ΕΛΛΗΝΙΚΟ ΜΕΣΟΓΕΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Τίτλος Πτυχιακής εργασίας:

Ανάπτυξη δισδιάστατου παιχνιδιού στο Unity

Μαυρογενάκης Νικόλαος ΑΜ 4949

Επιβλέπων εκπαιδευτικός:

Ιωάννης Παχουλάκης

Ιανουάριος 2024

Ευχαριστίες

Θα ήθελα να εκφράσω την εγκάρδια ευγνωμοσύνη μου προς τους γονείς μου, τους φίλους μου και τον επιβλέπων καθηγητή μου, τον Κ. Ιωάννη Παχουλάκη για την ανεκτίμητη υποστήριξη που μου παρείχαν κατά την διάρκεια υλοποίησης της πτυχιακής μου εργασίας. Ο καθηγητής μου απέδειξε την υποστήριξη του με την εμπειρία και τις συμβουλές τις οποίες παρείχε αποτελώντας ουσιαστική βοήθεια για την επίτευξη του στόχου. Χωρίς την στήριξη τους, λοιπόν, δεν θα ήταν επιτυχής η υλοποίηση αυτού του σημαντικού έργου.

Abstract

The subject of this thesis concerns the creation of a two-dimensional platform game which has been developed in the Unity Game Engine and is compatible for computers. Various tilesets were used to create the maps and obstacles, as well as the Aseprite program to develop the characters and different animations. In this platform game, called “Mr.JackCat”, the player is given the opportunity to collect a variety of objects, face enemies and also avoid difficult obstacles. Various mechanisms regarding the detection of collision between the main character and other objects and enemies have also been implemented, enabling several exciting animations and functions. The game environment and its graphics have an attractive design so that the player can enjoy it to the maximum along with a pleasant background music. As detailed below there is an explanation of how the game functions were designed and a presentation of the source code’s logic with a graphical display.

Σύνοψη

Το θέμα της πτυχιακής αφορά την δημιουργία ενός δισδιάστατου παιχνιδιού πλατφόρμας το οποίο έχει αναπτυχθεί στο Unity Game Engine και είναι συμβατό για υπολογιστές. Χρησιμοποιήθηκαν διάφορα tilesets για την δημιουργία των χαρτών και των εμποδίων όπως και το πρόγραμμα Aseprite για την ανάπτυξη των χαρακτήρων και διάφορων animations. Σε αυτό το παιχνίδι πλατφόρμας "Mr.JackCat" δίνεται η δυνατότητα στον παίκτη να συλλέξει διάφορα αντικείμενα, να αντιμετωπίσει εχθρούς, αλλά και να αποφύγει δύσκολα εμπόδια. Έχουν εφαρμοστεί, επίσης, διάφοροι μηχανισμοί ανίχνευσης σύγκρουσης του βασικού χαρακτήρα με άλλα αντικείμενα και εχθρούς, ενεργοποιώντας διάφορα συναρπαστικά animations και λειτουργίες. Το περιβάλλον του παιχνιδιού και τα γραφικά του έχουν ελκυστικό σχεδιασμό, ώστε ο παίκτης να μπορεί να το απολαύσει στο μέγιστο μαζί με μια ευχάριστη μουσική υπόκρουση. Παρακάτω εξηγείται λεπτομερώς ο τρόπος σχεδίασης των λειτουργιών του παιχνιδιού και η λογική του πηγαίου κώδικα με παρουσίαση γραφημάτων.

Πίνακας περιεχομένων

1. Εισαγωγή **1**

1.1 Περίληψη του παιχνιδιού 2

1.2 Κίνητρο για την δημιουργία και στόχοι 3

1.3 Δομή των κεφαλαίων 3

2. Τεχνολογίες και προγράμματα που χρησιμοποιήθηκαν **4**

2.1 Βασικό πρόγραμμα υλοποίησης 5

2.2 Δευτερεύοντα προγράμματα υλοποίησης παιχνιδιού 6

2.3 Σε τι χρησιμεύει η μηχανή πεπερασμένων καταστάσεων 6

3. Σχεδίαση βασικών χαρακτήρων και χαρτών **4**

3.1 Βασικό μενού και μενού παύσης 5

3.1.1 Μηχανή πεπερασμένων καταστάσεων των μενού 6

3.1.2 Μενού UI 6

3.2 Σχεδίαση βασικού παίκτη και animations 5

3.2.1 Διεπαφή χρήστη του παίκτη 6

3.3 Υλοποίηση πίστας 5

3.4 Εχθροί και εμπόδια 5

3.4.1 Εχθροί 6

3.4.2 Εμπόδια 6

3.5 Συλλεκτικά αντικείμενα 5

3.6 Τι είναι NPC παίκτες και πώς χρησιμοποιήθηκαν 5

4. Υλοποίηση του κώδικα

4.1 Κίνηση της κάμερας 5

4.2 Παίκτης 5

4.2.1 Κίνηση του παίκτη 6

4.2.2 Ρίψη αντικειμένων στους εχθρούς 6

4.2.3 Ζωή του παίκτη 6

4.2.4 Τέλος παιχνιδιού 6

4.3 Μενού 5

4.3.1 Βασικό μενού 6

4.3.2 Μενού παύσης 6

4.4 Εχθροί 6
4.4.1 Μαϊμού 6
4.4.2 Σκελετός 6
4.4.3 Αράχνη 6
4.4.4 Βασικός εχθρός 6
4.5 Εμπόδια 6
4.6 Διεπαφή χρήστη 6
4.6.1 Διάλογοι 6
4.6.2 Συλλογή αντικειμένων και απεικόνιση του σκορ 6
4.6.3 NPC χαρακτήρας 6
4.6.4 Ηχητικά εφέ 6
4.7 Σημεία αποθήκευσης προόδου 6
4.8 Αποθήκευση παιχνιδιού 6
5. Σύνοψη
5.1 Συμπεράσματα 6
5.2 Μελλοντικές εργασίες και επεκτάσεις 6
5.3 Προκλήσεις κατά την υλοποίηση του παιχνιδιού 6
Βιβλιογραφία

Λίστα εικόνων

Εικόνα 1: Διάγραμμα καταστάσεων του βασικού μενού 6
Εικόνα 2: Διάγραμμα καταστάσεων του μενού παύσης 7
Εικόνα 3: Βασικό μενού 8
Εικόνα 4: Μενού παύσης 8
Εικόνα 5: Τέλος παιχνιδιού 9
Εικόνα 6: Τερματισμός παιχνιδιού 9
Εικόνα 7: Περιβάλλον 2ης πίστας 10
Εικόνα 8: Εχθροί 11
Εικόνα 9: Παγίδα κυνηγιού 12
Εικόνα 10: Σταλακτίτες 12
Εικόνα 11: Λάβα 17
Εικόνα 12: Συλλεκτικά αντικείμενα 18
Εικόνα 13: NPC αγρότης 18
Εικόνα 14: Κίνηση της κάμερας 19
Εικόνα 15: Κίνηση του παίκτη μέρος 1 19
Εικόνα 16: Κίνηση του παίκτη μέρος 2 20
Εικόνα 17: Δημιουργία στιγμιότυπου 20
Εικόνα 18: Μετατόπιση αντικειμένου 21
Εικόνα 19: Ορισμός της μπάρας ζωής 21
Εικόνα 20: Ανίχνευση σύγκρουσης με τον εχθρό 22
Εικόνα 21: Μεταβλητή TouchLava 22
Εικόνα 22: Συνθήκη επαφής με λάβα 22
Εικόνα 23: Game Over 18
Εικόνα 24: Κουμπί New Game 19
Εικόνα 25: Κουμπί Exit Game 19
Εικόνα 26: Εμφάνιση χειρισμών 25
Εικόνα 27: Μενού παύσης 25
Εικόνα 28: Δημιουργία μπανάνας 26
Εικόνα 29: Μετατόπιση μπανάνας 27
Εικόνα 30: Παράδειγμα σημείου 1 και 2 27
Εικόνα 31: Κώδικας σκελετού μέρος 1ο 28
Εικόνα 32: Κώδικας σκελετού μέρος 2ο 28
Εικόνα 33: Κώδικας σκελετού μέρος 3ο 29
Εικόνα 34: Κώδικας σκελετού μέρος 4ο 29
Εικόνα 35: Όπλα του βασικού εχθρού μας 30
Εικόνα 36: Ρίψη φίλτρων 31
Εικόνα 37: Ιπτάμενες αράχνες 32
Εικόνα 38: Εκτόξευση βόμβας 32
Εικόνα 39: Παγίδα κυνηγιού 28
Εικόνα 40: Σταλακτίτες 28
Εικόνα 41: Πτώση κύβων 29
Εικόνα 42: Ορισμός πατέρα αντικειμένου 29
Εικόνα 43: Διάλογος πρωταγωνιστών 30
Εικόνα 44: Ρουτίνα εμφάνισης προτάσεων 30
Εικόνα 45: Επιλογή επόμενης πρότασης 31

Εικόνα 46: Συλλογή συλλεκτικών αντικειμένων 32
Εικόνα 47: Αποθήκευση του σκορ 32
Εικόνα 48: Μυστικός μοχλός 33
Εικόνα 49: Κώδικας λειτουργίας μοχλού 33
Εικόνα 50: Κώδικας NPC χαρακτήρα 34
Εικόνα 51: Ηχητικά εφέ 34
Εικόνα 52: Κώδικας ενεργού αντικειμένου αποθήκευσης 35
Εικόνα 53: Αλλαγή θέσης παίκτη 35
Εικόνα 54: Αλλαγή σημείου επαναφοράς 35
Εικόνα 55: Αποθήκευση παιχνιδιού 36
Εικόνα 56: Φόρτωση παιχνιδιού 36
Εικόνα 57: Μετάβαση πίστας και αυτόματη αποθήκευση 37
Εικόνα 58: Εμφάνιση πάνελ νίκης 37
Εικόνα 59: Μήνυμα μη αποθηκευμένου παιχνιδιού 39
Εικόνα 60: Επιλογή φόρτωσης παιχνιδιού 38

1. Εισαγωγή

1.1 Περίληψη παιχνιδιού

Το παιχνίδι “Mr.JackCat” είναι ένα συναρπαστικό παιχνίδι πλατφόρμας σχεδιασμένο στο Unity και δίνει την αίσθηση ενός κλασσικού ρετρό παιχνιδιού δεκαετίας του 80. Είναι αποκλειστικά σχεδιασμένο για υπολογιστές περιλαμβάνοντας 2 ανατρεπτικές πίστες με αυξανόμενη δυσκολία και μια τελική μάχη με τον δυσκολότερο εχθρό.

