Πανεπιστήμιο Δυτικής Αττικής

Σχολή Μηχανικών

Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών

Διπλωματική Εργασία

Ανάπτυξη RPG παιχνιδιού με Unity2D σε C#

Ερμής Καώνης-Τουτουντζής

711161117

Επιβλέπων: **Χρήστος Τρούσσας**

Διπλωματική Εργασία

Ανάπτυξη RPG παιχνιδιού με Unity2D σε C#

Ερμής Καώνης-Τουτουντζής
711161117

Εισηγητής:

ΧΡΗΣΤΟΣ ΤΡΟΥΣΣΑΣ, ΕΠ. ΚΑΘΗΓΗΤΗΣ

Christos Troussas
Digitally signed by Christos Troussas
Date: 2024.10.09 16:04:26 +03'00'

Εξεταστική Επιτροπή:

ΦΟΙΒΟΣ ΜΥΛΩΝΑΣ, ΑΝ. ΚΑΘΗΓΗΤΗΣ

Φοίβος-Απόστολος Μυλωνάς - Ψηφιακά υπογεγραμμένο
2024.10.09 20:19:29 +03'00'

ΑΚΡΙΒΗ ΚΡΟΥΣΚΑ, ΜΕΛΟΣ ΕΔΙΠ

Akrivi Krouska
Digitally signed by Akrivi Krouska
Date: 2024.10.09 17:42:40 +03'00'

Ημερομηνία εξέτασης 09/10/2024

Δήλωση Συγγραφέα Διπλωματικής Εργασίας

Ο κάτωθι υπογεγραμμένος Ερμής Καώνης-Τουτουντζής του Βασίλειου, με αριθμό μητρώου 711161117 φοιτητής του Τμήματος Μηχανικών Πληροφορικής και Υπολογιστών της Σχολής Μηχανικών του Πανεπιστημίου Δυτικής Αττικής, δηλώνω ότι:

«Βεβαιώνω ότι είμαι συγγραφέας αυτής της Διπλωματικής εργασίας και κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.

Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου».

Ο δηλών

Περίληψη

Η παρούσα διπλωματική εργασία περιγράφει την ανάπτυξη ενός δισδιάστατου (2D) RPG (Role Playing Game) παιχνιδιού, χρησιμοποιώντας την μηχανή Unity και τη γλώσσα προγραμματισμού C#. Το παιχνίδι ενσωματώνει μια σειρά από συστήματα και μηχανισμούς που βελτιώνουν την εμπειρία του παίκτη και προσφέρουν ένα δυναμικό περιβάλλον παιχνιδιού.

Η ανάπτυξη ενός παιχνιδιού απαιτεί μια καλά δομημένη και προσεκτική αρχιτεκτονική και υλοποίηση πολλαπλών συστημάτων για να δημιουργηθεί ένας δυναμικός κόσμος. Το μενού και το user interface (UI) αποτελούν τα βασικά εργαλεία για την πλοήγηση και την αλληλεπίδραση του παίκτη στον κόσμο του παιχνιδιού. Μέσα από τα μενού, ο χρήστης μπορεί να διαχειρίζεται τον εξοπλισμό, τα αντικείμενα, και τις ρυθμίσεις, ενώ παράλληλα του παρέχει σημαντικές πληροφορίες.

Ο παίκτης έχει τη δυνατότητα να κινηθεί ελεύθερα στον κόσμο μέσω συστημάτων κίνησης και μιας κάμερας που τον ακολουθεί. Τα συστήματα animations και animator προσθέτουν κινήσεις στον χαρακτήρα, όπως η επίθεση ή η συλλογή αντικειμένων. Ακόμα, διαθέτει ένα σύστημα επιπέδων, όπου ο παίκτης αποκτά εμπειρία (EXP) ολοκληρώνοντας αποστολές και εξουδετερώνοντας εχθρούς. Καθώς ο παίκτης προχωρά στα επίπεδα, αποκτά νέες δυνάμεις και δεξιότητες. Παράλληλα, το Tilemap χρησιμοποιείται για τη δημιουργία του κόσμου του παιχνιδιού, προσφέροντας εργαλεία για την κατασκευή και οργάνωση περιβαλλόντων.

Η αποθήκευση δεδομένων διασφαλίζει ότι η πρόοδος του παίκτη αποθηκεύεται και μπορεί να ανακτηθεί σε επόμενες συνεδρίες παιχνιδιού. Οι εχθροί έχουν τεχνητή νοημοσύνη που τους επιτρέπει να κινούνται, να επιτίθενται και να αλληλεπιδρούν με τον παίκτη, ενώ οι Non-Playable Characters (NPCs) προσφέρουν αποστολές (quests) και ανταμοιβές, καθιστώντας τον κόσμο του παιχνιδιού πιο ζωντανό.

Η μουσική και τα ηχητικά εφέ ενισχύουν την ατμόσφαιρα του παιχνιδιού. Το mini map βοηθάει τον παίκτη να πλοηγείται στον κόσμο και να βρίσκει σημαντικά σημεία ενδιαφέροντος. Τα αντικείμενα (items), όπως όπλα, εργαλεία, και φίλτρα, εμπλουτίζουν το παιχνίδι, παρέχοντας στον παίκτη τα μέσα για να εξελίσσεται και να αλληλεπιδρά με το περιβάλλον και τους εχθρούς. Τέλος, το παιχνίδι ακολουθεί μια κεντρική ιστορία, την οποία ο παίκτης καλείται να ανακαλύψει και να επιλύσει.

Λέξεις Κλειδιά: Role Playing Game, Unity, C#, Non Playable Characters, Αποστολές, Αποθήκευση Προόδου, User Interface

Abstract

This thesis describes the development of a two-dimensional (2D) RPG (Role Playing Game) using the Unity engine and the C# programming language. The game incorporates a series of systems and mechanisms that enhance the player experience and provide a dynamic gaming environment.

The development of a game requires a well-structured and carefully implemented architecture to create a dynamic world. The menu and user interface (UI) serve as the primary tools for player navigation and interaction within the game world. Through the menus, the user can manage equipment, items, and settings while also receiving vital information.

The player can move freely within the world using movement systems and a camera that follows the character. The animation and animator systems add actions to the character, such as attacking or collecting items. Additionally, there is a leveling system where the player gains experience points (EXP) by completing quests and defeating enemies. As the player advances through levels, they acquire new powers and skills. Simultaneously, the Tilemap is used to create the game world, providing tools for constructing and organizing environments.

Data persistence ensures that the player's progress is saved and can be retrieved in future game sessions. Enemies feature artificial intelligence (AI) that allows them to move, attack, and interact with the player, while Non-Playable Characters (NPCs) offer quests and rewards, making the game world more vibrant.

Music and sound effects enhance the game’s atmosphere. The mini-map assists the player in navigating the world and locating important points of interest. Items, such as weapons, tools, and potions, enrich the gameplay by providing the player with the means to progress and interact with the environment and enemies. Lastly, the game follows a central storyline, which the player is encouraged to discover and resolve.

Keyword: Role Playing Game, Unity, C#, Non-Playable Characters, Quests, Save System, User Interface.

Περιεχόμενα

1. Εισαγωγή 1
1.1 Περιγραφή του αντικειμένου της διπλωματικής εργασίας 1
1.2 Ιστορική αναδρομή στα παιχνίδια 1
1.3 Αρχιτεκτονική και Υλοποίηση Συστήματος 1
2. Θεωρητικό Υπόβαθρο και Ανασκόπηση Βιβλιογραφίας 3
2.1 Unity Layout 3
2.1.1 Scene Panel 3
2.1.2 Game Panel 4
2.1.3 Hierarchy Panel 4
2.1.4 Inspector Panel 4
2.1.5 Project Panel 4
2.1.6 Console Panel 5
2.1.7 Tile Palette Panel 5
2.1.8 Animator Panel 5
2.1.9 Animation Panel 5
2.1.10 Audio Mixer Panel 5
2.2 Σκηνή, Game Objects και Components 6
2.2.1 Σκηνή 6
2.2.2 Transform Component 7
2.2.3 Names και Tags 7
2.2.4 Layers 7
2.2.5 Parent-Child Relationship 7
2.2.6 Prefabs 8
2.3 Οπτικά Στοιχεία του περιβάλλοντος 2D 8
2.3.1 Sprites 8
2.3.2 Sprite Editor 9
2.3.3 Tilemaps 9
2.3.4 Sorting Layers και Order in Layer 9
2.3.5 Κάμερα στο 2D Περιβάλλον 10
2.3.6. Pixel Perfect Rendering 10
2.4 Σύστημα 2D Animation 10
2.4.1 Animator Controller 11
2.4.2 Sprite-Based Animation 11

2.4.3 Frame-by-Frame Animation 11
2.4.4 Animation Events 12
2.4.5 Scripting και Animation Control 12
2.5 Φυσική (Physics) στο 2D 13
2.5.1 Rigidbody 2D 13
2.5.2 Colliders 2D 13
2.5.3 Physics Materials 2D 14
2.5.4 Physics Simulation (Προσομοίωση Φυσικής) 14
2.5.5 Αλληλεπίδραση Με Σενάρια (Scripting Physics) 15
2.6 User Interface (UI) 16
2.6.1 Στοιχεία UI 16
2.6.2 Σχεδίαση UI 17
2.6.3 Διαχείριση UI Events 17
2.6.4 Layout Groups 17
2.7 Render Texture 18
2.7.1 Χρήσεις του Render Texture 18
2.7.2 Δημιουργία και Χρήση Render Textures 18
2.7.3 Χαρακτηριστικά και Ιδιότητες των Render Textures 19
2.8 Unity Asset Store 19
2.8.1 Χαρακτηριστικά του Unity Asset Store: 20
2.8.2 Package Manager 20
2.8.3 Text Mesh Pro 21
2.8.4 Art και Sound Assets 22
2.8.5 Path Creator 22
2.9 Visual Studio και C# σε Συνδυασμό με το Unity 23
2.9.1 Visual Studio 23
2.9.2 Γλώσσα Προγραμματισμού C# 24
2.9.3 Συνεργασία του Unity με το Visual Studio και την C# 24
2.10 Η Κλάση MonoBehaviour και Βασικές Μέθοδοι 25
2.10.1 Η κλάση MonoBehaviour 25
2.10.2 Δήλωση GameObjects και των Components 26
2.10.3 Βασικές Μέθοδοι της MonoBehaviour 26
2.10.4 Βασικές Μέθοδοι σε Συνεργασία με GameObjects 27
2.10.5 Μέθοδοι που Επηρεάζουν τον Χρόνο 28
2.10.6 Μέθοδοι για Εντοπισμού εισόδου 28

2.10.7 Χειρισμός Αρχείων 29
2.10.8 Enum 29
2.10.9 Events 30
2.11 Scriptable Objects 30
2.11.1 Χαρακτηριστικά και Πλεονεκτήματα 31
2.11.2 Χρήση των Scriptable Objects 31
2.11.3 Δημιουργία των Scriptable Objects 31
2.11.4 Scriptable Objects σε Συνδυασμό με άλλα Συστήματα 32
2.11.5 Σύγκριση με τα MonoBehaviour 32
2.12 Dictionaries 32
2.13 Optimization στο Unity2D 33
3. Μεθοδολογία 35
3.1 Συλλογή και Ανάλυση Απαιτήσεων 35
3.2 Σχεδιασμός Συστήματος και Αρχιτεκτονική 35
3.3 Υλοποίηση 35
3.4 Έλεγχος και Δοκιμές 36
3.5 Συνεχής Ανατροφοδότηση και Βελτιστοποίηση 36
3.6 Ολοκλήρωση 36
4. Αρχιτεκτονική και Υλοποίηση Συστήματος 37
4.1 Μενού και Διεπαφή Χρήστη (User Interface) 37
4.1.1 Μενού 37
4.1.2 Διεπαφή Χρήστη User Interface 38
4.2 Παίκτης 46
4.2.1 Κίνηση 46
4.2.2 Κάμερα 46
4.2.3 Χρήση όπλου/εργαλείου 47
4.2.4 Ζωή και Mana 49
4.2.5 Επίπεδο και Stats 50
4.2.6 Πόροι (Resources) 50
4.3 Animations και Animator 51
4.3.1 Animations 51
4.3.2 Animator 51
4.4 Tilemap 52
4.5 Αποθήκευση δεδομένων Data Persistence 55
4.6 Εχθροί 55

4.6.1 AI και Κίνηση 55
4.6.2 Εντοπισμός Παίκτη και Επίθεση 56
4.6.3 Ζωή και Ανταμοιβή 57
4.7 Non Playable Characters 57
4.7.1 Interactable NPCs 57
4.7.2 NPCs που περιφέρονται σε μονοπάτι 58
4.7.3 Σταθερά NPCs με Animations 59
4.7.4 Ζώα 59
4.8 Μουσική και Ηχητικά Εφέ 60
4.9 Mini Map 60
4.10 Αποστολές (Quests) 61
4.11 Αντικείμενα (Items) 63
4.12 Σκοπός του παιχνιδιού και Ιστορία 65
5. Συμπεράσματα και Επεκτάσεις Εφαρμογής 66
5.1 Συμπεράσματα και Επεκτασιμότητα 66

Κατάλογος Εικόνων

Εικόνα 1. Sprite Sheet που περιέχει πολλαπλά sprites 9
Εικόνα 2. Παράθυρο Ρυθμίσεων 38
Εικόνα 3. Παράθυρο που φαίνεται η Ζωή το Mana και το Επίπεδο 38
Εικόνα 4. Το παράθυρο Toolbar 39
Εικόνα 5. Το παράθυρο του Inventory 39
Εικόνα 6. Παράθυρο ειδοποίησης του παίκτη για αλληλεπίδραση με NPC 40
Εικόνα 7. Το παράθυρο του χαρακτήρα 41
Εικόνα 8. Παράθυρο Δεξιοτήτων 42
Εικόνα 9. Παράθυρο Διαλόγου με NPC 43
Εικόνα 10. Παράθυρο ανταμοιβής 43
Εικόνα 11. Παράθυρα Μαγαζιών 44
Εικόνα 12. Παράθυρο Αποστολών 44
Εικόνα 13. Παράθυρο μενού Παύσης 45
Εικόνα 14. Το script Tooltip όπως φαίνεται στον Inspector 45
Εικόνα 15. Τα παράθυρα Tooltip από διαφορετικά αντικείμενα 46
Εικόνα 16. Απεικόνιση του Script στον Inspector 47
Εικόνα 17. Attack Animation με Events 47
Εικόνα 18. Το εύρος εντοπισμού 48
Εικόνα 19. Το παράθυρο που εμφανίζεται όταν ο παίκτης φτάσει 0 ζωή 49
Εικόνα 20. Απεικόνιση των χαρακτηριστικών της εικόνας της ζωής 50
Εικόνα 21. Δέντρο και Πέτρωμα όπου ο παίκτης μπορεί να μαζέψει Πόρους 51
Εικόνα 22. Animator με Clip Animations 52
Εικόνα 23. Tile Palette με assets 53
Εικόνα 24. Το Grid με τα Tilemaps που περιέχει όπως φαίνεται στην Ιεραρχία 53
Εικόνα 25. Map Manager Script 54
Εικόνα 26. Scriptable Object για τα δεδομένα των μονοπατιών 54
Εικόνα 27. Εχθρός με 2 circle colliders και ένα box 56
Εικόνα 28. Interactable NPCs με τα trigger colliders 58
Εικόνα 29. Τα μονοπάτια ως πράσινες γραμμές που ακολουθούν τα NPC 58
Εικόνα 30. Σταθερά NPCs με animations 59
Εικόνα 31. Ζώα που περιφέρονται με AI 60
Εικόνα 32. Απεικόνιση του Mini Map με σύμβολα για τα σημαντικά αντικείμενα 61
Εικόνα 33. Το script Quest1 όπως φαίνεται με τα πεδία του 62
Εικόνα 34. Διαφορετικά Scriptable Objects όπως φαίνονται στον Inspector 64
Εικόνα 35. Παράθυρο με το κείμενο της Ιστορίας 65

Συντομογραφίες

RPG Role Playing Game

NPC Non-Playable Character

SFX Sound Effects

UI User Interface

EXP Experience

Κεφάλαιο 1

1. Εισαγωγή

1.1 Περιγραφή του αντικειμένου της διπλωματικής εργασίας

Η παρούσα διπλωματική εργασία εξετάζει την ανάπτυξη ενός δισδιάστατου (2D) RPG (Role Playing Game) παιχνιδιού χρησιμοποιώντας τη μηχανή Unity και τη γλώσσα προγραμματισμού C#. Η ανάπτυξη ενός παιχνιδιού είναι μια σύνθετη διαδικασία που απαιτεί προσεκτικό σχεδιασμό και συνδυασμό πολλών τεχνικών και εργαλείων. Η παρούσα αναφορά επικεντρώνεται στην ανάλυση της αρχιτεκτονικής και της υλοποίησης ενός τέτοιου παιχνιδιού, χρησιμοποιώντας τη μηχανή Unity και τη γλώσσα προγραμματισμού C#. Το παιχνίδι βασίζεται σε διάφορα συστήματα, τα οποία συνεργάζονται μεταξύ τους για να προσφέρουν μια εμπλουτισμένη εμπειρία στον χρήστη.

Το 2D RPG είναι ένα είδος παιχνιδιού που συνδυάζει αφήγηση, στρατηγική και εξερεύνηση, προσφέροντας στους παίκτες την ευκαιρία να ζήσουν μια εικονική περιπέτεια μέσα σε ένα φανταστικό δισδιάστατο κόσμο. Με την ανάπτυξη αυτού του παιχνιδιού, επιδιώκεται η δημιουργία μιας ολοκληρωμένης και δυναμικής εμπειρίας, που θα κρατήσει το ενδιαφέρον των χρηστών και θα τους ενθαρρύνει να εξερευνήσουν, να προχωρήσουν και να ανακαλύψουν.

1.2 Ιστορική αναδρομή στα παιχνίδια

Η ιστορία των RPG παιχνιδιών εκτείνεται σε πολλές δεκαετίες, με τις ρίζες τους να εντοπίζονται σε επιτραπέζια παιχνίδια ρόλων. Σταδιακά, με την εξέλιξη της τεχνολογίας, τα RPG μεταφέρθηκαν σε ψηφιακή μορφή, με σημαντικές αναπτύξεις σε 2D και 3D περιβάλλοντα. Το 2D RPG παιχνίδι αποκτά ιδιαίτερη αξία, καθώς συνδυάζει την απλότητα του σχεδιασμού με την πλούσια αφήγηση, δημιουργώντας μια μοναδική εμπειρία για τους παίκτες.

1.3 Αρχιτεκτονική και Υλοποίηση Συστήματος

Η ανάπτυξη του παιχνιδιού περιλαμβάνει την ανάλυση της αρχιτεκτονικής και της υλοποίησης πολλών συστημάτων, τα οποία συνεργάζονται για να προσφέρουν μια ρεαλιστική εμπειρία. Στο μενού και το user interface (UI), ο χρήστης μπορεί να διαχειρίζεται τον εξοπλισμό του, να παρακολουθεί τα στατιστικά του και να αλληλεπιδρά με διάφορες επιλογές. Ο παίκτης μπορεί να κινείται ελεύθερα στον κόσμο του παιχνιδιού, με συστήματα κίνησης και κάμερας που προσφέρουν δυναμική θέαση της δράσης.

Η υλοποίηση του παιχνιδιού ξεκινά με τον σχεδιασμό των βασικών μηχανισμών που διέπουν τη λειτουργία του. Η δημιουργία ενός λειτουργικού και εύχρηστου μενού και user interface (UI) είναι καίριας σημασίας, καθώς επιτρέπει στον παίκτη να διαχειρίζεται τον χαρακτήρα του, να παρακολουθεί τα στατιστικά του, να αλληλεπιδρά με τον εξοπλισμό και να αποκτά πληροφορίες για την πορεία του παιχνιδιού. Το UI αποτελεί το πρώτο σημείο επαφής του παίκτη με το παιχνίδι και πρέπει να είναι σαφές και φιλικό προς τον χρήστη.

Ο παίκτης αποτελεί τον κεντρικό άξονα του παιχνιδιού, και η ανάπτυξη των μηχανισμών που αφορούν την κίνηση, τη χρήση όπλων και εργαλείων και άλλων συστημάτων, παίζουν καθοριστικό ρόλο στη

συνολική εμπειρία. Επιπλέον, η προσθήκη ενός συστήματος επιπέδων και στατιστικών (Stats), δίνει την ευκαιρία στον παίκτη να εξελιχθεί και να γίνει ισχυρότερος καθώς προχωρά στο παιχνίδι.

Η προσθήκη animation προσδίδει ζωντάνια στους χαρακτήρες και τις δράσεις τους, ενώ η χρήση του Tilemap επιτρέπει τη δημιουργία ενός πλούσιου και λεπτομερούς περιβάλλοντος, που προσκαλεί τον παίκτη να εξερευνήσει τον κόσμο του παιχνιδιού.

Η υλοποίηση ενός συστήματος αποθήκευσης δεδομένων (Data Persistence) εξασφαλίζει την πρόοδο του παίκτη, ενώ οι εχθροί και οι Non Playable Characters (NPCs) προσφέρουν τις κύριες προκλήσεις και αλληλεπιδράσεις κατά τη διάρκεια της περιπέτειας. Τέλος, στοιχεία όπως η μουσική, τα ηχητικά εφέ, το Mini Map, οι αποστολές (Quests) και τα αντικείμενα (Items) ολοκληρώνουν τον κόσμο του παιχνιδιού, παρέχοντας στο χρήστη μια εμπλουτισμένη εμπειρία που εξελίσσεται και προκαλεί συνεχώς το ενδιαφέρον του.

Συνολικά, η ανάπτυξη αυτού του 2D RPG παιχνιδιού στο Unity αποσκοπεί στη δημιουργία μιας ελκυστικής και αλληλεπιδραστικής εμπειρίας για τους χρήστες, η οποία θα τους ενθουσιάσει και θα τους παρακινήσει να εξερευνήσουν τον κόσμο που έχει δημιουργηθεί.

Κεφάλαιο 2

2. Θεωρητικό Υπόβαθρο και Ανασκόπηση Βιβλιογραφίας

Στο κεφάλαιο αυτό θα αναλυθούν οι πληροφορίες που απαιτούνται να γνωρίζει κάποιος για να αναπτύξει μία εφαρμογή, παιχνίδι, στο Unity Engine. Οι πληροφορίες αφορούν κυρίως το Unity Engine 2D, δηλαδή θα αναλυθούν περισσότερο οι λεπτομέρειες για την δημιουργία ενός δισδιάστατου παιχνιδιού. Πέρα από το Unity, χρησιμοποιείται και το πρόγραμμα Visual Studio, το οποίο χρησιμοποιείται για τον προγραμματισμό των συστημάτων και της λογικής του παιχνιδιού, στην γλώσσα C#. Επομένως θα γίνει αναφορά και σε κάποιες έννοιες προγραμματισμού.

Το Unity Engine είναι μια από τις πιο διαδεδομένες και δημοφιλείς μηχανές ανάπτυξης παιχνιδιών, χρησιμοποιούμενη τόσο από ανεξάρτητους προγραμματιστές (indie developers) όσο και από μεγάλες εταιρείες στον τομέα του gaming και άλλων εφαρμογών πολυμέσων. Είναι μια ολοκληρωμένη πλατφόρμα ανάπτυξης που υποστηρίζει την ανάπτυξη 2D και 3D παιχνιδιών, εφαρμογών επαυξημένης πραγματικότητας (AR) και εικονικής πραγματικότητας (VR), καθώς και άλλων διαδραστικών εμπειριών. Η μηχανή είναι ιδιαίτερα δημοφιλής λόγω της ευκολίας χρήσης της, της δυνατότητας δημιουργίας παιχνιδιών για πολλές πλατφόρμες (multiplatform support), και της ισχυρής κοινότητας χρηστών και υποστήριξης.

Το Unity 2D είναι μια πλατφόρμα ενσωματωμένη στο Unity engine, σχεδιασμένη για την ανάπτυξη δισδιάστατων παιχνιδιών (2D games). Παρέχει ένα σύνολο εργαλείων και τεχνολογιών που επιτρέπουν την ανάπτυξη 2D παιχνιδιών γρήγορα και αποτελεσματικά, χωρίς να χρειάζεται να ασχοληθεί κάποιος με περιττές πολυπλοκότητες που αφορούν τον 3D κόσμο. Παρόλο που το Unity αρχικά σχεδιάστηκε για 3D ανάπτυξη, η ενσωμάτωση του Unity 2D την έκανε ιδανική για ένα μεγάλο εύρος παιχνιδιών και εφαρμογών, από απλά 2D platformers έως σύνθετα παιχνίδια στρατηγικής και παζλ.

2.1 Unity Layout

Στο Unity, το layout της ανάπτυξης είναι οργανωμένο σε διάφορα panels που βοηθούν την διαχείριση σε διαφορετικές πτυχές του έργου. Ακολουθεί μία ανάλυση των panels που χρησιμοποιήθηκαν κατά την δημιουργία του παιχνιδιού.

2.1.1 Scene Panel

Το Scene panel είναι το κύριο εργαλείο για την οπτική διαχείριση του κόσμου. Εδώ μπορούμε να δούμε και να επεξεργαστούμε τα αντικείμενα της σκηνής σε 3D ή 2D περιβάλλον. Μπορούν να γίνουν οι εξής λειτουργίες σε αυτό:

- **Πλοήγηση:** Επιτρέπει στους χρήστες να πλοηγούνται στον κόσμο, χρησιμοποιώντας εργαλεία όπως το Move, Rotate και Scale.
- **Επιλογή Αντικειμένων:** Μπορεί να γίνει επιλογή των αντικειμένων στη σκηνή για να επεξεργαστούν ή να προβληθούν.
- **Δημιουργία Αντικειμένων:** Μπορούν να προστεθούν νέα αντικείμενα και να τοποθετηθούν στη σκηνή.

- **Viewport Controls:** Παρέχει εργαλεία για ζουμ, περιστροφή και κίνηση στην οθόνη.

2.1.2 Game Panel

Το Game panel επιτρέπει στους χρήστες να προβάλλουν το παιχνίδι όπως θα εμφανίζεται στους παίκτες, παρέχοντας μια προεπισκόπηση της τελικής εμπειρίας. Μπορούν να γίνουν οι εξής λειτουργίες σε αυτό:

- **Play Mode:** Όταν πατηθεί το κουμπί "Play", μπορεί να γίνει δοκιμή του παιχνιδιού σε πραγματικό χρόνο.
- **Αλληλεπίδραση:** Επιτρέπει την αλληλεπίδραση με το παιχνίδι κατά την εκτέλεση, όπως θα έκανε και ένας χρήστης.
- **Επαναλαμβανόμενη Δοκιμή:** Διευκολύνει τη δοκιμή και τη ρύθμιση διαφόρων πτυχών του παιχνιδιού, όπως με τη λειτουργία παύσης, κάτι που δίνει την δυνατότητα να γίνει δυναμική αλλαγή διάφορων στοιχείων, επιτρέποντας έτσι να φανούν οι αλλαγές σε πραγματικό χρόνο.

2.1.3 Hierarchy Panel

Το Hierarchy panel εμφανίζει όλα τα αντικείμενα (GameObjects) που υπάρχουν στη σκηνή σε μορφή δέντρου. Μπορούν να γίνουν οι εξής λειτουργίες σε αυτό:

- **Δημιουργία και Διαχείριση Αντικειμένων:** Μπορούν να προστεθούν, να αφαιρεθούν ή να οργανωθούν τα αντικείμενα σε ιεραρχία.
- **Parenting:** Υποστηρίζει τη δυνατότητα καθορισμού γονικών και παιδικών σχέσεων ανάμεσα στα αντικείμενα.
- **Search Functionality:** Επιτρέπει την αναζήτηση συγκεκριμένων αντικειμένων για γρηγορότερη πλοήγηση.

2.1.4 Inspector Panel

Το Inspector panel εμφανίζει τις ιδιότητες και τα χαρακτηριστικά του επιλεγμένου αντικειμένου από το Hierarchy panel. Μπορούν να γίνουν οι εξής λειτουργίες σε αυτό:

- **Αλλαγή Ιδιοτήτων:** Μπορεί να γίνει επεξεργασία των ρυθμίσεων και των παραμέτρων των GameObjects, όπως τις θέσεις, τις κλίμακες, τα components κ.λπ.
- **Προσαρμοσμένα Scripts:** Εμφανίζει τις παραμέτρους που έχουν οριστεί σε προσαρμοσμένα scripts, επιτρέποντας εύκολη και γρήγορη παραμετροποίηση.
- **Δημιουργία Component:** Επιτρέπει την προσθήκη νέων components στα GameObjects για να επεκταθεί η λειτουργικότητά τους.

2.1.5 Project Panel

Το Project panel εμφανίζει όλα τα αρχεία και τους πόρους του έργου, οργανωμένα σε φακέλους. Μπορούν να γίνουν οι εξής λειτουργίες σε αυτό:

- **Διαχείριση Πόρων:** Μπορεί να γίνει οργάνωση των πόρων (εικόνες, assets, scripts, κ.λπ.) σε φακέλους για εύκολη πρόσβαση. Δεδομένου ότι υπάρχει ένας πολύ μεγάλος αριθμός αρχείων, είναι μια καλή τακτική να γίνεται μια οργάνωση αυτών, μέσα σε φακέλους με σωστές ονομασίες που να αντιπροσωπεύουν τα περιεχόμενα τους.
- **Εύρεση Πόρων:** Επιτρέπει την αναζήτηση συγκεκριμένων αρχείων ή πόρων.
- **Προσθήκη Νέων Πόρων:** Μπορούν να προστεθούν νέα αρχεία ή φάκελοι, όπως scripts ή assets απευθείας από αυτό το panel.

2.1.6 Console Panel

Το Console panel εμφανίζει μηνύματα, προειδοποιήσεις και σφάλματα που προκύπτουν κατά την εκτέλεση ή την ανάπτυξη του παιχνιδιού. Είναι ένα πολύ σημαντικό εργαλείο που μας βοηθάει στην αποσφαλμάτωση του κώδικα, καθώς εμφανίζει μηνύματα σφαλμάτων, αλλά και απλά μηνύματα που μπορούμε να εμφανίσουμε μέσα από scripts. Τα μηνύματα σφαλμάτων και προειδοποιήσεων εμφανίζονται είτε από μόνα τους όταν το Unity εντοπίσει κάτι λάθος, είτε μέσα από script κάνοντας χρήση το Debug.LogError() ή Debug.LogWarning(). Το σκέτο Debug.Log(), εμφανίζει ένα απλό μήνυμα.

2.1.7 Tile Palette Panel

Το Tile Palette panel χρησιμοποιείται για τη δημιουργία και την επεξεργασία tilemaps, που είναι χρήσιμα για τη δημιουργία 2D επιπέδων. Επιτρέπει την σχεδίαση και την τοποθέτηση tiles στον κόσμο του παιχνιδιού. Μπορεί επίσης να γίνει οργάνωση των tiles σε παλέτες για γρηγορότερη πρόσβαση. Μπορεί ακόμα να γίνει επεξεργασία αυτών, όπως η περιστροφή τους ή η αλλαγή μεγέθους.

