Αναφορά 3ης Άσκηση Unity
2024-2025

Ανάπτυξη Βιντεοπαιχνιδιών

(CEID_NE565)

Στοιχεία Φοιτητών

Υπεύθυνος Καθηγητής:

Κωνσταντίνος Τσίχλας

Μέλος 1

Ακαδημαϊκό Έτος : 9^{Ο} Εξάμηνο 2023-2024

Ονοματεπώνυμο: Ορέστης Αντώνιος Μακρής **Α-Μ** 1084516

Μέλος 2

Ακαδημαϊκό Έτος : 9^{Ο} Εξάμηνο 2023-2024

Ονοματεπώνυμο: Γρηγόρης Δελημπαλταδάκης **Α-Μ** 1084647

Πανεπιστήμιο Πατρών Τμήμα Μηχανικών Η/Υ και Πληροφορικής

Πάτρα 2024

Contents

Οδηγίες Εκτέλεσης 3
Στόχος του Παιχνιδιού 3
Πλήκτρα του Παιχνιδιού 3
Ερώτημα 1 – Δημιουργία Ολοκληρωμένου Παιχνιδιού 4
Περιγραφή Επιπέδων 4
Level 1 4
Level 2 5
Level 3 5
Level 4 6
Διαφορετικοί Τύποι Εχθρών 6
Normal Enemy 7
Light Enemy 7
Heavy Enemy 7
Συλλεκτικά Αντικείμενα 8
Keys 8
Health 8
Energy και Ore 8
Μενού 9
Ερώτημα 2 – Χρήση Tweening Συστήματος για Πόρτες 9
Ερώτημα 3 – Προσθήκη Walk Toggle 10
Ερώτημα 4 – Προσθήκη Ικανοτήτων στον Παίκτη 11
Jump 11
Throw Fireball 12
Shield 12
Dash 13
Ερώτημα 5 – Προσθήκη Ήχου 13
Σύνδεσμος του Project Folder 15

Οδηγίες Εκτέλεσης

Για την εκτέλεση του παιχνιδιού, ανοίξτε το project folder στο Unity, μεταβείτε στο Startup scene και πατήστε “Play”.

Στόχος του Παιχνιδιού

Το παιχνίδι αποτελείται από 4 levels, αυξανόμενης δυσκολίας.

Ο στόχος του παιχνιδιού είναι να εξουδετερώσετε όλους τους εχθρούς σε κάθε επίπεδο και να φτάσετε στο Goal. Για να το πετύχετε αυτό, θα χρειαστεί να βρείτε και να χρησιμοποιήσετε τα κατάλληλα κλειδιά, τα οποία ξεκλειδώνουν πόρτες μέσα στο επίπεδο. Επιπλέον, σε ορισμένες περιπτώσεις, θα πρέπει να εκτελέσετε προσεκτικό platforming για να προχωρήσετε.

Μόλις φτάσετε στο Goal του Level 4, το παιχνίδι θεωρείται ολοκληρωμένο και επανεκκινείται από το Level 1.

Πλήκτρα του Παιχνιδιού

Movement

Left Click	Κίνηση τύπου point-and-click.
Shift	Με κρατημένο το “Shift” ο παίκτης τρέχει.
Space	Ο παίκτης πραγματοποιεί άλμα.

Abilities

Right Click	Throw Fireball Ability. Προκαλεί damage στους εχθρούς.
Q	Shield Ability. Δημιουργεί μια προστατευτική ασπίδα γύρω από τον παίκτη.
W	Dash Ability. Γρήγορη κίνηση για μικρή απόσταση.

Menu

Esc	Ανοίγει το Menu.

Ερώτημα 1 – Δημιουργία Ολοκληρωμένου Παιχνιδιού

Περιγραφή Επιπέδων

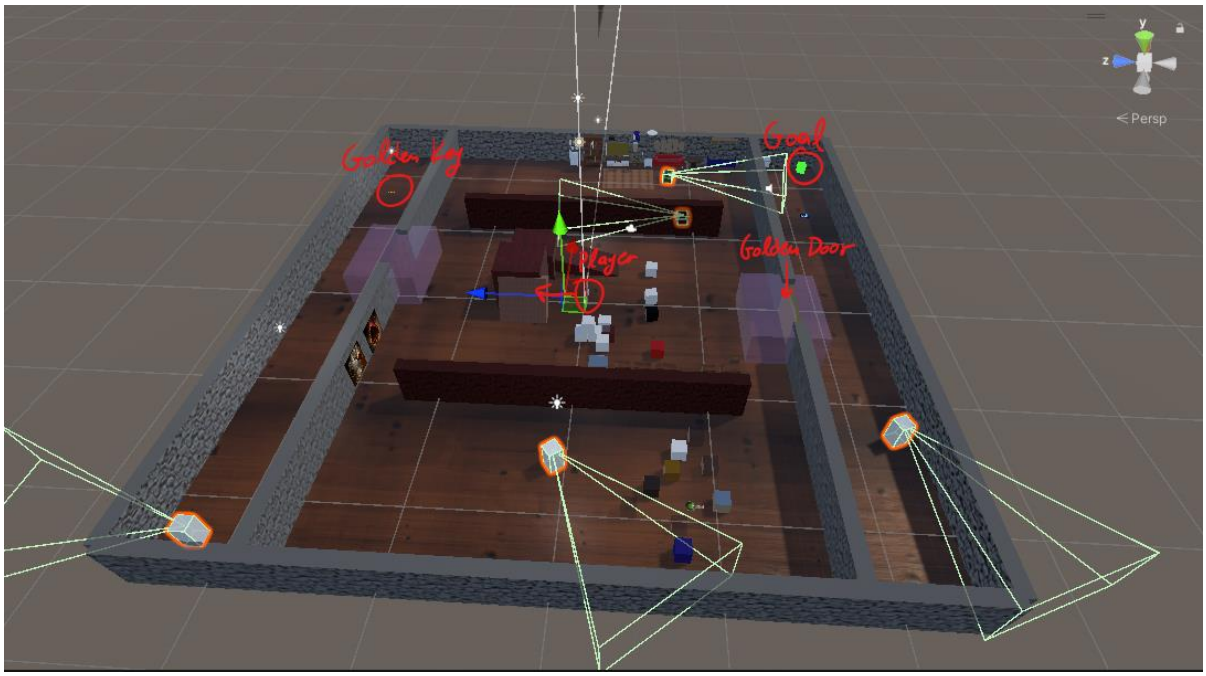
Σε κάθε επίπεδο, ο στόχος είναι να εξολοθρεύσετε όλους τους εχθρούς και να φτάσετε στο Goal. Αν ηττηθείτε από τους εχθρούς, το τρέχον επίπεδο θα επανεκκινηθεί.

