγοράσουν κομμάτια από τα παιχνίδια και να πειραματίζονται πάνω σε αυτά προσθέτοντας διάφορα όπλα και διάφορα άλλα γραφικά στοιχεία και animations. Είχαν επίσης την δυνατότητα να πουλήσουν αυτά τα mods και σαν ξεχωριστά games.

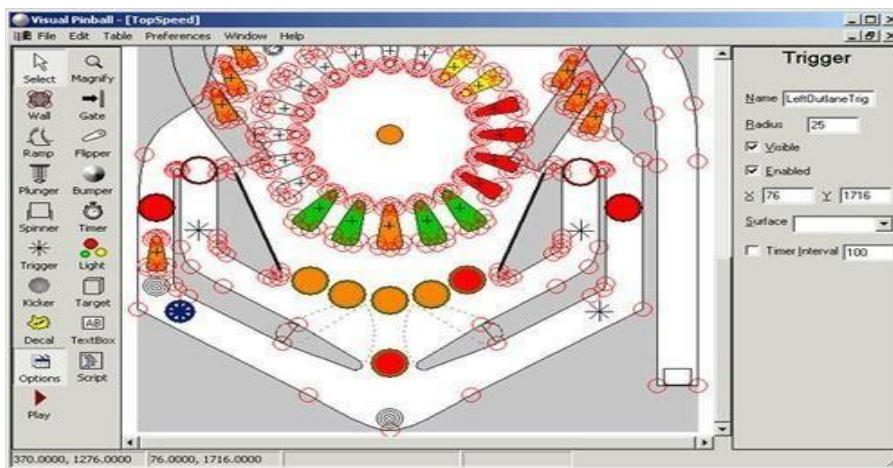
Αυτό έγινε πραγματικότητα το 1998 με την Unreal Engine, η οποία προμήθευε την Epic Games, όμως οι μηχανές αυτές ακόμα ήταν άγνωστες για τους developers καθώς οι μηχανές γραφικών χρησιμοποιούνταν ήδη από την δεκαετία του '80, αλλά μόνο για 2D. Οι μηχανές όμως αυτές δεν έδιναν ελευθερία στον προγραμματιστή, αφού διατηρούσαν κλειδωμένο τον βασικό μηχανισμό.

Μερικές από τις πρώτες μηχανές αναζήτησης αποτελούν:

- Pinball Construction Set (1983)
- ASCII's War Game Construction Kit (1983)
- Adventure Construction Set (1984)
- Shooter-Up Construction Kit (1987)



Εικόνα 1.1.1 - ASCII's War Game Construction Kit (1983)



Εικόνα 1.1.2 - Adventure Construction Set (1984)

Τι είναι η Unity;

Η Unity 3D engine αποτελεί μία δωρεάν μηχανή δημιουργίας παιχνιδιών και ίσως τη δημοφιλέστερη. Η διαφορά της με τις άλλες μηχανές οι οποίες είναι και αυτές ελεύθερες είναι ότι έχει πολύ μεγαλύτερο community υποστήριξης, δωρεάν asset store όπου ένας αρχάριος developer σαν και εμένα έχει τη δυνατότητα να βρει και να κατεβάσει οτιδήποτε 3D asset θέλει να προσθέσει στο παιχνίδι του και τέλος το User Interface της είναι διαμορφωμένο έτσι ώστε να κάνει την εμπειρία του χρήστη ευχάριστη και εύκολη ταυτόχρονα.

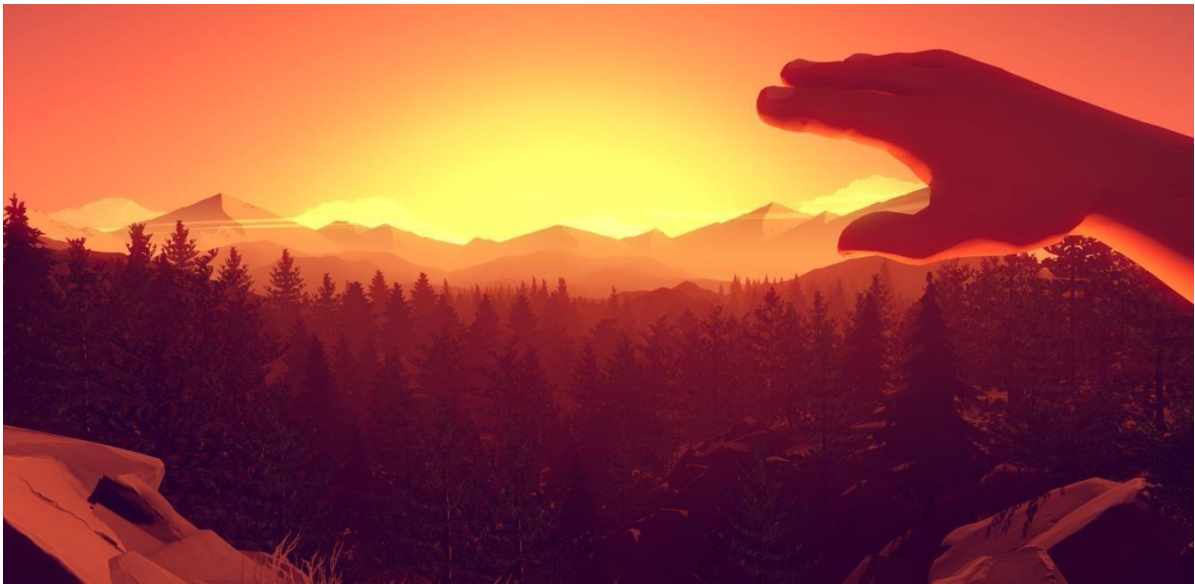
Η μηχανή παιχνιδιών Unity προσφέρει δύο γλώσσες προγραμματισμού για την ανάπτυξη βιντεοπαιχνιδιών, τη Javascript και τη C Sharp (C#). Για τη δημιουργία του βιντεοπαιχνιδιού της πτυχιακής χρησιμοποιήθηκε η C#, μια απλή, μοντέρνα, γενικού σκοπού και αντικειμενοστραφής γλώσσα προγραμματισμού.

Μερικά από τα πιο δημοφιλή παιχνίδια που έχουν δημιουργηθεί πάνω στην Unity 3D(όχι την δωρεάν έκδοση) είναι τα εξής:

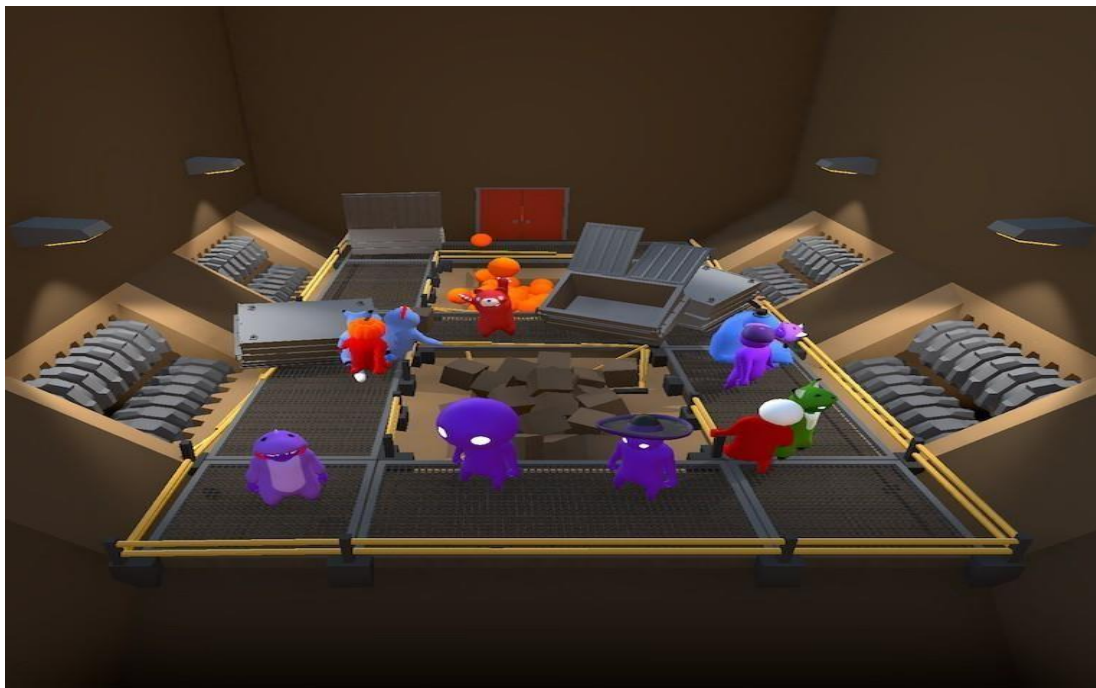
- Firewatch
- Ori and The Blind Forest
- Game Beasts
- SUPERHOT



Εικόνα 1.2.1 – SUPERHOT



Εικόνα 1.2.2 – Firewatch



Εικόνα 1.2.3 – Game Beasts



Εικόνα 1.2.4 – Ori and The Blind Forest

Ιστορική αναδρομή της Unity

Η Unity έκανε τα πρώτα της βήματα σαν εταιρεία το 2004 στην Δανία από τρεις προγραμματιστές, τον Joachim Ante, David Helgason και τον Nicholas Francis. Η επιτυχία της ήταν καθαρά αποτέλεσμα καινοτομίας πάνω στο γεγονός ότι την εποχή εκείνη πολλοί προγραμματιστές δεν μπορούσαν να φτιάξουν την δικιά τους μηχανή έτσι η unity προσέφερε μία ελευθερία σε πολλούς μελλοντικούς developers. Σύμφωνα με έρευνες χρησιμοποιείται πάνω από το 55% των προγραμματιστών που ασχολούνται εξ ολοκλήρου με την gaming βιομηχανία σε iOS και Android.

Η επιτυχία αυτή οφείλεται και στην δημιουργία των iPhone και του Appstore, διότι τότε παρατηρήθηκε μεγάλη προσαύξηση της χρήσης της. Τέλος, αποτελεί μία μηχανή με συνεχή υποστήριξη καθώς ανανεώνεται συνέχεια με καινούρια versions όπως το τελευταίο 2019.1.1 όπου προστέθηκε και η VFX Shaders δυνατότητα για τους χρήστες όπου μπορούν πλέον να δημιουργήσουν Visual Effects με Shaders χωρίς τον παραμικρό κώδικα από αυτούς.

Ρόλοι

Publisher

Ένας εκδότης βιντεοπαιχνιδιών είναι μια εταιρεία που εκδίδει βιντεοπαιχνίδια τα οποία είτε έχουν αναπτυχθεί εσωτερικά είτε έχουν αναπτυχθεί από εξωτερικό προγραμματιστή παιχνιδιών βίντεο. Όπως και με τους εκδότες βιβλίων ή τους εκδότες ταινιών DVD, οι εκδότες βιντεοπαιχνιδιών είναι υπεύθυνοι για την παραγωγή και την εμπορία του προϊόντος τους, συμπεριλαμβανομένης της έρευνας αγοράς και όλων των πτυχών της διαφήμισης. Όλες οι εταιρίες που δημιουργούν τύπου AAA παιχνίδια έχουν μεγάλο budget από τους εκδότες τους για την διαφήμιση του προϊόντος τους σε αντίθεση με τους indie developers.

Producer

Οι παραγωγοί έχουν σαν κύριες ευθύνες την παρακολούθηση της ομάδας προγραμματισμού, αναφορά τυχών προβλημάτων, πρόσληψη προσωπικού, συντήρηση του προϋπολογισμού, ακόμη και τον προγραμματισμό κρίσιμων κομματιών του παιχνιδιού. Γενικότερα αποτελούν ένα αναγκαίο συνδετικό κρίκο μεταξύ εργαζομένων και υφισταμένων έτσι ώστε να εξασφαλιστεί ο στόχος της εταιρείας όσο το δυνατόν πιο ομαλά και πετυχημένα.

Development Team

Η ομάδα ανάπτυξης μπορεί να αποτελείται από 20 μέχρι και 500 άτομα σε μία εταιρεία, πράγμα που εξαρτάται καθαρά από το στόχο της εταιρίας για το παιχνίδι που θα αναπτύξει. Οι ρόλοι που απαρτίζουν την συγκεκριμένη ομάδα είναι πολλοί όπως για παράδειγμα οι δοκιμαστές παιχνιδιών, ηθοποιοί. Ωστόσο οι σημαντικότεροι developers που συναντούμε σε μία τέτοια ομάδα είναι οι παρακάτω:

Artists

Ένας καλλιτέχνης παιχνιδιών είναι ουσιαστικά ο υπεύθυνος για την δημιουργία της τέχνης του παιχνιδιού. Ένας 2D καλλιτέχνης είναι υπεύθυνος για την δημιουργία των υφών (textures), περιβάλλοντα backdrops, περιβάλλοντα sprites κτλ. Ένας 3D καλλιτέχνης είναι υπεύθυνος για την δημιουργία τρισδιάστατων μοντέλων, cinematics κτλ.

Programmer

Οι προγραμματιστές καλούνται να κάνουν το παιχνίδι λειτουργικό, δηλαδή μέσω του κώδικα δίνουν ζωή στο όραμα των game designer και στα δημιουργήματα των καλλιτεχνών που ανέφερα παραπάνω. Οι προγραμματιστές είναι χωρισμένοι σε ομάδες συνήθως βάση με το ποιο κομμάτι κώδικα έχει αναλάβει ο καθένας. Κάποια από αυτά είναι τα εξής :

- Τεχνητή νοημοσύνη (AI)
- Ήχος (Sound)
- Φυσική (Physics)
- Γραφικά (Graphics)
- Gameplay generalist (Διάφορα mechanics του παιχνιδιού)
- Δίκτυα
- Διεπαφή χρήστη (UserInterface)
- Επεξεργασία εισόδου (Input processing)

Level Designer

Ένας σχεδιαστής επιπέδου είναι ένα πρόσωπο που δημιουργεί επίπεδα, προκλήσεις ή αποστολές για το παιχνίδι και βίντεο χρησιμοποιώντας ένα σύνολο προγραμμάτων. Αποτελεί σημαντικό επίπεδο στην ανάπτυξη ενός παιχνιδιού που έχει άμεσο αντίκτυπο στην εμπειρία του παίκτη και τη ψυχολογία του κατά τη διάρκεια που αυτός παίζει.

Sound Designer

Είναι οι τεχνικοί επαγγελματίες υπεύθυνοι για την δημιουργία ηχητικών εφέ και της τοποθέτησής τους στο χώρο του παιχνιδιού. Πολλές φορές επιβλέπουν την φωνητική ηχογράφηση των ηθοποιών και πολλές διεργασίες ακόμα.

Tester

Είναι η ομάδα που καλείται να ενημερώσει την ομάδα ανάπτυξης αν το παιχνίδι είναι έτοιμο, αν είναι διασκεδαστικό για τον χρήστη ή αν είναι ελαττωματικό και πώς μπορεί αυτό να αλλάξει. Απαιτεί καλές γνώσεις πληροφορικής και αντίληψης του κόσμου των βίντεο-παιχνιδιών για όσο το δυνατόν αποτελεσματικότερη και σαφέστερη μεταβίβαση της εμπειρίας στους developers.

Κεφάλαιο 2: User Interface Unity 3D

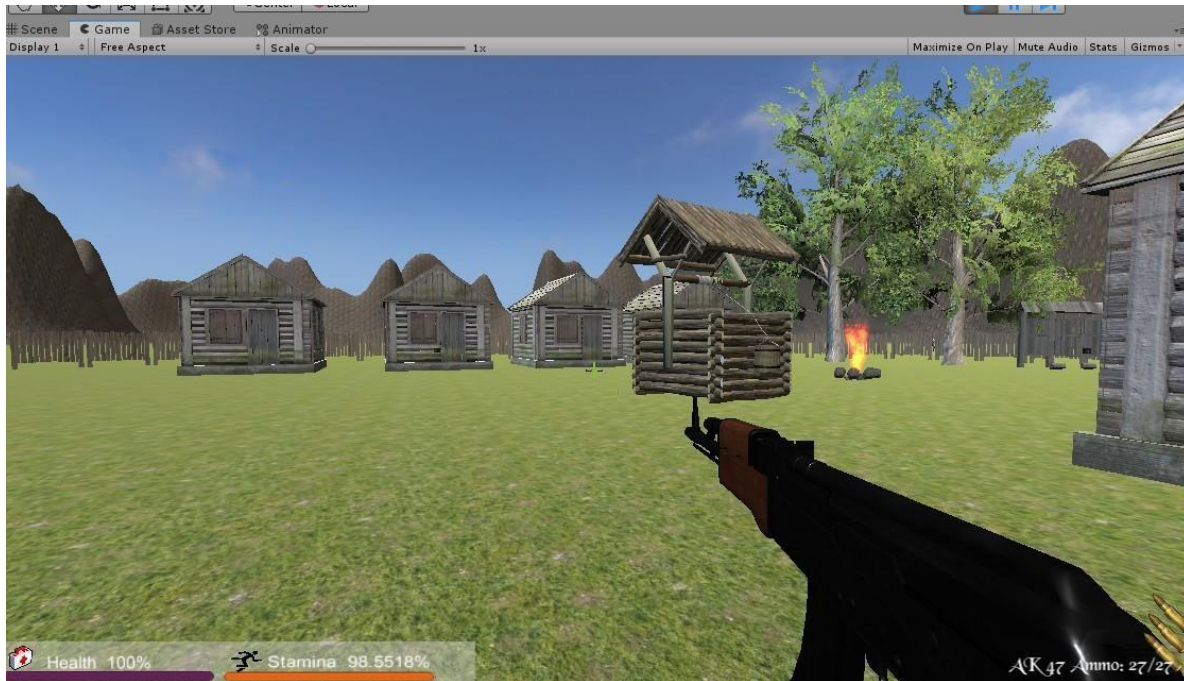
Scene View



Εικόνα 2.1.1 – 3D όψη σκηνής

Η όψη σκηνής είναι πιθανώς αυτή που ένας developer χρησιμοποιεί περισσότερο καθώς μέσω αυτής μπορεί να εισάγει αντικείμενα στο παιχνίδι (π.χ. Characters, buildings) τα τοποθετεί ακριβώς εκεί που αυτός πιστεύει ότι ανήκουν με τη σωστή φορά (rotation) και επίσης μπορεί να διαμορφώσει το level του παιχνιδιού (π.χ. προσθέτοντας δέντρα, βουνά, λίμνες). Η μηχανή διαθέτει και διάφορα shortcuts όσο αναφορά την χρήση της scene όψης που ο προγραμματιστής εξοικειώνεται μαζί τους όλο και περισσότερο όσο ασχολείται με αυτή.

Game View

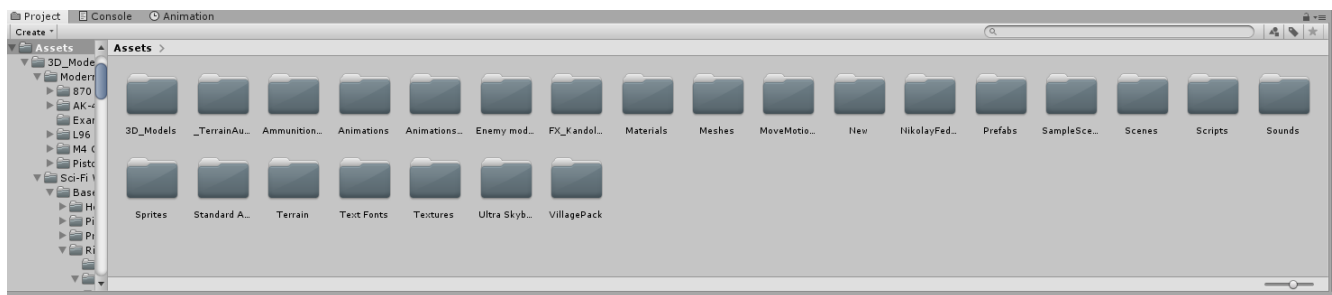


Εικόνα 2.2.1- Όψη Παιχνιδιού

Η όψη παιχνιδιού είναι ουσιαστικά η οπτική μέσα από την οποία θα βλέπει ο χρήστης-παίκτης όταν θα ξεκινήσει το παιχνίδι και κατά την διάρκεια του. Στο πάνω δεξί μέρος της οθόνης φαίνονται κάποιες επιλογές οι οποίες δίνουν τη δυνατότητα στον developer η μεγιστοποίηση παραθύρου, η αλλαγή ανάλυσης του παιχνιδιού, λειτουργίες που αφορούν τα FPS του παιχνιδιού που βρίσκεται υπό ανάπτυξη, η σίγαση επιλεγμένων αντικειμένων και πολλές λειτουργίες πιο εξειδικευμένες για προχωρημένους προγραμματιστές.

Ένα παιχνίδι έχει πολλές σκηνές και είναι δουλειά του developer να τις τοποθετήσει με σειρά για την καλύτερη εμπειρία του χρήστη. Για παράδειγμα, ένα παιχνίδι θα ξεκινήσει με την πρώτη σκηνή να είναι το Menu (play, options, multiplayer, quit) και ανάλογα με την επιλογή του χρήστη η μηχανή θα φορτώσει και θα εμφανίσει την κατάλληλη σκηνή αμέσως.