2.1.8 Animator Panel

Το Animator panel χρησιμοποιείται για τη διαχείριση και τη δημιουργία animations για GameObjects. Υπάρχει η δυνατότητα δημιουργία καταστάσεων (states), όπου το κάθε ένα μπορεί να είναι κάποιο διαφορετικό animation clip. Κάνοντας χρήση των μεταβάσεων (transistions) μπορεί να γίνεται εναλλαγή από κατάσταση σε κατάσταση, χρησιμοποιώντας έτσι μια λογική State Machine.

2.1.9 Animation Panel

Το Animation panel παρέχει εργαλεία για τη δημιουργία και την επεξεργασία animations για αντικείμενα. Επιτρέπει την προσθήκη keyframes, για να καθορίζει τον τρόπο με τον οποίο αλλάζουν οι ιδιότητες ενός αντικειμένου σε χρόνο. Ακόμα μπορεί να γίνει η δημιουργία και η διαχείριση animation clips για την χρήση τους σε GameObjects.

2.1.10 Audio Mixer Panel

Το Audio Mixer panel επιτρέπει τον διαχείριση και την ρύθμιση των ηχητικών εφέ και της μουσικής του παιχνιδιού. Υπάρχουν διαφορετικά Audio Channels, που επιτρέπει την διαχείριση διαφορετικών

καναλιών ήχου για την μουσική, τα ηχητικά εφέ και τις φωνές. Ακόμα μπορεί να γίνει εφαρμογή διάφορων εφέ για την καλύτερη ποιότητα ήχου.

2.2 Σκηνή, Game Objects και Components

Οι σκηνές (Scenes) στο Unity αποτελούν ένα από τα πιο θεμελιώδη στοιχεία της πλατφόρμας και λειτουργούν ως το κύριο περιβάλλον στο οποίο διαδραματίζεται το παιχνίδι ή η εφαρμογή. Κάθε σκηνή είναι ένα αυτόνομο σύνολο που περιλαμβάνει τα αντικείμενα, τα σενάρια και τα συστήματα που καθορίζουν τη συμπεριφορά και την εμφάνιση του κόσμου του παιχνιδιού.

Τα Game Objects αποτελούν την καρδιά κάθε σκηνής. Είναι οι βασικές μονάδες του κόσμου και μπορούν να αντιπροσωπεύουν οτιδήποτε, από χαρακτήρες, αντικείμενα, και εχθρούς μέχρι περιβάλλοντα, φώτα, κάμερες και πολλά άλλα. Κάθε Game Object ξεκινά ως μια κενή μονάδα, και η συμπεριφορά του ορίζεται με τη χρήση Components. Τα Game Objects είναι στην ουσία οι "δομικές πέτρες" του παιχνιδιού, χωρίς όμως τα Components δεν θα έχουν καμία λειτουργία.

Τα Components είναι τα στοιχεία που δίνουν στα Game Objects τις ιδιότητές τους και καθορίζουν τη συμπεριφορά τους. Κάθε Game Object μπορεί να αποτελείται από ένα ή περισσότερα components, και αυτό δίνει τη δυνατότητα για απεριόριστες παραλλαγές και προσαρμογές στα αντικείμενα. Παρακάτω, θα δούμε ορισμένα βασικά components και έννοιες που συνδέονται με τα Game Objects.

Τα Game Objects σε συνδυασμό με τα Components τους, προσφέρουν έναν εξαιρετικά ευέλικτο τρόπο για τη δημιουργία και τον έλεγχο στοιχείων στον κόσμο του παιχνιδιού.

2.2.1 Σκηνή

Μια σκηνή είναι ένα περιβάλλον εργασίας που περιλαμβάνει όλα τα Game Objects (αντικείμενα του παιχνιδιού) και τους κανόνες που διέπουν τη συμπεριφορά τους. Είναι δηλαδή σαν ένα «δοχείο» που φιλοξενεί αντικείμενα, σενάρια, φώτα, κάμερες, και γενικά τα πάντα που είναι απαραίτητα για τη λειτουργία ενός επιπέδου, ενός μενού ή ενός περιβάλλοντος.

Το Unity επιτρέπει τη χρήση πολλαπλών σκηνών ταυτόχρονα, οι οποίες μπορούν να φορτωθούν παράλληλα ή σταδιακά. Για παράδειγμα, ένα μεγάλο ανοιχτό παιχνίδι μπορεί να χωριστεί σε πολλές σκηνές που φορτώνονται ή εκφορτώνονται ανάλογα με την τοποθεσία του παίκτη.

Οι σκηνές μπορούν επίσης να χρησιμοποιηθούν για τη δημιουργία διαφόρων επιπέδων, μενού, περιοχών, ή ακόμα και διαφορετικών τμημάτων του κόσμου που αλλάζουν κατά τη διάρκεια του παιχνιδιού.

Το Unity προσφέρει διάφορες μεθόδους για τη φόρτωση σκηνών μέσω του κώδικα, όπως το SceneManager.LoadScene() και το SceneManager.LoadSceneAsync(). Δίνεται δηλαδή η δυνατότητα να φορτώνουν ή να μεταβαίνουν ανάμεσα σε σκηνές όταν ο παίκτης ολοκληρώνει ένα επίπεδο ή ξεκινά μια νέα περιοχή.

Η διαχείριση των σκηνών στο Unity είναι κεντρικής σημασίας για τη δομή και τη ροή ενός παιχνιδιού. Με τις σκηνές, μπορεί να καθοριστούν διαφορετικά τμήματα του παιχνιδιού, να δημιουργηθούν δυναμικές μεταβάσεις και να παρέχουν στον παίκτη νέες προκλήσεις και εμπειρίες. Κάθε σκηνή είναι ένας ξεχωριστός

κόσμος που μπορεί να φορτωθεί, να αποθηκευτεί και να ελεγχθεί ανεξάρτητα, επιτρέποντας εύκολη ανάπτυξη και συντήρηση.

2.2.2 Transform Component

Το πιο σημαντικό component που υπάρχει σε κάθε Game Object. Ελέγχει τη θέση, την περιστροφή και το μέγεθος του αντικειμένου στον κόσμο του παιχνιδιού. Οι τρεις βασικές παράμετροι του Transform είναι:

- **Position**: Η θέση του αντικειμένου στο χώρο (x, y, z).
- **Rotation**: Ο προσανατολισμός του αντικειμένου.
- **Scale**: Το μέγεθος του αντικειμένου σε κάθε άξονα.

Το Transform μπορεί να χρησιμοποιηθεί για να κινήσει, να περιστρέψει ή να αλλάξει το μέγεθος του αντικειμένου κατά την εκτέλεση του παιχνιδιού.

2.2.3 Names και Tags

Κάθε Game Object έχει ένα όνομα (Name) που βοηθά στον προσδιορισμό του μέσα στη σκηνή ή στον κώδικα. Είναι χρήσιμο για την αναφορά σε συγκεκριμένα αντικείμενα, είτε κατά τη διάρκεια της ανάπτυξης είτε κατά την εκτέλεση του παιχνιδιού.

Τα Tags είναι ετικέτες που ανατίθενται σε Game Objects για να τα ταξινομήσουμε σε κατηγορίες. Επιτρέπει την γρήγορη αναγνώριση μιας ομάδας αντικειμένων με παρόμοια χαρακτηριστικά. Για παράδειγμα, μπορεί να οριστεί ένα "Player" tag για τον χαρακτήρα του παίκτη, ένα "Enemy" tag για τους εχθρούς, ή ένα "Collectible" tag για αντικείμενα που ο παίκτης μπορεί να συλλέξει. Η χρήση των Tags επιτρέπει γρήγορη αναγνώριση και εύκολη αναζήτηση αντικειμένων μέσω κώδικα.

2.2.4 Layers

Τα Layers είναι παρόμοια με τα tags, αλλά χρησιμοποιούνται κυρίως για τον έλεγχο της αλληλεπίδρασης μεταξύ των αντικειμένων και της απόδοσης. Κάθε Game Object μπορεί να ανήκει σε ένα layer, και τα layers χρησιμοποιούνται για να περιορίσουν τις συγκρούσεις ή τις ορατές αλληλεπιδράσεις. Για παράδειγμα, οι κάμερες μπορούν να ρυθμιστούν ώστε να βλέπουν μόνο συγκεκριμένα layers. Επίσης, το σύστημα Physics μπορεί να χρησιμοποιήσει layers για να καθορίσει ποια αντικείμενα μπορούν να συγκρούονται μεταξύ τους.

2.2.5 Parent-Child Relationship

Στο Unity, τα Game Objects μπορούν να οργανωθούν σε ιεραρχία γονέων και παιδιών. Ένα Parent Game Object είναι το κεντρικό αντικείμενο, και κάθε Child Game Object είναι συνδεδεμένο με αυτό. Τα Child αντικείμενα κληρονομούν την κίνηση και τον προσανατολισμό από το Parent.

Για παράδειγμα, αν ένα Parent Game Object είναι μια πλατφόρμα και μετακινηθεί, όλα τα Child αντικείμενα (π.χ. επιπλέον στοιχεία πάνω στην πλατφόρμα) θα μετακινηθούν μαζί του. Αυτή η ιεραρχία διευκολύνει τη διαχείριση σύνθετων σκηνών και αντικειμένων που αποτελούνται από πολλά μέρη.

2.2.6 Prefabs

Τα Prefabs είναι ένας ισχυρός τρόπος επαναχρησιμοποίησης αντικειμένων. Ένα Prefab είναι ένα προσχεδιασμένο Game Object (με όλα του τα Components) που μπορεί να αποθηκευτεί και να χρησιμοποιηθεί επανειλημμένα σε διάφορα σημεία στο παιχνίδι. Για παράδειγμα, αν έχουμε δημιουργήσει έναν εχθρό και τον αποθηκεύσουμε ως Prefab, μπορούμε στη συνέχεια να τον επαναχρησιμοποιήσουμε σε πολλαπλές σκηνές.

Μία πολύ σημαντική λειτουργία και ο κυρίως λόγος χρήσης των Prefabs, είναι πως όταν τροποποιείτε το αρχικό Prefab, όλες οι εμφανίσεις του στον κόσμο του παιχνιδιού ενημερώνονται αυτόματα. Από το προηγούμενο παράδειγμα, αν θέλαμε να αλλάξουμε κάτι σε έναν εχθρό, π.χ. την ζωή του, αλλάζοντας το μόνο μια φορά από το Prefab, αλλάζουν όλοι, ενώ αν τα είχαμε τοποθετήσει ως ξεχωριστά αντικείμενα θα έπρεπε να αλλάξει χειροκίνητα το καθένα ξεχωριστά.

2.3 Οπτικά Στοιχεία του περιβάλλοντος 2D

Τα Στοιχεία του 2D Environment στην Unity αναφέρονται στα βασικά εργαλεία και δυνατότητες που χρησιμοποιούνται για τη δημιουργία του κόσμου ενός 2D παιχνιδιού. Το Unity 2D παρέχει στους προγραμματιστές τα μέσα για να αναπαραστήσουν γραφικά αντικείμενα, να τα τοποθετήσουν μέσα στη σκηνή και να τα διαχειριστούν οπτικά.

2.3.1 Sprites

Τα sprites είναι η θεμελιώδης μονάδα γραφικών σε ένα 2D παιχνίδι. Στην ουσία, ένα sprite είναι μια εικόνα ή ένα κινούμενο γραφικό που χρησιμοποιείται για να αναπαραστήσει έναν χαρακτήρα, αντικείμενο, φόντο ή οποιοδήποτε άλλο οπτικό στοιχείο του παιχνιδιού.

- Εισαγωγή Εικόνων ως Sprites: Αρχικά εισάγονται οι εικόνες (PNG, JPEG, κ.λπ.) και μετατρέπονται σε sprites. Αφού εισαχθούν, οι εικόνες γίνονται μέρος του Unity project και μπορούν να τοποθετηθούν στη σκηνή.
- Transform και Ρυθμίσεις: Τα sprites, όπως και όλα τα αντικείμενα που προστίθενται μέσα στην σκηνή, έχουν ιδιότητες όπως θέση, περιστροφή και μέγεθος μέσω του component Transform. Αυτές οι ιδιότητες επιτρέπουν την ελεύθερη τοποθέτηση των sprites στον 2D χώρο, διασφαλίζοντας ότι τα αντικείμενα εμφανίζονται εκεί που πρέπει.
- Sprite Renderer: Το Sprite Renderer component είναι υπεύθυνο για την απεικόνιση του sprite στην οθόνη. Μέσω του Sprite Renderer, μπορεί να αλλάξει η εμφάνιση του sprite (π.χ., αλλαγή χρώματος, προσθήκη διαφάνειας) και να καθοριστεί ποιες εικόνες εμφανίζονται.

2.3.2 Sprite Editor

Το Sprite Editor είναι ένα ισχυρό εργαλείο που επιτρέπει τον κατακερματισμό και την επεξεργασία μεγάλων εικόνων σε μικρότερα τμήματα ή sprites. Είναι ιδιαίτερα χρήσιμο όταν χρησιμοποιούνται sprite sheets (εικόνες που περιέχουν πολλές εικόνες σε ένα αρχείο) για να δημιουργηθούν animations.

Ένα παράδειγμα μπορούμε να το δούμε παρακάτω, χρησιμοποιώντας έναν από τους εχθρούς που υπάρχουν στο παιχνίδι, όπου βλέπουμε μία εικόνα η οποία περιέχει έναν χαρακτήρα σε διαφορετικές πόζες. Κάνοντας χρήση του Sprite Editor, μπορούμε να κόψουμε αυτή την εικόνα στα σημεία που επιθυμούμε, δημιουργώντας έτσι ένα sprite που μέσα του περιέχει πολλαπλά sprites και μπορούμε να τα χρησιμοποιήσουμε ξεχωριστά.

Εικόνα 1. Sprite Sheet που περιέχει πολλαπλά sprites

2.3.3 Tilemaps

Το Tilemap σύστημα του Unity είναι ένας τρόπος για να δημιουργηθούν περιβάλλοντα που αποτελούνται από μικρά επαναλαμβανόμενα πλακίδια (tiles), τα οποία τοποθετούνται σε ένα πλέγμα (grid). Τα tilemaps είναι ιδανικά για τη δημιουργία μεγάλων δισδιάστατων επιπέδων με οικονομικό τρόπο, όπως σε platformers, RPGs, ή puzzle games. Για την ορθή λειτουργία ενός Tilemap γίνεται χρήση των εξής χαρακτηριστικών.

- **Πλέγμα Tilemap (Grid)**: Το Tilemap Grid είναι ένα σύστημα πλέγματος, στο οποίο τοποθετούνται τα tiles. Το grid εξασφαλίζει ότι τα tiles ευθυγραμμίζονται τέλεια, διευκολύνοντας τον σχεδιασμό μεγάλων περιοχών ή σκηνικών χωρίς να χρειάζεται να τοποθετηθούν με το χέρι όλα τα στοιχεία.

- **Tile Palette**: Το Unity παρέχει ένα εργαλείο Tile Palette που επιτρέπει στους χρήστες να δημιουργούν, να επεξεργάζονται και να αποθηκεύουν σύνολα πλακιδίων (tiles). Με αυτό το εργαλείο, αρχικά γίνεται η επιλογή του αντικειμένου που θέλουμε να τοποθετήσουμε στη σκηνή και έπειτα ξεκινάμε να το τοποθετούμε όπως θα ζωγραφίζαμε με ένα πινέλο.

- **Layers**: Το Tilemap υποστηρίζει διαφορετικά επίπεδα (layers), που επιτρέπουν την δημιουργία πολυεπίπεδων σκηνών. Για παράδειγμα, μπορείς να υπάρχει ένα επίπεδο για το έδαφος, ένα άλλο για αντικείμενα όπως δέντρα, και ένα τρίτο για διακοσμητικά στοιχεία, όπως σύννεφα ή κτίρια στο παρασκήνιο.

2.3.4 Sorting Layers και Order in Layer

Το Unity 2D υποστηρίζει το concept των Sorting Layers, το οποίο επιτρέπει τον καθορισμό της σειράς με την οποία τα αντικείμενα εμφανίζονται στην οθόνη. Αυτό είναι ιδιαίτερα σημαντικό σε 2D παιχνίδια, όπου όλα τα αντικείμενα βρίσκονται στο ίδιο επίπεδο και χρειάζεται να οριστεί ποιο αντικείμενο εμφανίζεται μπροστά από κάποιο άλλο. Υπάρχουν δύο βασικοί τρόποι για τον καθορισμό αυτής της σειράς.

- **Sorting Layers**: Είναι ομάδες στις οποίες μπορούν να ανήκουν τα αντικείμενα. Αντικείμενα που ανήκουν σε υψηλότερο sorting layer εμφανίζονται μπροστά από αντικείμενα που ανήκουν σε χαμηλότερο sorting layer.

- **Order in Layer**: Μέσα σε κάθε sorting layer, μπορεί να καθοριστεί μια πιο συγκεκριμένη σειρά εμφάνισης χρησιμοποιώντας την παράμετρο "Order in Layer". Για παράδειγμα, μπορεί να υπάρχουν δύο sprites στο ίδιο sorting layer, αλλά να καθοριστεί ότι το ένα θα εμφανίζεται μπροστά από το άλλο μέσω του Order in Layer.

Αυτός ο μηχανισμός επιτρέπει τον έλεγχο της "οπτικής ιεραρχίας" των αντικειμένων σε ένα 2D παιχνίδι και βοηθά στη δημιουργία της αίσθησης του βάθους, παρόλο που τα αντικείμενα είναι επίπεδα.

2.3.5 Κάμερα στο 2D Περιβάλλον

Στην Unity, η κάμερα είναι το "μάτι" του παίκτη που βλέπει τη σκηνή. Στο 2D περιβάλλον, η κάμερα λειτουργεί με λίγο διαφορετικό τρόπο σε σχέση με το 3D, καθώς βλέπει τα πάντα σε δύο διαστάσεις και εστιάζει στη σωστή προβολή των δισδιάστατων στοιχείων.

Συνήθως η κάμερα στο Unity 2D χρησιμοποιεί orthographic projection, η οποία δεν εφαρμόζει την προοπτική με την οποία τα αντικείμενα φαίνονται μικρότερα όσο απομακρύνονται. Αυτό σημαίνει ότι όλα τα αντικείμενα στο 2D επίπεδο εμφανίζονται με το ίδιο μέγεθος, ανεξάρτητα από την απόστασή τους από την κάμερα.

Η κίνηση της κάμερας, σε 2D παιχνίδια, συχνά ακολουθεί τον χαρακτήρα του παίκτη, προσφέροντας ρευστή προβολή του κόσμου.

2.3.6. Pixel Perfect Rendering

Σε 2D παιχνίδια με pixel art, η σωστή απεικόνιση των γραφικών χωρίς να παραμορφώνονται τα pixels είναι κρίσιμη. Το Unity προσφέρει το εργαλείο Pixel Perfect Camera, το οποίο εξασφαλίζει ότι τα γραφικά παραμένουν καθαρά και ευκρινή ανεξαρτήτως ανάλυσης ή μεγέθους οθόνης, χρησιμοποιώντας τις εξής τεχνικές.

- **Pixel Perfect Mode**: Όταν ενεργοποιείται, το Unity προσαρμόζει την κίνηση της κάμερας και την απεικόνιση των sprites, ώστε να παραμένουν "pixel perfect", διασφαλίζοντας ότι τα sprites δεν παραμορφώνονται και παραμένουν ευκρινή σε κάθε καρέ.

- **Pixel Snap**: Αυτό το χαρακτηριστικό βοηθάει στην ευθυγράμμιση των sprites με το πλέγμα των pixels της οθόνης, ώστε να μην υπάρχει παραμόρφωση ή "κουνήματα" στα γραφικά κατά την κίνηση του χαρακτήρα.

2.4 Σύστημα 2D Animation

Το 2D Animation System του Unity είναι ένα πανίσχυρο εργαλείο που επιτρέπει στους δημιουργούς να ζωντανέψουν τους χαρακτήρες και τα αντικείμενα σε ένα 2D περιβάλλον. Παρέχει πληθώρα εργαλείων και δυνατοτήτων που διευκολύνουν την ανάπτυξη ρεαλιστικών και σύνθετων κινήσεων μέσα στο παιχνίδι, είτε

πρόκειται για κινήσεις χαρακτήρων, αλλαγές κατάστασης αντικειμένων, είτε για δυναμικές αλληλεπιδράσεις με το περιβάλλον.

2.4.1 Animator Controller

Η καρδιά του 2D Animation System είναι το Animator Controller. Αυτός είναι ένας διαχειριστής που επιτρέπει τη δημιουργία και τη διαχείριση διαφόρων animation clips, καθώς και τη σύνδεση τους με τους χαρακτήρες ή τα αντικείμενα που θα κινούνται. Ο Controller αυτός αποτελείται από τα εξής χαρακτηριστικά.

- **Animation States (Καταστάσεις)**: Το Animator Controller αποτελείται από animation states, που είναι βασικά διαφορετικές καταστάσεις κίνησης. Για παράδειγμα, ένας χαρακτήρα, μπορεί να έχει ένα state για το περπάτημα, ένα για το άλμα, και ένα για όταν ο χαρακτήρας μένει ακίνητος (idle).
- **Transitions (Μεταβάσεις)**: Οι μεταβάσεις μεταξύ αυτών των states καθορίζονται από προκαθορισμένες συνθήκες ή παραμέτρους. Για παράδειγμα, η μετάβαση από την κίνηση περπατήματος σε άλμα μπορεί να ενεργοποιείται όταν ο παίκτης πατάει το κουμπί άλματος. Η μετάβαση από το άλμα πίσω σε περπάτημα μπορεί να ενεργοποιηθεί όταν ο χαρακτήρας αγγίζει το έδαφος.
- **Blend Trees**: Τα blend trees είναι χρήσιμα εργαλεία για τη δημιουργία ομαλών μεταβάσεων μεταξύ διαφορετικών animations. Για παράδειγμα, σε ένα blend tree, μπορεί να συνδυαστούν τα animations για το περπάτημα και το τρέξιμο, όπου η ταχύτητα του χαρακτήρα θα ελέγχει τη μετάβαση από το ένα animation στο άλλο, εξασφαλίζοντας μια ρεαλιστική κίνηση.

2.4.2 Sprite-Based Animation

Στο 2D animation του Unity, το sprite-based animation είναι ο βασικός τύπος κινήσεων. Πρόκειται για τη διαδικασία εναλλαγής διαφορετικών εικόνων (sprites) με την πάροδο του χρόνου για να δημιουργηθεί η ψευδαίσθηση της κίνησης.

Για τη δημιουργία sprite animations, συχνά χρησιμοποιούνται τα sprite sheets, που αναλύσαμε προηγουμένως. Αυτά είναι μεγάλα αρχεία εικόνας που περιέχουν πολλαπλά καρέ κίνησης. Το σύστημα animation της Unity μπορεί να "κόψει" αυτά τα καρέ και να τα παίξει σε μια προκαθορισμένη σειρά για να δημιουργήσει κίνηση, όπως το τρέξιμο ή το άλμα ενός χαρακτήρα.

Ακόμα υπάρχουν τα animation clips, τα οποία είναι τα επιμέρους αρχεία που αποθηκεύουν μια συγκεκριμένη ακολουθία κινήσεων. Κάθε clip μπορεί να περιέχει μια σειρά από frames (καρέ), με κάθε καρέ να αντιπροσωπεύει ένα συγκεκριμένο sprite. Για παράδειγμα, ένα animation clip μπορεί να αναπαριστά το animation του περπατήματος ενός χαρακτήρα, με κάθε καρέ να δείχνει το χαρακτήρα σε διαφορετική θέση.

2.4.3 Frame-by-Frame Animation

Το frame-by-frame animation είναι μια πιο κλασική προσέγγιση όπου κάθε καρέ του animation είναι ένα μεμονωμένο sprite που αλλάζει κάθε φορά που ανανεώνεται το frame rate. Το Unity υποστηρίζει αυτή

τη μορφή, και είναι ιδανική για animations που δεν απαιτούν πολύπλοκες μεταβάσεις ή επαναλαμβανόμενα μοτίβα κίνησης.

2.4.4 Animation Events

Τα Animation Events είναι μια ισχυρή δυνατότητα του Unity που επιτρέπει να προσθέτουν ειδικά γεγονότα σε συγκεκριμένα καρέ ενός animation. Για παράδειγμα, όταν ένας χαρακτήρας κάνει μια κίνηση επίθεσης, μπορείς να προστεθεί ένα animation event που θα ενεργοποιήσει τον ήχο του χτυπήματος ή θα εφαρμόσει ζημιά στον αντίπαλο.

Τα animation events μπορούν να καλέσουν μεθόδους μέσα από scripts. Αυτό δίνει τη δυνατότητα να συνδεθεί το animation με τη λογική του παιχνιδιού. Για παράδειγμα, ένα animation event μπορεί να καλέσει ένα script για να ενεργοποιήσει μια δράση, όπως η συλλογή ενός αντικειμένου ή η αλλαγή της κατάστασης του χαρακτήρα.

2.4.5 Scripting και Animation Control

Οι κινήσεις και τα animations μπορούν να ελεγχθούν μέσω scripting, όπως το πότε θα ξεκινήσει ή θα σταματήσει ένα animation. Για παράδειγμα, μπορεί να παγώσει ένα animation όταν ο χαρακτήρας μπαίνει σε μια ειδική κατάσταση (state) ή να επαναληφθεί όταν ολοκληρώνεται μια δράση.

Μέσω scripting μπορούν ακόμα να γίνουν και λειτουργίες όπως blending και crossfade, που επιτρέπουν ομαλές μεταβάσεις μεταξύ διαφορετικών animations, ειδικά όταν θέλουμε να περάσει από μία κατάσταση σε άλλη χωρίς απότομες αλλαγές.

Οι μέθοδοι SetTrigger(), SetBool(), και SetFloat(), χρησιμοποιούνται για να αλληλεπιδρούν τα scripts με τον Animator και τους παραμέτρους του animation. Ο Animator είναι υπεύθυνος για τη διαχείριση και την εκτέλεση animation states (καταστάσεων) που εφαρμόζονται σε ένα GameObject. Αυτές οι μέθοδοι χρησιμοποιούνται για τον έλεγχο των παραμέτρων που επηρεάζουν τη ροή του animation στο Unity.

SetTrigger()

Η μέθοδος SetTrigger() χρησιμοποιείται για να ενεργοποιήσει μια παράμετρο τύπου "Trigger" στον Animator. Ένα Trigger είναι μια ειδική παράμετρος που ενεργοποιεί τη μετάβαση (transition) σε ένα νέο animation state όταν καλείται, και αμέσως μετά την ενεργοποίηση, επιστρέφει στην αρχική του κατάσταση. Χρησιμοποιείται συνήθως για να ξεκινήσει ένα συγκεκριμένο animation μία φορά, όπως ένα άλμα ή μια επίθεση, όταν συμβεί ένα συγκεκριμένο συμβάν (π.χ. όταν ο χρήστης πατήσει ένα κουμπί). Μόλις ενεργοποιηθεί το Trigger, το animation θα μεταβεί στην κατάσταση που είναι συνδεδεμένη με αυτό, και η παράμετρος θα "σβήσει" μετά την εκτέλεση της μετάβασης.

SetBool()

Η μέθοδος SetBool() χρησιμοποιείται για τον έλεγχο μιας παραμέτρου τύπου "Boolean" στον Animator. Ένα Boolean μπορεί να έχει μόνο δύο τιμές: true (αληθές) ή false (ψευδές). Αυτή η μέθοδος επιτρέπει τον έλεγχο αν ένα συγκεκριμένο animation state πρέπει να ενεργοποιηθεί ή όχι, ανάλογα με την κατάσταση του Boolean. Χρησιμοποιούνται συχνά για να ενεργοποιούν ή να απενεργοποιούν συγκεκριμένα animations που πρέπει να επαναλαμβάνονται όσο μια συνθήκη είναι αληθής.

SetFloat()

Η μέθοδος SetFloat() χρησιμοποιείται για τον έλεγχο μιας παραμέτρου τύπου "Float" στον Animator. Τα Float είναι δεκαδικοί αριθμοί που μπορούν να έχουν οποιαδήποτε τιμή, και είναι χρήσιμα όταν χρειάζεται πιο σύνθετος έλεγχος ενός animation, όπως η ταχύτητα ή η κατεύθυνση κίνησης. Χρησιμοποιείται συχνά για να ελέγχεται το βάρος (weight) μιας μετάβασης, την ταχύτητα του animation, ή για να καθορίσουν μεταβαλλόμενες συνθήκες, όπως την ταχύτητα κίνησης του παίκτη. Συνήθως συνδυάζεται με Blend Tree για να επιτρέψει ομαλές μεταβάσεις ανάμεσα σε διάφορα animations με βάση την τιμή του Float.

2.5 Φυσική (Physics) στο 2D

Η Φυσική (Physics) στο 2D στο Unity είναι ένα από τα πιο σημαντικά συστήματα που διαχειρίζονται τις αλληλεπιδράσεις των αντικειμένων στον 2D κόσμο. Το Unity προσφέρει μια ολόκληρη σουίτα εργαλείων και components που βοηθούν στην προσομοίωση της φυσικής κίνησης, της σύγκρουσης, της βαρύτητας και άλλων δυνάμεων σε δισδιάστατα παιχνίδια. Αυτό κάνει το παιχνίδι πιο ρεαλιστικό και παρέχει δυναμική αλληλεπίδραση μεταξύ των αντικειμένων.

2.5.1 Rigidbody 2D

Το **Rigidbody 2D** είναι η βασική μονάδα φυσικής για όλα τα αντικείμενα στο 2D περιβάλλον. Προσθέτοντας ένα Rigidbody 2D component σε ένα αντικείμενο, αυτό αποκτά φυσική ιδιότητα, δηλαδή μπορεί να αλληλεπιδράσει με τις δυνάμεις, τη βαρύτητα, τις συγκρούσεις και άλλους παράγοντες φυσικής. Το Rigidbody έχει τις εξής βασικές ιδιότητες.

- **Δυναμική Φυσική**: Τα αντικείμενα με ενεργοποιημένο Rigidbody 2D μπορούν να μετακινηθούν από δυνάμεις όπως η ώθηση (force) ή οι συγκρούσεις. Το αντικείμενο αποκτά βάρος και αδράνεια, και η κίνησή του εξαρτάται από τις δυνάμεις που του ασκούνται.

- **Βαρύτητα (Gravity)**: Το Rigidbody 2D έχει μια παράμετρο βαρύτητας (gravity scale) που καθορίζει πόσο επηρεάζεται το αντικείμενο από τη βαρύτητα. Αν το βάρος είναι θετικό, το αντικείμενο θα πέσει προς τα κάτω, ενώ αν είναι αρνητικό, θα επιπλεύσει προς τα πάνω. Στην συγκεκριμένη περίπτωση, επειδή το παιχνίδι χρησιμοποιεί μόνο τους άξονες x και y και επειδή η προβολή είναι από πάνω προς τα κάτω (top down), δεν γίνεται χρήση της βαρύτητας. Οπότε όλα τα αντικείμενα μέσα στο παιχνίδι έχουν την τιμή 0 στο gravity scale.

- **Κινήσεις και Δυνάμεις (Forces)**: Μπορεί να εφαρμοστούν δυνάμεις σε ένα Rigidbody 2D αντικείμενο χρησιμοποιώντας τη μέθοδο AddForce(). Αυτό είναι χρήσιμο για την προσομοίωση κίνησης, όπως η ώθηση ενός αντικειμένου ή η εκτόξευση ενός χαρακτήρα στον αέρα.