Σημειώνεται ότι οι πόρτες με το ξύλινο texture, ανοίγουν χωρίς κλειδί.

Level 1

Enemy Count: 5

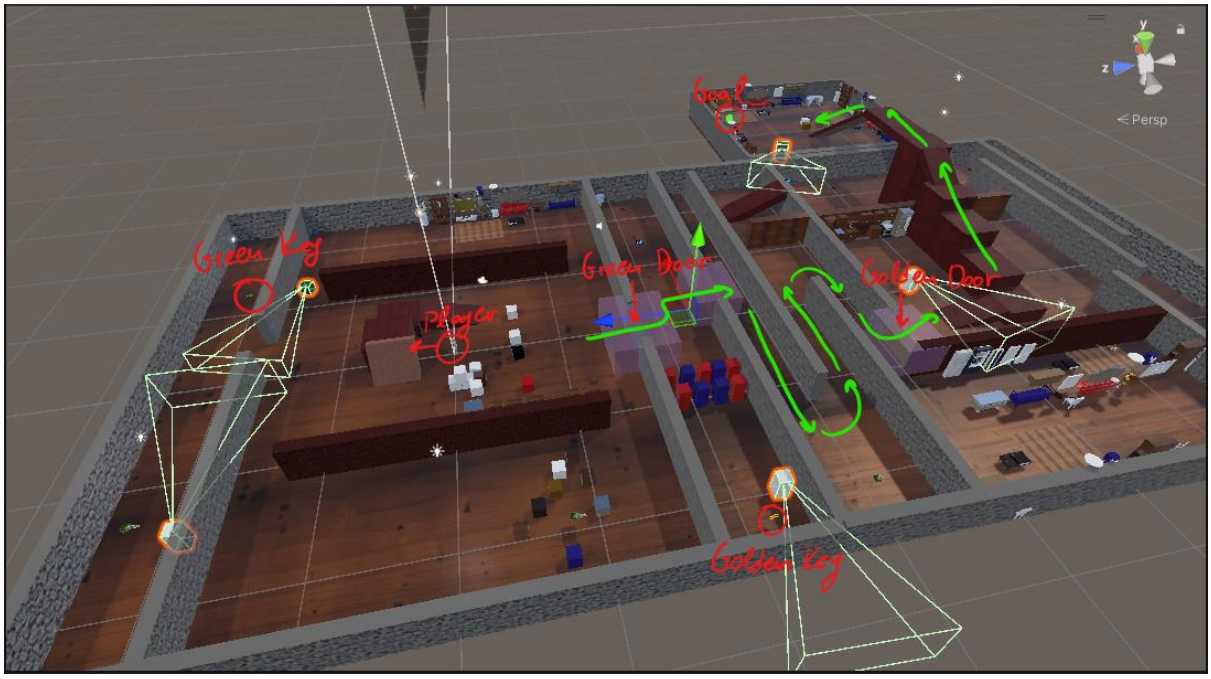
Για να φτάσετε στο Goal πρέπει να συλλέξετε το χρυσό κλειδί και να ανοίξετε την αντίστοιχη πόρτα.



Level 2

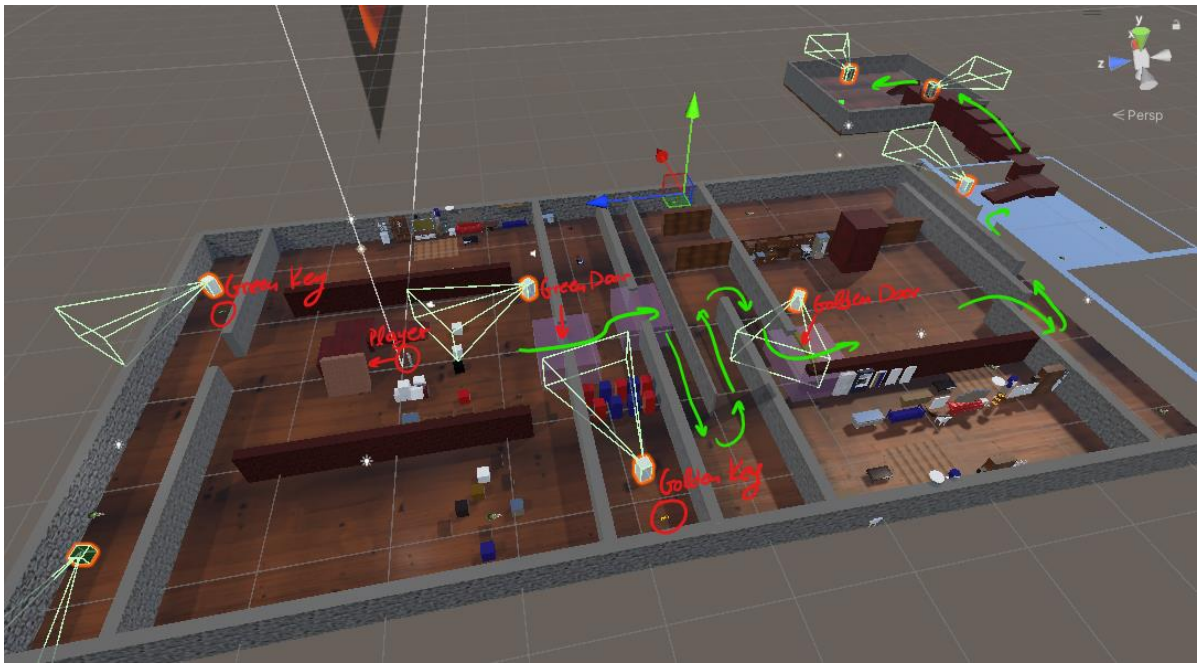
Enemy Count: 6

Στα επίπεδα 2 και 3 υπάρχουν δύο τύποι κλειδιών, πράσινα και μπλε, τα οποία ανοίγουν αντίστοιχες πόρτες. Ο παίκτης πρέπει να τα συλλέξει για να μπορέσει να προχωρήσει στο επίπεδο.



