ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πτυχιακή Εργασία

Τίτλος Πτυχιακής Εργασίας	Περιπέτεια στο διάστημα: Μια Unity 3D εφαρμογή τρίτου προσώπου Adventure in Space: A third person Unity 3D application
Ονοματεπώνυμο Φοιτητή	Φοίβος Θεοδωρίδης
Πατρώνυμο	Θεόδωρος
Αριθμός Μητρώου	Π/20002
Επιβλέπων	Παναγιωτόπουλος Θεμιστοκλής, Καθηγητής

Ημερομηνία Παράδοσης Ιούνιος 2024

Copyright ©

Ευχαριστίες

Με την παρούσα πτυχιακή εργασία ολοκληρώνεται ο κύκλος των σπουδών μου στο προπτυχιακό πρόγραμμα του Τμήματος Πληροφορικής, εμβαθύνοντας τις γνώσεις μου σε ένα ιδιαίτερα ελκυστικό κλάδο.

Η παρούσα πτυχιακή εργασία αποτελεί το αποτέλεσμα μιας σειράς αλληλεπιδράσεων με διάφορα άτομα, τα οποία έπαιξαν σημαντικό ρόλο στην εξέλιξη της. Σε αυτό το σημείο θα ήθελα να ευχαριστήσω αυτά τα άτομα, για την στήριξη αλλά και για την βοήθεια που μου πρόσφεραν προκειμένου να ολοκληρωθεί αυτή η προσπάθεια.

Ιδιαίτερες ευχαριστίες οφείλω στον καθηγητή μου και επιβλέπων της πτυχιακής εργασίας κ. Παναγιωτόπουλο Θεμιστοκλή για την επιστημονική αλλά και για την συμβουλευτική καθοδήγηση που μου παρείχε κατά την διάρκεια της συγγραφής της εργασίας με τις εύστοχες παρατηρήσεις του αλλά και για το πόσο φιλικός και πρόθυμος ήταν στην συνεργασία μας για να φέρω εις πέρας την πτυχιακή μου εργασία.

Επίσης, θα ήθελα να ευχαριστήσω τους/τις συμφοιτητές/τριες μου, αλλά και τους φίλους μου για την κατανόηση που έδειξαν όλο αυτό το διάστημα της συγγραφής της πτυχιακής μου εργασίας, ενθαρρύνοντας με καθημερινά με το δικό τους, μοναδικό τρόπο.

Κλείνοντας, ευχαριστώ του γονείς μου, Θεόδωρο και Χριστίνα, όπως και την αδελφή μου Άντρια που δέχθηκαν τις επιλογές μου και μου παρείχαν την στήριξη που χρειαζόμουν όλο αυτό το διάστημα και ευγνωμονώ να στέκονται πάντα δίπλα μου τόσο στις επιτυχίες όσο και στις αποτυχίες, δίνοντας μου την δύναμη να συνεχίσω να προσπαθώ για κάθε μου απόφαση.

Περίληψη

Η παρούσα πτυχιακή εργασία ασχολείται με το παιχνίδι “Adventure in Space ” η οποία είναι πλήρως υλοποιημένη. Η πτυχιακή εργασία υλοποιήθηκε στο Unity με την γλώσσα προγραμματισμού C#.

Αναλυτικότερα, η παρούσα αναφορά αποτελείται από τέσσερα κεφάλαια. Αρχικά, γίνεται μια εισαγωγή για την εφαρμογή και το πως λειτουργά, καθώς και ποιες ανάγκες εξυπηρετεί. Στο πρώτο κεφάλαιο, γίνεται αναφορά στα Scenes που υπάρχουν και παρουσιάζονται τα αντίστοιχα Scripts, περιγράφοντας αναλυτικά την χρήση του κάθε Script. Στο δεύτερο κεφάλαιο, αναφέρονται τα Animations που χρησιμοποιηθήκαν και το πως εφαρμόστηκαν με την χρήση του Animator. Στο τρίτο κεφάλαιο, φαίνεται η σωστή λειτουργία του κώδικα καθώς και το πως φαίνεται ο κόσμος. Τέλος, στο τέταρτο κεφάλαιο, εκφράζονται συμπεράσματα και μελλοντικές επεκτάσεις.

Σκοπός μου κατά την διάρκεια της συγγραφής, δεν ήταν μόνο η ορθή και όσο το δυνατόν πληρέστερη ανάλυση του θέματος. Έγινε προσπάθεια, έτσι ώστε το περιεχόμενο της εργασίας να είναι κατανοητό και σαφές, γι’ αυτό η ανάλυση του θέματος έγινε με την βοήθεια των Screenshots από τον κώδικα και το παιχνίδι.

Επέλεξα το συγκεκριμένο θέμα εργασίας λόγω του ενδιαφέροντος μου για ανάπτυξη παιχνιδιών τα οποία θα μπορέσουν να παίξουν και να απολαύσουν άτομα που τους ενδιαφέρουν τα συγκεκριμένα ήδη παιχνιδιών (RPG games). Όσο περνάνε τα χρόνια, πιστεύω πως ο κώδικας για τα παιχνίδια απλοποιείται και μέσω του διαδικτύου και του Asset Store για το κάθε engine, ο καθένας θα μπορεί να δημιουργήσει το παιχνίδι των ονείρων του.

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Scripts, Unity, C#, Animator, RPG, Asset Store

Abstract

This present dissertation focuses on the game "Adventure in Space" which is fully implemented. The thesis was developed in Unity using the C# programming language.

Specifically, this report consists of four chapters. In the first chapter, an introduction to the application and how it works is provided, as well as the needs it addresses. The second chapter discusses the existing Scenes and presents the corresponding Scripts, describing in detail the use of each Script. The third chapter mentions the Animations used and how they were implemented using the Animator. Finally, the fourth chapter expressed the conclusion and future extensions.

My aim during the writing of this dissertation was not only the accuracy and completing is as much as possible. An effort was made to ensure that the content of the dissertation is understandable and clear, which is why the analysis of the subject was done with the help of Screenshots from the code and the game.

I chose this topic due to my interest in game development, specifically games that can be played and enjoyed by people interested in this genre (RPG games. As the years go by, I believe that the coding for games will become more simplified, and through the internet and the Asset Store for each engine, everyone will be able to create the game of their dreams.

Key Words: Scripts, Unity, C#, Animator, RPG, Asset Store

Περιεχόμενα

Κατάλογος Εικόνων

Εισαγωγή

Αρχικά, το θέμα που επιλέχθηκε από μέρους μου για την πτυχιακή μου εργασία είναι το ακόλουθο:

Παιχνίδι που διαδραματίζεται σε ένα άλλο κόσμο, όπου ανακαλύπτονται νέοι πλανήτες και χρειάζονται άμεσα εξερεύνηση για να ελεγχθεί αν υπάρχει κάποιος κίνδυνος στον πλανήτη του κύριου χαρακτήρα.

Για την δημιουργία του παιχνιδιού, σκέφτηκα πως θα ήταν ωραία να φτιάξω ένα παιχνίδι φαντασίας, ένα παιχνίδι που δεν διαδραματίζεται στην Γη, αλλά σε ένα άλλο κόσμο. Η πτυχιακή αυτή είχε σκοπό την απόκτηση εμπειρίας στην πλατφόρμα Unity, με την χρήση της γλώσσας προγραμματισμού C#. Έχουν χρησιμοποιηθεί αρκετά αντικείμενα που διδάχθηκαν στο μάθημα, όπως NPCs, δημιουργία του χώρου, χρήση των Assets από το Unity Asset Store, διάλογοι, interactions, combat, μουσικής, animations, ρυθμίσεων του παιχνιδιού για την καλύτερη εμπειρία του χρήστη και χρήση έξυπνων συνδυασμών για το αποτέλεσμα άλλων στόχων όπως θάνατος του τέρατος και διαφορετικοί διάλογοι. Αυτό συγκεντρώνει το επιστημονικό ενδιαφέρον για τον κύριο λόγο ότι μπορούν να δημιουργηθεί ένα τεράστιο εύρος πραγμάτων με την χρήση λίγων εργαλείων. Δηλαδή, με την χρήση του Collision, μπόρεσα να δημιουργήσω την ομιλία των NPCs, καθώς και τα μηνύματα που στέλνει το άτομο που καθόρισε την αποστολή στον κύριο χαρακτήρα. Επιπρόσθετα, είναι πολύ ενδιαφέρον το τι μπορεί να δημιουργήσει κάποιος με λίγη φαντασία.

Το θέμα που αποφάσισα να ακολουθήσω ήταν μια περιπέτεια στο διάστημα, όπου ο Main Character εξερευνά ένα καινούργιο πλανήτη σαν αποστολή που τον έστειλαν τα άτομα από τον πλανήτη του. Κατά την διάρκεια του παιχνιδιού, ο Main Character λαμβάνει μηνύματα από το άτομο που τον επιβλέπει για να τον προειδοποιήσει για τυχόν κινδύνους αλλά και για να τον καθοδηγήσει. Με αυτόν τον τρόπο ο χρήστης ξέρει πότε πρέπει να είναι προσεκτικός. Ο σκοπός του παιχνιδιού είναι το να βοηθήσει αυτόν τον πλανήτη επειδή τον έχουν καταβάλει τέρατα, έτσι ώστε να πάρει την εμπιστοσύνη των κατοίκων του. Ο χρήστης μπορεί να μιλήσει με του κάτοικους αλλά και να σκοτώσει τα τέρατα που απειλούν τα άτομα του πλανήτη. Ήθελα να χρησιμοποιήσω όσες περισσότερες γνώσεις μπορούσα από το μάθημα αλλά και από το διαδίκτυο για να εμπλουτιστεί περισσότερο το παιχνίδι.

Ελπίζω ότι με αυτή την επιστημονική συνεισφορά, πολλοί άνθρωποι θα δουν την ομορφιά του Game Development και την ανεξαρτησία που υπάρχει στην δημιουργία δικού τους παιχνιδιού, καθώς και την απόλαυση που θα μπορούν να αποκτήσουν όταν δουν το τελικό αποτέλεσμα παρά τις τυχόν δυσκολίες που θα προκύψουν κατά την δημιουργία του παιχνιδιού αυτού.

Παρακάτω, θα αναλύσω τα scripts που χρησιμοποίησα όπως και screenshots από το παιχνίδι, όπου θα εξηγώ τον τρόπο που σκέφτηκα για την δημιουργία του κάθε script. Επίσης, θα δούμε εις βάθος τα Scenes που χρησιμοποίησα.

Κεφάλαιο 1: Scenes

1.1 Start Menu

Εδώ μπορείτε να δείτε την αρχική οθόνη του παιχνιδιού.

FIGURE 1: START MENU SCREEN

Αρχικά, για να δουλέψει το Start Menu, έπρεπε να δημιουργήσω τα κουμπιά που θα μου έδιναν την δύναμη να παίξω το παιχνίδι (Play), να δω τις ρυθμίσεις του παιχνιδιού (Options) καθώς και το κουμπί για κλείσιμο του παιχνιδιού (Quit). Αυτά τα 3 κουμπιά χρησιμοποίησαν το Script “StartMenu.cs” έτσι ώστε να γίνεται η εναλλαγή μεταξύ των Scenes.