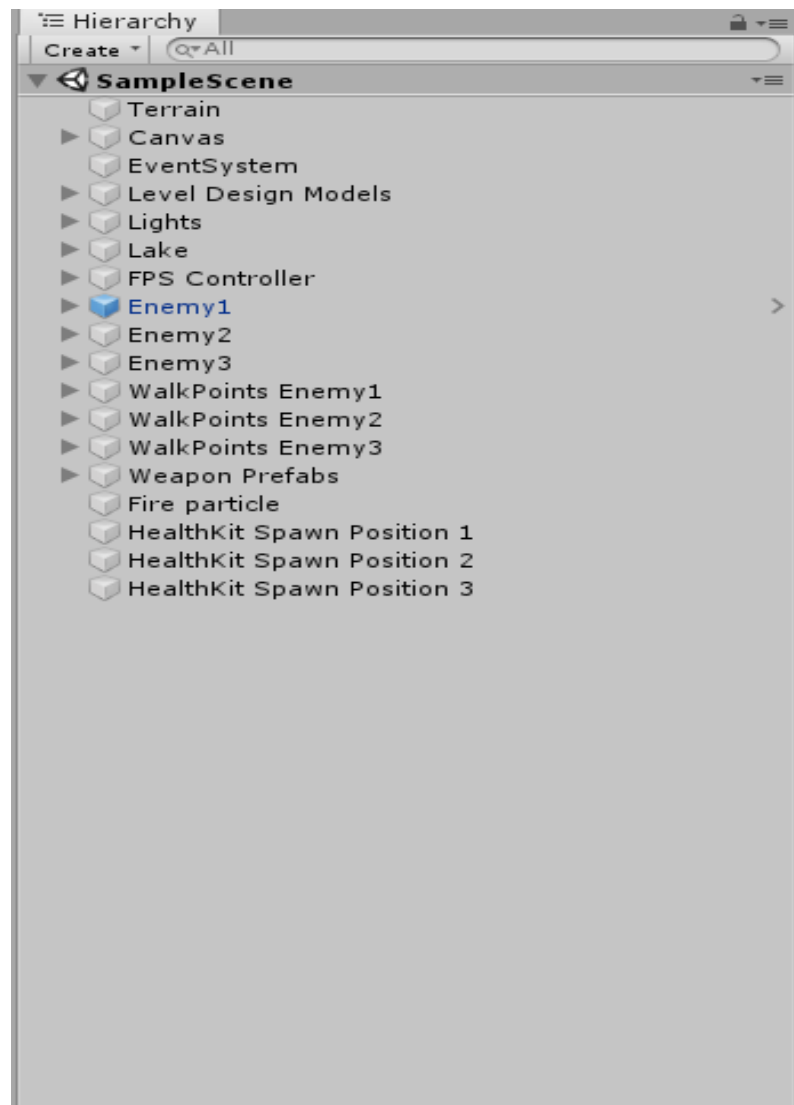
Project View



Εικόνα 2.3.1 – Όψη Έργου

Η παραπάνω όψη έχει να κάνει με τα assets του παιχνιδιού όπως για παράδειγμα scripts, 3D models, Sprites κτλ. Κατά τη γνώμη μου η τοποθέτηση αυτών σε ξεχωριστούς φακέλους ανάλογα του είδους τους είναι η καλύτερη τακτική για την ταχύτερη ανάπτυξη του παιχνιδιού και την αποφυγή λαθών.

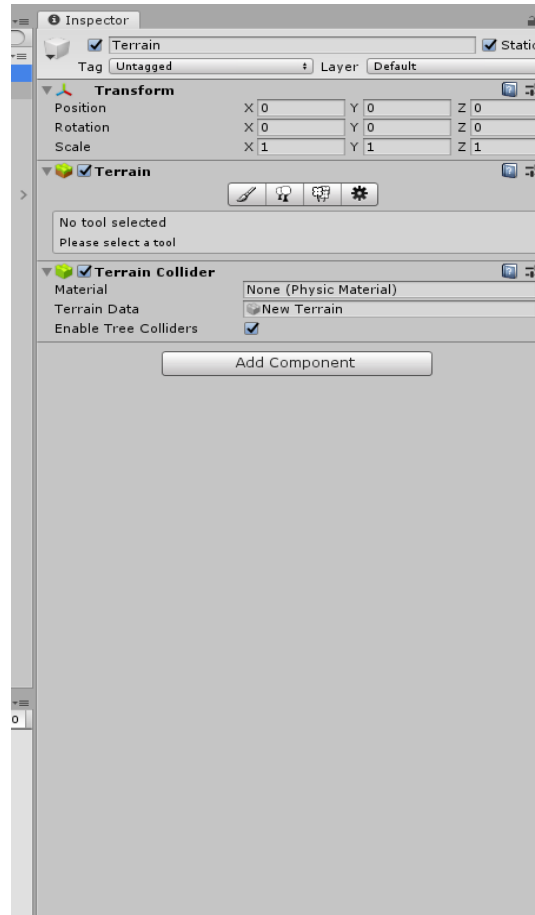
Hierarchy



Εικόνα 2.4.1 - Ιεραρχία

Στην όψη της ιεραρχίας βρίσκονται τα αντικείμενα που βρίσκονται στον 3D κόσμο του παιχνιδιού. Ο developer μπορεί να τα επιλέξει, να τα ομαδοποιήσει και γενικά να τα διαχειριστεί.

Inspector

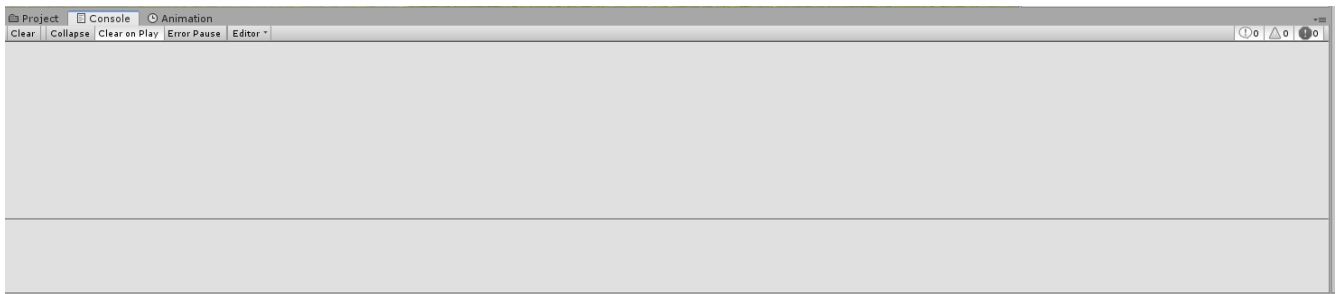


Εικόνα 2.5.1 - Inspector

Στην όψη ελέγχου ο χρήστης μπορεί να παρατηρήσει επεξεργαστεί και να αποθηκεύσει ορισμένες ιδιότητες όσο αναφορά ένα αντικείμενο (π.χ. Μέγεθος, φορά, θέση στον 3d κόσμο).

Στην εικόνα 2.5.1 φαίνεται τα στοιχεία του terrain του παιχνιδιού, μέσω των οποίων κατάφερα και το διαμόρφωσα. Υπάρχουν ορισμένες επιλογές μέσω των οποίων ανόρθωσα και χρωμάτισα το δάπεδο ώστε να δημιουργηθούν βουνά, χαμήλωσα και χρωμάτισα το δάπεδο για να δημιουργηθούν οι κατάλληλες συνθήκες λίμνης αφού αργότερα πρόσθεσα και το water pack που προσφέρεται από το Unity environment pack και τέλος πρόσθεσα δέντρα σε διάφορα σημεία της σκηνής. Όσο αναφορά την διαμόρφωση του χάρτη, θα αναλυθεί στο επόμενο κεφάλαιο που αφορά την ανάπτυξη και τη σχεδίαση του παιχνιδιού.

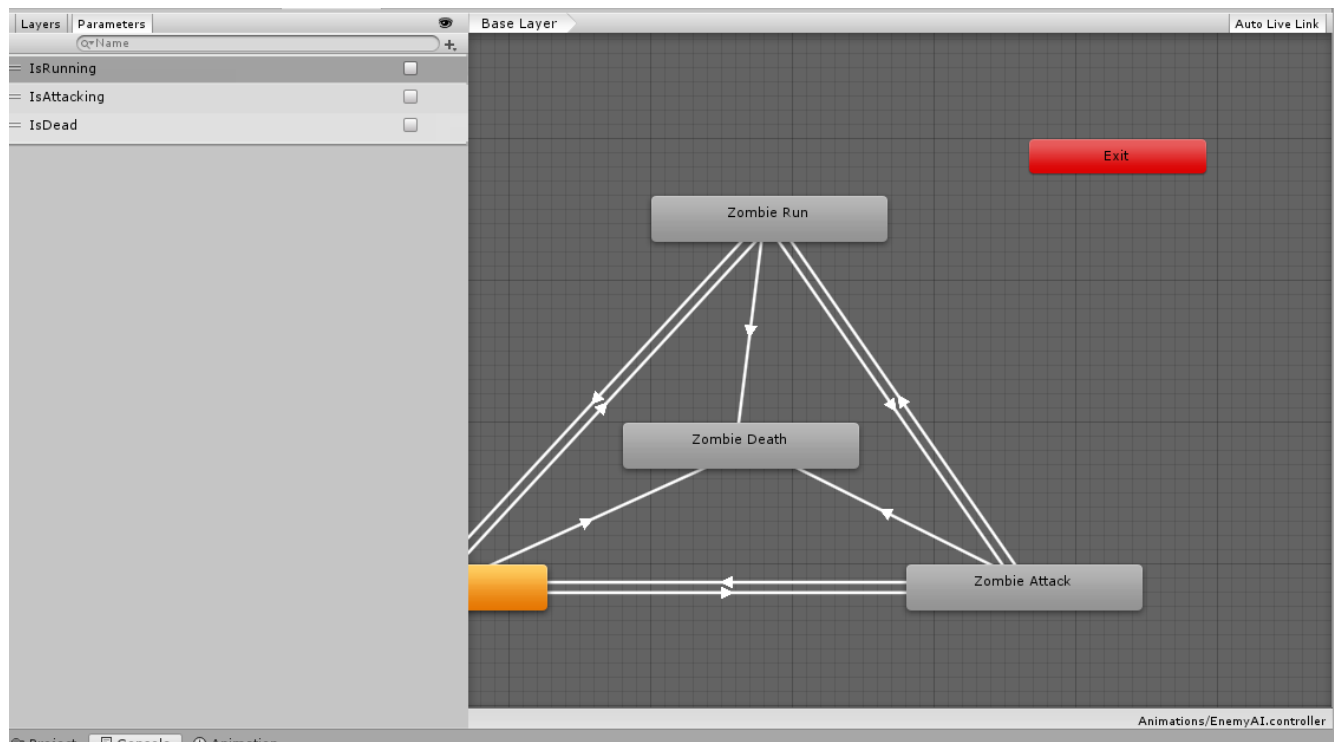
Console



Εικόνα 2.6.1 – Κονσόλα

Η όψη της κονσόλας αποτελεί ένα σημαντικό feature της unity καθώς βοηθάει τον προγραμματιστή να καταλάβει που βρίσκεται και αν όλα λειτουργούν σωστά. Εμφανίζει μηνύματα που έχει βάλει ο προγραμματιστής σε συγκεκριμένα σημεία του κώδικα και πιο σημαντικό, εμφανίζει τυχόν σφάλματα στον κώδικα καθώς και το σημείο στο οποίο βρίσκονται προσφέροντας επίσης την δυνατότητα ανακατεύθυνσης σε πραγματικό χρόνο του developer στο σημείο του σφάλματος του κώδικα ή και του μηνύματος αν επιλέξει πάνω στο feedback της console.

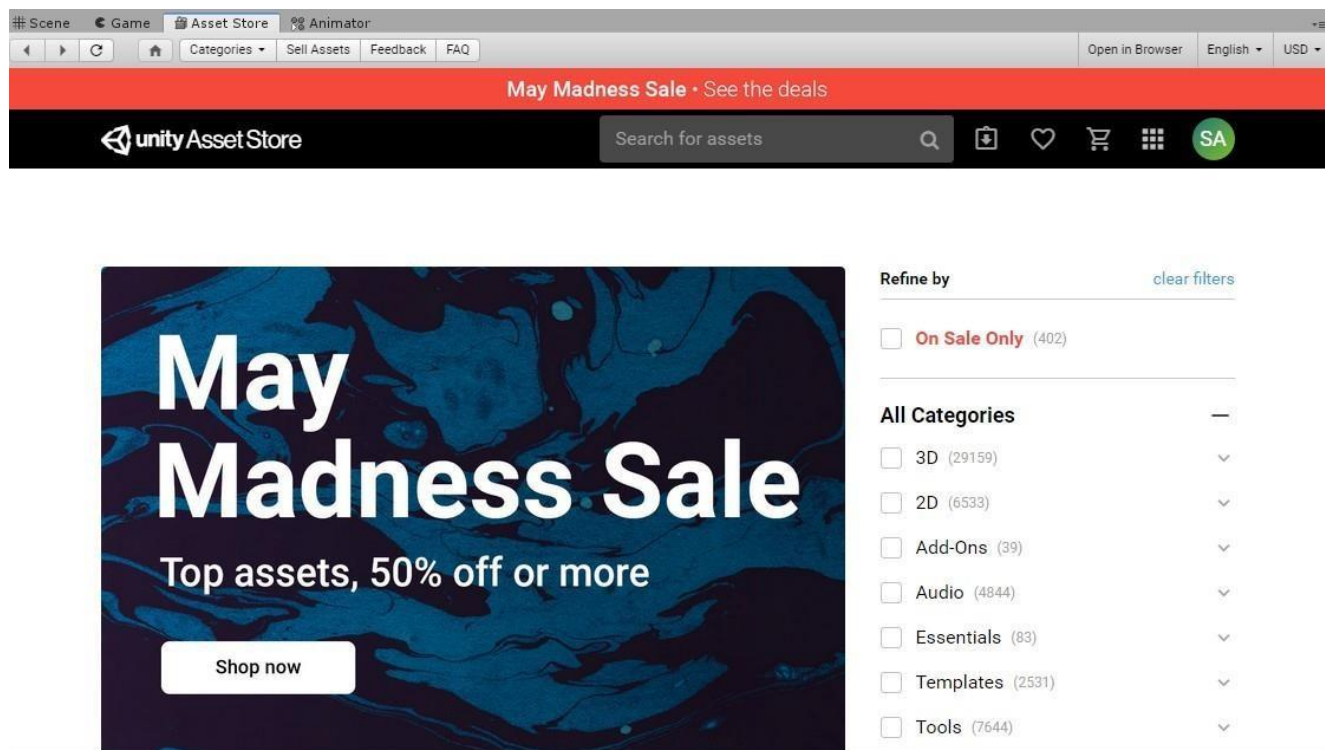
Animator



Εικόνα 2.7.1 – Animator

Ο animator όπως το λέει και η λέξη ασχολείται με τα animations δηλαδή κινήσεις χαρακτήρων και αντικειμένων που υπάρχουν μέσα στο παιχνίδι. Το animation που είναι χρωματισμένο με πορτοκαλί χρώμα είναι το αρχικό animation που θα έχει ένα αντικείμενο και για να αλλάξει η κατάστασή του χρησιμοποιούμε λογικές συνθήκες (π.χ. IsRunning, IsAttacking, IsDead) μέσω του κώδικα μας που όταν πάρουν τιμές αληθείς γίνεται η μετάβαση από ένα animation σε άλλο.

Asset Store



Εικόνα 2.8.1 – Asset Store

Η όψη του Asset Store επιτρέπει στο χρήστη να αναζητήσει διάφορα μοντέλα (π.χ. χαρακτήρες, όπλα, κτήρια, αντικείμενα) και να τα εισάγει στο έργο του είτε με πληρωμή είτε και δωρεάν αναλόγως την τιμή τους. Η Unity δεν παρέχει δυνατότητα σχεδιασμού 3D γραφικών για αυτό το παραπάνω κατάστημα αποτελεί μια χείρα βοηθείας για όσους developers δεν έχουν γνώσεις άλλων προγραμμάτων σχεδίασης.

Πολλά από τα μοντέλα που έχω μέσα στο project μου έχουν κατεβεί από το Unity Asset Store.

Tools – Εργαλεία της Unity

Translate Tool



Χρησιμεύει στην μετακίνηση ενός αντικειμένου προς τη διευθύνσεις x,y,z.



Rotate Tool

Χρησιμεύει στην περιστροφή ενός αντικειμένου ως προς τους άξονες x,y,z.

Scale Tool



Χρησιμεύει στην μεγιστοποίηση ή ελαχιστοποίηση του μεγέθους ενός αντικειμένου ως προς τους άξονες x,y,z.

Navigation Tool



Κρατώντας πατημένο το αριστερό κλικ του ποντικιού μετακινούμε την κάμερα αριστερά, δεξιά, πάνω και κάτω. Πιέζοντας ταυτόχρονα και το πλήκτρο alt μπορούμε να κάνουμε περιστροφή γύρω από το αντικείμενο που έχουμε εστιάσει. Πιέζοντας αυτή την φορά το δεξί κλικ του ποντικιού περιστρέφουμε την κάμερα προς την επιλεγμένη διεύθυνση που του δίνουμε (αριστερά, δεξιά, πάνω, κάτω) ενώ πατώντας ταυτόχρονα και το alt κάνουμε ζουμ.

Κεφάλαιο 3 : Ανάπτυξη του παιχνιδιού

Από όλους τους παραπάνω ρόλους εγώ ουσιαστικά ασχολήθηκα με το level design και κυρίως το programming κομμάτι του παιχνιδιού εφόσον το game design ήταν προκαθορισμένο από την προσφώνηση της εργασίας. Εδώ θα ήθελα να αναφέρω ότι όλα τα Assets που χρησιμοποιήθηκαν προσφέρονται δωρεάν από το Unity Asset Store ή από άλλα online sites καθώς δεν κατέχω καμία γνώση δημιουργίας τρισδιάστατων μοντέλων προς το παρόν. Ακόμη τα μοντέλα που χρησιμοποιήθηκαν μέσα στο παιχνίδι είναι σχετικά μικρά σε αριθμό για λόγους καθαρά βελτιστοποίησης και ομαλής εργασίας πάνω στη μηχανή. Τέλος όλα τα περιουσιακά στοιχεία που χρησιμοποιήθηκαν για την εργασία μου υπάρχουν στη βιβλιογραφία στο τέλος της αναφοράς.

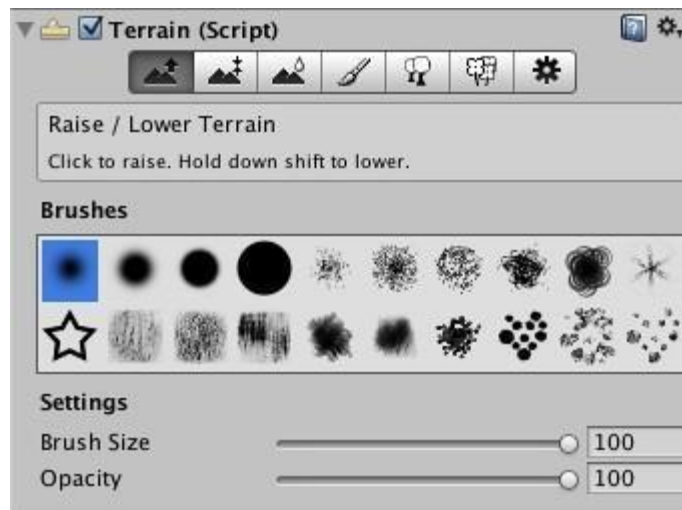
Level Design



Εικόνα 3.1.1 – Χωριό

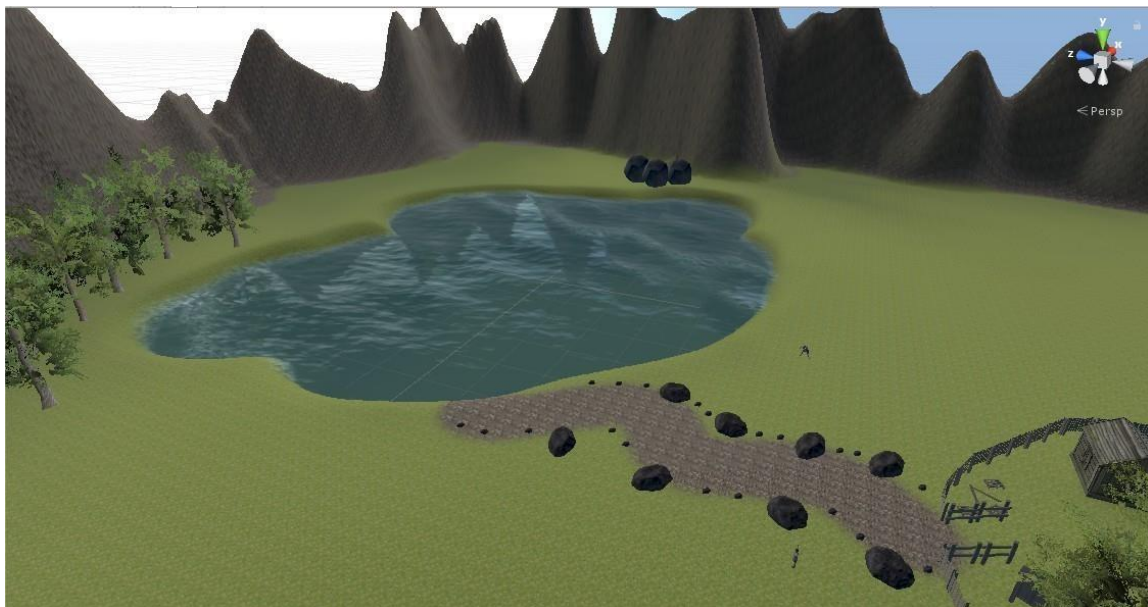
Εδώ να αναφέρουμε ότι το level design είναι ένας απαιτητικός τομέας πάνω στο gaming development και έχοντας υπόψη την μηδενική εμπειρία μου και το hardware του υπολογιστή μου, η σχεδίαση του χάρτη έγινε απλά και πειραματικά για όσο το δυνατόν ομαλή χρήση της μηχανής αποφεύγοντας την καθυστέρηση φορτώματος των σκηνών και την υπερθέρμανση του επεξεργαστή.