Level 3

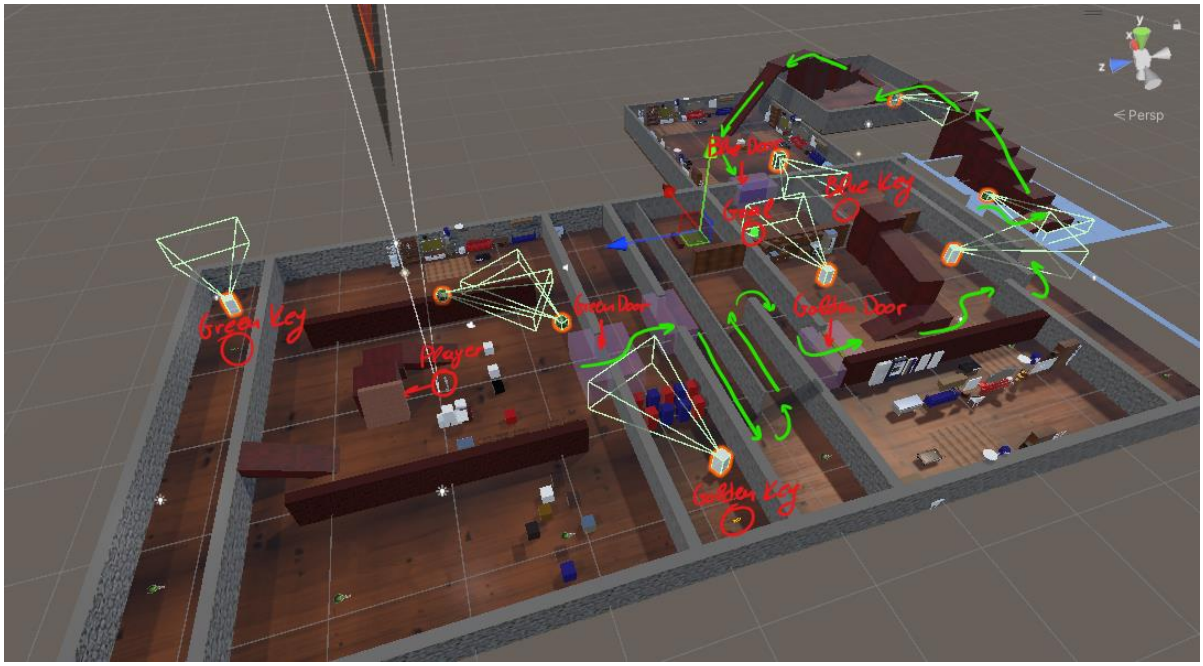
Enemy Count: 8



Level 4

Enemy Count: 9

Στο επίπεδο 4 υπάρχει ένας επιπλέον τύπος κλειδιού (μπλε), το οποίο ανοίγει την τελευταία πόρτα.



Διαφορετικοί Τύποι Εχθρών

Το παιχνίδι διαθέτει τρεις διαφορετικούς τύπους εχθρών, με διαφορετική ποσότητα ζωής και ταχύτητα κίνησης. Κάθε εχθρός αποτελεί ένα ορθογώνιο παραλληλεπίπεδο με διαφορετικές διαστάσεις.

Η κίνηση και η επίθεση κάθε εχθρού ελέγχονται από το script WanderingAI.cs. Το script FOV.cs διαχειρίζεται την ανίχνευση του παίκτη, όταν αυτός βρίσκεται εντός του οπτικού πεδίου (FOV) του εχθρού. Επιπλέον, στα enemy prefabs έχει προστεθεί ένα Wall Trigger object, το οποίο διαθέτει ένα Is Trigger box collider και το script DetectWalls.cs, αποτρέποντας τους εχθρούς από το να περνούν μέσα από τοίχους.

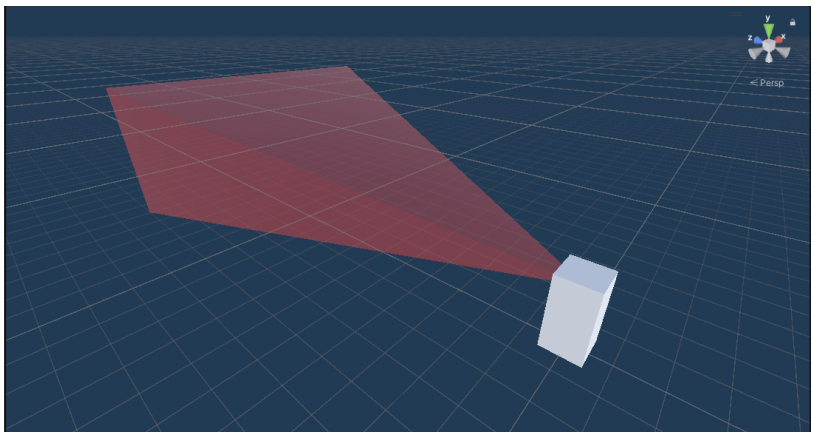
Συμπληρωματικά κάθε level διαθέτει ένα Enemies Manager object με το script EnemyManager.cs. Αυτό το script διαχειρίζεται το spawning και την καταμέτρηση των εχθρών σε κάθε επίπεδο, ενημερώνοντας το σύστημα όταν ένας οι όλοι οι εχθροί έχουν ηττηθεί.

