ΠΕΡΙΗΓΗΣΗ ΣΕ ΕΙΚΟΝΙΚΟ ΠΕΡΙΒΑΛΛΟΝ ΜΕ ΧΡΗΣΗ ΜΑΣΚΑΣ ΕΙΚΟΝΙΚΗΣ ΠΡΑΓΜΑΤΙΚΟΤΗΤΑΣ

Τ.Ε.Ι. ΠΕΛΟΠΟΝΝΗΣΟΥ

Σχολή Τεχνολογικών Εφαρμογών(έδρα: Σπάρτη)

Τμήμα Μηχανικών Πληροφορικής Τ.Ε.

ΣΥΝΤΑΚΤΗΣ:

ΔΗΜΑΣ ΓΕΩΡΓΙΟΣ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ:

ΔΕΛΗΓΙΑΝΝΙΔΗΣ ΣΤΑΥΡΟΣ

ΠΕΡΙΛΗΨΗ

Η εργασία αυτή έχει σαν σκοπό την δημιουργία ενός εικονικού περιβάλλοντος για κινητά τηλέφωνα με χρήση μάσκας εικονικής πραγματικότητας, στο οποίο υπάρχει βύθιση (Immersion, Απόσπαση της προσοχής από το πραγματικό περιβάλλον) αλλά και διαδραστικότητα (αλληλεπίδραση μεταξύ χρήστη και περιβάλλοντος). Η ιδέα είναι η δημιουργία ενός εχθρικού περιβάλλοντος στο οποίο ο χρήστης προσπαθεί να επιβιώσει. Παρακάτω γίνεται καταγραφή του τρόπου δημιουργίας του περιβάλλοντος και των αντικειμένων που βρίσκονται μέσα σε αυτό, με χρήση της παιχνιδομηχανής Unity.Επίσης γίνεται καταγραφή του τρόπου με τον οποίο ο χρήστης και οι αντίπαλοι κινούνται μέσα στο περιβάλλον αλλά και τις λειτουργίες τους. Τέλος, γίνεται καταγραφή του κώδικα πίσω από κάθε αντικείμενο εξηγώντας τις λειτουργίες του και τον λόγο για τον οποίο δημιουργήθηκε.

ΔΗΛΩΣΗ ΜΗ ΛΟΓΟΚΛΟΠΗΣ ΚΑΙ ΑΝΑΛΗΨΗΣ ΠΡΟΣΩΠΙΚΗΣ ΕΥΘΥΝΗΣ

"Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ενυπογράφως ότι είμαι αποκλειστικός συγγραφέας της παρούσας Πτυχιακής Εργασίας, για την ολοκλήρωση της οποίας κάθε βοήθεια είναι πλήρως αναγνωρισμένη και αναφέρεται λεπτομερώς στην εργασία αυτή. Έχω αναφέρει πλήρως και με σαφείς αναφορές, όλες τις πηγές χρήσης δεδομένων, απόψεων, θέσεων και προτάσεων, ιδεών και λεκτικών αναφορών, είτε κατά κυριολεξία είτε βάση επιστημονικής παράφρασης. Αναλαμβάνω την προσωπική και ατομική ευθύνη ότι σε περίπτωση αποτυχίας στην υλοποίηση των ανωτέρω δηλωθέντων στοιχείων, είμαι υπόλογος έναντι λογοκλοπής, γεγονός που σημαίνει αποτυχία στην Πτυχιακή μου Εργασία και κατά συνέπεια αποτυχία απόκτησης του Τίτλου Σπουδών, πέραν των λοιπών συνεπειών του νόμου περί πνευματικών δικαιωμάτων. Δηλώνω, συνεπώς, ότι αυτή η Πτυχιακή Εργασία προετοιμάστηκε και ολοκληρώθηκε από εμένα προσωπικά και αποκλειστικά και ότι, αναλαμβάνω πλήρως όλες τις συνέπειες του νόμου στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δε μου ανήκει διότι είναι προϊόν λογοκλοπής άλλης πνευματικής ιδιοκτησίας."

Όνομα και Επώνυμο Συγγραφέα (Με Κεφαλαία):……..

Υπογραφή (Ολογράφως, χωρίς μονογραφή): ……

Ημερομηνία (Ημέρα – Μήνας – Έτος): ……..

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΙΝΑΚΑΣ ΕΙΚΟΝΩΝ

ΕΙΣΑΓΩΓΗ

1.ΣΚΟΠΟΣ ΚΑΙ ΙΔΕΑ

Ο σκοπός της εργασίας αυτής είναι η δημιουργία, αλλά και η καταγραφή του τρόπου με τον οποίο δημιουργήθηκε, ενός εικονικού περιβάλλοντος με χρήση μάσκας εικονικής πραγματικότητας. Εικονική πραγματικότητα(Virtual Reality) είναι η χρήση της τεχνολογίας που μας προσφέρουν οι υπολογιστές για την δημιουργία ενός προσομοιωμένου περιβάλλοντος στο οποίο ο χρήστης αλληλεπιδρά αλλά και βυθίζεται μέσα σε αυτό, αντιθέτως με τα κοινά περιβάλλοντα στα οποία ο χρήστης αλληλεπιδρά με το περιβάλλον μόνο με την χρήση οθόνης χωρίς να υπάρχει βύθιση. Η τεχνολογία που χρησιμοποιήθηκε για την κατασκευή αυτού του περιβάλλοντος είναι η παιχνιδομηχανή Unity η οποία μας παρέχει την δυνατότητα δημιουργίας και διαμόρφωσης του περιβάλλοντος μας για χρήση σε κινητά τηλέφωνα και ταυτόχρονα για την χρήση μασκών εικονικής πραγματικότητας. Στόχος ήταν να υλοποιηθεί η ιδέα ενός ατμοσφαιρικού περιβάλλοντος στο οποίο ο χρήστης προσπαθεί να επιβιώσει από αλλεπάλληλα κύματα ζωντανών – νεκρών (Zombie).

2.ΥΛΙΚΑ

Τα υλικά τα οποία χρησιμοποιήθηκαν είναι τα εξής:

1. Android κινητό τηλέφωνο (API Level>19)
2. Μάσκα εικονικής πραγματικότητας (VR Shinecon)
3. Unity 2017.3.1f1 (64-bit)
4. Microsoft Visual Studio
5. OEM Bluetooth χειριστήριο
6. Gamepad tester (Android application, υποχρεωτικό για την σωστή λειτουργία του χειριστηρίου)
7. Διάφορα αντικείμενα(assets) της παιχνιδομηχανής Unity

ΚΕΦΑΛΑΙΟ 1

1.ΧΕΙΡΙΣΜΟΣ

Ο χειρισμός σε αυτό το περιβάλλον επιτεύχθηκε με την χρήση ενός Bluetooth χειριστηρίου σε σύνδεση με το κινητό τηλέφωνο. Εκτός από το χειριστήριο ο χρήστης έχει την δυνατότητα να στρέφει την κάμερα του παιχνιδιού, χρησιμοποιώντας το γυροσκόπιο που είναι ενσωματωμένο στο κινητό τηλέφωνο, με την βοήθεια ενός κώδικα (script) τον οποίο έχει δημιουργήσει η Google ακριβώς για τέτοιου είδους εφαρμογές(application). Ένα τμήμα του κώδικα :

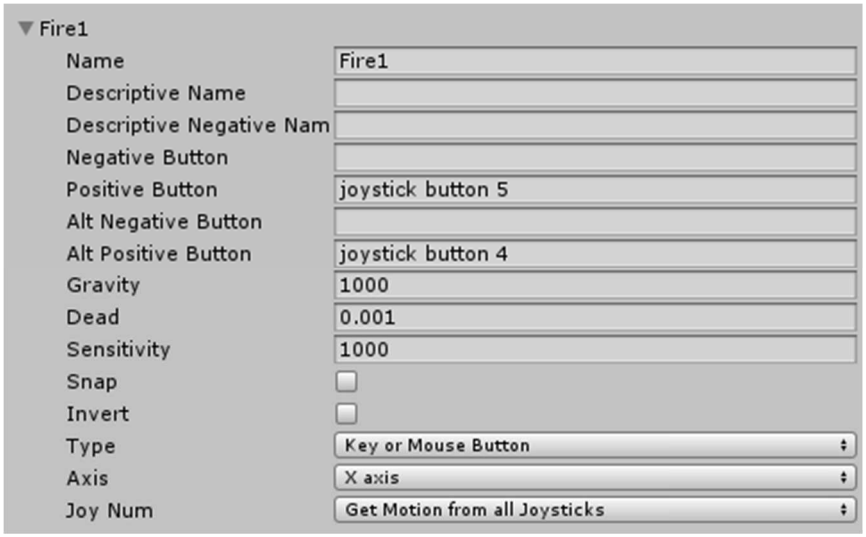
```
void OnEnable() {
  if (!SupportsPositionalTracking) {
    return;
  }
  standaloneUpdate = EndOfFrame();
  StartCoroutine(standaloneUpdate);
}
```

1.Κώδικας Google

Ο οποίος κώδικας ρωτάει την συσκευή αν υποστηρίζει αυτού του είδους την λειτουργία και ανανεώνει την θέση στο τέλος του κάθε frame(κάθε εικόνας που «τρέχει»).