- **Κινηματικά Rigidbody 2D (Kinematic Rigidbody 2D)**: Αυτά τα αντικείμενα δεν επηρεάζονται από τη βαρύτητα ή άλλες δυνάμεις, αλλά μπορούν να μετακινηθούν χειροκίνητα από το script. Είναι χρήσιμα όταν θέλεις πλήρη έλεγχο της κίνησης ενός αντικειμένου, όπως η κίνηση μιας πλατφόρμας ή ενός NPC που ακολουθεί μια προκαθορισμένη διαδρομή.

2.5.2 Colliders 2D

Τα Colliders 2D είναι τα εξαρτήματα που επιτρέπουν στα αντικείμενα να ανιχνεύουν και να αποκρίνονται σε συγκρούσεις με άλλα αντικείμενα. Τα colliders καθορίζουν το σχήμα του αντικειμένου στο οποίο μπορεί να συμβεί η σύγκρουση, χωρίς να επηρεάζουν την οπτική του εμφάνιση.

- **Box Collider 2D**: Είναι ο απλούστερος τύπος collider που εφαρμόζει ένα ορθογώνιο πλαίσιο γύρω από το αντικείμενο. Είναι ιδανικό για αντικείμενα με κανονικά γεωμετρικά σχήματα, όπως κιβώτια ή πλατφόρμες.
- **Circle Collider 2D**: Όπως υποδηλώνει το όνομά του, το Circle Collider 2D εφαρμόζει έναν κυκλικό collider στο αντικείμενο. Χρησιμοποιείται συνήθως για στρογγυλά αντικείμενα όπως μπάλες ή κυκλικοί εχθροί.
- **Polygon Collider 2D**: Αυτός ο collider χρησιμοποιείται για αντικείμενα με ακανόνιστες μορφές. Προσαρμόζεται στο ακριβές περίγραμμα του sprite, καθιστώντας τον ιδανικό για πολύπλοκα σχήματα που δεν μπορούν να καλυφθούν από απλά γεωμετρικά colliders.
- **Edge Collider 2D**: Χρησιμοποιείται για να δημιουργηθούν γραμμές σύγκρουσης. Είναι κατάλληλος για το σχεδιασμό σύνθετων γραμμικών επιφανειών, όπως διαδρομές σε παιχνίδια με φυσική.
- **Composite Collider 2D**: Επιτρέπει τον συνδυασμό πολλών colliders σε ένα, προσφέροντας μεγαλύτερη απόδοση και καλύτερη διαχείριση της φυσικής για πολύπλοκα σχήματα.

Κάθε collider έχει την επιλογή να γίνει trigger, δηλαδή είναι colliders που δεν προκαλούν φυσική σύγκρουση, αλλά ενεργοποιούν γεγονότα όταν τα αντικείμενα εισέρχονται ή εξέρχονται από την περιοχή τους. Τα triggers είναι πολύ χρήσιμα για τη δημιουργία αλληλεπιδράσεων στο παιχνίδι, όπως η ενεργοποίηση μιας πόρτας, η συλλογή ενός αντικειμένου, ή η είσοδος σε μια νέα περιοχή.

2.5.3 Physics Materials 2D

Τα Physics Materials 2D καθορίζουν τις ιδιότητες τριβής και ελαστικότητας ενός αντικειμένου κατά τις συγκρούσεις.

Η Τριβή (Friction) καθορίζει πόση αντίσταση ασκείται όταν δύο αντικείμενα τρίβονται μεταξύ τους. Ένα υλικό με υψηλή τριβή θα σταματήσει γρήγορα την κίνηση των αντικειμένων, ενώ ένα υλικό με χαμηλή τριβή θα επιτρέψει στα αντικείμενα να γλιστρούν πιο εύκολα.

Η Ελαστικότητα (Bounciness) καθορίζει πόσο αναπηδά ένα αντικείμενο όταν συγκρούεται με άλλο αντικείμενο. Ένα αντικείμενο με υψηλή ελαστικότητα θα αναπηδήσει πιο πολύ μετά από μια σύγκρουση.

2.5.4 Physics Simulation (Προσομοίωση Φυσικής)

Η προσομοίωση φυσικής στο Unity 2D τρέχει σε πραγματικό χρόνο και υπολογίζει τις δυνάμεις, τις συγκρούσεις και την αλληλεπίδραση μεταξύ των αντικειμένων. Αυτές οι προσομοιώσεις τρέχουν στο background και εξασφαλίζουν ότι το παιχνίδι παραμένει ρεαλιστικό όσον αφορά τη φυσική συμπεριφορά των αντικειμένων.

Το Unity τρέχει τη φυσική σε συγκεκριμένα χρονικά διαστήματα που ορίζονται από το Fixed Timestep. Αυτό εξασφαλίζει σταθερότητα στη φυσική, ανεξαρτήτως του πόσο γρήγορα ή αργά τρέχει το παιχνίδι σε διαφορετικά συστήματα.

Η ανίχνευση σύγκρουσης (Collision Detection) στο 2D περιβάλλον επιτρέπει την αναγνώριση της στιγμής που δύο αντικείμενα έρχονται σε επαφή. Υπάρχουν δύο τύποι ανίχνευσης: Discrete (διακριτή) και

Continuous (συνεχής). Η συνεχής ανίχνευση χρησιμοποιείται για ταχέως κινούμενα αντικείμενα, αποφεύγοντας να «διαπεράσουν» άλλα αντικείμενα λόγω χαμηλής ανάλυσης του χρονικού βήματος.

2.5.5 Αλληλεπίδραση Με Σενάρια (Scripting Physics)

Η φυσική στο Unity μπορεί να ελεγχθεί και μέσω scripting, παρέχοντας ευελιξία στην προσαρμογή της συμπεριφοράς των αντικειμένων. Για τον έλεγχο της φυσικής των αντικειμένων, στο συγκεκριμένο παιχνίδι, γίνεται χρήση των παρακάτω μεθόδων.

Πιο συγκεκριμένα, για την κίνηση του παίκτη, χρησιμοποιείται η μέθοδος MovePosition(). Αντί να αλλάξει άμεσα τη θέση του αντικειμένου, αυτή η μέθοδος χρησιμοποιεί τη φυσική του Unity για να μεταφέρει το αντικείμενο. Παίρνει μια τιμή Vector2, η οποία είναι η επιθυμητή θέση που θέλουμε να μετακινήσουμε το αντικείμενο. Αυτή την τιμή μπορούμε να την πολλαπλασιάζουμε με μία ακόμα τιμή, επηρεάζοντας έτσι την ταχύτητα με την οποία κινείται.

Η μέθοδος MoveTowards() χρησιμοποιείται για να μεταφέρει ένα αντικείμενο από τη μία θέση στην άλλη με σταθερή ταχύτητα. Αυτή η μέθοδος υπολογίζει την επόμενη θέση του αντικειμένου με βάση την τρέχουσα θέση, την επιθυμητή θέση και την ταχύτητα.

Και οι 2 μέθοδοι έχουν το ίδιο αποτέλεσμα, την κίνηση ενός αντικειμένου προς κάποιο σημείο, παρέχουν όμως διαφορετικές προσεγγίσεις για την κίνηση αυτή. Η επιλογή της κατάλληλης μεθόδου εξαρτάται από τις ανάγκες του παιχνιδιού και την επιθυμητή συμπεριφορά της κίνησης. Για AI και απλές μεταβάσεις, η MoveTowards() είναι κατάλληλη, ενώ για φυσικές και κινητικές αλληλεπιδράσεις, η MovePosition() είναι προτιμότερη.

Υπάρχουν οι μέθοδοι OnCollisionEnter2D(), OnCollisionExit2D(), OnCollisionStay2D(), καθώς και οι μέθοδοι OnTriggerEnter2D(), OnTriggerExit2D() και OnTriggerStay2D() χρησιμοποιούνται για την ανίχνευση και διαχείριση αλληλεπιδράσεων μεταξύ αντικειμένων στον 2D κόσμο. Αυτές οι μέθοδοι είναι ουσιαστικά callbacks που ενεργοποιούνται από φυσικές αλληλεπιδράσεις και βοηθούν στη δημιουργία ρεαλιστικών συμπεριφορών στα παιχνίδια.

- **OnCollisionEnter2D():** καλείται όταν ένα Rigidbody2D αντικείμενο (ή αντικείμενο με Collider2D) έρχεται σε επαφή με ένα άλλο αντικείμενο που έχει επίσης Collider2D. Αυτή η μέθοδος ενεργοποιείται μία φορά, τη στιγμή που συμβαίνει η σύγκρουση.
- **OnCollisionExit2D():** καλείται όταν ένα Rigidbody2D αντικείμενο απομακρύνεται από άλλο αντικείμενο με Collider2D. Αυτό συμβαίνει όταν η επαφή ανάμεσα στα δύο αντικείμενα σταματά.
- **OnCollisionStay2D():** καλείται σε κάθε frame που το αντικείμενο είναι σε επαφή με ένα άλλο Collider2D. Αυτή η μέθοδος ενεργοποιείται συνεχώς όσο η σύγκρουση παραμένει.
- **OnTriggerEnter2D():** καλείται όταν ένα αντικείμενο με Collider2D που έχει ενεργοποιημένο το Is Trigger εισέρχεται σε επαφή με άλλο αντικείμενο. Σε αντίθεση με τις προηγούμενες μεθόδους, δεν δημιουργεί φυσική σύγκρουση, αλλά ανιχνεύει την παρουσία ενός άλλου αντικειμένου.
- **OnTriggerExit2D():** καλείται όταν ένα αντικείμενο που εισήλθε σε μια περιοχή ενεργοποιημένη ως trigger απομακρύνεται από αυτήν.
- **OnTriggerStay2D():** καλείται σε κάθε frame όσο το αντικείμενο παραμένει σε επαφή με την περιοχή του trigger.

Η φυσική στο 2D σύστημα της Unity είναι εξαιρετικά ευέλικτη και παρέχει όλα τα απαραίτητα εργαλεία για τη δημιουργία ρεαλιστικών παιχνιδιών με φυσικές αλληλεπιδράσεις. Από τα βασικά συστήματα των rigidbody και colliders έως τις πιο σύνθετες προσομοιώσεις με joints και forces, προσφέρει ένα πλήρες φάσμα λειτουργιών που δίνει βάθος και ρεαλισμό στα 2D παιχνίδια.

2.6 User Interface (UI)

Η καλή σχεδίαση του UI είναι κρίσιμης σημασίας για την εμπειρία του παίκτη σε κάποιο παιχνίδι. Ένα καλά σχεδιασμένο UI μπορεί να βελτιώσει τη χρηστικότητα του παιχνιδιού και να καθοδηγήσει τον παίκτη. Τα στοιχεία του UI είναι τα ορατά κομμάτια που αλληλεπιδρούν με τον χρήστη και παρέχουν πληροφορίες ή ελέγχουν τη λειτουργία του παιχνιδιού

2.6.1 Στοιχεία UI

Το UI περιλαμβάνει διάφορα στοιχεία, όπως:

- **Κουμπιά (Buttons)**: Χρησιμοποιούνται για αλληλεπίδραση με το παιχνίδι, όπως η εκκίνηση του παιχνιδιού, η επιλογή επιλογών ή να ανοίξει κάποιο μενού. Μπορούν να εφαρμοστούν διάφορα εφέ όπως, hover και click για να παρέχετε ανατροφοδότηση στον χρήστη, δείχνοντας πότε ένα κουμπί είναι ενεργό ή πότε πατιέται.

- **Ετικέτες (Text)**: Χρησιμοποιούνται για να παρέχουν πληροφορίες στους παίκτες, όπως σκορ, στατιστικά του παίκτη ή οδηγίες. Υπάρχουν επιλογές για την ρύθμιση των γραμμάτων, όπως την γραμματοσειρά, το μέγεθος και το χρώμα.

- **Εικόνες (Images)**: Μπορούν να χρησιμοποιηθούν για διακοσμητικούς σκοπούς ή για να απεικονίσουν αντικείμενα και χαρακτήρες. Υπάρχουν δύο είδη εικόνων, η κανονική που δέχεται ένα sprite και η raw που δέχεται texture.

- **Sliders**: Χρησιμοποιούνται για τη ρύθμιση παραμέτρων όπως η ένταση ήχου ή η φωτεινότητα.

- **Πίνακες (Panels)**: Χρησιμοποιούνται για να οργανώνουν και ομαδοποιούν άλλα στοιχεία UI, διευκολύνοντας τη διαχείριση του χώρου στην οθόνη.

Το βασικότερο στοιχείο του UI είναι ο Καμβάς (Canvas). Είναι το στοιχείο πάνω στο οποίο τοποθετούνται και εμφανίζονται, όλα τα άλλα UI στοιχεία που αναφέρθηκαν παραπάνω. Υποστηρίζει διάφορες ρυθμίσεις για την κλίμακα και την αναλογία διάστασης, όπως:

- **Screen Space - Overlay:** Τα UI στοιχεία τοποθετούνται απευθείας πάνω από την κάμερα.

- **Screen Space - Camera:** Τα UI στοιχεία συνδέονται με μια συγκεκριμένη κάμερα, επιτρέποντας 3D εφέ.

- **World Space:** Το UI μπορεί να τοποθετηθεί σε 3D χώρο και να αλληλεπιδράσει με άλλα 3D αντικείμενα.

Όλα αυτά τα στοιχεία έχουν ένα Rect Transform, που είναι ένας ειδικός τύπος Transform που επιτρέπει τον έλεγχο του μεγέθους και της θέσης των UI στοιχείων.

2.6.2 Σχεδίαση UI

Η σχεδίαση του UI πρέπει να είναι φιλική προς το χρήστη και αισθητικά ευχάριστη. Ακολουθούν μερικές βασικές αρχές σχεδίασης:

- **Απλότητα**: Ένα απλό και καθαρό UI είναι πιο εύκολο να κατανοηθεί και να χρησιμοποιηθεί από τους παίκτες.
- **Σαφήνεια**: Οι πληροφορίες πρέπει να είναι σαφείς και ευανάγνωστες. Ένας τρόπος είναι η χρήση χρωμάτων με αντίθεση για να διευκολυνθεί η ανάγνωση.
- **Συνοχή**: Πρέπει να διατηρείται μία συνεπή αισθητική σε όλα τα στοιχεία UI. Δηλαδή να χρησιμοποιούνται τα ίδια χρώματα, γραμματοσειρές και στυλ.
- **Αλληλεπίδραση**: Η χρήση σαφή σημάτων για τις αλληλεπιδράσεις του χρήστη, όπως οπτικά εφέ όταν ο χρήστης πηγαίνει το ποντίκι πάνω από ένα κουμπί.

2.6.3 Διαχείριση UI Events

Με πολλά από τα UI στοιχεία, όπως τα κουμπιά ή τα sliders, ο χρήστης μπορεί να αλληλεπιδράσει μαζί τους. Υπάρχουν οι εξής μέθοδοι που μπορούν να χρησιμοποιηθούν ανάλογα με το στοιχείο και την χρήση που θέλουμε να κάνουμε.

- **OnClick Events**: Χρησιμοποιούνται όταν οι χρήστες αλληλεπιδρούν με το UI μέσω κλικ. Αυτές οι αλληλεπιδράσεις απαιτούν τον ορισμό δράσεων που θα εκτελούνται όταν ο χρήστης κάνει κλικ σε ένα κουμπί.
- **OnValueChanged Events**: Χρησιμοποιούνται για να ανιχνεύσουν αλλαγές σε sliders ή checkbox.
- **Drag & Drop Events**: Οι μέθοδοι αυτοί δίνουν την δυνατότητα μεταφοράς και απόθεσης αντικειμένων πάνω σε άλλα UI στοιχεία και καλούνται όταν γίνει το αντίστοιχο γεγονός.

2.6.4 Layout Groups

Τα Layout Groups είναι εργαλεία που διευκολύνουν τη ρύθμιση και την οργάνωση των UI στοιχείων. Επιτρέπουν την αυτόματη τοποθέτηση και διάταξη των στοιχείων σύμφωνα με προκαθορισμένους κανόνες. Υπάρχουν οι εξής τύποι.

- **Horizontal Layout Group:** Τοποθετεί UI στοιχεία οριζόντια.
- **Vertical Layout Group:** Τοποθετεί UI στοιχεία κάθετα.
- **Grid Layout Group:** Οργανώνει UI στοιχεία σε μορφή πλέγματος.

Η δημιουργία ενός καλού UI απαιτεί προσοχή στη λεπτομέρεια, καλή σχεδίαση και τεχνική γνώση. Ένα καλοσχεδιασμένο UI μπορεί να κάνει τη διαφορά στην εμπειρία του χρήστη, βελτιώνοντας την χρηστικότητα και καθιστώντας το παιχνίδι πιο ευχάριστο και διασκεδαστικό.

2.7 Render Texture

Το Render Texture στο Unity είναι ένα ισχυρό εργαλείο που επιτρέπει την δημιουργία και την απόδοση δυναμικού περιεχομένου σε μια υφή (texture), η οποία μπορεί να χρησιμοποιηθεί σε πραγματικό χρόνο σε διάφορα μέρη του παιχνιδιού, όπως στην απεικόνιση καμερών, στην επικάλυψη υφών, και σε διάφορα εφέ μετα-επεξεργασίας.

Είναι μια ειδική κατηγορία αντικειμένου Texture που αποθηκεύει την έξοδο μιας κάμερας ή μιας απόδοσης (rendering) σε ένα τετράγωνο ή ορθογώνιο επίπεδο αντί για την οθόνη. Η υφή που παράγεται μπορεί στη συνέχεια να χρησιμοποιηθεί ως πηγή σε υλικά ή shaders, προβάλλοντας τη δημιουργία ή τις σκηνές σε πραγματικό χρόνο σε επιφάνειες στο παιχνίδι.

2.7.1 Χρήσεις του Render Texture

Τα Render Textures μπορούν να χρησιμοποιηθούν σε πολλές περιπτώσεις και τεχνικές όπως:

Προβολή Πολλαπλών Καμερών:
Χρησιμοποιείται για να αποδίδονται οι εικόνες πολλαπλών καμερών σε διαφορετικές επιφάνειες ή UI στοιχεία, όπως για παράδειγμα οι οθόνες ασφάλειας σε ένα παιχνίδι, όπου κάθε οθόνη δείχνει διαφορετική προοπτική του περιβάλλοντος.

Εφέ Μετά-Επεξεργασίας (Post-Processing):
Η χρήση render textures είναι απαραίτητη για πολλά εφέ μετά-επεξεργασίας, όπως η αντανάκλαση, το bloom, και άλλα. Στην ουσία, η εικόνα της κάμερας αποθηκεύεται σε μια υφή, εφαρμόζονται τα εφέ και στη συνέχεια η τελική εικόνα προβάλλεται στην οθόνη.

Δυναμική Δημιουργία Υλικών και Εφέ:
Τα render textures μπορούν να χρησιμοποιηθούν για τη δημιουργία δυναμικών και ρευστών εφέ, όπως η προβολή ενός χαρακτήρα μέσα από έναν καθρέφτη, ή η προβολή ενός video feed σε έναν εικονικό κόσμο.

UI Στοιχεία:
Τα Render Textures μπορούν να ενσωματωθούν σε UI στοιχεία, δίνοντας τη δυνατότητα να δημιουργηθούν δυναμικά οπτικά εφέ στα μενού και σε άλλα στοιχεία του παιχνιδιού, όπως σε live video προβολές μέσα στο UI.

2.7.2 Δημιουργία και Χρήση Render Textures

1. Δημιουργία Render Texture
Η δημιουργία ενός Render Texture γίνεται μέσω του Unity Editor. Κάνοντας δεξί κλικ μέσα σε κάποιον φάκελο στο Project Panel και επιλέγοντας **Create > Render Texture**. Αυτό θα δημιουργήσει ένα αρχείο Render Texture σε αυτή την τοποθεσία.

2. Εκχώρηση σε Κάμερα
Για να χρησιμοποιηθει ένα Render Texture με μια κάμερα, αρχικά γίνεται επιλογή μιας κάμερας στην σκηνή. Έπειτα στον Inspector, στην επιλογή Target Texture, σύρετε το Render Texture που έχει δημιουργηθεί από πριν. Τώρα, η κάμερα δεν θα αποδίδει απευθείας στην οθόνη, αλλά θα αποθηκεύει την απόδοσή της στην υφή που έχει εκχωρηθεί.

3. Χρήση του Render Texture

Το Render Texture που αποθηκεύει την εικόνα της κάμερας μπορεί να χρησιμοποιηθεί σαν υλικό σε διάφορα αντικείμενα του παιχνιδιού. Μπορεί, για παράδειγμα, να εφαρμοστεί σε έναν κύβο ή ένα τετράγωνο panel για να προβληθεί το περιεχόμενο της κάμερας.

4. Χαρακτηριστικά Render Texture

- **Resolution (Ανάλυση)**: Μπορεί να καθοριστεί η ανάλυση του Render Texture. Μια υψηλή ανάλυση απαιτεί περισσότερη μνήμη, αλλά προσφέρει καλύτερη ποιότητα.
- **Depth Buffer**: Αν η σκηνή περιέχει εφέ βάθους (depth), μπορεί να ρυθμιστεί το depth buffer του Render Texture.
- **Anti-aliasing**: Παρέχει ομαλότητα στα περιγράμματα των αντικειμένων για να αποφευχθεί το aliasing.
- **Mip Maps**: Μπορούν να ενεργοποιηθούν ή να απενεργοποιηθούν τα mip maps για το Render Texture, κάτι που είναι χρήσιμο όταν το texture αποδίδεται σε αντικείμενα που αλλάζουν κλίμακα ή απόσταση από την κάμερα.

2.7.3 Χαρακτηριστικά και Ιδιότητες των Render Textures

Format (Μορφότυπο):

Μπορεί να επιλεχθεί το χρωματικό και το βάθος format της υφής. Αυτό περιλαμβάνει επιλογές όπως **ARGB32**, **RGB565**, και άλλα format που καθορίζουν πώς θα αποθηκευτούν τα δεδομένα της εικόνας και της πληροφορίας βάθους.

Dynamic Resolution:

Η υποστήριξη δυναμικής ανάλυσης επιτρέπει στο render texture να προσαρμόζει την ανάλυσή του ανάλογα με τις απαιτήσεις απόδοσης, κάτι που βελτιώνει τη συνολική αποδοτικότητα του παιχνιδιού.

Anti-Aliasing:

Τα render textures υποστηρίζουν anti-aliasing, το οποίο βελτιώνει την ποιότητα των γραφικών μειώνοντας την "οδοντωτή" εμφάνιση στις άκρες των αντικειμένων.

Cubemap:

Τα render textures μπορούν να αποθηκεύσουν εικόνες σε μορφή cubemap, η οποία είναι χρήσιμη για εφέ περιβάλλοντος και αντανάκλασης. Τα cubemaps αποδίδονται σε έξι πλευρές και χρησιμοποιούνται κυρίως για προσομοιώσεις περιβαλλοντικών αντανακλάσεων.

2.8 Unity Asset Store

Το **Unity Asset Store** είναι μια διαδικτυακή πλατφόρμα που παρέχει τη δυνατότητα να αγοραστούν, να πωληθούν και να μοιραστούν πόροι για την ανάπτυξη παιχνιδιών και εφαρμογών στο Unity. Αυτοί οι πόροι μπορεί να περιλαμβάνουν 3D μοντέλα, 2D sprites, ήχους, animations, shaders, scripts, εργαλεία ανάπτυξης και πολλά άλλα, τα οποία λέγονται πακέτα και μπορούν να βρεθούν στον Package Manager. Είναι ένας ζωτικής σημασίας χώρος για όσους αναπτύσσουν παιχνίδια, καθώς προσφέρει τη δυνατότητα να ενσωματώσουν έτοιμα στοιχεία και να επιταχύνουν τη διαδικασία ανάπτυξης.

2.8.1 Χαρακτηριστικά του Unity Asset Store:

Ποικιλία Πόρων:

Το Asset Store περιλαμβάνει χιλιάδες πόρους, από βασικά εργαλεία μέχρι προχωρημένα assets για κάθε πτυχή της ανάπτυξης παιχνιδιών. Αυτά καλύπτουν κατηγορίες όπως γραφικά, εργαλεία για scripting, εφέ φωτισμού, συστήματα UI και ήχου και πολλά άλλα.

Έτοιμοι Πόροι (Ready-to-Use Assets):

Πολλοί από τους πόρους είναι έτοιμοι για άμεση χρήση, με αποτέλεσμα να μειώνεται ο χρόνος ανάπτυξης και να αυξάνεται η παραγωγικότητα. Για παράδειγμα, αντί να δημιουργήσει κάποιος από την αρχή ένα 3D μοντέλο χαρακτήρα, μπορεί να αγοράσει ένα από το Asset Store και να το ενσωματώσει στο έργο του.

Εργαλεία Ανάπτυξης (Development Tools):

Το Asset Store προσφέρει επίσης εξειδικευμένα εργαλεία και plugins που βελτιώνουν την ανάπτυξη του παιχνιδιού, όπως συστήματα φυσικής, animation controllers, συστήματα διαχείρισης δεδομένων και πολλά άλλα.

Πόροι από την Κοινότητα:

Εκτός από τις επαγγελματικές εταιρείες, μεμονωμένοι χρήστες μπορούν να ανεβάσουν και να πωλήσουν τα δικά τους assets, συμβάλλοντας στην ευρύτερη κοινότητα προγραμματιστών του Unity.

Δωρεάν και Επί Πληρωμή Περιεχόμενο:

Στο Asset Store διατίθενται τόσο δωρεάν όσο και επί πληρωμή πόροι. Πολλοί δωρεάν πόροι είναι εξαιρετικής ποιότητας, ενώ τα επί πληρωμή assets συχνά συνοδεύονται από περισσότερες δυνατότητες ή υποστήριξη.

Πρόσβαση και Ενημερώσεις:

Τα assets που αγοράζονται παραμένουν διαθέσιμα στον λογαριασμό του χρήστη και ενημερώνονται συχνά από τους δημιουργούς τους, με βελτιώσεις ή υποστήριξη για τις νέες εκδόσεις του Unity.

2.8.2 Package Manager

Ο Unity Package Manager είναι ένα εργαλείο που επιτρέπει στους προγραμματιστές να διαχειρίζονται εξαρτήσεις και να ενσωματώνουν πακέτα που επεκτείνουν τη λειτουργικότητα της Unity. Μέσω του Package Manager, οι χρήστες μπορούν να εγκαταστήσουν, να ενημερώσουν, να αφαιρέσουν ή να διαμορφώσουν πακέτα που προσθέτουν συγκεκριμένες δυνατότητες στο project τους.

Τα πακέτα είναι κομμάτια λογισμικού που περιέχουν κώδικα, αρχεία, assets ή εργαλεία που μπορούν να προστεθούν σε ένα Unity project για να επεκτείνουν τη λειτουργικότητά του. Τα πακέτα είναι σχεδιασμένα για να είναι εύχρηστα και επαναχρησιμοποιήσιμα σε διαφορετικά έργα, επιτρέποντας έτσι την εύκολη κοινή χρήση και τη συντήρηση κώδικα.

Κάθε πακέτο μπορεί να περιέχει:

- **Κώδικα**: Σενάρια (scripts) που επεκτείνουν τη λειτουργικότητα του Unity.
- **Assets**: Υλικά, textures, ή μοντέλα που μπορούν να χρησιμοποιηθούν σε σκηνές.

- **Documentation**: Τεκμηρίωση που εξηγεί πώς να χρησιμοποιηθεί το πακέτο.

Λειτουργίες του Unity Package Manager

Εγκατάσταση Πακέτων
Μπορεί να γίνει εγκατάσταση νέων πακέτων, όπως το Cinemachine ή το TextMeshPro, τα οποία προσφέρουν προηγμένα εργαλεία για την ανάπτυξη παιχνιδιών.

Τα πακέτα μπορούν να προέρχονται από την επίσημη βιβλιοθήκη της Unity, το Asset Store, ή από ιδιωτικά αποθετήρια (repositories) όπως το GitHub.

Ενημέρωση Πακέτων
Ο Package Manager επιτρέπει την ενημέρωση των πακέτων στην τελευταία τους έκδοση, διατηρώντας την εφαρμογή σταθερή και σύγχρονη με τα τελευταία features και διορθώσεις ασφαλείας.

Αφαίρεση Πακέτων
Αν κάποιο πακέτο δεν είναι πλέον απαραίτητο για το project, ο Package Manager επιτρέπει την εύκολη απεγκατάστασή του για να μειωθεί το βάρος του έργου και να αποφεύγονται περιττές εξαρτήσεις.

Προβολή Πληροφοριών
Ο Package Manager εμφανίζει σημαντικές πληροφορίες για κάθε πακέτο, όπως την τρέχουσα έκδοση, τις αλλαγές που έχει υποστεί (changelog), καθώς και τις εξαρτήσεις που μπορεί να έχει με άλλα πακέτα.

Διαχείριση Εξαρτήσεων
Πολλά πακέτα μπορεί να εξαρτώνται από άλλα πακέτα για να λειτουργήσουν σωστά. Ο Package Manager χειρίζεται αυτόματα αυτές τις εξαρτήσεις, διασφαλίζοντας ότι όλα τα απαραίτητα στοιχεία θα εγκατασταθούν σωστά.

2.8.3 Text Mesh Pro

Το **Text Mesh Pro (TMP)** είναι ένα πανίσχυρο εργαλείο για τη δημιουργία και τη διαχείριση κειμένου στο Unity. Παρέχει εξελιγμένες δυνατότητες απόδοσης κειμένου με υψηλή ακρίβεια και ποιότητα, σε σχέση με το βασικό σύστημα κειμένου που προσφέρει το Unity. Το TMP προσφέρει ευέλικτες επιλογές τυπογραφίας και ενώ είναι πλέον ενσωματωμένο ως επίσημο στοιχείο του Unity, την πρώτη φορά που θα χρησιμοποιηθεί πρέπει να κατέβει ξεχωριστά.

Βασικά Χαρακτηριστικά του Text Mesh Pro:

Υψηλής Ποιότητας Κείμενο:
Το TMP αποδίδει κείμενο σε πολύ υψηλή ανάλυση και χρησιμοποιεί τεχνικές sub-pixel rendering για να προσφέρει καθαρό κείμενο, ανεξάρτητα από το μέγεθος του κειμένου ή την ανάλυση της οθόνης.

Κλιμάκωση Χωρίς Απώλειες (Scalability):
Σε αντίθεση με τα απλά fonts, τα TMP fonts διατηρούν την ευκρίνεια τους σε κάθε επίπεδο κλίμακας, κάτι που είναι ιδανικό για τις διαφορετικές αναλύσεις και μεγέθη στις οθόνες.

Κινούμενο Κείμενο (Text Animation):
Το TMP περιλαμβάνει δυνατότητες για εύκολη δημιουργία κινούμενων εφέ στο κείμενο, όπως χρώματα, σκιάσεις και παραμορφώσεις που μπορούν να εφαρμοστούν δυναμικά.

