Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Πληροφορική»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Σχεδιασμός και ανάπτυξη ενός 2D παιχνιδιού ρόλων (RPG) σε πλατφόρμα Unity** **Design and development of 2D role-playing game (RPG) on the Unity platform**
Ονοματεπώνυμο Φοιτητή	**Νικόλαος Δέγγλερης**
Πατρώνυμο	**Παναγιώτης**
Αριθμός Μητρώου	**ΜΠΠΛ18015**
Επιβλέπων	**Θεμιστοκλής Παναγιωτόπουλος, Καθηγητής**

Ημερομηνία Παράδοσης **Δεκέμβριος 2022**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Θ. Παναγιωτόπουλος Καθηγητής	Δ. Σωτηρόπουλος Επίκουρος Καθηγητής	Ι. Τασούλας Επίκουρος Καθηγητής

Κατάλογος Εικόνων

Κατάλογος Πινάκων

Περίληψη

Στόχος της παρούσας μεταπτυχιακής διατριβής είναι ο σχεδιασμός και η ανάπτυξη ενός 2D παιχνιδιού ρόλων (Role-Playing Games, RPGs) με τη χρήση της γλώσσας προγραμματισμού C# στο περιβάλλον του Microsoft Visual Studio 2017 και της πλατφόρμας Unity v.2020.3.26f1. Τα εργαλεία της πλατφόρμας που χρησιμοποιήθηκαν ήταν μηδενικού κόστους. Το παιχνίδι της παρούσας μελέτης δημιουργήθηκε για να μπορεί να εκτελεστεί σε υπολογιστή, έχοντας ως πηγή έμπνευσης το παγκοσμίου φήμης παιχνίδι Super Mario Bros, στο οποίο ο Super Mario αντίστοιχα προσπαθεί να σώσει την πριγκίπισσα Peach που απείχθει από τον Bowser. Επισημαίνεται ότι το παιχνίδι δημιουργήθηκε για ακαδημαϊκό σκοπό στα πλαίσια μεταπτυχιακής διατριβής, αφήνοντας περιθώρια βελτιστοποίησης και δημιουργίας επιπλέον σκηνών στο μέλλον.

Abstract

The aim of this master's thesis is the design and development of a 2D role-playing game (Role-Playing Games, RPGs) using the C# programming language in the environment of Microsoft Visual Studio 2017 and the Unity platform v.2020.3.26f1. The used platform tools were zero cost. The game of the present study was created to be run on a computer, inspired by the world-famous game Super Mario Bros., in which Super Mario tries to rescue Princess Peach who is abhorred by Bowser. It should be noted that the game was created for an academic purpose in the context of a master's thesis, leaving space for optimization and creation of additional scenes in the future.

Περιεχόμενα

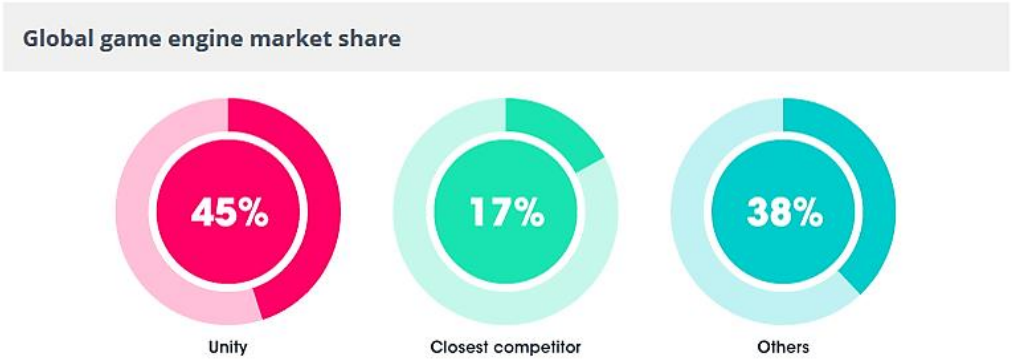
1. Εισαγωγή

1.1. Μηχανή παιχνιδιών Unity

Η πλατφόρμα Unity αποτελεί μια δημοφιλή μηχανή παιχνιδιών και ένα ολοκληρωμένο περιβάλλον ανάπτυξης (Integrated Development Environment, IDE) για τη δημιουργία διαδραστικών μέσων, συνήθως βιντεοπαιχνιδιών[1], που αναπτύχθηκε από την εταιρεία Unity Technologies το 2005. Αρχικός στόχος ήταν η δημιουργία μιας οικονομικής μηχανής παιχνιδιών με επαγγελματικά εργαλεία για ερασιτέχνες προγραμματιστές παιχνιδιών[1]. Για τον λόγο αυτό, σημαντικό πλεονέκτημα της Unity αποτελεί η χορήγηση δωρεάν άδειας στους χρήστες της για το βασικό λογισμικό της, οι οποίοι επιπροσθέτως έχουν τη δυνατότητα να χρησιμοποιήσουν 3D αλλά και 2D assets κατά τη δημιουργία των παιχνιδιών, μέσω του Asset Store της[2]. Η πρώτη έκδοση της Unity (1.0.0) κυκλοφόρησε στις 6 Ιουνίου του 2005 από τους David Helgason, Joachim Ante και Nicholas Francis στη Δανία, ενώ ακολούθησαν πολλαπλές τροποποιήσεις στο λογισμικό και το όνομά της μέχρι την τρέχουσα έκδοση[2] (τελευταία έκδοση κατά τη συγγραφή της παρούσας μελέτης είναι η 2022.1).

Όταν κυκλοφόρησε αρχικά, η Unity ήταν διαθέσιμη αποκλειστικά για Mac OS X και οι προγραμματιστές μπορούσαν να αναπτύξουν τις δημιουργίες τους μόνο σε λίγες πλατφόρμες[1]. Με την πάροδο του χρόνου η Unity ξεκίνησε να υποστηρίζεται και από Microsoft Windows. Η ανάπτυξη του iOS και η κυκλοφορία του App Store της Apple αποτέλεσε σημαντικό σταθμό για τη βελτιστοποίηση της Unity στα μέσα της δεκαετίας του 2000[2]. Έκτοτε, η Unity επεκτάθηκε για να υποστηρίξει την ανάπτυξη διαφόρων πλατφορμών (όπως Android, PlayStation, Steam VR, Google's ARCore, Vuforia), και άλλων λειτουργιών όπως διαφημίσεων δικτύου και ελέγχου εκδόσεων[2]. Έτσι, η Unity μπορεί να θεωρηθεί μία μηχανή γενικής χρήσης, η οποία σε αντίθεση με τις περισσότερες μηχανές παιχνιδιών δεν φιλοξενεί μόνο συγκεκριμένους τύπους ανάπτυξης λογισμικού. Παρόλα αυτά η Unity εξακολουθεί να ενθαρρύνει τους προγραμματιστές να δημιουργούν περιεχόμενο με συγκεκριμένους τρόπους. Για παράδειγμα, στην αρχή βελτιστοποιήθηκε για τη δημιουργία τρισδιάστατου (3D) περιεχομένου, έτσι ώστε για πολλά χρόνια να αναφέρεται ως «Unity3D»[2]. Η Unity έχει γραφθεί σε C και C++ και ο προγραμματισμός σε αυτή την πλατφόρμα γίνεται κυρίως με C# και Javascript. Στην παρούσα μελέτη η γλώσσα προγραμματισμού που χρησιμοποιήθηκε ήταν η C#.

Έρευνα που διεξήχθει από την Developer Economics σε πάνω από 10.000 προγραμματιστές, έδειξε ότι η πλειοψηφία (45%) του παγκόσμιου μερίδιου αγοράς των μηχανών παιχνιδιών αφορά την πλατφόρμα Unity (Εικόνα 1.1).



Εικόνα 1.1. Μερίδιο παγκόσμιας αγοράς της Unity σε σύγκριση με τις υπόλοιπες μηχανές παιχνιδιών (https://d3.harvard.edu/platform-digit/submission/unity-engine-a-unicorn-powering-the-video-game-and-vr-ar-economy/)

Παραδείγματα ορισμένων παιχνιδιών που έχουν αναπτυχθεί με την πλατφόρμα Unity περιλαμβάνουν: Thomas Was Alone (2010), Temple Run (2011), The Room (2012), RimWorld (2013), Hearthstone (2014), Kerbal Space Program (2015), Pokémon GO (2016), και Cuphead (2017)[3].

1.2. Παιχνίδια ρόλων – Role-Playing Games (RPGs)

Τα παιχνίδια ρόλων (Role-Playing Games, RPGs) αποτελούν μια κατηγορία παιχνιδιών στα οποία οι συμμετέχοντες αναλαμβάνουν το ρόλο φανταστικών χαρακτήρων και μέσω συνεργασίας δημιουργούν ή παρακολουθούν ιστορίες. Οι συμμετέχοντες καθορίζουν τις ενέργειες των χαρακτήρων τους εν μέρει βασισμένοι στον σχεδιασμό του χαρακτήρα τους, και οι ενέργειες πετυχαίνουν ή αποτυχαίνουν σύμφωνα με ένα, συνήθως πολύπλοκο, σύστημα κανόνων και οδηγιών. Στο πλαίσιο των κανόνων, οι παίκτες μπορούν να αυτοσχεδιάσουν ελεύθερα και οι επιλογές τους καθορίζουν την κατεύθυνση και την έκβαση των παιχνιδιών. Ορισμένα δημοφιλή παιχνίδια που ανήκουν στην εν λόγω κατηγορία παιχνιδιών είναι τα *Dungeons & Dragons*, το οποίο είναι το πρώτο εμπορικά διαθέσιμο RPG με έτος κυκλοφορίας το 1974, *Chrono Trigger* (1995), *World of Warcraft* (2004), *Dark Souls* (2011), *Pokémon - Red and Blue Version* (1996).

Σε αντίθεση με τα υπόλοιπα είδη παιχνιδιών που προάγουν τον ανταγωνισμό, τα RPGs προάγουν περισσότερο τη συνεργασία και την κοινωνικότητα δηλαδή τη λειτουργία της ομάδας. Για τον λόγο αυτό στο συγκεκριμένο είδος παιχνιδιών σπάνια υπάρχουν νικητές ή χαμένοι. Πριν αρχίσει μια ομάδα να παίζει ένα RPG είναι απαραίτητο οι παίκτες να δημιουργήσουν χαρακτήρες, μία διαδικασία πολύπλευρη που εξαρτάται κυρίως από τις επιλογές και τις αποφάσεις του παίκτη.

2. Σενάριο παιχνιδιού

Το παρόν παιχνίδι RPG, «*The Dungeon*», αφορά στην προσπάθεια του κεντρικού ήρωά του (Panduin) να διασώσει τον μικρό του αδελφό του (Pebblo), ο οποίος απήχθη από μοχθηρά τέρατα και βρίσκεται αιχμάλωτος σε ένα μπουντρούμι. Το «*The Dungeon*» εκτός από τη ψυχαγωγία του παίκτη προσβλέπει και στην ανάπτυξη της κριτικής του σκέψης, καθώς μέσα από αυτό καλείται να πάρει τις σωστές αποφάσεις οι οποίες θα τον οδηγήσουν στην απελευθέρωση του Pebblo. Ο παίκτης θα πρέπει να χειριστεί με στρατηγική τον Panduin, ώστε να καταφέρει να ανέβει σε επίπεδο και παράλληλα να εξελίξει τα όπλα του και να σκοτώσει τα μοχθηρά τέρατα. Το παιχνίδι έχει προγραμματιστεί έτσι ώστε όταν ο χαρακτήρας εξελίσσεται πέρα από τις μαχητικές του ικανότητες να αυξάνεται και η ζωή του, για να μπορέσει να ανταπεξέλθει στον δύσκολο και μακρινό του αγώνα.

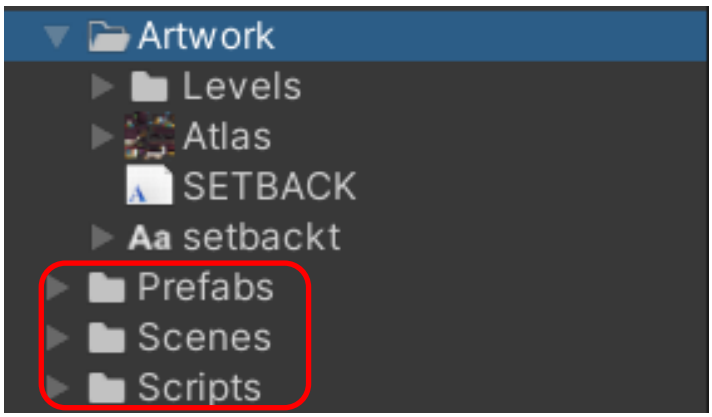
3. Υλοποίηση παιχνιδιού

3.1. Α΄ φάση: Δημιουργία σκηνών και features

Αρχικά, το κατέβασμα (download) της Unity v.2020.3.26f1 και του Microsoft Visual Studio 2017, πραγματοποιήθηκε από τους ακόλουθους συνδέσμους αντίστοιχα: https://unity.com/download και https://visualstudio.microsoft.com/downloads/.

Στήσιμο παιχνιδιού (δημιουργία Project)

Για τη σωστή οργάνωση του παιχνιδιού, στον κεντρικό φάκελο Artwork και μέσω της επιλογής Create Folder Artwork δημιουργήθηκαν οι ακόλουθοι υποφάκελοι: Prefabs, Scenes, Scripts (Εικόνα 3.1).

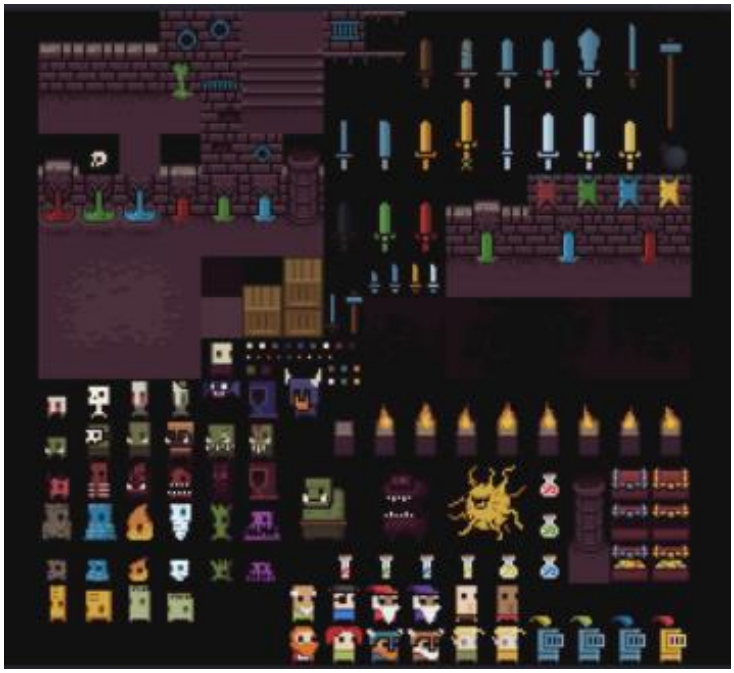


Εικόνα 3.1: Δημιουργία υποφάκελων στον κεντρικό φάκελο Artwork

Βασικά tabs για την υλοποίηση του παιχνιδιού είναι το Scene (για την προσθήκη και την αλλαγή των χαρακτηριστικών) και το Game (για την απεικόνιση του παιχνιδιού όπως θα φαίνεται και στον χρήστη).