Η Unity μας δίνει την δυνατότητα με απλές κινήσεις να φτιάξει εύκολα και γρήγορα το Terrain όπως ο developer το έχει φανταστεί. Για να δημιουργηθεί το Terrain ο χρήστης πρέπει να πατήσει στην επιλογή Game Object – 3d Object -Terrain, και αμέσως δημιουργείται ένα αντικείμενο Terrain εμφανίζοντας το στο panel Hierarchy. Το Object Terrain είναι από τα Default της Unity και έχει ενσωματωμένο Script όπου από εκεί θα βρούμε τα εργαλεία και θα κάνουμε πιο εύκολο τον σχεδιασμό του Map.



Εικόνα 3.1.2 – Inspector options

Με τις επιλογές αυτές σχεδίασα όλο τον χάρτη του παιχνιδιού που εν μέρει αποτελείται από βουνά, μία λίμνη, ένα δάσος και δέντρα διάσπαρτα σε όλο το μήκος του χάρτη.

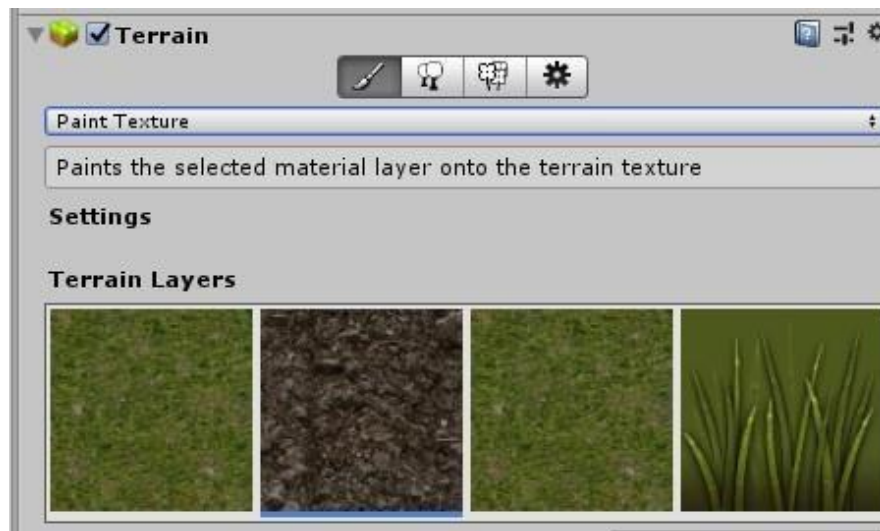


Εικόνα 3.1.3 – Λίμνη



Εικόνα 3.1.4 – Μονοπάτι προς το Δάσος

Οι υφές που παρατηρούμε στο έδαφος είναι και αυτή μία επιλογή που μας δίνεται από την Unity. Έτσι χρησιμοποίησα καφέ υφή για τα βουνά, πράσινη για το έδαφος και καφέ χωμάτινη πάλι για την δημιουργία μονοπατιών στις εξόδους του χωριού.



Εικόνα 3.1.5 – Textures

Γενικότερα στα 3D παιχνίδια οι περισσότεροι χαρακτήρες και τα αντικείμενα που βρίσκονται στη σκηνή μας αντιπροσωπεύονται από meshes ενώ στα 2D παιχνίδια από sprites. Ωστόσο υπάρχουν και άλλες οντότητες στην φύση που θα λέγαμε ότι είναι περισσότερο “υγρά” και “άυλα” και είναι δύσκολο να τα αναπαραστήσουμε στο 3D κόσμο με sprites και meshes. Τέτοια παραδείγματα αποτελούν η φωτιά, το νερό, ο καπνός, σύννεφα, μαγικά ξόρκια κτλ. Αυτό το σκοπό στην Unity εξυπηρετεί το Particle systems που χάρη σε αυτό κατάφερα να τοποθετήσω μία φωτιά στο κέντρο του χωριού.

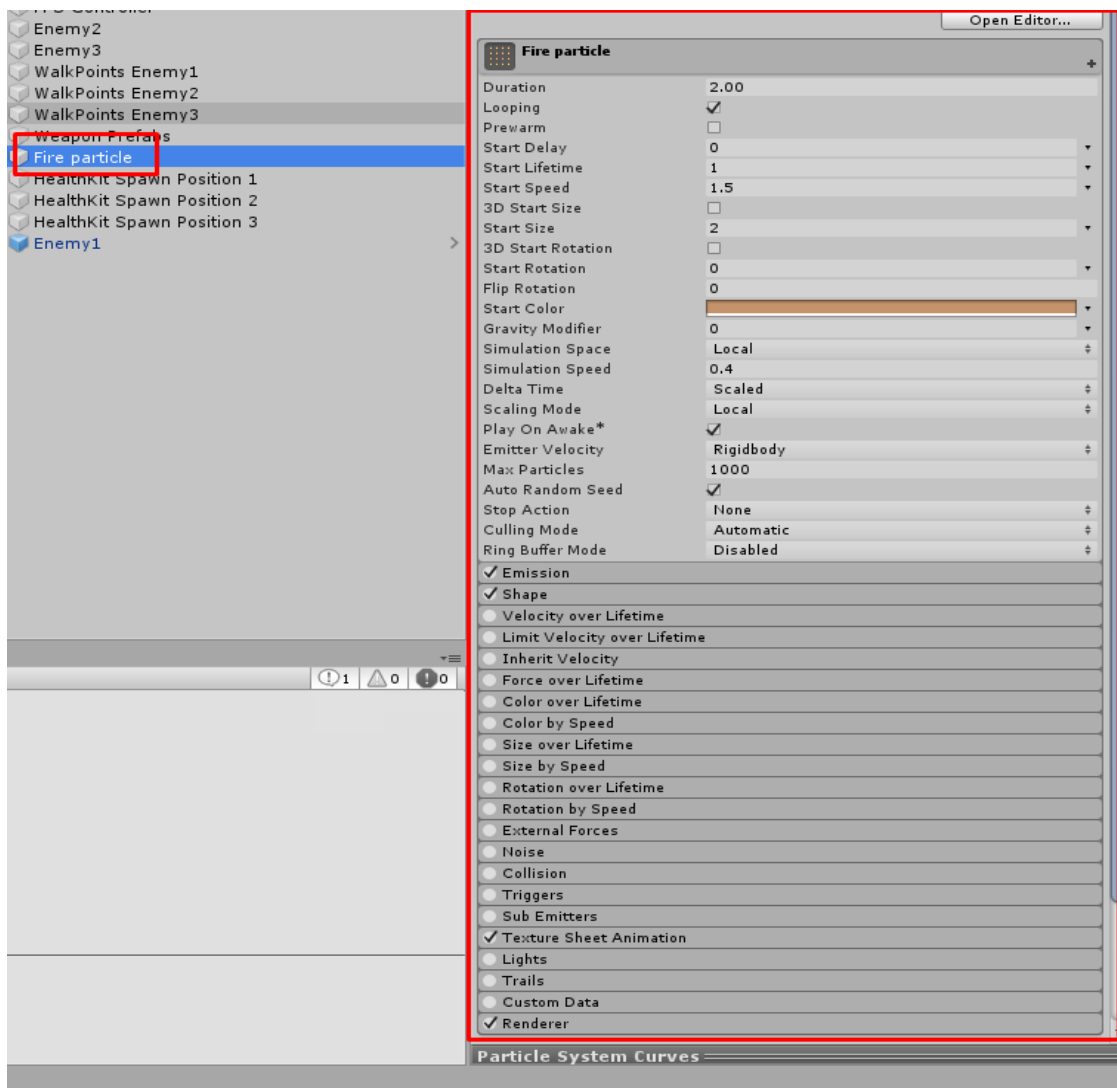


Εικόνα 3.1.6 – CampFire (Game View)



Εικόνα 3.1.7 – CampFire (Scene View)

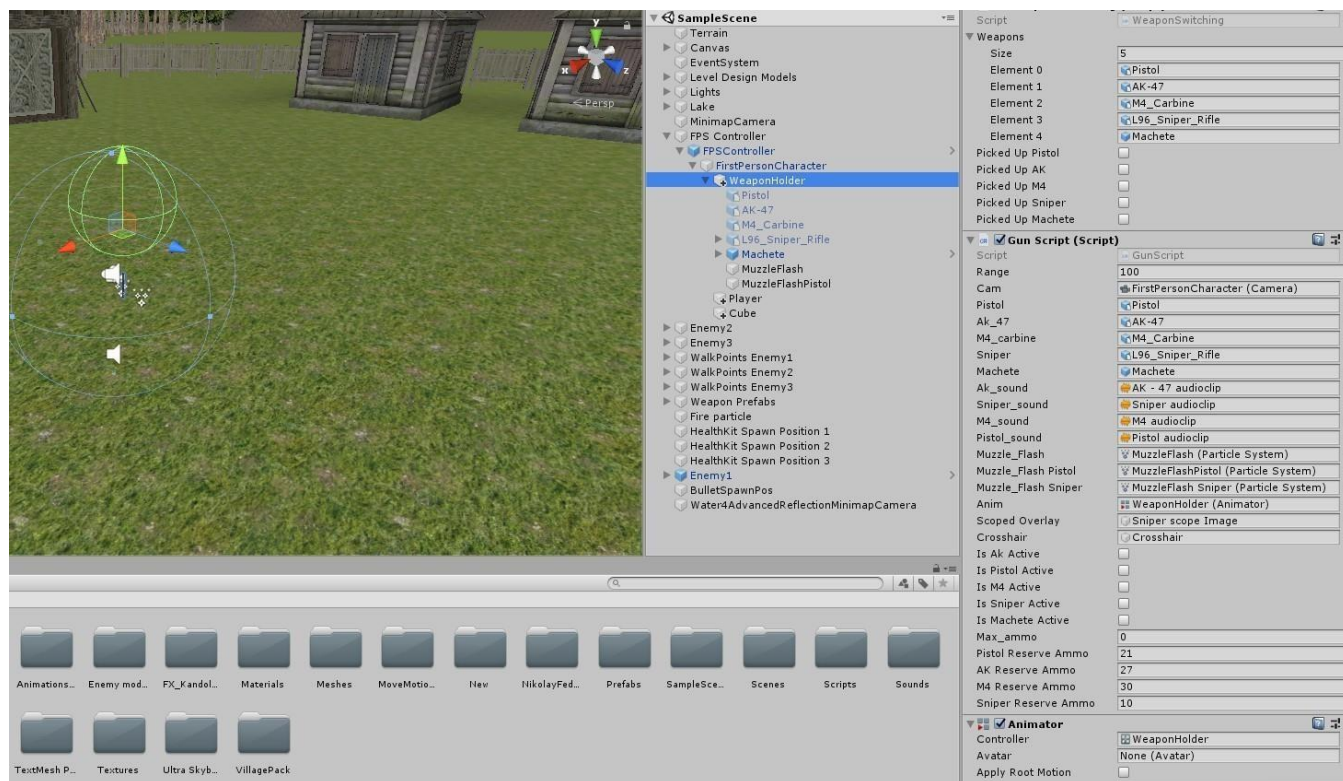
Το Particle System αποτελεί ένα σχετικά δύσκολο κομμάτι της Unity λόγω της περιπλοκότητας των επιλογών της αλλά αυτό δεν αποτελεί αρνητικό στοιχείο σε σύγκριση με της επιλογές που προσφέρει. Ειδικότερα μέσω αυτών ο χρήστης μπορεί να ορίσει την διάρκεια, την ώρα που θα ξεκινήσει και την ώρα που θα τελειώσει, το χρώμα, την επαναληπτικότητα του particle καιπολλάακόμη.



Εικόνα 3.1.7 – Particle system

Programming

Για την δημιουργία του παίκτη που θα χειρίζεται ο χρήστης χρησιμοποίησα τον FPScontroller που προσφέρει η unity στο πακέτο Characters ο οποίος είναι ήδη προγραμματισμένος από τους δημιουργούς της μηχανής να κινείται με τα πλήκτρα W A S D και να κάνει άλματα με το πλήκτρο Space. Δημιούργησα σαν παιδί (child) του FPScontroller ένα κενό αντικείμενο (empty object---> WeaponHolder) το οποίο θα διαχειρίζεται τα όπλα του παίκτη μέσω ενός script που έγραψα. Τα όπλα και αυτά με τη σειρά τους είναι παιδιά του αντικειμένου αυτού.



Εικόνα 3.2.1 – FPScontroller

Το WeaponHolder όπως βλέπουμε και στη παραπάνω φωτογραφία έχει τον Animator που διαχειρίζεται τα animations των όπλων όπως θα δούμε παρακάτω, έχει τον audiosource που διαχειρίζεται τον ήχο που βγάζει το κάθε όπλο όταν ο παίκτης πυροβολεί και προσκολλημένα 2 scripts το GunScript και το WeaponSwitching τα οποία διαχειρίζονται την συμπεριφορά πολλών μεταβλητών όπως για παράδειγμα damage, firerate κτλ. Ας ξεκινήσουμε λοιπόν με το WeaponSwitching script:

WeaponSwitching

```

public class WeaponSwitching : MonoBehaviour
{
    public GameObject[] weapons;
    public bool pickedUpPistol, pickedUpAK, pickedUpM4, pickedUpSniper, pickedUpMachete;

    void Start()
    {
        weapons[0].SetActive(false);
        weapons[1].SetActive(false);
        weapons[2].SetActive(false);
        weapons[3].SetActive(false);
        weapons[4].SetActive(false);
    }

    void Update() //Thetoume ta buttons me ta opoia theloume na emfanizoume to oplo pou kratame,koitame episis an to oplo to exoume kanei pick up oste na to xrisimopoisoume
    {
        if (Input.GetKeyDown("1") && pickedUpPistol == true)
        {
            weapons[0].SetActive(true);
            weapons[1].SetActive(false);
            weapons[2].SetActive(false);
            weapons[3].SetActive(false);
            weapons[4].SetActive(false);
        }
        else if (Input.GetKeyDown("2") && pickedUpAK == true)
        {
            weapons[0].SetActive(false);
            weapons[2].SetActive(false);
            weapons[3].SetActive(false);
            weapons[4].SetActive(false);
            weapons[1].SetActive(true);
        }
        else if (Input.GetKeyDown("3") && pickedUpM4 == true)
    }
}

```

Εικόνα 3.2.1.2 – WeaponSwitching Script

```

25         weapons[1].SetActive(false);
26         weapons[2].SetActive(false);
27         weapons[3].SetActive(false);
28         weapons[4].SetActive(false);
29     }
30     else if (Input.GetKeyDown("2") && pickedUpAK == true)
31     {
32         weapons[0].SetActive(false);
33         weapons[2].SetActive(false);
34         weapons[3].SetActive(false);
35         weapons[4].SetActive(false);
36         weapons[1].SetActive(true);
37     }
38     else if (Input.GetKeyDown("3") && pickedUpM4 == true)
39     {
40         weapons[0].SetActive(false);
41         weapons[1].SetActive(false);
42         weapons[2].SetActive(true);
43         weapons[3].SetActive(false);
44         weapons[4].SetActive(false);
45     }
46     else if (Input.GetKeyDown("4") && pickedUpSniper == true)
47     {
48         weapons[0].SetActive(false);
49         weapons[1].SetActive(false);
50         weapons[2].SetActive(false);
51         weapons[3].SetActive(true);
52         weapons[4].SetActive(false);
53     }
54     else if (Input.GetKeyDown("5") && pickedUpMachete == true)
55     {
56         weapons[0].SetActive(false);
57         weapons[1].SetActive(false);
58         weapons[2].SetActive(false);
59         weapons[3].SetActive(false);
60         weapons[4].SetActive(true);
61     }
}

```

Εικόνα 3.2.1.3 – WeaponSwitching Script

Στο παραπάνω script έχω δημιουργήσει ένα πίνακα όπου το κάθε στοιχείο του αντιστοιχεί και σε ένα από τα όπλα του παιχνιδιού. Έτσι ο παίχτης όταν θα πατάει ένα από τα εξής πλήκτρα (1),(2),(3),(4),(5) το script είναι υπεύθυνο να ενεργοποιήσει το αντίστοιχο όπλο στο Hierarchy και Game View. Για παράδειγμα με το πλήκτρο (2) ενεργοποιείται το όπλο Kalashnikov και απενεργοποιούνται τα υπόλοιπα.

Στη συνέχεια έγραψα και επικόλλησα στον FPScontroller το GunScript :

Gunscript

```

public class GunScript : MonoBehaviour
{
    private float nextTimeToFire = 0f;
    private float fireRate;
    public float range = 100f;
    private int Damage;
    public Camera cam;
    public GameObject pistol;
    public GameObject ak_47;
    public GameObject m4_carbine;
    public GameObject sniper;
    public GameObject machete;
    public AudioClip ak_sound;
    public AudioClip sniper_sound;
    public AudioClip m4_sound;
    public AudioClip pistol_sound;
    public ParticleSystem muzzle_Flash;
    public ParticleSystem muzzle_FlashPistol;
    public ParticleSystem muzzle_FlashSniper;
    public Animator anim;
    public GameObject scopedOverlay;
    public GameObject crosshair;

    public bool isAkActive, isPistolActive, isM4Active, isSniperActive, isMacheteActive;
    private bool AkOutOfAmmo, PistolOutOfAmmo, M4OutOfAmmo, SniperOutOfAmmo;
    private bool ScopedIn = false;
    private AudioSource audioSource;
    private Text ammo_txt;
    private ZombieHealth zh;
    private Zombie2Health zh2;
    private Zombie3Health zh3;
    private Camera fieldOfView;

    public int max_ammo=0;
    public int PistolReserveAmmo=21, AKReserveAmmo=27, M4ReserveAmmo=30, SniperReserveAmmo=10;
    private int PistolCurrentAmmo, AKCurrentAmmo, M4currentAmmo, SniperCurrentAmmo;
    private int PistolMaxAmmo = 7, AKMaxAmmo = 27, M4MaxAmmo = 30, SniperMaxAmmo = 5;
}

```

Εικόνα 3.2.2.4 - GunScript

```

private int PistolMaxAmmo = 7, AKMaxAmmo = 27, M4MaxAmmo = 30, SniperMaxAmmo = 5;
private float reload_time = 1.5f;
private bool IsReloading = false;

void Start()
{
    audioSource = GetComponent();
    ammo_txt = GameObject.Find("Ammo Text").GetComponent<Text>();
    zh = GameObject.Find("Enemy1").GetComponent<ZombieHealth>();
    zh2 = GameObject.Find("Enemy2").GetComponent<Zombie2Health>();
    zh3 = GameObject.Find("Enemy3").GetComponent<Zombie3Health>();
    fieldOfView = GameObject.Find("FirstPersonCharacter").GetComponent<Camera>();
    PistolCurrentAmmo = PistolMaxAmmo;
    AKCurrentAmmo = AKMaxAmmo;
    M4currentAmmo = M4MaxAmmo;
    SniperCurrentAmmo = SniperMaxAmmo;
}

void Update()
{
    if (IsReloading) //Edw checkarw an vriskomaste se reloading kai an nai tote thelw na min ektelestei o parakatw kwdikas,otan vgw apo reloading tote ola ektelountai ksana
    {
        return;
    }

    Check();

    if (PistolCurrentAmmo == 0 && PistolOutOfAmmo==false)
    {
        StartCoroutine(Reload());
        //return; //an ksekinisw to reload den thelw na ektelestei o ipolipos kwdikas opote xrisimopoiw tin entolei return
    }
    else if (AKCurrentAmmo == 0 && AkOutOfAmmo==false)
    {
        StartCoroutine(Reload());
        //return; //an ksekinisw to reload den thelw na ektelestei o ipolipos kwdikas opote xrisimopoiw tin entolei return
    }
}