Στο HUD του παιχνιδιού προστέθηκε ένας μετρητής που αυξάνεται κάθε φορά που ηττείται ένας εχθρός. Η λογική για την εμφάνιση του συνολικού αριθμού εχθρών του επιπέδου και του αριθμού των εχθρών που έχουν ηττηθεί προστέθηκε στο UIController.cs script του HUD Canvas object.

Normal Enemy

Speed: 3 m/s

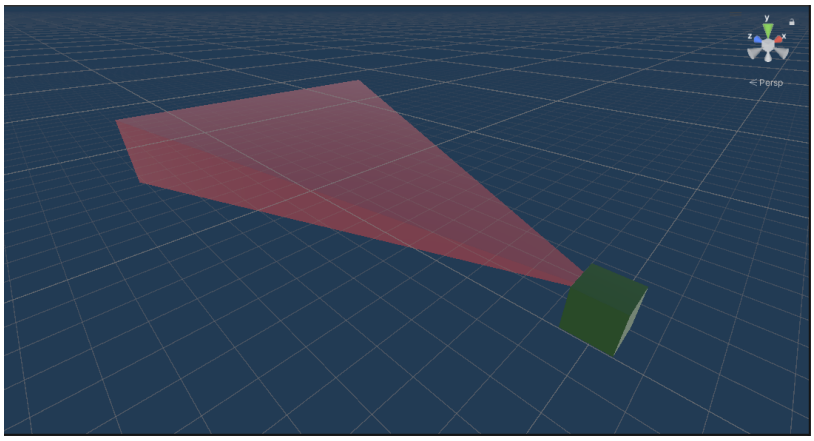
Health: 100



Light Enemy

Speed: 6 m/s

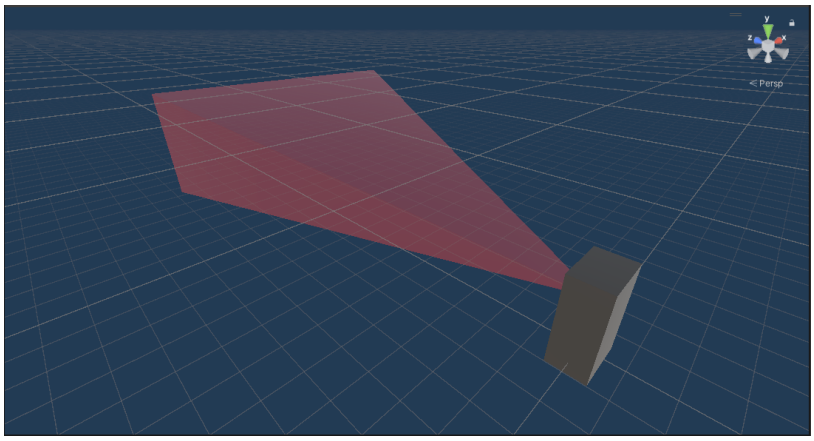
Health: 50



Heavy Enemy

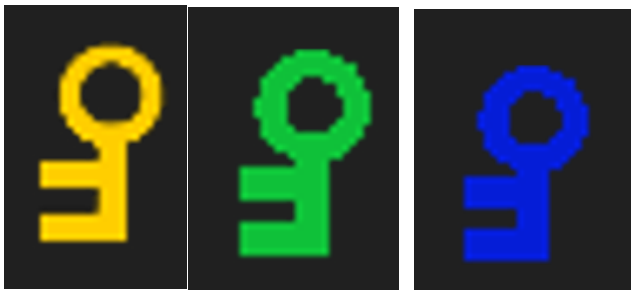
Speed: 1.5 m/s

Health: 200



Συλλεκτικά Αντικείμενα

Keys



Στο παιχνίδι μας, έχουμε τρεις διαφορετικούς τύπους κλειδιών που χρησιμοποιούνται για να ανοίγουν πόρτες με το ίδιο χρώμα. Συγκεκριμένα, τα κλειδιά είναι τα εξής:

Χρυσό Κλειδί (Gold Key): Όταν ο παίκτης συλλέγει το χρυσό κλειδί, αυτό προστίθεται στο inventory και μπορεί να χρησιμοποιηθεί για να ανοίξει της χρησες πόρτες που απαιτούν το χρυσό κλειδί. Levels 1, 2, 3, 4

Πράσινο Κλειδί (Green Key): Όταν ο παίκτης συλλέγει το πράσινο κλειδί, αυτό προστίθεται στο inventory και μπορεί να χρησιμοποιηθεί για να ανοίξει της πρασινες πόρτες που απαιτούν το πράσινο κλειδί. Levels 1, 2, 3, 4

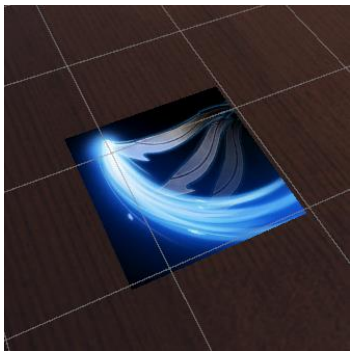
Μπλε Κλειδί (Blue Key): Όταν ο παίκτης συλλέγει το μπλε κλειδί, αυτό προστίθεται στο inventory και μπορεί να χρησιμοποιηθεί για να ανοίξει πόρτες που απαιτούν το μπλε κλειδί. Level 4

Health

Η χρήση αυτού του item, μέσω του menu, αυξάνει την ζωή του παίκτη κατά 25 μονάδες.

Energy και Ore

Τα συγκεκριμένα items δεν έχουν κάποια χρήση στο παιχνίδι και τα αφήσαμε για μελλοντική προσθήκη λειτουργιών.