1.1.1 StartMenu.cs

Assets > _Scripts > StartMenu.cs > ...

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4   using UnityEngine.SceneManagement;
5
    0 references
6   public class StartMenu : MonoBehaviour
7   {
        0 references
8       public void StartGame()
9       {
10          SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1); //loads the next scene, the game
11      }
12
        0 references
13      public void QuitGame()
14      {
15          Application.Quit(); //quit game
16      }
17
        0 references
18      public void OptionsMenu()
19      {
20          SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 2); //load options
21      }
22  }
```

FIGURE 2: STARTMENU.CS

Απ’ ότι βλέπουμε στον παραπάνω κώδικα, έχουμε 3 function, που αντιστοιχούν σε κάθε Scene και την έξοδο του παιχνιδιού. Μέσα στο Scene Manager όρισα με την σειρά τα Scenes (Start Menu, Game, Options) όπου είχαν τις τιμές 0-2 αντίστοιχα.

Στο πρώτο screenshot μπορείτε να δείτε και 2 σημαίες, την Ελληνική και την Αγγλική. Με το πάτημα των κουμπιών αυτών, ο χρήστης έχει την δυνατότητα να αλλάξει την γλώσσα του παιχνιδιού.

FIGURE 3: START MENU SCREEN GREEK

Αυτό επιτυγχάνεται με την χρήση του Script “Localization.cs”

1.1.2 Localization.cs

Localization.cs

Assets > _Scripts > Localization.cs > ...

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4
5   using UnityEngine.Localization.Settings;
6
7
    0 references
8   public class Localization : MonoBehaviour
9   {
        3 references
10      private bool active = false;
        0 references
11      public void ChangeLocale(int localeID){
12          if(active == true){
13              return;
14          }
15          StartCoroutine(SetLocale(localeID));
16      }
17
        1 reference
18      IEnumerator SetLocale(int _localeID){
19          active = true;
20          yield return LocalizationSettings.InitializationOperation;
21          LocalizationSettings.SelectedLocale=LocalizationSettings.AvailableLocales.Locales[_localeID];
22          active = false;
23      }
24  }
```

FIGURE 4: LOCALIZATION.CS

Στον παραπάνω κώδικα, γίνεται η εναλλαγή της γλώσσας με την βοήθεια του Localization package της Unity. Εφόσον δημιούργησα την IEnumerator SetLocale συνάρτηση, η οποία διαβάζει το localeid του κάθε

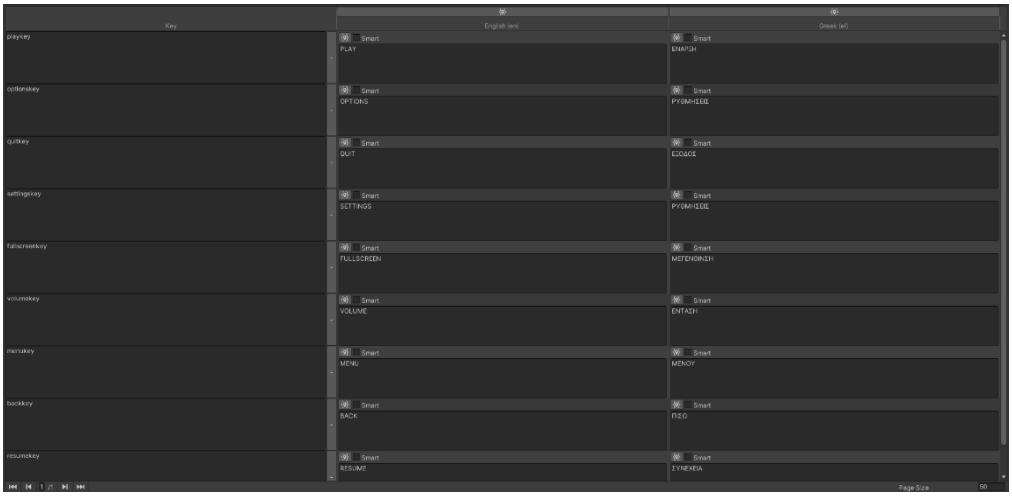


FIGURE 5: LOCALE

Locale για κάθε γλώσσα και με την Boolean μεταβλητή active και την πιο πάνω μέθοδο ChangeLocale ελέγχει αν τα κουμπιά που όρισα σε κάθε μενού για εύκολη αλλαγή γλώσσας πατήθηκαν ή όχι, χρησιμοποιώντας επίσης τα key που έβαλα στα Localization Tables του Asset Management του project μας. Αυτός ο κώδικας καλείτε στο OnClick() event κάθε κουμπιού μέσω του object LocalizationManager,διαβάζοντας το localeid της αντίστοιχης γλώσσας (Αγγλικά 0, Ελληνικά 1).

1.2 Game

Παρακάτω μπορείτε να δείτε τον κόσμο που δημιούργησα από ψηλά.

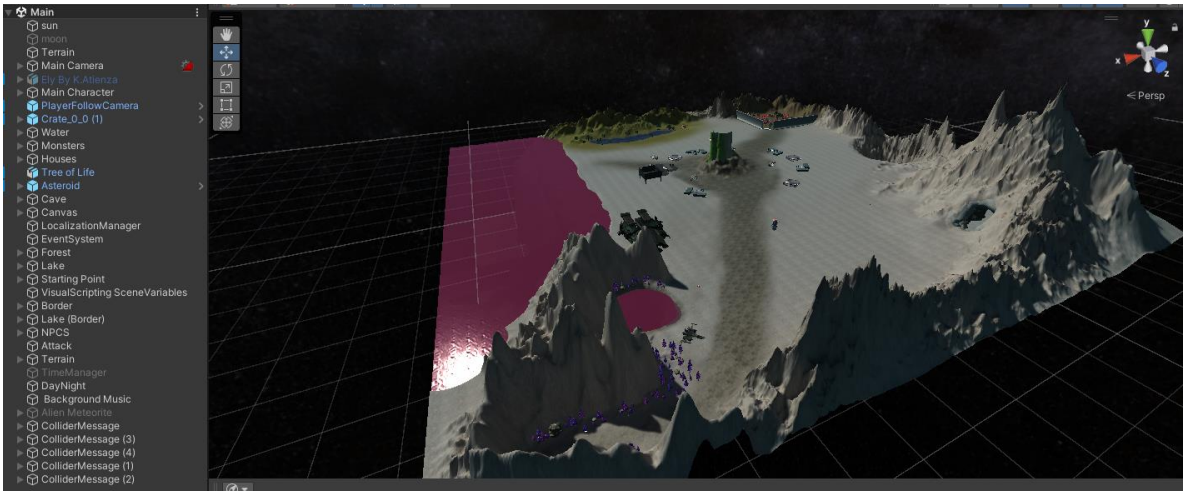


FIGURE 6: GAME FROM ABOVE

1.2.1 AttackScript.cs

AttackScript.cs

Assets > _Scripts > AttackScript.cs > ...

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4
    0 references
5   public class NewBehaviourScript : MonoBehaviour
6   {
7
        2 references
8       public GameObject MainCharacter;
9
10  //Attack Animation according to the pressed button
        0 references
11      void Update()
12      {
13          if(Input.GetKeyDown(KeyCode.Alpha1)){
14              MainCharacter.GetComponent<Animator>().Play("Breakdance Swipes");
15          }
16
17          if(Input.GetMouseButtonDown(0))
18          {
19              MainCharacter.GetComponent<Animator>().Play("hit1");
20          }
21      }
22  }
```

FIGURE 7: ATTACKSCRIPT.CS

Σε αυτό τον κώδικα, γίνονται τα animations της επίθεσης του χρήστη. Υπάρχουν δύο είδη τα οποία ελέγχονται από το κουμπί “1” στο πληκτρολόγιο και το αριστερό click του mouse. Αυτά τα κουμπιά δηλώθηκαν μέσα στο input manager από το unity. Με το κουμπί “1”, ο χρήστης εκτελεί ένα χορό, ενώ με το αριστερό click του mouse, ο χρήστης εκτελεί την επίθεση του.

1.2.2 Crate1.cs

Crate1.cs

Assets > _Scripts > Crate1.cs > ...

```
using System;
using System.Collections;
using System.Collections.Generic;
using Unity.VisualScripting;
using UnityEngine;
using UnityEngine.InputSystem;

0 references
public class Crate1 : MonoBehaviour
{
    1 reference
    public Animator animator;
    3 references
    [SerializeField] bool player=false;
    5 references
    bool isOpen = false;
    1 reference
    [SerializeField] private KeyCode interactKey = KeyCode.E; //Interact with a certain key

    2 references
    public AudioSource src;
    1 reference
    public AudioClip sfx1;
    1 reference
    public GameObject canvas;
    1 reference
    public GameObject canvas2;

    0 references
    void Update(){

        if (player==true)
            if (Input.GetKeyDown(interactKey)){
                src.clip=sfx1; //set the sound for the Crate
                src.Play(); //play the sound of the Crate
                isOpen = !isOpen;
                animator.SetBool("Open",isOpen);
                canvas.SetActive(!isOpen); //remove the current message
                canvas2.SetActive(isOpen); //make the new message appear
            }


    }

    0 references
    private void OnTriggerEnter(Collider other){
        if (other.gameObject.CompareTag("Player")){
            player=true;
        }
    }

    0 references
    private void OnTriggerExit(Collider other){
        if (other.gameObject.CompareTag("Player")){
            player=false;
        }
    }

}
```

FIGURE 8: CRATE1.CS

Σε αυτό τον κώδικα, βρίσκεται το animation που ανοίγει το αρχικό σεντούκι που βρίσκει ο χρήστης, όπου ορίζω τον ήχο ανοίγματος και κλεισίματός του, το άνοιγμα του με το Interact Key (δηλαδή το E). Επίσης εδώ υπάρχει ένα Trigger το οποίο ελέγχει αν ο χρήστης βρίσκεται κοντά από το σεντούκι για να μπορεί να το ανοίξει, ενώ αν δεν είναι μέσα σε αυτό το όριο να μην μπορεί να το ανοίξει.

1.2.3 DayNight.cs

DayNight.cs

Assets > _Scripts > DayNight.cs > DayNight > Update

```
1   using System;
2   using System.Collections;
3   using System.Collections.Generic;
4   using UnityEngine;
5
    0 references
6   public class DayNight : MonoBehaviour
7   {
8       [Range(0.0f, 1.0f)]
        12 references
9       public float time;
        1 reference
10      public float fullDayLength; //Sets day length
        1 reference
11      public float startTime = 0f;
        2 references
12      private float timeRate;
13
        2 references
14      public Vector3 noon;
15
16      [Header("Sun")]
        10 references
17      public Light sun;
        1 reference
18      public Gradient sunColor;
        1 reference
19      public AnimationCurve sunIntensity;
20
21      [Header("Moon")]
        8 references
22      public Light moon;
        1 reference
23      public Gradient moonColor;
        1 reference
24      public AnimationCurve moonIntensity;
25
26      [Header("Lighting")]
        1 reference
27      public AnimationCurve lightingIntensityMultiplier;
        1 reference
28      public AnimationCurve reflectionsIntensityMultiplier;
```

FIGURE 9: DAYNIGHT.CS

Σε αυτό τον κώδικα υλοποίησα το day-night circle, δηλαδή την αλλαγή της μέρας με την νύχτα στον κόσμο. Αρχικά δήλωσα μεταβλητές για την ώρα, την διάρκεια της μέρας, το φως, gradient χρώμα και κλίση της έντασης του ήλιου και του φεγγαριού που αντιστοιχούν στα δύο directional lights μας. Τέλος την κλίση των πολλαπλασιαστών έντασης του φωτός και των αντανακλάσεων.

DayNight.cs

Assets > _Scripts > DayNight.cs > DayNight > moon