Ο Πρωταγωνιστής μας είναι ένας γάτος, ο Mr.JackCat όπου μετά την απαγωγή της γυναίκας του από τον κακό επιστήμονα, “EvilFred”, είναι αποφασισμένος να την βρει και να την φέρει πίσω. Ξεκινάει, λοιπόν, το ταξίδι του στο οποίο έρχεται αντιμέτωπος με εχθρούς, εμπόδια μέχρι να αντιμετωπίσει τον βασικό εχθρό της ιστορίας μας. Θα μπορεί να συλλέξει διάφορα αντικείμενα όπως κέρματα και ψάρια, να πετάει αντικείμενα στους εχθρούς, αλλά ταυτόχρονα θα πρέπει να προσέχει στην εμφάνιση κρυφών εμποδίων που μπορούν να του μειώσουν την ζωή. Η ύπαρξη των σημείων αποθήκευσης κατά μήκος της πίστας βοηθούν τον παίκτη στην αποθήκευση της προόδου του σε περίπτωση που χάσει.

1.2 Κίνητρο για την δημιουργία του και στόχοι

Το κίνητρο για την εκπόνηση αυτής της πτυχιακής ήταν η αγάπη που είχα από μικρό παιδί για τα βιντεοπαιχνίδια, αλλά και τον τρόπο που αυτά δημιουργούνται. Αυτά έγιναν θεμέλιο βάζοντας ως μελλοντικούς μου στόχους την σχεδίαση ενός παιχνιδιού και προγραμματισμό του μέσω πηγαίου κώδικα χρησιμοποιώντας μια υψηλή γλώσσα προγραμματισμού. Έτσι, η παρούσα πτυχιακή εργασία στοχεύει στην δημιουργία ενός δομημένου παιχνιδιού υψηλού επιπέδου που θα δώσει στον παίκτη την αίσθηση ότι παίζει κάποιο παιχνίδι μεγάλης εταιρίας.

1.3 Δομή των κεφαλαίων

Κάθε κεφάλαιο παρουσιάζει αναλυτικά τον τρόπο με τον οποίο σχεδιάστηκε το παιχνίδι και πώς λειτουργεί ο κώδικας μέσα από περιγραφές και εικόνες. Αρχικά, παρουσιάζονται τα προγράμματα και τα βασικά εργαλεία που με βοήθησαν στην ολοκλήρωση του έργου όπως και μια εισαγωγή στην έννοια των μηχανών πεπερασμένων καταστάσεων. Στη συνέχεια, περιγράφεται η διαδικασία δημιουργίας των χαρακτήρων, των εχθρών, της πίστας εξηγώντας την λογική πίσω από τον σχεδιασμό τους. Αναφέρονται τα διάφορα επίπεδα αυξανόμενης δυσκολίας που θα αντιμετωπίσει ο παίκτης στην διάρκεια της πίστας όσο αφορά τους εχθρούς και τα εμπόδια, αλλά και τα διάφορα συλλεκτικά αντικείμενα που μπορεί να συλλέξει. Τέλος, επεξηγείτε το λειτουργικό κομμάτι του παιχνιδιού που αφορά τον προγραμματισμό σε υψηλή γλώσσα προγραμματισμού. Αναλύεται η λογική του πηγαίου κώδικα και πώς αυτός συμβάλλει, ώστε να μπορεί να λειτουργήσει σωστά η κίνηση του παίκτη, των εχθρών και γενικά η λειτουργικότητα του παιχνιδιού.

2. Τεχνολογίες και προγράμματα που χρησιμοποιήθηκαν

2.1 Βασικό πρόγραμμα υλοποίησης παιχνιδιού

Για την δημιουργία του, χρησιμοποιήθηκε το **Unity 2021.3.20f1** , μια από τις δημοφιλέστερες πλατφόρμες σχεδίασης παιχνιδιών. Το **Unity Game Engine** είναι ευρέως γνωστό για την ευκολία που παρέχει σε έναν νέο προγραμματιστή, αλλά και σε οποιονδήποτε χρήστη, να δημιουργήσει ένα ποιοτικό παιχνίδι μόνο με λίγες γραμμές κώδικα. Φιλοξενεί την γλώσσα προγραμματισμού C# που είναι υπεύθυνη για την λειτουργικότητα και τον χειρισμό διάφορων μηχανισμών. Με το πέρασμα του χρόνου η πλατφόρμα όλο και διευρύνει την συμβατότητα με περισσότερες συσκευές όπως κονσόλες, κινητά τηλέφωνα και υπολογιστές κερδίζοντας έτσι περισσότερο κοινό.

2.2 Δευτερεύοντα προγράμματα υλοποίησης παιχνιδιού

Το πρόγραμμα που με βοήθησε στην δημιουργία και επεξεργασία των χαρακτήρων, των εχθρών, αλλά και των διάφορων αντικειμένων ήταν το **Aseprite**. Το Aseprite είναι ένα ιδιόκτητο λογισμικό επεξεργασίας εικονοστοιχείων που υποστηρίζεται αποκλειστικά σε υπολογιστές. Σου δίνει την δυνατότητα σχεδίασης συναρπαστικών animations, πρωτότυπων χαρακτήρων και γενικά γραφικών για χρήση σε βιντεοπαιχνίδια.

Άλλο λογισμικό που έχει συμβάλει στην ολοκλήρωση της πτυχιακής είναι το **Gimp**. Το Gimp είναι ένα ελεύθερο λογισμικό βασιζόμενο στην επεξεργασία raster γραφικών, στην μετατροπή εικόνων σε διαφορετικές μορφές και στην γενικότερη διαχείριση τους. Με βοήθησε στην διόρθωση ατελειών στα φόντα του παιχνιδιού, αλλά και στην αλλαγή χρωμάτων σε ορισμένες εικόνες.

Τέλος, για την επεξεργασία των ήχων και της μουσικής του παιχνιδιού χρησιμοποιήθηκε το ελεύθερο λογισμικό **Audacity**. Το Audacity είναι ένα δωρεάν πρόγραμμα για επαγγελματική διαχείριση αρχείων ήχου που ο χρήστης μπορεί να εφαρμόσει διάφορα φίλτρα, να αφαιρέσει πιθανούς θορύβους και άλλες πολλές δυνατότητες.

2.3 Σε τι χρησιμεύει η μηχανή πεπερασμένων καταστάσεων

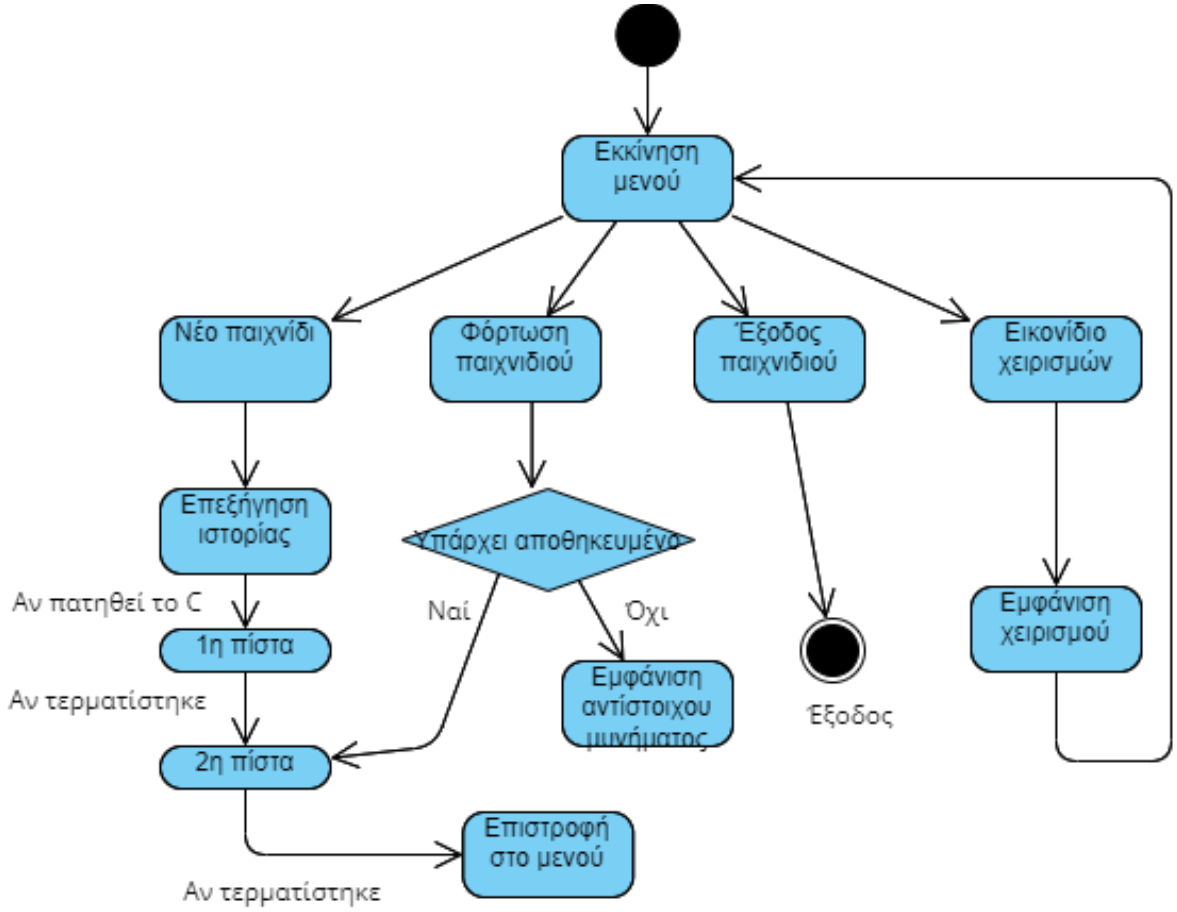
Στην συγκεκριμένη πτυχιακή εργασία η χρήση των μηχανών πεπερασμένων καταστάσεων ήταν κομβική για την σωστή λειτουργία του παιχνιδιού. Παρουσιάστηκαν δυσκολίες στον τρόπο με τον οποίο θα λειτουργεί ο κώδικας σε κάθε περίσταση, αναζητώντας μια λύση μοντελοποίησης του. Έτσι, αυτή η τεχνική χρησιμεύει στην μοντελοποίηση ενός προβλήματος με την γραφική αναπαράσταση των λογικών μεταβάσεων από μια κατάσταση σε μια άλλη. Στην συνέχεια θα παρουσιαστούν κάποια γραφήματα των μηχανών πεπερασμένων καταστάσεων για συγκεκριμένες λειτουργίες του παιχνιδιού.