Για τα collectible items, προστέθηκε μια λειτουργία που διαχειρίζεται τη ροή του παιχνιδιού μεταξύ των επιπέδων μέσω του **UIController**. Συγκεκριμένα, όταν ολοκληρώνεται ή αποτυγχάνει ένα επίπεδο, ενεργοποιείται μια ενέργεια που ενημερώνει το UI με ένα μήνυμα κατάλληλης κατάστασης (όπως "Level Complete!" ή "Level Failed") για λίγα δευτερόλεπτα πριν να προχωρήσει στο επόμενο επίπεδο ή να επανεκκινήσει το τρέχον. Αυτή η λειτουργία αποστέλλει επίσης εντολές στον Inventory Manager για την αφαίρεση ορισμένων αντικειμένων (π.χ. κλειδιών, ενέργειας, και ορυκτών) και εξασφαλίζει την ομαλή μετάβαση μεταξύ των επιπέδων, προσφέροντας μια πιο συνεκτική και ευχάριστη εμπειρία παιχνιδιού.

Αυτή η επιπλέον δυναμική λειτουργία, εκτός του κεντρικού project, συμβάλλει στην ενίσχυση της συνολικής ροής του παιχνιδιού, επιτρέποντας στον παίκτη να απολαύσει ένα συνεκτικό και διακριτικό gameplay ανάμεσα στις διαδοχικές σκηνές.

Ειδικότερα για τα κλειδιά, η διαγραφή τους μετά την ολοκλήρωση ενός επιπέδου αποτρέπει την ενεργοποίηση συστημάτων πόρτας με κλειδιά από προηγούμενα επίπεδα, διατηρώντας έτσι τη σωστή ροή του gameplay.

Μενού

Για την προβολή των νέων αντικειμένων, προσαρμόσαμε το script InventoryPopup.cs ώστε να διαχειρίζεται την εμφάνιση και οργάνωση πολλαπλών αντικειμένων στο **inventory**. Ο κώδικας διατρέχει τη λίστα με τα αντικείμενα και για κάθε θέση στο UI ενεργοποιεί το αντίστοιχο icon και label, προσπαθώντας πρώτα να φορτώσει το συγκεκριμένο sprite από τους πόρους του παιχνιδιού. Σε περίπτωση που δεν βρεθεί ειδικό εικονίδιο και το αντικείμενο περιέχει τη λέξη "key", χρησιμοποιεί το προκαθορισμένο εικονίδιο του κλειδιού από τα φάκελο Assets\Resources\Icons του παιχνιδιού.

Επιπρόσθετα, έχει προστεθεί ένας event trigger για κάθε εικόνα ώστε, με το πάτημα (click), να γίνεται επιλογή του αντικειμένου και να ενημερώνεται το UI (π.χ. ενεργοποίηση των κουμπιών equip και use, ανάλογα με το αντικείμενο που έχει επιλεχθεί). Αυτές οι αλλαγές επιτρέπουν στον παίκτη να βλέπει τα αντικείμενα που διατηρεί, να επιλέγει πιο εύκολα το ενεργό αντικείμενο, και να εκτελεί ενέργειες όπως η εξόπλιση (equip) ή η χρήση (use) τους, ενισχύοντας τη συνολική ροή και διαδραστικότητα του gameplay.

Ερώτημα 2 – Χρήση Tweening Συστήματος για Πόρτες

Παρακάτω ακολουθεί μια αναλυτική περιγραφή των τροποποιήσεων και των επεκτάσεων που υλοποιήσαμε στο project για την λειτουργία του συστήματος των πορτών:

Η χρήση του tweening μέσω του DOTween επιτρέπει στις πόρτες να ανοίγουν και να κλείνουν ομαλά και σταδιακά, δίνοντάς τους μια φυσική κίνηση. Συγκεκριμένα, όταν εκτελείται η εντολή Activate ή Deactivate, η μέθοδος DOMove μεταβάλλει τη θέση της πόρτας σε ένα καθορισμένο χρονικό διάστημα (duration), με αποτέλεσμα η μετάβαση από την κλειστή στην ανοιχτή θέση (και αντίστροφα) να εμφανίζεται ρεαλιστική και αισθητικά ευχάριστη, αποφεύγοντας τις ξαφνικές μετατοπίσεις. Επιπλέον, ο έλεγχος της κατάστασης μέσω της μεταβλητής isMoving διασφαλίζει ότι κατά τη διάρκεια αυτής της ομαλής μετάβασης δεν μπορούν να ενεργοποιηθούν νέες εντολές, καθιστώντας το σύστημα πιο αξιόπιστο και αντιδραστικό στις αλληλεπιδράσεις του παίκτη.

Στο Activate() και το Deactivate(), γίνεται έλεγχος τόσο του τρέχοντος state (αν η πόρτα είναι ανοιχτή ή κλειστή) όσο και αν αυτή κινείται, για να αποφευχθούν διπλές ενέργειες ή παρεμβαίσεις. Επιπλέον, υπάρχει η δυνατότητα "αυτοανύψωσης" της πόρτας μετά το κλείσιμο (auto-rise) με την χρήση της Invoke – που μπορεί να ενεργοποιηθεί απλά ξανά διασχίζοντάς το barrier της ενεργοποίησης της πόρτας. Αυτή η λογική εξασφαλίζει ότι οι πόρτες που απαιτούν κλειδί (key doors)

και αυτές που δεν το απαιτούν (non-key doors) συμπεριφέρονται σωστά, με τις key doors να παραμένουν ενεργές (χαμηλωμένες) μόλις ενεργοποιηθούν, εκτός αν ο παίκτης επαναπροσπαθήσει με το σωστό κλειδί.

Στο αρχείο DeviceTrigger.cs, ενισχύσαμε την επικοινωνία μεταξύ του trigger και της πόρτας μέσω της χρήσης SendMessage. Ο κώδικας ελέγχει αν το αντικείμενο που εισέρχεται στο trigger έχει την ετικέτα "Player", διασφαλίζοντας ότι μόνο ο παίκτης είναι αρμόδιος για την ενεργοποίηση και απενεργοποίηση της πόρτας. Επιπλέον, ο έλεγχος του requiredKey επιτρέπει την διάκριση ανάμεσα στις πόρτες που απαιτούν κλειδί και στις απλές, ώστε οι key doors να ενεργοποιούνται μόνο όταν ο παίκτης έχει κάνει equip το σωστό αντικείμενο από την Inventory.