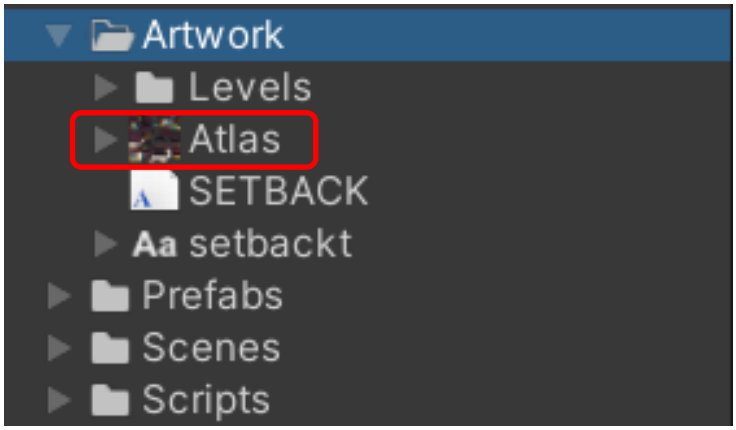
Δημιουργία Atlas

Μέσω του συνδέσμου https://0x72.itch.io/16x16-dungeon-tileset κατέβηκε η παρακάτω εικόνα (Εικόνα 3.2) που χρησιμοποιήθηκε ως βάση για τη δημιουργία των σκηνών, των χαρακτήρων και των τεράτων.



Εικόνα 3.2: Η κεντρική εικόνα που χρησιμοποιήθηκε ως βάση για τη δημιουργία των σκηνών του παιχνιδιού

Η συγκεκριμένη εικόνα αποθηκεύτηκε στον κεντρικό φάκελο Artwork με την ονομασία Atlas (Εικόνα 3.3).



Εικόνα 3.3: Δημιουργία Atlas στον κεντρικό φάκελο Artwork

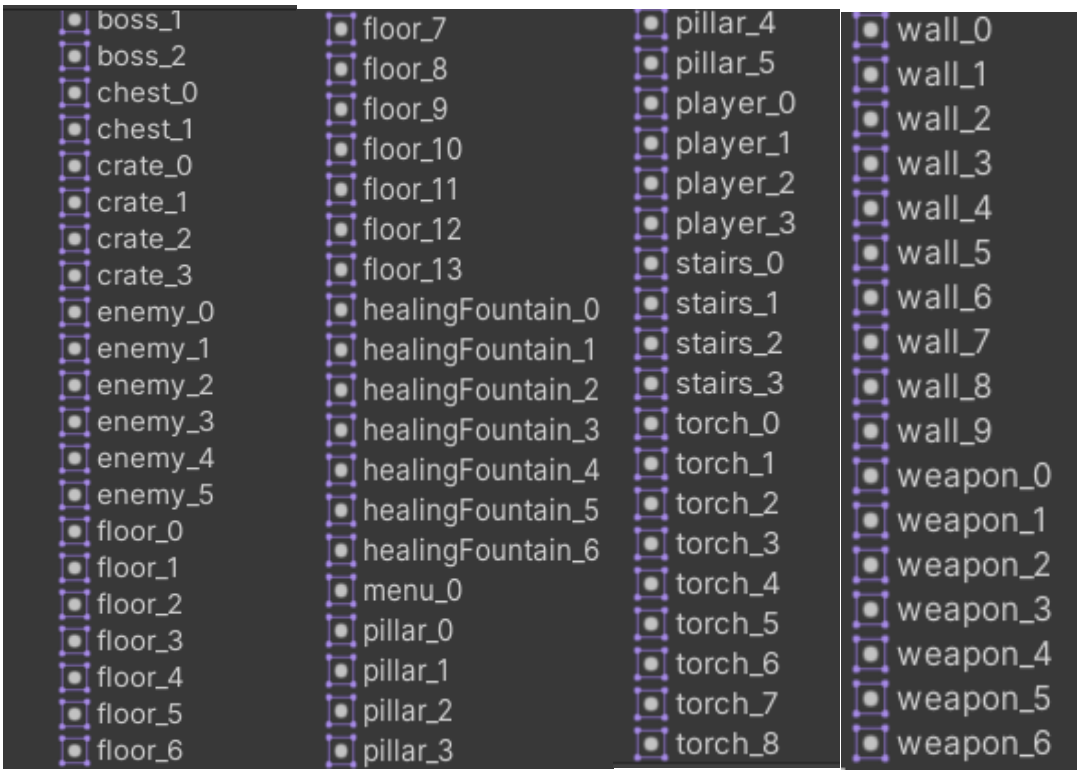
Επίσης, για την καλύτερη απεικόνιση των μηνυμάτων εντός του παιχνιδιού, χρησιμοποιήθηκε η γραμματοσειρά Setback TT BRK που κατέβηκε από τον παρακάτω σύνδεσμο: https://www.1001fonts.com/setback-tt-brk-font.html.

Δημιουργία graphics

Μετά τη δημιουργία του Atlas και μέσω της επιλογής Sprite Editor πραγματοποιήθηκε συλλογή των γραφικών του παιχνιδιού :

- Players
- NPCs
- Bosses
- Small Enemies
- Weapons
- Chests
- Healing Fountains
- Floor
- Walls
- Crates
- Stairs
- Pillars
- Portals
- Torches

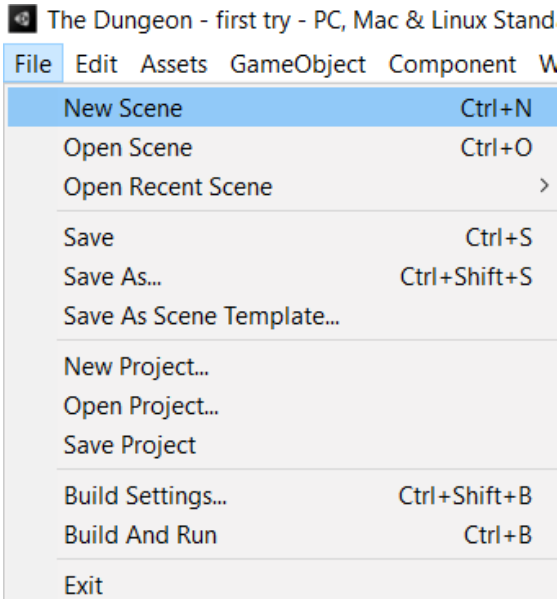
Όλα τα παραπάνω features που χρησιμοποιήθηκαν ήταν διάστασης 16x16 (Height x Width) και η ανάλυσή τους 800x600 pixels (προτεινόμενη ανάλυση για παλαιότερα παιχνίδια τύπου 2D). Με τη χρήση του tile palette (ονομάστηκε Dungeon και αποθηκεύτηκε στο Artwork Levels) αποθηκεύτηκαν όλα τα features (Εικόνα 3.4) τα οποία προστέθηκαν στο tilemap της κάθε σκηνής.



Εικόνα 3.4: Τα features των graphics

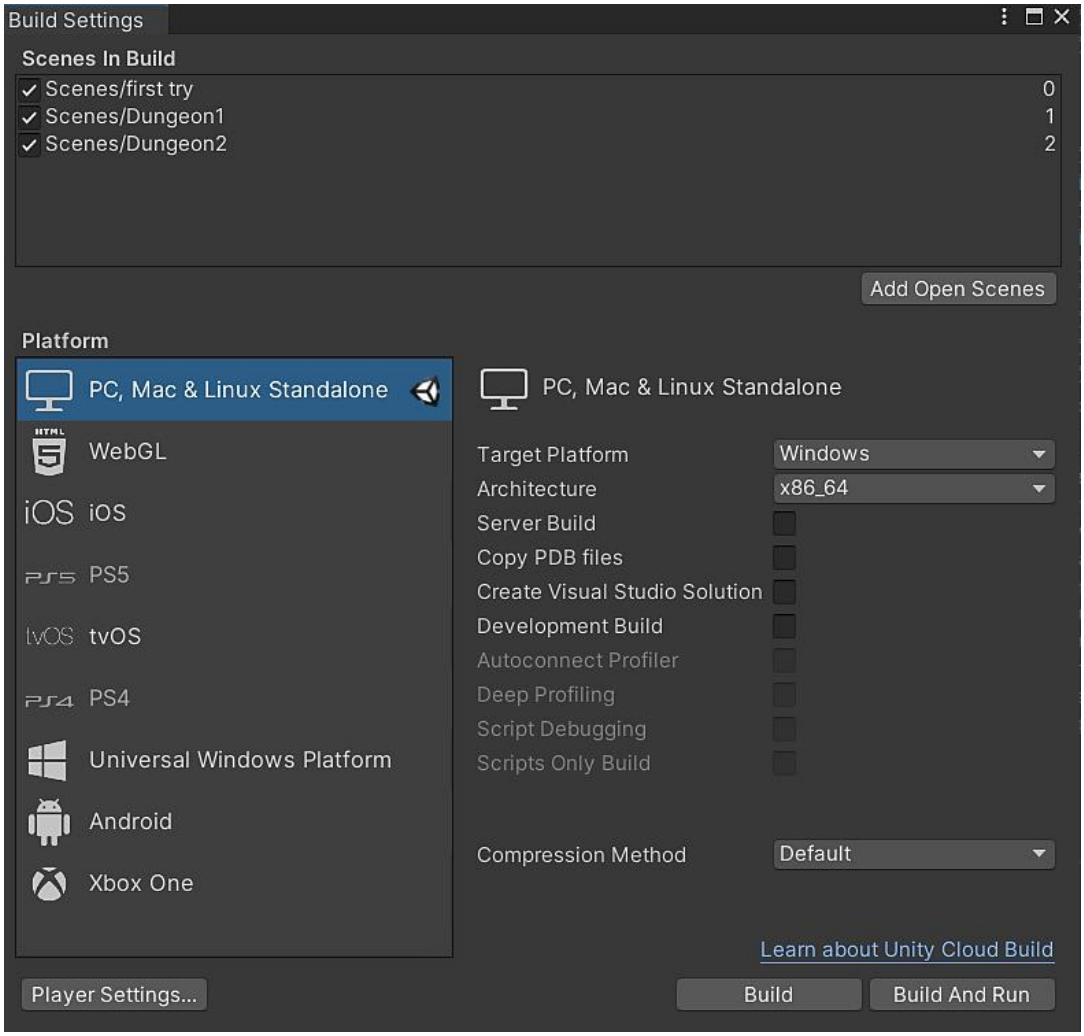
Δημιουργία σκηνών

Για τον σκοπό του παιχνιδιού δημιουργήθηκαν τρεις σκηνές (First try, Dungeon1, Dungeon2) όπως φαίνεται στην Εικόνα 3.5.



Εικόνα 3.5: Δημιουργία σκηνών

Στη σκηνή first try, ο παίκτης μαθαίνει να χειρίζεται τον χαρακτήρα και το μενού. Στη σκηνή Dungeon1, ο Panduin μονομαχεί με τα τέρατα και αφού εξελιχθεί τόσο σε levels και weapons, εισέρχεται στην τελευταία και δυσκολότερη σκηνή Dungeon2 στην οποία θα πρέπει να μονομαχήσει με τα τέρατα και να φτάσει στον Pebblo. Επίσης, οι συγκεκριμένες σκηνές θα πρέπει να ενεργοποιηθούν μέσω της Unity από την επιλογή Build Settings (Εικόνα 3.6) προκειμένου ο παίκτης να μπορεί να μεταφερθεί σε αυτές τις σκηνές (portals).

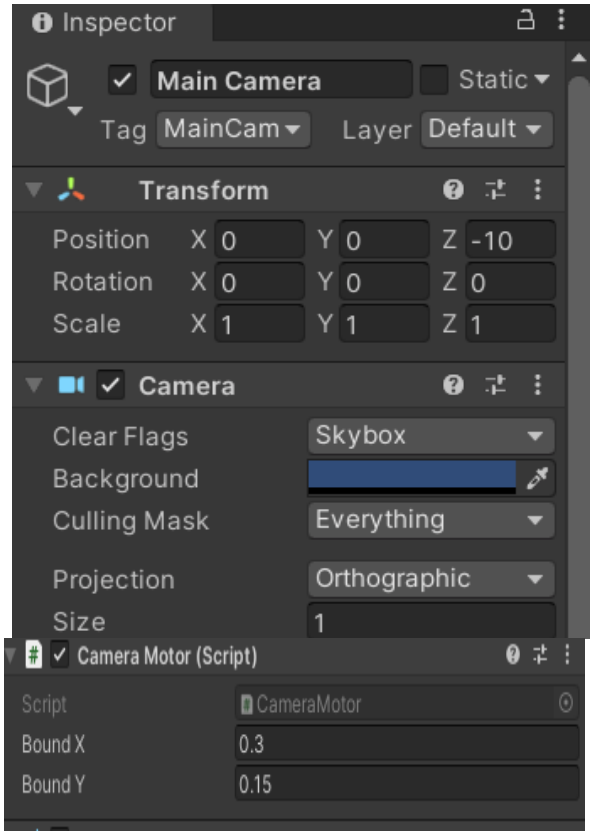


Εικόνα 3.6: Ενεργοποίηση σκηνών

Στήσιμο κάμερας

Η Main Camera ρυθμίστηκε ως εξής (Εικόνα 3.7):

1. Projection: orthographic (απεικόνιση για παιχνίδια 2D)
2. Size: 1 (100 pixels = 1 meter) για να είναι οπτικά αποδεκτό για το επικείμενο 2D game που δημιουργήθηκε
3. Δημιουργήθηκε το script CameraMotor.cs, στο οποίο ορίστηκαν το όρια του άξονα X & Y, ώστε η κάμερα να ακολουθεί μετά από κάποια απόσταση την κίνηση του παίκτη. Bound X = 0,3 & Bound Y = 0,15



Εικόνα 3.7: Ρύθμιση της Main Camera

Δημιουργία Tilemap & Layers

Για την οργάνωση των layers δημιουργήθηκε ένα κεντρικό 2D object tilemap που ονομάστηκε Grid και περιλαμβάνει τα κάτωθι tilemaps στις σκηνές που φτιάχτηκαν:

- Floor με order layer 0
- Design δηλωμένο sorting layer “floor” και με order layer 1 (περιλαμβάνει τα γραφικά του παιχνιδιού όπως chest, healing fountains κ.ά.)
- Floor-Other δηλωμένο sorting layer “floor” και με order layer 1 (περιλαμβάνει γραφικά)
- Other δηλωμένο sorting layer “floor” και με order layer 2 (περιλαμβάνει γραφικά)
- Collision δηλωμένο sorting layer “floor” και με order layer 5 (περιλαμβάνει τα όρια κάθε πίστας, που μπορεί να κινηθεί ο χαρακτήρας)

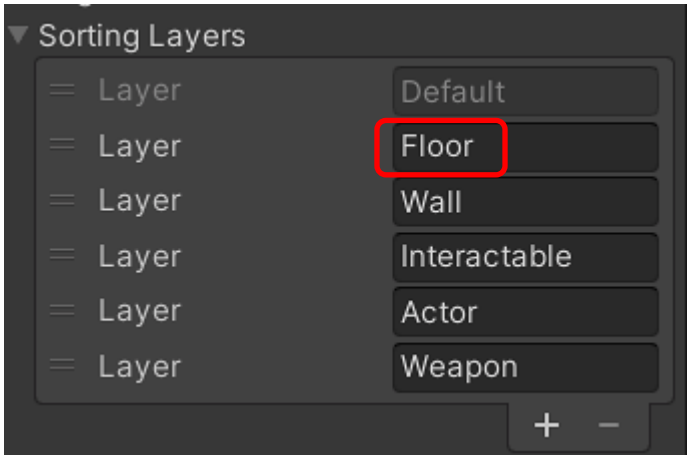
Συνολικά έχουν δημιουργηθεί 6 sorting layers (Εικόνα 3.8):

- Default
- Floor
- Wall

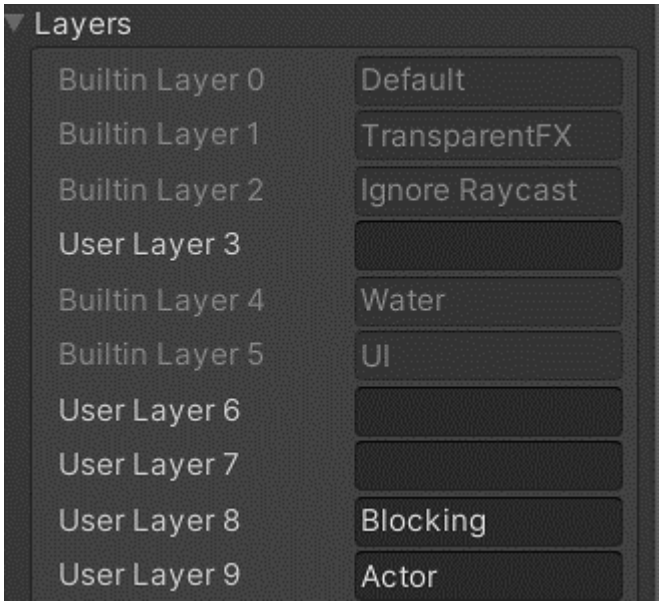
- Interactable
- Actor
- Weapon

Απώτερος σκοπός της οργάνωσης των συγκεκριμένων layers είναι το πως θα φαίνονται με προτεραιότητα μέσα στο παιχνίδι.

Επίσης έχουν δημιουργηθεί δύο κεντρικά layers, το actor και το blocking (Εικόνα 3.9). Το actor περιλαμβάνει όλους τους players, τα npc και τους enemies-bosses, ενώ το blocking αφορά τον καθορισμό των ορίων κίνησης των χαρακτήρων.

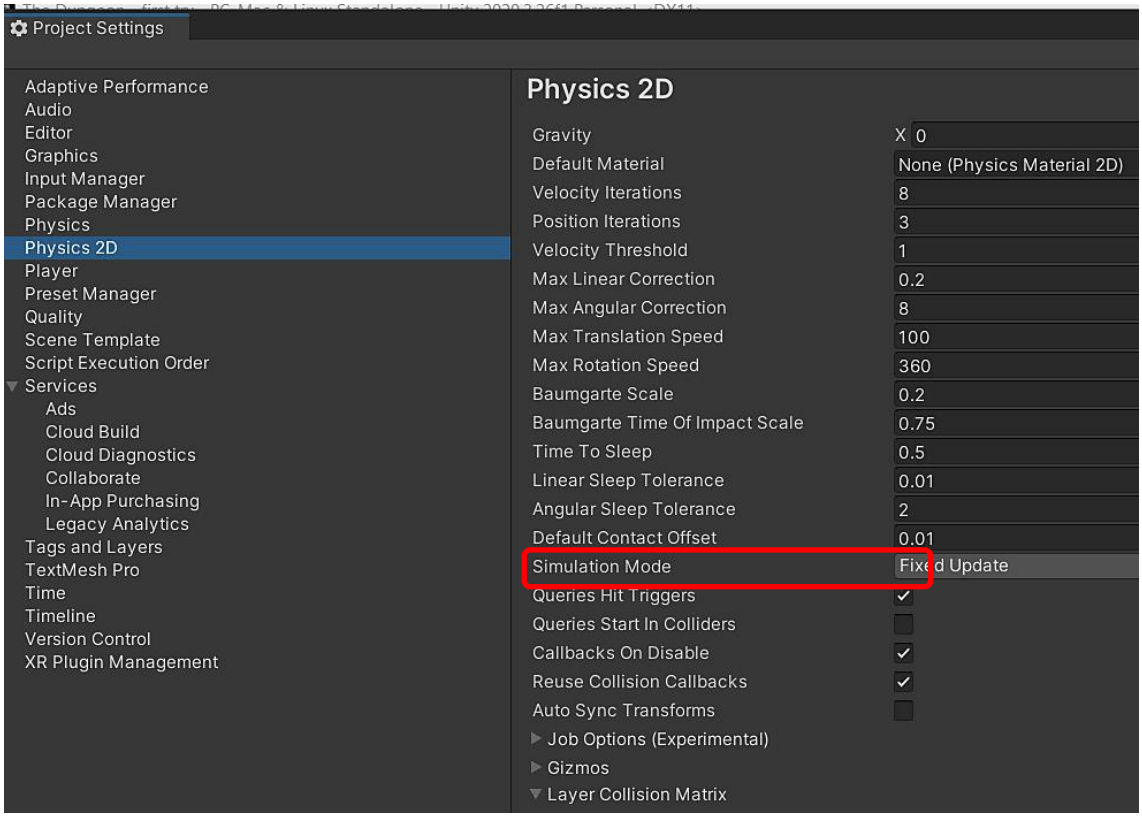


Εικόνα 3.8: Sorting Layers



Εικόνα 3.9: Κεντρικά layers

Σημειώνεται ότι για την ομαλή κίνηση του χαρακτήρα ώστε να μη μπλοκάρεται από τον ίδιο του τον εαυτό, απενεργοποιήθηκε η επιλογή Queries Start in Colliders (Εικόνα 3.10).



Εικόνα 3.10: Απενεργοποίηση της επιλογής Queries Start in Colliders

Δημιουργία Menu, Health Bar, Death Menu & Respawn

Δημιουργήθηκε ένα κεντρικό Panel (Create UI and Panel) το οποίο ονομάστηκε HUD και περιλαμβάνει τρία βασικά χαρακτηριστικά του παιχνιδιού, τα οποία είναι σημαντικά για τη λειτουργικότητα του παιχνιδιού.

Αυτά είναι το Menu, το Health Bar και το Death Menu.

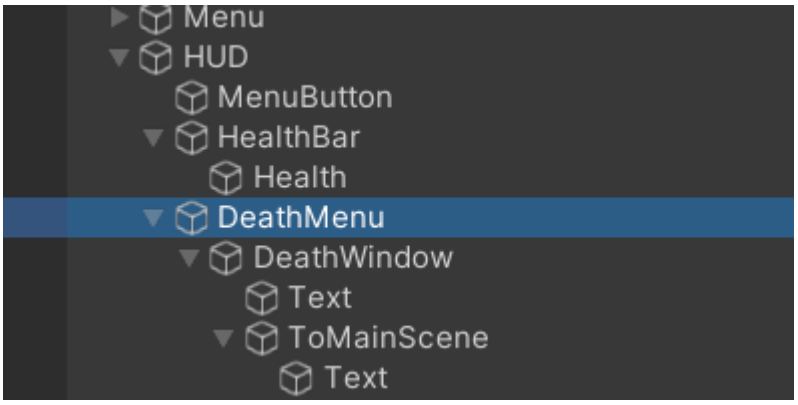
- Όταν ο παίκτης επιλέγει το εικονίδιο  , που βρίσκεται κάτω αριστερά τότε εμφανίζεται το Menu (Εικόνα 3.11) όπου ο παίκτης έχει τη δυνατότητα να εξελίξει το όπλο του, εφόσον έχει τα απαιτούμενα pesos, να αλλάξει παίκτη και να πληροφορηθεί για το level του αλλά και πόσο ακόμα απέχει από το να ανέβει level (XP bar). Για όλα αυτά δημιουργήθηκε το script CharacterMenu.cs το οποίο έχει φτιαχτεί με τέτοιον τρόπο ώστε να παρέχει τις παραπάνω επιλογές στον παίχτη. Τέλος έχει δημιουργηθεί και η επιλογή Exit πατώντας το κουμπί που βρίσκεται στο κάτω δεξιό μέρος (Εικόνα 3.12) όταν ανοίγουμε το Menu ή για συντομία το (control + Q).

Εικόνα 3.11: Menu

Εικόνα 3.12: Exit Button

- Το Health Bar του παίκτη βρίσκεται πάνω από το εικονίδιο του Menu κάτω στην αριστερή πλευρά της οθόνης και μειώνεται κάθε φορά που ο παίκτης τραυματίζεται από τους αντιπάλους ή αυξάνεται όταν λαμβάνει healing από τα Healing Fountains. Για τη δημιουργία της συγκεκριμένης επιλογής χρειάστηκε να γίνουν οι εξής ενέργειες:
    - Σε επίπεδο Unity δημιουργήθηκε UI image με ονομασία HealthBar στο κεντρικό panel HUD.
    - Στη συνέχεια μια UI image με ονομασία Health, η οποία θα αυξομειώνεται ανάλογα με το health του χαρακτήρα μας.
    - Σε επίπεδο κώδικα στο script GameManager.cs δημιουργήθηκε η function OnHitpointChange, με την οποία πραγματοποιείται το ανωτέρω και προστέθηκε στο GameManager της Unity στο Hitpoint Bar.
    - Τέλος, στο script Player.cs προστέθηκε η function ReceiveDamage.
- Κλείνοντας, το Death Menu 

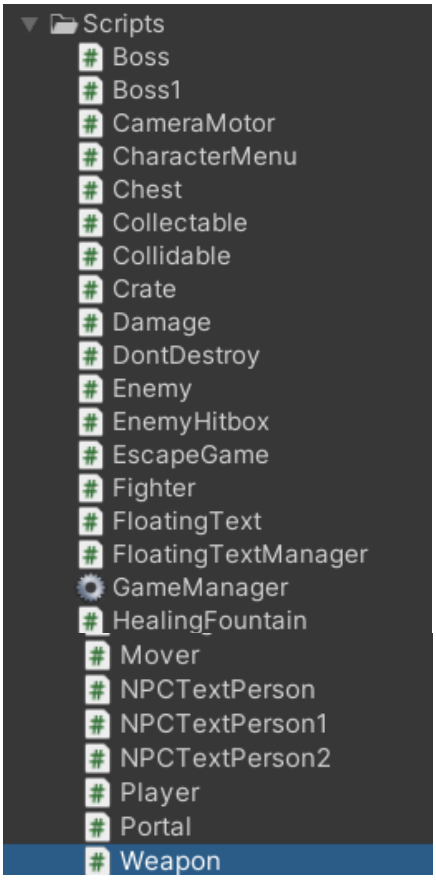
εμφανίζεται όταν ο παίκτης χάσει όλη του τη ζωή από τους αντιπάλους. Με την επιλογή respawn ο παίκτης επιστρέφει στην αρχική σκηνή (first try) και παίζει πάλι από την αρχή. Αρχικά, σε επίπεδο Unity δημιουργήθηκε ένα νέο panel με ονομασία DeathWindow και στη συνέχεια ένα button με όνομα ToMainScene με text RESPAWN. Έπειτα, όλα τα παραπάνω προστέθηκαν σε ένα νέο panel που ονομάστηκε DeathMenu (Εικόνα 3.13) στο οποίο προστέθηκε το σχετικό Animator που θα αναλυθεί παρακάτω στο Κεφάλαιο 3.3 που αφορά στην υλοποίηση των animations.



Εικόνα 3.13: HUD with Menu and Death Menu

3.2 Β΄ φάση : Δημιουργία scripts

Τα scripts που δημιουργήθηκαν απεικονίζονται στην Εικόνα 3.14 και περιγράφονται εν συντομία στον Πίνακα 3.1 και αναλυτικότερα στο ίδιο κεφάλαιο.



Εικόνα 3.14: Τα scripts που δημιουργήθηκαν

Πίνακας 3.1: Περιγραφή των scripts

Script Name	Sort Description
CameraMotor.cs	Follows Panduin's movement (added on main camera)
CharacterMenu.cs	Buy new weapons, information about experience level and pesos amount, and switching of players)
GameManager.cs	Script for the setup of whole game (weapon sprites and prices, player sprite, experience, hitpoint bar, menu, hud, deathmenu floating text Save/Load state etc.)
Collidable.cs	Script for everything we have collision such as NPC, Healing Fountain, Weapon, Enemy Hitbox
Collectable.cs	Script to collect pesos
Player.cs	Script declaring players attributes
Enemy.cs	Enemies' movement and range, experience, and damage
Mover.cs	Players' and enemies' movement
Boss.cs	Boss' movement and range, experience, and damage
Chest.cs	Gives pesos
Crate.cs	Destroyable item
Damage.cs	Damage of Panduins' weapon and push force
DontDestroy.cs	Script that keeps game objects between scenes
EnemyHitbox.cs	Received damage and push force
FloatingText.cs	Text display
FloatingTextManager.cs	Manager showing all floating text everywhere in the map
HealingFountain.cs	Players' healing
NPCTextPerson.cs	Neutral Person who gives hints
Portal.cs	Teleport between scenes
Weapon.cs	Weapons' upgrade and movement
Fighter.cs	Script used to declare hitpoint pushForce and damage to fighters
Quit.cs	Script made so we can exit from the game with two ways via exit button or by pressing Ctrl + Q

Δημιουργία Collidable.cs

Δημιουργήθηκε το συγκεκριμένο script για να οριστούν ποια objects θα θεωρούνται σαν collidables και στη συνέχεια θα προστεθεί μέσω της Unity η επιλογή Box Collider 2D. Με λίγα λόγια με ποια objects θα έρχεται σε επαφή ο χαρακτήρας μας και στη συνέχεια όπως θα δούμε στα επόμενα scripts το κάθε object θα έχει την δικιά του ιδιαιτερότητα. Τέτοια objects είναι τα chests, healing fountains, NPCs, portals, weapon, enemy hitbox.