3. Σχεδίαση βασικών χαρακτήρων και χαρτών

3.1 Βασικό μενού και μενού παύσης

3.1.1 Μηχανή πεπερασμένων καταστάσεων των μενού

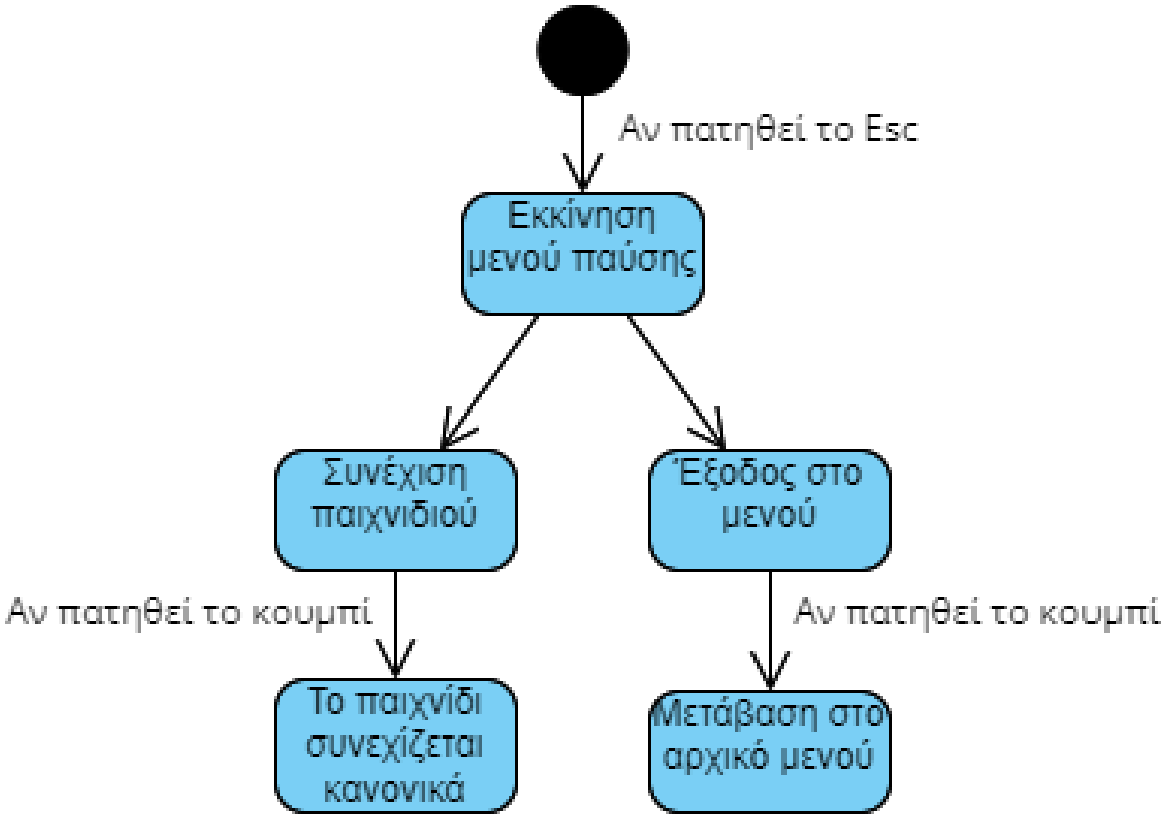
Για την λειτουργικότητα του μενού σχεδιάστηκαν 2 διαγράμματα καταστάσεων, μέσω της διαδικτυακής πλατφόρμας Visual Paradigm, που δείχνουν τις μεταβιβάσεις από μια κατάσταση σε μια άλλη. Αρχικά, παρουσιάζεται το διάγραμμα για το βασικό μενού υποδεικνύοντας τι θα συμβεί αν ο παίκτης πατήσει το αντίστοιχο κουμπί.



Εικόνα 1: Διάγραμμα καταστάσεων του βασικού μενού

Οι επιλογές στο κύριο μενού είναι τέσσερις. Ο χρήστης μπορεί να πατήσει “Νέο παιχνίδι” για να αρχίσει από την 1η πίστα και μετά αντίστοιχα την 2η πίστα. Όταν τερματίσει το παιχνίδι, επιστρέφεται πάλι στο κύριο μενού. Η επιλογή “Φόρτωση παιχνιδιού” είναι ενεργοποιημένη σε μια συγκεκριμένη συνθήκη. Όταν ο παίκτης τερματίσει την 1η πίστα, αλλά όχι την 2η πίστα και πραγματοποιήσει έξοδο από τον παιχνίδι, τότε γίνεται αυτόματη αποθήκευση. Εφόσον ισχύει αυτή η συνθήκη, πατώντας το κουμπί, σε μεταβιβάζει στην 2η πίστα. Αν δεν ισχύει η συνθήκη τότε εμφανίζεται ένα αντίστοιχο μήνυμα.

Στην περίπτωση που ο παίκτης θέλει να σταματήσει προσωρινά το παιχνίδι για οποιοδήποτε λόγο, τότε μπορεί να πατήσει το Escape κουμπί κατά την διάρκεια της πίστας. Πατώντας το θα εμφανιστεί το μενού παύσης και ο χρήστης επιλέγει το αντίστοιχο κουμπί. Έχει την επιλογή της “συνέχισης του παιχνιδιού” όταν είναι έτοιμος να συνεχίσει την πίστα, αλλά και την επιλογή της εξόδου στο κύριο μενού. Παρακάτω φαίνεται το αντίστοιχο διάγραμμα για το μενού παύσης.



Εικόνα 2: Διάγραμμα καταστάσεων του μενού παύσης

3.1.2 Μενού UI

Η κατασκευή των μενού έγινε μέσα από το Unity με την δημιουργία ενός βασικού καμβά μέσα στον οποίο πρόσθεσα τα αντίστοιχα κουμπιά με τα οποία θα μπορεί να έρθει σε επαφή ο παίκτης. Σε κάθε ένα από αυτά έχει προστεθεί μια συγκεκριμένη ειδική γραμματοσειρά και ποίκιλες αποχρώσεις χρωμάτων, ώστε το περιβάλλον να θυμίζει ένα κλασσικό ρετρό παιχνίδι. Επίσης, εκτός από τα μενού, δημιουργήθηκαν και άλλες διεπαφές χρήστη μέσω του Unity όπως είναι ο τερματισμός του παιχνιδιού, η ένδειξη του σκορ, οι διάλογοι με άλλους χαρακτήρες κ.α. Παρακάτω παραθέτω μερικές εικόνες των μενού και άλλων διεπαφών του χρήστη.

Εικόνα 3: Βασικό μενού

Εικόνα 4: Μενού παύσης

Εικόνα 5: Τέλος παιχνιδιού

Εικόνα 6: Τερματισμός παιχνιδιού

3.2 Σχεδίαση βασικού παίκτη και Animations

Η σχεδίαση του παίκτη έγινε μέσα από το πρόγραμμα επεξεργασίας εικονοστοιχείων Aseprite όπως και τα animations. Για αυτόν τον χαρακτήρα δημιουργήθηκαν τα περισσότερα animations με τον αριθμό να φτάνει στα οκτώ συνολικά, προσπαθώντας έτσι να κάνω το παιχνίδι πιο ελκυστικό στο μάτι του θεατή και πιο ευχάριστο στον χειρισμό. Συγκεκριμένα στα animations περιλαμβάνονται το περπάτημα, το άλμα, η ρίψη αντικειμένων, το χτύπημα από τον εχθρό, η αδράνεια κ.α.

3.2.1 Διεπαφή χρήστη του παίκτη

Κατά την διάρκεια του παιχνιδιού ο χρήστης θα μπορεί να παρακολουθεί τα στατιστικά νούμερα των αντικειμένων που συλλέγει όσο αυτός παίζει την πίστα, αλλά και το ποσοστό της ζωής που του απομένει. Γι' αυτό το σκοπό έχει τοποθετηθεί μια μπάρα ζωής στην πάνω αριστερά πλευρά της οθόνης αναγράφοντας το αντίστοιχο ποσοστό. Στην πάνω δεξιά πλευρά αναγράφεται ο αριθμός των ψαριών που έχουν συλλεχθεί.

3.3 Υλοποίηση πίστας

Για το σχεδιασμό της πίστας χρησιμοποιήθηκε το δωρεάν tileset πακέτο του Unity, Sunny Land. Το πακέτο αυτό παρέχει στον προγραμματιστή όλα τα εφόδια που του επιτρέπουν να χτίσει μια ολοκληρωμένη, όμορφη πίστα. Σαν πακέτο περιέχει αμέτρητους κύβους, διακοσμητικά αντικείμενα, εχθρούς, χαρακτήρες και διάφορα animations. Σε αυτήν την πτυχιακή εργασία χρησιμοποιήθηκαν μόνο κάποια φόντα, κύβοι και αντικείμενα διακόσμησης από αυτό το πακέτο. Γενικά, και για τις δύο πίστες μετά την κατασκευή τους, εφαρμόστηκαν οι μηχανισμοί ανίχνευσης σύγκρουσης μέσω Unity για τους κύβους και το έδαφος, καθώς δεν υπάρχουν αυτόματα στο συγκεκριμένο πακέτο tileset.

Η **1η πίστα** είναι μια κλασσική πίστα πλατφόρμας στην οποία ο βασικός στόχος είναι η μετακίνηση του παίκτη στον χώρο πραγματοποιώντας άλματα και βήματα μέχρι να τερματίσει την πίστα. Έχουν τοποθετηθεί, βάση των αξόνων x και y, διάφορα εμπόδια και εχθροί που ο παίκτης πρέπει να αποφύγει ή να εξολοθρεύσει. Επίσης, στο δημιουργημένο περιβάλλον, ο χρήστης μπορεί να συλλέξει αντικείμενα, κέρματα και ειδικά διαμάντια που πραγματοποιούν αυτόματη αποθήκευση στο ορισμένο σημείο. Η **2η και τελική πίστα** είναι μια αρένα μάχης μεταξύ του χαρακτήρα μας και του βασικού εχθρού. Η ύπαρξη εμποδίων και απρόβλεπτων παγίδων που εμφανίζονται απροσδόκητα, αυξάνουν κατά πολύ το επίπεδο δυσκολίας του παιχνιδιού όπως είναι λογικό. Σε σχέση με την 1η πίστα, στην 2η δεν υπάρχει μετακίνηση της κάμερας οπότε οι χαρακτήρες κινούνται στο προβλεπόμενο πεδίο της οθόνης.