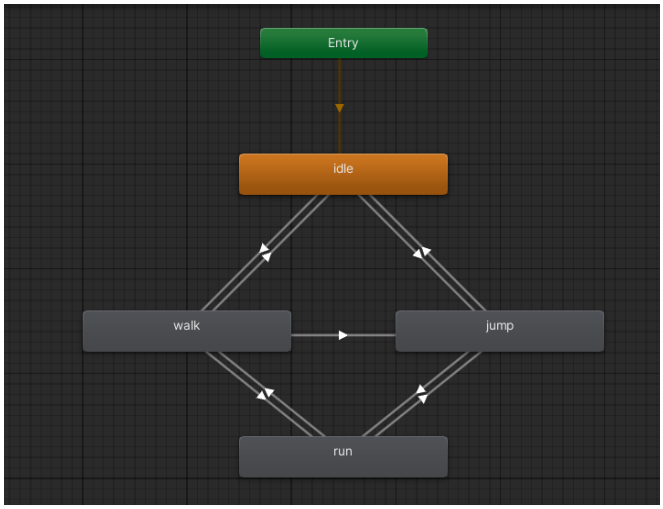
Με αυτές τις τροποποιήσεις, έχουμε επιτύχει έναν διαχωρισμό των λειτουργιών κάθε στοιχείου: οι πόρτες χειρίζονται ομαλά τις κινήσεις τους με την προστασία από συνεχείς ενεργοποιήσεις, ενώ τα triggers διαχειρίζονται τη χρήση των αντικειμένων με βάση την κατάσταση του παίκτη και τις απαραίτητες απαιτήσεις (όπως η χρήση του σωστού κλειδιού). Αυτή η προσέγγιση ενισχύει την αλληλεπίδραση και την ρεαλιστική αίσθηση του gameplay στο Unity.

Σε αυτό το project έχουμε υλοποιήσει τρεις τύπους θυρών. Συγκεκριμένα, η ξύλινη πόρτα ενεργοποιείται αυτόματα χωρίς να απαιτείται κλειδί, ενώ υπάρχει και ο τύπος των θυρών που απαιτούν τη χρήση τριών διαφορετικών κλειδιών (χρυσό, πράσινο και μπλε). Ο κώδικας στο DeviceOperator.cs επιτρέπει στον παίκτη να αλληλεπιδρά με τις πόρτες χρησιμοποιώντας μια ακτίνα (OverlapSphere) γύρω από τη θέση του. Όταν ο παίκτες βρίσκετε εντός της ακτίνας, ο κώδικας ελέγχει τα αντικείμενα που βρίσκονται μπροστά του (βάσει του προσανατολισμού) και στέλνει το μήνυμα "Operate" ώστε οι πόρτες να εκτελέσουν τη σωστή λειτουργία (άνωμα/κατάβασμα).

Ερώτημα 3 – Προσθήκη Walk Toggle

Το PointClickMovement.cs script ενημερώθηκε, ώστε να ελέγχει εάν ο χρήστης κρατάει πατημένο το πλήκτρο Shift. Σε αυτή την περίπτωση αυξάνεται η ταχύτητα του παίκτη.

Συμπληρωματικά, το animation του παίκτη ενημερώθηκε, ώστε να αποτυπώνει ομαλά την εναλλαγή μεταξύ περπατήματος (walking) και τρεξίματος (running).



Ερώτημα 4 – Προσθήκη Ικανοτήτων στον Παίκτη

Προστέθηκαν τρεις ικανότητες στον παίκτη. Ο χρήστης μπορεί να δει τις ικανότητες που διαθέτει και τα αντίστοιχα πλήκτρα ενεργοποίησης, κάτω-αριστερά στο HUD.

Συμπληρωματικά, για την οπτικοποίηση της διάρκειας του cooldown των ικανοτήτων “Shield” και “Dash”, προστέθηκε ένα κυκλικό UI pan effect στα icons.

Jump

Πλήκτρο Ενεργοποίησης: Space

Cooldown: None

Για την προσθήκη της δυνατότηταw άλματος για τον παίκτη, τροποποιήσαμε το script PointClickMovement.cs. Συγκεκριμένα, όταν ανιχνεύουμε ότι ο χαρακτήρας είναι στο έδαφος (μέσω ενός raycast που ελέγχει την απόσταση μέχρι το έδαφος floor στην προκείμενη περίπτωση), επιτρέπουμε στο χρήστη να πραγματοποιήσει άλμα πατώντας το πλήκτρο Space. Αν ο παίκτης πατήσει Space ενώ βρίσκεται στο έδαφος, εφαρμόζουμε μια δύναμη άλματος (jumpForce) που αυξάνει την κατακόρυφη ταχύτητα (vertSpeed), ενεργοποιώντας την κατάσταση "Jumping" στον Animator. Επιπλέον, όταν ο χαρακτήρας δεν είναι στο έδαφος, εφαρμόζουμε βαρύτητα ώστε να προσομοιώσουμε την πτώση, διατηρώντας ταυτόχρονα το όριο της τελικής ταχύτητας (terminalVelocity). Η λογική πίσω από τις τροποποιήσεις αυτές είναι να επιτευχθεί ένας φυσικός και ομαλός συνδυασμός κίνησης και άλματος, δίνοντας στον παίκτη την αίσθηση ότι κινείται με ρεαλιστικές ιδιότητες φυσικής, ενώ το σύστημα του Animator ενημερώνεται ανάλογα για τις αλλαγές στην κατάσταση κίνησης και άλματος.

Throw Fireball

Πλήκτρο Ενεργοποίησης: Right-Click

Cooldown: None

Για την υλοποίηση αυτής της ικανότητας δημιουργήθηκε ένα PlayerFireball prefab το οποίο έχει μικρότερες διαστάσεις από αυτό των εχθρών και προκαλεί 5 πόντους damage όταν έρθει σε επαφή με κάποιον εχθρό.