```
6   public class DayNight : MonoBehaviour
        0 references
30      void Start()
31      {
32          timeRate = 1.0f / fullDayLength;
33          time = startTime;
34      }
35
        0 references
36      void Update()
37      {
38          time += timeRate * Time.deltaTime; //time increase
39
40          if(time >= 1.0f) time = 0.0f;
41
42          sun.transform.eulerAngles = (time - 0.25f) * noon * 10.0f;
43          sun.transform.eulerAngles = (time - 0.75f) * noon * 10.0f; //light rotation
44
45          sun.intensity = sunIntensity.Evaluate(time);
46          moon.intensity = moonIntensity.Evaluate(time); //light intensity
47
48          sun.color = sunColor.Evaluate(time);
49          moon.color = moonColor.Evaluate(time); // change colors
50
51          // enabling/disabling sun & moon
52          if (sun.intensity == 0 && sun.gameObject.activeInHierarchy) sun.gameObject.SetActive(false);
53          else if (sun.intensity > 0 && !sun.gameObject.activeInHierarchy) sun.gameObject.SetActive(true);
54
55          if (moon.intensity == 0 && moon.gameObject.activeInHierarchy) moon.gameObject.SetActive(false);
56          else if (moon.intensity > 0 && !moon.gameObject.activeInHierarchy) moon.gameObject.SetActive(true);
57
58          //lighting and reflection
59          RenderSettings.ambientIntensity = lightingIntensityMultiplier.Evaluate(time);
60          RenderSettings.reflectionIntensity = reflectionsIntensityMultiplier.Evaluate(time);
61      }
62  }
```

FIGURE 10: DAYNIGHT.CS ΣΥΝΕΧΕΙΑ

Στην συνάρτηση Start(), δηλώνω την διάρκεια της μέρας ενώ στην Update() δηλώνω το πως προχωρά η ώρα με βάση τα δευτερόλεπτα μεταξύ των frame και όταν φτάνει στο άκρο να ξανά ξεκινάει απ' την αρχή. Έπειτα δηλώνω το rotation του φωτός με βάση των ήλιο σε συγκεκριμένη ώρα, ελέγχω την ένταση και τα χρώματα του φωτός με την χρήση της μεθόδου Evaluate και της ώρας. Μετά, ελέγχω την αλλαγή με βάση το αν είναι enabled ή disabled τα directional lights μέσα στην ιεραρχία. Τέλος δηλώνω RenderSettings μεταβλητές για το φως και αντανακλάσεις χρησιμοποιώντας πάλι την μέθοδο Evaluate.

1.2.4 DoorScript.cs

DoorScript.cs

Assets > _Scripts > DoorScript.cs > ...

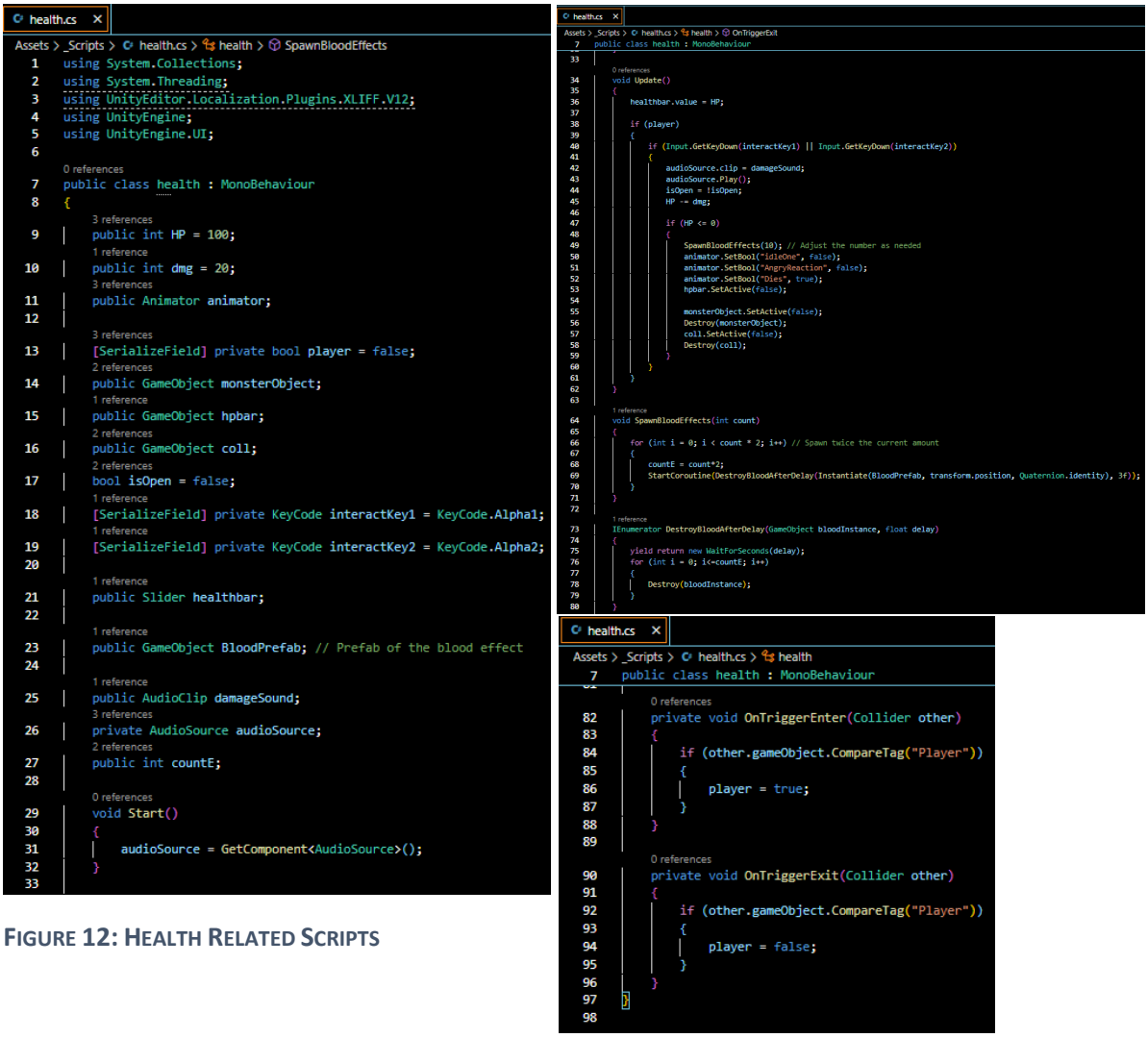
```
1   using UnityEngine;
2
    0 references
3   public class DoorScript : MonoBehaviour
4   {
        2 references
5       Animator animator;
        3 references
6       [SerializeField] bool player=false;
        3 references
7       bool isOpen = false;
        1 reference
8       [SerializeField] private KeyCode interactKey = KeyCode.E; //set the specific key to interact with doors
9
        0 references
10      void Start(){
11
12          animator = this.GetComponent<Animator>();
13
14      }
15
        0 references
16      void Update(){
17
18          if (player==true)
19              if (Input.GetKeyDown(interactKey)){
20                  isOpen = !isOpen;
21                  animator.SetBool("character_nearby",isOpen); //if character is nearby, open the door
22              }
23
24      }
25
26
            0 references
27      private void OnTriggerEnter(Collider other){
28              if (other.gameObject.CompareTag("Player")){
29                  player=true;
30              }
31      }
32
                0 references
33      private void OnTriggerExit(Collider other){
34              if (other.gameObject.CompareTag("Player")){
35                  player=false;
36              }
37      }
38  }
```

FIGURE 11: DOORSCRIPT.CS

Εδώ, βρίσκετε ο κώδικας για το άνοιγμα των πορτών στα σπίτια. Με το που πλησιάσει την οποιαδήποτε πόρτα ο χρήστης, το οποίο ελέγχετε με το Trigger, να μπορεί να πατήσει το Interact Key (“E”) για να μπορεί να ανοίξει και να κλείσει τις πόρτες.

1.2.5 health.cs, healthFlyingIncest.cs, healthMonsterScavenger.cs

health.cs



```
health.cs
Assets > _Scripts > health.cs > health > SpawnBloodEffects
1   using System.Collections;
2   using System.Threading;
3   using UnityEditor.Localization.Plugins.XLIFF.V12;
4   using UnityEngine;
5   using UnityEngine.UI;
6
    0 references
7   public class health : MonoBehaviour
8   {
        3 references
9       public int HP = 100;
        1 reference
10      public int dmg = 20;
        3 references
11      public Animator animator;
12
        3 references
13      [SerializeField] private bool player = false;
        2 references
14      public GameObject monsterObject;
        1 reference
15      public GameObject hpbar;
        2 references
16      public GameObject coll;
        2 references
17      bool isOpen = false;
        1 reference
18      [SerializeField] private KeyCode interactKey1 = KeyCode.Alpha1;
        1 reference
19      [SerializeField] private KeyCode interactKey2 = KeyCode.Alpha2;
20
        1 reference
21      public Slider healthbar;
22
        1 reference
23      public GameObject BloodPrefab; // Prefab of the blood effect
24
        1 reference
25      public AudioClip damageSound;
        3 references
26      private AudioSource audioSource;
        2 references
27      public int countE;
28
        0 references
29      void Start()
30      {
31          audioSource = GetComponent<AudioSource>();
32      }
33
```

```
health.cs
Assets > _Scripts > health.cs > health > OnTriggerExit
7   public class health : MonoBehaviour
33
        0 references
34      void Update()
35      {
36          healthbar.value = HP;
37
38          if (player)
39          {
40              if (Input.GetKeyDown(interactKey1) || Input.GetKeyDown(interactKey2))
41              {
42                  audioSource.clip = damageSound;
43                  audioSource.Play();
44                  isOpen = !isOpen;
45                  HP -= dmg;
46
47                  if (HP <= 0)
48                  {
49                      SpawnBloodEffects(10); // Adjust the number as needed
50                      animator.SetBool("idleOne", false);
51                      animator.SetBool("AngryReaction", false);
52                      animator.SetBool("Dies", true);
53                      hpbar.SetActive(false);
54
55                      monsterObject.SetActive(false);
56                      Destroy(monsterObject);
57                      coll.SetActive(false);
58                      Destroy(coll);
59                  }
60              }
61          }
62      }
63
        1 reference
64      void SpawnBloodEffects(int count)
65      {
66          for (int i = 0; i < count * 2; i++) // Spawn twice the current amount
67          {
68              countE = count*2;
69              StartCoroutine(DestroyBloodAfterDelay(Instantiate(BloodPrefab, transform.position, Quaternion.identity), 3f));
70          }
71      }
72
        1 reference
73      IEnumerator DestroyBloodAfterDelay(GameObject bloodInstance, float delay)
74      {
75          yield return new WaitForSeconds(delay);
76          for (int i = 0; i<=countE; i++)
77          {
78              Destroy(bloodInstance);
79          }
80      }
```

