Ταμπαξής Παλληκάρης Γιώργος

ΑΜ 1072487

Βλάχος Σταμάτης

ΑΜ 1072623

Link Google Drive: https://drive.google.com/drive/folders/1j-TA1bUSYTcOBUT7vF7t4EevIuAXCiyi?usp=sharing

Ερώτημα 1ο

Για την κατάβαση του χαρακτήρα η υλοποίηση και η λογική πίσω από αυτήν ήταν απλή. Το script του **Player** ελέγχει αν πατηθεί το πλήκτρο **(S)** ή το **(arrow)** και αν ο παίχτης είναι στο έδαφος **(is grounded)**. Όταν πατηθεί ένα από τα 2 κουμπιά τότε απενεργοποιείται προσωρινά η επαφή του παίχτη με την πλατφόρμα με το **Physics2D.IgnoreCollision**.

Ερώτημα 2ο

Η πρώτη επέκταση που έγινε ήταν προσθήκη ενός trap, τον **spikes**. Η λογική πίσω από τα spikes έχει ως εξής, όταν ο player πέσει πάνω τους ενεργοποιείται η μέθοδος **Die()** και στο script του Player έχουμε βάλει την μέθοδο **Die()** όπου επιστρέφει τον Player στην αρχική του θέση. Η ανίχνευση της σύγκρουσης γίνεται με την **OnCollisionEntrer2D**. Για τα κενά στα οποία μπορεί να πέσει κάτω ο Player βάλαμε ένα threshold το οποίο αν το ξεπεράσει ο player όσο πέφτει τον επιστρέφει πάλι στην αρχική του θέση **transform.position = startPosition;** . Η start position είναι μια μεταβλητή που καθορίζει την αρχική θέση του παίχτη και το transform.position τον μεταφέρει εκεί.

Για τις κινούμενες πλατφόρμες χρησιμοποιήσαμε τον κώδικα της ήδη δοσμένης πλατφόρμας απλά αλλάξαμε το **start** και **end** position και στους δύο άξονες έτσι ώστε να πάνε πάνω κάτω ή αριστερά δεξιά σε διαφορετικά σημεία του παιχνιδιού.

Για το τραμπολίνο φτιάξαμε ένα gameObject στο οποίο βάλαμε τα κατάλληλα components στον inspector, δημιουργήσαμε ένα script και ένα animator με animation. Η λογική πίσω από αυτό είναι ότι όταν κάνει detect collision, ελέγχει αν είναι ο παίχτης αυτός που κάνει την επαφή και αν είναι του ασκεί ένα bounceForce που τον κάνει να αναπηδά.

Για τον τελικό στόχο χρησιμοποιήσαμε μια σημαία την όπου όταν υπάρξει επαφή με τον player εμφανίζει ένα μήνυμα στο console που λέει **You win!** Αυτό έγινε πάλι με την μέθοδο **onTriggerEnter2D.**

Ερώτημα 3ο

Για αυτό το ερώτημα φτιάξαμε 3 scripts.

Wandering_Enemy.cs:

- Περιγράφει έναν περιπλανώμενο εχθρό που κινείται αριστερά-δεξιά εντός προκαθορισμένων ορίων.

- Ο εχθρός αλλάζει κατεύθυνση όταν φτάσει στα όρια περιπολίας ή όταν εντοπίσει εμπόδιο μέσω Raycast.

Shooting_Enemy.cs:

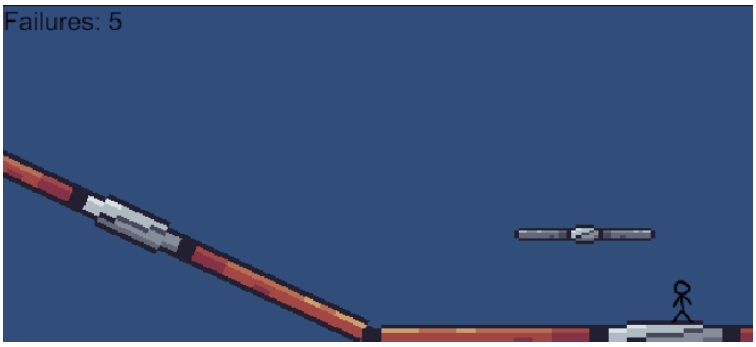
- Υλοποιεί έναν στατικό εχθρό που εντοπίζει τον παίκτη εντός εμβέλειας και πυροβολεί προς αυτόν.
- Περιλαμβάνει λογική περιστροφής του σημείου εκτόξευσης (firePoint) προς τον παίκτη και εκτόξευσης βλημάτων με συγκεκριμένο ρυθμό.

Projectile.cs:

- Διαχειρίζεται τη συμπεριφορά βλημάτων που εκτοξεύονται από εχθρούς.
- Τα βλήματα έχουν περιορισμένη διάρκεια ζωής και εξαφανίζονται όταν συγκρούονται με τον παίκτη.

Ερώτημα 4ο

Το ερώτημα αυτό έλεγε να φτιάξουμε ένα σύστημα καταμέτρησης αποτυχιών όπου να εμφανίζονται στο UI του παιχνιδιού. Όπως είπαμε και πριν έχουμε προσθέσει την λογική στα spikes στους αντιπάλους και όταν πέφτεις από την πίστα να επιστρέφεις στην αρχική σου θέση. Έτσι για αυτό το ερώτημα βάλαμε ένα failcounter όπου κάθε φορά που επιστρέφει στην αρχική θέση αυξάνεται κατά ένα, με την εισαγωγή του RegisterFailure();. Αυτό φαινόταν πάνω αριστερά καθόλη την διάρκεια του παιχνιδιού και με την χρήση των playerprefs αποθηκευόταν με την τιμή Failcount.



Ετσι ήταν στην αρχή πριν βάλουμε και τα άλλα ζητούμενα στο UI.

(Κάτι ακόμα που πρέπει να αναφερθεί είναι ότι όταν πέφτει πάνω σε spikes ο Player μπορεί να αυξηθεί κατά 2 το failcount γιατί τα spikes είναι 3 διαφορετικά game objects.)

Ερώτημα 5ο

Το pause menu περιλαμβάνει τις εξής λειτουργίες, με το πάτημα του p σταματάει το παιχνίδι και συνεχίζει με το πάτημα του για δεύτερη φορά. Το κουμπί restart του UI όταν πατηθεί γίνεται reset τον αποτυχιών και αν δεν βρίσκεσαι ήδη στην αρχή σε επιστρέφει στην αρχική θέση. Το resum απλά σε επιστρέφει στο παιχνίδι. Τώρα για τα επίπεδα δυσκολίας απλά κάθε φορά που επιλέγεις κάποιο από τα επίπεδα αυξάνει ή μειώνει την ταχύτητα τον αντιπάλων έτσι ώστε να είναι πιο έυκολο ή πιο δύσκολο να τους αποφύγεις. Αυτό γίνεται με το AdjustEnemySpeed(). Τέλος προσθέσαμε στο UI και την καταμέτρηση αποτυχιών που χρησιμοποιεί την ίδια λογική απλά κάναμε attach το text του pause_menu_canva στον Player.

Ένα πρόβλημα που υπάρχει είναι πως το όταν πατάς το p λειτουργεί μόνο όταν είναι cheked το box του pause_menu_canva. Δυστυχώς δεν καταφέραμε να το λύσουμε αυτό το ζήτημα, οπότε το βάλαμε πάνω πάνω το canvas και κάνουμε χειροκίνητα το pause δηλαδή κάνουμε check και uncheck το box και όταν είναι checked μπορούμε να πατήσουμε το p.