Η ρίψη του fireball ελέγχεται από το ThrowFireball.cs script. Πατώντας right-click σε οποιοδήποτε σημείο της οθόνης, ο παίκτης περιστρέφεται ομαλά προς την κατεύθυνση που επιλέχθηκε και αρχικοποιεί ένα PlayerFireball prefab.

Η συγκεκριμένη ικανότητα δεν έχει cooldown, καθώς αυτό θα έκανε το παιχνίδι πολύ δύσκολο.

Shield

Πλήκτρο Ενεργοποίησης: Q

Duration: 3 sec

Cooldown: 5 sec

Για την υλοποίηση αυτής της ικανότητας δημιουργήθηκε ένα Shield prefab με ένα μπλε διαφανές capsule mesh και collider.

Το shield ενεργοποιείται μέσω του UseShield.cs script. Εάν ο χρήστης πατήσει “Q”, το shield δεν χρησιμοποιείται ήδη και το cooldown έχει ολοκληρωθεί, αρχικοποιείται ένα Shield prefab γύρω από τον παίκτη, προστατεύοντας τον από τα fireballs των εχθρών.

Συμπληρωματικά, το ShieldController.cs script του Shied prefab ενημερώνει το UI για την κατάσταση του cooldown.

Dash

Πλήκτρο Ενεργοποίησης: W

Dash Speed: 20 m/s

Duration: 0.3 sec

Cooldown: 5 sec

Η συγκεκριμένη ικανότητα ελέγχεται μέσω του UseDash.cs script. Αυτό το script επιτρέπει στον παίκτη να εκτελέσει ένα “dash” προς τα εμπρός, διαχειρίζεται τη διάρκειά του, την περίοδο του cooldown και ενημερώνει το UI για την κατάσταση του cooldown.

Ερώτημα 5 – Προσθήκη Ήχου

Ο AudioManager χειρίζεται το σύστημα ήχου της εφαρμογής με έναν κεντρικό και τυποποιημένο τρόπο. Υλοποιημένος ως singleton, εξασφαλίζει ότι υπάρχει μόνο μία ενεργή παρουσία του κατά τη διάρκεια της λειτουργίας της εφαρμογής. Αυτό επιτυγχάνεται μέσα από το Awake(), όπου αν ήδη υπάρχει ένα instance, το νέο αντικείμενο καταστρέφεται, ενώ διαφορετικά ορίζεται το τρέχον αντικείμενο ως Instance και το αντικείμενο αυτό δεν καταστρέφεται κατά τη φόρτωση νέων σκηνών (DontDestroyOnLoad).

Ο κώδικας δημιουργεί δύο ξεχωριστές πηγές ήχου (AudioSource): μία για την κύρια μουσική και μία για τα sound effects (SFX). Στην περίπτωση της κύριας μουσικής, ο AudioSource ρυθμίζεται με το clip που έχει οριστεί στο inspector, η αναπαραγωγή γίνεται επαναλαμβανόμενη (loop), και ο ήχος παίζει άμεσα μετά τη διαμόρφωση του μέσω του Play(). Αυτή η προσέγγιση επιτρέπει την αδιάκοπη αναπαραγωγή επαναληπτικής μουσικής υπόκρουσης καθ' όλη τη διάρκεια του παιχνιδιού, ενώ προσφέρουμε τη δυνατότητα ρύθμισης του volume μέσω της μεταβλητής mainMusicVolume.

Για τα sound effects, δημιουργείται μια ανεξάρτητη πηγή ήχου (sfxSource) η οποία ρυθμίζεται ώστε να μην επαναλαμβάνει αυτόματα άλλα από τα one-shot εφέ που παίζονται με την κλήση της μεθόδου PlaySound(). Αυτή η μέθοδος επιτρέπει την αναπαραγωγή συγκεκριμένων ηχητικών αποσπασμάτων (π.χ. ήχο συλλογής αντικειμένου, ενεργοποίησης πορτών κ.λπ.) χωρίς να διακόπτουν την αναπαραγωγή του υπόλοιπου ήχου. Επίσης, παρέχεται η δυνατότητα υπέρβασης του default volume μέσω ενός προαιρετικού παραμέτρου volumeScale. Για παράδειγμα:

AudioManager.Instance.PlaySound(shootSound, 0.05f);

Σε αυτή την περίπτωση αρχικοποιούμε τον ήχο για τα player fireballs με την ένταση 0.05.

Οι μέθοδοι SetMainMusicVolume() και SetSfxVolume() ενημερώνουν τις εσωτερικές μεταβλητές volume και προσαρμόζουν άμεσα την ένταση των αντίστοιχων πηγών ήχου, ενώ με τις GetMusicVolume() και GetSfxVolume() μπορούμε να ανακτήσουμε τις τρέχουσες τιμές, εξασφαλίζοντας έτσι εύκολη διαχείριση και δυναμική ρύθμιση ήχου κατά την εκτέλεση.

Η οργάνωση αυτή του κώδικα προσφέρει έναν απομονωμένο χώρο διαχείρισης ήχου, όπου όλα τα ηχητικά εφέ και η μουσική της εφαρμογής ελέγχονται κεντρικά, καθιστώντας την διαχείριση του ήχου πιο εύκολη και πιο αξιόπιστη. Αυτό είναι ιδιαίτερα χρήσιμο όταν οι ρυθμίσεις του ήχου προσαρμόζονται μέσα από ένα UI, όπου οι χρήστες μπορούν να αλλάξουν οι εντάσεις της μουσικής ή των εφέ και οι αλλαγές αυτές αντικατοπτρίζονται άμεσα στο gameplay χωρίς επιπλέον διακοπές.