```

Εικόνα 3.2.2.5 - GunScript

```

81     else if (M4currentAmmo == 0 && M4OutOfAmmo==false)
82     {
83         StartCoroutine(Reload());
84         //return; //an ksekinisw to reload den thelw na ektelestei o ipolipos kwdikas opote xrisimopoiw tin entolei return
85     }
86     else if (SniperCurrentAmmo==0 && SniperOutOfAmmo==false)
87     {
88         StartCoroutine(Reload());
89         //return; //an ksekinisw to reload den thelw na ektelestei o ipolipos kwdikas opote xrisimopoiw tin entolei return
90     }
91
92     if (Input.GetButton("Fire1") && (Time.time>nextTimeToFire)&&((isAkActive)||isM4Active)||isPistolActive)||isSniperActive))) //se ayti tin entoli i unit
93     {
94         if(isAkActive == true)
95         {
96             AkRecoilOn();
97         }
98         else if(isM4Active == true)
99         {
100             M4RecoilOn();
101         }
102         else if(isPistolActive == true)
103         {
104             PistolRecoilOn();
105         }
106
107         nextTimeToFire = Time.time + 1 / fireRate;
108         Shoot();
109     }
110     else
111     {
112         AkRecoilOff();
113         M4RecoilOff();
114         PistolRecoilOff();
115         anim.SetBool("IsMeleeAttacking", false);
116     }

```

Εικόνα 3.2.2.6 – GunScript

```

114     }
115
116     if (Input.GetKeyDown("t") && machete.activeInHierarchy == true && machete.activeInHierarchy != false)
117     {
118         anim.SetBool("IsMeleeAttacking", true);
119     }
120
121
122
123
124
125     if (Input.GetButtonDown("Fire2") && isSniperActive == true)
126     {
127         ScopedIn = !ScopedIn;
128         anim.SetBool("IsScoped", ScopedIn);
129     }
130
131     if(ScopedIn == true)
132     {
133         StartCoroutine(OnScoped());
134     }
135     if(ScopedIn == false)
136     {
137         OnUnScoped();
138     }
139
140
141 }
142
143 void OnUnScoped()
144 {
145     fieldOfView.fieldOfView = 60;
146     scopedOverlay.SetActive(false);
147     crosshair.SetActive(true);
148 }
149
150 IEnumerator OnScoped()
151 {
152     yield return new WaitForSeconds(.20f);

```

Εικόνα 3.2.2.7 - GunScript

```

151 {
152     yield return new WaitForSeconds(.20f);
153     crosshair.SetActive(false);
154     scopedOverlay.SetActive(true);
155     fieldOfView.fieldOfView = 15;
156 }
157
158 void Shoot() //Koitime pio oplo einai energo oste na orisoume tis max sfaires tou,to damage tou,to sound ktl,koitime episis an exoun teliwseis oi sfaires etsi wste na min
159 {
160
161     if (isAkActive == true && AkOutOfAmmo == false)
162     {
163         max_ammo = 27;
164         Damage = 20;
165         audioSource.PlayOneShot(ak_sound, 0.7f);
166         AKcurrentAmmo -- 1;
167         muzzle_Flash.Play();
168     }
169     else if (isPistolActive == true && PistolOutOfAmmo == false)
170     {
171         max_ammo = 7;
172         Damage = 10;
173         audioSource.PlayOneShot(pistol_sound, 0.7f);
174         PistolcurrentAmmo -- 1;
175         muzzle_FlashPistol.Play();
176     }
177     else if (isM4Active == true && M4OutOfAmmo == false)
178     {
179         max_ammo = 30;
180         Damage = 16;
181         audioSource.PlayOneShot(m4_sound, 1.4f);
182         M4currentAmmo -- 1;
183         muzzle_Flash.Play();
184     }
185     else if (isSniperActive == true && SniperOutOfAmmo==false)
186     {
187         max_ammo = 5;
188         Damage = 40;

```

Εικόνα 3.2.2.8 – GunScript

```

189     audioSource.PlayOneShot(sniper_sound, 0.7f);
190     SniperCurrentAmmo -- 1;
191     muzzle_FlashSniper.Play();
192 }
193 RaycastHit hit; // Xrisimopouloume Raycasts gia na paroume plirofories gia to ti object xtipame
194 if (Physics.Raycast(cam.transform.position, cam.transform.forward, out hit, range)) //Edw kanoume cast ena raycast pou ksekiniei apo tin camera me kateuthinsi mprostis tis ston x
195 {
196     if (hit.transform.tag == "Enemy" && hit.transform.tag != null) //an to object pou xtipisame einai to zombie tote perna tou damage.
197     {
198         zh.TakeDamage(Damage);
199     }
200     else if (hit.transform.tag == "Enemy2" && hit.transform.tag != null)
201     {
202         zh2.TakeDamage(Damage);
203     }
204     else if (hit.transform.tag == "Enemy3" && hit.transform.tag != null)
205     {
206         zh3.TakeDamage(Damage);
207     }
208 }
209 }
210
211 void Check()
212 {
213     if (ak_47.activeInHierarchy == true && ak_47.activeInHierarchy != false) //Edw pali koitime pio object einai active oste na tou perasoume to kalilo fire rate kai na emfanisoume tis ka
214     {
215         fireRate = 5;
216         isAkActive = true;
217         isPistolActive = false;
218         isM4Active = false;
219         isSniperActive = false;
220         ammo_txt.text = ("AK 47 Ammo: " + AKcurrentAmmo + "/" + AKReserveAmmo);
221     }
222     else if (pistol.activeInHierarchy == true && pistol.activeInHierarchy != false)
223     {
224         fireRate = 1;

```

Εικόνα 3.2.2.9 - GunScript

```

224     fireRate = 1;
225     isPistolActive = true;
226     isAKActive = false;
227     isM4Active = false;
228     isSniperActive = false;
229     ammo_txt.text = ("Pistol Ammo: " + PistolCurrentAmmo + "/" + PistolReserveAmmo);
230 }
231 else if (m4_carbine.activeInHierarchy == true && m4_carbine.activeInHierarchy != false)
232 {
233     fireRate = 5;
234     isM4Active = true;
235     isPistolActive = false;
236     isSniperActive = false;
237     isAKActive = false;
238     ammo_txt.text = ("M4 Ammo: " + M4CurrentAmmo + "/" + M4ReserveAmmo);
239 }
240 else if (sniper.activeInHierarchy == true && sniper.activeInHierarchy != false)
241 {
242     fireRate = 0.5f;
243     isSniperActive = true;
244     isAKActive = false;
245     isPistolActive = false;
246     isM4Active = false;
247     ammo_txt.text = ("Sniper Ammo: " + SniperCurrentAmmo + "/" + SniperReserveAmmo);
248 }
249 }
250 else if (machete.activeInHierarchy == true && machete.activeInHierarchy != false)
251 {
252     isSniperActive = false;
253     isAKActive = false;
254     isPistolActive = false;
255     isM4Active = false;
256     isMacheteActive = true;
257 }
258 }
259 }
260

```

Εικόνα 3.2.2.10 – GunScript

```

261 IEnumerator Reload() // Kathe fora pou oi sfaires tellwnoun automata to oplo kanei reloading,i coroutines eksipiretous periptwseis pou theloume na exoume kapio delay ,opws sto
262 {
263     IsReloading = true;
264     yield return new WaitForSeconds(reload_time);
265     if (PistolCurrentAmmo == 0)
266     {
267         if (PistolReserveAmmo > 7)
268         {
269             PistolCurrentAmmo = 7;
270             PistolReserveAmmo -= 7;
271         }
272         else
273         {
274             PistolCurrentAmmo = PistolReserveAmmo;
275             PistolReserveAmmo = 0;
276         }
277         if (PistolCurrentAmmo == 0 && PistolReserveAmmo == 0)
278         {
279             PistolOutOfAmmo = true;
280         }
281         else
282         {
283             PistolOutOfAmmo = false;
284         }
285     }
286     else if (AKCurrentAmmo == 0)
287     {
288         if (AKReserveAmmo > 27)
289         {
290             AKCurrentAmmo = 27;
291             AKReserveAmmo -= 27;
292         }
293         else
294         {
295             AKCurrentAmmo = AKReserveAmmo;
296         }
297     }
298 }
299

```

Εικόνα 3.2.2.11 - GunScript

```

306     AKCurrentAmmo = AKReserveAmmo;
307     AKReserveAmmo = 0;
308 }
309 if (AKCurrentAmmo == 0 && AKReserveAmmo == 0) //edw checkarw an exoun teliusei telw i sfaires gt o kwdikas mpainei se ena atermon vrogo kai den ekteilete o kwdikas kathws pr
310 {
311     AKOutOfAmmo = true;
312 }
313 else
314 {
315     AKOutOfAmmo = false;
316 }
317 }
318 }
319 else if (M4CurrentAmmo == 0)
320 {
321     if (M4ReserveAmmo > 30)
322     {
323         M4CurrentAmmo = 30;
324         M4ReserveAmmo -= 30;
325     }
326     else
327     {
328         M4CurrentAmmo = M4ReserveAmmo;
329         M4ReserveAmmo = 0;
330     }
331     if (M4CurrentAmmo == 0 && M4ReserveAmmo == 0)
332     {
333         M4OutOfAmmo = true;
334     }
335     else
336     {
337         M4OutOfAmmo = false;
338     }
339 }
340 }
341 else if (SniperCurrentAmmo == 0)
342 {

```

Εικόνα 3.2.2.12 – GunScript

```

331     else if (SniperCurrentAmmo == 0)
332     {
333         if (SniperReserveAmmo > 5)
334         {
335             SniperCurrentAmmo = 5;
336             SniperReserveAmmo -= 5;
337         }
338         else
339         {
340             SniperCurrentAmmo = SniperReserveAmmo;
341             SniperReserveAmmo = 0;
342         }
343         if (SniperCurrentAmmo == 0 && SniperReserveAmmo == 0)
344         {
345             SniperOutOfAmmo = true;
346         }
347         else
348         {
349             SniperOutOfAmmo = false;
350         }
351     }
352     IsReloading = false;
353 }
354 }
355 void AkRecoilOn()
356 {
357     anim.SetBool("IsFiring", true);
358 }
359 }
360 void AkRecoilOff()
361 {
362     anim.SetBool("IsFiring", false);
363 }
364 }
365 void M4RecoilOn()
366 {
367     anim.SetBool("IsFiring", true);
368 }

```

Εικόνα 3.2.2.13 - GunScript

```

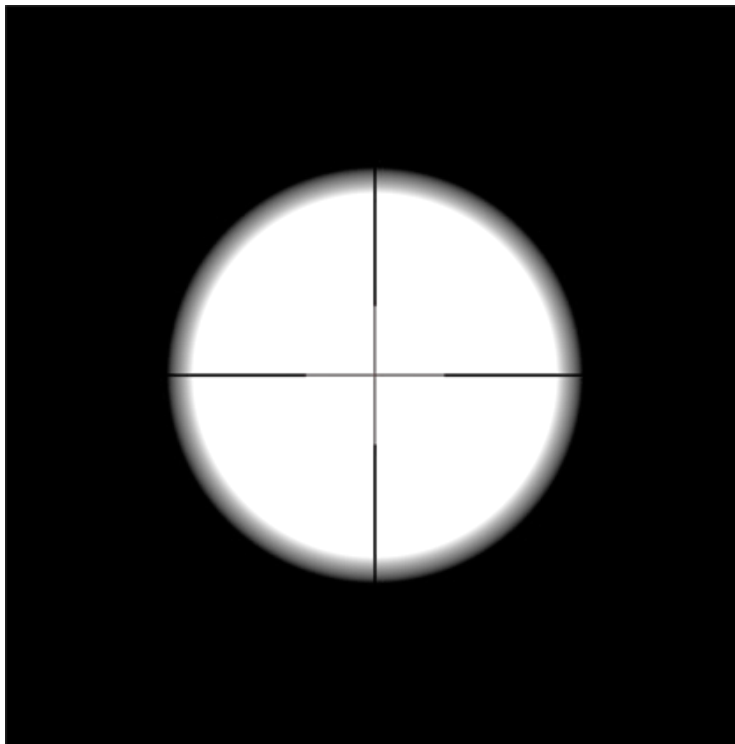
365 void M4RecoilOn()
366 {
367     anim.SetBool("IsFiring", true);
368 }
369 }
370 void M4RecoilOff()
371 {
372     anim.SetBool("IsFiring", false);
373 }
374 }
375 void PistolRecoilOn()
376 {
377     anim.SetBool("IsFiring", true);
378 }
379 }
380 void PistolRecoilOff()
381 {
382     anim.SetBool("IsFiring", false);
383 }
384 }
385 }

```

Εικόνα 3.2.2.14 – GunScript

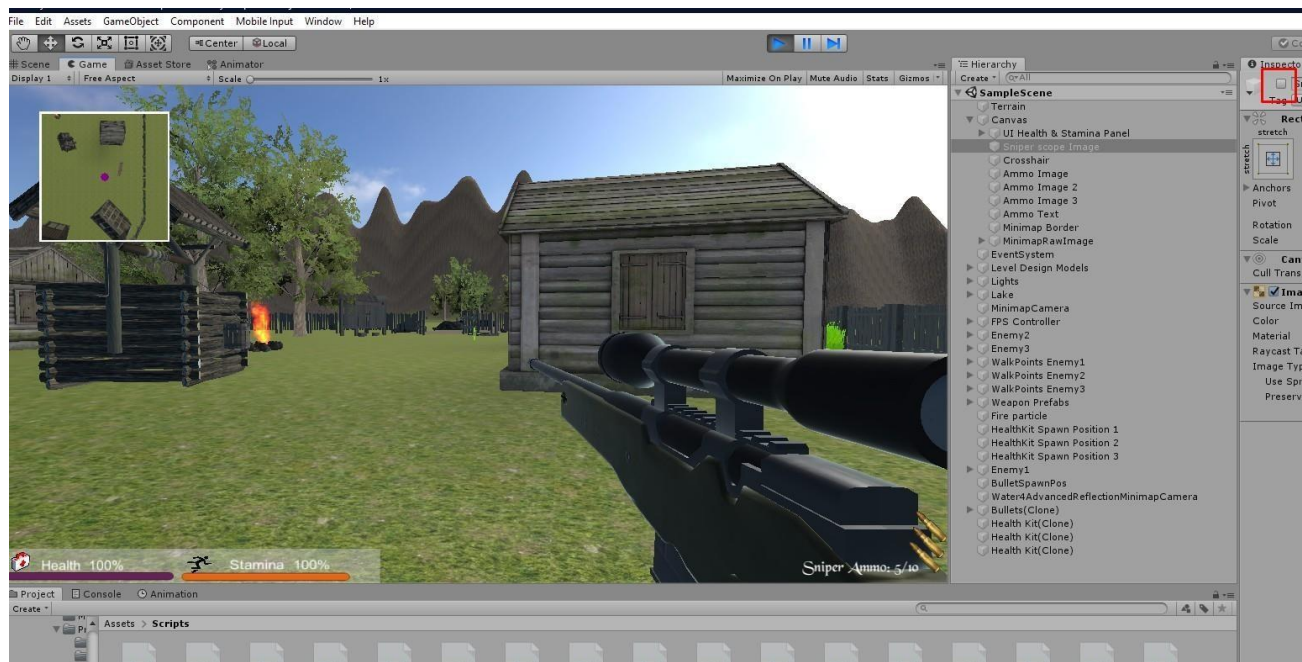
Το παραπάνω script είναι υπεύθυνο για τα εξής :

- Στην Update() μέθοδο, μέσω των Raycast δημιουργεί μία ακτίνα με αρχή το κέντρο κάμερας και κατεύθυνση προς τα εμπρός όταν ο παίκτης πατήσει το πλήκτρο fire1 (αριστερό κλικ), αν αυτή η ακτίνα χτυπήσει αντικείμενο του οποίου έχω προσθέσει ετικέτα "Enemy" τότε περνάει damage στον εχθρό.
- Αναλαμβάνει να καθορίσει το fireRate για το κάθε όπλο ξεχωριστά μέσω της συνάρτησης Check().
- Μέσω της συνάρτησης Shoot() καθορίζει το damage και τα πυρομαχικά για το κάθε όπλο ξεχωριστά και προσέχει ποιο όπλο χρησιμοποιεί ο παίκτης έτσι ώστε να ενεργοποιηθεί ο κατάλληλος ήχος.
- Η couritine OnScope και η συνάρτηση UnScoped είναι υπεύθυνες εξ ολοκλήρου για το Sniper όπλο που υπάρχει στο παιχνίδι και έτσι ο παίκτης όταν πατήσει το πλήκτρο Fire2 (δεξί κλικ) θα σκοπεύσει ή και το αντίθετο αναλόγως την κατάσταση στην οποία ήδη βρίσκεται η μεταβλητή bool ScopedIn. Ουσιαστικά όταν ο παίκτης θα σκοπεύσει ο κώδικας θα ενεργοποιήσει στην Hierarchy την εξής εικόνα σκοπεύτρου :

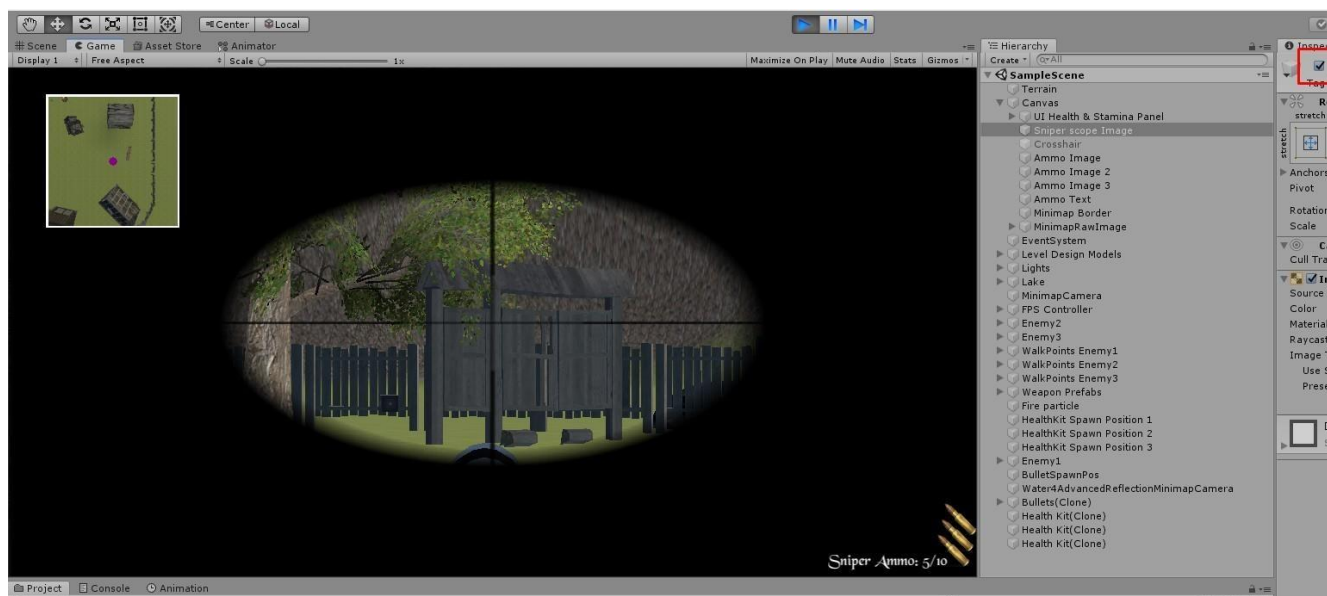


Εικόνα 3.2.2.15 – Sniper scope image

Δηλαδή όταν δεν σκοπεύουμε το στοιχείο ενεργοποίησης στον inspector της εικόνας sniper scope είναι ανενεργό:



Εικόνα 3.2.2.16 - Inactive



Εικόνα 3.2.2.17 - Active

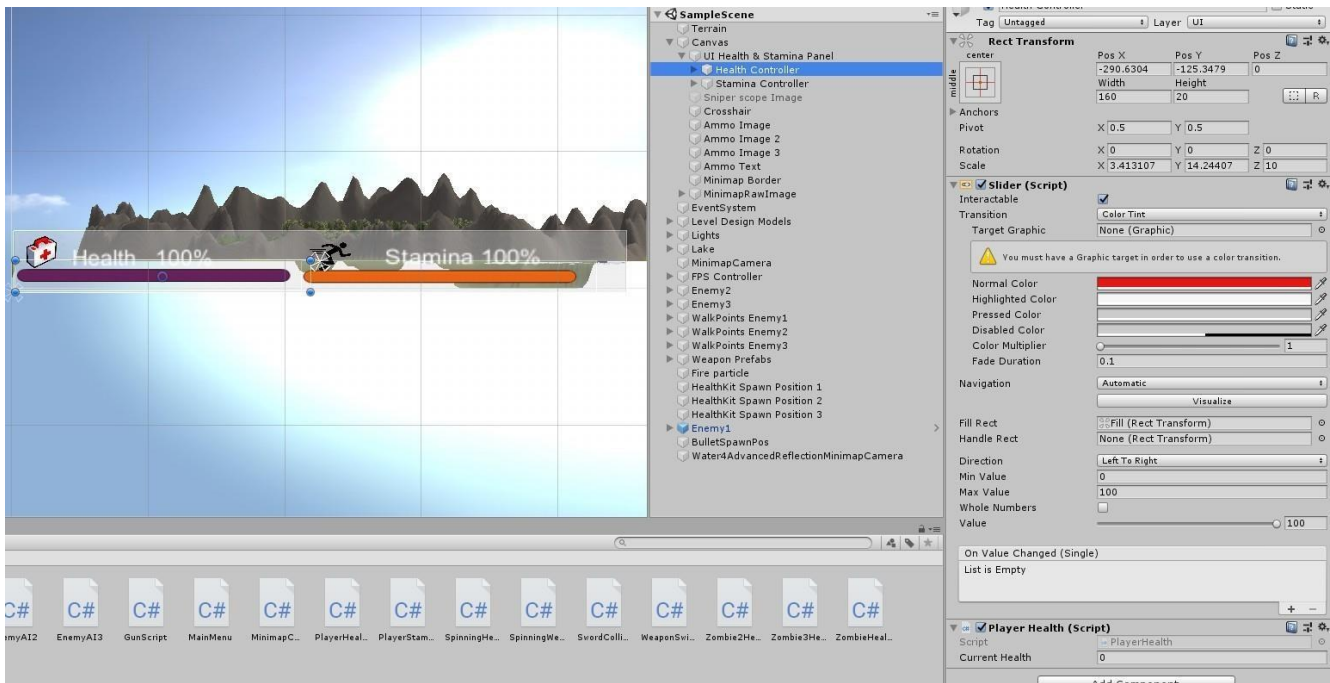
Όταν όμως σκοπεύσουμε, όπως βλέπουμε στην εικόνα 3.2.2.17, μέσω της εντολής `scopedoverlay.setactive(true)` στην γραμμή 154 του κώδικα η κατάλληλη εικόνα ενεργοποιείται:

Επιπρόσθετα έκανα μία αύξηση στο FieldOfView της κάμερας για να είναι πιο αληθοφανές το σκόπευτρο.