Υποστήριξη Υλικών (Material Support):

Το Text Mesh Pro υποστηρίζει τα Unity materials, επιτρέποντας τη χρήση textures και shaders για εφέ κειμένου, όπως λάμψη, ανακλάσεις ή ανάγλυφα.

Font Atlas Generation:

Το TMP δημιουργεί **Font Atlases**, που είναι υφές που περιέχουν όλα τα σύμβολα ενός συγκεκριμένου font σε υψηλή ποιότητα. Αυτό επιτρέπει την ταχύτερη απόδοση κειμένου σε σύγκριση με την παραδοσιακή μέθοδο.

Υποστήριξη Unicode:

Το TMP υποστηρίζει πλήρως τα **Unicode** σύμβολα, που σημαίνει ότι μπορούν να χρησιμοποιηθούνμ σύμβολα και κείμενο από πολλές διαφορετικές γλώσσες και γραφικά συστήματα.

Πλούσια Διάταξη Κειμένου (Rich Text):

Με τη δυνατότητα υποστήριξης Rich Text, μπορεί να ενσωματωθεί μορφοποιημένο κείμενο, επιτρέποντας αλλαγές στα χρώματα, στο μέγεθος και στη μορφοποίηση σε μια ενιαία σειρά κειμένου.

Προηγμένη Διαχείριση Κειμένου:

Το TMP προσφέρει προηγμένα χαρακτηριστικά διαχείρισης κειμένου, όπως **kerning**, **spacing**, **line justification**, και **wrapping**, επιτρέποντας πλήρη έλεγχο της εμφάνισης και της ροής του κειμένου.

Σύγκριση Text Mesh Pro με το Βασικό Σύστημα Κειμένου του Unity

Το Unity προσφέρει ένα βασικό σύστημα κειμένου, αλλά το **Text Mesh Pro** είναι σαφώς ανώτερο σε ό,τι αφορά την ποιότητα, την απόδοση και τις δυνατότητες διαμόρφωσης του κειμένου. Το TMP παρέχει δυνατότητες όπως:

- **Υψηλότερη ακρίβεια στην απόδοση κειμένου**.
- **Περισσότερες επιλογές μορφοποίησης και στυλ**.
- **Καλύτερη απόδοση σε μεγάλες οθόνες ή υψηλές αναλύσεις**.

2.8.4 Art και Sound Assets

Διάφορα art assets, όπως αντικείμενα, εικόνες, sprites αλλά και διάφοροι ήχοι, έχουν ληφθεί από το Asset Store. Τα assets αυτά δεν έχουν υποστεί κάποιου είδους επεξεργασία και έχουν χρησιμοποιηθεί όπως λήφθηκαν.

2.8.5 Path Creator

Ένα ακόμα asset που λήφθηκε από το asset store, είναι το Path Creator. Αυτό το asset περιέχει scripts που χρησιμοποιούνται για την δημιουργία μονοπατιών. Ακόμα περιέχει scripts για να κάνει αντικείμενα να ακολουθούν αυτό το μονοπάτι.

Το asset αυτό περιέχει διάφορες επιλογές για μονοπάτια, στο παιχνίδι έγινε χρήση μόνο των μονοπατιών Bezier. Δηλαδή, τοποθετώντας το μονοπάτι στην σκηνή, στην αρχή είναι μια γραμμή με αρχή και τέλος. Στην γραμμή αυτή μπορούμε να προσθέσουμε σημεία, δίνοντας του έτσι το σχήμα που θέλουμε.

Το script το οποίο επιτρέπει στο αντικείμενο να ακολουθήσει αυτό το μονοπάτι, επιτρέπει ακόμα και την ρύθμιση της ταχύτητας και της λογικής που θα ακολουθήσουν όταν φτάσουν στο τέλος του μονοπατιού. Υπάρχουν τρεις επιλογές:

- **Loop**: Σε αυτήν την επιλογή, μόλις ο NPC φτάσει στο τέλος του προκαθορισμένου μονοπατιού, ξεκινά αμέσως από την αρχή, επαναλαμβάνοντας την ίδια διαδρομή ξανά και ξανά χωρίς διακοπή.
- **Reverse**: Εδώ, όταν ο NPC φτάσει στο τέλος της διαδρομής του, αντί να ξεκινήσει πάλι από την αρχή, αλλάζει κατεύθυνση και κινείται αντίστροφα προς το σημείο εκκίνησης. Μόλις φτάσει στην αρχή, επαναλαμβάνει την ίδια διαδικασία, πηγαίνοντας ξανά προς το τέλος.
- **Stop**: Στην επιλογή αυτή, όταν ο NPC φτάσει στο τέλος της διαδρομής του, σταματά την κίνησή του και παραμένει στη θέση αυτή χωρίς να επιστρέφει ή να ξεκινά ξανά.

2.9 Visual Studio και C# σε Συνδυασμό με το Unity

Το Visual Studio και η γλώσσα προγραμματισμού C# είναι δύο σημαντικά εργαλεία που λειτουργούν άμεσα με το Unity για την ανάπτυξη παιχνιδιών και εφαρμογών. Ο συνδυασμός αυτών των εργαλείων δημιουργεί ένα πανίσχυρο περιβάλλον για να δημιουργήσουν, να δοκιμάσουν και να παραμετροποιήσουν την λογική και την λειτουργικότητα των παιχνιδιών.

2.9.1 Visual Studio

Το Visual Studio είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης λογισμικού (IDE) που παρέχεται από τη Microsoft και χρησιμοποιείται για τη συγγραφή και την επεξεργασία κώδικα. Είναι το προτεινόμενο IDE για προγραμματισμό στο Unity, καθώς υποστηρίζει τη γλώσσα C#, η οποία είναι η βασική γλώσσα scripting στο Unity.

Βασικά Χαρακτηριστικά του Visual Studio:

Επεξεργασία και Σύνταξη Κώδικα:

Το Visual Studio προσφέρει ισχυρά εργαλεία επεξεργασίας κώδικα, όπως αυτόματη συμπλήρωση (IntelliSense), έλεγχο συντακτικών λαθών σε πραγματικό χρόνο και ανάλυση του κώδικα. Το IntelliSense μπορεί να υποδεικνύει τις διαθέσιμες κλάσεις, μεθόδους και παραμέτρους κατά την σύνταξη του κώδικα, βοηθώντας την παραγωγικότητα και τη μείωση των λαθών.

Debugging (Αποσφαλμάτωση):

Το Visual Studio περιλαμβάνει ένα εξελιγμένο εργαλείο debugging, που επιτρέπει την εκτέλεση και την ανάλυση του κώδικα σε πραγματικό χρόνο. Μπορούν να χρησιμοποιηθούν **breakpoints**, να παρακολουθηθούν τιμές μεταβλητών και να εντοπιστούν και να διορθωθούν σφάλματα κατά τη διάρκεια της εκτέλεσης της εφαρμογής.

Integration με το Unity:

Υπάρχει άμεση ενσωμάτωση του Unity στο Visual Studio, επιτρέποντας τη σύνδεση του κώδικα και των scripts του Unity στο IDE. Κάθε φορά που γίνεται αποθήκευση ενός script στο Visual Studio, αυτόματα ενημερώνεται και το Unity.

Διαχείριση Έργων:

Μέσα από το Visual Studio μπορεί να γίνει διαχείριση των αρχείων του έργου, οργανώνοντας τα scripts και τις κλάσεις, ενώ υπάρχει και υποστήριξη για την προσθήκη εξωτερικών βιβλιοθηκών.

Refactoring (Αναδιάρθρωση Κώδικα):

Το Visual Studio προσφέρει λειτουργίες refactoring, που επιτρέπουν την βελτίωση της δομή του κώδικά χωρίς να αλλάξει η λειτουργικότητά του. Αυτό περιλαμβάνει αλλαγές ονομάτων μεθόδων και μεταβλητών, αναδιάταξη κώδικα και πολλά άλλα.

2.9.2 Γλώσσα Προγραμματισμού C#

Η C# είναι η κύρια γλώσσα προγραμματισμού που χρησιμοποιείται στο Unity για τη δημιουργία scripts. Είναι μια αντικειμενοστραφής γλώσσα που αναπτύχθηκε από τη Microsoft και ενσωματώνει χαρακτηριστικά όπως:

Object-Oriented Programming (OOP):

Η C# υποστηρίζει την αντικειμενοστραφή προσέγγιση, που σημαίνει ότι η δομή των δεδομένων και η συμπεριφορά τους μοντελοποιούνται μέσω αντικειμένων και κλάσεων. Αυτό καθιστά τη διαχείριση της λογικής του παιχνιδιού πιο οργανωμένη και εύκολα επαναχρησιμοποιήσιμη.

Στατική Τυποποίηση (Static Typing):

Η C# είναι μια στατικά τυποποιημένη γλώσσα, που σημαίνει ότι οι μεταβλητές δηλώνονται με συγκεκριμένους τύπους δεδομένων (int, float, string, κλπ.). Αυτό βοηθά στην αποτροπή σφαλμάτων χρόνου εκτέλεσης, καθώς πολλά πιθανά σφάλματα μπορούν να εντοπιστούν κατά τη σύνταξη του κώδικα.

Ευρεία Χρήση και Βιβλιοθήκες:

Η C# διαθέτει μια μεγάλη γκάμα βιβλιοθηκών και frameworks, που μπορούν να χρησιμοποιηθούν για την ανάπτυξη εφαρμογών και παιχνιδιών, και το Unity υποστηρίζει πλήρως την ενσωμάτωσή τους στα projects.

2.9.3 Συνεργασία του Unity με το Visual Studio και την C#

Κάνοντας χρήση του Unity για την ανάπτυξη παιχνιδιών, τα scripts που γράφονται είναι σε C# και επεξεργάζονται στο Visual Studio. Το Unity επιτρέπει τη δημιουργία scripts που ελέγχουν τη συμπεριφορά των αντικειμένων στο παιχνίδι, από τις κινήσεις των χαρακτήρων μέχρι τη διαχείριση φυσικών συστημάτων και τη δημιουργία UI.

Βασική ροή εργασίας:

Δημιουργία και επεξεργασία script:

Στο Unity, δημιουργείται ένα νέο C# script μέσω του Unity Editor. Όταν ανοιχθεί, ανοίγει αυτόματα στο Visual Studio, όπου μπορείτε να ξεκινήσει να γίνεται σύνταξη του κώδικα.

Σύνταξη και Αποθήκευση:

Κατά την σύνταξη του κώδικα, το IntelliSense του Visual Studio παρέχει προτάσεις για κλάσεις, μεθόδους και παραμέτρους, ενώ παράλληλα ελέγχει για συντακτικά λάθη. Όταν γίνει αποθήκευση του αρχείου, το Unity ανιχνεύει τις αλλαγές και μεταγλωττίζει τον κώδικα. Σε περίπτωση συντακτικών λαθών εμφανίζεται μήνυμα στο Console Panel.

Εκτέλεση του κώδικα στο Unity:

Στο Unity, ο κώδικας που έχει γραφτεί εφαρμόζεται στα αντίστοιχα Game Objects και το παιχνίδι εκτελείται με βάση τη λογική που έχει υλοποιηθεί στο script.

Debugging και Testing:

Χρησιμοποιώντας το Visual Studio, μπορείτε να γίνει debugging στον κώδικά ενώ το παιχνίδι τρέχει. Μπορεί να γίνει παρακολούθηση στις τιμές των μεταβλητών σε πραγματικό χρόνο και να εντοπιστούν τα σημεία του κώδικα που χρειάζονται βελτίωση ή διόρθωση.

Παραδείγματα Ενσωμάτωσης Unity και C#:

MonoBehaviour Scripts:

Κάθε C# script που δημιουργείται στο Unity βασίζεται στην κλάση MonoBehaviour, που επιτρέπει τον χειρισμό ενός αντικειμένου, όπως η αρχική του δημιουργία (Start), οι ενημερώσεις καρέ (Update) και η σύγκρουση με άλλα αντικείμενα.

Events και Input Handling:

Με τη χρήση της C# στο Unity, μπορεί να γίνει χειρισμός διαφόρων events, όπως το πάτημα κουμπιών, οι κινήσεις του ποντικιού ή η σύγκρουση αντικειμένων στο παιχνίδι.

Physics και Animation:

Μέσω του scripting στη C#, μπορούν να ελεγχθούν η φυσική και οι κινήσεις των αντικειμένων, καθώς και να δημιουργηθούν αλληλεπιδράσεις μεταξύ των διαφόρων components του Unity.

Η χρήση του Visual Studio ως IDE για τη συγγραφή C# scripts στο Unity παρέχει ένα ισχυρό και αποτελεσματικό περιβάλλον ανάπτυξης παιχνιδιών. Η C# επιτρέπει στους προγραμματιστές να δημιουργήσουν εξελιγμένη λογική παιχνιδιού, ενώ το Visual Studio υποστηρίζει τη διαδικασία με δυνατότητες όπως debugging, IntelliSense και εργαλεία refactoring. Ο συνδυασμός αυτών των τεχνολογιών επιτρέπει τη δημιουργία παιχνιδιών υψηλής ποιότητας με πιο αποτελεσματικό τρόπο.

2.10 Η Κλάση MonoBehaviour και Βασικές Μέθοδοι

Η MonoBehaviour είναι η κύρια κλάση από την οποία κληρονομούν τα περισσότερα scripts στο Unity. Είναι βασική για την ανάπτυξη παιχνιδιών καθώς παρέχει τη δομή για την επικοινωνία μεταξύ του παιχνιδιού και των scripts που ελέγχουν τη συμπεριφορά των αντικειμένων (GameObjects). Όταν δημιουργείτε ένα script στο Unity, αυτόματα κληρονομεί από την MonoBehaviour, δίνοντάς σας πρόσβαση σε πολλές σημαντικές λειτουργίες, όπως οι κύκλοι ζωής των αντικειμένων, το rendering, το input και την φυσική.

2.10.1 Η κλάση MonoBehaviour

Η MonoBehaviour είναι η θεμελιώδης κλάση για scripting στο Unity. Όλα τα scripts που κληρονομούν από αυτήν την κλάση μπορούν να συνδεθούν σε GameObjects και να εκτελούν συγκεκριμένες ενέργειες. Μέσω της MonoBehaviour, τα scripts μπορούν να χειριστούν τα GameObjects και τα Components τους,

να ανταποκρίνονται σε γεγονότα (events), και να ελέγχουν πότε και πώς να εκτελούνται ορισμένες λειτουργίες, όπως η ενημέρωση κάθε καρέ (frame) ή η αλληλεπίδραση με άλλα αντικείμενα.

Η MonoBehaviour έχει προκαθορισμένα σημεία στον κύκλο ζωής των GameObjects. Αυτά τα σημεία είναι χρήσιμα για την οργάνωση του κώδικα, ειδικά όταν χρειάζεστε να εκτελεστούν ενέργειες σε συγκεκριμένες χρονικές στιγμές.

2.10.2 Δήλωση GameObjects και των Components

Στο Unity, τα GameObjects και τα Components είναι οι βασικές δομές που καθορίζουν τη λειτουργία και τη συμπεριφορά των στοιχείων μέσα σε μια σκηνή. Η δήλωση και η διαχείρισή τους μέσω των scripts είναι κρίσιμη για την ανάπτυξη παιχνιδιών.

Ένα GameObject είναι οποιοδήποτε αντικείμενο που μπορεί να εμφανιστεί στη σκηνή του παιχνιδιού. Αυτό μπορεί να είναι ένα χαρακτήρα, ένα αντικείμενο, ένα περιβάλλον ή ακόμα και η κάμερα. Μπορούν να δηλωθούν μέσα στο script και να αποκτήσει κάποιος πρόσβαση σε αυτά.

Τα Components είναι τα χαρακτηριστικά ή οι συμπεριφορές που προσδιορίζουν πώς λειτουργεί ένα GameObject. Κάθε GameObject μπορεί να έχει πολλαπλά Components όπως Transform, Rigidbody, Collider, Scripts και άλλα, τα οποία μπορούν να δηλωθούν μέσα στο script και να αποκτήσει κάποιος πρόσβαση σε αυτά.

Το [SerializeField] είναι μια ατομική δήλωση που χρησιμοποιείται στο Unity για να επιτρέπει σε ιδιωτικά πεδία (private fields) να είναι ορατά στον Inspector του Unity. Αυτό είναι χρήσιμο γιατί μπορεί να διατηρηθεί η ασφάλεια των δεδομένων, ενώ ταυτόχρονα επιτρέπεται η παραμετροποίηση μέσω του Inspector.

2.10.3 Βασικές Μέθοδοι της MonoBehaviour

Start():Η μέθοδος Start() εκτελείται μία φορά κατά την αρχή της ζωής του GameObject, αμέσως μετά την εκτέλεση της Awake() και πριν από το πρώτο Update(). Χρησιμοποιείται κυρίως για την αρχικοποίηση μεταβλητών και για να τεθούν οι βασικές τιμές ή καταστάσεις πριν ξεκινήσει η λογική του παιχνιδιού.

Awake(): Η Awake() εκτελείται ακόμα νωρίτερα από το Start(), δηλαδή μόλις φορτωθεί το script ή δημιουργηθεί το GameObject. Είναι χρήσιμη για να οριστούν βασικές ενέργειες που πρέπει να συμβούν πριν από οτιδήποτε άλλο στο παιχνίδι, όπως η φόρτωση δεδομένων ή η αρχικοποίηση αντικειμένων που απαιτούνται από άλλες κλάσεις.

Update(): Η Update() είναι μία από τις πιο σημαντικές μεθόδους της MonoBehaviour. Καλείται σε κάθε καρέ (frame) του παιχνιδιού και χρησιμοποιείται για να εκτελείτε λογική που εξαρτάται από το χρόνο, όπως κίνηση χαρακτήρων, χειρισμός των inputs του χρήστη ή λογική AI.

FixedUpdate(): Η FixedUpdate() εκτελείται σε σταθερά χρονικά διαστήματα και είναι πιο κατάλληλη για λειτουργίες που σχετίζονται με φυσική (physics), όπως η κίνηση αντικειμένων ή η αλληλεπίδραση μέσω forces και collisions. Είναι ιδανική για κώδικα που πρέπει να είναι ανεξάρτητος από τον αριθμό των καρέ.

OnApplicationQuit(): Η OnApplicationQuit() καλείται όταν το παιχνίδι κλείνει. Είναι χρήσιμη για την εκτέλεση καθαριστικών ενεργειών, όπως η αποθήκευση δεδομένων ή η απελευθέρωση πόρων.

Instantiate(): Η Instantiate() χρησιμοποιείται για τη δημιουργία αντιγράφων (instances) ενός GameObject κατά τη διάρκεια του παιχνιδιού. Είναι μια ευρέως χρησιμοποιούμενη μέθοδος για την προσθήκη αντικειμένων στη σκηνή δυναμικά, όπως η δημιουργία εχθρών ή αντικειμένων από τον παίκτη.

Destroy():Η Destroy() χρησιμοποιείται για να καταστρέψει ένα GameObject ή ένα Component, αφαιρώντας το από τη σκηνή. Αυτό μπορεί να χρησιμοποιηθεί όταν δεν χρειάζεστε πια ένα αντικείμενο στο παιχνίδι, όπως όταν ένας εχθρός σκοτώνεται.

DontDestroyOnLoad(): Η DontDestroyOnLoad() χρησιμοποιείται για να εξασφαλίσει ότι ένα αντικείμενο δεν θα καταστραφεί όταν αλλάζει σκηνή στο παιχνίδι. Είναι χρήσιμη για αντικείμενα που πρέπει να παραμείνουν καθ' όλη τη διάρκεια του παιχνιδιού, όπως η μουσική στο background ή τα δεδομένα του παίκτη.

FindObjectOfType<>(): Η FindObjectOfType<>() επιστρέφει το πρώτο GameObject που βρέθηκε στη σκηνή με τον συγκεκριμένο τύπο που δίνεται ως γενικός τύπος (generic type). Χρησιμοποιείται για να βρεθούν αντικείμενα με συγκεκριμένα components χωρίς την ανάγκη αναφοράς σε αυτά από πριν.

Random.Range(): Η Random.Range() επιστρέφει μια τυχαία τιμή μέσα σε ένα συγκεκριμένο εύρος. Για ακέραιους αριθμούς (int), η τιμή θα είναι από την ελάχιστη τιμή έως και μία μονάδα πριν από τη μέγιστη. Για κινητές τιμές (float), περιλαμβάνονται και τα άκρα.

2.10.4 Βασικές Μέθοδοι σε Συνεργασία με GameObjects

Πολλές μέθοδοι επιτρέπουν τη διαχείριση και τον έλεγχο των GameObjects και των Components τους. Αυτές οι μέθοδοι είναι απαραίτητες για τη δομή και τη συμπεριφορά των αντικειμένων στο παιχνίδι.

SetActive(): Η SetActive() είναι μια μέθοδος που χρησιμοποιείται για να ενεργοποιήσει ή να απενεργοποιήσει ένα GameObject και τα παιδιά του (αν υπάρχουν). Όταν ένα GameObject είναι ανενεργό, δεν εκτελεί καμία ενέργεια και τα scripts του δεν ενημερώνονται. Το σημαντικό εδώ είναι ότι δεν καταστρέφεται αλλά απλώς «κρύβεται» από τη σκηνή και τον χρήστη. Είναι χρήσιμη για περιπτώσεις που θέλουμε να εμφανίζουμε ή να αποκρύπτουμε αντικείμενα, χωρίς να τα διαγράψουμε, π.χ. κουμπιά στο UI ή οπτικά εφέ που δεν πρέπει να φαίνονται πάντα.

FindGameObjectWithTag(): Η μέθοδος FindGameObjectWithTag() ψάχνει στη σκηνή για ένα GameObject που έχει ένα συγκεκριμένο Tag. Η μέθοδος επιστρέφει το πρώτο GameObject που ταιριάζει με το Tag που δόθηκε ως όρισμα.

GetComponent<>(): Η GetComponent<>() είναι μία από τις πιο σημαντικές μεθόδους στο Unity, καθώς επιτρέπει την πρόσβαση στα Components ενός GameObject. Κάθε GameObject μπορεί να περιέχει διάφορα Components, όπως Rigidbody, Collider, Renderer, ή custom scripts, και αυτή η μέθοδος δίνει τη δυνατότητα να τα χειριστούμε μέσα από το script. Είναι χρήσιμη όταν χρειάζεται να αλληλεπιδράσουμε με διάφορα στοιχεία του GameObject, όπως να εφαρμόσουμε δυνάμεις σε ένα Rigidbody ή να αλλάξουμε το κείμενο ενός Text component.

LoadScene(): Η μέθοδος LoadScene() χρησιμοποιείται για τη φόρτωση μιας νέας σκηνής στο Unity. Η σκηνή μπορεί να οριστεί είτε με το όνομά της είτε με το index της, όπως καθορίζεται στο build settings. Αυτή η μέθοδος είναι σημαντική για τη μετάβαση μεταξύ σκηνών, όπως για παράδειγμα από το main menu στο gameplay ή από το ένα επίπεδο στο άλλο.

SetParent(): Η μέθοδος SetParent() χρησιμοποιείται για να ορίσει το parent ενός GameObject, δημιουργώντας σχέσεις parent-child. Ένα GameObject που είναι child ενός άλλου ακολουθεί τη θέση, την

περιστροφή και την κλίμακα του parent. Αυτή η μέθοδος είναι ιδιαίτερα χρήσιμη όταν θέλουμε να μεταφέρουμε αντικείμενα σε διαφορετικά σημεία της ιεραρχίας κατά τη διάρκεια του παιχνιδιού. Χρησιμοποιείται συχνά για αντικείμενα που πρέπει να «δένονται» σε άλλα αντικείμενα, όπως για παράδειγμα μια κάμερα που ακολουθεί τον παίκτη ή ένα όπλο που προσαρμόζεται στο χέρι ενός χαρακτήρα.

SetAsFirstSibling(): Η μέθοδος SetAsFirstSibling() χρησιμοποιείται για να τοποθετήσει το GameObject στην πρώτη θέση της ιεραρχίας των αδελφών (siblings) του, δηλαδή μεταξύ άλλων παιδιών του ίδιου parent. Η σειρά των παιδιών στην ιεραρχία μπορεί να επηρεάσει την εμφάνιση στο UI, καθώς τα αντικείμενα με χαμηλότερο index εμφανίζονται πάνω από τα αντικείμενα με υψηλότερο index.

2.10.5 Μέθοδοι που Επηρεάζουν τον Χρόνο

Εάν θέλουμε να επιτύχουμε αναμονή ή καθυστερήσεις, μπορούμε να χρησιμοποιήσουμε έναν Enumerator. Είναι μια μέθοδος που επιστρέφει IEnumerator και χρησιμοποιείται συνήθως για την υλοποίηση μιας ακολουθίας ενεργειών που εκτελούνται με την πάροδο του χρόνου. Οι Enumerators χρησιμοποιούνται μαζί με τη μέθοδο StartCoroutine() για να δημιουργήσουν καθυστερήσεις ή να διαχειριστούν διαδικασίες που διαρκούν αρκετά frames.

Οι Enumerators περιέχουν την εντολή yield return. Η πιο συνηθισμένη μορφή είναι η χρήση του yield return new WaitForSeconds(), το οποίο επιτρέπει την αναμονή για συγκεκριμένο χρονικό διάστημα πριν από την εκτέλεση της επόμενης ενέργειας.

Το Time.timeScale είναι μια στατική ιδιότητα της κλάσης Time που καθορίζει την ταχύτητα με την οποία προχωρά ο χρόνος στο παιχνίδι. Είναι ένα από τα πιο χρήσιμα εργαλεία για τον έλεγχο του ρυθμού του παιχνιδιού. Ανάλογα την τιμή που παίρνει έχει και διαφορετικές επιδράσεις.

- Η προεπιλεγμένη τιμή του Time.timeScale είναι 1.0, που σημαίνει κανονική ροή του χρόνου.
- Μια τιμή μικρότερη από 1.0 (π.χ. 0.5) επιβραδύνει την ροή του χρόνου στο παιχνίδι, κάνοντάς το να κινείται πιο αργά.
- Μια τιμή μεγαλύτερη από 1.0 (π.χ. 2.0) επιταχύνει την ροή του χρόνου.

Χρησιμοποιείται συνήθως για εφέ slow-motion ή fast-forward. Επίσης, μπορεί να χρησιμοποιηθεί για την παύση του παιχνιδιού, όταν η τιμή ορίζεται στο 0.

Το Time.deltaTime είναι μια στατική ιδιότητα που επιστρέφει το χρονικό διάστημα που έχει περάσει από την τελευταία καρέ (frame). Είναι κρίσιμη για την επίτευξη ομαλής κίνησης και χρονισμού στο παιχνίδι. Είναι συνήθως πολύ μικρό, της τάξης των 0.016 (ή 16 ms) όταν το παιχνίδι τρέχει με 60 καρέ ανά δευτερόλεπτο. Αυτή η τιμή αλλάζει ανάλογα με το πόσο γρήγορα ή αργά τρέχει το παιχνίδι σε σχέση με το ρυθμό των καρέ. Συνήθως χρησιμοποιείται για την προσαρμογή της ταχύτητας κίνησης και της φυσικής, ώστε να είναι ανεξάρτητες από το frame rate.

2.10.6 Μέθοδοι για Εντοπισμού εισόδου

Για να διαχειριστούμε τις εισόδους από το πληκτρολόγιο και το ποντίκι, υπάρχουν συγκεκριμένες μέθοδοι που είναι απαραίτητες. Η μέθοδος Input.GetButtonDown() ανιχνεύει αν ένα συγκεκριμένο πλήκτρο πατήθηκε στο τρέχων frame. Αυτή η μέθοδος χρησιμοποιείται κυρίως για την εντόπιση του πρώτου στιγμιότυπου πατήματος ενός πλήκτρου, κάτι που είναι χρήσιμο όταν θέλουμε να ανιχνεύσουμε ένα στιγμιαίο γεγονός (όπως το άνοιγμα ενός μενού ή το άλμα του παίκτη). Τα πλήκτρα αυτά ορίζονται στο Input Manager και έχουν προσαρμοσμένα ονόματα

Αντίστοιχα για να ανιχνεύεται πότε θα πατηθεί κάποιο κουμπί από το ποντίκι υπάρχει η μέθοδος Input.GetMouseButtonDown(). Υπάρχουν τρεις κύριες μέθοδοι για τον εντοπισμό αυτό.

- Input.GetMouseButtonDown(0): Ελέγχει αν το αριστερό κουμπί του ποντικιού πατήθηκε.
- Input.GetMouseButtonDown(1): Ελέγχει αν το δεξί κουμπί του ποντικιού πατήθηκε.
- Input.GetMouseButtonDown(2): Ελέγχει αν το μεσαίο κουμπί (ροδέλα) πατήθηκε.

2.10.7 Χειρισμός Αρχείων

Ο χειρισμός αρχείων στο Unity είναι κρίσιμος για τη διαχείριση δεδομένων και την αποθήκευση πληροφοριών που χρησιμοποιούνται από το παιχνίδι. Το Unity παρέχει διάφορες μεθόδους και κλάσεις για τη διαχείριση αρχείων.

- **Path.Combine()**: Η μέθοδος Path.Combine χρησιμοποιείται για τη σύνθεση διαδρομών (paths) αρχείων και φακέλων. Είναι χρήσιμη για να διασφαλίσει ότι οι διαδρομές σχηματίζονται σωστά, ανεξάρτητα από το αν περιέχουν ή όχι χαρακτήρες διαχωρισμού (π.χ. "/" ή "").
- **File.Exists()**: Η μέθοδος File.Exists ελέγχει αν ένα αρχείο υπάρχει σε μια καθορισμένη διαδρομή. Επιστρέφει true αν το αρχείο υπάρχει και false αν δεν υπάρχει.
- **StreamReader.ReadToEnd()**: Ο StreamReader είναι μια κλάση που χρησιμοποιείται για την ανάγνωση δεδομένων από αρχεία. Η μέθοδος ReadToEnd διαβάζει ολόκληρο το περιεχόμενο ενός αρχείου και επιστρέφει το περιεχόμενο ως μια συμβολοσειρά.
- **JsonUtility.FromJson()/ToJson()**: Η κλάση JsonUtility παρέχει μεθόδους για την αναπαράσταση δεδομένων σε μορφή JSON. Η μέθοδος ToJson μετατρέπει ένα αντικείμενο σε μια συμβολοσειρά JSON, ενώ η FromJson αναπαριστά μια JSON συμβολοσειρά ως αντικείμενο.
- **Path.GetDirectoryName**: Η μέθοδος αυτή επιστρέφει τη διαδρομή του φακέλου που περιέχει ένα αρχείο, εφόσον δοθεί η διαδρομή του αρχείου.

2.10.8 Enum

Οι Enums στο Unity και στην C# γενικότερα είναι ειδικοί τύποι δεδομένων που επιτρέπουν τον ορισμό μιας ομάδα σταθερών τιμών. Αυτές οι σταθερές τιμές είναι συνήθως λογικά σχετιζόμενες και συχνά χρησιμοποιούνται για να αναπαραστήσουν κατάσταση, τύπους ή κατηγορίες.

Οφέλη Χρήσης Enums

- **Αναγνωσιμότητα:** Οι Enums βελτιώνουν την αναγνωσιμότητα του κώδικα. Αντί να χρησιμοποιούνται συμβολοσειρές για παράδειγμα, μπορούν να χρησιμοποιηθούν κατανοητές ονομασίες που εξηγούν τη σημασία τους, κάνοντας έτσι πιο κατανοητό τον κώδικα.