Δημιουργία Collectable.cs

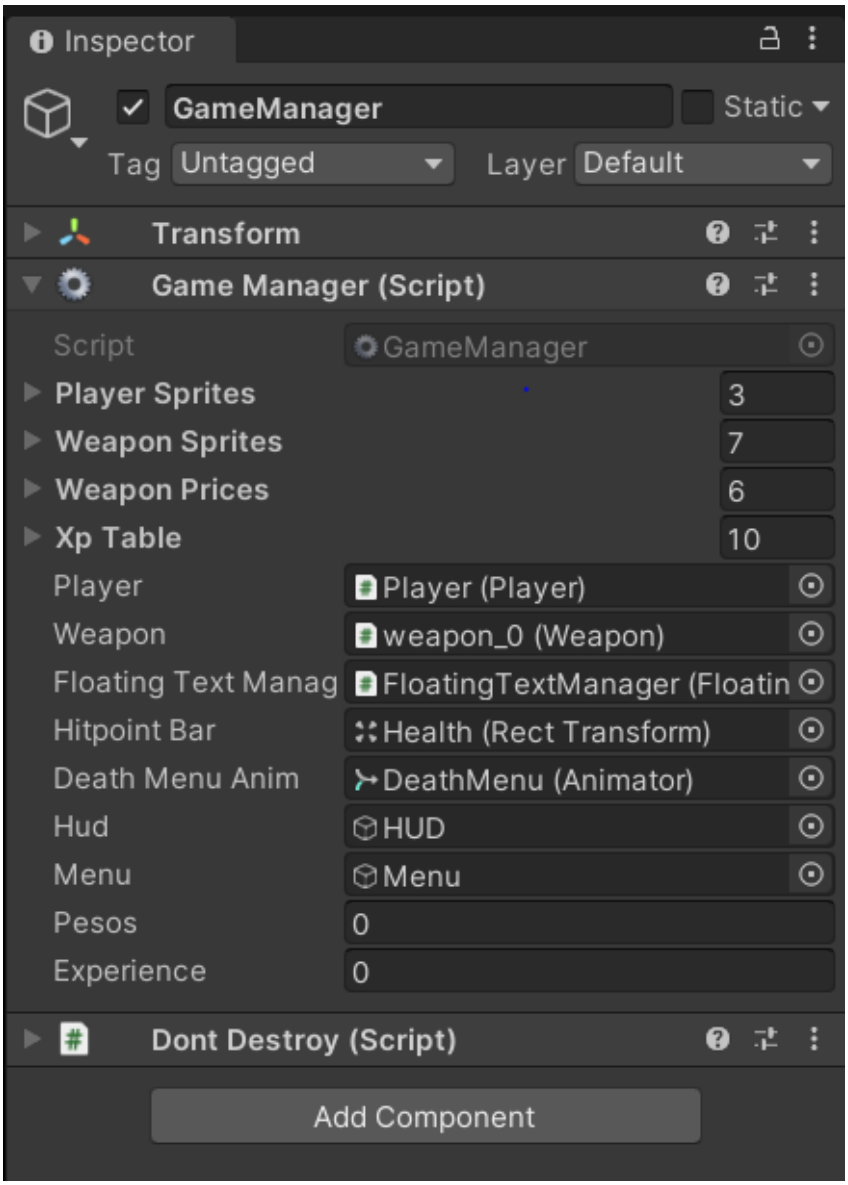
Δημιουργήθηκε το συγκεκριμένο script για να οριστούν τα object με τα οποία θα έρχεται σε επαφή ο χαρακτήρας μας και θα του δίνουν κάποια πράγματα. Στην προκειμένη περίπτωση το μοναδικό στοιχείο που φτιάχτηκε και δίνει κάποια πλεονεκτήματα στον χαρακτήρα μας είναι τα chests τα οποία δίνουν pesos και μπορεί να τα χρησιμοποιήσει για να αγοράσει νέα όπλα. Όπως αναφέρθηκε προηγουμένως, τα παραπάνω objects είναι και collidable, για αυτό τον λόγο κατά τη δημιουργία του script collectable.cs, ορίστηκε ως children του Collidable, παίρνοντας έτσι τις μεταβλητές του.

Public class Collectable : Collidable

Τέλος προστέθηκε η επιλογή να μπορεί να κάνει συλλογή των pesos μόνο ο παίκτης μας.

Δημιουργία GameManager.cs

Δημιουργήθηκε το συγκεκριμένο script, για να μπορούμε να κρατάμε όλα τα χαρακτηριστικά του παίκτη μας σε μια τάξη. Είναι public για να το βλέπουν όλα τα υπόλοιπα script και static για να έχουμε access από παντού. Το φτιάχνουμε ουσιαστικά για να είναι available σε όλο το game. Δημιουργήθηκε στην Unity new empty Object με την ονομασία GameManager. Επίσης ορίστηκαν τα βασικά μας references του παιχνιδιού player, upgrade weapon (εξέλιξη του όπλου), floating text manager (διάλογος με τα NPCs), xp level (πώς παίρνει xp και αυξάνεται το level του παίκτη), hitpoint bar (η μεταβολή της life bar του παίκτη), menu, death menu (respawn στην first try σκηνή), hud. Επίσης στο συγκεκριμένο βάλαμε το Save/Load state, όπου ο παίκτης κατά την εναλλαγή των σκηνών δεν χάνει τα pesos, τα weapons και τα level του (Εικόνα 3.15).



Εικόνα 3.15: GameManager object

Δημιουργία Quit.cs

Το συγκεκριμένο script δημιουργήθηκε με σκοπό ο παίκτης να μπορεί να κάνει Exit από το παιχνίδι είτε με τη χρήση του κουμπιού που βρίσκεται στο Menu στα δεξιά είτε με τη χρήση των κουμπιών Control + Q. Προστέθηκε σε επίπεδο Unity σαν Button object κάτω από το Menu και ονομάστηκε Button_Quit.

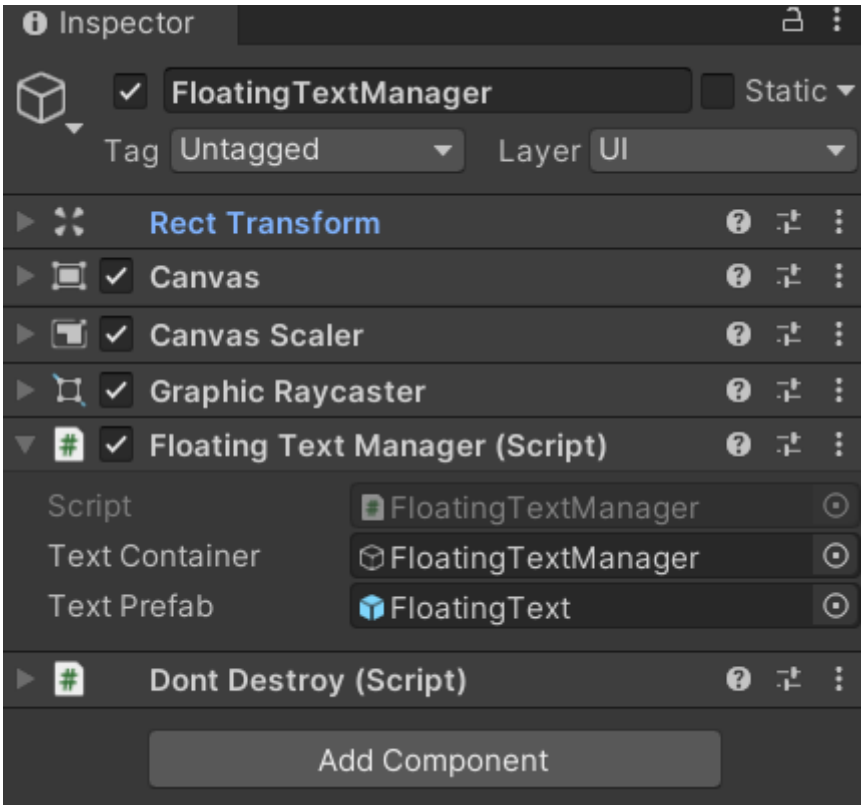
Δημιουργία DontDestroy.cs

Το συγκεκριμένο script δημιουργήθηκε με σκοπό ο παίκτης να μην χάνει τα weapons, τα pesos και τα level του με την εναλλαγή σκηνών μετά τη χρήση των portals. Προστέθηκε σε επίπεδο Unity σαν component στα objects Player, GameManager, FloatingTextManager, HUD, Menu.

Δημιουργία FloatingText.cs & FloatingTextManager.cs

Το FloatingText.cs script δημιουργήθηκε με σκοπό την απεικόνιση του floating text. Συνδέθηκε με το GameManager.cs μέσω της function ShowText, η οποία χρησιμοποιείται και σε άλλα scripts, όπως στο NPCTextPerson.cs και στο Chest.cs, προκειμένου όταν ο παίκτης μας να έρχεται σε επαφή με κάποιο NPC ή σε κάποιο Chest να βγαίνει το αντίστοιχο μήνυμα που έχουμε γράψει. Επιπλέον δημιουργήθηκε το FloatingTextManager.cs με σκοπό την απεικόνιση του floating text παντού σε όλο το παιχνίδι.

Σε επίπεδο Unity δημιουργήθηκε νέο UI Panel το οποίο ονομάσαμε FloatingTextManager και στο οποίο προστέθηκε το script FloatTextManager.cs όπως επίσης το text container & το text prefab (Εικόνα 3.16).



Εικόνα 3.16: FloatingTextManager object

Δημιουργία Mover.cs

Δημιουργήθηκε το script Mover.cs για να ορίσουμε την κίνηση των players και των enemies. Οτιδήποτε μπορεί να πολεμήσει μπορεί και να κινηθεί για αυτό τον λόγο ορίσαμε ότι το συγκεκριμένο script θα παίρνει τις μεταβλητές από το Fighter.cs (που αναλύεται παρακάτω).

Public class Mover : Fighter

Επίσης στο συγκεκριμένο script ορίσαμε και το push force και recovery που θα κάνει ο παίκτης μας και οι enemies, μετά την επίθεση.

Δημιουργία Damage.cs

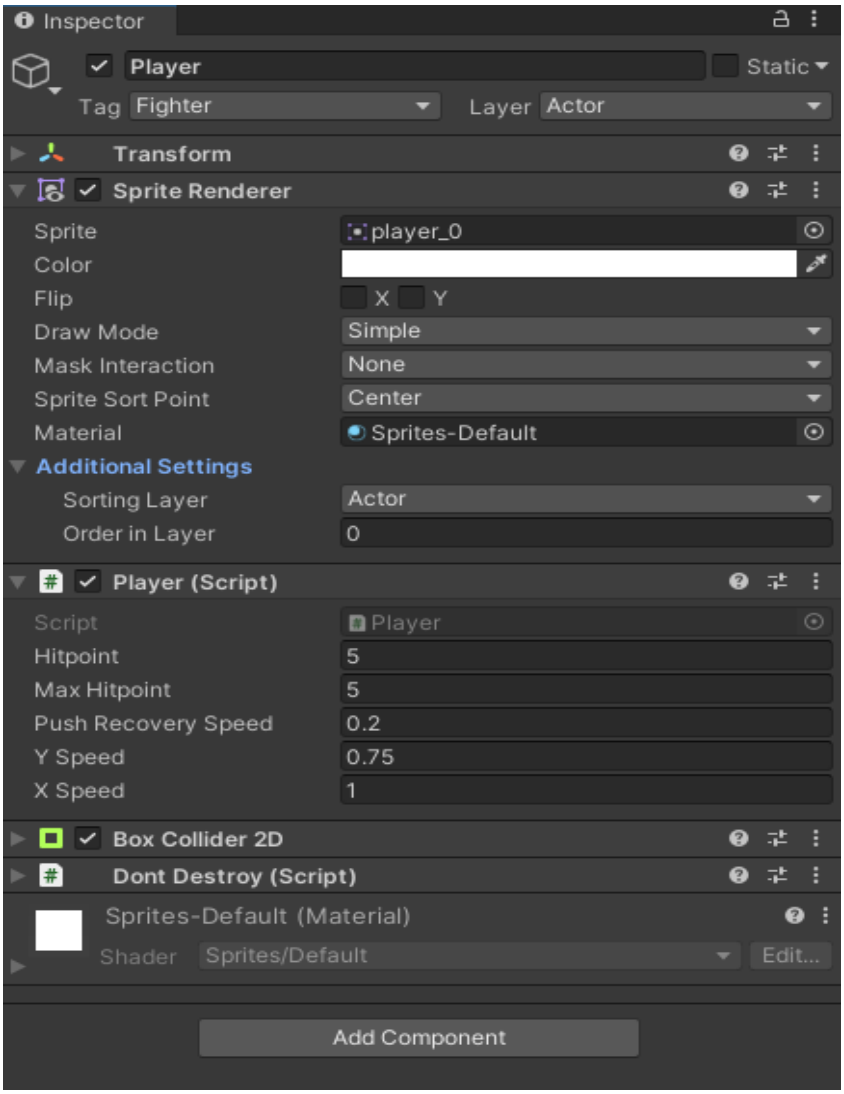
Σκοπός της δημιουργίας του script Damage.cs, είναι για να δηλώσουμε το damage που θα έχει το weapon του παίκτη αλλά και το push force.

Δημιουργία Player.cs

Ένα από τα πιο βασικά scripts που έχουν φτιαχτεί είναι αυτό του Player, το οποίο έχει ονομαστεί Player.cs. Όπως και στις προηγούμενες περιπτώσεις ορίστηκε η αλληλουχία μεταξύ του συγκεκριμένου script με το Mover.cs.