Το Bluetooth χειριστήριο χρειάστηκε και αυτό να ρυθμιστεί για το τι λειτουργίες θα έχει το κάθε πλήκτρο του. Το Unity προσφέρει έναν οδηγό στον οποίο ρυθμίζεις τη είσοδο που δέχεται από το χειριστήριο και καλεί αναλόγως την μέθοδο που έχουμε ορίσει.

Στην παρακάτω εικόνα βλέπουμε την μέθοδο με όνομα “Fire1” ,τα «θετικά» πλήκτρα (πλήκτρα τα οποία όταν πατηθούν η μέθοδος “Fire1” θα γυρίσει true) τα οποία είναι το joystick button 4 και 5, και διάφορες ρυθμίσεις για το πώς θα λειτουργεί αυτή η μέθοδος.



2.Οδηγός Unity 2

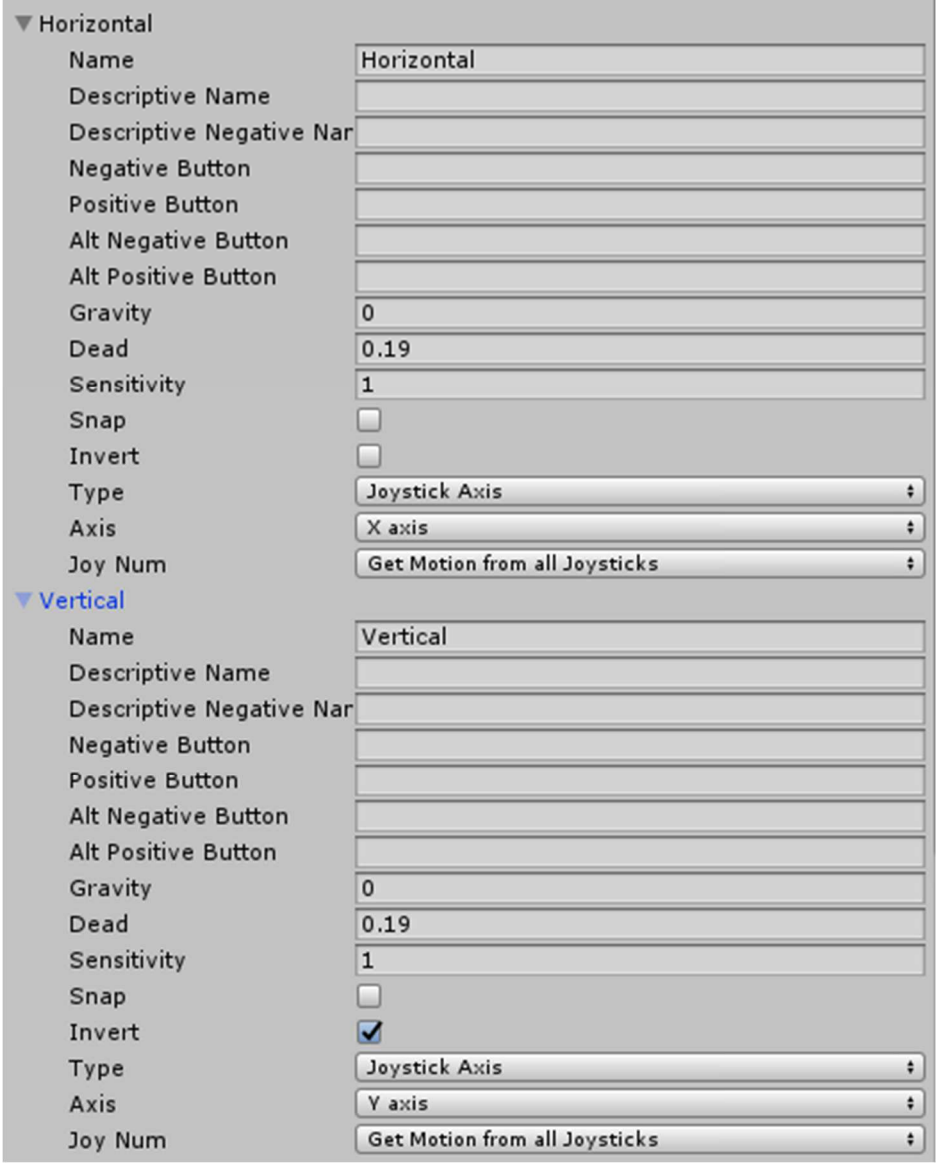
Για να βρούμε ποιο πλήκτρο αντιστοιχεί σε τι στο χειριστήριο μας, χρησιμοποιούμε την εφαρμογή Gamepad Tester που αναφέρθηκε παραπάνω. Θα πρέπει επίσης να δούμε στο εγχειρίδιο της Unity για το ποιο πλήκτρο αντιστοιχεί στο joystick button 4 και 5.

3.Κωδικοί πλήκτρων

4.Κωδικοί χειριστηρίου

Στην εικόνα 4 βλέπουμε ότι το πλήκτρο που θέλουμε να ενεργοποιεί την μέθοδο "Fire1" όταν το πατάμε αντιστοιχεί στο πλήκτρο L1. Το πλήκτρο L1 στην εικόνα 3 , έχει τον αριθμό 4 που σημαίνει ότι μπορούμε να κάνουμε την αντιστοιχία στον οδηγό(εικόνα 2) σύμφωνα με αυτό. Οπότε καθώς έχουμε βάλει στα «θετικά» πλήκτρα το joystick button 4 και 5(στο χειριστήριο μας L1 και R1 αντίστοιχα), όταν τα πατάμε η μέθοδος "Fire1" θα γίνεται αληθής.

Η αντιστοιχία του μοχλού γίνεται με διαφορετικό τρόπο από τα πλήκτρα. Στον οδηγό της, η παιχνιδομηχανή Unity δίνει την δυνατότητα επιλογής του τύπου εισόδου που θα δεχόμαστε για τις μεθόδους οριζόντιας κίνησης και κάθετης κίνησης. Οπότε επιλέξαμε ότι η είσοδος θα είναι τύπου μοχλού (Joystick Axis) όπως βλέπουμε παρακάτω:



5.Οδηγός Unity 2

Βλέπουμε ότι εκτός από τον τύπο εισόδου υπάρχει και η επιλογή από ποιόν άξονα να γίνεται θετική ή αρνητική η κάθε μέθοδος. Στην οριζόντια μέθοδο έχουμε θέσει τον άξονα X και στην κάθετη μέθοδο έχουμε θέσει τον άξονα Y. Επειδή ο μοχλός του χειριστήριού στέλνει τιμές -1 , 0 ή +1 για κάθε άξονα αντίστοιχα, δεν χρειάζεται να συμπληρωθούν τα αρνητικά και τα θετικά πλήκτρα που ενεργοποιούν την μέθοδο. Επίσης όπως βλέπουμε στην παραπάνω εικόνα έχει γίνει και η επιλογή "Invert",η οποία αντιστρέφει τις εισόδους, στον κάθετο άξονα γιατί το χειριστήριο στέλνει τιμές +1 όταν βρίσκεται στην κάτω θέση ενώ θα ήταν προτιμότερο να στέλνει -1, για να μην γίνει αλλαγή στον κώδικα.

1.2 ΚΙΝΗΣΗ ΧΡΗΣΤΗ – ΑΝΤΙΠΑΛΩΝ

Καθώς ρυθμίστηκε το χειριστήριο μας, πρέπει τώρα οι μεθόδοι που ενεργοποιούνται, να αξιοποιούνται με κάποιο τρόπο. Στην παρακάτω εικόνα 5 βλέπουμε ότι όταν ενεργοποιείται η μέθοδος "Fire1" τότε γίνεται κλήση της μεθόδου "Shoot()" που θα δούμε παρακάτω.

```
void Update () {
    if (Input.GetButtonDown("Fire1"))
    {
        Shoot();
    }
}
```

6.Μέθοδος Fire1 χρήση

Για την κίνηση του χρήστη χρησιμοποιήθηκε ένα standard asset της παιχνιδομηχανής Unity, το οποίο λέγεται "Rigid body Fps Controller" , με το οποίο το Unity δέχεται τις τιμές που στέλνει το χειριστήριο και τις μετατρέπει σε κίνηση όπως βλέπουμε στις παρακάτω εικόνες:

```
private Vector2 GetInput()
{

    Vector2 input = new Vector2
        {
            x = CrossPlatformInputManager.GetAxis("Horizontal"),
            y = CrossPlatformInputManager.GetAxis("Vertical")
        };
    movementSettings.UpdateDesiredTargetSpeed(input);
    return input;
}
```

7.Ανάθεση τιμών κίνησης x,y

```
public void UpdateDesiredTargetSpeed(Vector2 input)
{
    if (input == Vector2.zero) return;
    if (input.x > 0 || input.x < 0)
    {
        //strafe
        CurrentTargetSpeed = StrafeSpeed;
    }
    if (input.y < 0)
    {
        //backwards
        CurrentTargetSpeed = BackwardSpeed;
    }
    if (input.y > 0)
    {
        //forwards
        //handled last as if strafing and moving forward at the same time forwards speed should take precedence
        CurrentTargetSpeed = ForwardSpeed;
    }
```

8.Χρήση τιμών κίνησης

```
// always move along the camera forward as it is the direction that it being aimed at
Vector3 desiredMove = cam.transform.forward*input.y + cam.transform.right*input.x;
```

9.Μετατροπή τιμών σε σημεία στο χώρο

Βλέπουμε ότι δέχεται τις τιμές του χειριστηρίου και τις μετατρέπει σε σημεία στο χώρο με την ιδιότητα κάθε αντικειμένου στο περιβάλλον "transform".