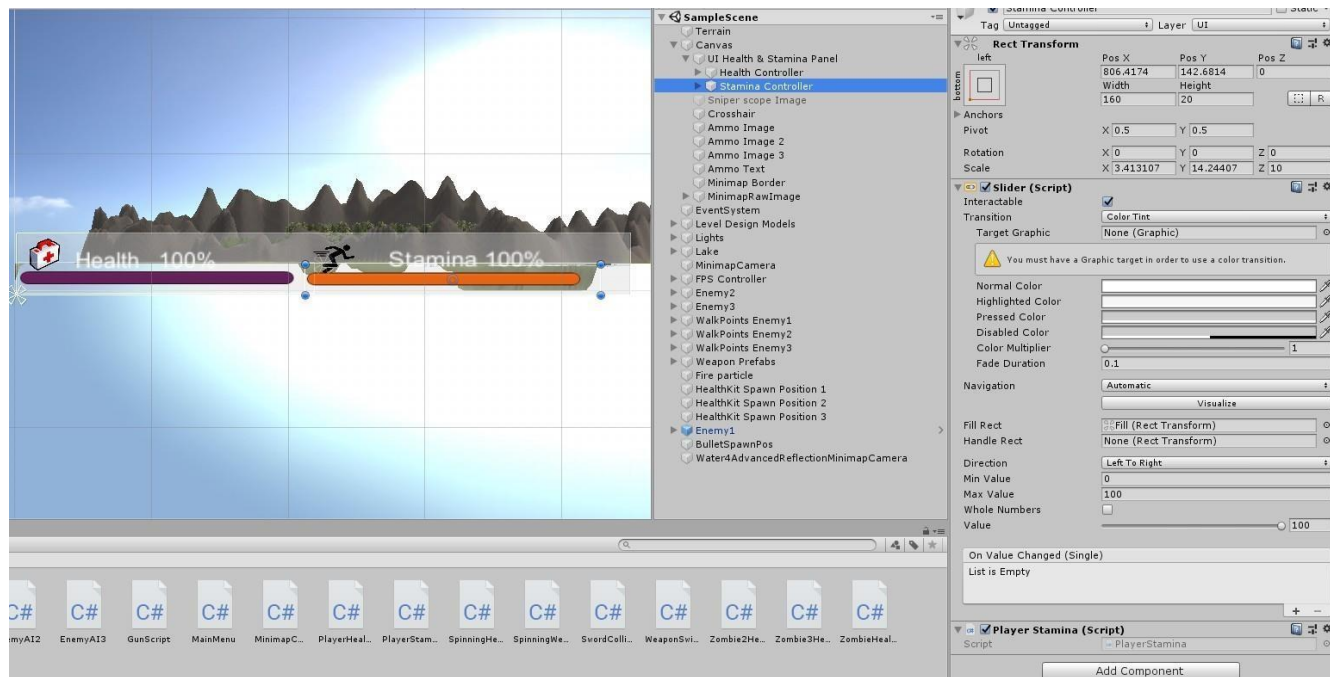
- Η couritine Reload() αναλαμβάνει να ξαναγεμίσει το όπλο με σφαίρες μετά από 1.5 δευτερόλεπτο όταν το όπλο έχει 0 σφαίρες στον “γεμιστήρα” του.
- Οι συναρτήσεις AkRecoilOn(), M4RecoilOn(), PistolRecoilOn() ουσιαστικά ενεργοποιούν τα animation των εκάστοτε όπλων όταν πυροβολούν και τα απενεργοποιούν με τις συναρτήσεις AkRecoilOff(), M4RecoilOff(), PistolRecoilOff().

PlayerHealth – PlayerStamina

Σε όλα τα παιχνίδια πρώτου προσώπου και όχι μόνο, έχουμε παρατηρήσει ότι τα UI στοιχεία παίζουν καθοριστικό ρόλο στην ψυχαγωγία και πολυπλοκότητα του παιχνιδιού. Για παράδειγμα ένα παιχνίδι στο οποίο ο παίκτης μόνο πυροβολεί και σκοτώνει εχθρούς θεωρείται βαρετό από ένα σημείο και μετά για αυτό το στοιχείο της επιβίωσης είναι σημαντικό να υπάρχει εξαρχής. Πρόσθεσα λοιπόν στο παιχνίδι δύο μπάρες ζωής και αντοχής του παίκτη. Στο Hierarchy view της Unity δημιούργησα δύο UI Sliders,άλλαξα το background χρώμα σε κόκκινο, το fill χρώμα σε πράσινο και κίτρινο αντίστοιχα για την ζωή και την αντοχή, και πρόσθεσα στο κάθε ένα από ένα script που διαχειρίζεται τη συμπεριφορά του κάθε slider ανάλογα με την κατάσταση στην οποία βρίσκεται ο παίκτης (π.χ. αν δέχεται χτύπημα ,αν τρέχει). Τον κάθε slider τον τοποθέτησα μέσα σε ένα κενό αντικείμενο (Health Controller, Stamina Controller) για καλύτερη και ταχύτερη διαχείριση δεδομένων, μία τεχνική που χρησιμοποίησα συχνά κατά τη διάρκεια της πτυχιακής εργασίας μου.



Εικόνα 3.2.3.1 – Health Controller



Εικόνα 3.2.3.2 – Stamina Controller

Player Health script:

```

6 public class PlayerHealth : MonoBehaviour
7 {
8     private int maxHealth = 100;
9     public int currentHealth;
10    private Slider sldr;
11    private Text txt;
12    private int damage_amount;
13
14
15
16    void Start()
17    {
18        sldr = GetComponent<Slider>();
19        txt = GameObject.Find("Health Text").GetComponent<Text>();
20        currentHealth = maxHealth;
21    }
22
23    // Update is called once per frame
24    void Update()
25    {
26        sldr.value = currentHealth;
27        txt.text = ("Health " + currentHealth + "%");
28
29        if (currentHealth > 100)
30        {
31            currentHealth = 100;
32        }
33    }
34
35    public int DamageAmount(int damage) //Edw pername damage ston Player,i parakatw sinartisi kaleite apo to script damage collider.
36    {
37        currentHealth -= damage;
38        return currentHealth;
39    }
40
41 }

```

Εικόνα 3.2.3.3 – Player Health script

Στον παραπάνω κώδικα ουσιαστικά μέσω της συνάρτησης Update() ενημερώνουμε σε κάθε frame την τιμή της ζωής του παίκτη. Η συνάρτηση DamageAmount() όπως παρατηρούμε είναι δηλωμένη ως public έτσι ώστε να είναι προσβάσιμη και από άλλα scripts που έχω δημιουργήσει. Έτσι όταν πληρείται μία κατάσταση, ένα άλλο script (το damageCollider συγκεκριμένα όπως θα δούμε παρακάτω) περνάει στην συνάρτηση το ποσό του χτυπήματος που δέχεται ο παίκτης μας και αφαιρεί από τη συνολική ζωή του παίκτη το ποσό του χτυπήματος. Τέλος έχω προσθέσει μία συνθήκη στην οποία ελέγχω αν η ζωή είναι πάνω από 100 και αν είναι την ορίζω 100 όσο δηλαδή η maximumHealth του παίκτη.

Player Stamina script:

```

19 void Start()
20 {
21     stamina_sldr = GetComponent<Slider>();
22     stamina_txt = GameObject.Find("Stamina Text").GetComponent<Text>();
23     current_stamina = max_stamina;
24     fps = GameObject.Find("FPSController").GetComponent<FirstPersonController>(); //Pernoyme to component tou script tou FPS controller oste na exoume prosvasi sto speed tou FPScontroller
25     cc = GameObject.Find("FPSController").GetComponent<CharacterController>(); //Kanoume to idio gia na mporoume na doume an o paixtis kinite.
26     run_speed = fps.m_RunSpeed;
27 }
28
29 // Update is called once per frame
30 void Update()
31 {
32     stamina_sldr.value = current_stamina;
33     stamina_txt.text = "Stamina " + current_stamina + "%";
34
35     if (Input.GetKey(KeyCode.LeftShift) && cc.velocity.magnitude>0) //Edw koitame an patame to left shift kai an o paixtis kinite oste na elatosoume to stamina tou paixti
36     {
37         current_stamina = current_stamina - (Time.deltaTime / stamina_FallRate) * 10;
38     }
39     else if (!Input.GetKey(KeyCode.LeftShift)) //An oxi tote ayksiseto
40     {
41         current_stamina = current_stamina + (Time.deltaTime / stamina_RegainRate) * 4;
42     }
43
44     if (current_stamina <= 0) //an to stamina einai katw apo 0 tote kane ti taxitita tou paixti isi me 0 tote kane ton Player na min mporei na treksei
45     {
46         current_stamina = 0;
47         fps.m_RunSpeed = fps.m_WalkSpeed;
48     }
49     else if (current_stamina >= 100)
50     {
51         current_stamina = max_stamina;
52     }
53
54     if (current_stamina > 0)
55     {
56         fps.m_RunSpeed = run_speed;
57     }
58 }
59
60
61
62
63
64

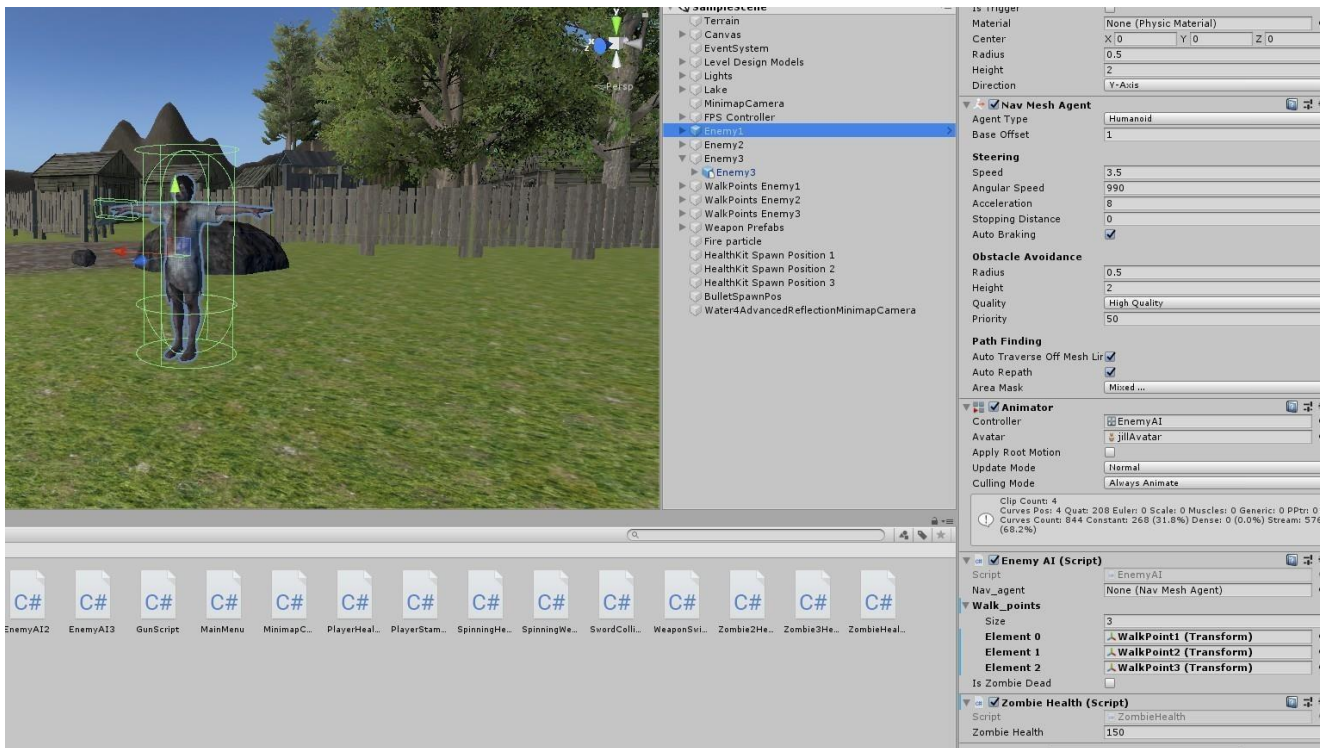
```

Εικόνα 3.2.3.4 – Player Stamina Script

Μέσω της Start() συνάρτησης και συγκεκριμένα των μεταβλητών *fps* και *cc* έχω αποκτήσει πρόσβαση στα script FPScontroller και CharacterController του αντικειμένου – κάμερας FPScontroller έτσι ώστε να ελέγχω αν ο παίκτης βρίσκεται σε κατάσταση τρεξίματος ή όχι. Αν ναι, τότε μειώνω το stamina γραμμικά δηλαδή με την ίδια τιμή συνεχόμενα και αν όχι, αυξάνω το stamina πάλι γραμμικά. Τέλος αν το stamina του παίκτη είναι ίσο με μηδέν τότε αποτρέπω τον παίκτη από το να τρέξει ορίζοντας την μεταβλητή *run_speed* ίση με το *walk_speed*.

ZombieHealth – EnemyAI (τεχνητή νοημοσύνη εχθρού)

Το παιχνίδι περιέχει τρεις εχθρούς (σχετικά μικρός αριθμός καθαρά λόγω βελτιστοποίησης).



Εικόνα 3.2.4.1 – Enemy1

Όπως βλέπουμε στην εικόνα **3.2.4.1** το μοντέλο έχει προσαρμοσμένο πάνω του collider, nav mesh Agent, animator και δύο script που διαχειρίζονται τη συμπεριφορά του bot. Παρακάτω θα αναλύσουμε τη λειτουργία των δύο αυτών script.

ZombieHealth script

```

1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class ZombieHealth : MonoBehaviour
6 {
7     public int zombieHealth = 150;
8     private Animator anim;
9     private EnemyAI ai;
10
11     void Start()
12     {
13         anim = GetComponent<Animator>();
14         ai = GetComponent<EnemyAI>();
15     }
16
17     void Update()
18     {
19         if (zombieHealth <= 0) //An to health tou enemy zombie einai katw apo 0 energopioume to katalilo animation kai eidopoume to enemy AI script oti einai nekro gia na diakopsoume tin sum
20         {
21             zombieHealth = 0;
22             ai.nav_agent.speed = 0f; //Zweis tyti tin entoli,to zombie byggare kai gia 1sec enw to death animation epaize,akoma akolothouse ton Player
23             ai.isZombieDead = true;
24             anim.SetBool("IsDead", true);
25         }
26
27     }
28
29     public void TakeDamage(int damageAmount) // Ede pername to damage sto zombie,i parakatw sinartisi kaleite apo to gun script scriptaki.
30     {
31         zombieHealth -= damageAmount;
32     }
33 }

```

Εικόνα 3.2.4.2 – ZombieHealth Script

Το παραπάνω script περιέχει τρεις μεθόδους, την Start(), την Update() η οποία καλείται κάθε frame και την TakeDamage(). Η μέθοδος TakeDamage() όπως παρατηρούμε έχει δηλωθεί ως public έτσι ώστε να μπορεί να καλεστεί από το GunScript. Όταν λοιπόν ο παίκτης πυροβολεί με κάποιο όπλο, δημιουργείται ένα Ray όπως εξηγήσαμε και παραπάνω. Αν αυτό το Ray χτυπήσει το bot, αμέσως το GunScript στέλνει την τιμή του χτυπήματος αναλόγως το όπλο που χρησιμοποιεί ο παίκτης, στην συνάρτηση TakeDamage(), η οποία με τη σειρά της θα μειώσει την τιμή της ζωής του bot κατά τη τιμή χτυπήματος. Η συνάρτηση Update() λοιπόν ελέγχει συνεχώς αν η τιμή της ζωής του bot είναι μικρότερη ή ίση με μηδέν, αν ναι τότε σταματάει την κίνηση του bot και ενεργοποιεί το animation θανάτου του. Τέλος έχω δημιουργήσει ένα αντικείμενο (ea) του script EnemyAI, έτσι ώστε να έχω πρόσβαση σε αυτό και όταν το bot μας έχει ζωή μηδέν να ενημερώνω το παραπάνω script μέσω της μεταβλητής isZombieDead έτσι ώστε να καταστρέφεται το αντικείμενο (το bot) μετά από ένα συγκεκριμένο χρονικό διάστημα από την σκηνή του παιχνιδιού.

Στη συνέχεια έχουμε την AI (τεχνητή νοημοσύνη) των bot του παιχνιδιού.