Public class Player : Mover

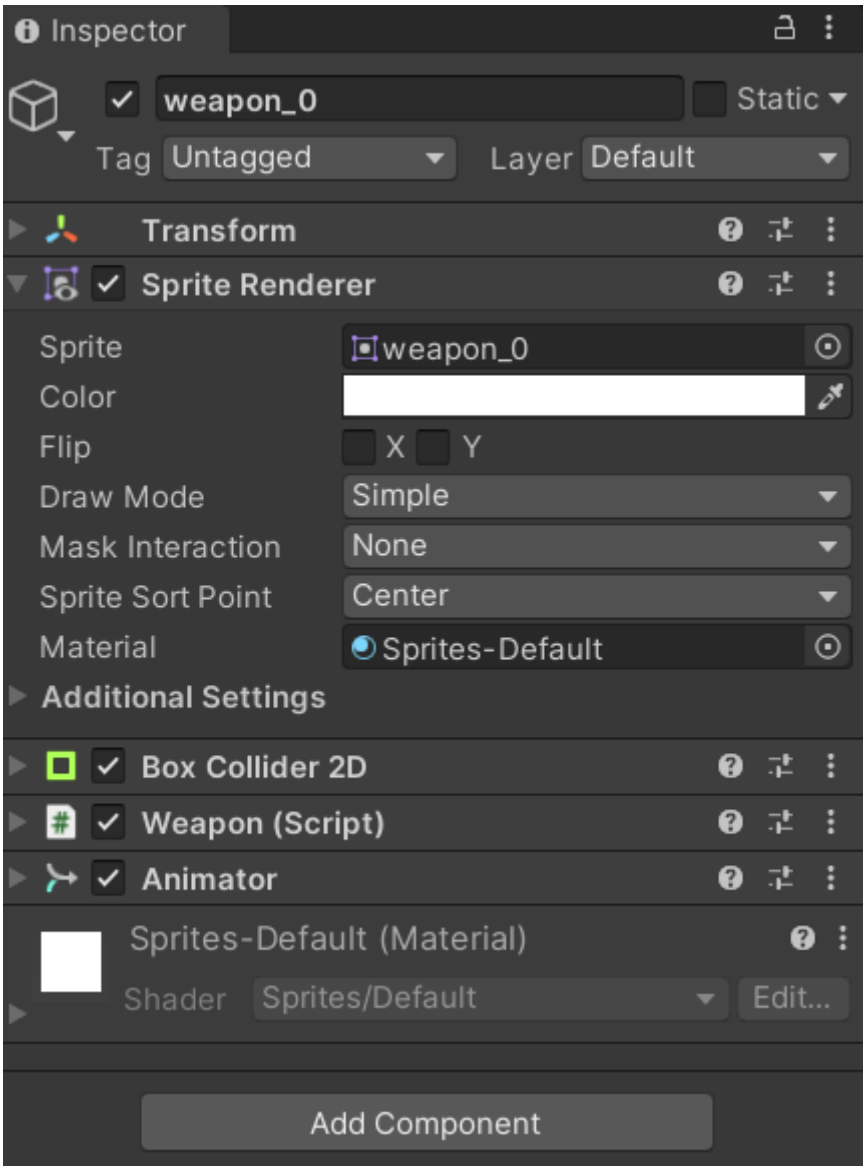
Στο συγκεκριμένο script ορίστηκαν αρχικά η εικόνα και η κίνηση (Horizontal & Vertical axis) του παίκτη μας, αν δέχεται damage, αν έχει πεθάνει και το respawn του στην αρχική σκηνή, τη δυνατότητα να αλλάξουμε αμφίεση, την αύξηση σε levels και το healing του. Παρακάτω βλέπουμε πώς έγινε η σύνδεση όλων των ανωτέρω με την Unity (Εικόνα 3.17).



Εικόνα 3.17: Player object

Δημιουργία Weapon.cs

Για να μπορέσει ο player να κάνει επίθεση στους enemies/bosses δημιουργήθηκε το συγκεκριμένο script Weapon.cs, στο οποίο έχει δηλωθεί το damage που θα κάνει και το push force ανάλογα με τα level του όπλου, το upgrade του, την κίνησή του (swing) και το cooldown (swing = 0,5f, δηλαδή 2 φορές/sec) κάθε επίθεσης με το weapon. Αφού φτιάξαμε το script, προχωρήσαμε σε επίπεδο Unity στη μεταφορά του weapon_0 κάτω από το object player. Στη συνέχεια ορίστηκαν οι συντεταγμένες, προστέθηκε το layer weapon. Επίσης ορίστηκαν τα όριά του μέσω του BoxCollider2D και προστέθηκε το script Weapon.cs μέσω του add component (Εικόνα 3.18). Τέλος να επισημανθεί ότι έχει δημιουργηθεί και animation όσον αφορά την κίνηση του weapon, το οποίο θα αναλυθεί στο παρακάτω στο Κεφάλαιο 3.3 (Animation).

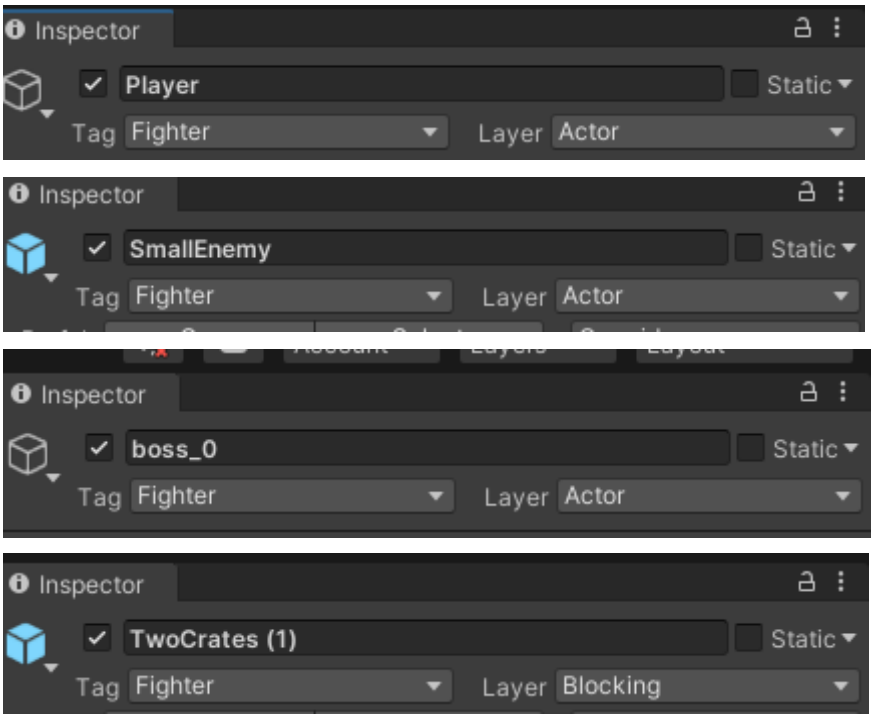


Εικόνα 3.18: Weapon object

Δημιουργία Fighter.cs

Το script αυτό χρησιμοποιείται από όσα objects μπορούν να κάνουν damage, να δεχθούν και να πεθάνουν, δηλαδή από τον player και τους enemies, Το script του fighter, όπως αναφέρθηκε παραπάνω δίνει τις μεταβλητές του στο script Mover.cs και αυτό με τη σειρά του σε όλα τα objects που μπορούν να κάνουν επίθεση αλλά και να κινηθούν μέσα στο map (Εικόνα 3.19).

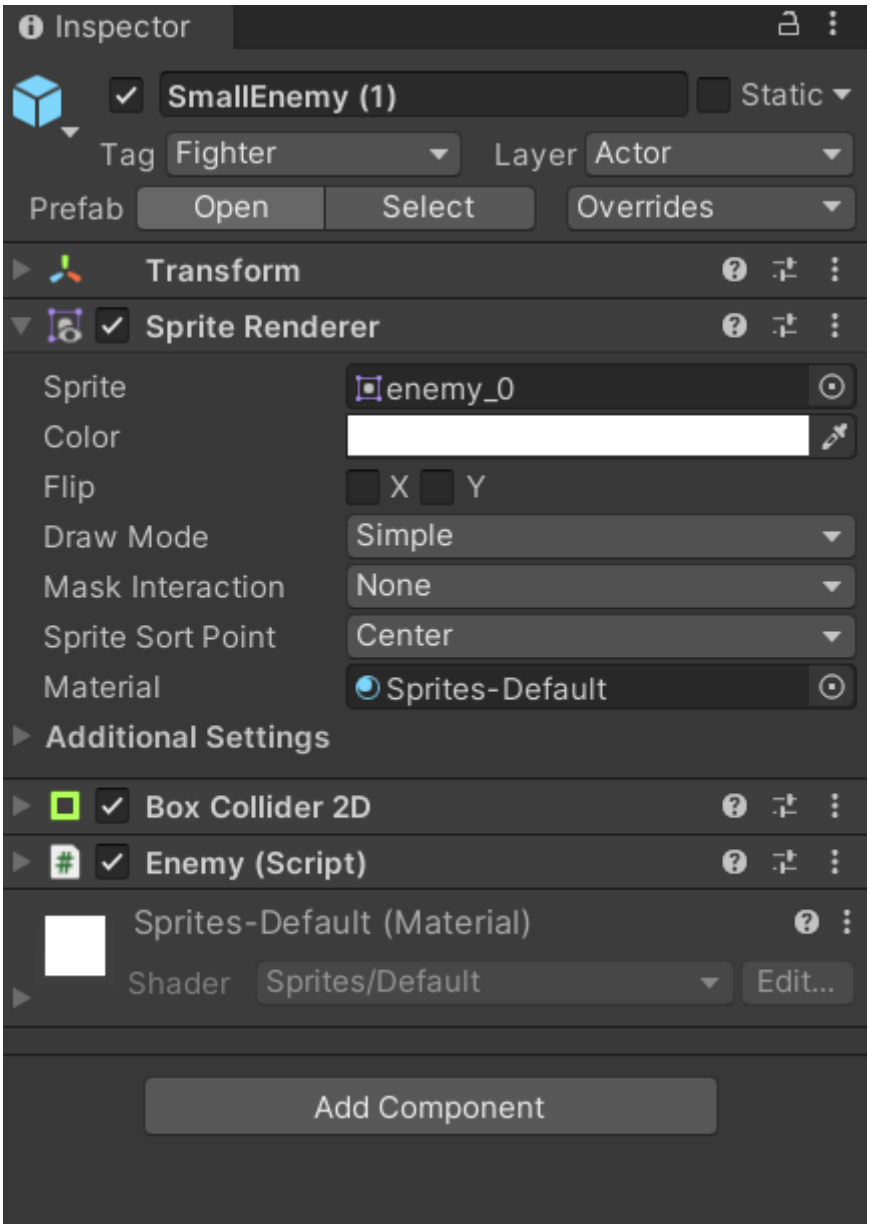
Επίσης δημιουργήθηκε το tag Fighter και στη συνέχεια δηλώσαμε το tag στον player, στους enemies, bosses και στα crates (όπως θα δούμε παρακάτω).



Εικόνα 3.19: Player, SmallEnemy, Boss, Crates Fighter Tag

Δημιουργία Enemy.cs

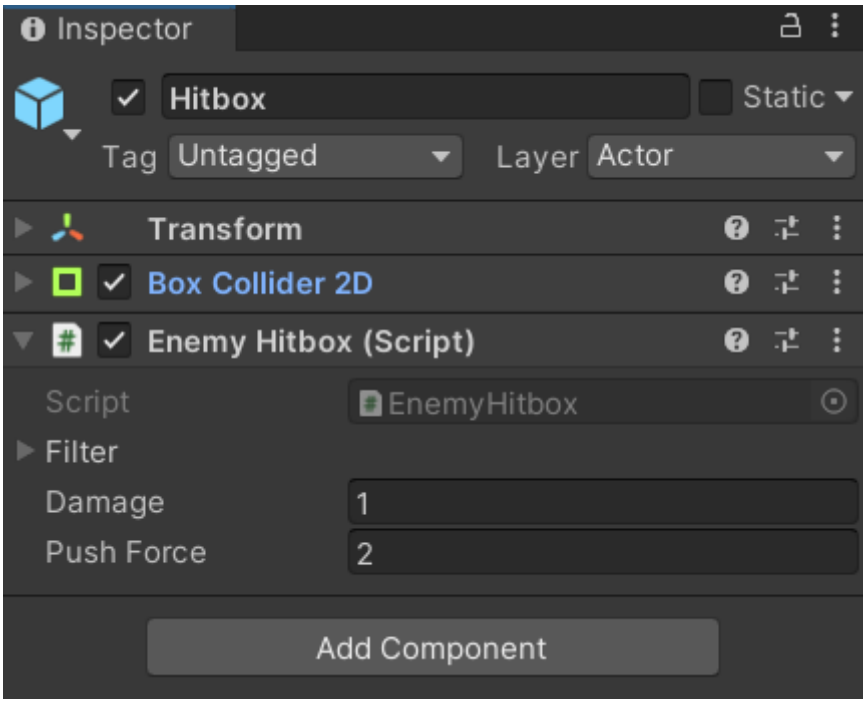
Επίσης αλληλένδετο με το script του Mover.cs είναι αυτό του Enemy.cs, στο οποίο ορίζουμε τους enemies να μπορούν να κινηθούν σαν Fighters και ορίζουμε την κίνηση που εκτελούν, το trigger range όταν πλησιάζει ο παίκτης, αλλά και όταν αυτοί πεθαίνουν από τον παίκτη μας, τότε να εξαφανίζονται από το map και να δίνουν xp. Σε επίπεδο Unity προχωρήσαμε στην εισαγωγή sorting layer Actor και tag Fighter στο object enemy_0 (Εικόνα 3.20). Επίσης έγινε προσθήκη του BoxCollider 2D component για να δηλώσουμε τα όρια του σώματός του.