Εκτός από κίνηση του χρήστη έχουμε και την κίνηση του αντιπάλου, η οποία έχει κατεύθυνση προς τον χρήστη. Αυτό γίνεται πάλι με την ιδιότητα "transform" όπου σαν τιμές δέχεται το σημείο που βρίσκεται εκείνη την στιγμή ο χρήστης. Αυτή η ιδιότητα διορθώνεται σε κάθε frame,μέσω της μεθόδου "Update" της παιχνιδομηχανής Unity που τρέχει αυτόματα σε κάθε frame, όπου πιθανών αλλάζει η θέση του χρήστη και έτσι έχουμε ένα ακριβές τρόπο να ακολουθεί ο αντίπαλος τον χρήστη. Παρακάτω βλέπουμε τον τρόπο με τον οποίο γίνεται:

```
private void Start()
{
    anim = GetComponent<Animator>();
    target = targetPlayer.instance.player.transform;
    rbody = GetComponent<Rigidbody>();
}
```

class UnityEngine.Rigidbod

10.Στόχος παίχτης

```
if (walk == true && dead==false)
{
    transform.LookAt(target);
    transform.position = Vector3.MoveTowards(transform.position, target.position, Time.deltaTime*4);

}
```

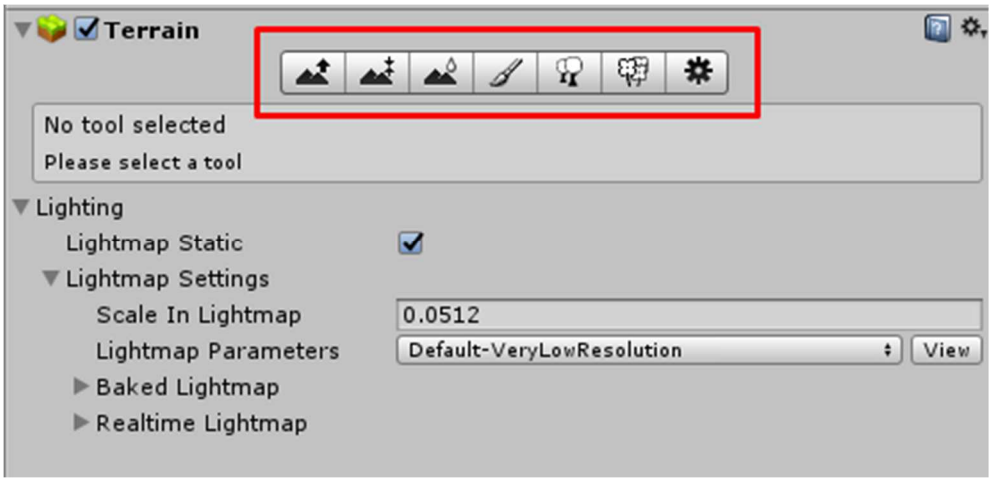
11.Κίνηση προς παίχτη

Όπως βλέπουμε στην εικόνα 9 με την μέθοδο "targerPlayer.transform" παίρνουμε εκείνη την χρονική στιγμή τις τιμές του "transform" για τον χρήστη και τις αποθηκεύουμε στην μεταβλητή "target" την οποία την χρησιμοποιούμε για να αλλάξουμε τις τιμές του "transform" του αντικειμένου μας όπως βλέπουμε στην εικόνα 10. Για να είναι ρεαλιστική η κίνηση και να μην υπάρχει το φαινόμενο της «τηλεμεταφοράς» χρησιμοποιούμε και τον χρόνο που περνάει, με την κλάση "Time". Με απλά λόγια «άλλαξε τη θέση από αυτή που βρίσκεσαι τώρα, προς τη θέση του στόχου με χρόνο "Time"».

ΚΕΦΑΛΑΙΟ 2

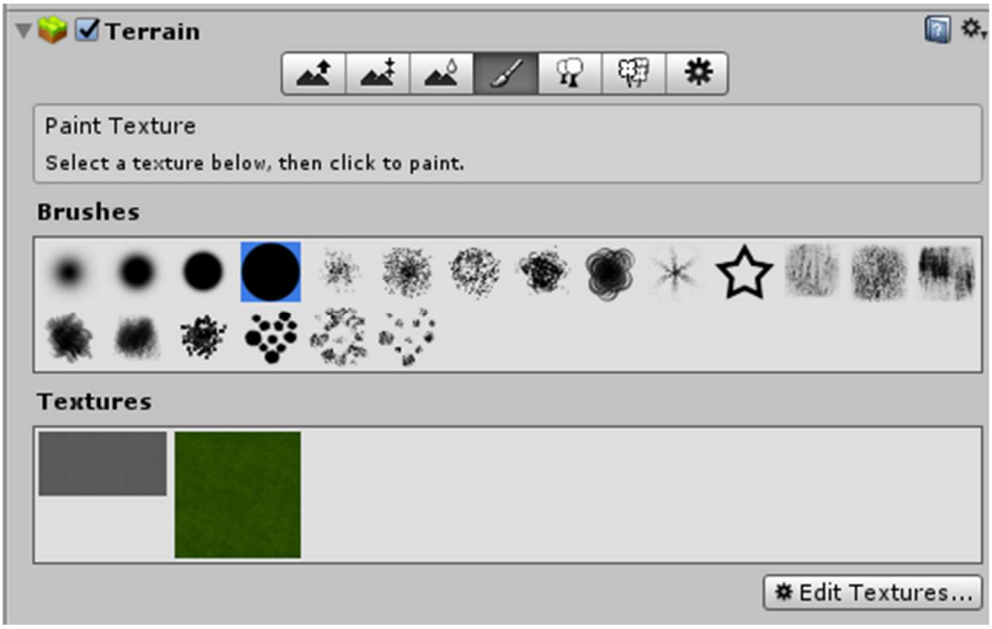
2.1 ΔΗΜΙΟΥΡΓΙΑ ΕΔΑΦΟΥΣ

Η δημιουργία του εδάφους στο περιβάλλον μας είναι πολύ απλή και έγινε με την χρήση του αντικειμένου "Terrain" της παιχνιδομηχανής Unity. Αρχικά δημιουργούμε ένα αντικείμενο τύπου "Terrain" και το μορφοποιούμε με τα υπάρχοντα στο Unity εργαλεία όπως βλέπουμε παρακάτω:

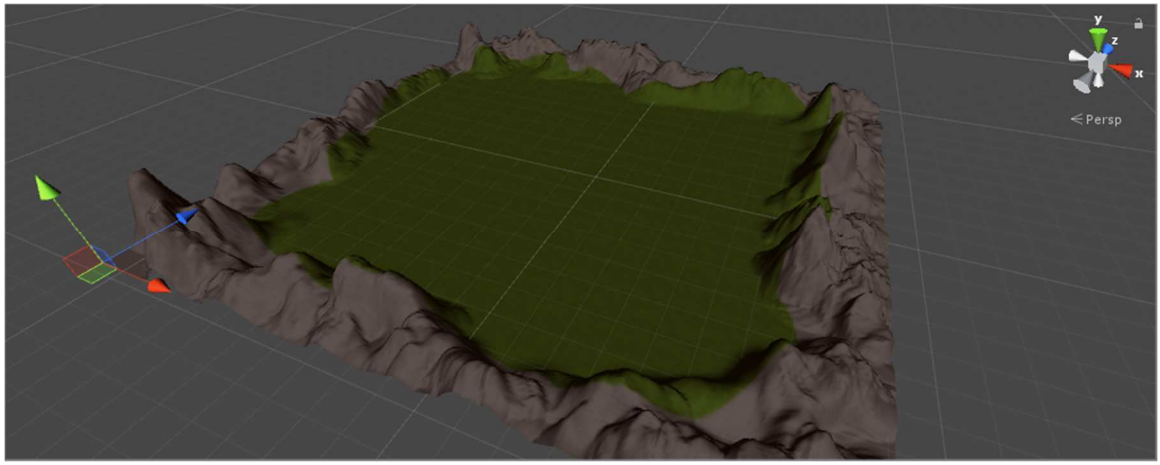


12.Έδαφος 1

Αφού κατασκευάσουμε το "Terrain" πρέπει να το χρωματίσουμε επιλέγοντας το είδος της βούρτσας και την υφή(texture) όπως βλέπουμε παρακάτω:



13.Έδαφος 2



14.Έδαφος Τελικό

Το τελικό αποτέλεσμα του εδάφους το βλέπουμε στην παραπάνω εικόνα. Τα κύρια εργαλεία που χρησιμοποιήθηκαν για την δημιουργία του εδάφους είναι:

1. Εργαλείο ρύθμισης ύψους(αύξηση-μείωση ύψους σε συγκεκριμένη ακτίνα της αρεσκείας μας)
2. Εργαλείο ομαλοποίησης (Μείωση της αιχμηρότητας των κορυφών ή των κοιλάδων)
3. Εργαλείο χρωματισμού εδάφους