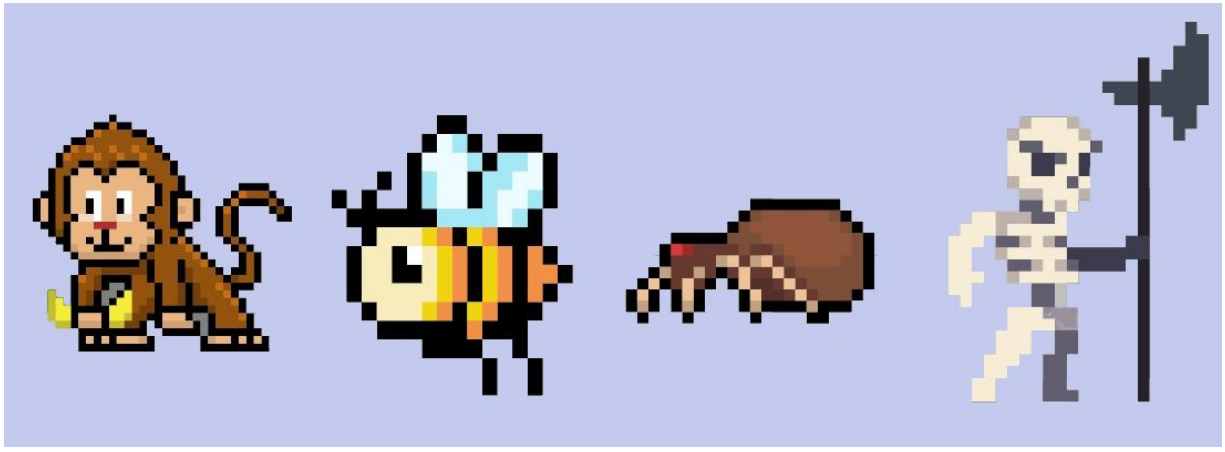
Εικόνα 7: Περιβάλλον 2ης πίστας

3.4 Εχθροί και εμπόδια

3.4.1 Εχθροί

Όπως σε κάθε παιχνίδι, έτσι και σε αυτό δεν θα μπορούσαν να λείπουν οι εχθροί που αδιαμφησβήτητα είναι το σημαντικότερο κομμάτι μετά τον πρωταγωνιστή μας, ειδικά σε ένα παιχνίδι πλατφόρμας. Δημιουργήθηκαν αρκετοί, κυρίως για την 1η πίστα με τον αριθμό να φτάνει τους πέντε εχθρούς. Το πρόγραμμα δημιουργίας τους και των animations τους ήταν το Aseprite. Ανάμεσα τους είναι οι αράχνες, οι ιπτάμενες αράχνες, οι σκελετοί, οι μέλισσες, οι μαϊμούδες με τον κάθε εχθρό να έχει το δικό του επίπεδο δυσκολίας. Κάθε ένας από αυτούς προστέθηκε στην αντίστοιχη πίστα αναλόγως το περιβάλλον που διαδραματίζεται. Οι μαϊμούδες και οι μέλισσες τοποθετήθηκαν στην ζούγκλα για τον λόγο ότι κάποιος τις συναντάει εκεί, ενώ οι σκελετοί και οι αράχνες βρίσκονται σε σπηλιές για τον ίδιο λόγο.

Οι **μαϊμούδες** είναι ο πρώτος εχθρός της πίστας με το μικρότερο επίπεδο δυσκολίας. Βασικό όπλο τους είναι η ρίψη μπανάνας στον παίκτη με αποτέλεσμα αν ανιχνευτεί σύγκρουση με αυτόν, μειώνεται το ποσοστό ζωής του κατά 10%. Αντίστοιχα, αν η μαϊμού χτυπηθεί από τον παίκτη, πρέπει αυτός να την χτυπήσει δύο φορές για να την εξολοθρεύσει. Οι **μέλισσες** είναι ο δεύτερος εχθρός, αλλά και ο πρώτος που κάνει περιοδική κίνηση από ένα σημείο σε ένα άλλο. Ουσιαστικά, μετακινείται πετώντας στον χώρο με τον παίκτη να πρέπει απλά να μην έρθει σε επαφή μαζί της. Το επίπεδο δυσκολίας αυξάνεται λίγο παραπάνω με την μείωση της ζωής του παίκτη να φτάνει στο 20%.



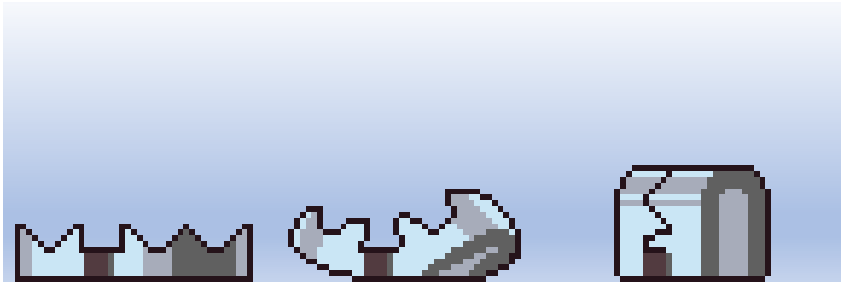
Εικόνα 8: Εχθροί

Μετά ακολουθούν οι **σκελετοί** οι οποίοι κατατάσσονται προτελευταίοι στο επίπεδο δυσκολίας. Αυτοί όπως και οι αράχνες, κατοικούν σε σκοτεινά μέρη όπως σπηλιές, κάστρα, υπόγεια. Συγκεκριμένα, οι σκελετοί κινούνται στον χώρο κρατώντας μια αξίνα και επιβλέπουν ποιος τους πλησιάζει. Αν ο παίκτης πλησιάσει αρκετά κοντά, τότε ο σκελετός σταματάει την μετατόπιση και αρχίζει να χτυπάει με την αξίνα. Αν ανιχνευθεί σύγκρουση μεταξύ των δύο, η ζωή του παίκτη θα μειωθεί κατά 25% της συνολικής ζωής. Αναλόγως, για να εξολοθρευθεί ο σκελετός, πρέπει ο παίκτης να τον χτυπήσει 3 φορές.

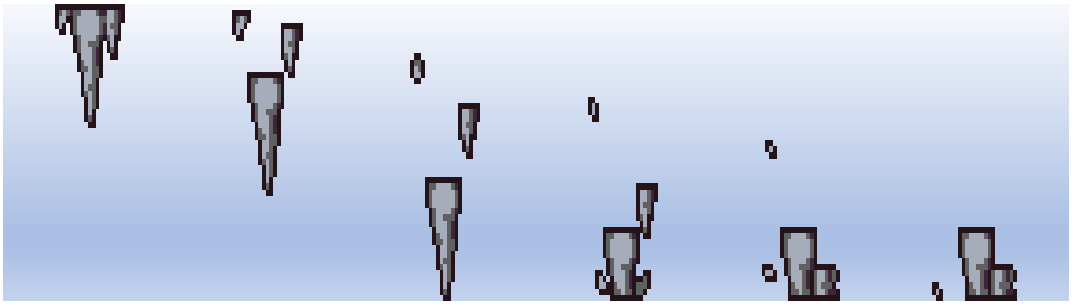
Ο τελευταίος και πιο δύσκολος εχθρός του παιχνιδιού είναι οι **αράχνες** με το υψηλότερο επίπεδο δυσκολίας. Οι αράχνες χωρίζονται σε 2 κατηγορίες, δηλαδή στις ιπτάμενες με ιστό και σε αυτές που κινούνται στο έδαφος χωρίς όμως να υπάρχει διαφορά στο ποσοστό μείωση της ζωής του παίκτη. Αν αυτός έρθει σε επαφή μαζί τους, τότε θα αφαιρεθεί το 30% της ζωής του, ενώ η αράχνη αναλόγως χρειάζεται 3 χτυπήματα για την εξολόθρευση της.

3.4.2 Εμπόδια

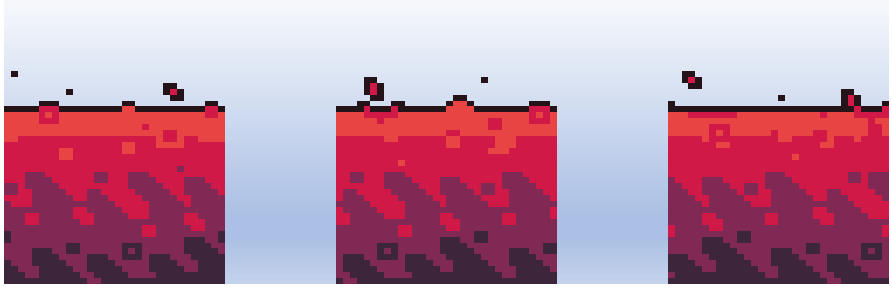
Από κανένα παιχνίδι πλατφόρμας δεν θα μπορούσαν να λείπουν τα εμπόδια και οι παγίδες που με αυτά γίνεται αυτομάτως πιο ευχάριστο και διαδραστικό προς τον παίκτη. Έτσι, και εδώ τοποθετήθηκαν σε όλο το εύρος της 1ης και 2ης πίστας διάφορες παγίδες και εμπόδια. Για αυτόν τον σκοπό αγοράστηκε το “Pixel Art Animated Traps” asset πακέτο το οποίο περιέχει μεγάλη ποικιλία από animated παγίδες. Συνολικά από το πακέτο, χρησιμοποιήθηκαν τέσσερα εμπόδια ανάμεσα τους η λάβα, οι σταλακτίτες, οι κοφτερές ακίδες, προσθέτοντας σε μερικά κίνηση, ενώ σε άλλα τοποθετήθηκαν απλά σαν εικόνες.



Εικόνα 9: Παγίδα κυνηγιού



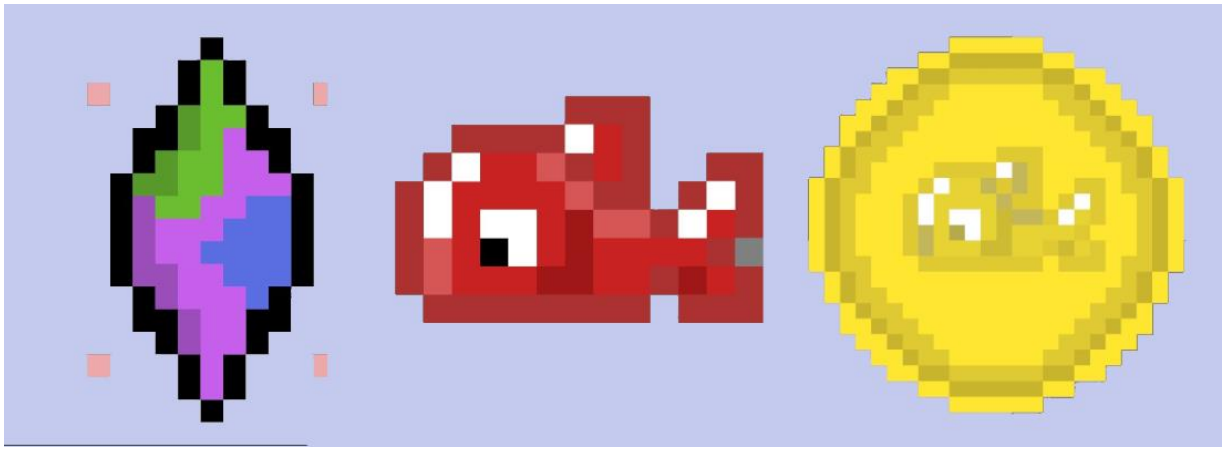
Εικόνα 10: Σταλακτίτες



Εικόνα 11: Λάβα

3.5 Συλλεκτικά αντικείμενα

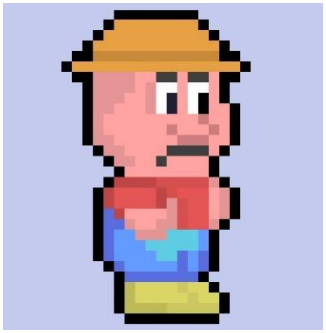
Η ύπαρξη των συλλεκτικών αντικειμένων σίγουρα είναι κάτι που μπορεί να αλλάξει όλη την δυναμική ενός παιχνιδιού κρατώντας τους παίκτες απασχολημένους κατά την διάρκεια της πίστας. Το παιχνίδι αυτομάτως γίνεται πιο ανταγωνιστικό με τον παίκτη να κυνηγάει το ανώτερο αριθμό συλλογής αντικειμένων θεωρώντας το κάτι σαν επιβράβευση. Εδώ προστέθηκαν τέσσερα συλλεκτικά αντικείμενα όπως τα ψάρια, τα χρυσά ψάρια και το πολύχρωμο διαμάντι. Για κάθε ένα ψάρι που συλλέγει, ο παίκτης κερδίζει ένα βαθμό και αναλόγως διαφοροποιείται το σκορ στην οθόνη. Αν μαζέψει το χρυσό ψάρι, το οποίο έχει την μορφή κέρματος, κερδίζει δέκα βαθμούς. Τέλος, το πολύχρωμο διαμάντι αποτελεί ένα προσωρινό σημείο αποθήκευσης για όσο διάστημα είναι ανοιχτή η συγκεκριμένη πίστα. Στην περίπτωση που ο πρωταγωνιστής μας χάσει, αλλά έχει συλλέξει το αντίστοιχο διαμάντι, η θέση του επαναφέρεται εκεί.