EnemyAI

```
7 {
8     public NavMeshAgent nav_agent;
9     private Transform player_position;
10    public Transform[] walk_points;
11    private int walkpoint_index=0;
12    private Vector3 target_direction;
13    private Animator anim;
14    private PlayerHealth ph;
15    public bool isZombieDead = false;
16    enum EnemyState {IsWalking,IsRunning,IsAttacking};
17    EnemyState enemyState;
18    private Vector3 startingPos;
19
20
21    void Start()
22    {
23        nav_agent = GameObject.FindGameObjectWithTag("Enemy").GetComponent<NavMeshAgent>();
24        player_position = GameObject.FindGameObjectWithTag("Player").GetComponent<Transform>();
25        anim = GetComponent<Animator>();
26        ph = GameObject.Find("Health Controller").GetComponent<PlayerHealth>();
27        enemyState=EnemyState.IsWalking;
28        startingPos = this.gameObject.transform.position;
29        walk_points[0] = GameObject.Find("WalkPoint1").GetComponent<Transform>();
30        walk_points[1] = GameObject.Find("WalkPoint2").GetComponent<Transform>();
31        walk_points[2] = GameObject.Find("WalkPoint3").GetComponent<Transform>();
32    }
33
34    // Update is called once per frame
35    void Update()
36    {
37
38        if (isZombieDead == true) //An to zombie einai nekro tote min ekteleseis tipota ksana kai katestrepe to zombie se 15 seconds
39        {
40            Destroy(gameObject, 3f);
41            return;
42        }
43
44        //target_direction = player_position.position - transform.position;
```

Εικόνα 3.2.4.3 - EnemyAI

```

44 //target_direction = player_position.position - transform.position;
45
46 if (Vector3.Distance(this.transform.position, player_position.position) > 4f) //An i apostasi tou player me to zombie einai panw apo 4m tote energopioise to walking animation kai pig
47 {
48     enemyState = EnemyState.IsWalking;
49     nav_agent.speed = 0.5f;
50
51     if (walkpoint_index < 3) //pigene pros to katalilo walk point
52     {
53         nav_agent.SetDestination(walk_points[walkpoint_index].position);
54
55         if (Vector3.Distance(transform.position, walk_points[walkpoint_index].position) < 1f) //an eftases sto walkpoint se apostasi katw apo 1m tote pigene sto epomeno.
56         {
57             walkpoint_index += 1;
58         }
59     }
60
61     else if (walkpoint_index >= 3) //edw koitame an eftase sto teleyteo walk point kai to ksanastelnoyme sto 1o.
62     {
63         walkpoint_index = 0;
64     }
65 }
66
67 if (Vector3.Distance(transform.position, player_position.position) <= 6f) //An i apostasi metaksi paixti kai zombie einai mikroteri apo 6m tote energopioise to running animation,ayksi
68 {
69     enemyState = EnemyState.IsRunning;
70     nav_agent.speed = 3f;
71     nav_agent.SetDestination(player_position.position);
72
73     if (Vector3.Distance(transform.position, player_position.position) <= 1.6f) //An i apostasi metaksi paixti kai zombie einai mikroteri apo 1.5m stamata kai energopioise to attack
74     {
75         enemyState = EnemyState.IsAttacking;
76         nav_agent.isStopped = true;
77     }
78     else if (Vector3.Distance(transform.position, player_position.position) > 1.6f) //An i apostasi metaksi paixti kai zombie einai megaliteri apo 1.5f tote energopioise to running anim
79     {
80         enemyState = EnemyState.IsRunning;

```

Εικόνα 3.2.4.4 – EnemyAI

```

79
80     enemyState = EnemyState.IsRunning;
81     nav_agent.isStopped = false;
82 }
83
84 }
85
86 if (Vector3.Angle(target_direction, transform.forward) < 100f && Vector3.Distance(transform.position, player_position.position) < 10f && Vector3.Distance(transform.position, player_po
87 {
88     nav_agent.speed = 3f;
89     enemyState = EnemyState.IsRunning;
90     nav_agent.SetDestination(player_position.position);
91 }
92
93 //Koitame se ti state einai wste na energopioisoume to katalilo animation.
94 if (enemyState == EnemyState.IsRunning)
95 {
96     anim.SetBool("IsAttacking", false);
97     anim.SetBool("IsRunning", true);
98 }
99 else if (enemyState == EnemyState.IsAttacking )
100 {
101     anim.SetBool("IsRunning", false);
102     anim.SetBool("IsAttacking", true);
103 }
104
105 else if (enemyState == EnemyState.IsWalking)
106 {
107     anim.SetBool("IsAttacking", false);
108     anim.SetBool("IsRunning", false);
109 }
110
111 }
112
113

```

Εικόνα 3.2.4.5 – EnemyAI

Ο παραπάνω κώδικας πραγματεύεται την τεχνητή νοημοσύνη, δηλαδή τη συμπεριφορά των bots στο παιχνίδι. Ειδικότερα:

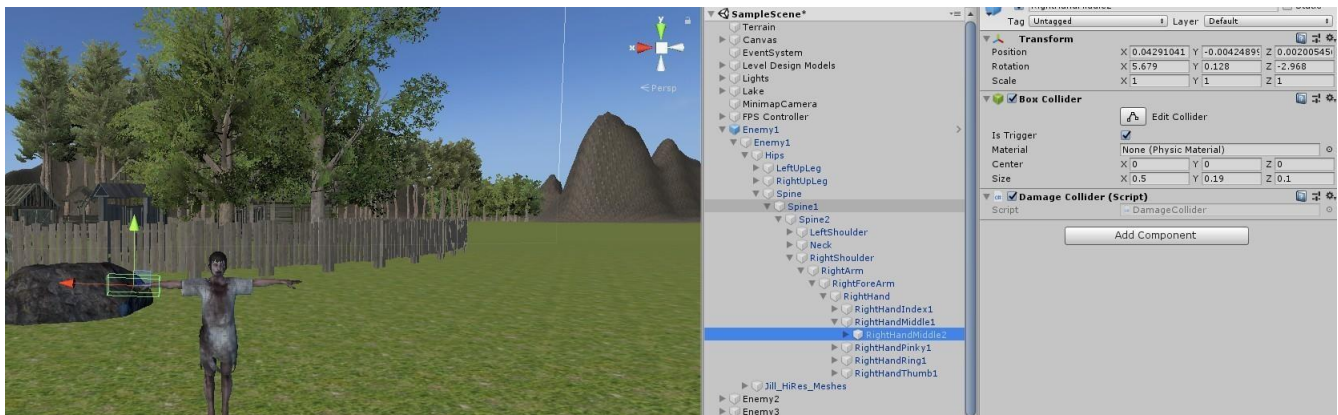
- Χρησιμοποιώντας enum έχω δημιουργήσει τις καταστάσεις (IsRunning, IsWalking, IsAttacking), όπου για την κάθε μία θα ενεργοποιείται το κατάλληλο animation.
- Μέσω της Update() συνάρτησης που τρέχει κάθε frame, αναζητείται η τιμή απόστασης μεταξύ του παίκτη μας από το εχθρικό bot. Αν αυτή η τιμή είναι μικρότερη του 6 τότε ορίζεται σαν κατεύθυνση του bot ο ίδιος ο παίκτης, ορίζεται το IsRunning ως state και αυξάνεται η ταχύτητα του bot.
- Αν το bot βρίσκεται σε IsRunning κατάσταση και η τιμή απόστασης player και bot είναι μικρότερη από 1,6 τότε σταματάει η κίνηση του bot και ορίζεται το IsAttacking ως state.
- Στην γραμμή **86** του κώδικα ελέγχεται το οπτικό πεδίου του bot μας, όπου αν η γωνία μεταξύ μίας νοητής γραμμής που ξεκινάει από το bot προς τα εμπρός (transform.forward) και της θέσης του παίκτη μας είναι μικρότερη από 100 μοίρες τότε θεωρούμε πως ο παίκτης είναι μέσα στο οπτικό πεδίο του εχθρικού bot άρα αυξάνεται η ταχύτητα του, αλλάζει κατεύθυνση του bot προς τον παίκτη και ορίζεται το IsRunning ως state.
- Όταν ο παίκτης δεν βρίσκεται σε κοντινή απόσταση ή στο οπτικό πεδίο του bot, το bot είναι προγραμματισμένο να ακολουθεί ένα μονοπάτι από walking points (**βλ. εικόνα 3.2.4.4, σελίδα 51-60**). Τα walking points αυτά είναι ουσιαστικά empty objects που δημιουργήσα και τοποθέτησα στον χάρτη, όπου παίρνω την θέση τους στον χώρο και μέσω ενός πίνακα τα ορίζω σαν κατεύθυνση των bot, όταν το bot φτάσει σε πολύ κοντινή απόσταση από ένα walking point τότε ορίζεται ως κατεύθυνση το επόμενο point δηλαδή το επόμενο στοιχείο του πίνακα. Όταν το bot φτάσει στο τελευταίο walking point τότε επαναλαμβάνεται η διαδικασία από την αρχή ορίζοντας ως επόμενο walking point το πρώτο στοιχείο του πίνακα. Το κάθε bot έχει δικό του διαφορετικό walking path από τα άλλα.



Εικόνα 3.2.4.6 – Walking Points

- Τέλος στην αρχή της Update() συνάρτησης ελέγχεται σε κάθε frame αν η μεταβλητή isZombieDead είναι αληθής, αν ναι τότε με την εντολή return σταματάει να εκτελείται οποιαδήποτε σειρά κώδικα κάτω από αυτή και καταστρέφεται το μοντέλο του bot στην σκηνή παιχνιδιού.

DamageCollider



Εικόνα 3.2.5.1 – Box Collider

```

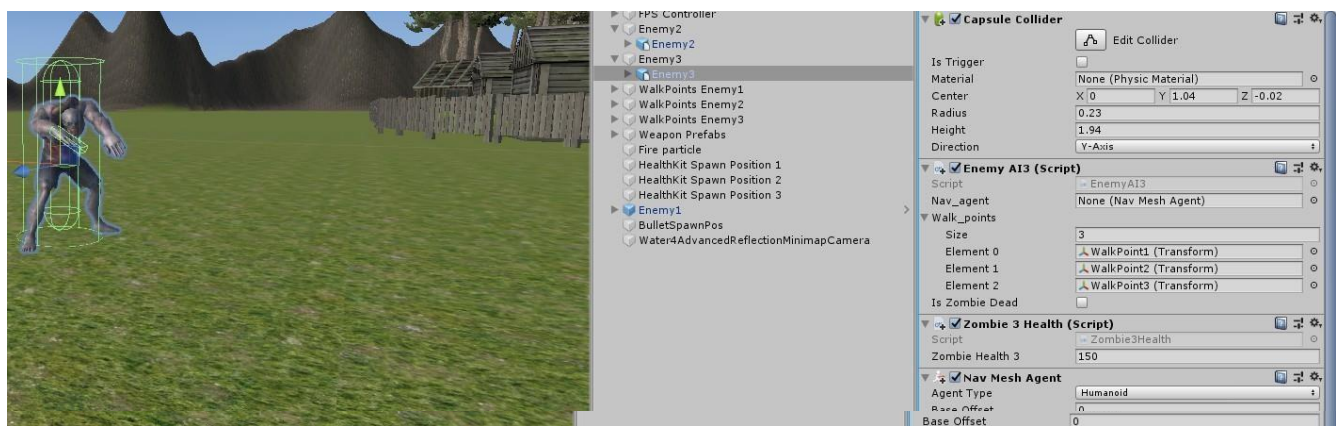
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class DamageCollider : MonoBehaviour
6  {
7      private PlayerHealth ph;
8
9      void Start()
10     {
11         ph = GameObject.Find("Health Controller").GetComponent<PlayerHealth>();
12     }
13
14     void OnTriggerEnter(Collider other) //edw exw ena trigger event opou otan akoumpaei ton player na tou pernaei damage
15     {
16         if (other.gameObject.name == "FPSController")
17         {
18             ph.DamageAmount(20);
19         }
20     }
21 }
22
23

```

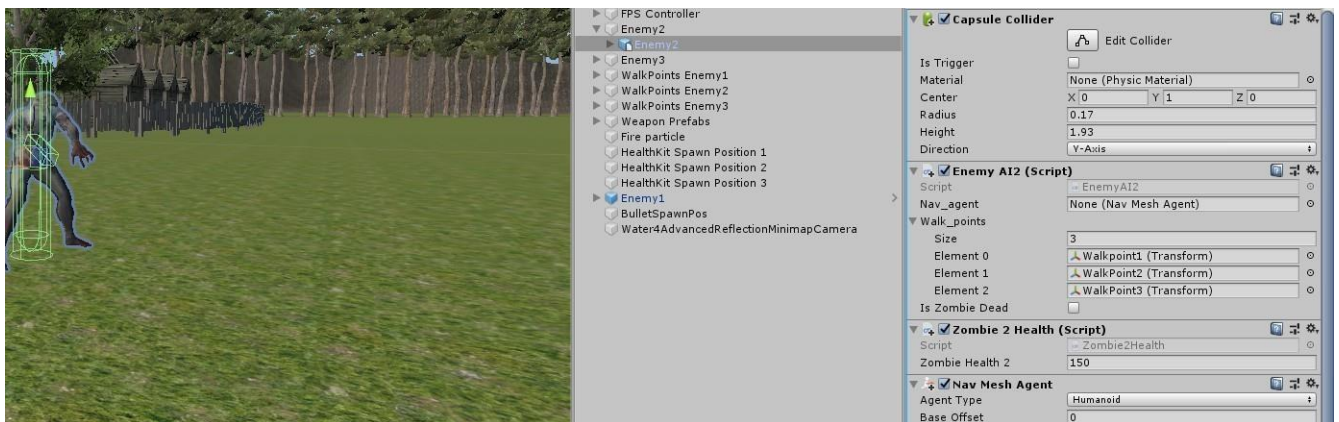
Εικόνα 3.2.5.2 – Damage Collider script

Όπως παρατηρούμε στην εικόνα **3.2.5.1** έχω προσαρμόσει στο χέρι του μοντέλου ένα box collider και στον inspector έχω επιλέξει την επιλογή IsTrigger. Τα παραπάνω μου δίνουν την επιλογή να ανιχνεύω οποιαδήποτε πλέον επαφή μεταξύ του box collider (του bot θεωρητικά) με τον παίκτη και να του καταλογίζω ζημιά. Αυτό ακριβώς συμβαίνει στη γραμμή κώδικα 15-21 (**βλ εικόνα 3.2.5.1**) όπου ελέγχεται αν η επαφή συμβαίνει με τον παίκτη και αν ναι τότε έχοντας πρόσβαση στο script PlayerHealth περνάμε ζημιά μεγέθους 20.

Εδώ αναφέρω ότι τα bot έχουν διαφορετικά script τεχνητής νοημοσύνης και ζωής με τον ίδιο ακριβώς κώδικα καθώς διαφορετικά μοντέλα δεν μπορούν να έχουν το ίδιο script με άλλα.

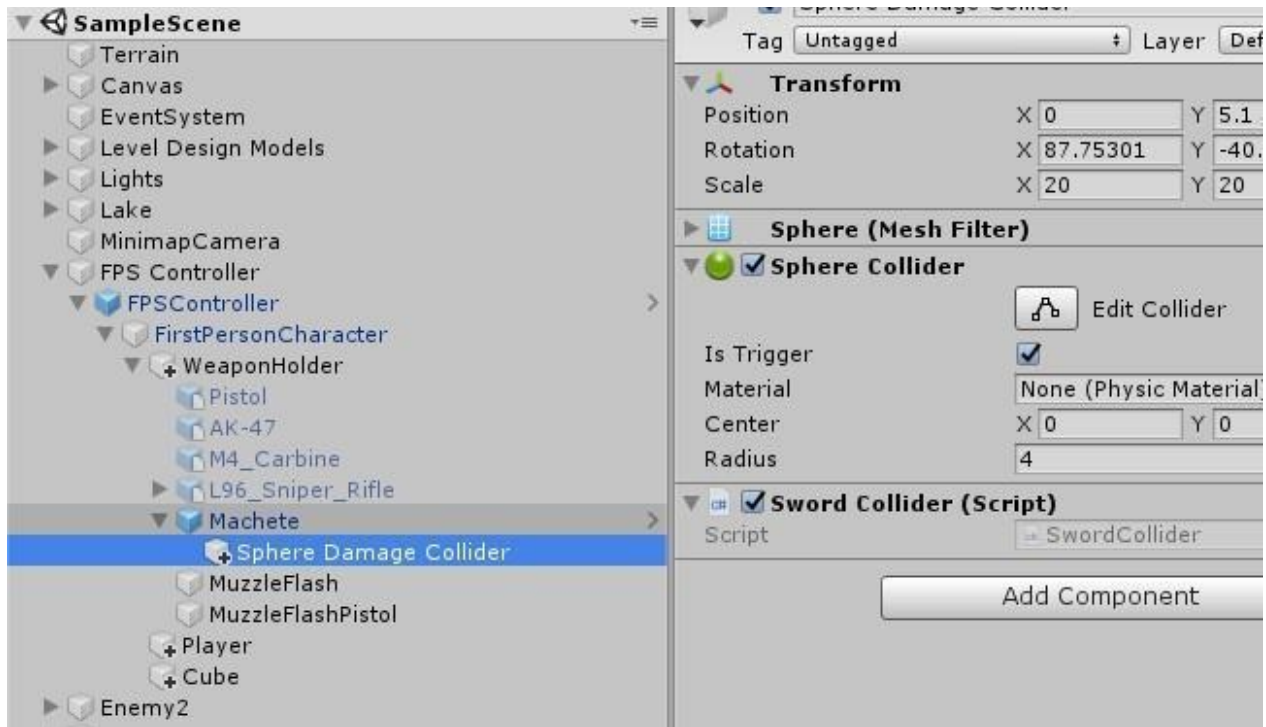


Εικόνα 3.2.1 – Enemy AI3 & Zombie 3 Health



Εικόνα 3.2.2 – Enemy AI2 & Zombie 2 Health

Sword Collider



Εικόνα 3.2.6.1 – Sword Collider

```

4
5 public class SwordCollider : MonoBehaviour
6 {
7     private ZombieHealth zh;
8     private Zombie2Health zh2;
9     private Zombie3Health zh3;
10
11     void Start()
12     {
13         if(GameObject.Find("Enemy1")!= null)
14         {
15             zh = GameObject.Find("Enemy1").GetComponent<ZombieHealth>();
16         }
17
18         zh2 = GameObject.Find("Enemy2").GetComponent<Zombie2Health>();
19         zh3 = GameObject.Find("Enemy3").GetComponent<Zombie3Health>();
20     }
21
22     void OnTriggerEnter(Collider other) //edw exw ena trigger event opou otan akoumpaei ton player na tou pernaei damage
23     {
24         if (other.gameObject.name == "Enemy1")
25         {
26             zh.TakeDamage(10);
27         }
28         else if(other.gameObject.name == "Enemy2")
29         {
30             zh2.TakeDamage(10);
31         }
32         else if(other.gameObject.name == "Enemy3")
33         {
34             zh3.TakeDamage(10);
35         }
36     }
37 }
38

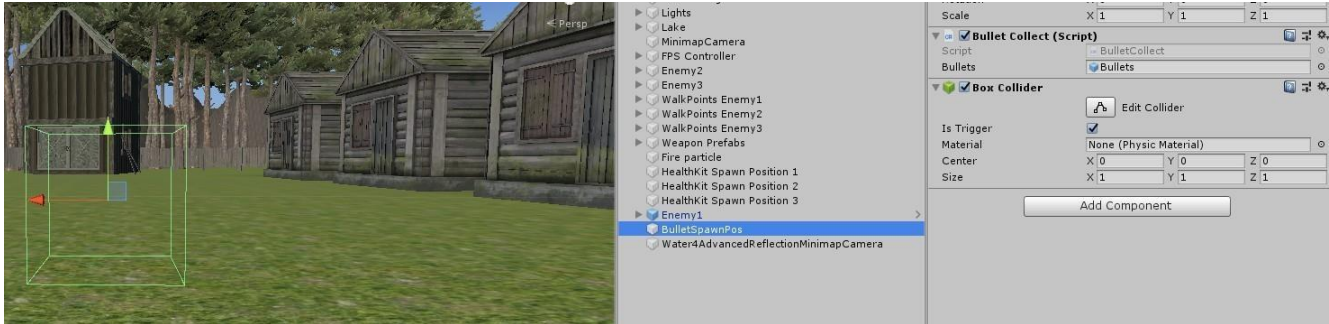
```

Εικόνα 3.2.6.2 – Sword Collider Script

Για την επαφή μεταξύ σπαθιού και bot πρόσθεσα στο prefab machete ένα σφαιρικό collider (βλ εικόνα 3.2.6.1) και ενεργοποίησα την επιλογή isTriggered. Επίσης επισύναψα και το

SwordCollider script, μέσω του οποίου όταν ο collider έρθει σε επαφή με ένα από τα bot, περνάει στον αντίστοιχο εχθρό ζημιά τιμής 10 **(βλ εικόνα 3.2.6.2 ,σειρά 23-35)**.

BulletCollector



Εικόνα 3.2.7.1 – BulletSpawnPos

Δημιούργησα ένα empty object σε ένα σημείο του χάρτη το οποίο θα είναι και το σημείο όπου θα γίνονται respawn τα πυρομαχικά των όπλων του παίκτη. Το κενό αυτό αντικείμενο έχει προσαρμοσμένα ένα script και ένα box collider για να ανιχνεύεται πότε ο παίκτης είναι σε απόσταση ώστε να μπορεί να σηκώσει τα πυρομαχικά από το έδαφος.

```
8 private GunScript gs;
9 public GameObject bullets;
10 private GameObject clone;
11 // Start is called before the first frame update
12 void Start()
13 {
14     gs = GameObject.Find("WeaponHolder").GetComponent<GunScript>();
15     clone = Instantiate(bullets, gameObject.transform.position, gameObject.transform.rotation);
16 }
17
18 void OnTriggerEnter(Collider other)
19 {
20     if (other.name == "FPSController" && Input.GetKeyDown(KeyCode.E))
21     {
22         if (gs.isActive == true)
23         {
24             gs.AKReserveAmmo = gs.AKReserveAmmo + 27;
25         }
26         else if (gs.isMAActive == true)
27         {
28             gs.MAReserveAmmo = gs.MAReserveAmmo + 30;
29         }
30         else if (gs.isPistolActive == true)
31         {
32             gs.PistolReserveAmmo = gs.PistolReserveAmmo + 7;
33         }
34         else if (gs.isSniperActive == true)
35         {
36             gs.SniperReserveAmmo = gs.SniperReserveAmmo + 5;
37         }
38         Destroy(clone);
39         StartCoroutine(RespawnBullets());
40     }
41 }
42
43
```

Εικόνα 3.2.7.2 – BulletCollector script

```

43
44 IEnumerator RespawnBullets()
45 {
46
47     yield return new WaitForSeconds (2f);
48     clone = Instantiate(bullets, this.transform.position, this.transform.rotation);
49
50 }
51
52

```

Εικόνα 3.2.7.3 – BulletCollector script

Το παραπάνω script λοιπόν ελέγχει αν ο παίκτης έχει εισέλθει στο πεδίο του collider. Αν ναι, τότε ο player πατώντας το κουμπί “Ε” προσθέτει στα δευτερεύοντα πυρομαχικά του τον αντίστοιχο αριθμό που χρειάζεται το όπλο το οποίο είναι ενεργό και τα πυρομαχικά που βρίσκονται στο έδαφος εξαφανίζονται από την σκηνή του παιχνιδιού. Μέσω της couritine RespawnBullets() (βλ. **εικόνα 3.2.7.3**), το prefab των πυρομαχικών επανεμφανίζεται μετά από 2 δευτερόλεπτα.