2.2 ΛΟΙΠΑ ΔΙΑΚΟΣΜΗΤΙΚΑ-ΑΝΤΙΠΑΛΟΙ

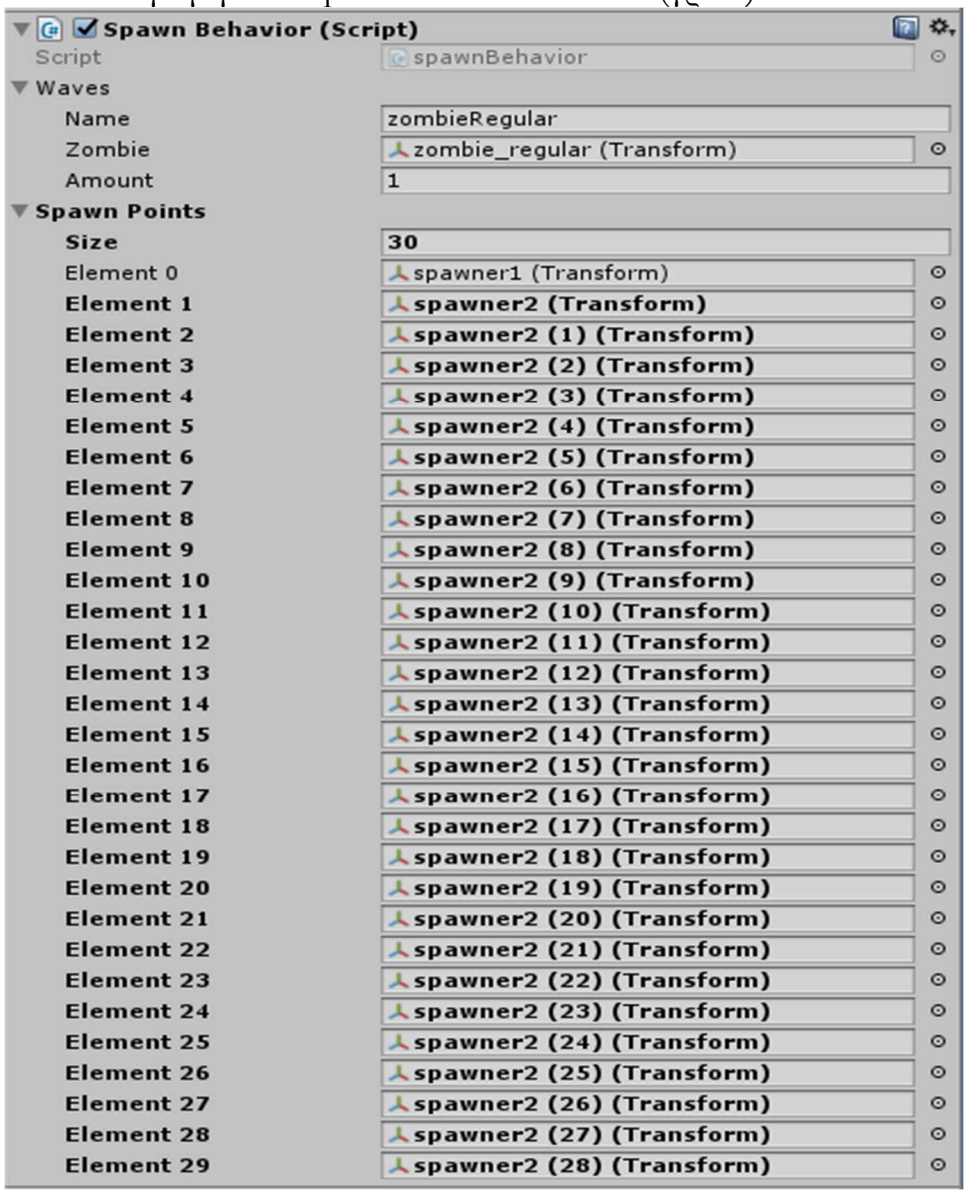
Τα υπόλοιπα διακοσμητικά π.χ. δέντρα, σπίτι, δαυλοί είναι έτοιμα αντικείμενα κατεβασμένα από το κατάστημα της παιχνιδομηχανής Unity, τα οποία τα εισαγάγαμε και τα τοποθετήσαμε στο σημείο της αρεσκείας μας.

Με τον ίδιο τρόπο δημιουργήθηκαν και οι αντίπαλοι, με τη διαφορά ότι δεν τους θέλαμε τοποθετημένους σε συγκεκριμένα σημεία. Για το λόγο αυτό κατασκευάστηκαν διάφορα σημεία στο περιβάλλον όπου ο αντίπαλος μπορεί να δημιουργηθεί(spawn).

```
9     [System.Serializable]
10    public class Wave
11    {
12        public string name;
13        public Transform zombie;
14        public int amount=1;
15    }
16    public Wave waves;
17    public Transform[] spawnPoints;
18
19    private float searchCountdown = 1f;
20    void Update () {
21        searchCountdown -= Time.deltaTime;
22        if (searchCountdown <= 0f)
23        {
24            if (GameObject.FindGameObjectWithTag("Zombie") == null)
25            {
26                //Debug.Log("Didn't find any zeds");
27                logic(waves);
28            }
29            else
30            {
31                //Debug.Log("Found zeds");
32            }
33        }
34    }
35    void logic(Wave wave)
36    {
37        for (int i = 0; i < wave.amount;i++)
38        {
39            spawnZombie(wave.zombie,wave.amount);
40        }
41        wave.amount = wave.amount + 1;
42
43    }
44    void spawnZombie(Transform zombie, int amount)
45    {
46        Transform spawnPoint = spawnPoints[Random.Range(0,spawnPoints.Length)];
47        Instantiate(zombie, spawnPoint.position, transform.rotation);
48
49    }
```

15.Spawner

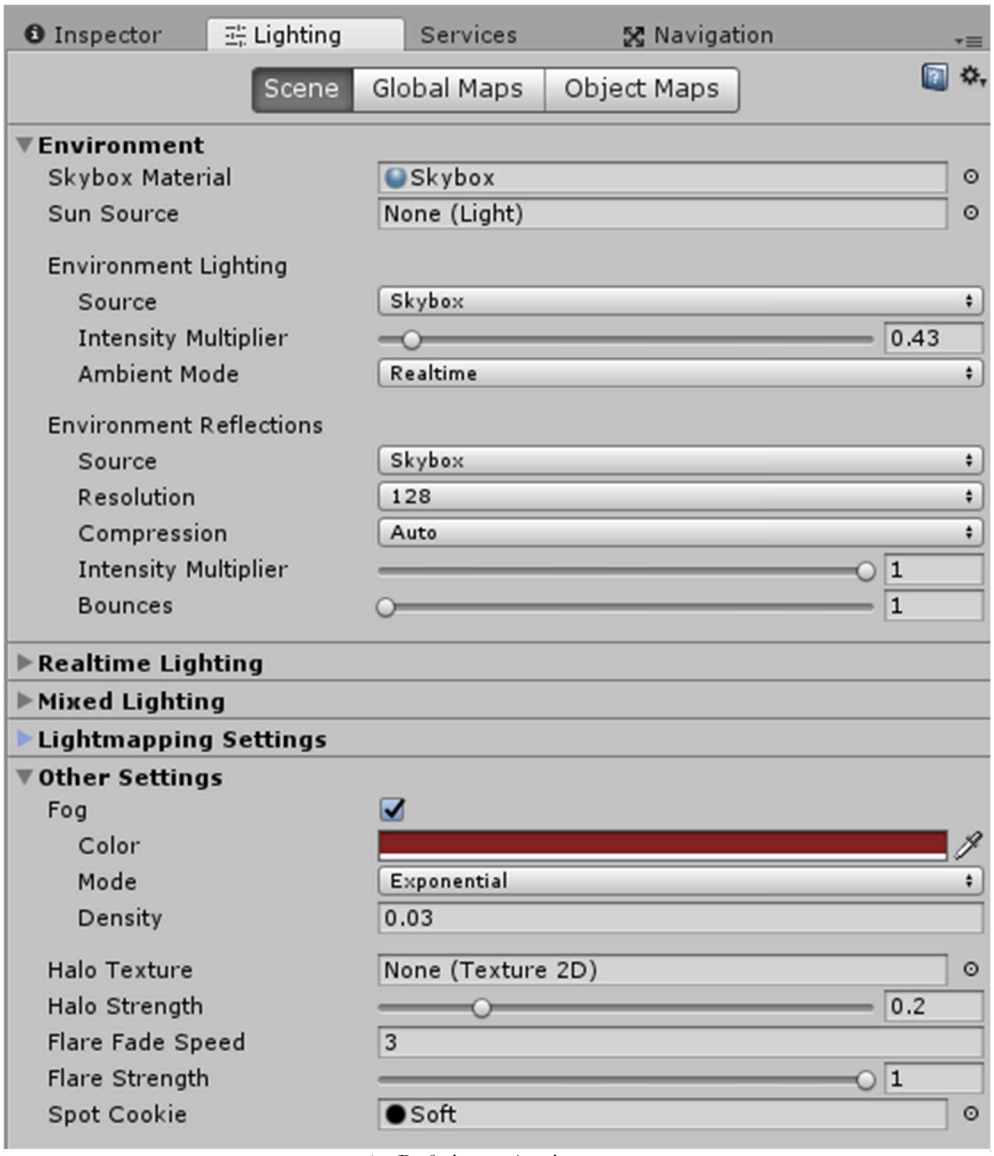
Όπως βλέπουμε στην παραπάνω εικόνα 13 δημιουργούμε έναν πίνακα με αντικείμενα "spawnPoints" (γρ.17).Γεμίζουμε τον πίνακα με τα σημεία που δημιουργήθηκαν στο περιβάλλον προηγουμένως όπως βλέπουμε στην εικόνα 14. Στη συνέχεια γίνεται έλεγχος για το αν υπάρχουν αντίπαλοι σε κάθε frame,εάν δεν υπάρχουν τότε εκτελείται η μέθοδος "logic()" η οποία παίρνει σαν μεταβλητή ένα αντικείμενο κλάσης "Wave"(κλάση Wave γρ.10). Για κάθε "amount" του αντικειμένου της κλάσης "Wave" αυτή η μέθοδος θα καλεί την μέθοδο "spawnZombie()" . Η μέθοδος "spawnZombie()" δημιουργεί ένα αντικείμενο "transform" με όνομα "spawnPoint" και παίρνει τιμές "transform" από ένα τυχαίο αντικείμενο που βρίσκεται μέσα στον πίνακα "spawnPoints"(γρ.46). Έπειτα καλείται η μέθοδος "Instantiate" η οποία δημιουργεί έναν κλώνο του αντικειμένου "zombie" στη τιμή του "spawnPoint.transform" (γρ.47).