Εικόνα 12: Συλλεκτικά αντικείμενα

3.6 Τι είναι NPC παίκτες και πώς χρησιμοποιήθηκαν

Ένας NPC χαρακτήρας είναι ο οποιοσδήποτε χαρακτήρας σε ένα παιχνίδι που δεν μπορεί να κινείται από τον χρήστη. Είναι απλά ένα μέρος της σχεδίασης της πίστας με αυτό να σημαίνει ότι κατέχει τον ρόλο ενός κομπάρσου. Ο χειρισμός του γίνεται αποκλειστικά από τον υπολογιστή μέσα από ένα καθορισμένο σετ εντολών που πιθανότατα να επηρεάσουν την ιστορίας του παιχνιδιού. Ο NPC παίκτης που σχεδιάστηκε για το παιχνίδι είναι ένας αγρότης με αυτόν να δίνει συμβουλές στον παίκτη μας στην διάρκεια της πίστας.



Εικόνα 13: NPC αγρότης

4. Υλοποίηση του κώδικα

4.1 Κίνηση της κάμερας

Για την μετακίνηση της κάμερας στον χώρο, δηλώνεται ένα τρισδιάστατο διάνυσμα με τους άξονες x και y, να είναι η θέση του παίκτη. Για τον άξονα y, ορίστηκε ένα offset που μας επιτρέπει να προσθέσουμε έξτρα απόκλιση στην κάμερα αν χρειαστεί. Κάθε φορά, λοιπόν, που ο παίκτης προβεί σε κάποια μετατόπιση, αυτή ανιχνεύεται με την κάμερα να μετακινείται μαζί με τον παίκτη από την παλιά της θέση στην καινούργια μέσω της transform.position εντολής.

```
public class CameraFollow : MonoBehaviour{

    public float FollowSpeed = 2f;
    public float yOffset =10f;
    public Transform target;

    void Update()
    {
        Vector3 newPos = new Vector3(target.position.x,target.position.y + yOffset,-10f);
        transform.position = Vector3.Slerp(transform.position,newPos,FollowSpeed*Time.deltaTime);
    }

}
```

Εικόνα 14: Κίνηση της κάμερας

4.2 Παίκτης

4.2.1 Κίνηση του παίκτη

Η κίνηση του παίκτη πραγματοποιείται αν ο χρήστης πατήσει συγκεκριμένα πλήκτρα του πληκτρολογίου ενεργοποιώντας μετατοπίσεις και διάφορα animations. Αν πατηθούν το δεξί και αριστερό πλήκτρο ο παίκτης αντίστοιχα θα μετατοπιστεί οριζόντια με μια ορισμένη ταχύτητα. Έτσι, θα ξεκινήσει και το αντίστοιχο animation περπατήματος του παίκτη κάνοντας την τιμή του αληθές. Για πιο αρμονικό περπάτημα, το sprite θα περιστρέφεται 180 μοίρες στον άξονα y για να φαίνεται ότι περπατάει στην σωστή φορά. Αυτό επιτυγχάνεται με τον ορισμό της μεταβλητής “FacingRight” που ανιχνεύει την φορά του παίκτη, αλλά και από που θα περιστραφεί. Τέλος, όλες αυτές οι συνθήκες θα εκτελούνται μόνο αν ο παίκτης είναι ζωντανός.

Στην άλλη περίπτωση που πατηθεί το Space, ο παίκτης θα πηδήξει στον αέρα μέσω ενός δισδιάστατου διανύσματος. Στην συνθήκη αυτή η μεταβλητή “isJump” ανιχνεύει αν αυτός βρίσκεται στον αέρα ή στο έδαφος για να αρχίσει το animation. Άρα, αν ανιχνευθεί σύγκρουση με το έδαφος, η μεταβλητή αυτή παίρνει την τιμή ψευδές.

```
void OnCollisionEnter2D(Collision2D other){
    if(other.gameObject.CompareTag("Ground")){
        isJump=false;
    }
}
```

Εικόνα 15: Κίνηση του παίκτη μέρος 1

```
void Update(){
    if(health.c!=1 && health.currentHealth>=0){
    if(Input.GetKey(KeyCode.RightArrow)&&!FacingRight){
        FacingRight=!FacingRight;
        transform.Rotate(0,180,0);
    }
    if(Input.GetKey(KeyCode.LeftArrow)&&FacingRight){
        FacingRight=!FacingRight;
        transform.Rotate(0,180,0);
    }
    if(Input.GetKey(KeyCode.RightArrow)||Input.GetKey(KeyCode.LeftArrow)){
        anim.SetBool("isRunning",true);
        moveInput = Input.GetAxisRaw("Horizontal");
        rb.velocity = new Vector2(moveInput * speed, rb.velocity.y);
    }
    else{
        anim.SetBool("isRunning",false);
    }
    if(Input.GetKeyDown(KeyCode.Space)&&!isJump){
        anim.SetTrigger("jump");
        rb.velocity=Vector2.up *jumpForce;
        isJump=true;
    }
    }
}
```

Εικόνα 16: Κίνηση του παίκτη μέρος 2

4.2.2 Ρίψη αντικειμένων στους εχθρούς

Ο παίκτης εκτός από τις βασικές κινήσεις στον χάρτη, θα μπορεί να αμυνθεί από τις επιθέσεις των εχθρών πετώντας τους αντικείμενα. Δημιουργήθηκε, λοιπόν ένα prefab αντικείμενο μιας μπάλας σαν όπλο για το οποίο υλοποιήθηκαν δύο αρχεία. Το πρώτο αρχείο αναφέρει πώς αν πατηθεί το πλήκτρο X, θα δημιουργηθεί ένα στιγμιότυπο της μπάλας στη θέση που είναι ορισμένη βάση του ‘p’ GameObject. Επίσης, το ίδιο αρχείο αλλάζει προσωρινά το είδος της σύγκρουσης του παίκτη αν πετάξει μια μπάλα και για όσο διαρκεί η βολή αυτή. Αυτό συμβαίνει για να μην υπάρχει κάποια περίεργη μετατόπιση του παίκτη αν έρθει σε επαφή με την μπάλα.

```
void Start(){
    anim = GetComponent<Animator>();
}

void Update(){
    if(Input.GetKeyDown(KeyCode.X)){
        GetComponent<Rigidbody2D>().collisionDetectionMode = CollisionDetectionMode2D.Discrete;
        Instantiate(ball, p.position, Quaternion.identity);
        anim.SetTrigger("isThrowing");
        StartCoroutine(EnablePlayerCollisionsAfterDelay(0.3f));
    }

}
IEnumerator EnablePlayerCollisionsAfterDelay(float delay)
{
    yield return new WaitForSeconds(delay);
    GetComponent<Rigidbody2D>().collisionDetectionMode = CollisionDetectionMode2D.Continuous;
}
```

Εικόνα 17: Δημιουργία στιγμιότυπου

Στο δεύτερο αρχείο αν πατηθεί το ίδιο κουμπί, αρχικά σχηματίζεται μια ακτίνα την οποία θα ακολουθήσει το αντικείμενο μας καθορίζοντας ταυτόχρονα την απόσταση βολής του. Ύστερα, μέσω ενός δισδιάστατου διανύσματος, η μπάλα μας θα μετατοπιστεί προς τα δεξιά, έχοντας καταφέρει την ψευδαίσθηση της ρίψης. Τέλος, όταν η μπάλα τελικά ακουμπήσει το έδαφος ή τον εχθρό, τότε μόνο θα καταστραφεί.

```
void Start()
{
    rb=GetComponent<Rigidbody2D>();
    anim=GetComponent<Animator>();
    if(Input.GetKeyDown(KeyCode.X)){
        hit= Physics2D.Raycast(ball.transform.position,Vector2.right*ball.transform.localScale.x,distance);
        hit.collider.gameObject.GetComponent<Rigidbody2D>().velocity=new Vector2(ball.transform.localScale.x,1)*ThrowForce;
    }
}
 void Update(){
 }

 void OnCollisionEnter2D(Collision2D collision){
    if(collision.gameObject.name == "Mr.JackCat"){
    }
    else{
        Destroy(ball);
    }
 }
```

Εικόνα 18: Μετατόπιση αντικειμένου

4.2.3 Ζωή του παίκτη

Για την εμφάνιση της συνολικής ζωής του παίκτη χρησιμοποιήθηκε μια μπάρα στο πάνω μέρος της οθόνης. Υλοποιήθηκε με την χρήση ενός slider όπου μετακινείται αριστερά όταν αυτός χτυπηθεί από κάτι. Έχουμε ορίσει, λοιπόν, μια μέγιστη ακέραια τιμή για αυτόν τον slider που χαρακτηρίζει το ποσοστό ζωής του παίκτη και αναλόγως τα χτυπήματα που δέχεται, τότε μειώνεται αυτή βάση ενός ακεραίου.

```
public Slider slider;

public void SetMaxHealth(int health){
    slider.maxValue = health;
    slider.value = health;
}
public void SetHealth(int health){
    slider.value = health;
}
```

Εικόνα 19: Ορισμός της μπάρας ζωής

Άρα, αν ανιχνευθεί σύγκρουση με έναν από τους εχθρούς, όπως για παράδειγμα με την μέλισσα, τότε καλείται η παρακάτω συνθήκη. Μας αναφέρει πώς αν η ζωή του παίκτη έχει φτάσει το μηδέν, ενεργοποιείται το animation όπου πέφτει κάτω, αλλά ταυτόχρονα ενεργοποιείται ένας χρονοδιακόπτης που μετά την λήξη του, κάνει ενεργό το πάνελ του “Game Over”. Αντιθέτως, αν η ζωή του είναι μεγαλύτερη του μηδέν, θα αναπαραχθεί ένα animation χτυπήματος και η ζωή θα μειωθεί με τέτοιο τρόπο, ώστε να μην πέσει από το όριο του μηδενός σε κάποια αρνητική τιμή.

```
if(other.gameObject.CompareTag("Bee")){
    if(currentHealth<=0){
      c=1;
      anim.SetTrigger("isAlive");
      Invoke("deathPanel",2);
    }
 if(currentHealth>0){
    anim.SetTrigger("hit");
    anim.SetBool("isHitted",false);
    if(20>currentHealth){
      hitted_Sound.PlayOneShot(hitted,1);
      dam=currentHealth;
      TakeDamage(dam);
    }
    else if(20<=currentHealth){
      hitted_Sound.PlayOneShot(hitted,1);
      TakeDamage(20);
    }
   }
}
```