- **Προστασία από λάθη:** Χρησιμοποιώντας Enums, περιορίζονται οι πιθανές τιμές που μπορεί να λάβει μια μεταβλητή, γεγονός που μειώνει τα σφάλματα που προκύπτουν από την εισαγωγή λανθασμένων δεδομένων. Για παράδειγμα, αν υπάρχει έναν Enum για "PlayerState" με τις τιμές "Idle", "Running", "Jumping", ο προγραμματιστής μπορεί να είναι σίγουρος ότι η μεταβλητή μπορεί να πάρει μόνο αυτές τις τιμές.

2.10.9 Events

Στο Unity, τα events είναι μια ισχυρή μέθοδος για την αλληλεπίδραση με το UI (User Interface) ή άλλα αντικείμενα μέσα στο παιχνίδι. Χρησιμοποιούνται για να ανιχνεύσουν ενέργειες του χρήστη (π.χ. drag and drop), να προκαλέσουν συγκεκριμένες αντιδράσεις σε κλικ ή να χειριστούν αλληλεπιδράσεις με αντικείμενα.

.Invoke():Η .Invoke() είναι μια μέθοδος που καλεί ένα event ή μια delegate function που έχει οριστεί. Μπορεί να χρησιμοποιηθεί το .Invoke() για να ενεργοποιηθεί χειροκίνητα ένα event ή μια συνάρτηση μετά από ένα χρονικό διάστημα.

OnDrop(): Το OnDrop() είναι ένα event που χρησιμοποιείται σε περιβάλλοντα drag and drop. Ενεργοποιείται όταν ο χρήστης "αφήσει" (drop) ένα αντικείμενο σε ένα συγκεκριμένο σημείο. Χρησιμοποιείται συχνά σε UI συστήματα, όπου αντικείμενα μπορούν να μεταφερθούν και να τοποθετηθούν (drop) σε άλλες περιοχές ή slots. Για να λειτουργήσει σωστά, το αντικείμενο που θα δεχτεί το drop πρέπει να υλοποιεί το interface IDropHandler.

OnDrag(): Το OnDrag() ενεργοποιείται καθώς ο χρήστης σύρει ένα αντικείμενο (drag) γύρω από την οθόνη. Είναι μέρος του συστήματος για την αλληλεπίδραση με αντικείμενα που μπορούν να μεταφερθούν. Χρησιμοποιείται συνήθως σε περιβάλλοντα όπου ο χρήστης πρέπει να σύρει αντικείμενα (drag) σε νέες θέσεις. Υλοποιείται χρησιμοποιώντας το IDragHandler interface.

OnEndDrag(): Το OnEndDrag() ενεργοποιείται όταν ο χρήστης τελειώσει την ενέργεια του "συρσίματος" (drag). Δηλαδή, όταν το αντικείμενο σταματήσει να σύρεται. Είναι χρήσιμο για να ανιχνεύει πότε ο χρήστης ολοκληρώσει το drag ενός αντικειμένου και θέλουμε να σταματήσει η κίνηση ή να εκτελεστεί μια τελική ενέργεια.

OnPointerEnter(): Το OnPointerEnter() ενεργοποιείται όταν ο δείκτης του ποντικιού εισέρχεται στην περιοχή του αντικειμένου. Αυτό το event είναι πολύ χρήσιμο για την αλληλεπίδραση με UI αντικείμενα όταν ο χρήστης περνάει το ποντίκι πάνω από αυτά. Χρησιμοποιείται συνήθως για την εμφάνιση tooltips, την αλλαγή του χρώματος ή της εικόνας ενός κουμπιού όταν το ποντίκι το πλησιάζει.

OnPointerExit(): Το OnPointerExit() ενεργοποιείται όταν ο δείκτης του ποντικιού αποχωρεί από την περιοχή του αντικειμένου. Αυτό το event χρησιμοποιείται για να ανιχνεύσει πότε ο χρήστης βγάζει το ποντίκι του από ένα αντικείμενο. Χρησιμοποιείται για να αναιρέσουμε αλλαγές που κάναμε με το OnPointerEnter(), όπως την απόκρυψη tooltips ή την επαναφορά του χρώματος ή της εικόνας του αντικειμένου.

2.11 Scriptable Objects

Τα Scriptable Objects στο Unity είναι ένας τύπος δεδομένων που επιτρέπει την αποθήκευση και την οργάνωση δεδομένων ανεξάρτητα από τα σκηνικά αντικείμενα (GameObjects). Αυτή η λειτουργία είναι εξαιρετικά χρήσιμη για τη διαχείριση και τη χρήση δεδομένων που χρειάζονται αποθήκευση, χωρίς την ανάγκη δημιουργίας ενός αντικειμένου στην ίδια τη σκηνή. Τα Scriptable Objects βοηθούν στη διατήρηση της μνήμης και στη βελτιστοποίηση της απόδοσης του παιχνιδιού, ενώ προσφέρουν έναν καθαρό και οργανωμένο τρόπο διαχείρισης δεδομένων.

Ένα Scriptable Object είναι μια κλάση στο Unity που κληρονομεί από την βασική κλάση ScriptableObject και χρησιμοποιείται για την αποθήκευση δεδομένων σε ένα διαρκές αρχείο, το οποίο μπορεί να αναπαραχθεί ή να τροποποιηθεί χωρίς την ανάγκη να βρίσκεται στη σκηνή. Τα Scriptable

Objects δεν απαιτούν από ένα GameObject να τα φιλοξενεί, καθιστώντας τα ιδανικά για την αποθήκευση δεδομένων όπως στατιστικά χαρακτήρων, ρυθμίσεις παιχνιδιού, δεδομένα αποστολών, ήχους ή προφίλ όπλων.

2.11.1 Χαρακτηριστικά και Πλεονεκτήματα

Διαμοιραζόμενα Δεδομένα:
Μπορούν να χρησιμοποιηθούν σε πολλές σκηνές ή αντικείμενα, διαμοιράζοντας κοινά δεδομένα σε διαφορετικά GameObjects ή κώδικα χωρίς την ανάγκη πολλαπλών αντιγράφων.

Αντικειμενοστραφής Οργάνωση:
Βοηθούν στη δημιουργία μιας αντικειμενοστραφούς δομής δεδομένων, επιτρέποντας στους προγραμματιστές να οργανώνουν τα δεδομένα τους με τρόπο σαφή και διαχειρίσιμο, παρόμοιο με την κληρονομικότητα των κλάσεων στην αντικειμενοστραφή προγραμματιστική λογική.

Βελτίωση Απόδοσης:
Τα Scriptable Objects επιτρέπουν τη διατήρηση δεδομένων εκτός της σκηνής, αποφεύγοντας την ανάγκη επαναλαμβανόμενης δημιουργίας αντικειμένων στο παιχνίδι και τη συνεχή δέσμευση μνήμης.

Ευκολία Συντήρησης:
Καθώς τα Scriptable Objects μπορούν να τροποποιηθούν εύκολα μέσω του Unity Editor, η συντήρηση και η προσαρμογή των δεδομένων μπορεί να γίνει χωρίς να χρειαστεί τροποποίηση του κώδικα ή αναδιάταξη των σκηνών.

Ευέλικτη Χρήση:
Μπορούν να χρησιμοποιηθούν σε πλήθος εφαρμογών, όπως για τη διαχείριση στατιστικών, αντικειμένων, αποστολών, διαλόγων, ηχητικών δεδομένων και πολλών άλλων.

2.11.2 Χρήση των Scriptable Objects

Τα Scriptable Objects είναι εξαιρετικά χρήσιμα σε περιπτώσεις όπου χρειάζεται να γίνει αποθήκευση των δεδομένων που:

- **Χρησιμοποιούνται σε πολλά GameObjects**: Για παράδειγμα, ένα σύστημα με στατιστικά όπλων που χρησιμοποιούνται από πολλούς χαρακτήρες.
- **Δε χρειάζεται να υπάρχουν σε μια σκηνή**: Τα δεδομένα μπορούν να υπάρχουν ανεξάρτητα από τις σκηνές και να είναι προσβάσιμα από τον κώδικα ανά πάσα στιγμή.
- **Πρέπει να αποθηκευτούν με διαρκή τρόπο**: Μπορούν να αποθηκευτούν σε αρχεία και να χρησιμοποιηθούν εκ νέου κατά τη διάρκεια του παιχνιδιού ή της ανάπτυξης.

2.11.3 Δημιουργία των Scriptable Objects

Για την δημιουργία ενός Scriptable Object, πρέπει να ακολουθηθούν τα εξής βήματα:

1. **Δημιουργία της κλάσης ScriptableObject:** Αρχικά πρέπει να δημιουργηθεί μια νέα κλάση που να κληρονομεί από την κλάση ScriptableObject.

2. **Προσθήκη στο Unity Editor:** Η χρήση του χαρακτηρισμού [CreateAssetMenu] επιτρέπει την εύκολη δημιουργία αντικειμένων από το Scriptable Object μέσω του Unity Editor. Αυτός ο χαρακτηρισμός εμφανίζει την επιλογή στο μενού, ώστε να μπορούν να δημιουργηθούν αρχεία Scriptable Object από το Unity.

3. **Χρήση στο Project:** Μόλις δημιουργηθεί το Scriptable Object, τα δεδομένα μπορούν να γεμίσουν μέσω του Unity Editor και να χρησιμοποιηθούν σε οποιοδήποτε GameObject ή σύστημα του παιχνιδιού.

2.11.4 Scriptable Objects σε Συνδυασμό με άλλα Συστήματα

- **Event Systems**: Τα Scriptable Objects μπορούν να χρησιμοποιηθούν ως μέρος ενός συστήματος event, αποθηκεύοντας καταστάσεις ή πληροφορίες που θα μεταδοθούν σε άλλα μέρη του παιχνιδιού.

- **Διαχείριση Καταστάσεων**: Σε παιχνίδια με πολύπλοκες καταστάσεις, τα Scriptable Objects μπορούν να χρησιμοποιηθούν για την παρακολούθηση των καταστάσεων του παιχνιδιού, αποθηκεύοντας κρίσιμα δεδομένα χωρίς την ανάγκη σκηνικών αντικειμένων.

2.11.5 Σύγκριση με τα MonoBehaviour

Τα **Scriptable Objects** και τα **MonoBehaviour** είναι δύο βασικά εργαλεία του Unity, αλλά εξυπηρετούν διαφορετικούς σκοπούς:

- Τα **MonoBehaviour** χρησιμοποιούνται για να ορίσουν τη συμπεριφορά ενός αντικειμένου μέσα στη σκηνή.

- Τα **Scriptable Objects** αποθηκεύουν δεδομένα ανεξάρτητα από τις σκηνές και μπορούν να χρησιμοποιηθούν ως βάση για την οργάνωση και διαμοιρασμό δεδομένων σε διάφορα σημεία του έργου.

2.12 Dictionaries

Στη C# και στο Unity, το Dictionary είναι μια γενική συλλογή (generic collection) που αποθηκεύει δεδομένα σε μορφή "κλειδί-τιμή" (key-value pairs). Ένα Dictionary επιτρέπει την γρήγορη αποθήκευση και την αναζήτηση τιμών με βάση ένα μοναδικό κλειδί, καθιστώντας το μια πολύ αποδοτική δομή δεδομένων για περιπτώσεις όπου χρειάζεται να γίνει συσχέτιση και προσπέλαση δεδομένων με βάση μοναδικά αναγνωριστικά.

Βασικά Χαρακτηριστικά του Dictionary:

- **Κλειδιά (Keys)**: Κάθε τιμή μέσα στο Dictionary συνδέεται με ένα μοναδικό κλειδί, το οποίο χρησιμοποιείται για την αναζήτησή της.

- **Τιμές (Values)**: Οι τιμές αντιπροσωπεύουν τα δεδομένα που αποθηκεύονται και μπορούν να είναι οποιουδήποτε τύπου.
- Τα **κλειδιά πρέπει να είναι μοναδικά**: Δεν μπορεί να υπάρχουν διπλότυπα κλειδιά στο Dictionary.
- Η αναζήτηση είναι πολύ γρήγορη, καθώς το Dictionary χρησιμοποιεί hashing για να βρει τις τιμές.

Κύριες Μέθοδοι και Ιδιότητες του Dictionary

- **Add(key, value)**: Προσθέτει ένα νέο ζεύγος κλειδιού-τιμής στο Dictionary.
- **Remove(key)**: Αφαιρεί ένα ζεύγος κλειδιού-τιμής από το Dictionary.
- **TryGetValue(key, out value)**: Προσπαθεί να αναζητήσει μια τιμή με βάση το κλειδί και επιστρέφει αν βρέθηκε ή όχι.
- **ContainsKey(key)**: Ελέγχει αν υπάρχει το κλειδί μέσα στο Dictionary.
- **Count**: Επιστρέφει τον αριθμό των στοιχείων (ζευγών κλειδιού-τιμής) που υπάρχουν στο Dictionary.

Οι μέθοδοι OnBeforeSerialize() και OnAfterDeserialize() είναι ειδικές μέθοδοι που χρησιμοποιούνται στην σειριοποίηση (serialization) και την αποσειριοποίηση (deserialization) δεδομένων στο Unity. Αυτές οι μέθοδοι εφαρμόζονται όταν ένα αντικείμενο πρέπει να μετατραπεί σε μορφή που μπορεί να αποθηκευτεί ή να μεταφερθεί (serialization) ή να επαναφερθεί σε λειτουργική κατάσταση (deserialization).

OnBeforeSerialize()

Αυτή η μέθοδος καλείται πριν από την έναρξη της σειριοποίησης ενός αντικειμένου. Είναι χρήσιμη όταν πρέπει να μετατραπούν ή να προετοιμαστούν δεδομένα πριν αποθηκευτούν. Για παράδειγμα, μπορεί να γίνει χειρισμός ή οργάνωση των δεδομένων πριν αποθηκευτούν σε ένα αρχείο ή στα metadata του Unity Editor.

OnAfterDeserialize()

Αυτή η μέθοδος καλείται μετά την ολοκλήρωση της αποσειριοποίησης. Χρησιμοποιείται για να επαναφερθούν τα δεδομένα ή να ανακατασκευαστεί η δομή τους μετά την ανάγνωση από αποθηκευμένη κατάσταση. Με αυτή τη μέθοδο, μπορούν να γίνουν προσαρμογές στα δεδομένα μετά την ανάκτηση τους από ένα αρχείο ή από ένα scriptable object.

Αυτές οι μέθοδοι είναι απαραίτητες όταν αποθηκεύονται δεδομένα σε μορφή JSON, ειδικά για σύνθετες δομές δεδομένων όπως Dictionaries.

2.13 Optimization στο Unity2D

Η βελτιστοποίηση (optimization) είναι ένα κρίσιμο κομμάτι της ανάπτυξης παιχνιδιών στο Unity. Η σωστή βελτιστοποίηση μπορεί να βελτιώσει την απόδοση του παιχνιδιού, μειώνοντας τη χρήση της CPU και της GPU, εξασφαλίζοντας ομαλή λειτουργία σε όλες τις συσκευές. Ακολουθούν διάφορες στρατηγικές και τεχνικές βελτιστοποίησης για τα Unity 2D παιχνίδια.

1. Βελτιστοποίηση Sprites

Τα sprites είναι θεμελιώδη στοιχεία στα 2D παιχνίδια και καταναλώνουν μεγάλο μέρος των πόρων, ειδικά όταν δεν έχουν βελτιστοποιηθεί.

- **Sprite Atlas**: Μπορεί να γίνει χρήση Sprite Atlases για να γίνει ομαδοποίηση των sprites σε μία υφή (texture). Αυτό μειώνει τα texture swaps και τα draw calls, βελτιώνοντας σημαντικά την απόδοση.
- **Compression**: Συμπίεση των υφών (textures) με κατάλληλα formats για να γίνει μειώσει της χρήσης μνήμης. Η συμπίεση βοηθάει στο να μειωθεί το μέγεθος του αρχείου των sprites χωρίς μεγάλη απώλεια ποιότητας.
- **Pixel Per Unit (PPU)**: Ρύθμιση σωστών τιμών του PPU στα sprites για να εξασφαλιστεί ότι αποδίδονται στις σωστές αναλογίες και χωρίς παραμόρφωση, εξοικονομώντας πόρους.

2. Μείωση Draw Calls

Κάθε φορά που το Unity στέλνει δεδομένα στη GPU για να αποδώσει (render) ένα αντικείμενο, γίνεται ένα draw call. Τα πολλά draw calls μπορούν να μειώσουν την απόδοση.

- **Batching**: Χρησιμοποιώντας το Dynamic Batching και το Static Batching για να γίνει ομαδοποίηση των αντικειμένων με παρόμοια χαρακτηριστικά σε λιγότερα draw calls.
- **Sprite Sorting Layers**: Κάνοντας βελτιστοποίηση στο sorting layers και στο order in layer ώστε να μειωθεί ο αριθμός των αντικειμένων που πρέπει να αποδοθούν ταυτόχρονα.

3. Βελτιστοποίηση Φυσικής (Physics)

Η φυσική μπορεί να είναι μια ακριβή λειτουργία, ακόμη και σε 2D παιχνίδια.

- **Collision Layers**: Να γίνει μείωση των συγκρούσεων μεταξύ αντικειμένων μέσω των collision layers. Αυτό επιτρέπει στον υπολογιστή φυσικής να αγνοεί αντικείμενα που δεν χρειάζεται να συγκρούονται, μειώνοντας τις συγκρούσεις που πρέπει να υπολογιστούν.
- **FixedUpdate**: Οι υπολογισμοί φυσικής να γίνονται μόνο όπου είναι απαραίτητο, χρησιμοποιώντας το FixedUpdate() σωστά για τις κινήσεις που χρειάζονται ακριβείς υπολογισμούς φυσικής.

4. Σενάρια (Scripts) και Performance

Ο τρόπος που γράφονται τα scripts έχει μεγάλο αντίκτυπο στην απόδοση.

- **Avoid Update Overuse**: Αποφυγή χρήσης της μεθόδου Update() αλόγιστα, ειδικά για λειτουργίες που δεν χρειάζεται να εκτελούνται κάθε καρέ. Αντίθετα, μπορεί να γίνει χρήση του InvokeRepeating(), Coroutines, ή να γίνει έλεγχος συγκεκριμένων συνθηκών μέσα στην Update(), έτσι ώστε να μην εκτελείται συνέχεια άσκοπα.

Κεφάλαιο 3

3. Μεθοδολογία

Στο κεφάλαιο αυτό αναλύεται η μεθοδολογία που ακολουθήθηκε για την ανάπτυξη του παιχνιδιού, έχει χρησιμοποιηθεί μια προσαρμοσμένη εκδοχή της ευέλικτης ανάπτυξης λογισμικού (Agile), η οποία επιτρέπει την τμηματική υλοποίηση και συνεχή βελτίωση του έργου, διασφαλίζοντας ότι κάθε στάδιο ανάπτυξης είναι πλήρως λειτουργικό και δοκιμασμένο. Συγκεκριμένα, η μεθοδολογία θα περιλαμβάνει τα παρακάτω στάδια.

3.1 Συλλογή και Ανάλυση Απαιτήσεων

Σε αυτή τη φάση, θα καθοριστούν οι λειτουργικές και μη λειτουργικές απαιτήσεις του παιχνιδιού. Ο στόχος είναι η δημιουργία ενός ολοκληρωμένου αρχικού σχεδίου που περιλαμβάνει τα βασικά χαρακτηριστικά που θα χρησιμοποιηθούν. Οι λειτουργικές απαιτήσεις περιλαμβάνουν όλα τα χαρακτηριστικά και τις δυνατότητες που πρέπει να περιέχει το παιχνίδι. Οι μη λειτουργικές απαιτήσεις, από την άλλη, αφορούν στοιχεία όπως η απόδοση του παιχνιδιού, η επεκτασιμότητα του συστήματος και η φιλικότητα προς τον χρήστη (UX).

3.2 Σχεδιασμός Συστήματος και Αρχιτεκτονική

Αφού έχουν καθοριστεί οι απαιτήσεις, ακολουθεί ο σχεδιασμός της αρχιτεκτονικής του συστήματος. Σε αυτό το στάδιο, λαμβάνονται αποφάσεις σχετικά με την υλοποίηση των βασικών μηχανισμών του παιχνιδιού, όπως:

- Δομή κώδικα: Χωρισμός του έργου σε επιμέρους μονάδες (modules), όπως σύστημα μάχης, σύστημα NPC, inventory, UI κ.λπ., με ξεκάθαρες διασυνδέσεις.
- Σχεδίαση μηχανισμών: Λεπτομερής σχεδίαση των βασικών συστημάτων του παιχνιδιού, όπως το πώς θα λειτουργεί το leveling, το crafting των υλικών, και το σύστημα μάχης.
- Επιλογή τεχνολογιών: Απόφαση για τα εργαλεία και τις βιβλιοθήκες που θα χρησιμοποιηθούν, όπως για τη διαχείριση ήχου, γραφικών και φυσικής.

3.3 Υλοποίηση

Εφόσον έχει γίνει ο σχεδιασμός και έχουν καθοριστεί οι απαιτήσεις που απαιτούνται, μπορεί να ξεκινήσει η ανάπτυξη, η οποία θα πραγματοποιηθεί σε διαδοχικά κομμάτια, με κάθε κομμάτι να επικεντρώνεται σε συγκεκριμένα στοιχεία ή συστήματα του παιχνιδιού, όπως η αλληλεπίδραση με τους NPCs, το σύστημα επιπέδων, το σύστημα μάχης, και το UI. Για κάθε σύστημα, θα γίνεται ανάπτυξη, ενσωμάτωση, και αρχικός έλεγχος του κάθε χαρακτηριστικού.

3.4 Έλεγχος και Δοκιμές

Κατά τη διάρκεια και μετά την ολοκλήρωση κάθε συστήματος, πραγματοποιούνται δοκιμές για να διασφαλιστεί η ομαλή λειτουργία και η ποιότητα του παιχνιδιού. Οι έλεγχοι περιλαμβάνουν:

- Unit testing: Έλεγχος μεμονωμένων κομματιών κώδικα (π.χ. έλεγχος της σωστής λειτουργίας ενός συστήματος inventory).
- Integration testing: Έλεγχος της ομαλής λειτουργίας όταν τα επιμέρους υποσυστήματα ενσωματώνονται μεταξύ τους (π.χ. η σύνδεση μεταξύ inventory και του συστήματος μάχης).
- Playtesting: Δοκιμές για την αξιολόγηση της εμπειρίας παιχνιδιού (UX) και τη διαπίστωση τυχόν σφαλμάτων ή δυσκολιών που δεν είχαν εντοπιστεί από άλλες δοκιμές.

Στόχος των δοκιμών είναι να εντοπιστούν σφάλματα, να αξιολογηθεί η σταθερότητα του παιχνιδιού και να διασφαλιστεί ότι όλα τα συστήματα λειτουργούν όπως αναμένεται.

3.5 Συνεχής Ανατροφοδότηση και Βελτιστοποίηση

Η μεθοδολογία Agile επιτρέπει την τακτική ανατροφοδότηση σε κάθε φάση ανάπτυξης, κάτι που επιτρέπει να γίνουν διορθώσεις και βελτιώσεις με ευελιξία. Οι αλλαγές και οι διορθώσεις μπορεί να περιλαμβάνουν:

- Βελτιστοποίηση συστημάτων: Προσαρμογή του κώδικα για καλύτερη απόδοση ή διόρθωση τυχόν προβλημάτων που εντοπίζονται κατά τους ελέγχους.
- Αναπροσαρμογή σχεδιασμού: Αν χρειαστεί, αλλαγές στους μηχανισμούς ή στο gameplay, με βάση τα αποτελέσματα από τους ελέγχους και τις ανάγκες των χρηστών.

Αυτή η φάση είναι συνεχής καθ' όλη τη διάρκεια ανάπτυξης, εξασφαλίζοντας ότι το παιχνίδι παραμένει λειτουργικό και βελτιώνεται σταδιακά.

3.6 Ολοκλήρωση

Το τελικό στάδιο περιλαμβάνει την ολοκλήρωση του παιχνιδιού και την οριστική δοκιμή όλων των χαρακτηριστικών και συστημάτων. Σε αυτή τη φάση γίνονται:

- Τελικές δοκιμές: Εκτελούνται δοκιμές στην πλατφόρμα που μας αφορά αλλά σε διαφορετικές διαμορφώσεις για παράδειγμα, σε διαφορετικές συσκευές ή σε διαφορετικές αναλύσεις οθόνης, για να διασφαλιστεί ότι το παιχνίδι είναι σταθερό.

- Βελτιστοποίηση απόδοσης: Τελικές βελτιστοποιήσεις για να διασφαλιστεί ότι το παιχνίδι λειτουργεί απρόσκοπτα χωρίς προβλήματα απόδοσης ή καθυστερήσεις.

Αυτή η μεθοδολογία διασφαλίζει ότι το παιχνίδι θα υλοποιηθεί με οργανωμένο τρόπο, θα δοκιμαστεί επαρκώς και θα προσφέρει μια ποιοτική εμπειρία.

Κεφάλαιο 4

4. Αρχιτεκτονική και Υλοποίηση Συστήματος

Στο παρόν κεφάλαιο θα παρουσιαστούν και θα αναλυθούν λεπτομερώς τα συστήματα και οι μηχανισμοί που έχουν υλοποιηθεί στην εφαρμογή. Ειδικότερα, θα εξεταστούν οι τεχνικές πτυχές που αφορούν την ανάπτυξη και ενσωμάτωση των λειτουργιών της εφαρμογής. Όσον αφορά τα assets που έχουν χρησιμοποιηθεί, αυτά περιλαμβάνουν τα καλλιτεχνικά στοιχεία (art assets), όπως είναι οι χαρακτήρες, τα animations, το παρασκήνιο, δηλαδή η σχεδίαση του περιβάλλοντος μέσω των tiles, αλλά και όλα τα στοιχεία που σχετίζονται με το User Interface (UI), καθώς και οι ήχοι, όπως τα Sound Effects (SFX) και η μουσική. Επειδή έχουν ληφθεί από ελεύθερες πηγές που τα προσφέρουν δωρεάν και δεν έχουν υποστεί περαιτέρω επεξεργασία ή προσαρμογή, στο παρόν κεφάλαιο δεν θα πραγματοποιηθεί κάποια ανάλυση αυτών των assets, καθώς δεν εμπίπτουν στον κύριο τομέα ανάπτυξης της εφαρμογής, αλλά αποτελούν υλικό που έχει αξιοποιηθεί όπως διατέθηκε από τις αρχικές πηγές.

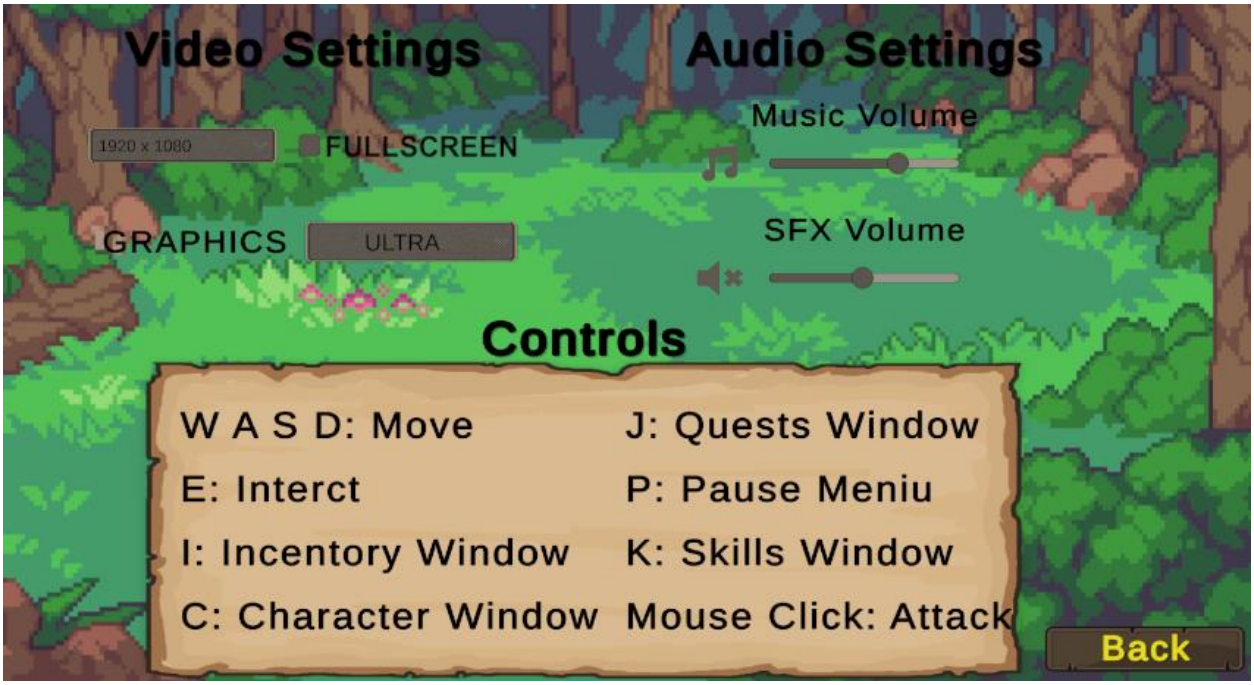
4.1 Μενού και Διεπαφή Χρήστη (User Interface)

Η εμπειρία του χρήστη ξεκινά από τη στιγμή που ανοίγει την εφαρμογή και αλληλεπιδρά με το αρχικό μενού του παιχνιδιού. Από εκεί, ο χρήστης έχει τη δυνατότητα να επιλέξει ανάμεσα σε διάφορες βασικές λειτουργίες όπως να ξεκινήσει το παιχνίδι, να ρυθμίσει παραμέτρους μέσω των Settings, να δει τα Credits, ή να κάνει έξοδο από το παιχνίδι μέσω της επιλογής Exit. Το UI που χρησιμοποιείται στο αρχικό μενού και σε όλες τις υπόλοιπες οθόνες, όπως το Inventory, τα καταστήματα και τα στατιστικά του παίκτη, βασίζεται σε προκατασκευασμένα assets από το Unity Asset Store, εξασφαλίζοντας έτσι οπτική συνοχή και εμφάνιση.

4.1.1 Μενού

Η πρώτη οθόνη που αντικρίζει ο χρήστης είναι το αρχικό μενού, το οποίο περιλαμβάνει τις εξής επιλογές με τη μορφή κουμπιών, τα οποία συνοδεύονται από ηχητικά εφέ όταν πατιούνται:

- **Start**: Η επιλογή αυτή φορτώνει την κεντρική σκηνή του παιχνιδιού, επιτρέποντας στον παίκτη να ξεκινήσει το παιχνίδι.
- **Options**: Ανοίγει ένα παράθυρο όπου ο χρήστης μπορεί να προσαρμόσει την ένταση του ήχου και να αλλάξει την ανάλυση της οθόνης για να βελτιώσει την εμπειρία παιχνιδιού του. Ακόμα σε αυτό μπορεί να δει όλα τα κουμπιά που θα χρειαστεί να χρησιμοποιήσει κατά την διάρκεια του παιχνιδιού.
- **Credits**: Πατώντας το κουμπί αυτό, ο χρήστης μπορεί να δει τα ονόματα των δημιουργών του παιχνιδιού, καθώς και άλλων συντελεστών.
- **Exit**: Κλείνει την εφαρμογή και επιστρέφει τον χρήστη στην επιφάνεια εργασίας.