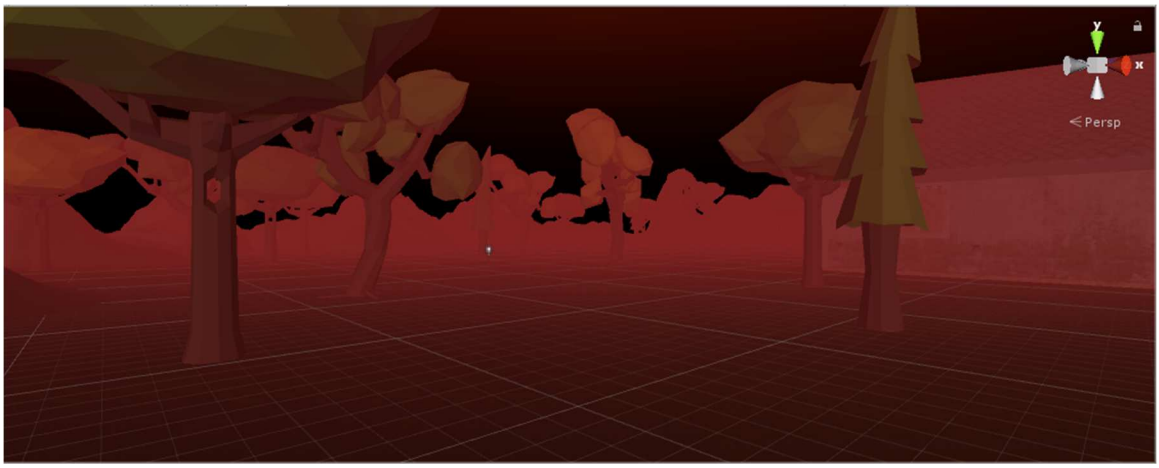
16.Πίνακας spawner 1

2.3 ΑΤΜΟΣΦΑΙΡΑ

Η ατμόσφαιρα του περιβάλλοντος μας δημιουργήθηκε με την λειτουργία - ρύθμιση "Lighting" της παιχνιδομηχανής Unity. Όπως βλέπουμε παρακάτω στην εικόνα 16 έχουμε επιλέξει την επιλογή "Fog" η οποία δημιουργεί μια ομίχλη με χρώμα, πυκνότητα και τρόπο γεμίσματος που εμείς επιθυμούμε:

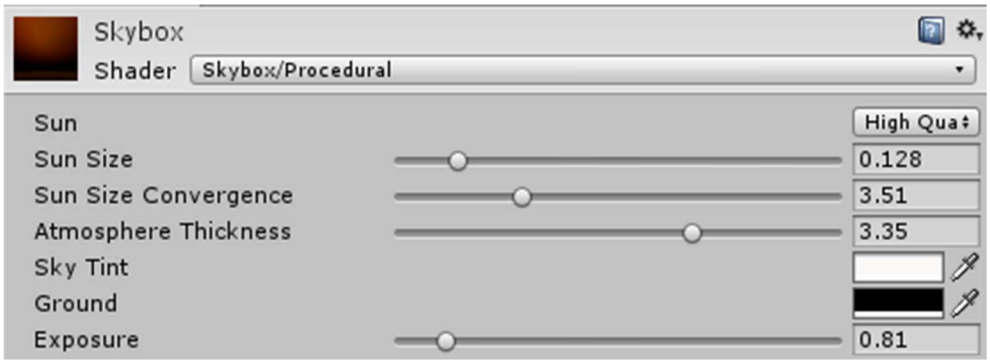


17.Ρυθμίσεις Ατμόσφαιρας



18. Ατμόσφαιρα

Παρατηρούμε ότι εκτός από την ομίχλη και τα αντικείμενα στον χώρο, υπάρχει και ο ουρανός. Ο ουρανός δημιουργήθηκε με την χρήση αντικειμένου της παιχνιδομηχανής Unity με όνομα "Skybox". Το αντικείμενο αυτό περιέχει ρυθμίσεις δημιουργώντας έτσι τον ουρανό που επιθυμούμε:



19. Ρυθμίσεις Ουρανού

Το τελικό αποτέλεσμα:

20.Ουρανός

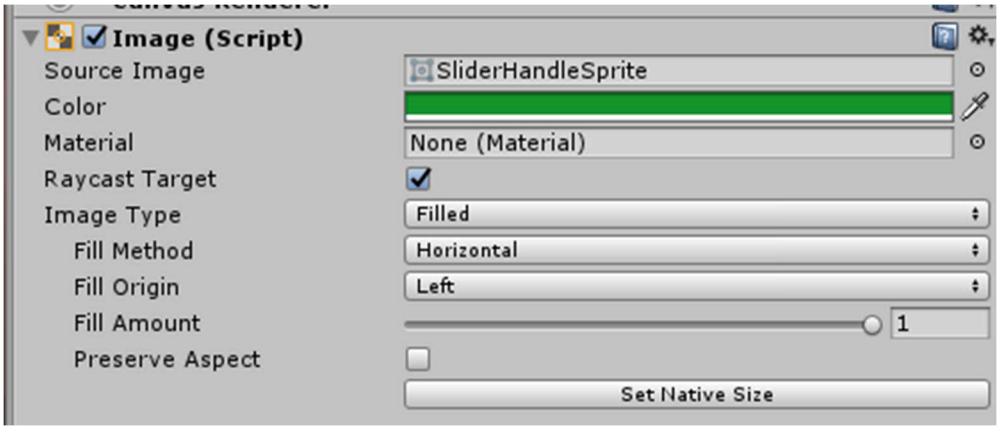
ΚΕΦΑΛΑΙΟ 3

3.1 ΔΙΕΠΑΦΗ ΧΡΗΣΤΗ (USER INTERFACE)

Η δημιουργία U.I. στο Unity είναι εύκολη χρησιμοποιώντας το αντικείμενο "Canvas" της παιχνιδομηχανής Unity. Θέτοντας το να κάνει render(να δημιουργείται) στην κεντρική μας κάμερα, οτιδήποτε βάλουμε πάνω στον καμβά θα εμφανιστεί απευθείας στην κάμερα μας. Έτσι δημιουργούμε 3 αντικείμενα μέσα στον καμβά.
Ένα για τον χρόνο επιβίωσης τύπου text,ένα για το μέτρημα αντιπάλων που σκοτώθηκαν τύπου text και ένα ακόμα αντικείμενο τύπου image για να φαίνονται οπτικά οι πόντοι ζωής μας. Το αντικείμενο τύπου image περιέχει και ένα αντικείμενο text για να φαίνονται οι πόντοι ζωής και γραπτά.

21.Πόντοι ζωής

Οι πόντοι ζωής είναι δυναμικοί και σε κάθε χτύπημα θα πρέπει να μειώνονται. Θέλουμε όμως εκτός από το text να αντιδράει και η πράσινη μπάρα τύπου image, οπότε ρυθμίζουμε την εικόνα μας να είναι τύπου "filled"(γεμίζει τμηματικά) και από ποια κατεύθυνση να γεμίζει(εικόνα 16). Καθώς ρυθμίστηκε έτσι μπορούμε να ελέγξουμε το γέμισμά της μέσω κώδικα όταν δεχόμαστε χτύπημα.