```
health.cs
Assets > _Scripts > health.cs > health
7   public class health : MonoBehaviour
        0 references
82      private void OnTriggerEnter(Collider other)
83      {
84          if (other.gameObject.CompareTag("Player"))
85          {
86              player = true;
87          }
88      }
89
        0 references
90      private void OnTriggerExit(Collider other)
91      {
92          if (other.gameObject.CompareTag("Player"))
93          {
94              player = false;
95          }
96      }
97  }
98
```

FIGURE 12: HEALTH RELATED SCRIPTS

healthFlyingInsect.cs και healthMonsterScavenger.cs είναι τα ίδια Scripts. Ο λόγος που δημιούργησα 3 scripts αντί για ένα, είναι επειδή είχα σκοπό να κάνει κάτι διαφορετικό το κάθε τέρας.

Στους πιο κάτω κώδικες βρίσκονται οι δηλώσεις των μεταβλητών για τον κώδικα των health bar των τεράτων NPC. Δηλώνω αρχικά το μέγεθος του HP του κάθε τέρατος και του damage που μπορεί να κάνει πάνω τους ο χρήστης.

Χρησιμοποιώντας το αριστερό κλικ του mouse που είναι το attack που αναφέραμε πιο πάνω, κάθε φορά που το πατάει ο χρήστης, εφόσον βρίσκεται κοντά από το κάθε τέρας το οποίο ελέγχεται όπως και προηγουμένως από Trigger, το HP του τέρατος που επιλέγει να χτυπήσει μειώνεται κατά 20.

Όταν το HP φτάσει κάτω από 0, το τέρας κάνει το ανάλογο animation ή βγάζει αίμα και εξαφανίζεται.

1.2.6 HealthNpc.cs

HealthNpc.cs

Assets > _Scripts > HealthNpc.cs > ...

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4   using UnityEngine.UI;
5
    0 references
6   public class HealthNpc : MonoBehaviour
7   {
        4 references
8       Animator animator;
        1 reference
9       public int HP = 100; //defines a certain amount of health to the NPC
10
        1 reference
11      public Slider healthbar;
        2 references
12      [SerializeField] bool player=false;
13
        0 references
14      void Start(){
15
16      animator = this.GetComponent<Animator>();
17      animator.SetBool("wave",true);
18
19      }
20
        0 references
21      void Update(){
22          healthbar.value=HP; //updates the NPC's healthbar if it takes damage
23      }
24
25
        0 references
26      private void OnTriggerEnter(Collider other){
27          if (other.gameObject.CompareTag("Player")){
28              player=true;
29              animator.SetBool("wave",false);
30          }
31      }
32
        0 references
33      private void OnTriggerExit(Collider other){
34          if (other.gameObject.CompareTag("Player")){
35              player=false;
36              animator.SetBool("wave",true);
37          }
38
39      }
40  }
```

FIGURE 13: HEALTHNPC.CS

Εδώ είναι ο κώδικας που ορίζω το health bar των NPC, αλλά και το wave animation τους, δηλαδή όταν τους πλησιάζει ο χρήστης, τον χαιρετάνε.

1.2.7 LookAtCamera.cs

LookAtTheCamera.cs

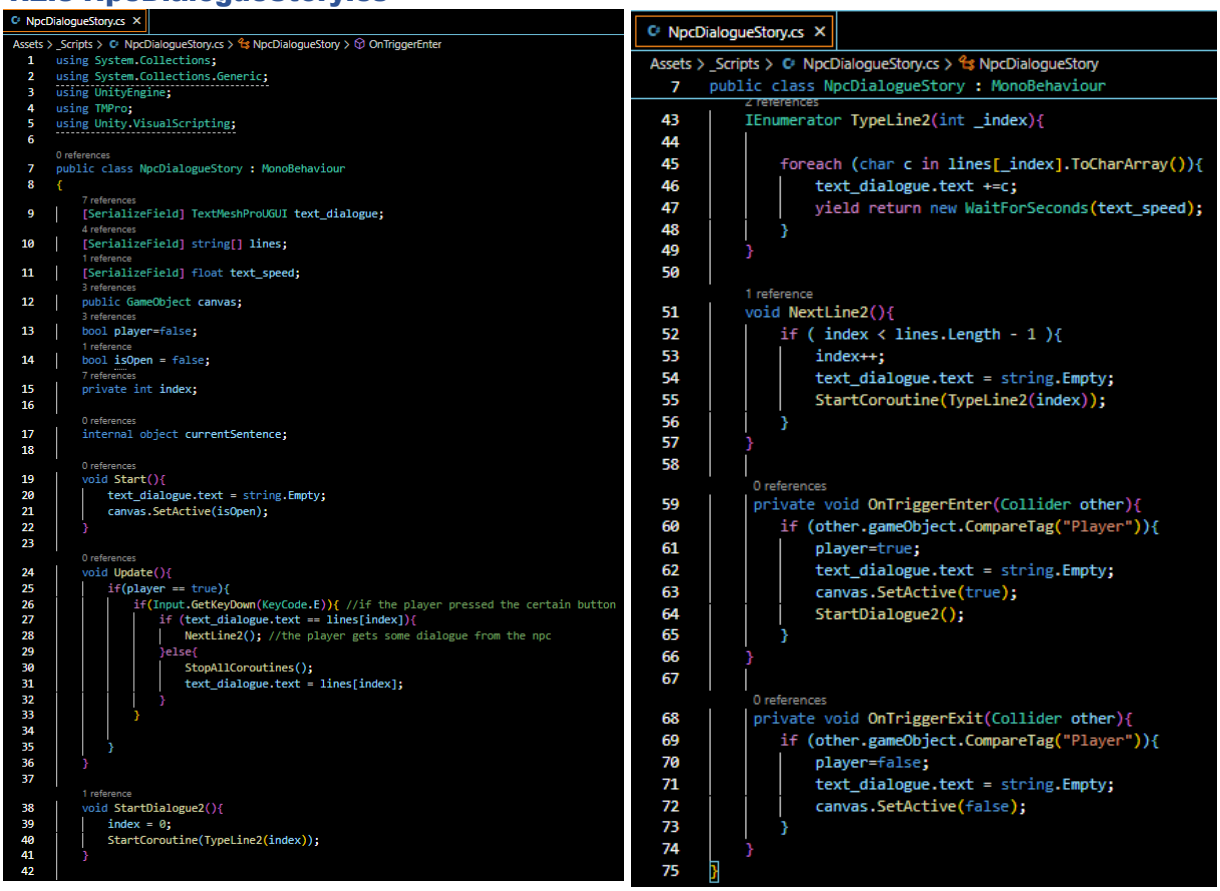
Assets > _Scripts > LookAtTheCamera.cs > ...

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4
    0 references
5   public class LookAtTheCamera : MonoBehaviour
6   {
        1 reference
7       public Transform cam;
        0 references
8       void LateUpdate()
9       {
10          transform.LookAt(cam);
11      }
12  }
```

FIGURE 14: LOOKATTHECAMERA.CS

Στον παραπάνω κώδικα διαχειρίζεται το αντικείμενο να βλέπει πάντα στην πλευρά της κάμερας που έχει ο χρήστης.

1.2.8 NpcDialogueStory.cs



```cs
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using TMPro;
using Unity.VisualScripting;

public class NpcDialogueStory : MonoBehaviour
{
    [SerializeField] TextMeshProUGUI text_dialogue;
    [SerializeField] string[] lines;
    [SerializeField] float text_speed;
    public GameObject canvas;
    bool player=false;
    bool isOpen = false;
    private int index;

    internal object currentSentence;

    void Start(){
        text_dialogue.text = string.Empty;
        canvas.SetActive(isOpen);
    }

    void Update(){
        if(player == true){
            if(Input.GetKeyDown(KeyCode.E)){ //if the player pressed the certain button
                if (text_dialogue.text == lines[index]){
                    NextLine2(); //the player gets some dialogue from the npc
                }else{
                    StopAllCoroutines();
                    text_dialogue.text = lines[index];
                }
            }

        }
    }

    void StartDialogue2(){
        index = 0;
        StartCoroutine(TypeLine2(index));
    }

    IEnumerator TypeLine2(int _index){

        foreach (char c in lines[_index].ToCharArray()){
            text_dialogue.text +=c;
            yield return new WaitForSeconds(text_speed);
        }
    }

    void NextLine2(){
        if ( index < lines.Length - 1 ){
            index++;
            text_dialogue.text = string.Empty;
            StartCoroutine(TypeLine2(index));
        }
    }

    private void OnTriggerEnter(Collider other){
        if (other.gameObject.CompareTag("Player")){
            player=true;
            text_dialogue.text = string.Empty;
            canvas.SetActive(true);
            StartDialogue2();
        }
    }

    private void OnTriggerExit(Collider other){
        if (other.gameObject.CompareTag("Player")){
            player=false;
            text_dialogue.text = string.Empty;
            canvas.SetActive(false);
        }
    }
}
```

FIGURE 15: NPCDIALOGUESTORY.CS

Στον παραπάνω κώδικα διαχειρίζομαι την εμφάνιση του κείμενου του διαλόγου του NPC και τον τρόπο με τον οποίο εξελίσσεται ο διάλογος μεταξύ του χρήστη με tag player όταν πατάει ένα καθορισμένο κουμπί (E). Τα λόγια που εμφανίζονται έχουν καθορισμένη ταχύτητα και μπορούν να εμφανίζονται ολόκληρα όταν πατηθεί το κουμπί (E).

1.2.9 PauseMenu.cs

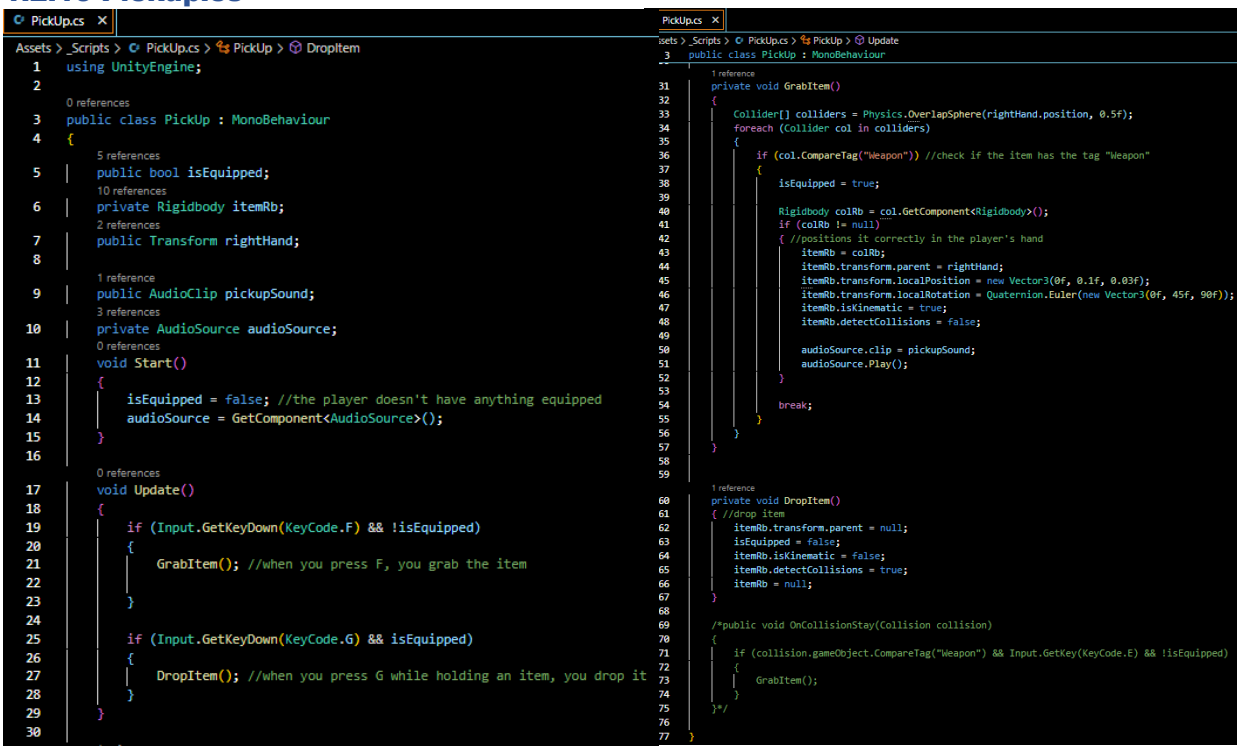
```cs
using UnityEngine;
using UnityEngine.SceneManagement;