Εικόνα 2. Παράθυρο Ρυθμίσεων

4.1.2 Διεπαφή Χρήστη User Interface

Η διεπαφή χρήστη αποτελείται από διάφορα παράθυρα που επιτρέπουν την αλληλεπίδραση και την ενημέρωση του παίκτη σχετικά με την κατάστασή του στο παιχνίδι. Ορισμένα παράθυρα, όπως τα Stats, το Toolbar και το Minimap, είναι μόνιμα ορατά κατά τη διάρκεια του παιχνιδιού και παρέχουν σημαντικές πληροφορίες για την πρόοδο και την κατάσταση του παίκτη. Τα υπόλοιπα παράθυρα μπορούν να ανοιχτούν μέσω πληκτρολογίου ή αλληλεπίδρασης με NPCs.

Όλα τα παράθυρα μπορούν να κλείσουν είτε πατώντας το κουμπί, από το πληκτρολόγιο, με το οποίο άνοιξαν, είτε πατώντας το κουμπί X, σε όσα υπάρχει, είτε πατώντας το κουμπί Escape, από το πληκτρολόγιο.

Stats: Αυτό το παράθυρο βρίσκεται στην κάτω αριστερή γωνία της οθόνης και εμφανίζει πληροφορίες σχετικά με τη ζωή και το mana του παίκτη, παρέχοντας συνεχή ενημέρωση για την υγεία και την ενέργεια του χαρακτήρα. Ακόμα σε αυτό φαίνεται και το τρέχων επίπεδο του παίκτη.

Εικόνα 3. Παράθυρο που φαίνεται η Ζωή το Mana και το Επίπεδο

Toolbar: Στο κάτω μέρος της οθόνης, το Toolbar εμφανίζει τα αντικείμενα που έχει στην κατοχή του ο παίκτης. Ο παίκτης μπορεί να τα επιλέξει και να τα χρησιμοποιήσει γρήγορα πατώντας το αντίστοιχο κουμπί.

Εικόνα 4. Το παράθυρο Toolbar

Minimap: Το minimap στην πάνω δεξιά γωνία της οθόνης επιτρέπει στον παίκτη να δει μια κάτοψη της περιοχής γύρω του, βοηθώντας τον στην πλοήγηση και τον εντοπισμό σημαντικών τοποθεσιών.

Τα υπόλοιπα παράθυρα ανοίγουν κατόπιν εντολής από τον χρήστη ή μέσα από συγκεκριμένες ενέργειες, όπως η αλληλεπίδραση με NPCs.

Inventory

Το παράθυρο του αποθέματος (Inventory) ανοίγει όταν ο χρήστης πατήσει το πλήκτρο "I" στο πληκτρολόγιο. Σε αυτό το παράθυρο εμφανίζονται όλα τα αντικείμενα που ο παίκτης κουβαλάει μαζί του, οργανωμένα σε ένα καθαρό και εύχρηστο πλέγμα. Κάθε αντικείμενο αντιπροσωπεύεται από ένα κουτάκι στο πλέγμα, όπου κάθε κουτάκι αντιστοιχεί σε μια θέση που μπορεί να περιέχει ένα μόνο αντικείμενο ανά κατηγορία, με εξαίρεση τα αντικείμενα που διαθέτουν την ιδιότητα Stack, τα οποία επιτρέπουν την αποθήκευση περισσότερων από ένα αντικειμένων στην ίδια θέση.

Η διάταξη του Inventory βασίζεται σε ένα Grid Layout Group, το οποίο δημιουργεί ένα οπτικά τακτοποιημένο και εύκολο στη χρήση πλέγμα για την τοποθέτηση των αντικειμένων. Κάθε κουτί λειτουργεί ως ένα κουμπί που μπορεί να αλληλεπιδράσει με τον χρήστη. Ο χρήστης μπορεί να επιλέξει τα αντικείμενα και να τα μετακινήσει είτε εντός του Inventory είτε μεταξύ του Toolbar και του Inventory. Η μετακίνηση πραγματοποιείται με drag-and-drop μέσω της χρήσης των μεθόδων OnBeginDrag(), OnDrag(), και OnEndDrag(), επιτρέποντας την εύκολη και ευέλικτη διαχείριση των αντικειμένων.

Εικόνα 5. Το παράθυρο του Inventory

Επιπλέον, ο χρήστης έχει τη δυνατότητα να χρησιμοποιήσει ένα αντικείμενο απλά πατώντας πάνω του. Αυτό προσφέρει άμεση πρόσβαση σε σημαντικά αντικείμενα, όπως φίλτρα, όπλα ή εργαλεία, βελτιώνοντας την αλληλεπίδραση και την απόκριση του χαρακτήρα σε πραγματικό χρόνο.

Interact

Όταν ο παίκτης πλησιάσει σε κάποιο NPC με το οποίο μπορεί να αλληλεπιδράσει, εμφανίζεται ένα διακριτικό παράθυρο ειδοποίησης στην οθόνη. Αυτό το παράθυρο ενημερώνει τον παίκτη να πατήσει το πλήκτρο “E” για να ξεκινήσει η αλληλεπίδραση, ανοίγοντας έτσι το παράθυρο του διαλόγου. Το σύστημα αυτό προσφέρει έναν ομαλό τρόπο για τον παίκτη να γνωρίζει πότε είναι δυνατή η αλληλεπίδραση, ενώ ταυτόχρονα διατηρεί τον ρυθμό του παιχνιδιού χωρίς να τον αποσπά.

Εικόνα 6. Παράθυρο ειδοποίησης του παίκτη για αλληλεπίδραση με NPC

Character

Πατώντας το πλήκτρο "C", ο παίκτης ανοίγει το παράθυρο του χαρακτήρα, το οποίο του προσφέρει μια ολοκληρωμένη εικόνα για τις ιδιότητες (Attributes) και τα χαρακτηριστικά (Stats) του χαρακτήρα του. Σε αυτό το σημείο, ο παίκτης μπορεί να διαχειριστεί βασικές παραμέτρους που αφορούν την ανάπτυξη του χαρακτήρα, όπως η υγεία (Health), η ζημιά (Damage) που προκαλεί στους εχθρούς και η συνολική εμπειρία (Experience) που έχει συσσωρεύσει. Ο χρήστης έχει επίσης τη δυνατότητα να προσθέσει ή να αλλάξει εξοπλισμό (Equipment) ανάλογα με τις ανάγκες του.

Στο παράθυρο αυτό εμφανίζονται κενές θέσεις (slots) για εξοπλισμό. Κάθε θέση αντιστοιχεί σε ένα συγκεκριμένο είδος αντικειμένου, όπως κράνος, θώρακα, όπλο, ή πανοπλία. Ο χρήστης μπορεί να τοποθετήσει τον κατάλληλο εξοπλισμό στις θέσεις αυτές και να βελτιώσει τις δυνατότητες του χαρακτήρα. Ανάλογα με το είδος του εξοπλισμού, κάθε αντικείμενο έχει επιπτώσεις σε διαφορετικά χαρακτηριστικά του χαρακτήρα, όπως η άμυνα (Defense) ή η ζημιά (Melee Damage).

Καθώς ο παίκτης ανεβαίνει επίπεδα, έχει τη δυνατότητα να αναβαθμίζει τις ιδιότητές του επιλέγοντας πώς θα κατανεμηθούν οι πόντοι που αποκτά. Σε περίπτωση που ο χρήστης επιθυμεί να δοκιμάσει διαφορετικό συνδυασμό ιδιοτήτων, υπάρχει επιλογή να επαναφέρει τις αναβαθμίσεις του και να επιλέξει ξανά ποια χαρακτηριστικά θα βελτιώσει.

Τα χαρακτηριστικά (Stats), όπως η υγεία (Health), η ζημιά (Damage) και η εμπειρία (EXP), αποτελούν το βασικότερο μέσο με το οποίο ο παίκτης κατανοεί τις δυνατότητες του χαρακτήρα του. Αν και ο παίκτης δεν μπορεί να αλλάξει άμεσα τα στατιστικά αυτά, οι τιμές τους ανεβαίνουν ανάλογα με τις ιδιότητες που επιλέγει να αναβαθμίσει. Για παράδειγμα, η αύξηση της ιδιότητας Strength ενισχύει τη ζημιά σώμα με σώμα, ενώ η άμυνα (Defense) εξαρτάται κυρίως από τον εξοπλισμό που φοράει ο παίκτης. Η εμπειρία (EXP) αυξάνεται καθώς ο παίκτης εξολοθρεύει εχθρούς ή ολοκληρώνει αποστολές.

Εικόνα 7. Το παράθυρο του χαρακτήρα

Ο χρήστης μπορεί να βάλει εξοπλισμό στα κενά κουτάκια, κάθε κουτάκι παίρνει μόνο ένα συγκεκριμένο είδος εξοπλισμού, το οποίο αναγράφεται και πάνω. Στο παράθυρο αυτό μπορεί ακόμα να αναβαθμίσει τις ιδιότητές του κάθε φορά που ανεβαίνει επίπεδο. Έχει επίσης την επιλογή να τις επαναφέρει από την αρχή σε περίπτωση που θέλει να διαλέξει έναν άλλο συνδυασμό για να τα αναβαθμίσει.

Τα χαρακτηριστικά (Stats) δείχνουν τις δυνατότητες του παίκτη, δηλαδή πόση ζωή έχει, την ζημιά που κάνει στους εχθρούς, την εμπειρία που έχει και άλλα. Οι τιμές σε αυτά δεν μπορούν να πειραχτούν από την χρήστη, όχι άμεσα τουλάχιστον. Ανεβάζοντας τα Attributes ανεβαίνουν και τα αντίστοιχα Stats κατά κάποιο ποσοστό. Για παράδειγμα, η αύξηση της ιδιότητας Strength ενισχύει τη ζημιά, ενώ η άμυνα (Defense) εξαρτάται κυρίως από τον εξοπλισμό που φοράει ο παίκτης. Η εμπειρία (EXP) αυξάνεται καθώς ο παίκτης εξολοθρεύει εχθρούς ή ολοκληρώνει αποστολές.

Skills

Το παράθυρο δεξιοτήτων ανοίγει με το πλήκτρο "K", επιτρέποντας στον παίκτη να διαχειριστεί τις δεξιότητές (Skills) του. Κάθε δεξιότητα εμφανίζεται σε κουτάκια-κουμπιά, τα οποία ενεργοποιούνται καθώς ο παίκτης κερδίζει πόντους δεξιοτήτων (Skill Points). Για να ξεκλειδώσει μια δεξιότητα, ο παίκτης πρέπει να διαθέτει τον απαιτούμενο αριθμό πόντων, οι οποίοι αποκτώνται κάθε φορά που ανεβαίνει επίπεδο. Κάθε επίπεδο προσφέρει έναν πόντο, με τον οποίο ο παίκτης μπορεί να ξεκλειδώσει μια δεξιότητα.

Εικόνα 8. Παράθυρο Δεξιοτήτων

Οι δεξιότητες αυτές ξεκλειδώνονται με μια σειρά, ξεκινώντας από την πρώτη δεξιότητα στην κορυφή και συνεχίζοντας προς τα κάτω. Ο παίκτης πρέπει πρώτα να ξεκλειδώσει την αρχική δεξιότητα για να προχωρήσει στην επόμενη, και αυτή η διαδικασία επαναλαμβάνεται καθώς ανεβαίνει επίπεδα και συγκεντρώνει πόντους δεξιοτήτων.

Dialogue

Το παράθυρο διαλόγου εμφανίζεται όταν ο παίκτης πλησιάσει ένα NPC που μπορεί να αλληλεπιδράσει και πατήσει το πλήκτρο "E". Σε αυτό το παράθυρο εμφανίζεται η εικόνα και το όνομα του NPC, ενώ ακολουθεί ένα κείμενο διαλόγου το οποίο εμφανίζεται σταδιακά, ανά χαρακτήρα, δίνοντας μια πιο ζωντανή αίσθηση στον διάλογο. Η ροή του κειμένου επιτυγχάνεται μέσω των μεθόδων IEnumerator και WaitForSeconds(), που καθορίζουν την ταχύτητα με την οποία εμφανίζεται το κείμενο.

Σε ορισμένους διαλόγους, ο παίκτης μπορεί να επιλέξει ανάμεσα σε δύο βασικές ενέργειες: να ανοίξει ένα κατάστημα (Shop) ή να ξεκινήσει μια αποστολή (Quest). Αν ο NPC είναι έμπορος, το παράθυρο θα οδηγήσει στο κατάστημα, ενώ αν είναι χαρακτήρας που αναθέτει αποστολές, θα εμφανιστεί ο σχετικός διάλογος.

Εικόνα 9. Παράθυρο Διαλόγου με NPC

Το παράθυρο διαλόγου αποστολών έχει παρόμοια μορφή με το παράθυρο διαλόγου NPC, με τη διαφορά ότι παρουσιάζεται η εικόνα και το όνομα του εχθρού ή του αντικειμένου που πρέπει να βρει ο παίκτης.

Εικόνα 10. Παράθυρο ανταμοιβής

Quest Progress Dialogue

Στην περίπτωση που ανοίξει το παράθυρο διαλόγου αποστολών, έπειτα μπορεί να αποδεχθεί την αποστολή εμφανίζοντας έτσι διαφορετικά παράθυρα διαλόγου ανάλογα το στάδιο που βρίσκεται. Υπάρχουν δύο διαφορετικά παράθυρα ένα που εμφανίζεται κατά την διάρκεια της αποστολής, το οποίο είναι ίδιο με αυτό του απλού διαλόγου αλλά με άλλο κείμενο και ένα όταν την ολοκληρώσει, που δείχνει την ανταμοιβή.

Shops

Όταν ο παίκτης επιλέξει την εντολή Open Shop μέσα από έναν διάλογο με NPC, ανοίγει το παράθυρο καταστήματος (Shop). Σε αυτό το παράθυρο, ο παίκτης μπορεί να δει μια λίστα από αντικείμενα (Items) που διατίθενται προς πώληση. Κάθε αντικείμενο απαιτεί είτε χρήματα (Gold) είτε πρόσθετους πόρους για να αγοραστεί. Αν ο παίκτης διαθέτει τους απαραίτητους πόρους και χρήματα, μπορεί να κάνει κλικ στο αντικείμενο και αυτό προστίθεται αυτόματα στο Inventory του χαρακτήρα. Στην πραγματικότητα, δημιουργείται ένα κλώνος του αντικειμένου στο Inventory.

Εικόνα 11. Παράθυρα Μαγαζιών

Quests

Το παράθυρο των αποστολών ενεργοποιείται με το πάτημα του πλήκτρου “J”. Μέσα από αυτό το παράθυρο, ο παίκτης μπορεί να δει αναλυτικά όλες τις ενεργές αποστολές του, μαζί με τις λεπτομέρειες κάθε αποστολής. Επιπλέον, εμφανίζεται η πρόοδος που έχει σημειώσει σε κάθε αποστολή, βοηθώντας τον παίκτη να παρακολουθεί τα βήματα που του απομένουν για να ολοκληρώσει κάθε στόχο.

Εικόνα 12. Παράθυρο Αποστολών

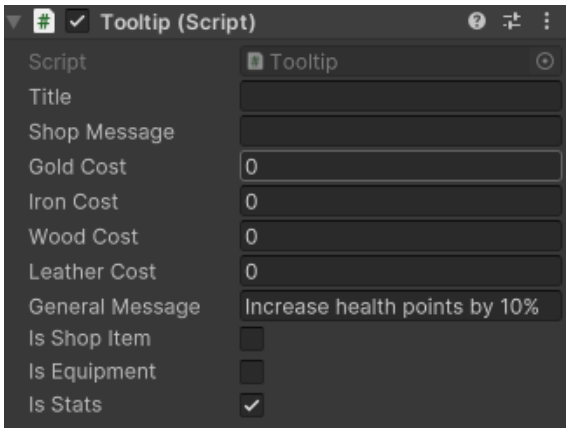
Pause Menu

Το παράθυρο του μενού παύσης εμφανίζεται με το πλήκτρο “P” και σταματάει την ροή του χρόνου μέσα στο παιχνίδι. Μπορεί να προσαρμόσει τις ρυθμίσεις του παιχνιδιού, όπως την ένταση του ήχου και την ανάλυση της οθόνης, ή να επιλέξει να αποχωρήσει. Οι επιλογές περιλαμβάνουν την επιστροφή στο κύριο μενού ή την πλήρη έξοδο από το παιχνίδι και επιστροφή στην επιφάνεια εργασίας.

Εικόνα 13. Παράθυρο μενού Παύσης

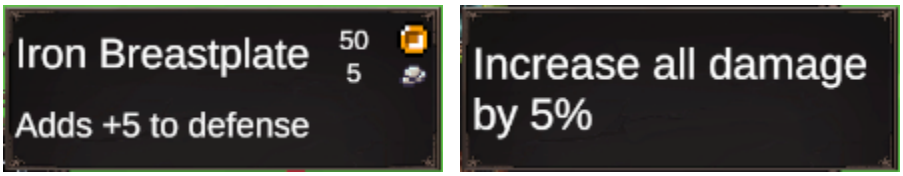
Tooltip

Το Tooltip είναι ένα παράθυρο που εμφανίζεται όταν ο παίκτης περνά το ποντίκι του πάνω από ένα αντικείμενο ή μια δεξιότητα. Αυτή η μικρή, αλλά σημαντική προσθήκη παρέχει άμεση πληροφορία για το αντικείμενο που εξετάζει ο παίκτης. Στα καταστήματα, το tooltip εμφανίζει πληροφορίες όπως το τι κάνει το αντικείμενο και πόσο κοστίζει. Παρόμοια, στις δεξιότητες εμφανίζονται πληροφορίες σχετικά με την επίδραση της κάθε δεξιότητας στον χαρακτήρα.



Εικόνα 14. Το script Tooltip όπως φαίνεται στον Inspector

Το Tooltip προσαρμόζεται δυναμικά ανάλογα με το αντικείμενο που επιλέγεται. Ένα script διαχειρίζεται την εμφάνιση του περιεχομένου, παρουσιάζοντας διαφορετικές πληροφορίες ανάλογα με το είδος του αντικειμένου που καλείται.



Εικόνα 15. Τα παράθυρα Tooltip από διαφορετικά αντικείμενα

4.2 Παίκτης

Με την έναρξη του παιχνιδιού, ο χρήστης αντικρίζει έναν χαρακτήρα που αντιπροσωπεύει τον παίκτη. Αυτός ο χαρακτήρας είναι υπό τον πλήρη έλεγχο του χρήστη καθ' όλη τη διάρκεια του παιχνιδιού και διαθέτει συγκεκριμένες ιδιότητες, οι οποίες καθορίζουν τη συμπεριφορά του. Οι βασικές λειτουργίες του παίκτη περιγράφονται στις επόμενες ενότητες.

4.2.1 Κίνηση

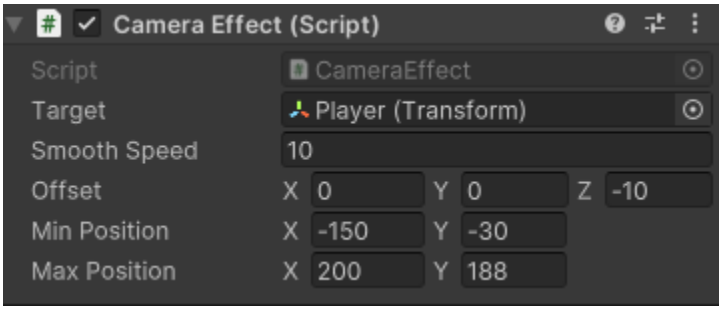
Η κίνηση του παίκτη πραγματοποιείται μέσω των πλήκτρων W, A, S, D, τα οποία επιτρέπουν τη μετακίνηση του χαρακτήρα προς τα πάνω, αριστερά, κάτω και δεξιά αντίστοιχα. Η διαδικασία αυτή ελέγχεται μέσα από τη συνάρτηση Update(), η οποία καταγράφει και αποθηκεύει τις κινήσεις του παίκτη στους άξονες x και y. Στη συνέχεια, ελέγχει την κατεύθυνση του παίκτη κατά μήκος του άξονα x, ώστε να περιστρέψει το sprite του παίκτη προς την κατάλληλη κατεύθυνση. Αυτό επιτρέπει στο sprite να αντικατοπτρίζει οπτικά τη σωστή κίνηση. Μετά από αυτό το βήμα, η συνάρτηση ενημερώνει τον animator με την τιμή της ταχύτητας κίνησης, ενεργοποιώντας το αντίστοιχο animation της κίνησης, ώστε να ξεκινήσει να παίζει.

Η διαδικασία κίνησης ολοκληρώνεται με τη χρήση της συνάρτησης FixedUpdate(), η οποία χειρίζεται τον υπολογισμό της πραγματικής κίνησης στον χώρο. Με τη μέθοδο MovePosition του RigidBody του παίκτη, προστίθεται στην τρέχουσα θέση η κατεύθυνση που αποθηκεύτηκε νωρίτερα, πολλαπλασιασμένη με την ταχύτητα του χαρακτήρα και το Time.fixedDeltaTime. Η ταχύτητα είναι μια προκαθορισμένη σταθερή τιμή που πολλαπλασιάζεται με μια μεταβλητή, η οποία εξαρτάται από το αν υπάρχει μονοπάτι στην περιοχή στην οποία κινείται ο παίκτης. Αυτή η δυναμική επιτρέπει την ομαλή κίνηση του χαρακτήρα στον χώρο, προσαρμόζοντας την ταχύτητά του ανάλογα με το περιβάλλον.

4.2.2 Κάμερα

Στη σκηνή της εφαρμογής υπάρχουν δύο κάμερες, με την κεντρική κάμερα να ακολουθεί τον παίκτη. Η κεντρική κάμερα ελέγχεται μέσω ενός script, το οποίο καθορίζει τα όρια της οπτικής γωνίας της, δηλαδή τα σημεία που μπορεί να φτάσει η κάμερα, περιορίζοντας έτσι την κίνηση στον x και y άξονα. Αυτό διασφαλίζει ότι η κάμερα παρακολουθεί τον παίκτη χωρίς να απομακρύνεται πέρα από τα καθορισμένα όρια.

Για να προσδώσουμε μια πιο φυσική και ελκυστική αίσθηση στην κίνηση της κάμερας, έχει προστεθεί και μια μικρή καθυστέρηση στην παρακολούθηση του παίκτη. Αυτή η καθυστέρηση δημιουργεί ένα εφέ που κάνει την κίνηση της κάμερας πιο ομαλή και κινηματογραφική, βελτιώνοντας την εμπειρία του χρήστη.



Εικόνα 16. Απεικόνιση του Script στον Inspector

Από την εικόνα βλέπουμε, ότι υπάρχει το Target, δηλαδή ο στόχος που θέλουμε να ακολουθεί η κάμερα, το οποίο είναι το transform του παίκτη. Το smooth speed είναι η καθυστέρηση που έχει η κάμερα όσο ακολουθεί τον παίκτη και τέλος βλέπουμε τα όρια που έχουν τεθεί.

Η δεύτερη κάμερα στη σκηνή χρησιμοποιείται για το mini map, το οποίο θα αναλυθεί λεπτομερέστερα σε επόμενη ενότητα.

4.2.3 Χρήση όπλου/εργαλείου

Κάθε φορά που ο χρήστης πατήσει αριστερό κλικ στο ποντίκι, γίνεται ένας έλεγχος για αν κρατάει κάποιο όπλο ή εργαλείο και αν ναι τότε ενημερώνεται ο animator για να παίξει το αντίστοιχο animation.

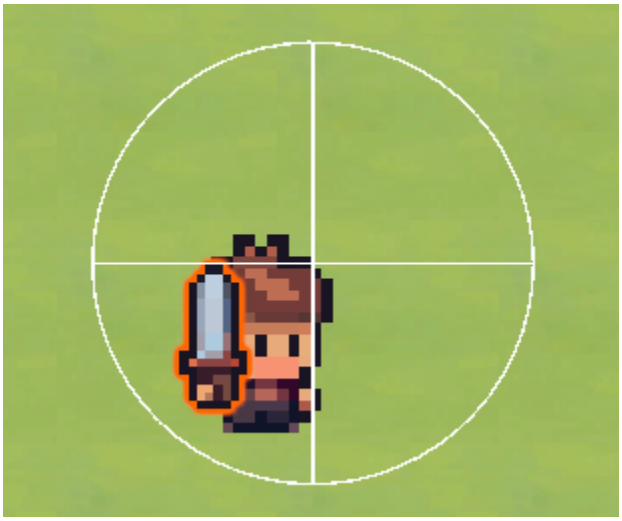
Όταν ο χρήστης πατήσει το αριστερό κουμπί του ποντικιού, πραγματοποιείται ένας έλεγχος για το αν ο παίκτης κρατάει κάποιο όπλο ή εργαλείο. Αν κρατάει, τότε ενημερώνεται ο animator ώστε να ενεργοποιηθεί το αντίστοιχο animation.

Όσον αφορά τα όπλα, όταν ξεκινά το animation της επίθεσης, σε συγκεκριμένα frames του animation, έχουν προγραμματιστεί events που καλούν διάφορες συναρτήσεις, για να εκτελέσουν κάποιες λειτουργίες, όπως φαίνεται και στην εικόνα.

Εικόνα 17. Attack Animation με Events

Για παράδειγμα, στο 0 frame, καλείται μια συνάρτηση που μειώνει την ταχύτητα του παίκτη στο μισό. Αυτή η επιβράδυνση έχει σχεδιαστεί για να προσθέσει ρεαλισμό στην κίνηση του χαρακτήρα κατά την επίθεση, κάνοντάς την να φαίνεται πιο δυναμική και αληθοφανής.

Στο frame 8, καλείται μια συνάρτηση η οποία υπολογίζει αν υπάρχουν εχθροί εντός μιας περιοχής γύρω από τον παίκτη, η οποία έχει καθοριστεί από μία εμβέλεια γύρω από ένα σημείο στον παίκτη. Η συνάρτηση εντοπίζει όλα τα colliders που βρίσκονται σε αυτή την περιοχή και αν υπάρχουν κάποια που να ανήκουν σε εχθρούς τότε, τους προκαλεί τη ζημιά (damage) που έχει προκαθοριστεί. Στην εικόνα που συνοδεύει την περιγραφή, φαίνεται το εύρος εντός του οποίου ο παίκτης μπορεί να εντοπίσει και να χτυπήσει εχθρούς.



Εικόνα 18. Το εύρος εντοπισμού

Στο frame 9, το animation ολοκληρώνεται και καλείται μια συνάρτηση που επαναφέρει την ταχύτητα του παίκτη στην κανονική της τιμή. Η ίδια διαδικασία εφαρμόζεται και για τα υπόλοιπα όπλα, με τη διαφορά ότι τα sprites του κάθε animation προσαρμόζονται ανάλογα με το αντικείμενο που χρησιμοποιεί ο παίκτης. Επιπλέον, η συνάρτηση που ενεργοποιεί την επίθεση μπορεί να καλείται σε διαφορετικό frame, ώστε να ταιριάζει καλύτερα με το ρυθμό του κάθε animation.

Όσον αφορά τα εργαλεία, η λειτουργία τους ακολουθεί παρόμοια λογική με τα όπλα, με τη διαφορά ότι αντί για τη μέθοδο επίθεσης, ενεργοποιείται η αντίστοιχη μέθοδος για τη συλλογή πόρων. Αυτή η μέθοδος ελέγχει αν το αντικείμενο που έρχεται σε επαφή με το εργαλείο είναι κάποιο από το οποίο ο παίκτης μπορεί να συλλέξει πόρους. Εάν ναι, το εργαλείο αρχίζει να προκαλεί ζημιά στο αντικείμενο έως ότου καταστραφεί. Όταν συμβεί αυτό, το αντικείμενο εξαφανίζεται και αποδίδει τους αντίστοιχους πόρους στον παίκτη.

Τα εργαλεία μπορεί να είναι διαφόρων τύπων, όπως το τσεκούρι, το οποίο χρησιμοποιείται για την κοπή δέντρων, ή η αξίνα, η οποία είναι κατάλληλη για τη θραύση πετρωμάτων. Καθένα από αυτά τα εργαλεία έχει συγκεκριμένη χρήση, καθορίζοντας την αλληλεπίδραση του παίκτη με το περιβάλλον, κάνοντας τη διαδικασία της συλλογής πόρων πιο αποτελεσματική και εξειδικευμένη.

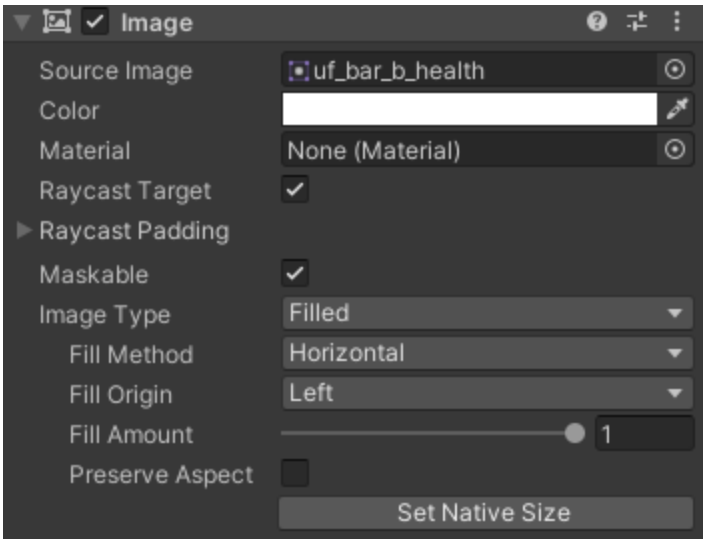
4.2.4 Ζωή και Mana

Ο παίκτης διαθέτει δύο βασικές μπάρες, μία για τη ζωή (health) και μία για τη μαγική ενέργεια (mana). Αν η ζωή του παίκτη φτάσει στο μηδέν, τότε ο χαρακτήρας «πεθαίνει». Σε αυτήν την περίπτωση, αρχικά εμφανίζεται ένα παράθυρο που ενημερώνει τον παίκτη ότι πέθανε και τον μεταφέρει αυτόματα σε μια προκαθορισμένη τοποθεσία και χάνει ένα ποσοστό από τα χρήματά του ως αντίτιμο για την επαναφορά του. Η αρχική τιμή της ζωής είναι προκαθορισμένη, αλλά μπορεί να αυξηθεί κατά τη διάρκεια του παιχνιδιού μέσω διαφόρων αναβαθμίσεων ή ενεργειών.

Εικόνα 19. Το παράθυρο που εμφανίζεται όταν ο παίκτης φτάσει 0 ζωή

Υπάρχει μια συνάρτηση που ενεργοποιείται κάθε φορά που ένας εχθρός επιτίθεται και έρχεται σε επαφή με τον παίκτη. Αυτή η συνάρτηση αφαιρεί ένα συγκεκριμένο ποσό ζωής από τον παίκτη, ανάλογα με τον εχθρό. Αντίθετα, υπάρχει και συνάρτηση που επιτρέπει την αναπλήρωση ζωής, για παράδειγμα όταν ο παίκτης πίνει ένα φίλτρο ή βρίσκεται κοντά σε κάποια φωτιά. Στο χάρτη σε συγκεκριμένα σημεία έχουν τοποθετηθεί φωτιές, οι οποίες επιτρέπουν την αργή αλλά σταθερή ανάκτηση ζωής όταν ο παίκτης βρίσκεται κοντά τους, μέχρι να φτάσει στο μέγιστο επίπεδο.