22.Ρυθμίσεις εικόνας

Ο έλεγχος του γεμίσματος γίνεται στην γραμμή 36 εικόνα 17.Διαιρούμε με το 100 γιατί το μέγεθος γεμίσματος παίρνει τιμές από 0-1 και οι πόντοι ζωής μας από το 0-100.Επίσης τα άλλα δύο text αντικείμενα μας ανανεώνονται στον ίδιο κώδικα. Η μεταβλητή "killCount" ελέγχεται από άλλο κώδικα που θα δούμε παρακάτω.

```
33          tSurvived = tSurvived + Time.deltaTime;
34          timeSurvived.text = "Time Survived:" + tSurvived.ToString("f1");
35          killCount.text = "Kill Count: " + killcount.ToString();
36          healthBar.fillAmount = health / 100f;
37          healthText.text = "HP" + health + "/100";
```

23.Κώδικας καμβά 1

3.2 ΟΠΛΟ

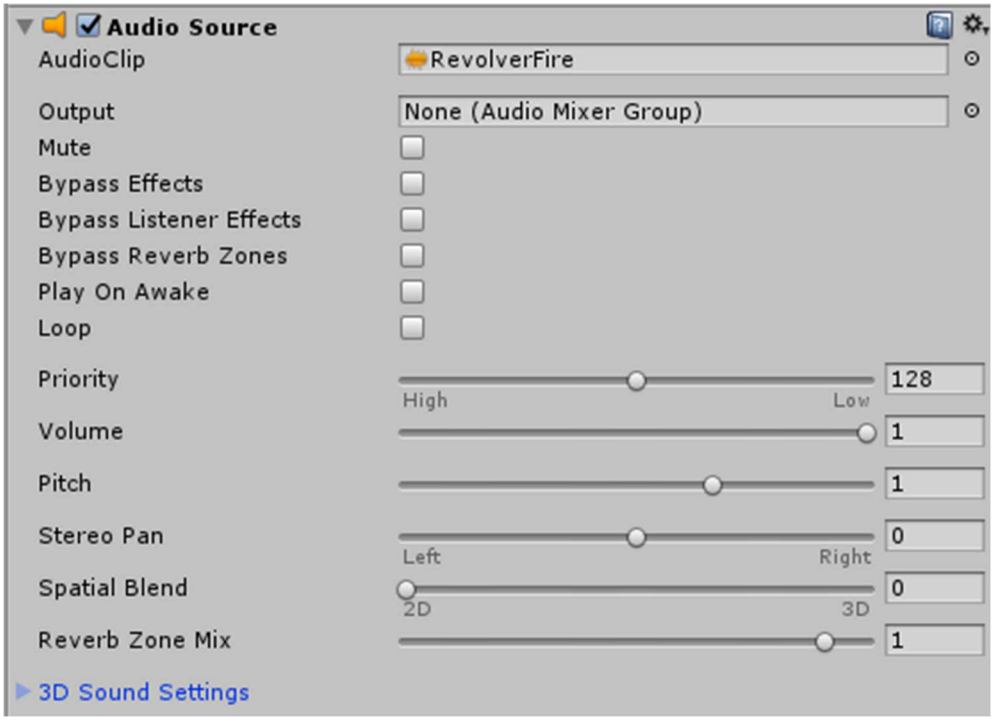
Το όπλο μας οπτικά είναι έτοιμο από το κατάστημα της Unity. Έπρεπε όμως να δημιουργηθεί η λάμψη του στομίου του όπλου και το σημείο πρόσκρουσης της σφαίρας. Στην πραγματικότητα δεν χρησιμοποιούμε σφαίρες αλλά την μέθοδο της παιχνιδομηχανής Unity "Raycast" η οποία μέθοδος στέλνει μία ακτίνα από το αρχικό μας αντικείμενο προς το μέρος που κοιτάμε. Εάν χτυπήσει κάτι τότε επιστρέφει το αντικείμενο που χτύπησε. Χρησιμοποιείται όπως βλέπουμε παρακάτω(γρ.23):

```
18      void Shoot()
19      {
20          gunshot.Play();
21          muzzleFlash.Play();
22          RaycastHit hit;
23          if (Physics.Raycast(fpsCam.transform.position, fpsCam.transform.forward, out hit))
24          {
25              Debug.Log(hit.transform.name);
26              Target target=hit.transform.GetComponent<Target>();
27              if (target != null)
28              {
29                  target.tookDamage(damage);
30              }
31
32              GameObject impactParticle = Instantiate(impactEffect, hit.point, Quaternion.LookRotation(hit.normal));
33              Destroy(impactParticle,2f);
34          }
35
36      }
37  }
```

24.Συμπεριφορά Όπλου 1

Εάν χτυπηθεί κάτι τότε καλείται η μέθοδος tookDamage().(Όλα τα αντικείμενα εκτός από του αντιπάλους θεωρούνται κενά(target=null))

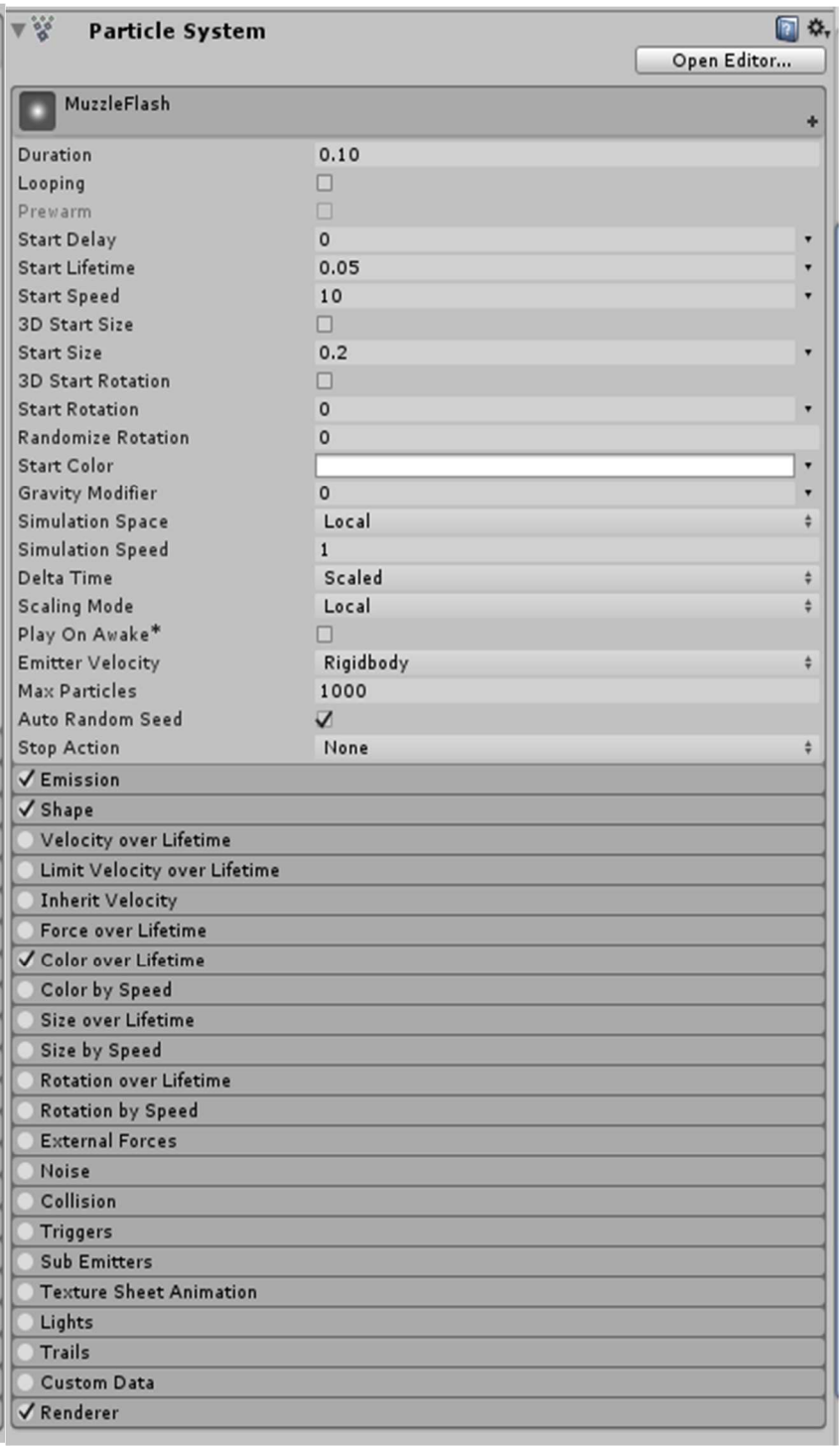
Όπως είχαμε δει στην εικόνα 6 όταν ο χρήστης πατήσει ένα συγκεκριμένο πλήκτρο τότε εκτελείται η μέθοδος Shoot() που βλέπουμε παραπάνω. Βλέπουμε επίσης ότι στη μέθοδο "Shoot()" εκτελείται η μέθοδος "gunshot.Play()" και "muzzleFlash.Play()" , και γίνεται κλωνοποίηση του σημείου πρόσκρουσης της σφαίρας με τη μέθοδο "Instantiate"(γρ.32 εικόνα 18). Η μέθοδος "gunshot.Play()" εκτελεί τον ήχο που έχουμε βάλει σαν ιδιότητα στο αντικείμενο μας, όπως βλέπουμε παρακάτω:



25. Ήχος Όπλου 1

Το σημείο πρόσκρουσης και η λάμψη του στομίου του όπλου δημιουργήθηκαν μέσω του αντικειμένου της παιχνιδομηχανής Unity "particle system" όπου με τις κατάλληλες ρυθμίσεις καταλήξαμε στο αποτέλεσμα της αρεσκείας μας(εικόνες 20 και 21).