Εικόνα 3.20: SmallEnemy object

Δημιουργία EnemyHitbox.cs

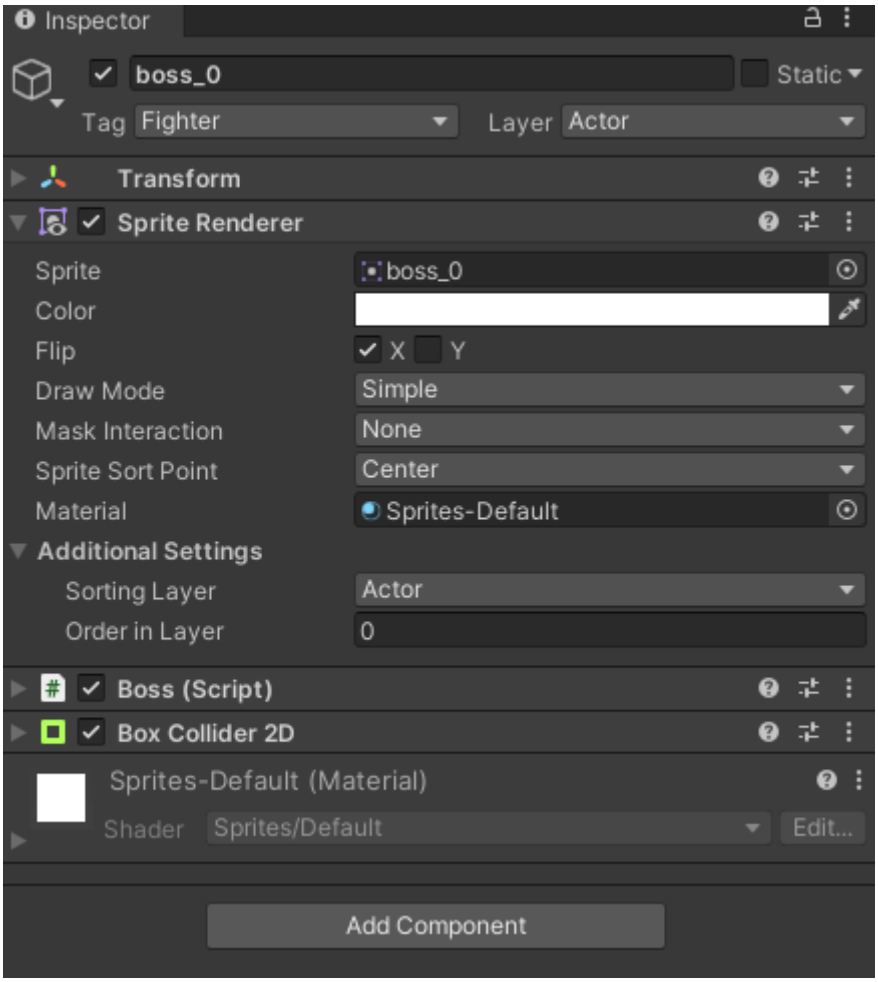
Επιπλέον αλληλένδετο με το ανωτέρω script του Enemy.cs είναι αυτό του EnemyHitbox.cs, στο οποίο ορίζουμε το σώμα των enemies, το damage και το push force που κάνουν στον παίκτη μας. Σε επίπεδο Unity προχωρήσαμε στην προσθήκη ενός 2D object κάτω από το SmallEnemy και το ονομάσαμε Hitbox μέσω του add component προστέθηκε το script EnemyHitbox.cs (Εικόνα 3.21).



Εικόνα 3.21: Hitbox object

Δημιουργία Boss.cs

Η λογική του script Boss.cs είναι παρόμοια με του Enemy.cs, ορίστηκαν όπως και στους enemies, σαν fighters. Προστέθηκε η κίνηση που εκτελούν, το trigger range όταν πλησιάζει ο παίκτης, το damage που κάνουν και το push force. Σε επίπεδο Unity έγινε εισαγωγή στο Boss σαν sorting layer Actor και tag Fighter, επίσης έγινε προσθήκη του BoxCollider 2D component για να δηλώσουμε τα όριά του (Εικόνα 3.22).



Εικόνα 3.22: Boss object

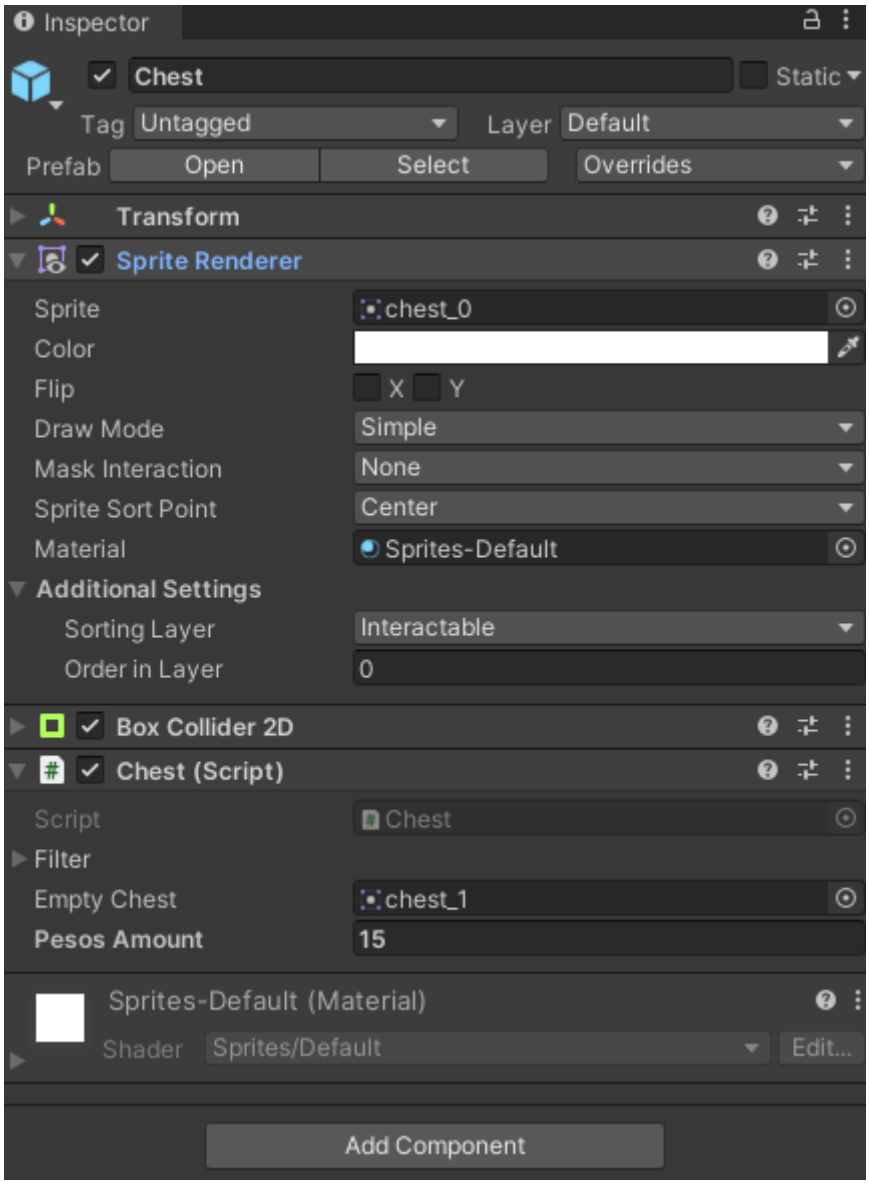
Δημιουργία Chest.cs

Δημιουργήθηκε ένα feature για τα chest με 2 επιλογές, chest_0 όταν είναι γεμάτο και chest_1 όταν το συλλέξει ο χαρακτήρας μας αυτό να αδειάζει. Για την παραπάνω επιλογή δημιουργήθηκε ένα νέο script, το οποίο το ονομάσαμε Chest.cs και το ορίσαμε να παίρνει τις μεταβλητές του script Collectable.cs, που με τη σειρά του υποδηλώθηκε να παίρνει τις μεταβλητές του από το Collidable.cs, όπως θα βλέπουμε από τα κάτωθι:

Public class Collectable : Collidable

Public class Chest : Collectable

Επίσης μέσω της Unity δώσαμε την επιλογή σε κάθε chest να δίνει διαφορετικό amount of pesos (Εικόνα 3.23).



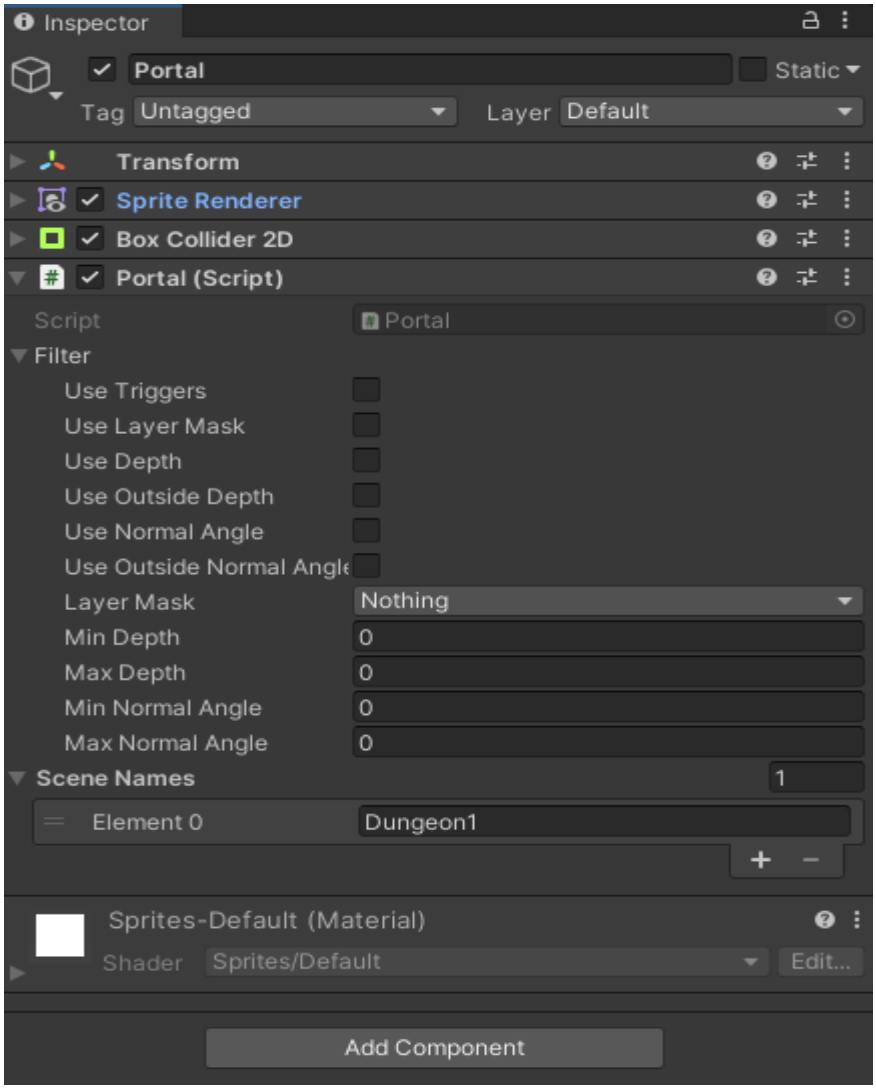
Εικόνα 3.23: Chest object

Δημιουργία Portals.cs

Δημιουργήθηκε επιπλέον μια επιλογή όπου ο χαρακτήρας θα μπορεί να κάνει teleport μεταξύ των σκηνών. Για αρχή δημιουργήθηκε ένα νέο 2D Object (Sprites Square) και στη συνέχεια φτιάχτηκε το script Portal.cs, με το οποίο δώσαμε στον παίκτη τη δυνατότητα αυτή. Έπειτα, βάλαμε στην επιλογή scene name τη σκηνή που θέλουμε το portal να μεταφέρει τον παίκτη μας.

Στη δικιά μας περίπτωση θέλουμε τα portals να έχουν την εξής ροή σκηνών:

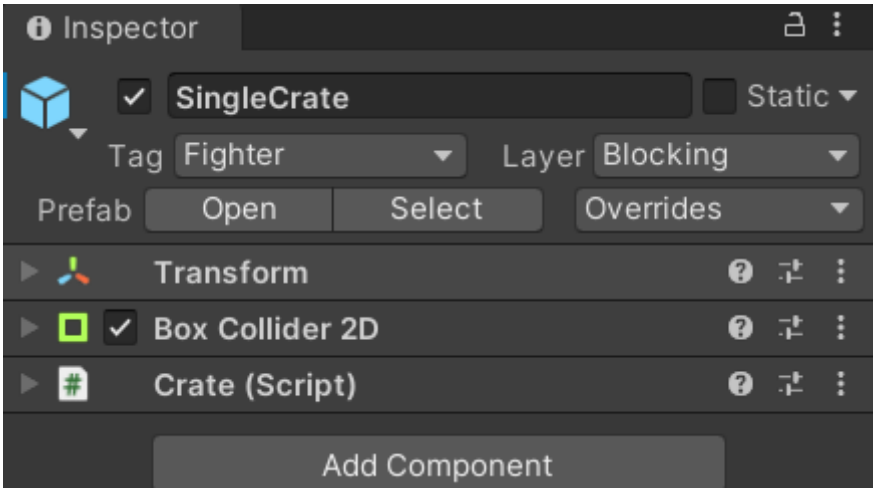
Παρόλα αυτά έχει δοθεί και η δυνατότητα, μέσω του Portal.cs, το κάθε portal να μεταφέρει randomly τον παίκτη σε οποιαδήποτε σκηνή. Αυτό μπορεί να ενεργοποιηθεί αν προσθέσουμε στα elements του portal τις σκηνές στο scene names (Εικόνα 3.24).



Εικόνα 3.24: Portal object

Δημιουργία Crates.cs

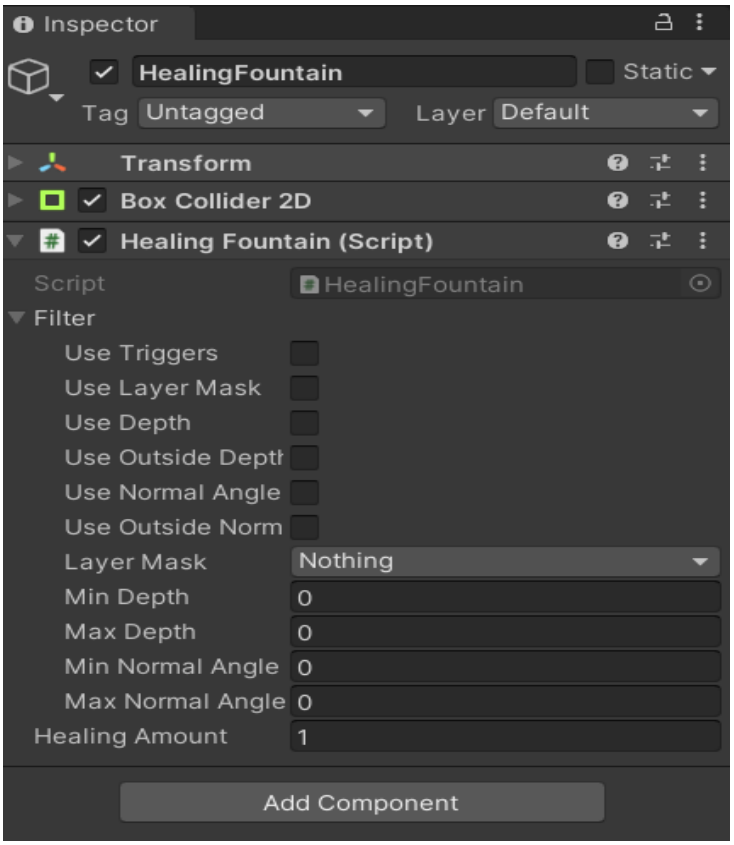
Για να γίνει πιο ελκυστικό το παιχνίδι μας δημιουργήσαμε ένα επιπλέον script Crate.cs, στο οποίο έχουμε ορίσει σαν Fighter και Blocking layer τα κουτιά ούτως ώστε ο παίκτης να μπορεί να τα διαλύει, καθότι αυτά κλείνουν τον δρόμο του παίκτη και δεν τον αφήνουν να εισέλθει σε νέα δωμάτια που έχουν δημιουργηθεί. Δεν έχουμε δώσει την επιλογή στα κουτιά να κάνουν Damage στον παίκτη μας, παρά μόνο να δέχονται έτσι ώστε να μπορούν να διαλυθούν από το weapon μας. Δημιουργήθηκαν 2 Game Objects Single Crates & Two Crates, τα οποία έχουν τοποθετηθεί στις σκηνές για να αυξήσουν τη δυσκολία του παιχνιδιού αλλά και να βελτιστοποιήσουν το user experience (Εικόνα 3.25).