Οι δύο μπάρες ζωής και mana (κόκκινο και μπλε) παρουσιάζονται με τη μορφή οριζόντιων ράβδων που γεμίζουν και αδειάζουν ανάλογα με το τρέχον ποσό σε σχέση με το μέγιστο. Αυτές οι μπάρες είναι εικόνες που ενημερώνονται δυναμικά και έχουν ρυθμιστεί ώστε να γεμίζουν ή να αδειάζουν οριζόντια, ξεκινώντας από τα αριστερά, όπως φαίνεται από τα χαρακτηριστικά της εικόνας.



Εικόνα 20. Απεικόνιση των χαρακτηριστικών της εικόνας της ζωής

4.2.5 Επίπεδο και Stats

Στο παιχνίδι, ο παίκτης έχει τη δυνατότητα να ανέβει επίπεδο, βελτιώνοντας τις ικανότητές του. Για να το πετύχει αυτό, πρέπει να συγκεντρώσει έναν συγκεκριμένο αριθμό πόντων εμπειρίας (EXP). Μόλις επιτευχθεί ο απαραίτητος αριθμός πόντων, ο παίκτης ανεβαίνει επίπεδο, και οι τρέχοντες πόντοι EXP μηδενίζονται, ξεκινώντας από την αρχή τη διαδικασία συγκέντρωσης για το επόμενο επίπεδο. Το ανώτατο όριο επιπέδων που μπορεί να φτάσει είναι το δέκα (10), και από εκεί και πέρα, ανεξάρτητα από το πόσους πόντους εμπειρίας συγκεντρώσει, δεν μπορεί να ανέβει παραπάνω. Οι βασικοί τρόποι με τους οποίους ο παίκτης μπορεί να μαζέψει EXP είναι μέσω της εξολόθρευσης εχθρών ή της ολοκλήρωσης αποστολών (Quests). Το επίπεδο αυτό φαίνεται στο ίδιο παράθυρο που είναι και η ζωή και το Mana του παίκτη.

Ο παίκτης διαθέτει επίσης μια σειρά από χαρακτηριστικά (Stats), τα οποία είναι ορατά μέσω του παραθύρου "Character". Αυτά τα χαρακτηριστικά αντιπροσωπεύουν τις δυνατότητες του, όπως τη ζημιά που προκαλεί με κάθε τύπο όπλου, την πιθανότητα να κάνει μεγαλύτερη ζημιά με ένα χτύπημα (Critical Chance), την άμυνα, τη ζωή και το mana του. Όλα τα χαρακτηριστικά, εκτός από την άμυνα, μπορούν να βελτιωθούν από τον παίκτη κάθε φορά που ανεβαίνει επίπεδο. Η άμυνα εξαρτάται από τον εξοπλισμό που φοράει ο παίκτης και μπορεί να επηρεαστεί με την αλλαγή ή την αναβάθμισή του.

Κάθε φορά που ο παίκτης ανεβαίνει επίπεδο, κερδίζει πόντους Attribute και Skill, τους οποίους μπορεί να ξοδέψει για να ενισχύσει τις δυνάμεις του. Αυτοί οι πόντοι είναι διαθέσιμοι μέσω των παραθύρων "Character" και "Skill" αντίστοιχα. Οι πόντοι Attribute χρησιμοποιούνται για να αυξήσουν βασικά χαρακτηριστικά, όπως η ζημιά ή η αντοχή του παίκτη, ενώ οι πόντοι Skill επιτρέπουν την ανάπτυξη νέων ικανοτήτων και δεξιοτήτων.

4.2.6 Πόροι (Resources)

Καθ' όλη τη διάρκεια του παιχνιδιού, ο παίκτης μπορεί να συλλέξει διάφορους πόρους που είναι ζωτικής σημασίας για την πρόοδό του. Αυτοί οι πόροι περιλαμβάνουν χρήματα (Gold), σίδερο (Iron), ξύλο (Wood) και δέρμα (Leather). Τα χρήματα είναι ένας βασικός πόρος που ο παίκτης μπορεί να αποκτήσει είτε

εξολοθρεύοντας εχθρούς, οι οποίοι ρίχνουν ένα συγκεκριμένο ποσό χρυσού κατά τον θάνατό τους, είτε ολοκληρώνοντας αποστολές.

Το σίδερο και το ξύλο, από την άλλη, μπορούν να συλλεχθούν από ειδικά σημεία στον χάρτη. Για το σίδερο, υπάρχουν συγκεκριμένα πετρώματα που ο παίκτης πρέπει να εξορύξει χρησιμοποιώντας το κατάλληλο εργαλείο. Ομοίως, για το ξύλο, ο παίκτης πρέπει να σπάσει συγκεκριμένα δέντρα με ένα τσεκούρι. Μόλις καταστραφούν τα αντικείμενα αυτά, εξαφανίζονται και ο παίκτης συλλέγει τους πόρους. Το δέρμα όπως και τα χρήματα πέφτει από εχθρούς, σε αντίθεση όμως με τα χρήματα, το δέρμα πέφτει από συγκεκριμένους εχθρούς, όπως οι λύκοι και υπάρχει κάποια πιθανότητα να πέσει.

Εικόνα 21. Δέντρο και Πέτρωμα όπου ο παίκτης μπορεί να μαζέψει Πόρους

4.3 Animations και Animator

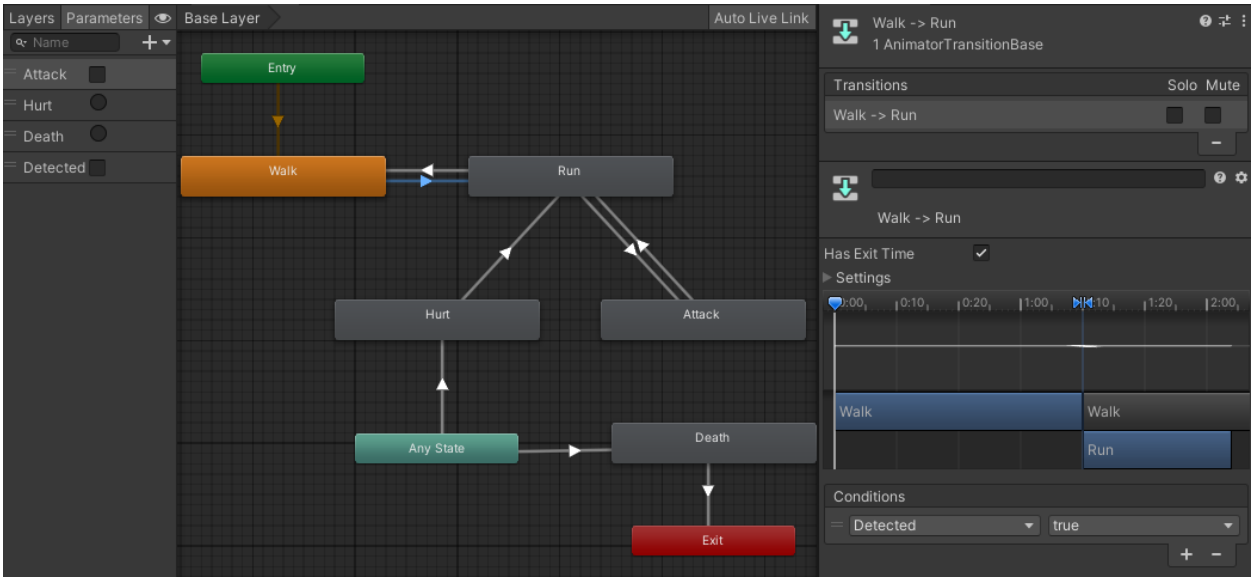
Όπως αναφέρθηκε προηγουμένως, όλα τα assets που χρησιμοποιούνται για τα animations, όπως οι χαρακτήρες του παίκτη, των εχθρών, των NPCs, καθώς και οποιοδήποτε άλλο στοιχείο με κίνηση, έχουν ληφθεί από εξωτερικές πηγές. Αυτά τα assets χρησιμοποιούνται χωρίς επιπλέον επεξεργασία ή τροποποιήσεις.

4.3.1 Animations

Για κάθε χαρακτήρα, όπως ο παίκτης, οι εχθροί, τα NPCs και τα ζώα, έχουν δημιουργηθεί τα απαραίτητα animations. Οι περισσότεροι χαρακτήρες διαθέτουν πολλαπλά animation clips, το καθένα εκ των οποίων εξυπηρετεί διαφορετικό σκοπό. Για παράδειγμα, ο παίκτης έχει ένα clip για την κατάσταση Idle, το οποίο παίζει όταν ο χαρακτήρας δεν κινείται. Υπάρχει επίσης το Walk για την κίνηση, καθώς και διάφορα Attack clips, που διαφοροποιούνται ανάλογα με το όπλο ή εργαλείο που κρατάει. Τέλος, υπάρχει ένα animation για την κατάσταση Death, που παίζει όταν ο παίκτης πεθαίνει.

4.3.2 Animator

Ο Animator είναι υπεύθυνος για τον έλεγχο των animations. Στον Animator έχουν προστεθεί όλα τα clips που σχετίζονται με κάθε χαρακτήρα ή αντικείμενο. Κάθε clip συνοδεύεται από αντίστοιχες παραμέτρους, οι οποίες ελέγχονται μέσω των scripts ώστε να εκτελεστεί η επιθυμητή ενέργεια.



Εικόνα 22. Animator με Clip Animations

Για παράδειγμα όπως βλέπουμε, ένας εχθρός έχει ως προεπιλεγμένο animation το Walk. Όταν η παράμετρος Detected γίνει true, ο Animator μεταβαίνει στην κατάσταση Run, όπου ο εχθρός αρχίζει να τρέχει. Η μετάβαση από τη μία κατάσταση στην άλλη πραγματοποιείται μέσω των «βελών» που συνδέουν τα animations και περιλαμβάνουν προϋποθέσεις για το πότε και πώς γίνεται η αλλαγή, η οποία ελέγχεται από τα scripts.

4.4 Tilemap

Όλα τα στατικά assets που εμφανίζονται στο παρασκήνιο, όπως τα δέντρα, το χορτάρι, τα μονοπάτια, το νερό, τα κτήρια και άλλα αντικείμενα του περιβάλλοντος, έχουν σχεδιαστεί μέσω του Tile Palette. Κάθε στοιχείο του περιβάλλοντος έχει τοποθετηθεί ξεχωριστά και ζωγραφιστεί σε ένα Tilemap, το οποίο βρίσκεται εντός ενός grid στη σκηνή.

Εικόνα 23. Tile Palette με assets

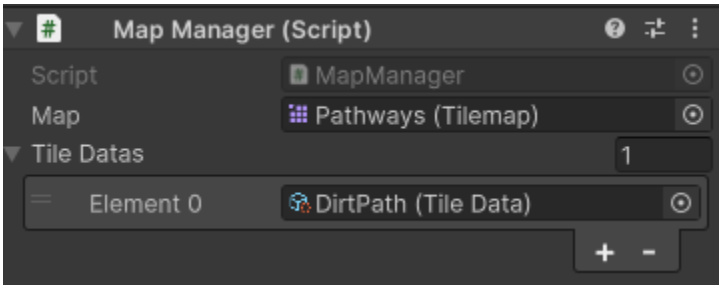
Το grid αυτό περιέχει πολλά διαφορετικά Tilemaps, καθένα εκ των οποίων εξυπηρετεί έναν συγκεκριμένο σκοπό. Η χρήση πολλών Tilemaps επιτρέπει τη δημιουργία βάθους στο περιβάλλον. Για παράδειγμα, στο Ground Tilemap έχουν σχεδιαστεί το έδαφος και το χορτάρι, ενώ πάνω σε αυτά έχουν τοποθετηθεί τα μονοπάτια και τα κτήρια.

Εικόνα 24. Το Grid με τα Tilemaps που περιέχει όπως φαίνεται στην Ιεραρχία

Εκτός από την οπτική απεικόνιση, κάποια Tilemaps έχουν επιπλέον ιδιότητες, όπως Tilemap Colliders 2D. Αυτά τα colliders εξασφαλίζουν ότι ο παίκτης δεν μπορεί να περάσει μέσα από αντικείμενα όπως το νερό, οι βράχοι ή τα δέντρα. Για παράδειγμα, τα Tilemaps Water, Ledge, Buildings, SolidObjects και Trees περιέχουν colliders που περιορίζουν την κίνηση του παίκτη, αποτρέποντάς τον από το να διασχίσει αυτά τα εμπόδια.

Ένα ιδιαίτερο Tilemap είναι το Pathways, το οποίο αναπαριστά τα μονοπάτια. Όταν ο παίκτης κινείται πάνω σε αυτά, η ταχύτητά του αυξάνεται κατά ένα προκαθορισμένο ποσοστό, ανάλογα με την τρέχουσα ταχύτητά του.

Η διαχείριση των μονοπατιών γίνεται μέσω ενός script που ονομάζεται Map Manager. Σε αυτό το script, ορίζεται το Tilemap που θεωρείται μονοπάτι (στη συγκεκριμένη περίπτωση το Pathways) και του δίνεται μια λίστα με scriptable objects. Στο συγκεκριμένο παράδειγμα, έχει δηλωθεί μόνο ένα αντικείμενο στη λίστα, αλλά θα μπορούσαν να προστεθούν περισσότερα, προσφέροντας τη δυνατότητα να διαφοροποιούνται τα μονοπάτια και να επηρεάζουν διαφορετικά την ταχύτητα του παίκτη.



Εικόνα 25. Map Manager Script

Τέλος, το Scriptable Object περιέχει μια λίστα με όλα τα assets που θεωρούνται μονοπάτια και έναν αριθμό που καθορίζει το πόσο θα πολλαπλασιαστεί η ταχύτητα του παίκτη όταν κινείται πάνω σε αυτά.

Εικόνα 26. Scriptable Object για τα δεδομένα των μονοπατιών

4.5 Αποθήκευση δεδομένων Data Persistence

Έχει αναπτυχθεί ένα σύστημα αποθήκευσης δεδομένων που επιτρέπει στον χρήστη να διατηρεί την πρόοδό του στο παιχνίδι ακόμη και μετά το κλείσιμό του. Όταν ο παίκτης επιστρέψει στο παιχνίδι, θα βρει τα δεδομένα του αποθηκευμένα, όπως το επίπεδο του, τους πόρους και αντικείμενα που έχει συλλέξει, την τοποθεσία του στον κόσμο και άλλα στοιχεία.

Η διαδικασία αυτή στηρίζεται στην κλάση GameData, στην οποία έχουν δηλωθεί όλα τα δεδομένα που πρέπει να αποθηκευτούν. Αυτά τα δεδομένα μετατρέπονται σε ένα dictionary και αποθηκεύονται σε ένα αρχείο που βρίσκεται μέσα στον φάκελο του παιχνιδιού, σε ένα προκαθορισμένο μονοπάτι και με συγκεκριμένο όνομα.

Για την ομαλή λειτουργία της αποθήκευσης και φόρτωσης των δεδομένων έχει υλοποιηθεί ένα interface IDataPersistence, το οποίο περιέχει δύο μεθόδους, LoadData() και SaveData() η οποίες είναι κενές. Το interface αυτό προστίθεται σε κάθε κλάση που έχει δεδομένα προς αποθήκευση, επομένως κληρονομεί και τις δύο προηγούμενες μεθόδους, οι οποίες κάνουν τις απαραίτητες λειτουργίες για την αποθήκευση των δεδομένων που χρειάζονται.

Έχει ακόμα υλοποιηθεί η κλάση DataPersistanceManager, στην οποία υπάρχουν οι εξής μέθοδοι:

- **LoadGame():** Αυτή η μέθοδος εκτελείται όταν ο χρήστης ανοίξει το παιχνίδι και φορτώνει τα αποθηκευμένα δεδομένα από το αρχείο στις αντίστοιχες μεταβλητές. Αν το παιχνίδι τρέχει για πρώτη φορά, δεν υπάρχουν δεδομένα προς φόρτωση, ούτε έχει δημιουργηθεί το αρχείο. Σε αυτή την περίπτωση, η μέθοδος καλεί την NewGame().
- **NewGame()**: Εκτελείται μόνο την πρώτη φορά που ο χρήστης ανοίγει το παιχνίδι. Δημιουργεί το αρχείο στο οποίο θα αποθηκευτούν τα δεδομένα του παίκτη.
- **SaveGame()**: Όταν ο χρήστης κλείσει την εφαρμογή, καλείται αυτή η μέθοδος και αποθηκεύει όλα τα δεδομένα που βρίσκονται στο dictionary στο αρχείο του παιχνιδιού, διασφαλίζοντας ότι δεν θα χαθεί η πρόοδος του που έχει γίνει μέχρι στιγμής.

Τα δεδομένα αυτά αποτελούνται από απλές μεταβλητές, λίστες, Scriptables Objects και Dictionary. Κάνοντας χρήση της κλάση FileDataHandler, όλα αυτά τα δεδομένα αποθηκεύονται και φορτώνονται σε ένα αρχείο.

Στην περίπτωση του Dictionary δεν μπορεί να αποθηκευτεί στην μορφή που είναι. Για τον λόγο αυτό καλείται πρώτα η SerializableDictionary, η οποία έχει τις μεθόδους OnBeforeSerialize() και OnAfterDeserialize() κάνοντας έτσι σειριοποίηση και αποσειριοποίηση, μετά την φόρτωση, του Dictionary.

4.6 Εχθροί

Στο παιχνίδι υπάρχει ποικιλία εχθρών. Παρά τις διαφορές τους σε επίπεδο εμφάνισης και animations, όλοι οι εχθροί μοιράζονται την ίδια βασική λογική για την κίνησή τους, τις επιθέσεις και τον θάνατό τους.

4.6.1 AI και Κίνηση

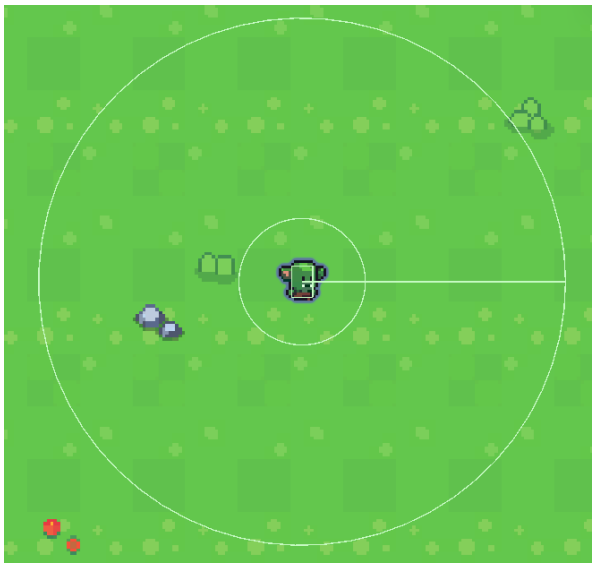
Όλοι οι εχθροί ξεκινούν τοποθετημένοι σε συγκεκριμένα σημεία μέσα στον κόσμο του παιχνιδιού. Ξεκινώντας το παιχνίδι ο κάθε εχθρός παίρνει μία τυχαία τοποθεσία μέσα στον κόσμο, που εξαρτάται από

την αρχική τοποθεσία που είχε και μια τυχαία τιμή μέσα σε ένα συγκεκριμένο εύρος που του έχει δοθεί. Έχοντας αυτή την τοποθεσία, ξεκινάει να κατευθύνεται προς αυτήν, γίνεται και ένας έλεγχος για την κατεύθυνση που έχει όσο αφορά τον άξονα y, έτσι ώστε να το περιστρέφει κάθε φορά προς την σωστή κατεύθυνση. Μόλις φτάσει σε αυτό το σημείο τότε επαναλαμβάνεται η διαδικασία και βρίσκει μια νέα τοποθεσία για να κατευθυνθεί. Ο λόγος που χρησιμοποιείται η αρχική τοποθεσία που είχε κάθε φορά, είναι για να σιγουρέψουμε ότι το μέρος που θα περιφέρεται θα είναι κοντά στην αρχική του τοποθεσία και δεν θα περιπλανηθεί πολύ μακριά.

Κατά τη διάρκεια της κίνησής τους, οι εχθροί μπορούν να συναντήσουν αντικείμενα με colliders, όπως δέντρα ή άλλα εμπόδια. Για να αποφευχθεί το ενδεχόμενο να κολλήσουν σε αυτά, πολλά από τα αντικείμενα έχουν οριστεί σε ένα ξεχωριστό layer με την ονομασία obstacles. Όταν ένας εχθρός συναντήσει ένα τέτοιο αντικείμενο, καλείται η μέθοδος OnCollisionStay2D, η οποία καλεί την μέθοδο για να βρει μια νέα τοποθεσία, αναγκάζοντας έτσι τον εχθρό να αλλάξει κατεύθυνση προς την νέα τοποθεσία. Η χρήση της OnCollisionStay αντί της OnCollisionEnter εξασφαλίζει ότι ο εχθρός θα συνεχίσει να προσπαθεί να βγει από το εμπόδιο, σε περίπτωση δηλαδή που η νέα τοποθεσία ήταν προς την ίδια κατεύθυνση, αντί να παραμείνει κολλημένος σε αυτό.

4.6.2 Εντοπισμός Παίκτη και Επίθεση

Κάθε εχθρός έχει τρία διαφορετικά colliders, δύο circle colliders ως triggers και ένα box collider. Το μεγαλύτερο circle collider, το εξωτερικό, χρησιμοποιείται για να εντοπίζει τον παίκτη. Όταν ο παίκτης εισέλθει στην εμβέλεια του εχθρού, το trigger ενεργοποιείται και ο εχθρός αρχίζει να κινείται προς το μέρος του. Για να προσφέρει καλύτερη ανατροφοδότηση στον χρήστη, όταν ο παίκτης εντοπιστεί, εμφανίζεται στιγμιαία ένα κόκκινο θαυμαστικό πάνω από τον εχθρό. Εάν ο παίκτης βγει από την εμβέλεια του εχθρού, αυτός συνεχίζει να τον κυνηγά για 4 δευτερόλεπτα. Αν κατά τη διάρκεια αυτού του διαστήματος δεν ξανά εντοπιστεί ο παίκτης, ο εχθρός επιστρέφει στην τελευταία του τυχαία τοποθεσία.



Εικόνα 27. Εχθρός με 2 circle colliders και ένα box

Το δεύτερο circle collider, το εσωτερικό, χρησιμοποιείται για την ενεργοποίηση του attack animation. Όταν ο παίκτης μπει σε αυτή την εμβέλεια, το animation ξεκινά να παίζει. Για την επίθεση χρησιμοποιείται

η ίδια λογική με αυτή του παίκτη, δηλαδή σε κάποιο συγκεκριμένο frame του animation καλείται μια μέθοδος που προκαλεί ζημιά στον παίκτη αν είναι ακόμα εντός εμβέλειας.

Το box collider εξυπηρετεί δύο σκοπούς, πρώτον, αποτρέπει τον εχθρό από το να περνάει μέσα από αντικείμενα όπως δέντρα, και δεύτερον, επιτρέπει στον παίκτη να εντοπίζει αν ο εχθρός είναι εντός της εμβέλειας του για να του προκαλέσει ζημιά.

4.6.3 Ζωή και Ανταμοιβή

Όλοι οι εχθροί, όπως και ο παίκτης, έχουν την ζωή τους. Αυτή η ζωή δεν φαίνεται κάπου που να μπορεί να την δει ο χρήστης. Όπως είδαμε προηγουμένως, ο παίκτης μέσα από το Attack animation καλεί μία μέθοδο και αν κάποιος εχθρός είναι μέσα σε μια εμβέλεια τότε του προκαλεί κάποια ζημιά. Όταν δηλαδή το box collider, που είδαμε παραπάνω, είναι μέσα στην κατάλληλη εμβέλεια κατά την ώρα της επίθεσης τότε καλείται η μέθοδος αυτή. Όταν ο εχθρός δέχεται ζημιά, το sprite του αλλάζει χρώμα σε κόκκινο στιγμιαία, δίνοντας οπτική ένδειξη στον παίκτη ότι το χτύπημα ήταν επιτυχημένο.

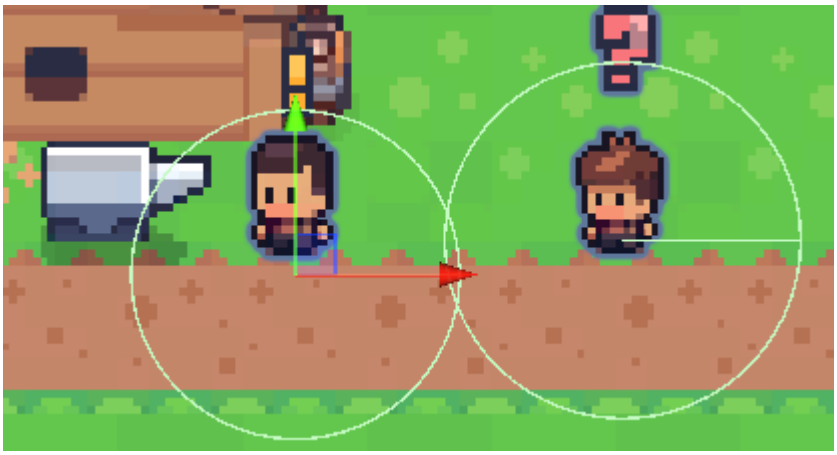
Όταν η ζωή του εχθρού φτάσει στο μηδέν, ενεργοποιείται το death animation και ο παίκτης λαμβάνει κάποιες ανταμοιβές. Αρχικά, ο παίκτης κερδίζει πόντους εμπειρίας (EXP), ακολουθεί ένα χρηματικό ποσό (gold), και τέλος, ορισμένοι εχθροί έχουν πιθανότητα να ρίξουν αντικείμενα, όπως για παράδειγμα, οι λύκοι που μπορεί να ρίξουν δέρμα. Κάθε εχθρός έχει προκαθορισμένα αντικείμενα που μπορεί να ρίξει, μαζί με τις πιθανότητες για το καθένα. Τέλος, όταν ο εχθρός πεθαίνει, το game object του καταστρέφεται.

4.7 Non Playable Characters

Εκτός από τους εχθρούς, το παιχνίδι περιλαμβάνει και άλλους χαρακτήρες, με κάποιους από τους οποίους ο παίκτης μπορεί να αλληλεπιδράσει.

4.7.1 Interactable NPCs

Σε διάφορα σημεία του κόσμου υπάρχουν NPCs με τα οποία ο παίκτης μπορεί να αλληλεπιδράσει. Αυτά τα NPCs διαθέτουν circle colliders που λειτουργούν ως triggers. Όταν ο παίκτης βρεθεί εντός της εμβέλειάς τους και πατήσει το κουμπί “E”, ξεκινάει ένας διάλογος. Έπειτα ανάλογα το NPC ανοίγει ένα παράθυρο για αγορές (shop) ή αποστολές (quests). Τα NPCs αυτά βρίσκονται σε συγκεκριμένες θέσεις χωρίς να μετακινούνται και παίζουν ένα idle animation. Ακόμα αυτά που ανοίγουν μαγαζιά έχουν ένα θαυμαστικό πάνω από το κεφάλι τους, ενώ αυτά που δίνουν αποστολές έχουν ένα ερωτηματικό.



Εικόνα 28. Interactable NPCs με τα trigger colliders

4.7.2 NPCs που περιφέρονται σε μονοπάτι

Με ορισμένα NPCs δεν μπορεί να αλληλεπιδράσει ο παίκτης, αλλά κινούνται κατά μήκος προκαθορισμένων μονοπατιών. Αυτά τα μονοπάτια έχουν δημιουργηθεί με τη χρήση έτοιμου asset από το Unity Asset Store. Το μονοπάτι αποτελείται από μια ευθεία γραμμή με αρχή και τέλος. Σε αυτό μπορούν να προστεθούν επιπλέον σημεία για να του δοθεί οποιαδήποτε κλίση ή σχήμα.

Τα NPCs αυτά κινούνται συνεχώς, παίζοντας ένα συγκεκριμένο animation, για παράδειγμα walking. Έχουν ένα script που τους κατευθύνει και επιτρέπει τη ρύθμιση της ταχύτητάς τους και της λογικής που θα ακολουθήσουν όταν φτάσουν στο τέλος του μονοπατιού, η οποία μπορεί να είναι μία από τις καταστάσεις Loop, Reverse και Stop.

Εικόνα 29. Τα μονοπάτια ως πράσινες γραμμές που ακολουθούν τα NPC

4.7.3 Σταθερά NPCs με Animations

Υπάρχουν επίσης NPCs που παραμένουν σταθερά σε συγκεκριμένες τοποθεσίες, εκτελώντας συνεχώς ένα συγκεκριμένο animation. Ούτε με αυτά μπορεί να αλληλεπιδράσει ο παίκτης. Αυτοί οι χαρακτήρες ενισχύουν την ατμόσφαιρα και την αίσθηση ζωντάνιας του παιχνιδιού. Για παράδειγμα όπως φαίνεται παρακάτω υπάρχει ένα που σκάβει το χώμα και ένα που ποτίζει φυτά.

Εικόνα 30. Σταθερά NPCs με animations

4.7.4 Ζώα

Τέλος όσο αφορά τα NPCs, υπάρχουν και τα ζώα που είναι ένα ακόμη στοιχείο που εμπλουτίζει τον κόσμο του παιχνιδιού. Αυτά τα πλάσματα περιφέρονται τυχαία, χωρίς τη δυνατότητα αλληλεπίδρασης με τον παίκτη. Η κίνησή τους είναι παρόμοια με εκείνη των εχθρών, καθώς επιλέγουν τυχαίες τοποθεσίες για να κινηθούν προς αυτές. Αν συναντήσουν κάποιο εμπόδιο στη διαδρομή τους, τότε επιλέγουν μια νέα τοποθεσία για να συνεχίσουν την περιπλάνησή τους. Η παρουσία των ζώων στο παιχνίδι δεν εξυπηρετεί μόνο διακοσμητικούς σκοπούς, αλλά ενισχύει την αίσθηση ότι ο κόσμος είναι ζωντανός και γεμάτος κίνηση, κάνοντάς τον πιο πλούσιο για τον παίκτη.

Εικόνα 31. Ζώα που περιφέρονται με AI

4.8 Μουσική και Ηχητικά Εφέ

Όσον αφορά τη μουσική και τα ηχητικά εφέ (SFX) που έχουν ενσωματωθεί στο παιχνίδι, προέρχονται από εξωτερικές πηγές και χρησιμοποιούνται όπως είναι, χωρίς περαιτέρω επεξεργασία ή τροποποίηση. Η ακουστική εμπειρία του παιχνιδιού βασίζεται στο Audio Listener, που βρίσκεται ενσωματωμένος στην κεντρική κάμερα. Ο Audio Listener διασφαλίζει ότι όλοι οι ήχοι είναι σωστά αντιληπτοί από τον παίκτη.