Εικόνα 20: Ανίχνευση σύγκρουσης με τον εχθρό

Ανάμεσα σε όλα τα αντικείμενα που ανιχνεύεται σύγκρουση με τον παίκτη, ένα εμπόδιο έχει προγραμματιστεί διαφορετικά και αυτό είναι η λάβα. Στα εμπόδια, που βρίσκονται στην δεύτερη πίστα μόνο, έχει προστεθεί μια έξτρα συνθήκη με μια μεταβλητή αληθείας, την "TouchLava".

```
if(other.gameObject.name == "poison(Clone)" && TouchLava==false){
    if(currentHealth<=0){
      c=1;
      anim.SetTrigger("isAlive");
      Invoke("deathPanel",2);
    }
```

Εικόνα 21: Μεταβλητή TouchLava

Όσο αυτή είναι ψευδής, σημαίνει ότι δεν έχει έρθει σε επαφή με την λάβα, άρα μπορεί να ανιχνευθεί οποιαδήποτε άλλη σύγκρουση. Όμως, αν ανιχνευθεί σύγκρουση με την λάβα, αμέσως μετατρέπεται σε αληθές. Έτσι, τρέχει το αντίστοιχο animation, μειώνεται η ζωή του παίκτη στο εκατό τοις εκατό και εμφανίζεται πάλι το πάνελ "Game Over" μετά από δύο δευτερόλεπτα.

```
if(other.gameObject.CompareTag("Lava")){
      c=1;
      TouchLava=true;
      anim.SetTrigger("Burn");
      Invoke("deathPanel",2);
      TakeDamage(100);
```

Εικόνα 22: Συνθήκη επαφής με λάβα

4.2.4 Τέλος παιχνιδιού

Σε αυτή την ενότητα θα αναλύσουμε τον τρόπο με τον οποίο τερματίζεται το παιχνίδι όταν ο παίκτης χάσει από φυσικά αίτια όπως η πτώση έξω από το πεδίο της πίστας μας και πώς γίνεται επανεκκίνηση στο τελευταίο σημείο αποθήκευσης. Ο παρακάτω κώδικας μας λέει πώς αν η θέση του παίκτη είναι εκτός των ορίων της πίστας, τότε ο παίκτη έχει χάσει άρα θα εμφανιστεί το πάνελ "GameOver", θα σταματήσει ο χρόνος και θα παίξει ένα αντίστοιχο αρχείο ήχου. Αν πατηθεί το Space, ενώ είναι ενεργό το πάνελ, επαναφέρει την πίστα στο τελευταίο σημείο αποθήκευσης. Επίσης, μια σημαντική μεταβλητή στον παρακάτω κώδικα είναι η Sound που χρησιμοποιείται για να μην ακουστεί το αρχείο ήχου πολλαπλές φορές από την στιγμή αναπαραγωγής του.

```
void Update(){
    if(playerview.transform.position.y<-2.7){
        if(Sound==false){
        GameOver_Sound.PlayOneShot(gameOver,0.5F);
        GameOverUI.SetActive(true);
        Time.timeScale = 0f;
        Sound=true;
        }
    if(Input.GetKeyDown(KeyCode.Space)&&GameOverUI.activeSelf){
        Sound=false;
        GameOverUI.SetActive(false);
        Time.timeScale = 1f;
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);
    }
    }
}
```

Εικόνα 23: Game Over

4.3 Μενού

4.3.1 Βασικό μενού

Η υλοποίηση του μενού ίσως αποτελεί το σημαντικότερο κομμάτι του παιχνιδιού, ώστε αυτό να λειτουργεί σωστά και να φαίνεται σωστά δομημένο. Αρχικά, δημιουργήθηκαν τέσσερα κουμπιά της διεπαφής χρήστη του Unity. Στα τρία από αυτά έχει εφαρμοστεί μηχανισμός ανίχνευσης του κλικ του ποντικού. Αυτό σημαίνει πώς όταν ο χρήστης πατήσει ένα κουμπί, θα εκτελεστεί κάποια λειτουργία. Το πρώτο κουμπί απλά αρχίζει την επεξήγηση της ιστορίας μας μέσα από εικόνες και ξεκινάει από την πρώτη πίστα. Σε αυτό το κουμπί έχει συνδεθεί η συνάρτηση "PlayGame" άρα θα αρχίσει να επεξηγεί την ιστορία του παιχνιδιού κάνοντας ενεργά διάφορα πάνελ.

Ο παίκτης πατώντας κάθε φορά το C θα μεταβιβάζεται στην επόμενη εικόνα της ιστορίας κάνοντας την ενεργή και αντίστοιχα η προηγούμενη γίνεται ανενεργή. Η μεταβλητή next και η χρήση της συνάρτησης "NextImage", λύνουν ένα μικρό σφάλμα που παρουσιάστηκε στον κώδικα. Βοηθούν στην υπερβολικά γρήγορη μεταβίβαση των εικόνων πατώντας το πλήκτρο C παραλείποντας μερικές φορές κάποιες ενδιάμεσες εικόνες. Άρα, η συνάρτηση προσθέτει μια μικρή χρονοκαθυστέρηση ανάμεσα στα πατήματα. Τέλος, όταν φτάσουμε πια στο τελευταίο πάνελ και πιέσουμε το C, απλά τρέχει η πρώτη σκηνή και αρχίζει η πίστα.

```
void Update(){
    if(Input.GetKeyDown(KeyCode.C) && story_1.activeSelf && next==0){
        next=1;
        story_1.SetActive(false);
        story_2.SetActive(true);
        Invoke("NextImage",1);
    }
    if(Input.GetKeyDown(KeyCode.C) && story_2.activeSelf && next==0){
        next=1;
        story_2.SetActive(false);
        story_3.SetActive(true);
        Invoke("NextImage",1);
  }
  if(Input.GetKeyDown(KeyCode.C) && story_3.activeSelf && next==0){
        SceneManager.LoadScene("Level_1");
  }
}

public void PlayGame(){
    story_1.SetActive(true);
}
```

Εικόνα 24: Κουμπί New Game

Το επόμενο κουμπί είναι το "Load Game" που αν ο χρήστης το πατήσει, φορτώνεται αποθηκευμένη πρόοδος του παίκτη. Συγκεκριμένα, φορτώνεται η δεύτερη πίστα αν ο παίκτης έχει τερματίσει την 1η, αλλά αποχώρησε από το παιχνίδι για οποιοδήποτε λόγο. Το τρίτο κουμπί είναι το "Exit Game" και με αυτό γίνεται προφανώς έξοδος από το παιχνίδι και η συνάρτηση "QuitGame" έχει συνδεθεί σε αυτό.

```
public void QuitGame(){
    Application.Quit();
    Debug.Log("Application has quit.");
}
```

Εικόνα 25: Κουμπί Exit Game

Το τελευταίο κουμπί απλά βοηθάει τον χρήστη να μάθει τον χειρισμό του παιχνιδιού και ποια πλήκτρα πρέπει να πατήσει για να προβεί στις αντίστοιχες κινήσεις. Στον κώδικα έχει δηλωθεί ένας μετρητής ο οποίος αν πατηθεί το κουμπί, αυξάνεται κατά ένα. Ύστερα, έχουν δημιουργηθεί δύο συνθήκες που αναφέρουν πώς αν ο μετρητής είναι ζυγός ή περιττός αριθμός, τότε θα εμφανίζει την εικόνα με τους χειρισμούς ή θα την εξαφανίζει αντίστοιχα.

```
public void Controls_appear(){
    f=f+1;
    if(f%2!=0){
        controls.SetActive(true);
    }
    else if(f%2==0){
        controls.SetActive(false);
    }
}
```

Εικόνα 26: Εμφάνιση χειρισμών

4.3.2 Μενού παύσης

Το μενού παύσης υλοποιήθηκε με τον ίδιο περίπου τρόπο όπως το βασικό μας μενού χρησιμοποιώντας κουμπιά και στα οποία συνδέονται συγκεκριμένες συναρτήσεις. Τα δύο κουμπιά μας αποτελούνται από το Resume και το Exit Game. Το πρώτο επιστρέφει τον παίκτη στην πίστας μας, ενώ η δεύτερη, μας οδηγεί στο βασικό μενού. Ο συνολικός μηχανισμός, δηλαδή η εμφάνιση του μενού παύσης, ενεργοποιείται με το πάτημα του Escape πλήκτρου. Όταν πατηθεί ελέγχεται αν το παιχνίδι είναι ήδη σταματημένο ή όχι. Αν δεν είναι, τρέχει η συνάρτηση "Pause". Αυτή σταματάει τον χρόνο, την μουσική και εμφανίζεται το μενού παύσης. Από την άλλη αν είναι ήδη σταματημένο, εκτελείται η συνάρτηση "Resume" η οποία συνεχίζει την εκτέλεση του παιχνιδιού.

```
void Update()
{
    if(Input.GetKeyDown(KeyCode.Escape)){
        if(GameIsPaused){
            Resume();
        }
        else{
            Level_1_music.Stop();
            Pause();
        }
    }
public void Resume(){
    Level_1_music.PlayOneShot(level_1,0.5F);
    pauseMenuUI.SetActive(false);
    Time.timeScale = 1f;
    GameIsPaused = false;
}
public void Pause(){
    pauseMenuUI.SetActive(true);
    Time.timeScale = 0f;
    GameIsPaused = true;
}
public void QuitGame(){
    Time.timeScale = 1f;
    SceneManager.LoadScene("StartMenu");
}
```

Εικόνα 27: Μενού παύσης

4.4 Εχθροί

4.4.1 Μαϊμού

Ο πρώτος εχθρός που συναντάμε στην πίστα είναι η μαϊμού και αποτελεί θεωρητικά τον πιο εύκολο χαρακτήρα για να εξολοθρεύσουμε. Η επίθεση που πραγματοποιεί προς τον παίκτη είναι το πέταγμα μπανάνας ανιχνεύοντας πρώτα την ακριβείς θέση του και αμέσως μετά την εκτοξεύει στην κατεύθυνση του παίκτη. Έχουν δημιουργηθεί δύο αρχεία για την λειτουργία της μαϊμούς.