public class PauseMenu : MonoBehaviour
{
    public static bool GameIsPaused = false;
    public GameObject pauseMenuUI;
    public GameObject playerObject;

    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Escape)) //when you press "Esc", your game pauses or resumes, depending
        {                                     // on the current state
            if (GameIsPaused)
            {
                Resume();
            }
            else
            {
                Pause();
            }
        }
    }

    public void Resume()
    {
        pauseMenuUI.SetActive(false); //disable the Pause Menu UI
        Time.timeScale = 1f; //time is going normally
        GameIsPaused = false; //the game is not paused

        playerObject.SetActive(true); //the player is visible
    }

    public void Pause()
    {
        pauseMenuUI.SetActive(true); //shows the Pause Menu UI
        Time.timeScale = 0f; //time is stopped
        GameIsPaused = true;

        playerObject.SetActive(false);
    }
```

```cs
public class PauseMenu : MonoBehaviour

    public void LoadMenu()
    {
        SceneManager.LoadScene("Options Menu"); //loads the scene Options Menu when you press "Menu"
    }

    public void QuitGame()
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex - 1); //Quits the game
    }
}
```

FIGURE 16: PAUSEMENU.CS

Στο παραπάνω κώδικα γίνεται το pause του παιχνιδιού, όπου ελέγχετε η κατάσταση του παιχνιδιού, δηλαδή αν είναι paused ή όχι. Αν το παιχνίδι είναι paused, τότε με το esc button, ενεργοποιείται η συνάρτηση Resume(). Στην συνάρτηση Resume(), ο χρόνος συνεχίζεται κανονικά και το Game δεν είναι πλέον paused. Διαφορετικά, αν το παιχνίδι δεν είναι paused, τότε γίνεται η συνάρτηση Pause(). Στην συνάρτηση Pause(), ο χρόνος σταματάει και το Game δεν είναι πλέον paused. Αν ο χρήστης επιθυμεί να δει τα settings, τότε τον παίρνω σε άλλο scene, στο οποίο μπορεί να δει τα settings του παιχνιδιού. Επίσης, ο χρήστης έχει την ευκαιρία να κάνει quit το παιχνίδι με το πάτημα του κουμπιού "Quit".

1.2.10 Pickup.cs



```
Assets > _Scripts > PickUp.cs > PickUp > DropItem
using UnityEngine;

0 references
public class PickUp : MonoBehaviour
{
    5 references
    public bool isEquipped;
    10 references
    private Rigidbody itemRb;
    2 references
    public Transform rightHand;

    1 reference
    public AudioClip pickupSound;
    3 references
    private AudioSource audioSource;
    0 references
    void Start()
    {
        isEquipped = false; //the player doesn't have anything equipped
        audioSource = GetComponent<AudioSource>();
    }

    0 references
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.F) && !isEquipped)
        {
            GrabItem(); //when you press F, you grab the item

        }

        if (Input.GetKeyDown(KeyCode.G) && isEquipped)
        {
            DropItem(); //when you press G while holding an item, you drop it
        }
    }
```

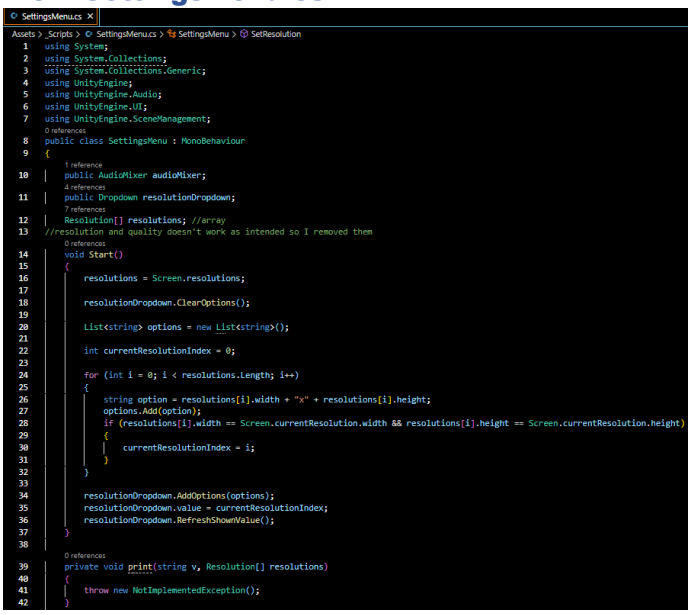
```
    1 reference
    private void GrabItem()
    {
        Collider[] colliders = Physics.OverlapSphere(rightHand.position, 0.5f);
        foreach (Collider col in colliders)
        {
            if (col.CompareTag("Weapon")) //check if the item has the tag "Weapon"
            {
                isEquipped = true;

                Rigidbody colRb = col.GetComponent<Rigidbody>();
                if (colRb != null)
                { //positions it correctly in the player's hand
                    itemRb = colRb;
                    itemRb.transform.parent = rightHand;
                    itemRb.transform.localPosition = new Vector3(0f, 0.1f, 0.03f);
                    itemRb.transform.localRotation = Quaternion.Euler(new Vector3(0f, 45f, 90f));
                    itemRb.isKinematic = true;
                    itemRb.detectCollisions = false;

                    audioSource.clip = pickupSound;
                    audioSource.Play();
                }

                break;
            }
        }
    }


    1 reference
    private void DropItem()
    { //drop item
        itemRb.transform.parent = null;
        isEquipped = false;
        itemRb.isKinematic = false;
        itemRb.detectCollisions = true;
        itemRb = null;
    }

    /*public void OnCollisionStay(Collision collision)
    {
        if (collision.gameObject.CompareTag("Weapon") && Input.GetKey(KeyCode.E) && !isEquipped)
        {
            GrabItem();
        }
    }*/
}
```

FIGURE 17: PICKUP.CS

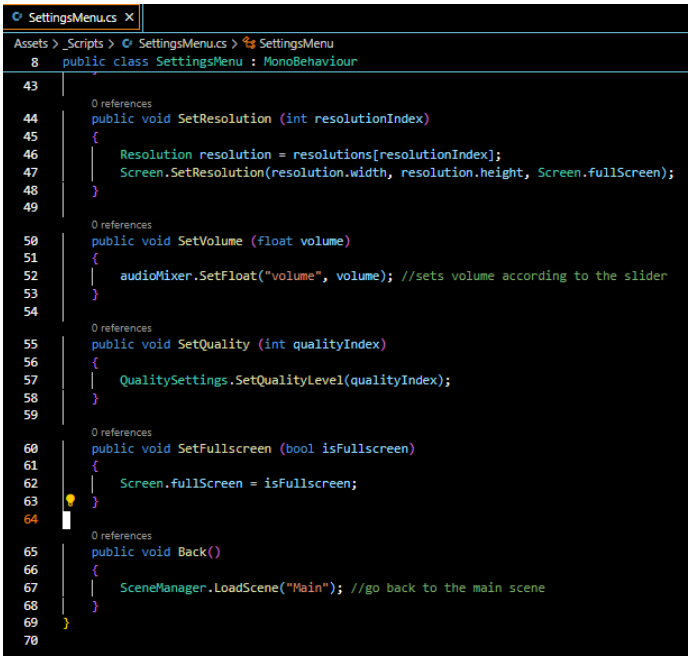
Με τον κώδικα αυτό, ο χρήστης μπορεί να παίρνει στο χέρι του αντικείμενα που βρίσκει στο έδαφος ή στα σεντούκια και έχει την δυνατότητα να χρησιμοποιήσει. Επίσης, αν δεν θέλει κάποιο αντικείμενο, τότε δικαιούται να το αφήσει στο έδαφος.

1.3 Settings

1.3.1 SettingsMenu.cs



```
SettingsMenu.cs
Assets > _Scripts > SettingsMenu.cs > SettingsMenu > SetResolution
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Audio;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
0 references
public class SettingsMenu : MonoBehaviour
{
    1 reference
    public AudioMixer audioMixer;
    4 references
    public Dropdown resolutionDropdown;
    7 references
    Resolution[] resolutions; //array
//resolution and quality doesn't work as intended so I removed them
    0 references
    void Start()
    {
        resolutions = Screen.resolutions;

        resolutionDropdown.ClearOptions();

        List<string> options = new List<string>();

        int currentResolutionIndex = 0;

        for (int i = 0; i < resolutions.Length; i++)
        {
            string option = resolutions[i].width + "x" + resolutions[i].height;
            options.Add(option);
            if (resolutions[i].width == Screen.currentResolution.width && resolutions[i].height == Screen.currentResolution.height)
            {
                currentResolutionIndex = i;
            }
        }

        resolutionDropdown.AddOptions(options);
        resolutionDropdown.value = currentResolutionIndex;
        resolutionDropdown.RefreshShownValue();
    }

    0 references
    private void print(string v, Resolution[] resolutions)
    {
        throw new NotImplementedException();
    }
```



```
SettingsMenu.cs
Assets > _Scripts > SettingsMenu.cs > SettingsMenu
8   public class SettingsMenu : MonoBehaviour

    0 references
    public void SetResolution (int resolutionIndex)
    {
        Resolution resolution = resolutions[resolutionIndex];
        Screen.SetResolution(resolution.width, resolution.height, Screen.fullScreen);
    }

    0 references
    public void SetVolume (float volume)
    {
        audioMixer.SetFloat("volume", volume); //sets volume according to the slider
    }

    0 references
    public void SetQuality (int qualityIndex)
    {
        QualitySettings.SetQualityLevel(qualityIndex);
    }

    0 references
    public void SetFullscreen (bool isFullscreen)
    {
        Screen.fullScreen = isFullscreen;
    }