Εικόνα 3.25: Crate object

Δημιουργία Healing Fountains.cs

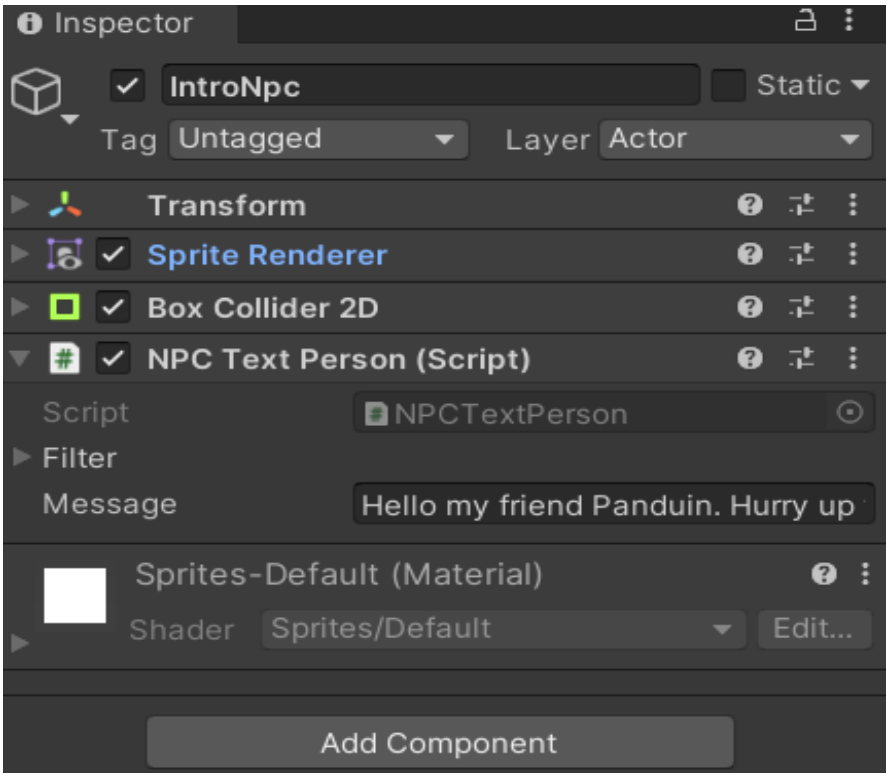
Επιπλέον δημιουργήσαμε τα Healing Fountains ώστε ο παίκτης να μπορεί να ανακτά το χαμένο life από τα χτυπήματα που δέχεται από τους αντιπάλους. Για αυτό τον λόγο φτιάξαμε ένα νέο script HealingFountain.cs και φτιάξαμε στο Player.cs την function public void Heal. Ο συνδυασμός των 2 script που αναφέρθηκαν έγινε ώστε ο παίκτης όταν εισέρχεται στο οριζόμενο πεδίο των healing fountain να γεμίζει το life του. Τέλος δημιουργήθηκε ένα empty game object ονομαζόμενο Healing Fountain στο οποίο ορίσαμε τα όρια που θα δέχεται heal ο παίκτης μέσω του component BoxCollider2D και προσθέσαμε ως component το HealingFountain.cs (Εικόνα 3.26).



Εικόνα 3.26: Healing Fountain object

Δημιουργία NPCs.cs

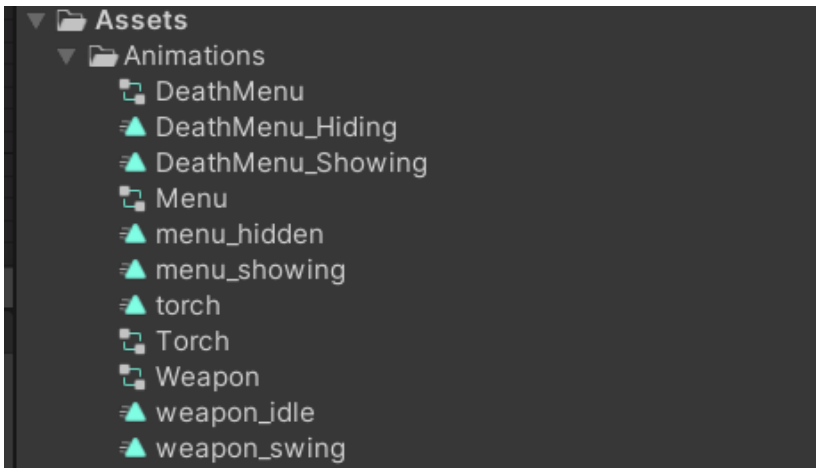
Οι συγκεκριμένοι χαρακτήρες δημιουργήθηκαν, καθαρά για να βελτιστοποιήσουν το user experience, δίνοντας βοηθητικές συμβουλές στον παίκτη κατά την εξέλιξη της ροής του παιχνιδιού. Σε αυτή την περίπτωση φτιάχτηκε το script NPCTextPerson.cs, στο οποίο ορίσαμε ότι όταν ο παίκτης έρχεται σε επαφή με τα NPC να υπάρχει ένα text που θα δίνει συμβουλές στον παίκτη, να έχει συγκεκριμένη διάρκεια ώστε να μπορεί να το διαβάσει ο παίκτης και να έχει κάποιο cooldown μέχρι να ενεργοποιηθεί εκ νέου. Τέλος, σε κάθε NPC που έχει προστεθεί στο map δίνει και διαφορετική συμβουλή στον Panduin (Εικόνα 3.27).



Εικόνα 3.27: NPCs object

3.3 Animation

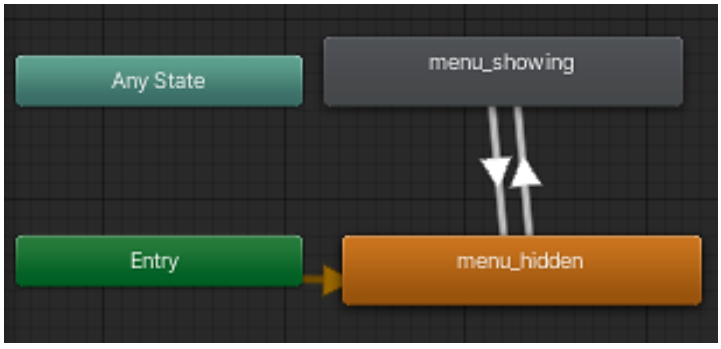
Για την καλύτερη αναπαράσταση του παιχνιδιού δημιουργήθηκαν ορισμένα animations (Εικόνα 3.28), συμπεριλαμβανομένου του DeathMenu, του Menu, του Torch και του Weapon swing.



Εικόνα 3.28: Δημιουργία animations

- **Menu**:

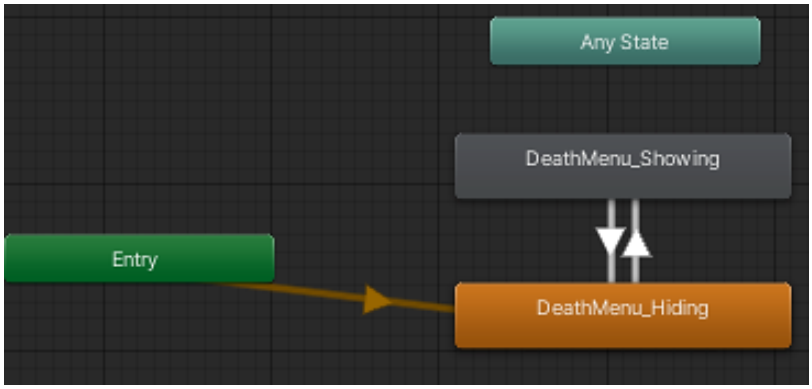
  Δημιουργήθηκε ένας animator controller με το εξής flow:



Εικόνα 3.29: Menu animator

Στην αρχή το μενού είναι κρυμμένο (menu_hidden) και εμφανίζεται (menu_showing) όταν επιλέγεται το κουμπί του menu.

- **DeathMenu:**

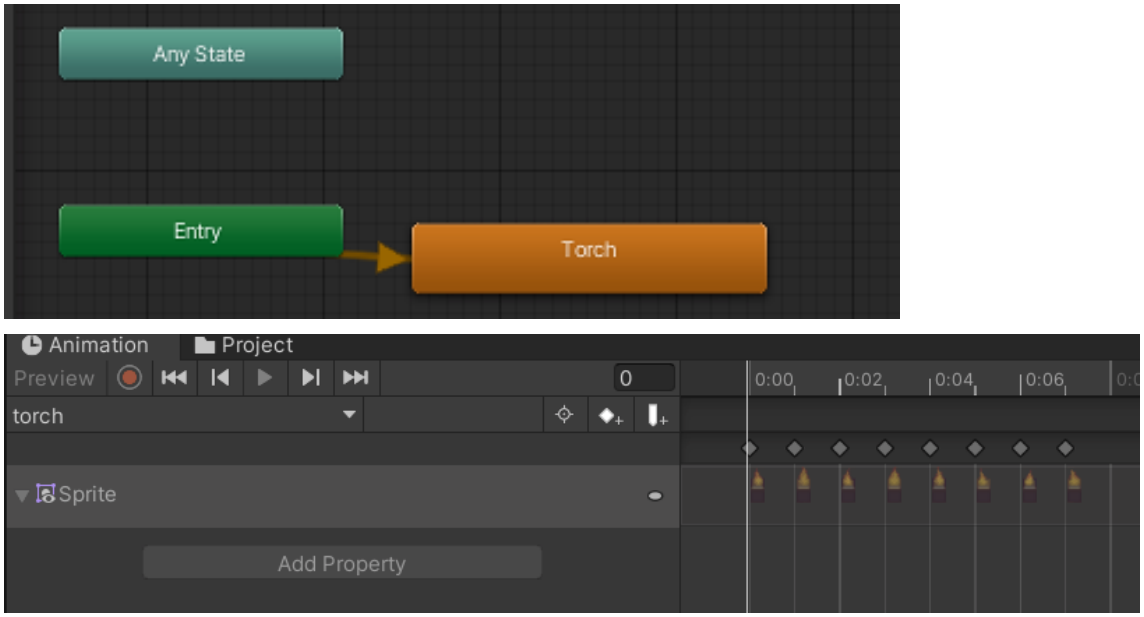


Εικόνα 3.30: Death Menu animator

Στην αρχή το μενού είναι κρυμμένο (DeathMenu_Hiding) και εμφανίζεται (DeathMenu_Showing) όταν ο παίκτης χάσει όλη του τη ζωή. Ο παίκτης έχει τη δυνατότητα να πατήσει respawn για να ξαναπαίξει.

- **Torch:**

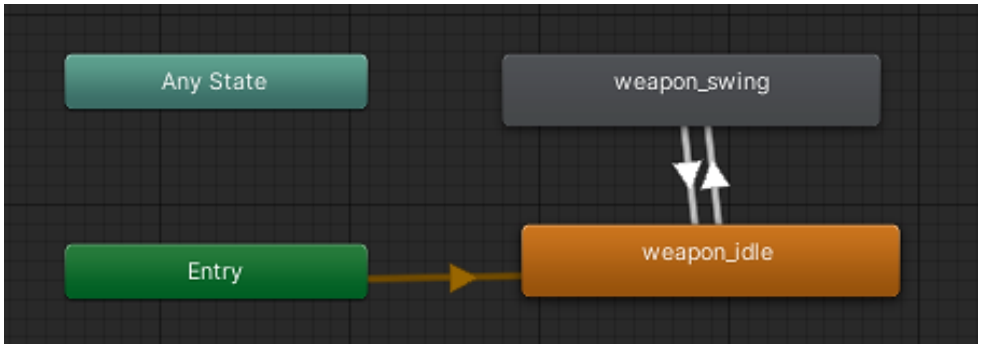
  Η δημιουργία του συγκεκριμένου animation με την κίνηση της φλόγας, αποσκοπούσε στη βελτίωση της ψυχαγωγικής εμπειρίας του χρήστη.



Εικόνα 3.31: Torch animator

- **Weapon swing:**

  Η υλοποίηση του συγκεκριμένου animation είναι πιο σύνθετη σε σχέση με τα ανωτέρω. Ο παίκτης όσο δεν χρησιμοποιεί το όπλο του, αυτό βρίσκεται σε κατάσταση idle (weapon_idle), ενώ όταν το χρησιμοποιεί βρίσκεται σε κατάσταση swing (weapon_swing), επομένως η πρώτη φάση είναι η κατάσταση weapon_idle και όταν γίνει trigger (space = attack button) μεταβαίνει σε κατάσταση weapon_swing, όπως φαίνεται παρακάτω.



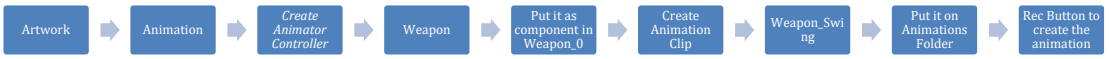
Εικόνα 3.32: Weapon animator

Η κίνηση του όπλου έχει προγραμματιστεί ώστε να είναι κυκλική και να ακολουθεί τόσο καθοδική όσο και ανοδική πορεία (weapon swing), επανερχόμενο στην αρχική του θέση (weapon idle).

Στην συνέχεια δημιουργήσαμε μια αμφίδρομη σχέση ανάμεσα σε Weapon_Swing & Weapon_Idle μέσω του Make Transition. Στα settings κάναμε τα χρονικά περιθώρια 0 και στο Weapon_Idle βάλαμε conditions swing και βγάλαμε το exit time. Επίσης στο weapon_swing βγάλαμε το loop time και κάναμε τις απαραίτητες αλλαγές και στο weapon.cs.