Η μουσική είναι διαχωρισμένη σε δύο διαφορετικά κομμάτια: ένα που παίζει στο κεντρικό μενού και ένα που ακούγεται κατά τη διάρκεια του παιχνιδιού. Οι παίκτες έχουν τη δυνατότητα να προσαρμόσουν την ένταση τόσο της μουσικής όσο και των ηχητικών εφέ μέσω δύο ξεχωριστών sliders στις ρυθμίσεις του παιχνιδιού. Αν επιθυμούν, μπορούν να απενεργοποιήσουν τελείως τον ήχο με ένα απλό πάτημα του αντίστοιχου κουμπιού.

Για τα ηχητικά εφέ, έχει δημιουργηθεί ένας Audio Manager. Στον Audio Manager έχει καταχωριστεί μια λίστα με όλους τους ήχους του παιχνιδιού. Κάθε ήχος μπορεί να ενεργοποιηθεί ξεχωριστά από τα scripts, ανάλογα με τις απαιτήσεις του παιχνιδιού. Για παράδειγμα, όταν ο παίκτης χρησιμοποιεί το σπαθί του, παίζεται ο ήχος SwordSwing που αντιπροσωπεύει το χτύπημα.

Ορισμένοι ήχοι, όπως ο ήχος των βημάτων του παίκτη, δεν ακολουθούν αυτό το μοτίβο. Αυτοί οι ήχοι είναι συνδεδεμένοι απευθείας με το αντικείμενο από όπου προέρχεται ο ήχος. Για παράδειγμα με τον ήχο των βημάτων, βρίσκονται στον ίδιο τον χαρακτήρα του παίκτη και ενεργοποιούνται όταν ο παίκτης κινείται, δηλαδή όταν πατάει κάποιο από τα πλήκτρα W, A, S, D.

4.9 Mini Map

Στο παιχνίδι έχει ενσωματωθεί ένα mini map, δηλαδή ένας μικρός χάρτης που βρίσκεται στην επάνω δεξιά γωνία της οθόνης και απεικονίζει από ψηλά την περιοχή του κόσμου γύρω από τον παίκτη. Αυτό έχει

επιτευχθεί με τη χρήση μιας δεύτερης κάμερας, η οποία τοποθετείται ψηλά και ακολουθεί τον παίκτη, προσφέροντας μια πανοραμική θέα του κόσμου.

Για την απεικόνιση του χάρτη, χρησιμοποιείται ένα render texture, πάνω στο οποίο προβάλλει η δεύτερη κάμερα. Το render texture τοποθετείται στην οθόνη στο επιθυμητό σημείο και γύρω του υπάρχει ένα πλαίσιο με ειδικό φίλτρο, το οποίο ταιριάζει με την αισθητική του υπόλοιπου παιχνιδιού.

Δεδομένου ότι ο χάρτης είναι μικρός και δείχνει τον κόσμο από ψηλά, δεν είναι απαραίτητο να εμφανίζονται όλα τα αντικείμενα, όπως τα διακοσμητικά στοιχεία (π.χ. λουλούδια), καθώς κάτι τέτοιο θα έκανε δυσδιάκριτα τα σημαντικά σημεία. Γι’ αυτόν τον λόγο, όποια αντικείμενα θέλουμε να παραλείψουμε και να μην φαίνονται, έχουν τοποθετηθεί σε ένα ειδικό layer, το οποίο ονομάζεται MinimapIgnore. Η κάθε κάμερα έχει την δυνατότητα επιλογής των layer που θα προβάλει, με αυτόν τον τρόπο μπορούμε να επιλέξουμε να μην δείχνει το συγκεκριμένο, έτσι η κάμερα του mini map παραβλέπει τα αυτά αντικείμενα.

Αντίθετα, σημαντικά αντικείμενα όπως εχθροί ή NPCs με τα οποία ο παίκτης αλληλεπιδρά, πρέπει να εμφανίζονται στον χάρτη με μεγαλύτερη σαφήνεια. Γι' αυτόν τον λόγο, έχουν προστεθεί εικόνες πάνω σε αυτά τα αντικείμενα, ώστε να φαίνονται ευκρινώς στον mini map. Αυτές οι εικόνες όμως δεν θέλουμε να φαίνονται από την κεντρική κάμερα, για τον λόγο αυτό έχουν τοποθετηθεί σε ένα ξεχωριστό layer που ονομάζεται Minimap, το οποίο είναι ορατό μόνο από την κάμερα του mini map και όχι από την κεντρική κάμερα του παιχνιδιού.

Στην παρακάτω εικόνα βλέπουμε έναν μπλε κύκλο που δείχνει τον παίκτη, με κόκκινο κύκλο τους εχθρούς και βλέπουμε και άλλες εικόνες όπως ένα σφυρί, μία καρδιά και ένα ερωτηματικό που αντιπροσωπεύουν τον blacksmith, από όπου μπορεί να αγοράσει εξοπλισμό, την φωτιά και ένα NPC που δίνει quest αντίστοιχα.

Εικόνα 32. Απεικόνιση του Mini Map με σύμβολα για τα σημαντικά αντικείμενα

4.10 Αποστολές (Quests)

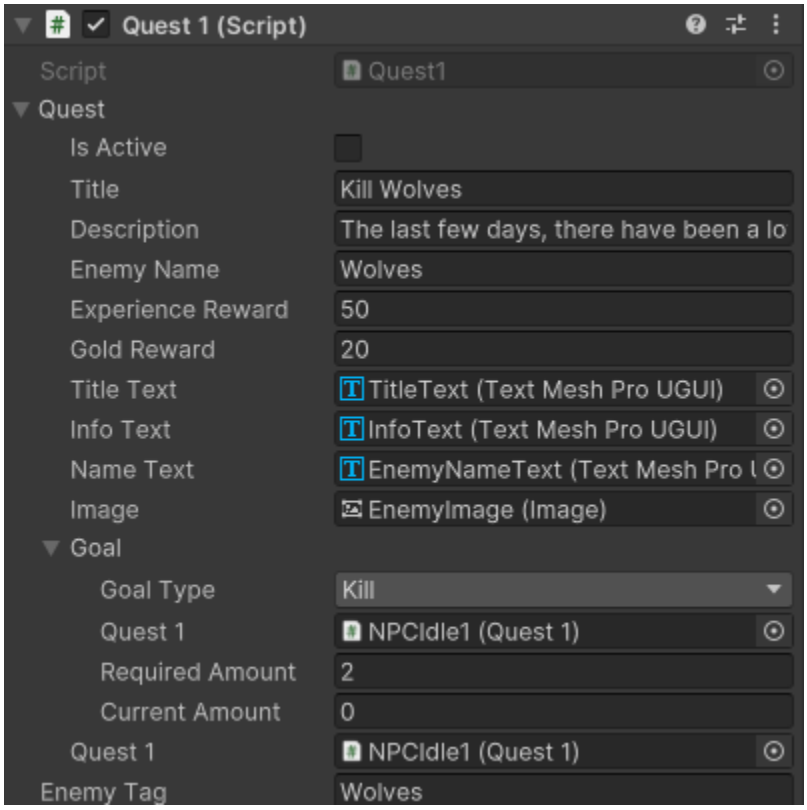
Κατά τη διάρκεια του παιχνιδιού, ο παίκτης μπορεί να συναντήσει διάφορους NPCs που του προσφέρουν αποστολές (quests). Αυτές οι αποστολές δίνουν στον παίκτη την ευκαιρία να αναλάβει διάφορες προκλήσεις, οι οποίες όταν ολοκληρωθούν επιτυχώς, προσφέρουν ανταμοιβές με τη μορφή εμπειρίας (EXP) και χρημάτων (gold).

Υπάρχουν δύο βασικά είδη αποστολών στο παιχνίδι, καθεμία με ξεχωριστούς στόχους και απαιτήσεις. Το πρώτο είδος αποστολής έχει ως κύριο στόχο την εξολόθρευση ενός συγκεκριμένου αριθμού εχθρών. Το δεύτερο είδος αποστολής επικεντρώνεται στη συλλογή ενός καθορισμένου αριθμού αντικειμένων, τα οποία μπορεί να βρει ο παίκτης είτε μέσα από την εξερεύνηση του κόσμου είτε από τα λάφυρα που αφήνουν πίσω τους οι εχθροί.

Σε κάθε αποστολή, τόσο ο στόχος όσο και ο απαιτούμενος αριθμός εχθρών ή αντικειμένων διαφέρουν, ανάλογα με το NPC που την προσφέρει. Αυτές οι πληροφορίες είναι προκαθορισμένες και ποικίλλουν, κάνοντας κάθε αποστολή μοναδική και προσαρμοσμένη στον χαρακτήρα και την πρόκληση που προσφέρει το εκάστοτε NPC. Η ποικιλία αυτή διασφαλίζει ότι οι αποστολές παραμένουν ενδιαφέρουσες, ενώ προσφέρουν διαφορετικά επίπεδα δυσκολίας, διατηρώντας το ενδιαφέρον του παίκτη σε κάθε νέα πρόκληση που καλείται να αντιμετωπίσει.

Στην παρακάτω εικόνα, βλέπουμε το script της αποστολής εξολόθρευσης όπως εμφανίζεται στον Inspector. Παρατηρούμε ότι υπάρχουν πεδία για τον τίτλο και την περιγραφή της αποστολής, τα οποία εμφανίζονται στον παίκτη όταν ξεκινά ο διάλογος με το NPC. Επιπλέον, υπάρχει ένα πεδίο για την εικόνα και το όνομα του εχθρού που πρέπει να αντιμετωπιστεί.

Στη συνέχεια, φαίνονται οι ανταμοιβές που προσφέρονται κατά την ολοκλήρωση της αποστολής, οι οποίες περιλαμβάνουν EXP και gold. Τέλος, στο πεδίο "Goal" καθορίζεται ο στόχος της αποστολής. Αυτός μπορεί να είναι είτε η εξόντωση ενός προκαθορισμένου αριθμού εχθρών, είτε η συλλογή συγκεκριμένων αντικειμένων, όπως φρούτα ή υλικά (gather).



Εικόνα 33. Το script Quest1 όπως φαίνεται με τα πεδία του

Κάθε NPC που προσφέρει αποστολές έχει συνδεδεμένο αυτό το script, και οι τιμές των πεδίων τροποποιούνται ανάλογα με την αποστολή που προσφέρει κάθε φορά, προσφέροντας ποικιλία και διαφορετικές προκλήσεις στον παίκτη.

Κάθε αποστολή (Quest) αποτελείται από τρία διακριτά στάδια, τα οποία διαμορφώνουν την αλληλεπίδραση του παίκτη με το NPC που την προσφέρει. Στο πρώτο στάδιο, όταν ο παίκτης προσεγγίζει το NPC και ξεκινά τη συνομιλία, εμφανίζεται το αρχικό παράθυρο διαλόγου. Από εκεί, ο παίκτης μπορεί να προχωρήσει στην οθόνη με τις λεπτομέρειες της αποστολής. Μόλις αποδεχθεί την αποστολή, περνάει στο δεύτερο στάδιο, όπου η αποστολή θεωρείται ενεργή. Εάν ο παίκτης μιλήσει ξανά με το NPC κατά τη διάρκεια της αποστολής, θα εμφανιστεί ένα νέο παράθυρο διαλόγου, το οποίο του υπενθυμίζει ότι η αποστολή βρίσκεται σε εξέλιξη και περιμένει να ολοκληρωθεί.

Όταν ο παίκτης ολοκληρώσει την αποστολή, περνάει στο τρίτο στάδιο. Επιστρέφοντας στο NPC, εμφανίζεται ένα παράθυρο διαλόγου με την ανταμοιβή του παίκτη για την επιτυχία του. Μόλις κλείσει αυτό το παράθυρο, το script του NPC, το οποίο περιέχει όλες τις λειτουργίες και αλληλεπιδράσεις με τον παίκτη, απενεργοποιείται, αποτρέποντας τον παίκτη από το να ξανά λάβει την ίδια αποστολή. Το NPC παραμένει στον κόσμο, απλώς δεν μπορεί πλέον να αλληλεπιδράσει μαζί του ο παίκτης.

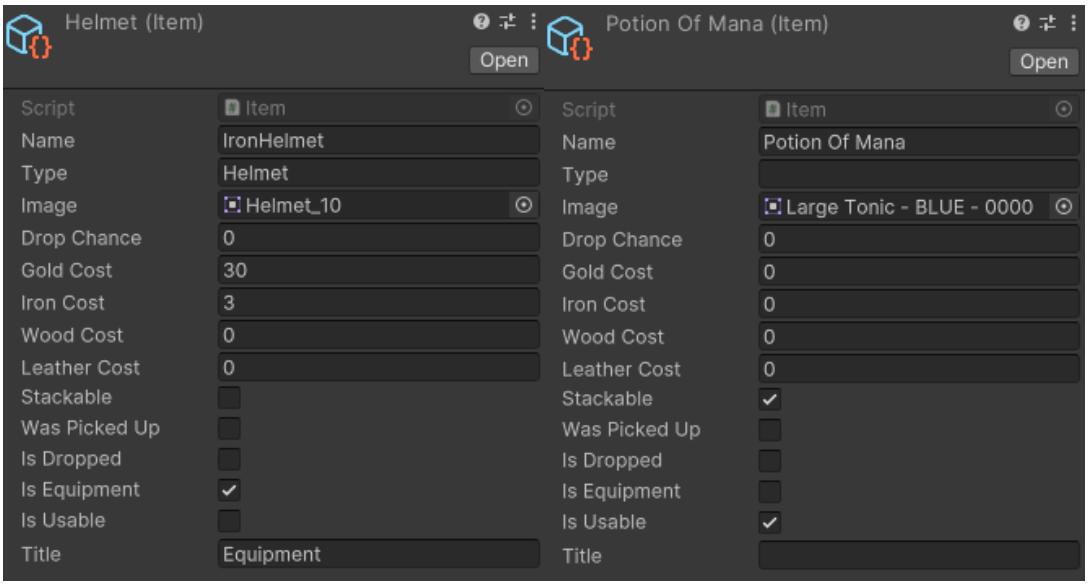
4.11 Αντικείμενα (Items)

Κατά την διάρκεια του παιχνιδιού ο παίκτης μπορεί να βρει διάφορα αντικείμενα (Items). Όλα τα αντικείμενα είναι υλοποιημένα ως Scriptable Objects και κατηγοριοποιούνται σε τέσσερις βασικές ομάδες, οι οποίες διαφοροποιούνται βάσει μιας τιμής τύπου bool.

- Εξοπλισμός (Equipment)
- Αντικείμενα για Χρήση (Usable Items)
- Αντικείμενα που πέφτουν από εχθρούς (Loot)
- Πόροι (Resources) και Αντικείμενα που Συλλέγει (Gatherable)

Κάθε Scriptable Object αντικειμένου περιέχει διάφορα πεδία δεδομένων, αλλά δεν χρειάζεται να συμπληρωθούν όλα για κάθε τύπο. Ανάλογα με το είδος του αντικειμένου, διαφορετικά πεδία.

- **Όνομα (Name)** και **Εικόνα (Image)**: Υποχρεωτικά πεδία για όλα τα αντικείμενα.
- **Τύπος (Type)**: Απαιτείται μόνο για τον εξοπλισμό, καθώς χρησιμοποιείται στο παράθυρο "Character" όταν ο παίκτης προσπαθεί να τοποθετήσει κάποιο αντικείμενο στον σωστό εξοπλισμό (slots).
- **Drop Chance**: Χρησιμοποιείται μόνο για αντικείμενα που πέφτουν από εχθρούς, καθορίζοντας την πιθανότητα να βρεθεί το αντικείμενο όταν ο εχθρός ηττηθεί.
- **Κόστος Πόρων**: Σχετίζεται με αντικείμενα που μπορούν να αγοραστούν από μαγαζιά, καθορίζοντας τους πόρους που απαιτούνται για την αγορά τους.
- **Stackable**: Καθορίζει αν τα αντικείμενα μπορούν να στοιβάζονται σε ένα κουτάκι στο Inventory ή στο Toolbar, επιτρέποντας την αποθήκευση πολλαπλών μονάδων του ίδιου αντικειμένου.
- **Τίτλος**: Χρησιμοποιείται μόνο για εξοπλισμό, εμφανίζοντας χρήσιμες πληροφορίες όταν ο παίκτης τοποθετεί τον δείκτη του πάνω στο αντικείμενο.



Εικόνα 34. Διαφορετικά Scriptable Objects όπως φαίνονται στον Inspector

Equipment

Τα αντικείμενα που ανήκουν στην κατηγορία του εξοπλισμού μπορούν να τοποθετηθούν στα κουτάκια εξοπλισμού (slots) του παραθύρου "Character". Όταν ο παίκτης φοράει ένα τέτοιο αντικείμενο, ελέγχεται μέσω κώδικα τι είδους αντικείμενο είναι και ενισχύει τα αντίστοιχα χαρακτηριστικά (Stats) του παίκτη. Αν ο εξοπλισμός αφαιρεθεί ή αντικατασταθεί με κάποιο άλλο αντικείμενο, οι τιμές στα Stats προσαρμόζονται αναλόγως.

Usable Items

Τα αντικείμενα για χρήση είναι αντικείμενα που ο παίκτης μπορεί να ενεργοποιήσει για να αποκτήσει κάποια άμεση ιδιότητα ή όφελος. Τα Usable Items μπορεί να είναι φίλτρα (Potions), τα οποία ο παίκτης μπορεί να αγοράσει από μαγαζιά ή Gatherable που μπορεί να βρει στον κόσμο και να τα μαζέψει. Οι κύριες κατηγορίες φίλτρων που είναι διαθέσιμες περιλαμβάνουν:

- **Φίλτρο Ανάκτησης Ζωής**: Επαναφέρει την ζωή του παίκτη (HP).
- **Φίλτρο Ανάκτησης Mana**: Επαναφέρει το mana του παίκτη, επιτρέποντάς του να χρησιμοποιεί ξανά μαγικές ικανότητες.
- **Φίλτρο Αποτροπής Ζημιάς**: Προστατεύει τον παίκτη από τη λήψη ζημιάς για ένα συγκεκριμένο χρονικό διάστημα.
- **Φίλτρο Αύξησης Ταχύτητας**: Αυξάνει την ταχύτητα κίνησης του παίκτη για ένα ορισμένο χρονικό διάστημα.

Στην παρούσα υλοποίηση το μόνο Gatherable που υπάρχει είναι το μήλο, το οποίο αν χρησιμοποιήσει του ανεβάζει ένα μικρό ποσοστό ζωής.

Loot

Τα Loot Items είναι αντικείμενα που πέφτουν από εχθρούς όταν αυτοί νικηθούν. Το πεδίο Drop Chance καθορίζει την πιθανότητα να πέσει το αντικείμενο, δίνοντας τυχαία χαρακτηριστικά στη συλλογή πόρων

και ενισχύοντας την αίσθηση ανταμοιβής για την εξολόθρευση εχθρών. Αυτά τα αντικείμενα είναι συνήθως πόροι ή ειδικά αντικείμενα που μπορούν να χρησιμοποιηθούν για αγορά εξοπλισμού όπως το δέρμα, ή άλλες διαδικασίες εντός του παιχνιδιού.

Resources και Gatherable

Τα resources είναι υλικά όπως το ξύλο και το σίδερο, τα οποία ο παίκτης μπορεί να συλλέξει από διάφορα αντικείμενα του περιβάλλοντος, όπως δέντρα και πετρώματα. Χρησιμοποιώντας τα κατάλληλα εργαλεία, ο παίκτης μπορεί να σπάσει αυτά τα αντικείμενα και να αποκτήσει τα αντίστοιχα resources.

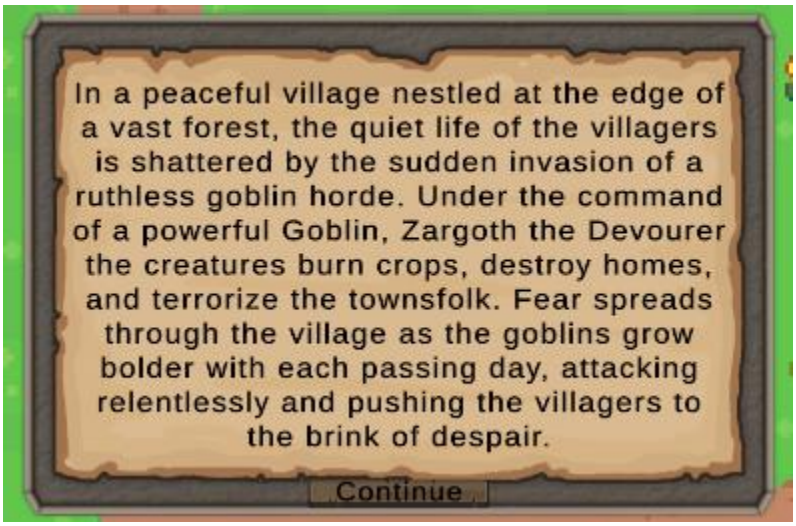
Τα gatherables, από την άλλη, είναι αντικείμενα που βρίσκονται διάσπαρτα στον κόσμο του παιχνιδιού, όπως τα μήλα και μπορούν να μαζευτούν απευθείας από τον παίκτη χωρίς τη χρήση εργαλείων.

4.12 Σκοπός του παιχνιδιού και Ιστορία

Μόλις ξεκινήσει το παιχνίδι, εμφανίζεται στον χρήστη ένα παράθυρο με κείμενο που αφηγείται μια ιστορία. Η ιστορία αυτή αποκαλύπτει ότι το χωριό, στο οποίο βρίσκεται, απειλείται από μια σειρά επιθέσεων από εχθρικά Goblins και ότι οι χωρικοί χρειάζονται την βοήθεια του παίκτη, καθώς οι επιθέσεις εντείνονται. Ο στόχος του είναι να προστατεύσει το χωριό και να εξουδετερώσει τον αρχηγό των Goblins.

Η αφήγηση αυτή δεν είναι απλώς ένα σκηνικό, δίνει νόημα και σκοπό στον παίκτη, ωθώντας τον να αναπτύξει τις δυνάμεις του, να δυναμώσει και να γίνει ικανός να αντιμετωπίσει τους εχθρούς που τον περιμένουν.

Όταν τελικά ο παίκτης καταφέρει να νικήσει τον αρχηγό, ένα νέο παράθυρο εμφανίζεται, παρόμοιο με το αρχικό, που είχε την ιστορία. Αυτό το παράθυρο τον ενημερώνει ότι ολοκλήρωσε την αποστολή του και έσωσε το χωριό. Αν και ο παίκτης μπορεί να συνεχίσει να εξερευνά και να παίζει αν το επιθυμεί, η κύρια ιστορία ολοκληρώνεται σε αυτό το σημείο, σηματοδοτώντας την επιτυχία του.



Εικόνα 35. Παράθυρο με το κείμενο της Ιστορίας

Κεφάλαιο 5

5. Συμπεράσματα και Επεκτάσεις Εφαρμογής

Η παρούσα εργασία εστίασε στην ανάλυση και υλοποίηση ενός συστήματος για την ανάπτυξη ενός 2D παιχνιδιού με τη χρήση της Unity. Η εργασία ξεκίνησε με την ανασκόπηση των βασικών εργαλείων και των τεχνικών ανάπτυξης που χρησιμοποιούνται ευρέως στη βιομηχανία παιχνιδιών, προσφέροντας μία ολοκληρωμένη εικόνα της αρχιτεκτονικής του συστήματος.

Συγκεκριμένα, έγινε ανάλυση των δυνατοτήτων του Unity, όπως το περιβάλλον εργασίας του (UI), η χρήση των Panels (Inspector, Game, Scene, Project, κ.ά.), καθώς και η διαχείριση των GameObjects και των Components. Οι μέθοδοι της κλάσης MonoBehaviour, τα Scriptable Objects και η συνεργασία του Unity με το Visual Studio και τη γλώσσα προγραμματισμού C# αποτελούν βασικούς πυλώνες της υλοποίησης του παιχνιδιού. Επιπλέον, δόθηκε έμφαση στην προσομοίωση της φυσικής, στη χρήση animations, στα sprites και στη διαχείριση του UI, τα οποία συνέβαλαν στη δημιουργία μίας πλούσιας και λειτουργικής εμπειρίας για τον χρήστη.

Μέσα από τη διαδικασία της υλοποίησης, δομήθηκε ένα λειτουργικό σύστημα που επιτρέπει στον παίκτη να αλληλεπιδρά με το περιβάλλον και τους χαρακτήρες του παιχνιδιού, να πραγματοποιεί αποστολές και να προχωρά την ιστορία. Το σύστημα περιλαμβάνει επίσης δυναμική αλληλεπίδραση με NPCs, αποθήκευση δεδομένων, και διαχείριση πόρων.

Συνολικά, η εργασία αυτή δείχνει πως η χρήση του Unity σε συνδυασμό με τη γλώσσα προγραμματισμού C# παρέχει μία ισχυρή πλατφόρμα για την ανάπτυξη 2D παιχνιδιών, δίνοντας έμφαση τόσο στη λειτουργικότητα όσο και στη δημιουργική έκφραση του σχεδιαστή παιχνιδιών.

5.1 Συμπεράσματα και Επεκτασιμότητα

Από την ανάλυση προκύπτει πως το σύστημα που υλοποιήθηκε είναι πλήρως λειτουργικό και προσφέρει ένα δυναμικό περιβάλλον για τη δημιουργία 2D παιχνιδιών. Η δουλειά αυτή όχι μόνο ικανοποιεί τις απαιτήσεις της εργασίας αλλά παρέχει και τη βάση για περαιτέρω επεκτάσεις.

Το παιχνίδι που υλοποιήθηκε περιλαμβάνει όλα τα βασικά συστήματα που απαιτούνται για τη λειτουργία του, αλλά υπάρχουν περιθώρια για περαιτέρω ανάπτυξη και βελτιστοποίηση. Οι βελτιώσεις αυτές θα μπορούσαν να ενισχύσουν την εμπειρία του χρήστη, κάνοντάς την πιο πλούσια και διαδραστική. Για παράδειγμα, η προσθήκη διαφορετικών τύπων όπλων και εξοπλισμού θα έδινε στον παίκτη τη δυνατότητα να επιλέξει το προσωπικό του στυλ παιχνιδιού, είτε για παράδειγμα επιλέγοντας μαγεία είτε επιθέσεις από απόσταση με τόξο. Επίσης, η ενσωμάτωση περισσότερων μηχανισμών αλληλεπίδρασης με το περιβάλλον, όπως διάσπαρτα κουτιά θησαυρών (treasure chest) ή η δυνατότητα ψαρέματος, θα παρότρυνε τον παίκτη να εξερευνήσει τον κόσμο για να ανακαλύψει πολύτιμα αντικείμενα και ενδιαφέροντα σημεία.

Ο κόσμος του παιχνιδιού θα μπορούσε επίσης να επεκταθεί, προσθέτοντας νέα επίπεδα με διαφορετικά περιβάλλοντα και εχθρούς, προσφέροντας έτσι μεγαλύτερη ποικιλία στην εξερεύνηση και την πρόκληση. Η ιστορία του παιχνιδιού θα μπορούσε να γίνει πιο ζωντανή και συναρπαστική με την ενσωμάτωση cut scenes και σημαντικών NPCs, τα οποία θα προωθούσαν την αφήγηση μέσω αποστολών.

Σε τεχνικό επίπεδο, κάποια συστήματα όπως το σύστημα των αποστολών (Quests) θα μπορούσε να βελτιωθεί ώστε να διευκολύνει την επεκτασιμότητα. Με την τρέχουσα υλοποίηση, η δυναμική προσθήκη νέων αποστολών είναι περίπλοκη. Μια καλύτερη δομή κώδικα θα μπορούσε να επιτρέψει ευκολότερη και πιο ευέλικτη επέκταση του συστήματος αποστολών, κάνοντάς το πιο προσαρμόσιμο.

Παράρτημα: Κώδικας Υλοποίησης

Η εφαρμογή με τον κώδικα και όλα τα assets που χρησιμοποιήθηκαν έχουν αναρτηθεί στην πλατφόρμα GitHub. Το Project μπορεί να βρεθεί εδώ.

Βιβλιογραφία

1. Adam, B. (2019). Learn Unity Programming with C#. Apress.
2. Barrera, J. (2020). Hands-On Unity 2020 Game Development: Build, customize, and optimize professional games using Unity 2020 and C#. Packt Publishing.
3. Breslin, J. (2018). Mastering Unity 2D Game Development: Build platformers, adventure games, and RPGs with C# and Unity. Packt Publishing.
4. Hocking, J. (2015). Unity in Action: Multiplatform Game Development in C# with Unity 5. Manning Publications.
5. Ritchie, J. (2020). Unity from Zero to Proficiency (Foundations): A Step-by-step Guide to Creating Your First Game. Patrick Felicia Publishing.
6. Santos, F. (2017). Mastering Unity 2017 Game Development with C#. Packt Publishing.
7. Unity Technologies. (2023). Unity User Manual (2023.2). Unity Technologies. Ανακτήθηκε από: https://docs.unity3d.com/Manual/index.html
8. Microsoft. (2023). Visual Studio Documentation. Microsoft. Ανακτήθηκε από: https://learn.microsoft.com/en-us/visualstudio/?view=vs-2023
9. Unity Technologies. (2024). Unity Asset Store: Create & Sell Assets. Ανακτήθηκε από: https://assetstore.unity.com/
10. Unity Learn. (n.d.). Creating a 2D RPG in Unity. Ανακτήθηκε από: https://learn.unity.com/course/creating-a-2d-rpg
11. Troussas C, Krouska A, Sgouropoulou C. Enriching Mobile Learning Software with Interactive Activities and Motivational Feedback for Advancing Users' High-Level Cognitive Skills. Computers. 2022; 11(2):18. https://doi.org/10.3390/computers11020018
12. Troussas, C., Virvou, M., and Espinosa, K.J. (2015). Using Visualization algorithms for discovering patterns in groups of users for tutoring multiple languages through social networking. Journal of. Networks, 10(12), pp. 668–674.
13. Virvou, M., Troussas, C., Caro, J., Espinosa, K.J. (2012). User Modeling for Language Learning in Facebook. In: Sojka, P., Horák, A., Kopeček, I., Pala, K. (eds) Text, Speech and Dialogue. TSD 2012. Lecture Notes in Computer Science(), vol 7499. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-32790-2_42
14. Troussas, C., Chrysafiadi, K., Virvou, M. (2018). Machine Learning and Fuzzy Logic Techniques for Personalized Tutoring of Foreign Languages. In: Penstein Rosé, C., et al. Artificial Intelligence in Education. AIED 2018. Lecture Notes in Computer Science(), vol 10948. Springer, Cham. https://doi.org/10.1007/978-3-319-93846-2_67
15. C. Troussas, A. Krouska and M. Virvou, "Integrating an Adjusted Conversational Agent into a Mobile-Assisted Language Learning Application," 2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI), Boston, MA, USA, 2017, pp. 1153-1157, doi: 10.1109/ICTAI.2017.00176.

16. Kanetaki, Z., Stergiou, C., Bekas, G., Troussas, C., & Sgouropoulou, C. (2022). A Hybrid Machine Learning Model for Grade Prediction in Online Engineering Education. International Journal of Engineering Pedagogy (iJEP), 12(3), pp. 4–24. https://doi.org/10.3991/ijep.v12i3.23873.