    0 references
    public void Back()
    {
        SceneManager.LoadScene("Main"); //go back to the main scene
    }
}
```

FIGURE 18: SETTINGSMENU.CS

Στον κώδικα για τις ρυθμίσεις, ο χρήστης έχει την δυνατότητα να ορίσει την ένταση του παιχνιδιού όσο επιθυμεί. Επιπρόσθετα, μπορεί να έχει τον παιχνίδι σε fullscreen ή όχι, ανάλογα με την προτίμηση του. Όταν τελειώσει με τις αλλαγές των ρυθμίσεων, μπορεί να πάει πίσω στο PauseMenu με το πάτημα του κουμπιού "Back."

Κεφάλαιο 2: Animator

2.1 Main Character

Ο Main Character έχει την δυνατότητα να περπατήσει, πηδήξει, να τρέξει και να είναι ακίνητος. Σύμφωνα με τον παρακάτω Animator, ο χρήστης πάντα καταλήγει στο Idle Position του όταν τελειώσει με κάποιο animation.

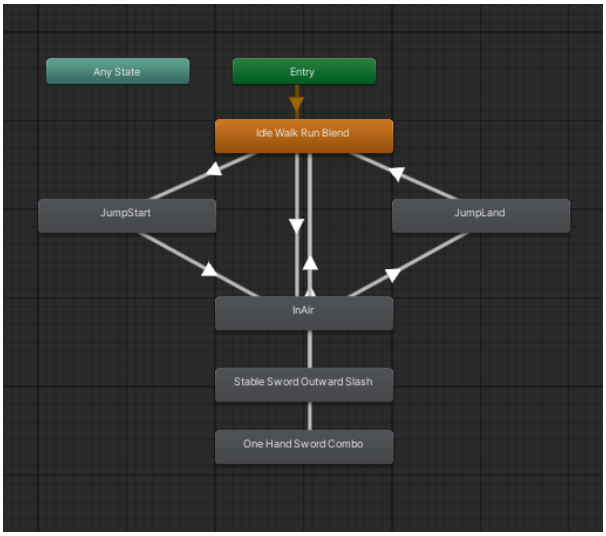


FIGURE 19: MAIN CHARACTER ANIMATION

2.2 Monsters

Τα τέρατα έχουν την δυνατότητα να κάνουν react όταν ο χρήστης πάει κοντά, να χτυπήσουν τον χρήστη, να πεθάνουν, να τρέξουν ή να μην κάνουν τίποτα (idle animation). Τα τέρατα, όπως και ο Main Character, πάντα καταλήγουν στο Idle animation του όταν τελειώνουν με το current animation.

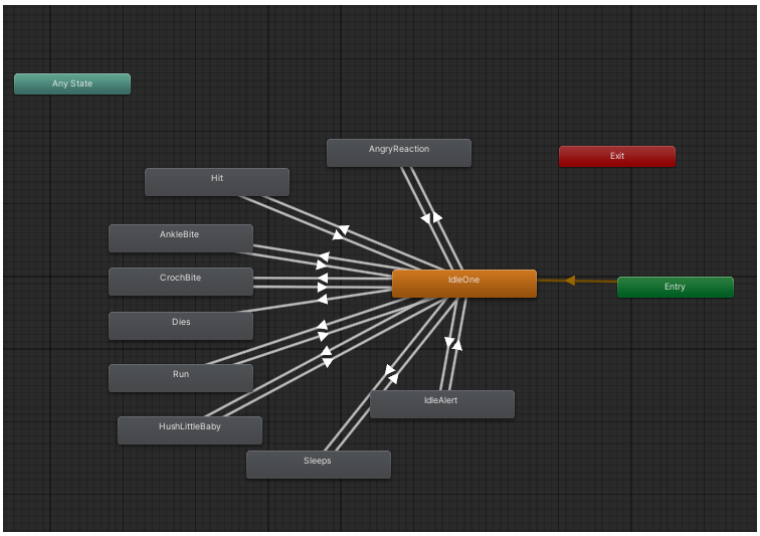


FIGURE 20: MONSTERS ANIMATOR

2.3 NPCs

Το κάθε NPC μπορεί να χαιρετήσει, να χορέψει ή να χειροκροτήσει.

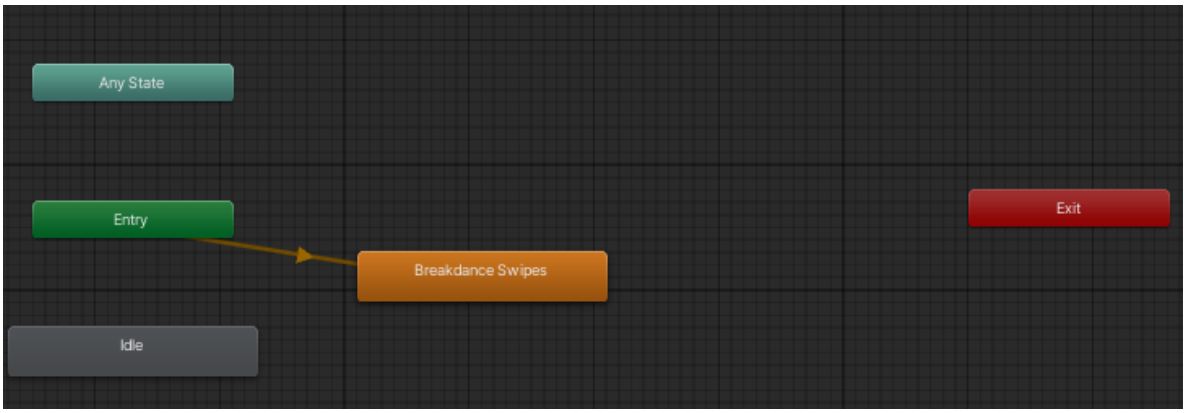


FIGURE 21: MOUSE ANIMATOR

FIGURE 22: ARCHER ANIMATOR

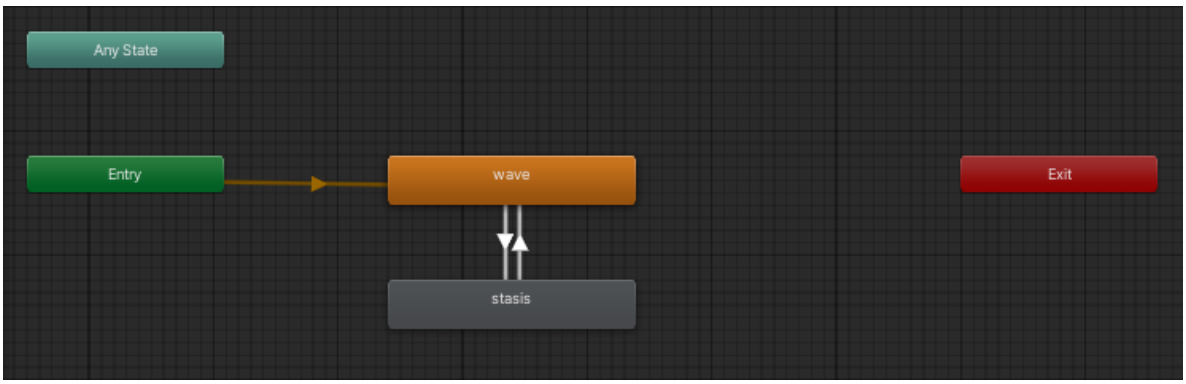


FIGURE 23: PILOT ANIMATOR

2.4 Doors

Οι πόρτες έχουν την δυνατότητα να ανοίξουν και να κλείσουν με το πάτημα του κουμπιού “E”.

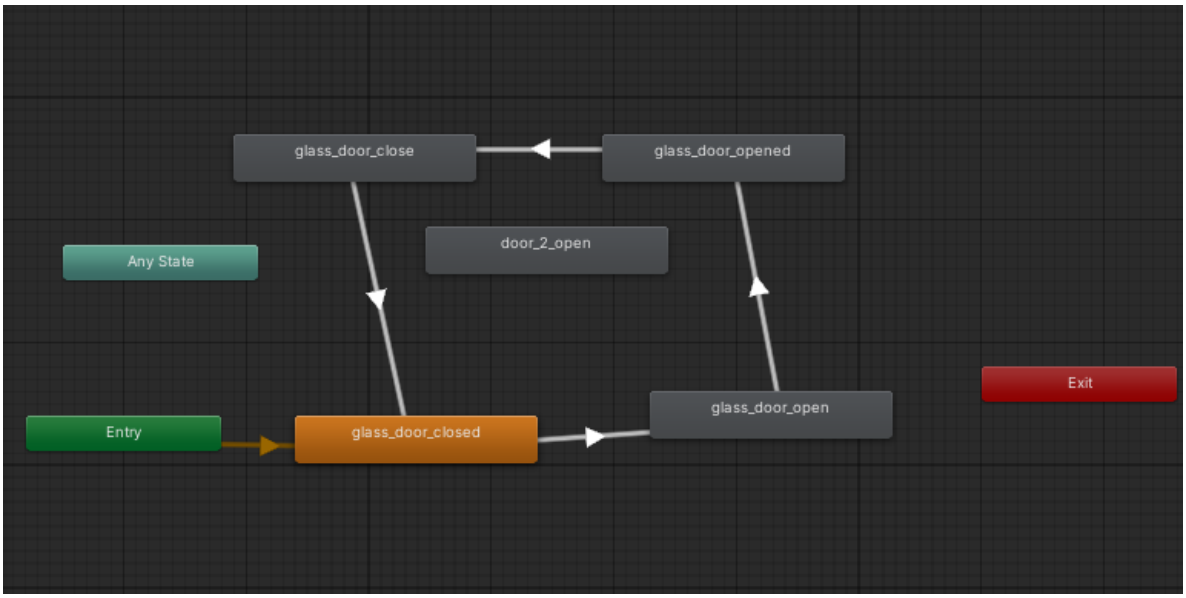


FIGURE 24: DOORS ANIMATOR

Κεφάλαιο 3: Παράδειγμα Σωστής Λειτουργίας

Αρχικό μενού, δηλαδή το τι βλέπει ο χρήστης μόλις ανοίξει το παιχνίδι

FIGURE 25: START MENU

Πατώντας το Play, βρίσκεται μέσα στον κόσμο στο αρχικό σημείο του κόσμου και της εξερεύνησής του

FIGURE 26: SPAWNED

Το πρώτο μήνυμα που παίρνει ο χαρακτήρας