Spinning Health Kit



Εικόνα 3.2.8.1 – Health Kit spawning position

Έχω προσθέσει τρία health kit στο παιχνίδι έτσι ώστε όταν παίκτης βρίσκεται σε χαμηλά επίπεδα ζωής, να αυξήσει την τιμή της ζωής του κατά 20%. Δημιουργώντας λοιπόν τρία empty objects σε διαφορετικά σημεία στο χάρτη, έχω προσκολλήσει έναν collider για την ανίχνευση επαφής του kit με τον παίκτη και ενός script για την συμπεριφορά του.

```

5 public Class SpinningHealthKit : MonoBehaviour
6 {
7     private int speed = 100;
8     PlayerHealth ph;
9     public GameObject healthKit;
10    private bool isPickingUpHealthKit = false; //για να τσεκάρουμε αν σίκωε μία φορά idi to health kit allws an spamaroume to e,emfanizonte polla health kits kai bugarei to game.
11    GameObject clone;
12
13    void Start()
14    {
15        ph = GameObject.Find("Health Controller").GetComponent<PlayerHealth>();
16        clone = Instantiate(healthKit, gameObject.transform.position, gameObject.transform.rotation);
17    }
18
19    // Update is called once per frame
20    void Update()
21    {
22        if (clone != null) //Για να μην iparxei null reference error
23        {
24            clone.transform.Rotate(Vector3.up, speed * Time.deltaTime);
25        }
26    }
27
28
29    void OnTriggerEnter(Collider other) //oso o paixtis einai mesa sto box collider kai patisei to button E tou anebazei to health kata 20p,eksafanizei to health kit kai to kanei respawn meta
30    {
31        if (other.name == "FPSController" && Input.GetKeyDown(KeyCode.E) && isPickingUpHealthKit == false && ph.currentHealth < 100)
32        {
33            ph.currentHealth += 20;
34            isPickingUpHealthKit = true;
35            StartCoroutine(RespawnHealthKit());
36        }
37    }

```

Εικόνα 3.2.8.2 – Spinning health kit script

```

38 IEnumerator RespawnHealthKit()
39 {
40     Destroy(clone);
41     yield return new WaitForSeconds(3f);
42     isPickingUpHealthKit = false;
43     clone = Instantiate(healthKit, gameObject.transform.position, gameObject.transform.rotation);
44 }
45
46
47

```

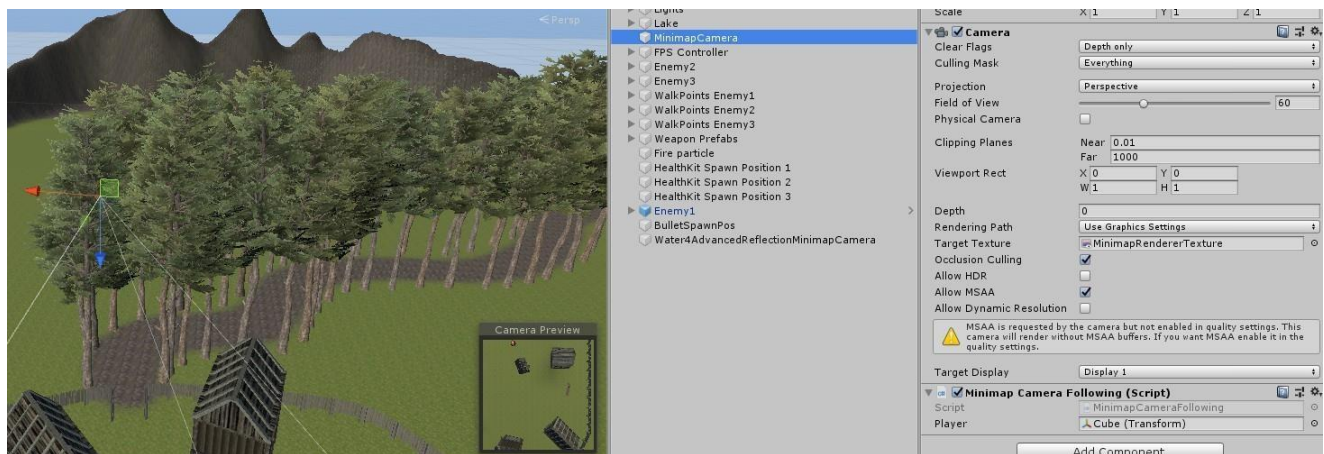
Εικόνα 3.2.8.3 – Spinning health kit script

Στη συνάρτηση Update() (**βλ εικόνα 3.2.8.2**) ελέγχουμε αν το kit υπάρχει στην σκηνή παιχνιδιού και αν ναι το περιστρέφουμε γύρω από τον εαυτό του. Έστερα αν ο παίκτης έχει την επαρκή απόσταση από το kit, αν πιέσει το πλήκτρο “E” και αν η ζωή του είναι μειωμένη, τότε αυξάνει τη ζωή του κατά 20% και καταστρέφεται το kit από τη σκηνή παιχνιδιού(**βλ εικόνα 3.2.8.2,σειρά 29- 36**). Τέλος μέσω της couroutine RespawnHealthKit(), επανεμφανίζεται το kit μετά από 3 δευτερόλεπτα αφού έχει καταστραφεί.

Προσθήκες στο παιχνίδι

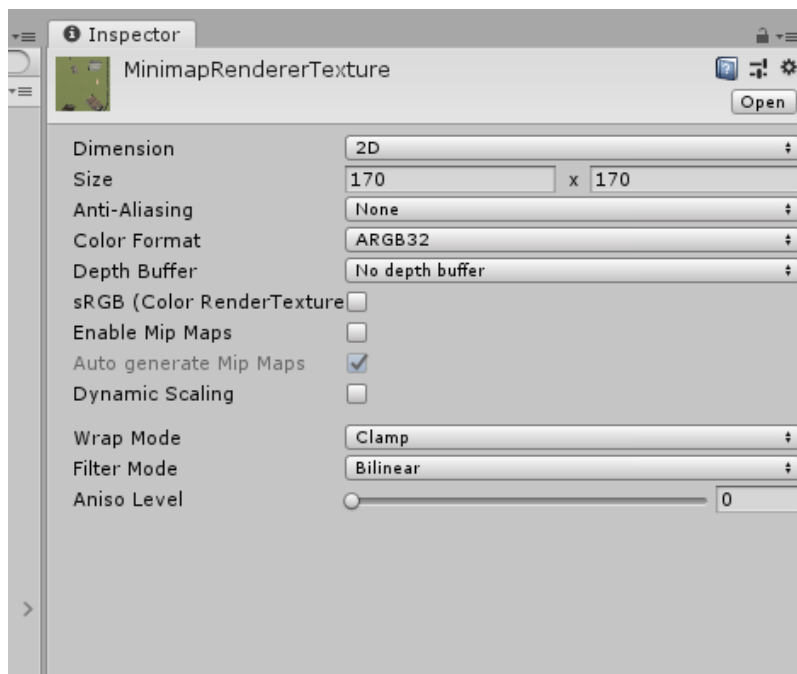
Minimap

Για τις ανάγκες του minimap, χρειάστηκε να προσθέσω στο παιχνίδι μια δεύτερη κάμερα και να την τοποθετήσω ακριβώς πάνω από τον παίκτη.

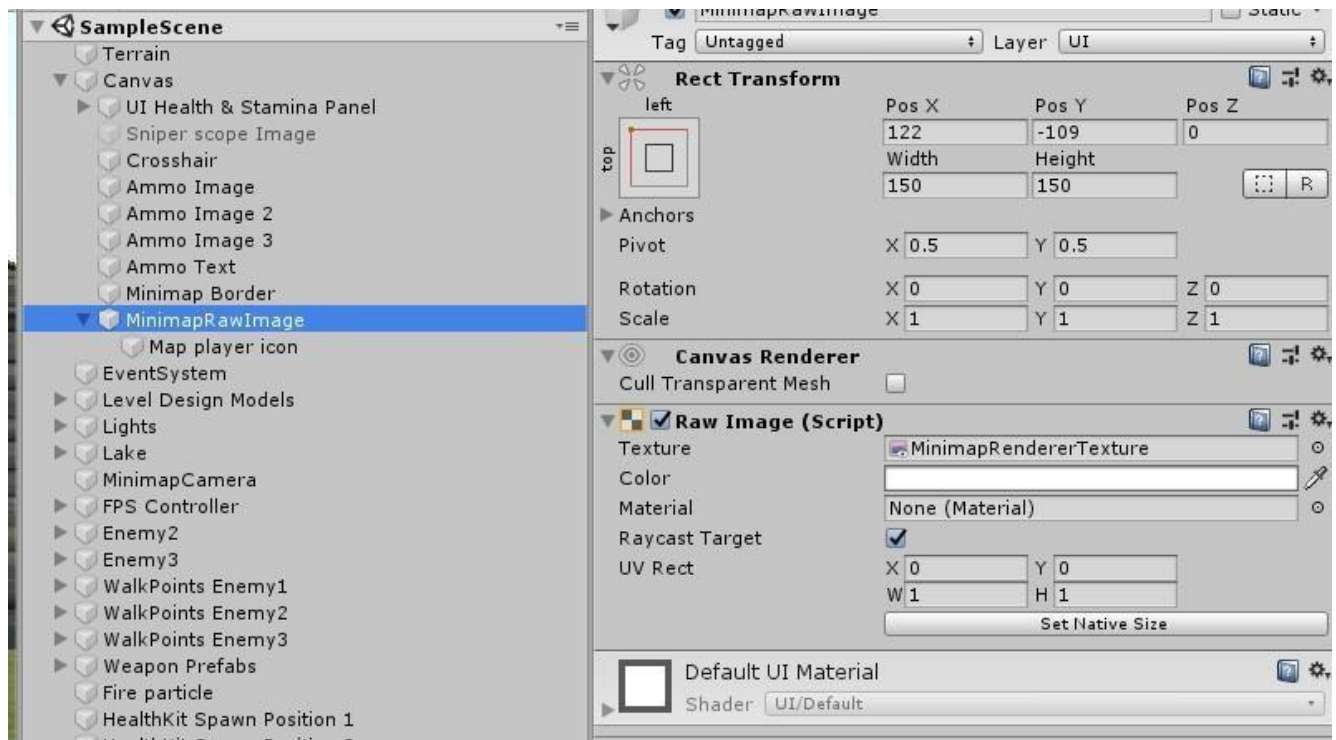


Εικόνα 3.3.1.1 – Minimap Camera

Στη συνέχεια δημιούργησα ένα rendering texture και ένα UI element raw image, έτσι ώστε το rendering texture να λαμβάνει την εικόνα από την minimap camera και να την περνάει στην raw image με αποτέλεσμα ο χρήστης να βλέπει τον χάρτη από κάθετη γωνία.



Εικόνα 3.3.1.2 – Rendering texture



Εικόνα 3.3.1.3 – Raw image

Ύστερα πρόσθεσα σαν παιδί του raw image ένα εικονίδιο στο κέντρο του ακριβώς το οποίο αντικατοπτρίζει τη θέση του παίκτη μας στο minimap. Τέλος, δημιούργησα ένα script το οποίο αναγκάζει την κάμερα του minimap να ακολουθεί τη θέση του παίκτη όταν αυτός κινείται.

```
Assembly CSharp
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class MinimapCameraFollowing : MonoBehaviour
6 {
7     public Transform player;
8
9     void Update()
10    {
11        Vector3 newPos = player.position;
12        newPos.y = transform.position.y;
13        this.gameObject.transform.position = newPos;
14    }
15 }
16
```

Εικόνα 3.3.1.4 – MinimapCameraFollowing script

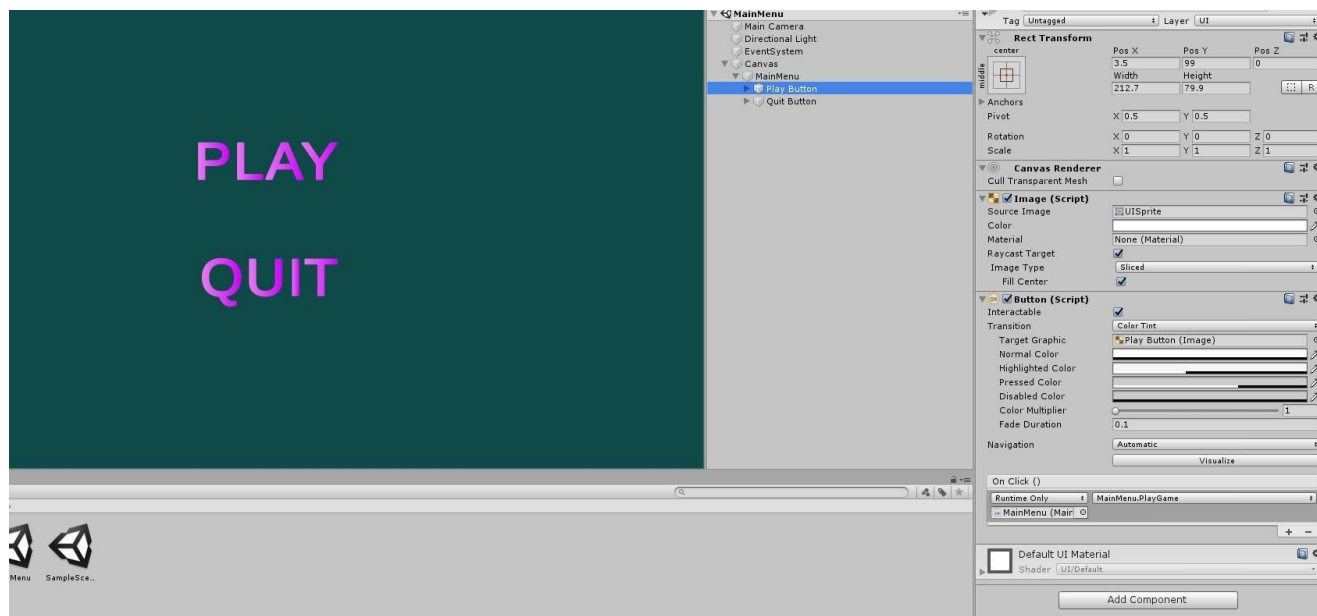
Main Menu

Θεώρησα ενδιαφέρον να προσθέσω ένα μενού που να δίνει την επιλογή στον παίκτη μέσω δύο πλήκτρων να παίξει ή να εγκαταλείψει το παιχνίδι. Δημιούργησα λοιπόν μία καινούρια σκηνή, στον Hierarchy view πρόσθεσα δύο UI buttons με όνομα **PLAY** & **QUIT** και επεξεργάστηκα διάφορα στοιχεία του όπως για παράδειγμα τα χρώματα των κειμένων τους. Το UI button δίνει στον developer την επιλογή να επιλέξει μια συνάρτηση από ένα script η οποία θα καλείται κάθε φορά που ο χρήστης θα επιλέγει το συγκεκριμένο πλήκτρο. Έγραψα λοιπόν δύο συναρτήσεις οι οποίες ξεκινούν και τερματίζουν το παιχνίδι αντίστοιχα για το κάθε πλήκτρο.

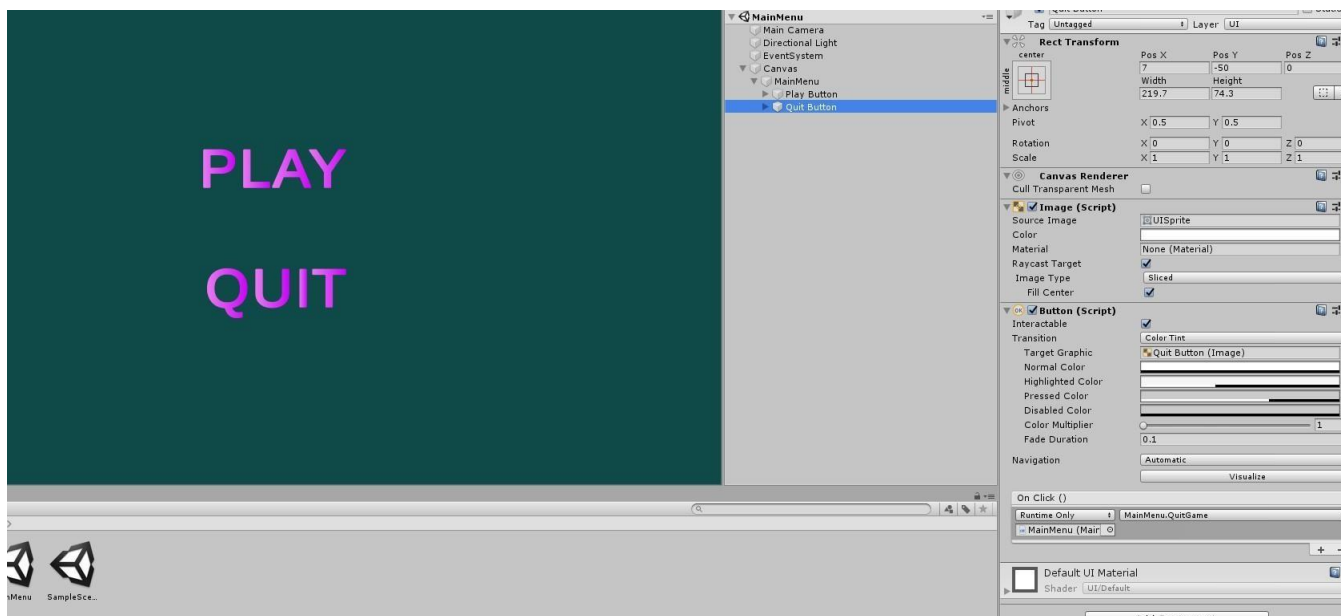
```
public class MainMenu : MonoBehaviour
{
    public void PlayGame()
    {
        SceneManager.LoadScene("SampleScene");
    }

    public void QuitGame()
    {
        Application.Quit();
    }
}
```

Εικόνα 3.3.2.1 – MainMenu script



Εικόνα 3.3.2.2 – Play Button



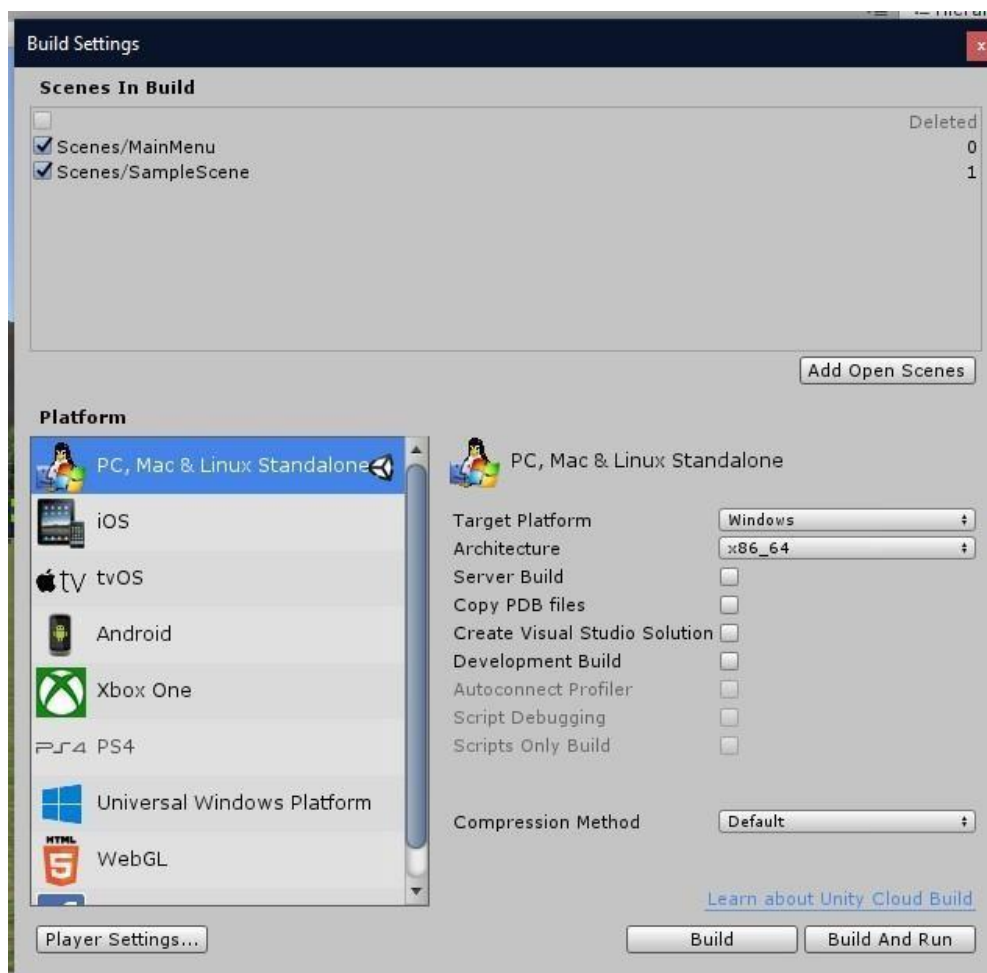
Εικόνα 3.3.2.3 – Quit Button

Κεφάλαιο 4 : Ολοκλήρωση και Walkthrough παιχνιδιού

Χτίσιμο παιχνιδιού

Όταν πλέον η ανάπτυξη του παιχνιδιού έχει φτάσει στο τέλος της, έχει έρθει η ώρα για τον προγραμματιστή να επιλέξει την μορφή της εφαρμογής που θα δημιουργηθεί. Μπορεί να επιλέξει λοιπόν την πλατφόρμα, όπως PS4 και XboxOne, ή το λειτουργικό σύστημα όπως Windows & android, και να ρυθμίσει τα στοιχεία που το αφορούν, όπως την αρχιτεκτονική στα Windows. Επιπρόσθετα δίνεται η επιλογή γραφικών και ανάλυσης του παιχνιδιού. Τέλος ο προγραμματιστής μπορεί να αλλάξει και το εικονίδιο του παιχνιδιού με κάποιο της αρεσκείας του.