Στο πρώτο αρχείο υπάρχει ένας χρονοδιακόπτης ο οποίος από την στιγμή εκκίνησης του, ελέγχεται συνεχώς η τιμή του. Αν περάσουν τρία δευτερόλεπτα και η θέση του παίκτη είναι αρκετά κοντά στην μαϊμού, τότε αναπαράγεται το animation της μαϊμούς, δημιουργείται ένα στιγμιότυπο της μπανάνας και ταυτόχρονα μηδενίζει ο χρονοδιακόπτης. Επίσης, στο ίδιο αρχείο έχει καθοριστεί η αντίδραση της μαϊμούς στην περίπτωση σύγκρουσης από την μπάλα του παίκτη. Με την χρήση ενός μετρητή, αν η μαϊμού χτυπηθεί μια φορά, τότε αναπαράγεται ανάλογο animation, ενώ αν χτυπηθεί δύο φορές, τότε αυτή καταστρέφεται.

```
void Update()
{
    timer+=Time.deltaTime;
    if(timer > 3 && player.transform.position.x>min&&player.transform.position.x<max){
        timer = 0;
        anim.SetTrigger("isThrowing");
        anim.SetTrigger("isIdle");
        Instantiate(banana,bananaPos.position,Quaternion.identity);
    }
}
void OnCollisionEnter2D(Collision2D collision){
    if(collision.gameObject.name == "ball(Clone)"){
        if(hits==2){
            Destroy(monk);
        }
        anim.SetTrigger("isHitted");
        anim.SetTrigger("isIdle");
        hits=hits+1;
    }
}
```

Εικόνα 28: Δημιουργία μπανάνας

Αμέσως μετά την δημιουργία του πρώτου στιγμιότυπου της μπανάνας, η συνέχεια της διαδικασίας εφαρμόζεται στο δεύτερο αρχείο κώδικα που έχει συνδεθεί στο prefab αντικείμενο της μπανάνας. Εκεί, υπολογίζεται η κατεύθυνση όπου θα πεταχτεί το αντικείμενο βάση της θέσης του παίκτη και της μπανάνας. Τέλος, δημιουργείται ένα δισδιάστατο διάνυσμα και μέσω αυτού μετατοπίζεται τελικά το στιγμιότυπο της μπανάνας με μία καθορισμένη δύναμη βολής.

Εν τέλει όλη αυτή η διαδικασία θα επαναλαμβάνεται συνεχόμενα κάθε τρία δεύτερα και όσο ο παίκτης πλησιάζει την μαϊμού. Τέλος, στο δεύτερο αρχείο κώδικα καθορίζετε, επίσης, ο τρόπος με τον οποίο καταστρέφεται η μπανάνα. Αν περάσουν τρία δεύτερα χωρίς να έρθει σε επαφή με τον παίκτη η μπανάνα, τότε εξαφανίζεται, ενώ στην αντίθετη περίπτωση, πάλι καταστρέφεται αυτόματα.

```
void Start()
{
    rb=GetComponent<Rigidbody2D>();
    player= GameObject.FindGameObjectWithTag("Player");
   if(player.transform.position.x>min){
    Vector3 direction = player.transform.position - transform.position;
    rb.velocity = new Vector2(direction.x,direction.y).normalized*force;

}

void Update()
{
    timer+= Time.deltaTime;
    if(timer>3){
        Destroy(gameObject);
    }
}
void OnTriggerEnter2D(Collider2D other){
    if(other.gameObject.CompareTag("Player")){
        Destroy(gameObject);
   }
}
```

Εικόνα 29: Μετατόπιση μπανάνας

4.4.2 Σκελετός

Ο σκελετός είναι ο πρώτος εχθρός που προγραμματίστηκε, ώστε να μετακινείται στην πίστα. Ουσιαστικά, ξεκινάει το περπάτημα από ένα καθορισμένο σημείο Α(point 1) στον χώρο μέχρι να φτάσει σε ένα τελικό σημείο Β(point 2) και ύστερα επιστρέφει στο πρώτο σημείο ξανά. Αυτή η διαδικασία επαναλαμβάνεται για όσο είναι φορτωμένη η πρώτη πίστα στο παιχνίδι μας, αλλά και όσο είναι ζωντανός ο σκελετός.

Εικόνα 30: Παράδειγμα σημείου 1 και 2

Πιο συγκεκριμένα, έχουμε δηλώσει έναν πίνακα σημείων, τον “wayPoints”, ο οποίος όμως είναι αρχικά άδειος. Σε αυτόν θέλουμε να ορίσουμε τα σημεία μετακίνησης Α και Β και να τον χρησιμοποιήσουμε αργότερα στον κώδικα μας. Μέσω του αντικειμένου ways, λοιπόν, που περιέχει τα δύο σημεία και μέσω μια επανάληψης for, ορίζουμε τα σημεία του ways στον πίνακα “wayPoints”. Επίσης, πριν ξεκινήσει η κίνηση του χαρακτήρα, δηλώνουμε την μεταβλητή “targetPos” όπου αποτελεί την θέση μετακίνησης του, δηλαδή το Β.

```
public void Awake()
{
    wayPoints= new Transform[ways.transform.childCount];
    for(int i=0;i<ways.gameObject.transform.childCount;i++){
        wayPoints[i]=ways.transform.GetChild(i).gameObject.transform;
    }
}
private void Start(){
    anim=GetComponent<Animator>();
    pointCount=wayPoints.Length;
    pointIndex=1;
    targetPos = wayPoints[pointIndex].transform.position;
}
```

Εικόνα 31: Κώδικας σκελετού μέρος 1ο

Η κίνηση του σκελετού θα συμβεί με ένα τρισδιάστατο διάνυσμα το οποίο προσδιορίζει την θέση μετακίνησης του σκελετού, το targetPos και το βήμα που θα πραγματοποιείται κάθε φορά. Αφού ξεκινήσει να κινείται ο σκελετός, θα ελέγχεται συνεχώς με έλεγχο ισότητας αν έφτασε τελικά το σημείο Β. Κατά την κίνηση, αν ο παίκτης τον πλησιάσει, αλλάζει το animation και αρχίζει το χτύπημα με την αξίνα σαν αμυντική αντίδραση.

```
private void Update(){
    var step=speedMultiplier* speed*Time.deltaTime;
    transform.position=Vector3.MoveTowards(transform.position,targetPos,step);
        anim.SetBool("isWalking",true);

    if(transform.position == targetPos){
        NextPoint();
    }
    if(Player.transform.position.x>=33.48 && Player.transform.position.x<=34.53){
        anim.SetTrigger("attack");
        anim.SetBool("isAttacking",true);
    }

}
```

Εικόνα 32: Κώδικας σκελετού μέρος 2ο

Εν τέλη, αν η θέση του βρίσκεται στο Β σημαίνει πώς πρέπει να αλλάξει κατεύθυνση άρα τρέχει μια συνάρτηση, η "NextPoint”. Εκεί, ελέγχεται η φορά κίνησης με την “pointIndex” μεταβλητή και ορίζεται η καινούργια θέση μετακίνησης μέσω της targetPos. Ταυτόχρονα, με την αλλαγή κατεύθυνσης, περιστρέφεται 180 μοίρες το sprite του σκελετού για πιο ρεαλιστική απεικόνιση. Εκτός από την αλλαγή κατεύθυνσης, η “NextPoint” περιέχει και το κάλεσμα μιας περιοδικής ρουτίνας, της “WaitNextPoint”, η οποία καθορίζει τον χρόνο που ο σκελετός θα περιμένει πριν ξεκινήσει το περπάτημα αν βρίσκεται σε οποιοδήποτε από τα δύο σημεία.

```
void NextPoint(){
    if(pointIndex==pointCount-1){
        direction=-1;
        transform.Rotate(0,180,0);
    }
    if(pointIndex==0){
        direction=1;
        transform.Rotate(0,180,0);
    }
    pointIndex+=direction;
    targetPos=wayPoints[pointIndex].transform.position;
    StartCoroutine(WaitNextPoint());
}
IEnumerator WaitNextPoint(){
    speedMultiplier = 0;
    yield return new WaitForSeconds(waitDuration);
    speedMultiplier=1;
}
```

Εικόνα 33: Κώδικας σκελετού μέρος 3ο

Τέλος, το παρακάτω κομμάτι κώδικα περιγράφει τις αντιδράσεις του σκελετού σε περίπτωση χτυπήματος από τον παίκτη. Αρχικά, η επαφή με την μπάλα απλά θα αυξήσει τον μετρητή hit κατά ένα, υποδεικνύοντας έτσι το χτύπημα το οποίο συνέβη στον σκελετό. Αν ο μετρητής φτάσει τον αριθμό δύο, τότε ο σκελετός θα εξολοθρευθεί με χρονοκαθυστέρηση 1.8 δευτερολέπτου για να προλάβει η αναπαραγωγή ολόκληρου του animation.

```
void OnCollisionEnter2D(Collision2D collision){
    if(collision.gameObject.name == "ball(Clone)"){
        anim.SetTrigger("hit");
        anim.SetBool("isHitted",true);
        if(hits==3){
            anim.SetBool("isDead",true);
            Invoke("destroy",1.8f);
        }
        hits=hits+1;
    }
}

void destroy(){
        Destroy(skelet);
}
```

Εικόνα 34: Κώδικας σκελετού μέρος 4ο

4.4.3 Αράχνη

Όπως, έχουμε προαναφέρει η αράχνη είναι ο πιο δύσκολος εχθρός του παιχνιδιού, καθώς μια επαφή του παίκτη μπορεί να αποβεί μοιραία. Ο αλγόριθμος λειτουργίας της είναι βασισμένος στον κώδικα του σκελετού, ενώ πάλι υπάρχει μετατόπιση του εχθρού από ένα σημείο σε ένα άλλο. Το επίπεδο μείωσης της ζωής του παίκτη αποτελεί την μοναδική αλλαγή στο κώδικα με την αράχνη. Είναι ο πιο ανταγωνιστικός εχθρός άρα όπως είναι λογικό αν ο παίκτης έρθει σε επαφή μαζί της, χάνει το 30% της συνολικής ζωής του.

Εκτός από τις αράχνες της 1ης πίστας, υπάρχουν και οι ιπτάμενες αράχνες της 2ης πίστας. Χρησιμοποιούν το ίδιο αρχείο κώδικα, αλλά διαφοροποιούνται σε κάποια σημαντικά σημεία. Το πρώτο είναι ότι η κίνηση της ιπτάμενης αράχνης πραγματοποιείται κάθετα που σημαίνει διαφορετικός ορισμός των σημείων Α και Β, ενώ το δεύτερο ότι ο παίκτης πρέπει να την χτυπήσει πέντε φορές με την μπάλα για να καταστραφεί.

4.4.4 Βασικός εχθρός

Ο βασικός εχθρός μας έχει πολλά όπλα τα οποία εφαρμόζει, ώστε να δυσκολέψει τον πρωταγωνιστή μας στην μεγάλη μάχη που δίνουν μεταξύ τους στην τελευταία πίστα. Γι' αυτό τον λόγο αυτό το αρχείο κώδικα είναι το μεγαλύτερο και πιο περίπλοκο περιέχοντας animations και διάφορες αντιδράσεις του εχθρού.

Γενικά, υπάρχουν τρία στάδια στα οποία μεταβαίνει ο "EvilFred" όταν επιτίθεται στον Mr.JackCat. Στο πρώτο πετάει δηλητηριώδης φίλτρα στον παίκτη, στο δεύτερο ενεργοποιεί τις ιπτάμενες αράχνες, ενώ στο τρίτο εκτοξεύει μπάλες από το κανόνι. Η μετάβαση από στάδιο σε στάδιο, πραγματοποιείται μόνο αν ο παίκτης χτυπήσει τον εχθρό μας στο κεφάλι. Αλλιώς ο "EvilFred" επαναλαμβάνει τον προεπιλεγμένο τρόπο επίθεσης.