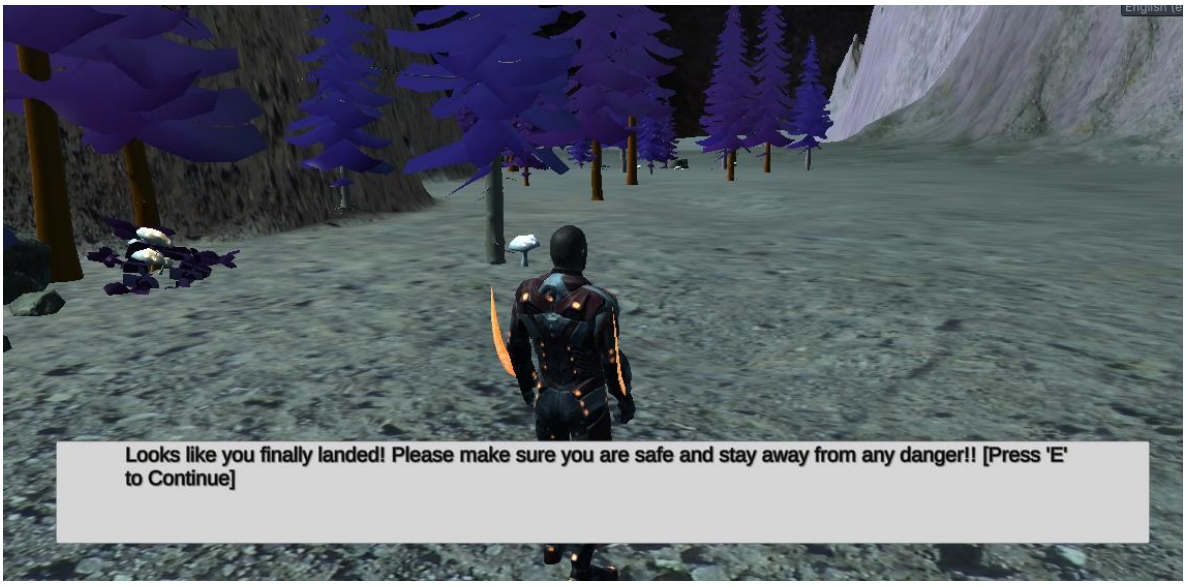


FIGURE 27: FIRST DIALOGUE

Εδώ βλέπουμε το NPC να μας χαιρετά

FIGURE 28: FIRST NPC INTERACTION

Στα αριστερά βρίσκεται το σεντούκι, όπου υπάρχει μέσα το σπαθί

FIGURE 29: CHEST OPEN AND COLLECT

Ο χρήστης μπορεί να πάρει το σπαθί

FIGURE 30: SWORD PICKUP

Εδώ βλέπουμε τον πρώτο εχθρό

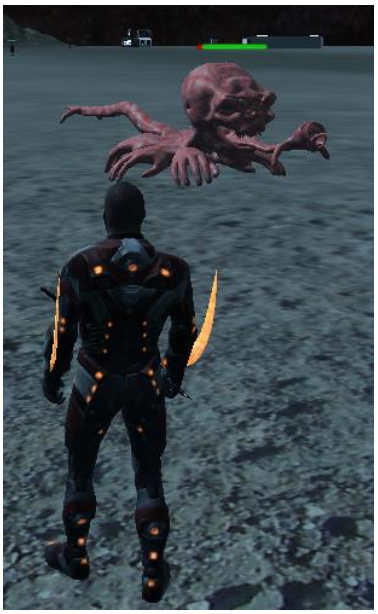


FIGURE 31: FIRST ENEMY

Όταν ο εχθρός πεθάνει, εμφανίζεται αίμα

FIGURE 32: BLOOD AFTER KILL

Παίρνουμε την πρώτη μας προειδοποίηση

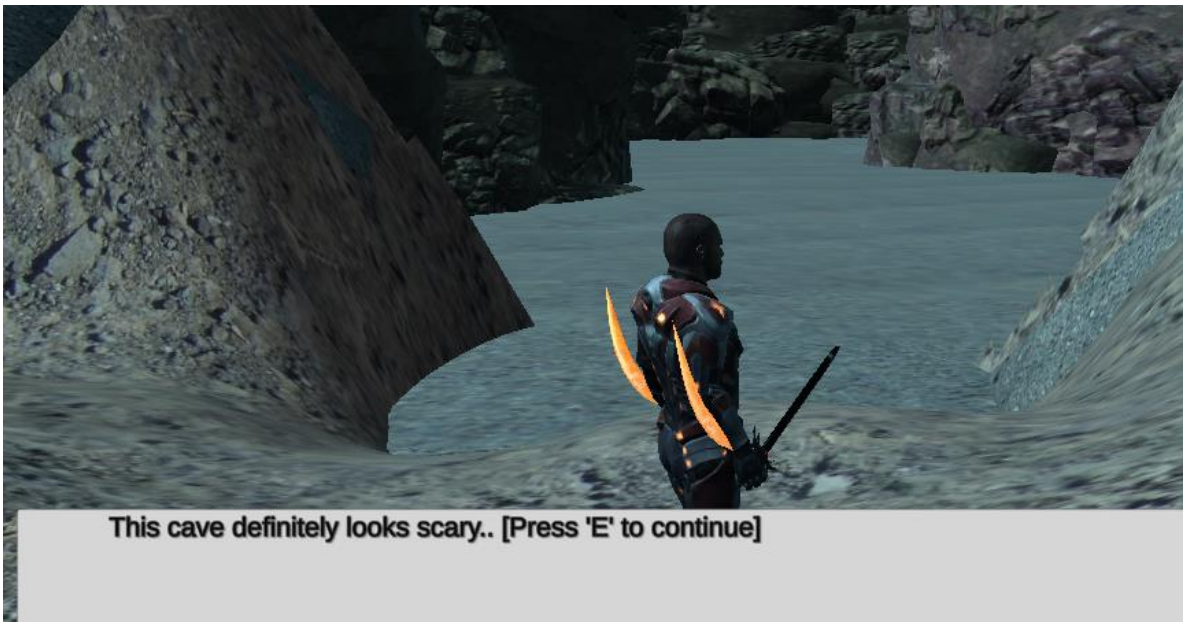


FIGURE 33: FIRST WARNING

Στο βάθος αριστερά μπορούμε να προσέξουμε ένα πολύ ισχυρό εχθρό

FIGURE 34: DANGEROUS ENEMY

Μπορούμε επίσης να βρούμε τα άτομα που ζουν στον πλανήτη αυτό και να χορέψουμε μαζί τους με το κουμπί “1” στο πληκτρολόγιο

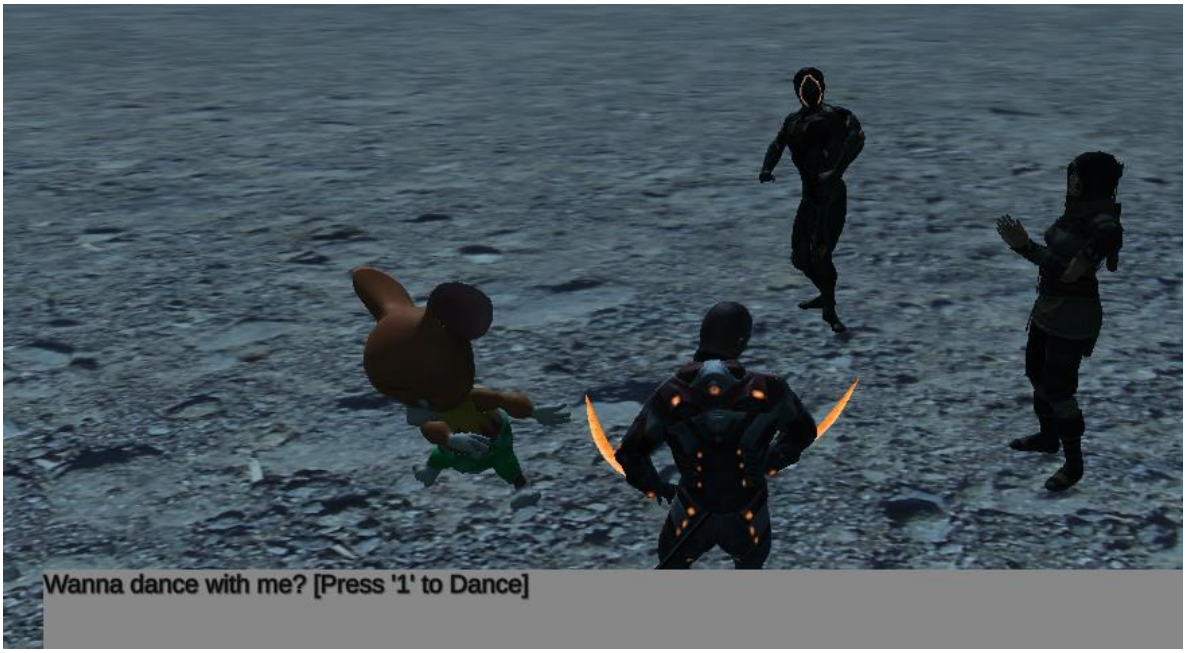


FIGURE 35: HOW TO DANCE

Το χωριό μέσα στον πλανήτη, εδώ μπορούμε να βρούμε κάποια NPC και να μιλήσουμε μαζί τους

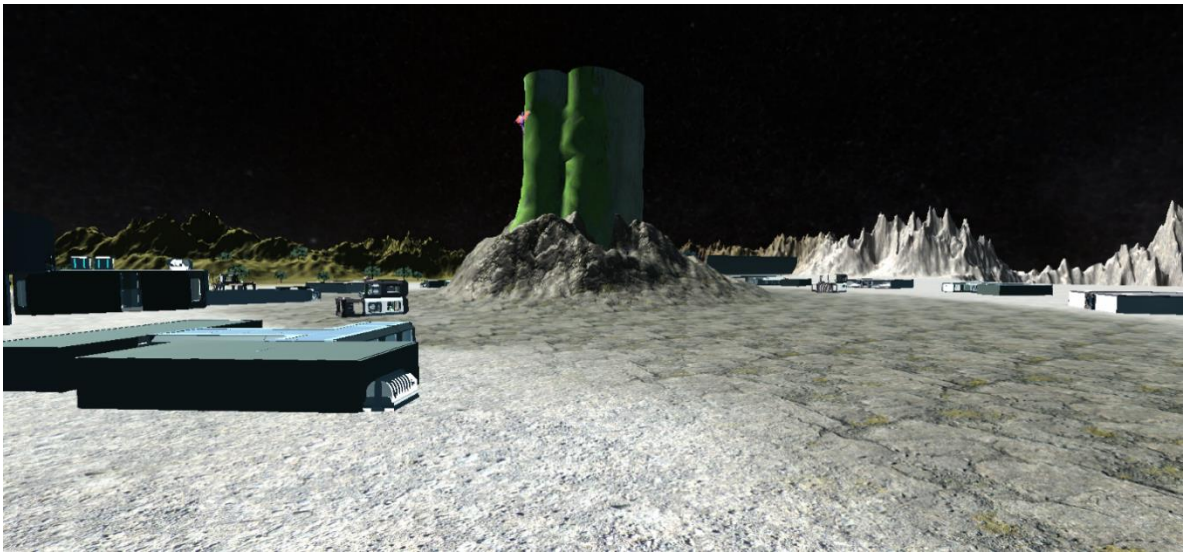


FIGURE 36: VILLAGE

Κάποια από τα NPC που μπορεί ο χρήστης να μιλήσει

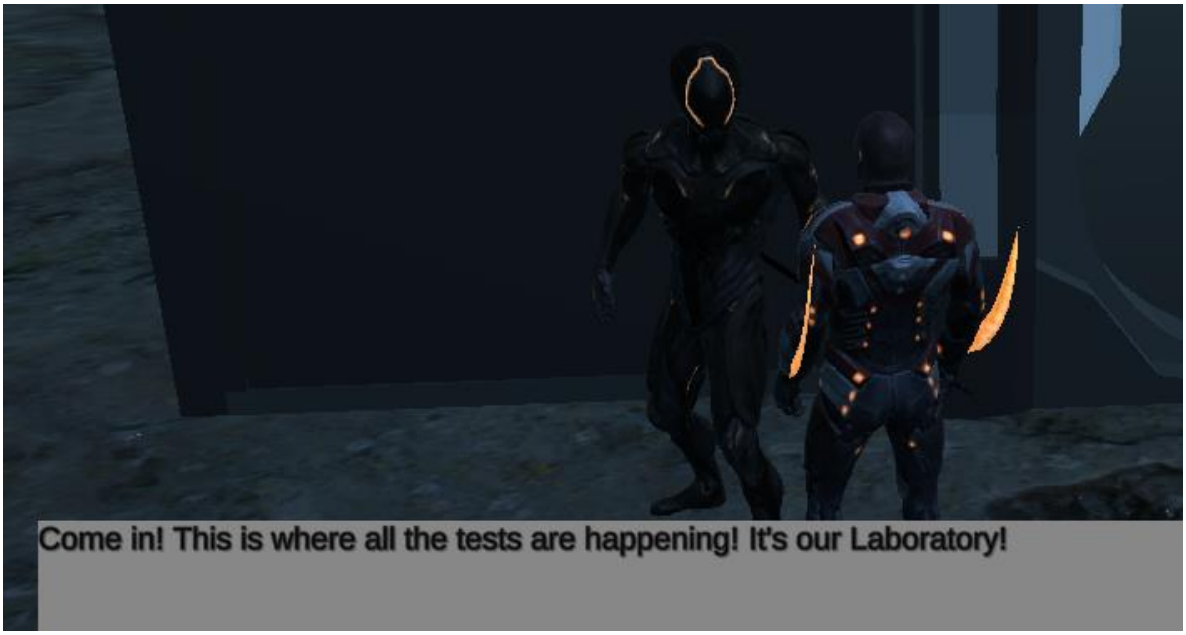


FIGURE 37: NPC INTERACTION

Είσοδος στο δάσος με τα τέρατα με την σχετική προειδοποίηση

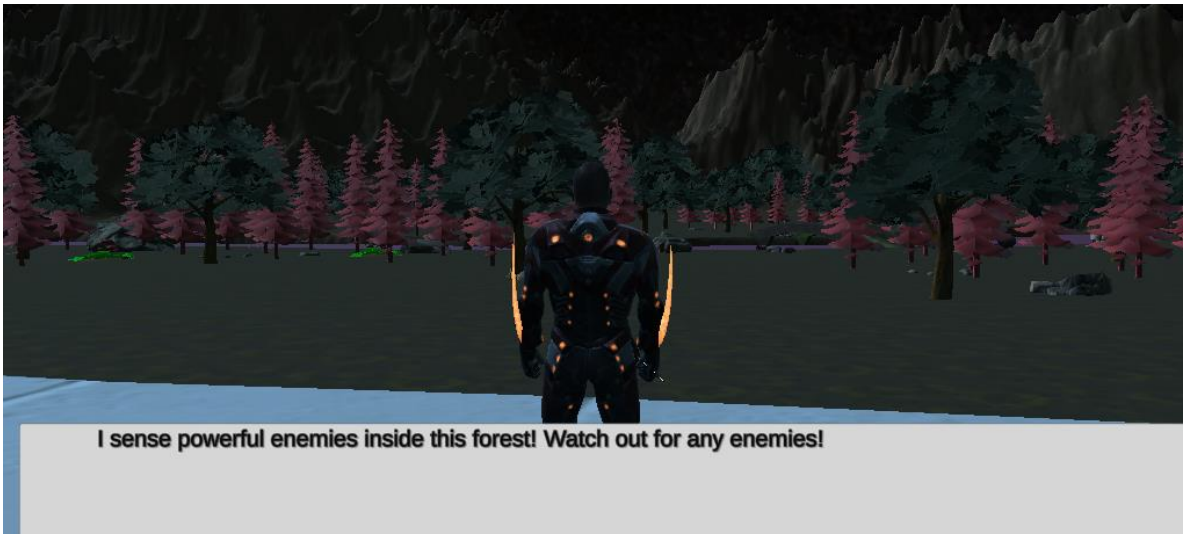


FIGURE 38: FOREST WARNING

Τέρατα στο δάσος

FIGURE 39: FOREST MONSTERS

Διαφορά της μέρας με την νύχτα:

Μέρα

FIGURE 40: DAY

FIGURE 41: NIGHT

Εδώ έχουμε τις ρυθμίσεις

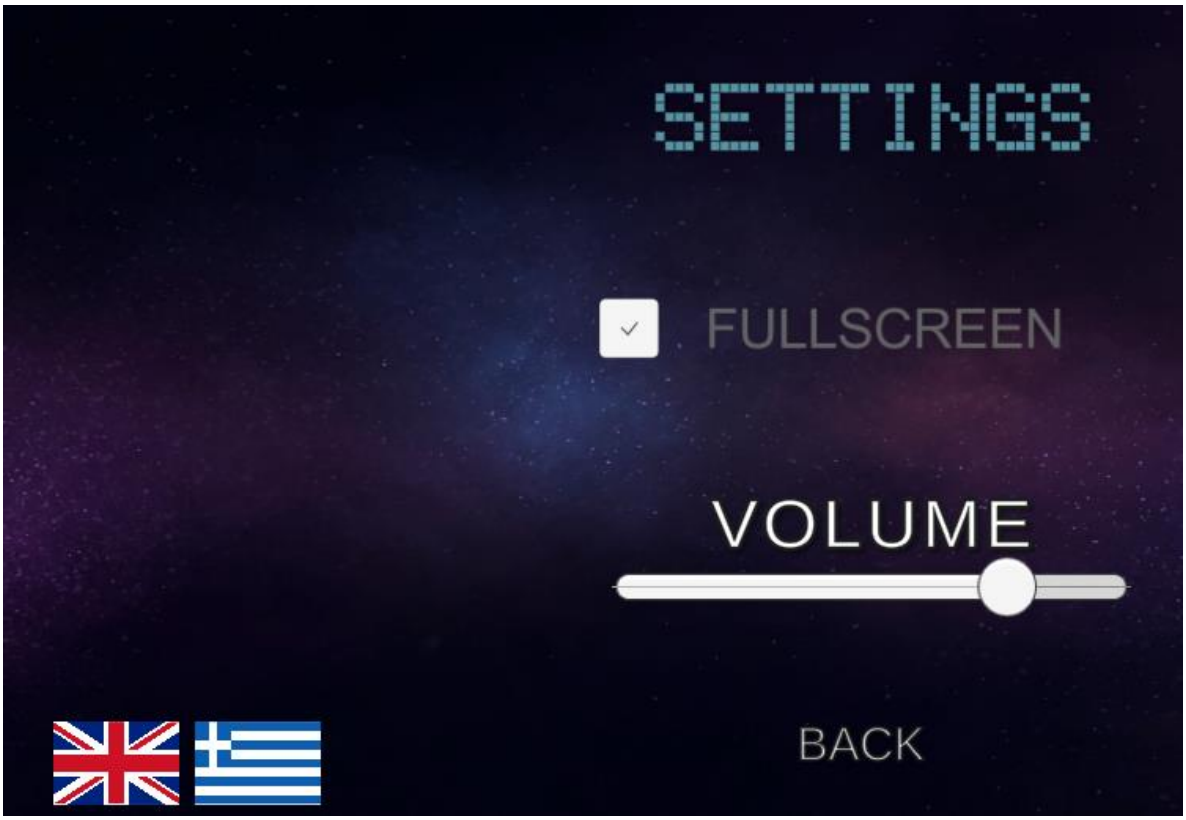


FIGURE 42: SETTINGS

Κεφάλαιο 4: Συμπεράσματα και Μελλοντικές Επεκτάσεις

Θα ήταν απίθανο αν υπήρχε το τέλειο παιχνίδι για όλους. Όταν γίνεται η ανάπτυξη ενός παιχνιδιού, οι developers έχουν ως στόχο την δημιουργία παιχνιδιού για ένα πλήθος ανθρώπων ή μιας συγκεκριμένης ηλικίας. Συνεπώς, το "….." δεν είναι ούτε τέλειο αλλά ούτε ιδανικό για όλους τους ανθρώπους. Αρχικά, εταιρίες που φτιάχνουν παιχνίδια, χωρίζουν