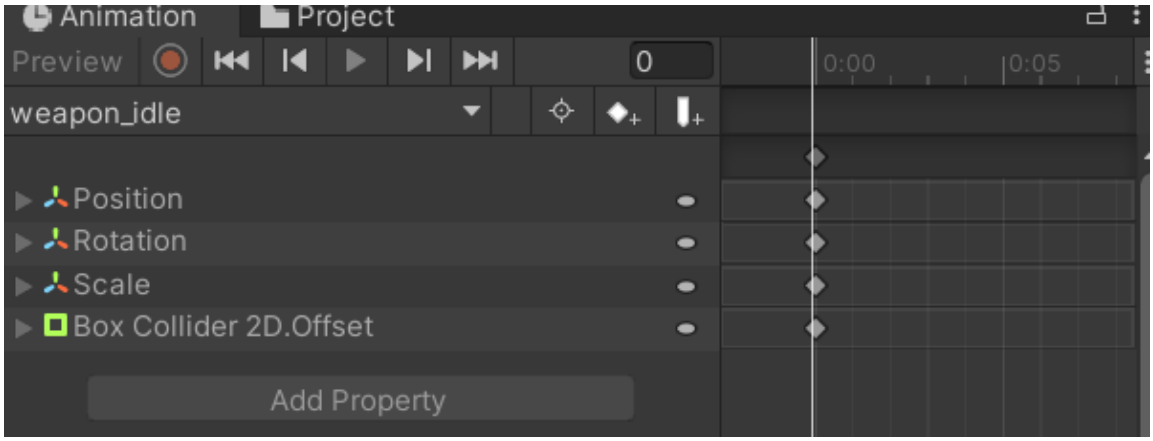
Τα βήματα για την δημιουργία του συγκεκριμένου animation είναι τα εξής:

Weapon Swing:

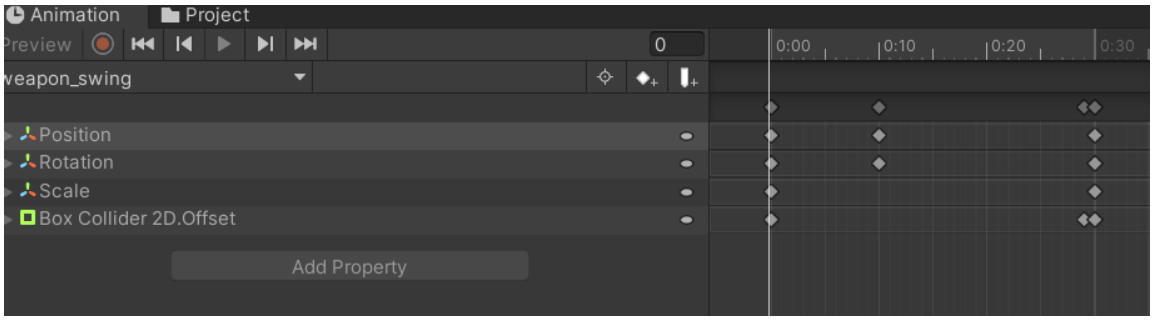


Weapon Idle:

Παραπάνω παρουσιάζεται ο τρόπος δημιουργίας του Animation, του Weapon_Swing και του Weapon_Idle (Εικόνα 3.33 και 3.34). Στη συνέχεια έγινε η προσθήκη του trigger “swing” στο Parameter, προκειμένου να γίνει trigger η κίνηση του όπλου.



Εικόνα 3.33: Weapon idle animation



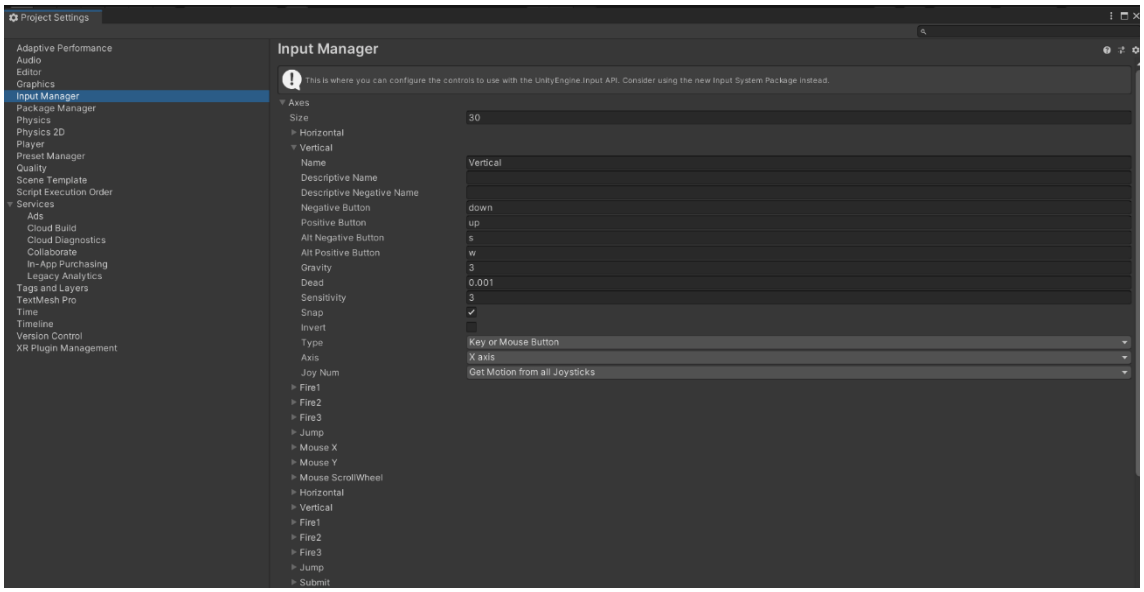
Εικόνα 3.34: Weapon swing animation

3.4 Game manual

Ο χειρισμός του παίκτη γίνεται με τη χρήση του πληκτρολογίου ως εξής:

1. Με το Left arrow ή A ο παίκτης κινείται προς τα αριστερά
2. Με το Right arrow ή D ο παίκτης κινείται προς τα δεξιά
3. Με το Up arrow ή W ο παίκτης κινείται προς τα επάνω
4. Με το Down arrow ή S ο παίκτης κινείται προς τα κάτω
5. Με το Space ο παίκτης κάνει επίθεση με το σπαθί του
6. Με Control + Q κάνουμε Exit από το παιχνίδι

Η πλοήγηση του παίκτη ρυθμίστηκε μέσα από την πλατφόρμα Unity, όπως φαίνεται στην Εικόνα 3.35.



Εικόνα 3.35: Πλήκτρα πλοήγησης παίκτη

3.5 Build and Run game

Η τελευταία πινελιά για την ολοκλήρωση του παιχνιδιού είναι να κάνουμε build το παιχνίδι μέσω της Unity και στη συνέχεια μπορούμε δίνοντας τα αρχεία που θα δημιουργηθούν στον φάκελό μας (The Dungeon.exe & The Dungeon_Data folder) οποιοσδήποτε να παίξει το παιχνίδι μας από τον υπολογιστή του. Η διαδικασία είναι πολύ απλή, θα χρειαστεί να επιλέξουμε το πεδίο File στην συνέχεια Build Settings (Εικόνα 3.36) και τέλος θα επιλέξουμε το Player settings.

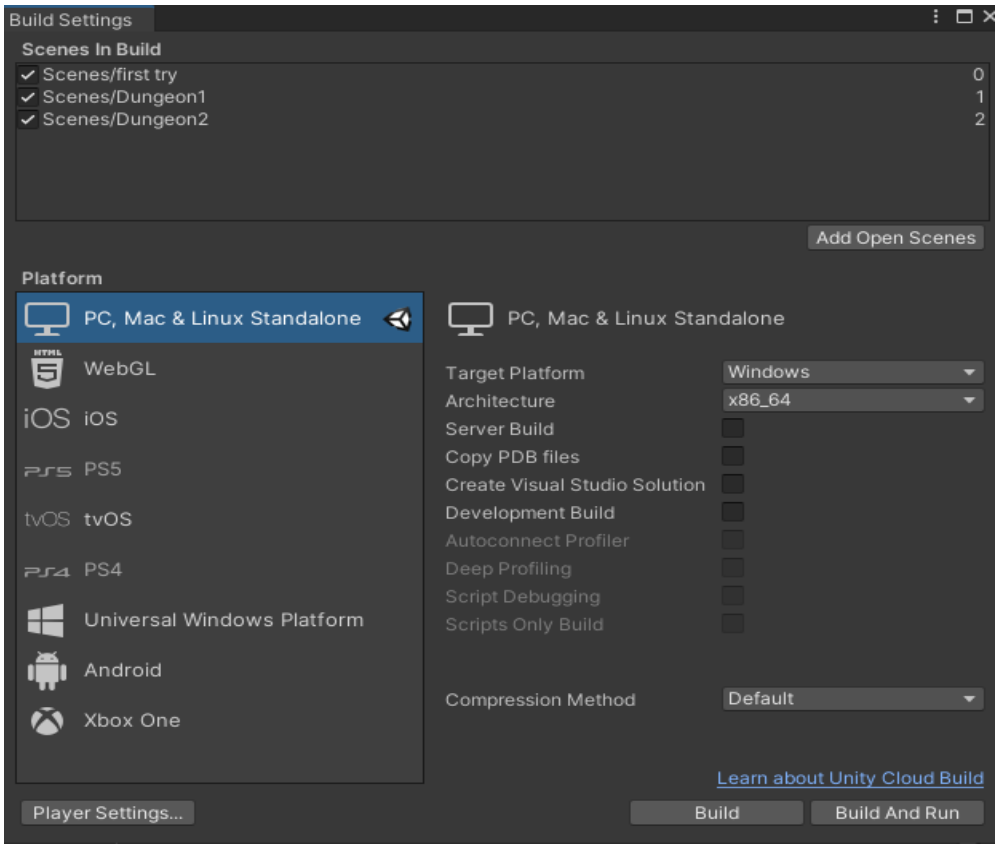
Player settings: Resolution and Presentation (Εικόνα 3.37):

- Fullscreen Mode: Windowed
- Default Screen Width: 800
- Default Screen Height: 600

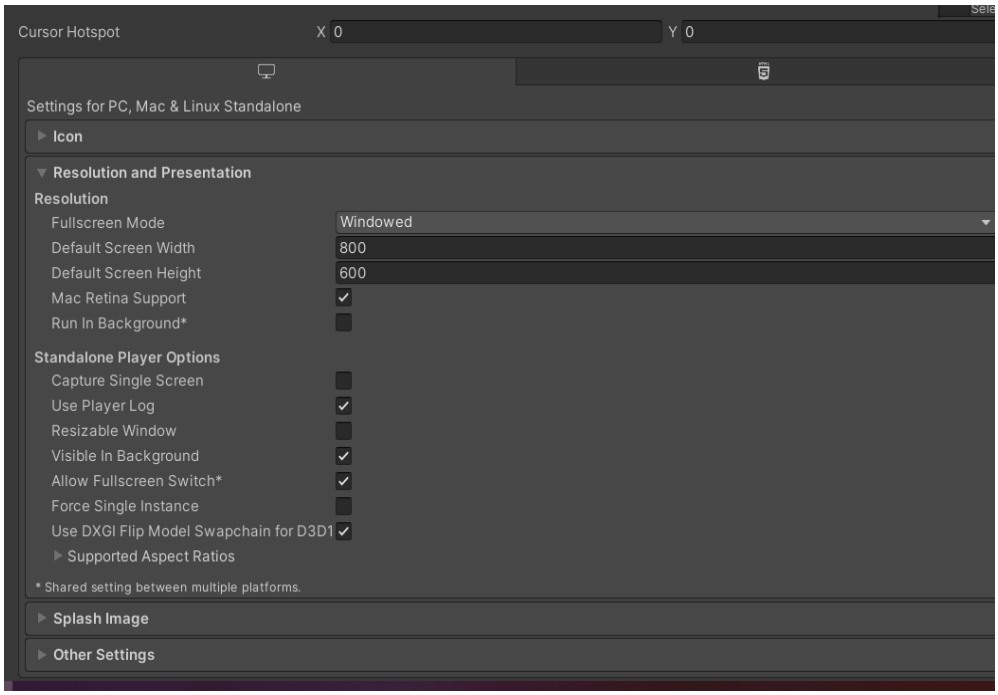
(όπως είχαμε αναφέρει στην αρχή της παρουσίασης ότι το resolution του παιχνιδιού θα είναι 800x600).

Τέλος θα πατήσουμε την επιλογή Build and Run και το παιχνίδι θα είναι έτοιμο!

Καλή απόλαυση!



Εικόνα 3.36: Build and Run Game



Εικόνα 3.37: Modify Resolution and Screen Mode

4. Συμπεράσματα και μελλοντικές επεκτάσεις

Από την παρούσα εργασία γίνεται αντιληπτό ότι η δημιουργία ενός παιχνιδιού RPG με τη χρήση της πλατφόρμας Unity απαιτεί όχι μόνο γνώση της συγκεκριμένης πλατφόρμας, αλλά και γνώσεις προγραμματισμού. Η πορεία υλοποίησης του παρόντος σύνθετου project έδειξε ότι η δημιουργία ενός τέτοιου παιχνιδιού απαιτεί αρκετό χρόνο προκειμένου να είναι λειτουργικό και να γίνει ορθή σύνδεση τόσο του κώδικα (scripts) όσο και των animations-features. Όπως έχει ήδη αναφερθεί στην περίληψη της παρούσας μεταπτυχιακής μελέτης, το συγκεκριμένο παιχνίδι υλοποιήθηκε για εκπαιδευτικό-ακαδημαϊκό και όχι για εμπορικό σκοπό, αφήνοντας έτσι περιθώρια μελλοντικής βελτίωσης και εξέλιξης. Ορισμένες προτάσεις για τη μελλοντική εξέλιξη του συγκεκριμένου παιχνιδιού είναι η προσθήκη περισσότερων σκηνών όπως επίσης και η αύξηση του βαθμού δυσκολίας για την αντιμετώπιση των τεράτων. Τέλος, θα μπορούσαν να προστεθούν περισσότερα features σε αυτό με σκοπό να γίνει πιο ελκυστικό στον χρήστη και στην εμπειρία που θα αποκομίσει.

5. System Full Specifications

- System Model: VivoBook_ASUSLaptop X570ZD/X570ZD Laptop from ASUSTeK Computer Inc.
- CPU: AMD Ryzen 7 2700U with Radeon Vega Mobile Gfx 2.20 GHz
- Display: 15.60 inch 16:9, 1920 x 1080 pixel 141 PPI, IPS, glossy: no
- Operating System: Microsoft Windows 10 Home 64 Bit
- GPU: NVIDIA GeForce GTX 1050
- RAM: 8 GB
- Storage: SSD 256 GB
- Weight: 1.97 kg
- Mouse: Razer DeathAdder Elite
- Keyboard: Razer Blackwidow V3

6. Βιβλιογραφία

1. Haas, J. K. (2014). A History of the Unity Game Engine. Retrieved from https://digitalcommons.wpi.edu/iqp-all/3207
2. Nicoll, B., & Keogh, B. (2019). The Unity Game Engine and the Circuits of Cultural Software. doi:10.1007/978-3-030-25012-6
3. Halpern, J. (2019). Developing 2D Games with Unity. doi:10.1007/978-1-4842-3772-4