Το UI του audio συστήματος έχει επίσης σχεδιαστεί με γνώμονα την ευκολία και την άμεση αλληλεπίδραση. Μέσα από τον Unity Editor, έχουμε υλοποιήσει sliders που επιτρέπουν την ρύθμιση του master volume, της κύριας μουσικής και του SFX. Οι αλλαγές που γίνονται στα sliders καλούν μεθόδους όπως οι SetMainMusicVolume() και SetSfxVolume() του AudioManager, με αποτέλεσμα

να ενημερώνεται δυναμικά το επίπεδο του ήχου στο παιχνίδι. Έτσι, ο χρήστης μπορεί να προσαρμόζει την εμπειρία ήχου σύμφωνα με τις προτιμήσεις του και να απολαμβάνει ένα πιο ευχάριστο και εξατομικευμένο περιβάλλον παιχνιδιού. Το volume των sfx είναι ορισμένα έτσι ώστε να εφαρμόζουν την μεταβολή της έντασης του ήχου ως % επι της μεταβλητής volumeScale για κάθε διαφορετικό sound.

Συνολικά, η δομή αυτή του AudioManager εξασφαλίζει όχι μόνο μια καθαρή και οργανωμένη διαχείριση του ήχου, αλλά επίσης συμβάλλει στη συνολική αισθητική εμπειρία του παιχνιδιού μέσω της ομαλής και συνεπούς του λειτουργίας. Η δυνατότητα να αλλάζουν δυναμικά τα ηχητικά επίπεδα καθιστά το σύστημα ευέλικτο και έτοιμο να αντιμετωπίσει τις ανάγκες ενός σύγχρονου παιχνιδιού, ενώ η προσέγγιση με βάση το singleton και τις διακεκριμένες πηγές ήχου επιτρέπει την ανάπτυξη ενός σταθερού και επεκτάσιμου συστήματος ήχου.

1. **Κύρια Μουσική (mainMusic):**
    - **Σκοπός:** Παρέχει το background music του παιχνιδιού.
    - **Τοποθεσία/Αναφορά:** Ρυθμίζεται στον επιθεωρητή μέσω του AudioManager.
2. **Ήχος Αύξησης Υγείας (**healthIncreaseSound **στο** PlayerManager**):**
    - **Σκοπός:** Παίζεται όταν αυξάνεται η υγεία του παίκτη.
    - **Τοποθεσία/Αναφορά:** Αναφέρεται στο PlayerManager για να παρέχει ακουστική ανάδραση όταν ο παίκτης κερδίζει υγεία.
3. **Ήχος Νίκης (**winSound **στο** ObjectiveTrigger**):**
    - **Σκοπός:** Παίζεται όταν ο παίκτης φτάνει στον στόχο και έχουν εξολοθρευτεί όλοι οι εχθροί.
    - **Τοποθεσία/Αναφορά:** Χρησιμοποιείται στο ObjectiveTrigger για να παρέχει ακουστική ανάδραση για την ολοκλήρωση του επιπέδου.
4. **Ήχος Πυροβολισμού player (**shootSound **στο** PlayerFireball**):**
    - **Σκοπός:** Παίζεται όταν ο παίκτης ρίχνει ένα fireball.
    - **Τοποθεσία/Αναφορά:** Χρησιμοποιείται στο PlayerFireball για να παρέχει ακουστική ανάδραση όταν δημιουργείται ένα fireball.
5. **Ήχος Πυροβολισμού Εχθρού (**enemyShootSound **στο** EnemyFireball**):**
    - **Σκοπός:** Παίζεται όταν ένας εχθρός πυροβολεί ένα fireball.
    - **Τοποθεσία/Αναφορά:** Χρησιμοποιείται στο EnemyFireball για να παρέχει ακουστική ανάδραση όταν δημιουργείται ένα enemy fireball.
6. **Ήχος Χτυπήματος Παίκτη (**playerHitSound **στο** EnemyFireball**):**
    - **Σκοπός:** Παίζεται όταν ο παίκτης χτυπιέται από ένα fireball εχθρού.
    - **Τοποθεσία/Αναφορά:** Χρησιμοποιείται στο EnemyFireball για να παρέχει ακουστική ανάδραση όταν ο παίκτης χτυπιέται.
7. **Ήχος Συλλογής Αντικειμένου (**collectItemSound **στο** CollectibleItem**):**
    - **Σκοπός:** Παίζεται όταν ο παίκτης συλλέγει ένα αντικείμενο.

- **Τοποθεσία/Αναφορά:** Χρησιμοποιείται στο CollectibleItem για να παρέχει ακουστική ανάδραση όταν συλλέγεται ένα αντικείμενο.

8. **Ήχος Ενεργοποίησης Ασπίδας (**shieldSound **στο** ShieldController**):**
    - **Σκοπός:** Παίζεται όταν ενεργοποιείται η ασπίδα του παίκτη.
    - **Τοποθεσία/Αναφορά:** Χρησιμοποιείται στο ShieldController για να παρέχει ακουστική ανάδραση όταν ενεργοποιείται η ασπίδα.
9. **Ήχος Χτυπήματος Εχθρού (**hitSound **στο** ReactiveTarget**):**
    - **Σκοπός:** Παίζεται όταν ένας εχθρός χτυπιέται από την επίθεση του παίκτη.
    - **Τοποθεσία/Αναφορά:** Χρησιμοποιείται στο ReactiveTarget για να παρέχει ακουστική ανάδραση όταν ένας εχθρός χτυπιέται.
10. **Ήχος Θανάτου Εχθρού (**deathSound **στο** ReactiveTarget**):**
    - **Σκοπός:** Παίζεται όταν ένας εχθρός ηττάται.
    - **Τοποθεσία/Αναφορά:** Χρησιμοποιείται στο ReactiveTarget για να παρέχει ακουστική ανάδραση όταν ένας εχθρός πεθαίνει.
11. **Ήχος Ανοίγματος/Κλεισίματος Πόρτας (doorSound στο** DoorOpenDevice**):**
    - **Σκοπός:** Παίζεται όταν ανοίγει ή κλείνει μια πόρτα.
    - **Τοποθεσία/Αναφορά:** Χρησιμοποιείται στο DoorOpenDevice για να παρέχει ακουστική ανάδραση όταν ενεργοποιείται ή απενεργοποιείται μια πόρτα.

Σύνδεσμος του Project Folder

https://drive.google.com/file/d/1JglBDp2smrCDq8HYuu2u7X73rgPWE6Tq/view?usp=sharing