τις λειτουργίες σε διάφορες ομάδες, είτε αυτό είναι το Animation, τα Scripts ή το Optimization. Αδιαμφισβήτητα, η κάθε ομάδα έχει την δική της λειτουργία που ασχολείται, έτσι ώστε το παιχνίδι να είναι όσο καλύτερο γίνεται στον κάθε τομέα. Δηλαδή, η ανάπτυξη ενός παιχνιδιού είναι πολύ ενδιαφέρον και διασκεδαστικό αλλά με μία ομάδα ανθρώπων που έχουν την δυνατότητα να καλύψουν όλους τους τομείς, τότε θα μπορεί κανείς να θεωρήσει ότι έφτιαξε το "τέλειο" παιχνίδι για τα άτομα που τους αρέσει το είδος το παιχνιδιού αυτού. Επιπρόσθετα, σαν μελλοντική επέκταση, θα μπορούσα να υλοποιήσω την εφαρμογή για να έχει περισσότερες αποστολές, παραπάνω interaction με NPCs καθώς και μεγαλύτερη επέκταση του χάρτη, όπου ο χρήστης θα μπορεί να εξερευνήσει και άλλες περιοχές αλλά και καινούργιους πλανήτες. Δεν μπορώ να αγνοήσω το γεγονός ότι τα γραφικά θα μπορούσαν να είναι ακόμη καλύτερα με την βοήθεια κάποιου ζωγράφου ο οποίος φτιάχνει 3d μοντέλα, καθώς και ένα πιο όμορφο περιβάλλον.

Συνοψίζοντας, το κάθε άτομο μπορεί να απολαύσει ένα διαφορετικό είδος παιχνιδιού, γι' αυτό δεν μπορεί κάποιος να φτιάξει ένα παιχνίδι που το απολαμβάνουν όλοι. Όμως, υπάρχουν τόσα πολλά παιχνίδια και θα υπάρξουν ακόμα περισσότερα, τα οποία θα δώσουν την ευκαιρία σε όλους τους Game Developers να δημιουργήσουν τον δικό τους, απολαυστικό κόσμο. Τα παιχνίδια γίνονται όλο καλύτερα, μαθαίνοντας από λάθη παλιών παιχνιδιών καθώς και από το τι αρέσει στο κάθε άτομο.

Πίνακας Ορολογίας

Script - Κώδικας

Scene - Σκηνές Παιχνιδιού

Main Character - Κύριος Χαρακτήρας

Interaction - Αλληλεπίδραση

Combat - Μάχη

Game Development - Ανάπτυξη Παιχνιδιού

Screenshot - Στιγμιότυπο

Gradient - Βαθμίδα

NPC (Non-Playable Character) - Παίκτης που δεν παίζεται

Fullscreen - Πλήρης Οθόνη

Βιβλιογραφία

https://assetstore.unity.com

https://unity.com

https://sites.google.com/view/sovolosvrcourse/