26.Σημείο Πρόσκρουσης

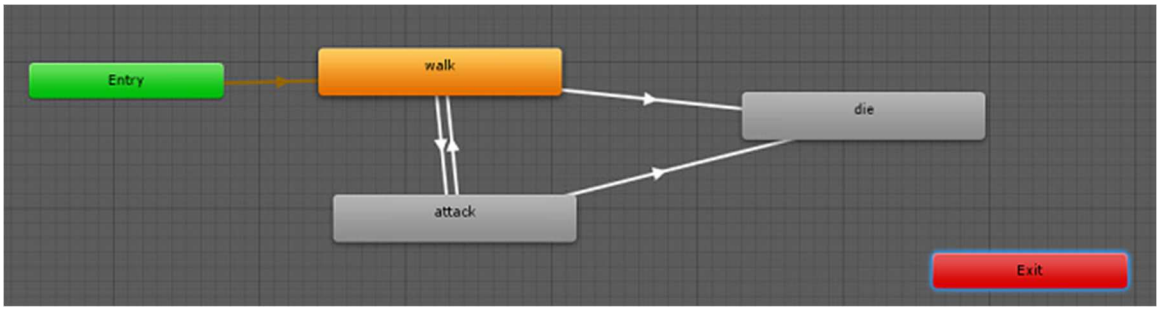


27. Λάμψη Στομίου

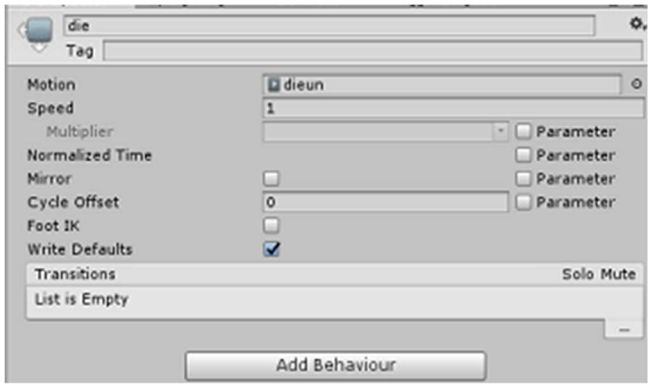
ΚΕΦΑΛΑΙΟ 4

4.1 ΚΙΝΗΣΗ ΑΝΤΙΠΑΛΟΥ (ANIMATION)

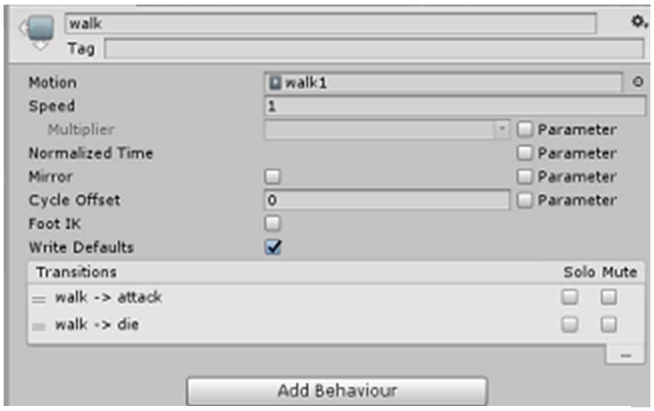
Προηγουμένως στο κεφάλαιο 1.2 είδαμε πως μπορούμε να μεταφέρουμε ένα αντικείμενο, στη συγκεκριμένη περίπτωση τους αντιπάλους, προς το μέρος που βρίσκεται ο χρήστης. Για να είναι ρεαλιστική η κίνηση πρέπει να προσθέσουμε και ένα είδος περπατήματος. Μαζί λοιπόν με τον αντίπαλο ,που βρήκαμε στο κατάστημα της παιχνιδομηχανής Unity, έχουμε και κάποιες κινήσεις(animation) που μπορούμε να χρησιμοποιήσουμε. Αυτές που χρησιμοποιήθηκαν είναι η κίνηση περπατήματος, η κίνηση επίθεσης, και η κίνηση θανάτου. Ο τρόπος με τον οποίο χρησισμοποιήθηκαν αυτές οι κινήσεις είναι με την λειτουργία της παιχνιδομηχανής Unity "Animator". Δημιουργήθηκαν τρεις καταστάσεις μία για περπάτημα, μία για επίθεση και μία για τον θάνατο, στις οποίες καταστάσεις περάστηκαν οι κινήσεις μας όπως φαίνεται παρακάτω:



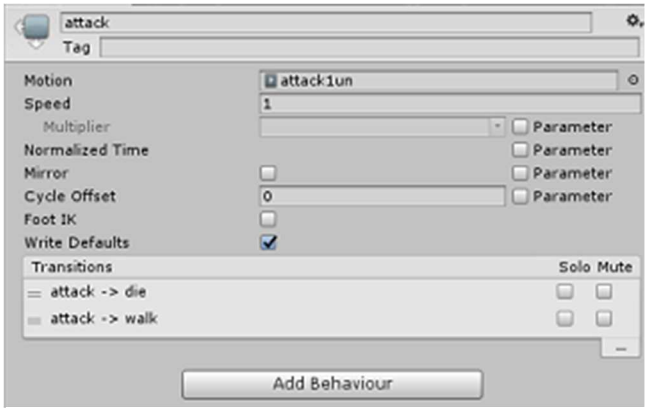
28. Animator Καταστάσεις



29.Κατάσταση θανάτου

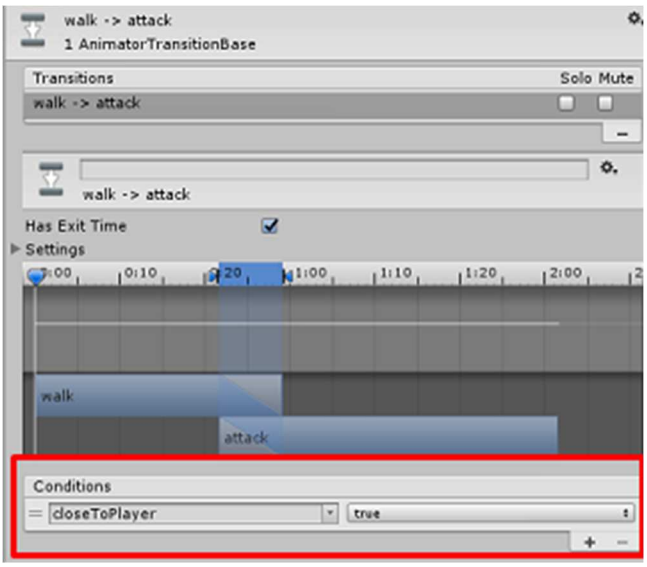


30. Κατάσταση Περπατήματος

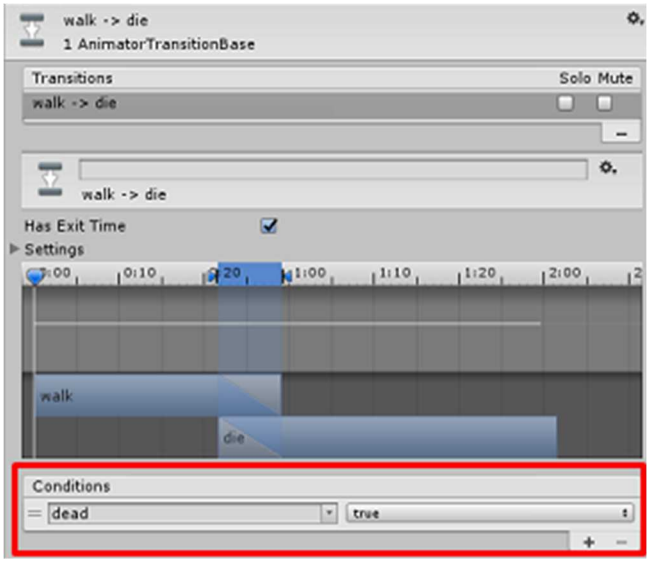


31. Κατάσταση επίθεσης

Για να γίνει η μετάβαση από την μία κατάσταση στην άλλη χρησιμοποιούμε κάποιες συνθήκες οι οποίες όταν γίνουν αληθής τότε μπορεί να γίνει η μετάβαση αλλιώς γίνεται επανάληψη της κίνησης της κατάστασης. Οι συνθήκες φαίνονται παρακάτω:



32. Συνθήκη επίθεσης



33. Συνθήκη θανάτου

Οι μεταβλητές "closeToPlayer" και "dead" ελέγχονται από το script του αντιπάλου που θα δούμε παρακάτω. Η ίδια συνθήκη χρησιμοποιήται για να γίνει η μετάβαση στην κατάσταση "dead", δηλαδή σε όποια κατάσταση βρίσκεται εκείνη τη στιγμή, είτε στην κατάσταση επίθεσης είτε στην κατάσταση περπατήματος , μπορεί να μεταβεί στην κατάσταση θανάτου.

```
31      if ((target.transform.position - this.transform.position).sqrMagnitude < 4 * 4)
32      {
33          walk = false;
34          anim.SetBool("closeToPlayer", true);
35
36          if (Manager.attacked==true)
37          {
38              hit.Play();
39              Manager.health = Manager.health - damage;
40              Manager.attacked = false;
41          }
42
43      }
44      else
45      {
46          walk = true;
47          anim.SetBool("closeToPlayer", false);
```