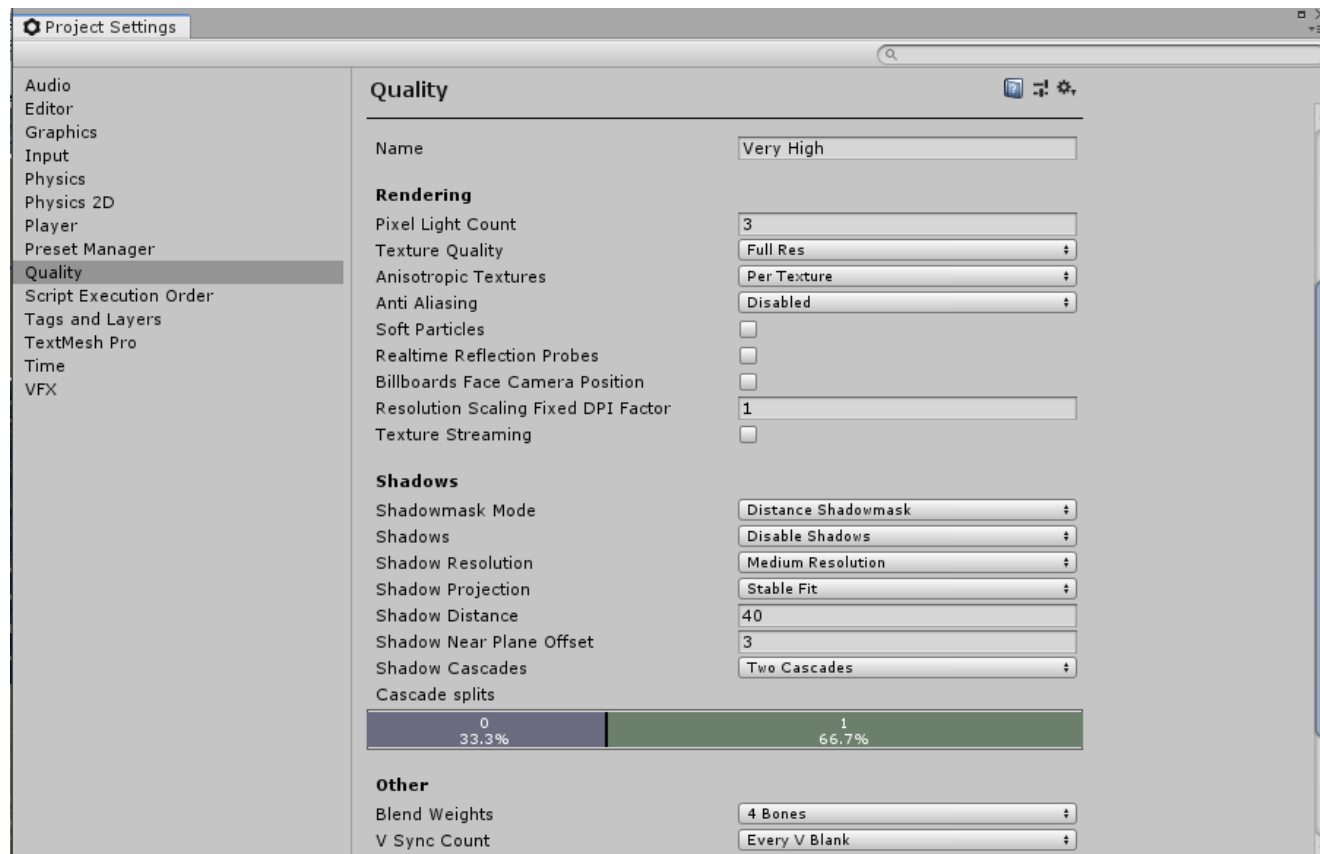
Ένα παιχνίδι μπορεί να έχει πολλές διαφορετικές σκηνές και ο προγραμματιστής οφείλει να τις βάλει σε μια σειρά ώστε το παιχνίδι να ξεκινήσει από το σωστό level.



Εικόνα 4.1.1 – Build Settings view

Επίσης ο προγραμματιστής πριν πραγματοποιήσει το χτίσιμο του παιχνιδιού έχει τη δυνατότητα να αλλάξει τις ρυθμίσεις των γραφικών για να καταστήσει το παιχνίδι ίσως πιο ελαφρύ και ομαλό στη περίπτωση που χρησιμοποιεί βαριά Visual Effects για παράδειγμα, μέσω των project

settings. Ωστόσο, έχει ακόμη πολλές επιλογές σε σχέση με τις ρυθμίσεις του παιχνιδιού τις οποίες δεν χρειάστηκε να εξερευνήσω λόγω ότι το παιχνίδι που δημιούργησα είναι πολύ ελαφρύ.



Εικόνα 4.1.2 – Project settings view

Άδειες και δικαιώματα

Είναι απολύτως νόμιμο να διανέμετε ή να πουλάτε παιχνίδια που γίνονται με την δωρεάν έκδοση της Unity. Το μόνο που δεν μπορείτε να κάνετε είναι να χρησιμοποιείται μία δωρεάν δοκιμαστική έκδοση 30 ημερών της Pro (πληρώνεις) έκδοσης της Unity για να κάνετε ένα παιχνίδι και στη συνέχεια να πουλήσετε αυτό το παιχνίδι χωρίς να αγοράσετε την Unity Pro.

Δηλαδή, μπορείτε να δημιουργήσετε και να πουλήσετε ένα παιχνίδι με την δωρεάν έκδοση της Unity χωρίς να πληρώνετε δικαιώματα ή μερίδιο εσόδων.

Πάνω από το 25% των προγραμματιστών κερδίζουν πάνω από 5.000 δολάρια το μήνα στην πλατφόρμα iOS και το Android βλέπει το 18% της δουλειάς τους επίσης καλό. Τα παιχνίδια χτύπησαν την αγορά σήμερα και είναι τα πιο κερδοφόρα. Οι περισσότερες εφαρμογές παιχνιδιών γίνονται με ελεύθερο χρόνο. Έτσι, το κόστος δεν είναι υπερβολικά υψηλό για την ανάπτυξη της εφαρμογής για κινητά με όρους χρημάτων.

Για έως και πέντε έδρες, μπορείτε να εγγραφείτε στο Unity Pro στην ειδική τιμή των \$ 75 ανά μήνα για περιορισμένη μεταβατική περίοδο, μετά την οποία η τιμή θα επανέλθει στην κανονική τιμή συνδρομής των \$ 125 ανά μήνα:

Unity	Personal	Plus	Pro	Enterprise
Much more than a game engine. Unity is a platform offering everything you need to create successful games and engaging experiences.	All the features for beginners & hobbyists to get started. Learn more	For serious creators looking to bring their vision to life. Learn more	For professionals looking to profit from advanced customization and complete flexibility. Learn more	A tailored solution to suit your organization's creative goals. Learn more
Free No credit card required	\$35 per seat/month	\$125 per seat/month	Contact us	
Download now	Select plan	Select plan	Contact us	

Εικόνα 4.2.1 – Licences

Walkthroug

Παρακάτω παραθέτω screenshots του παιχνιδιού κατά τη διάρκεια του Gameplay του.



Εικόνα 4.3.1 – Main Menu



Εικόνα 4.3.2 – Village



Εικόνα 4.3.3 – Zombie Enemy 1



Εικόνα 4.3.4 – Zombie Enemy 2



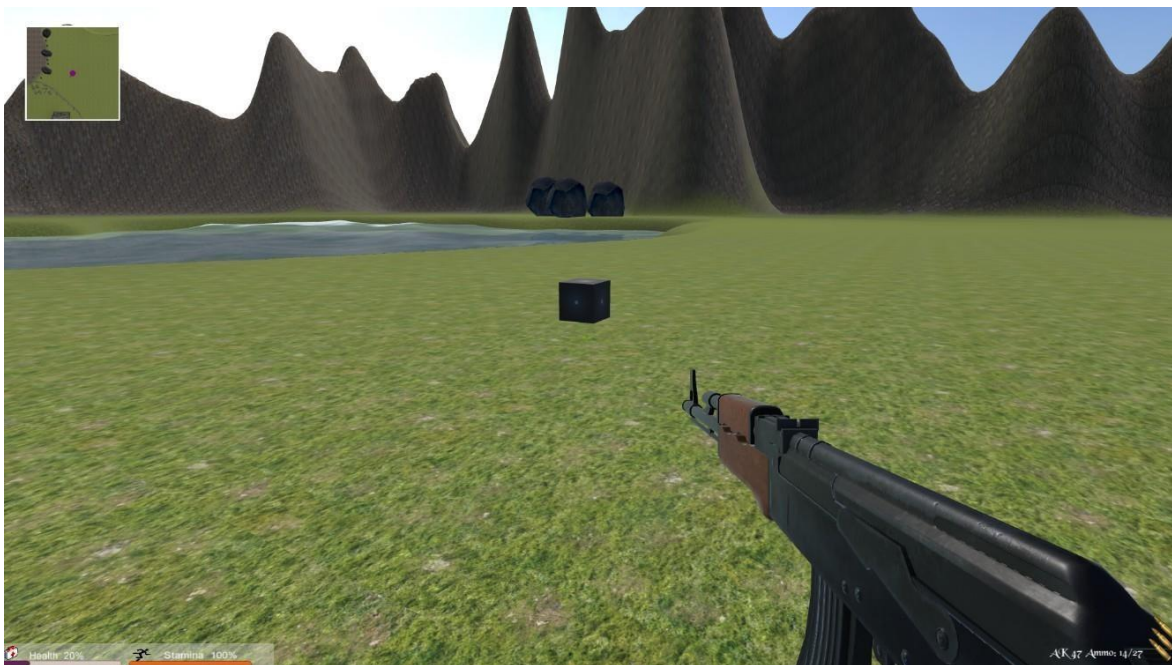
Εικόνα 4.3.5 – Zombie Enemy 1



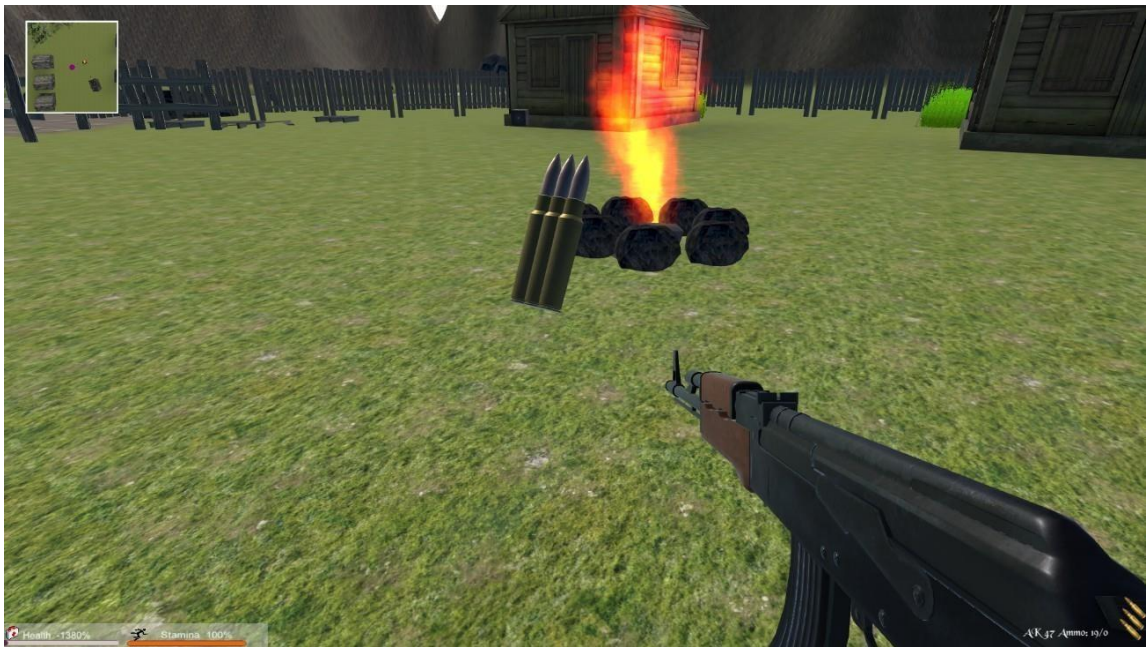
Εικόνα 4.3.6 – Zombie Enemy 2



Εικόνα 4.3.7 – Zombie Enemy 2



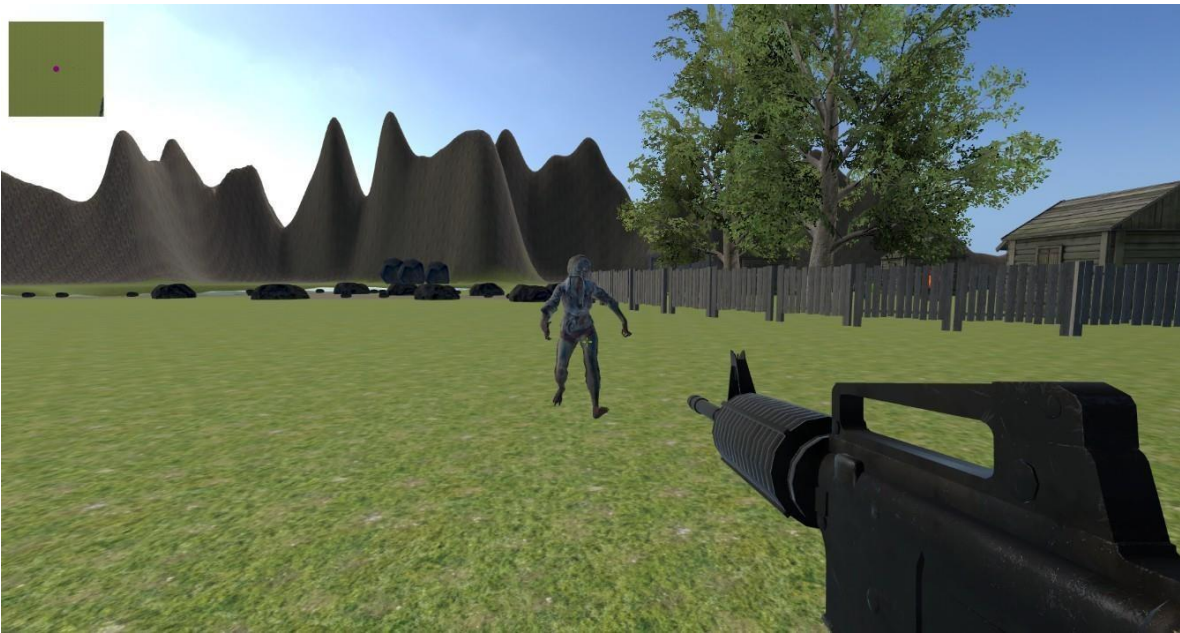
Εικόνα 4.3.8 – Health Kit



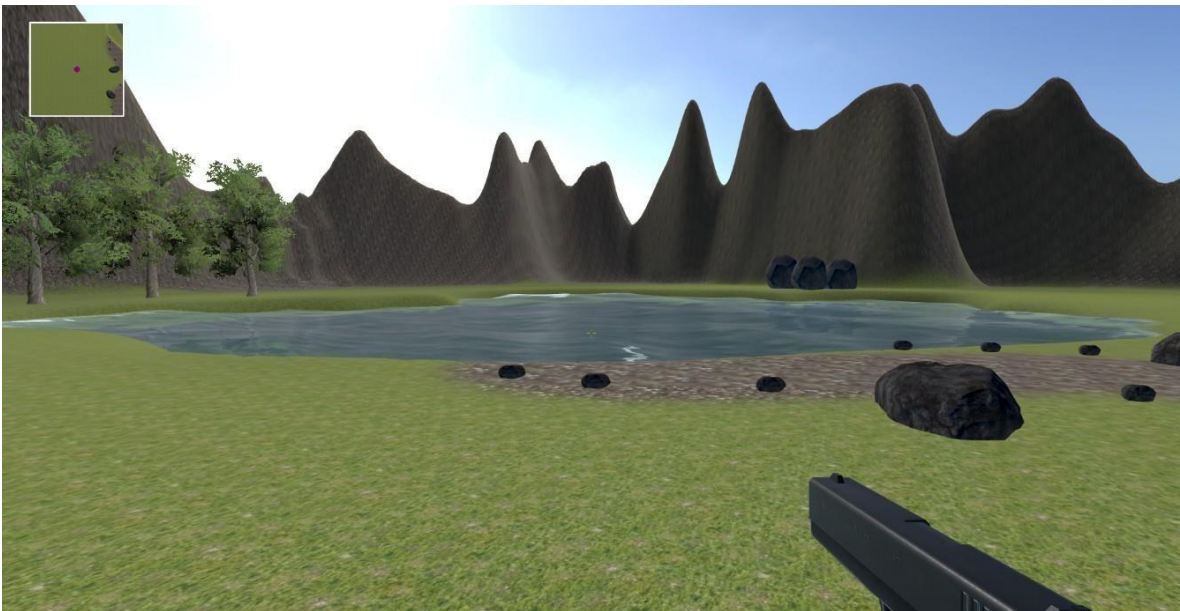
Εικόνα 4.3.9 – Ammo supplies



Εικόνα 4.3.10 – Enemy 3



Εικόνα 4.3.11 – Zombie Enemy 3



Εικόνα 4.3.12 – Lake and Mountains



Εικόνα 4.3.13 –Path to the Woods

Κεφάλαιο 5 : Συμπεράσματα και βιβλιογραφία

Συμπεράσματα

Η δημιουργία του παραπάνω βιντεοπαιχνιδιού αποτελεί μια ευχάριστη και ενδιαφέρον εμπειρία της ζωής μου και είμαι σίγουρος πως στο μέλλον θα ασχοληθώ και με περαιτέρω τομείς της βιομηχανίας των παιχνιδιών όπως για παράδειγμα το 3D modeling, έτσι ώστε να είμαι σε θέση να αναπτύξω και μόνος μου ολοκληρωμένα παιχνίδια. Ένα από τα πολλά πράγματα που έμαθα κατά τη διάρκεια αυτής της πτυχιακής, είναι ότι ένα παιχνίδι δεν είναι απλό να υλοποιηθεί καθώς χρειάζεται πολύ δουλειά κατά την ανάπτυξη και ακόμη περισσότερη για την διόρθωση συνεχόμενων bug καθώς από ότι κατάλαβα και ο ίδιος, τα bugs ποτέ δεν σταματούν να εμφανίζονται. Γενικότερα, η δημιουργία

ενός παιχνιδιού είναι μία αρκετά περίπλοκη διαδικασία που απαιτεί γνώσεις σε πολλούς τομείς άμα πρόκειται να ολοκληρωθεί πλήρως από ένα άτομο. Υπάρχουν όμως πολλά εργαλεία που καθιστούν αυτό το εγχείρημα πραγματοποιήσιμο.

Η Unity είναι μία πάρα πολύ καλή μηχανή δημιουργίας παιχνιδιών με πολλά εργαλεία, αλλά χρειάζονται ώρες εξάσκησης και αναζήτησης στο internet για την σωστή εξοικείωση του χρήστη.

Βιβλιογραφία

Παρακάτω παραθέτω τα site τα οποία επισκέφτηκα κατά τη διάρκεια της ανάπτυξης της πτυχιακής για τον χρήση ορισμένων μοντέλων εντός του παιχνιδιού αλλά και σελίδες που με βοήθησαν να υλοποιήσω την αναφορά στο κομμάτι της ιστορίας της Unity, των βιντεοπαιχνιδιών κτλ.

[1] <http://assetstore.unity.com/>

[2] <http://devassets.com/>

[3] https://en.wikipedia.org/wiki/Video_game_development

[4] <http://brackeys.com/>