34.Αλλαγή μεταβλητών

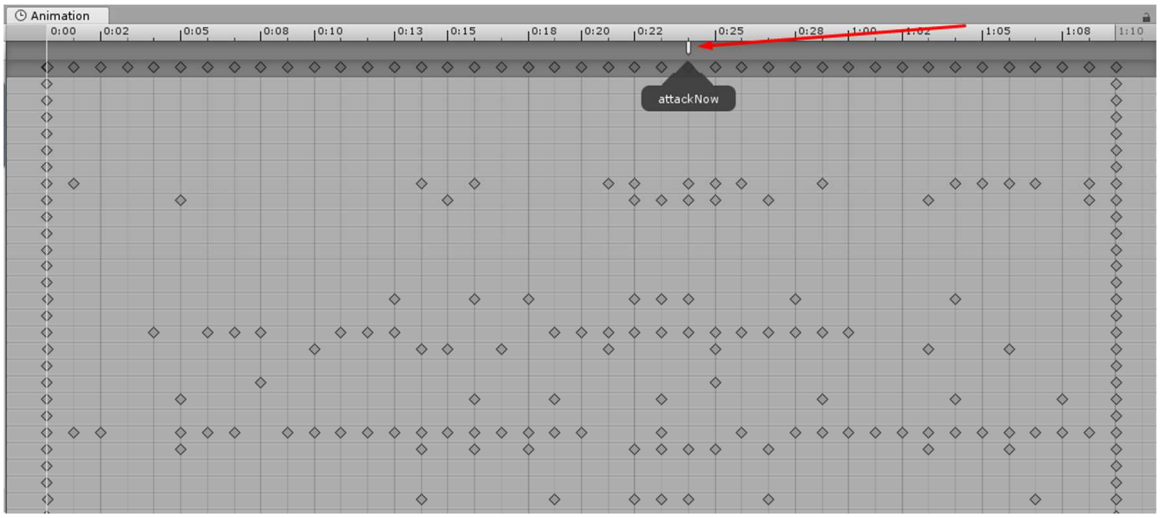
Όπως βλέπουμε στην εικόνα 34 αν η θέση του χρήστη είναι σε μικρότερη απόσταση από 4x4 από τον αντίπαλο τότε η μεταβλητή του animator "closeToPlayer" γίνεται αληθής και γίνεται η μετάβαση από την κατάσταση "walk" στην κατάσταση "attack". Σε περίπτωση που σταματήσει να ισχύει αυτό τότε η μεταβλητή αυτή γίνεται ψευδής και έχουμε επιστροφή στην κατάσταση "walk".

```
61      public void tookDamage(int amount)
62      {
63          hp -= amount;
64          zedHealthBar.fillAmount = hp / 100f;
65          if(hp <= 0)
66          {
67              anim.SetBool("dead", true);
68              walk = false;
69              dead = true;
70              Death();
71          }
```

35. Μέθοδος tookDamage()

Στην εικόνα 35 βλέπουμε την μέθοδο tookDamage() η οποία καλείται από το όπλο μας όπως είδαμε προηγουμένως. Εάν οι πόντοι ζωής του αντιπάλου φτάσουν στο μηδέν ή πιο κάτω τότε η μεταβλητή του animator "dead" γίνεται αληθής και σε οποιαδήποτε κατάσταση και να βρίσκεται γίνεται η μετάβαση στην κατάσταση "dead" και η κίνηση μας ενεργοποιείται.

Εκτός από τις ενεργοποιήσεις των καταστάσεων έχουμε προσθέσει στην κίνηση επίθεσης του αντιπάλου έναν ήχο σε ένα συγκεκριμένο frame της κίνησης επίθεσης. Ο τρόπος που υλοποιήθηκε είναι με τη λειτουργία "Animation" της παιχνιδομηχανής Unity και "addEvent" που βρίσκεται μέσα στο animation όπως βλέπουμε παρακάτω:



36. Πίνακας Animation

Όπως βλέπουμε στην εικόνα 36 στο σημείο(frame) μεταξύ 0:22 και 0:25 τοποθετήθηκε ένα "event" το οποίο καλεί την μέθοδο "attackNow()" και όπως είδαμε στην εικόνα 34, καθώς θα γίνει αληθής η συνθήκη με την μεταβλητή "Manager.attacked", θα αναπαραχθεί ο ήχος και θα γίνει αφαίρεση πόντων ζωής από τον χρήστη.

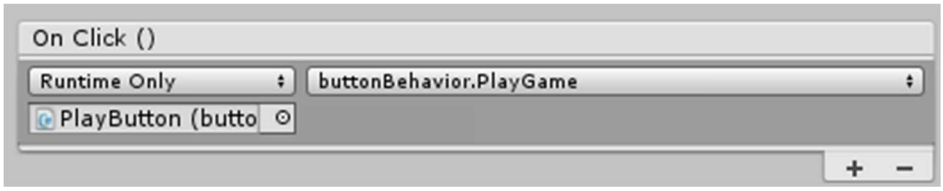
```
void attackNow()
{
    Manager.attacked = true;
}
```

37.Μέθοδος attackNow()

ΚΕΦΑΛΑΙΟ 5

5.1 ΜΕΝΟΥ, ΕΙΣΑΓΩΓΗ , ΤΕΛΟΣ

Για το μενού χρησιμοποιήθηκε ένας καμβάς ο οποίος δημιουργείται(render) στην κεντρική μας κάμερα με φόντο την εικόνα επιλογής μας και με ένα αντικείμενο τύπου πλήκτρο (Button). Όταν ενεργοποιηθεί το συγκεκριμένο πλήκτρο, τότε φορτώνεται η επόμενη σκηνή. Η επόμενη σκηνή καλείται με την χρήση της βιβλιοθήκης της παιχνιδομηχανής Unity "SceneManagement" και κλήση της μεθόδου "SceneManager.LoadScene()" στην οποία τοποθετούμε το όνομα της σκηνής που θέλουμε να φορτώσουμε.

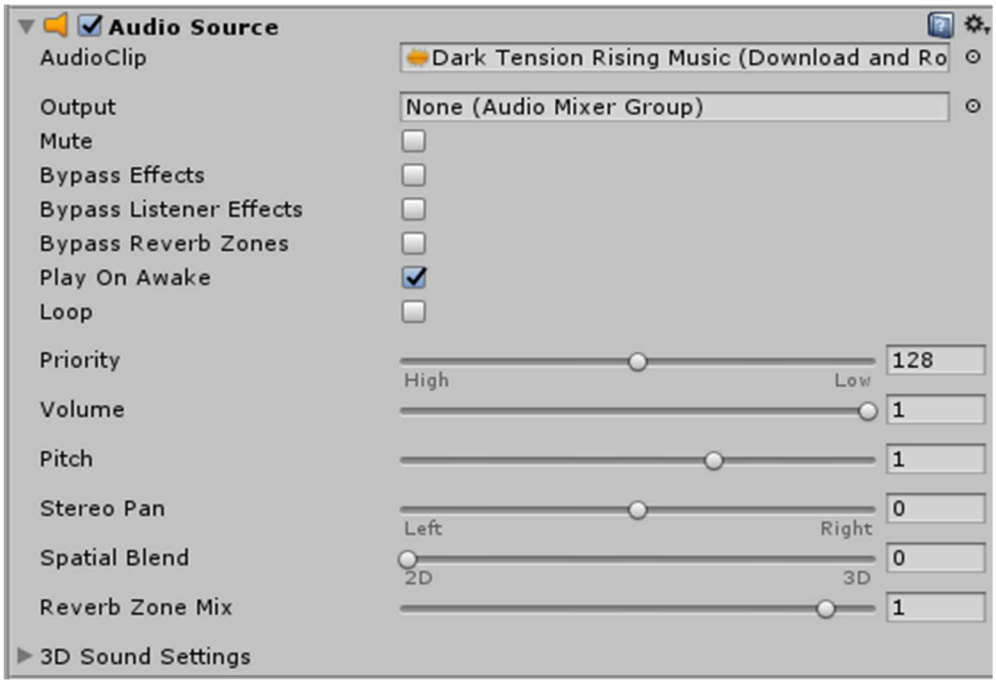


38. Πλήκτρο Μενού

```
public void PlayGame()
{
    SceneManager.LoadScene("intro");
}
```

39. Κώδικας πλήκτρου μενού

Επίσης έχουμε χρησιμοποιήσει έναν ήχο ο οποίος αναπαράγεται στην εκκίνηση της σκηνής(PlayOn Awake).



40. Μουσική μενού

Για την εισαγωγή και το τέλος ο τρόπος υλοποίησης είναι ο ίδιος. Χρησιμοποιήθηκε η μέθοδος "image.CrossFadeAlpha()" η οποία θέτει σαν πρώτο όρισμα την αρχική τιμή "Alpha" της εικόνας, την ταχύτητα «ξεθωριάσματος», την οποία την ορίζουμε εμείς από την μεταβλητή fadespeed και την τιμή false, η οποία αναφέρεται στο αν θέλουμε να αγνοήσει την κλίμακα του χρόνου.

```
26      void Show()
27      {
28          image.CrossFadeAlpha(1.0f, fadeSpeed, false);
29      }
30      void Fade()
31      {
32          image.CrossFadeAlpha(0.0f, fadeSpeed + 0.5f, false);
33      }
```

41.CrossFadeAlpha

Στη συνέχεια απλά καλούμε τις μεθόδους "show()" και "fade()" που εκτελούν την μέθοδο "crossFadeAlpha()" και αφού ολοκληρωθεί καλούμε πάλι την μέθοδο "SceneManager.LoadScene()" και φορτώνεται η επόμενη σκηνή.

ΠΗΓΕΣ

1. https://forum.unity.com/
2. https://answers.unity.com/index.html
3. https://docs.unity3d.com/Manual/index.html
4. https://unity3d.com/learn/tutorials
5. http://wiki.unity3d.com/index.php/Main_Page
6. http://forum.brackeys.com/