Εικόνα 35: Όπλα του βασικού εχθρού μας

Αρχικά, έχει δηλωθεί μια μεταβλητή αληθείας, η "collide", που υποδεικνύει αν ο εχθρός έχει χτυπηθεί από τον παίκτη. Επίσης, έχει δηλωθεί και ένας μετρητής, ο c, πάλι με αρχική τιμή 0 που παίζει κομβικό ρόλο. Η μάχη αρχίζει, λοιπόν, και ο εχθρός πετάει πρώτα τα πράσινα φίλτρα στον Mr.JackCat και η μεταβλητή collide είναι 0.

Για κάθε φίλτρο δημιουργείται αρχικά ένα στιγμιότυπο, καθώς αποτελεί prefab αντικείμενο, και ύστερα με την βοήθεια ενός δεύτερου αρχείου κώδικα μετακινείται προς τον παίκτη. Μετά αυξάνεται ο μετρητής C που αναφέρει πόσα φίλτρα έχουν πεταχτεί. Όταν φτάσει τον αριθμό έντεκα, σταματάει η εκτόξευση φίλτρων για έξι δευτερόλεπτα με χρήση χρονοδιακόπτη. Σε αυτό το διάστημα, ο παίκτης έχει την ευκαιρία να χτυπήσει τον EvilFred στο κεφάλι, ώστε να μειωθεί η ζωή του. Αν ανιχνευτεί σύγκρουση, η collide μεταβλητή γίνεται ένα, αλλιώς επαναλαμβάνεται το πέταγμα των φίλτρων. Στην περίπτωση που γίνει ένα, μηδενίζονται ο μετρητής, οι χρονοδιακόπτες και η ζωή του εχθρού μειώνεται κατά 30%. Επίσης, η μεταβλητή "hitted" αυξάνεται κατά ένα κάθε φορά που ο εχθρός χτυπιέται βοηθώντας στην μετάβαση κάθε σταδίου.

```
if(currentHealth1>0 && c==11 && hitted==0){
  hitted=1;
  anim.SetTrigger("isHitted");
  anim.SetTrigger("isIdle");
  Hurt_sound.PlayOneShot(hurt,0.5F);
  TakeDamageVillain(30);
  anim.SetBool("isPrepare",false);
  collide=1;
  timer2=0;
  c=0;
  timer=0;
}
```

```
if(collide==0){
timer+=Time.deltaTime;
if(timer > 1 && c<11){
    c=c+1;
    timer = 0;
    anim.SetTrigger("isThrowing");
    anim.SetTrigger("isIdle");
    Instantiate(poison,poisonPos.position,Quaternion.identity);
}
if(c==11){
    anim.SetBool("isPrepare",true);
    timer2+=Time.deltaTime;
    if(timer2>6){
    anim.SetBool("isPrepare",false);
    timer2=0;
    c=0;
   }
}
}
```

Εικόνα 36: Ρίψη φίλτρων

Στο επόμενο στάδιο η μεταβλητή collide είναι πλέον ένα και ενεργοποιούνται οι ιπτάμενες αράχνες κάνοντας την τιμή του global αντικειμένου next_level, αληθές. Αμέσως αναγνωρίζεται η αλλαγή της τιμής από το πρόγραμμα και αυτόματα ενεργοποιείται το αρχείο με τις ιπτάμενες αράχνες, ώστε να ξεκινήσει η κίνηση τους. Ουσιαστικά, όταν αυτές εξολοθρευθούν, δεν καταστρέφονται, αλλά απλά απενεργοποιούνται δίνοντας την δυνατότητα στον Mr.JackCat να χτυπήσει τον εχθρό. Αν πάλι ανιχνευθεί σύγκρουση στο κεφάλι του EvilFred, η μεταβλητή collide παίρνει την τιμή δύο και μειώνεται η ζωή του κατά 30%. Στην άλλη περίπτωση, συνεχίζουν το πέταγμα οι αράχνες.

```
else if(collide==1){
  anim.SetBool("push_b",true);
  next_level.SetActive(true);
  if(!Spider1.activeSelf && !Spider2.activeSelf){
    anim.SetBool("no_push",true);
  }
}

if(currentHealth1>0 && !Spider1.activeSelf && !Spider2.activeSelf && hitted==1){
  hitted=2;
  anim.SetBool("no_push",false);
  anim.SetBool("push_b",false);
  TakeDamageVillain(30);
  anim.SetTrigger("isHitted");
  anim.SetTrigger("isIdle");
  Hurt_sound.PlayOneShot(hurt,0.5F);
  collide=2;
}
```

Εικόνα 37: Ιπτάμενες αράχνες

Αμέσως μετά την μετατροπή της τιμής σε δύο, αρχίζει ο τρίτος και τελευταίος μηχανισμός ο οποίος είναι η εκτόξευση μπάλας μέσα από το κανόνι. Με την χρήση χρονοδιακόπτη, κάθε δύο δευτερόλεπτα θα εκτοξεύεται μια μπάλα. Αν μετατοπιστούν εφτά μπάλες, σταματάει η εκτόξευση για έξι δεύτερα και ο παίκτης μπορεί πάλι να χτυπήσει τον εχθρό σε αυτό το διάστημα, αλλιώς θα επαναληφθεί ξανά η εκτόξευση. Αυτό το στάδιο επαναλαμβάνεται συνέχεια μέχρι η ζωή του EvilFred να μηδενιστεί. Τέλος, όταν γίνει μηδέν η ζωή, η μεταβλητή collide παίρνει την τιμή -1 και ο εχθρός πεθαίνει. Απενεργοποιούνται οι colliders και μετακινείται το κελί προς τα πάνω απελευθερώνοντας την γυναίκα του πρωταγωνιστή μας.

```
else if(collide==2){
    timer+=Time.deltaTime;
if(timer > 2 && c1<7){
    c1=c1+1;
    timer = 0;
    anim.SetBool("push_b",true);
    Instantiate(bomb,bombPos.position,Quaternion.identity);
}
if(c1==7){
  if(currentHealth1==0){
    end=1;
  }
    timer2+=Time.deltaTime;
    if(timer2>2)
    anim.SetBool("no_push",true);
    if(timer2>6){
    timer2=0;
    c1=0;
    anim.SetBool("no_push",false);
   }
}
}
```

Εικόνα 38: Εκτόξευση βόμβας

4.5 Εμπόδια

Κατά την υλοποίηση του παιχνιδιού χρησιμοποιήθηκε μεγάλη ποικιλία από εμπόδια με το κάθε ένα να έχει έναν ξεχωριστό μηχανισμό. Το πρώτο και πιο απλό που δημιουργήθηκε είναι το **εμπόδιο κυνηγιού** και βρίσκεται κάτω στο γρασίδι. Αν ο παίκτης πλησιάσει και έρθει σε επαφή μαζί του, αμέσως αυτό κλείνει και τον χτυπάει.

```
void Update()
{

}
void OnCollisionEnter2D(Collision2D other){
    if(other.gameObject.CompareTag("Player")){
        anim.SetTrigger("closed");
        anim.SetBool("isClosed",true);
}
}
```

Εικόνα 39: Παγίδα κυνηγιού

Το δεύτερο εμπόδιο αποτελεί έναν τελείως διαφορετικό μηχανισμό, καθώς πραγματοποιείται πτώση αντικειμένου από ένα υψηλό σημείο. Πιο συγκεκριμένα αναφέρομαι στους **σταλακτίτες** που μετατοπίζονται μέσω κώδικα από πάνω προς τα κάτω στην προσπάθεια να χτυπήσουν το παίκτη. Πρώτον, ελέγχεται αν ο παίκτης κινείται κάτω από τον σταλακτίτη σε ένα ορισμένο πεδίο. Αν ναι, τότε καλείται η συνάρτηση falling η οποία μετατοπίζει τον σταλακτίτη μέσω ενός διανύσματος. Αυτή η διαδικασία πραγματοποιείται με καθυστέρηση μισού δευτέρου για να δοθεί δυνατότητα στον παίκτη να το αποφύγει.

```
void Update()
{
    if(Player.transform.position.x>=x_min && Player.transform.position.x<=x_max && Player.transform.position.y<=y_max && Player.transform.position.y>=y_min){
        anim.SetTrigger("Falling");
        anim.SetBool("isFell",true);
        Invoke("falling",0.5f);
    }
}
void falling(){
        Rock.transform.position=new Vector3(x_move,y_move,z_move);
}
```

Εικόνα 40: Σταλακτίτες

Το επόμενο εμπόδιο είναι η **πτώση των κύβων** και αποτελεί μια κρυφή, δύσκολη παγίδα στην οποία πρέπει να δοθεί μεγάλη προσοχή. Ουσιαστικά, ο παίκτης πατώντας στον κύβο ενεργοποιείται μηχανισμός που ρίχνει αυτό το αντικείμενο και έτσι ο παίκτης πρέπει ταχύτατα να το προσπεράσει. Αρχικά, αν ανιχνευθεί επαφή με τον κύβο, η μεταβλητή αληθείας, block_touch γίνεται αληθές. Αυτή η αλλαγή εντοπίζεται και αμέσως καλείται η συνάρτηση fall. Η συνάρτηση απλά μετατοπίζει τον κύβο κάθετα προς τα κάτω σε ένα συγκεκριμένο σημείο με την χρήση διανύσματος.

```
void Update()
{
    if(block_touch==true){
        Invoke("fall",0.5f);
    }
}

void OnCollisionEnter2D(Collision2D other){
    if(other.gameObject.CompareTag("Player")){
       block_touch=true;
}
}

public void fall(){
    var step=speedMultiplier* speed*Time.deltaTime;
    transform.position=Vector3.MoveTowards(transform.position,point1.transform.position,step);
}
```

Εικόνα 41: Πτώση κύβων

Οι επόμενες δύο παγίδες λειτουργούν μέσω του ίδιου αρχείου κώδικα παρόμοιο με της αράχνης έχοντας ελάχιστες αλλαγές. Οι παγίδες αυτές είναι οι κινούμενες **κοφτερές ακίδες** και οι **κινούμενες πλατφόρμες** και λειτουργούν με τον ίδιο ακριβώς τρόπο. Ουσιαστικά, μετατοπίζονται συνεχώς από το σημείο Α, στο σημείο Β, αλλά και το αντίστροφο προσπαθώντας να πληγώσουν τον παίκτη.

Όσο αφορά τον κώδικα, χρησιμοποιήθηκαν οι περισσότερες υπάρχουσες συναρτήσεις του αρχείου της αράχνης με μικρές διαφορές που θα αναφέρουμε αμέσως. Η μεγάλη διαφορά με το αρχείο της αράχνης είναι ότι αν έρθουν σε επαφή, το εμπόδιο αυτό ορίζεται πατέρας βοηθώντας στην ταυτόχρονη μετακίνηση του παίκτη και του εμποδίου. Απ' την άλλη, στην περίπτωση που δεν υπάρχει κάποια επαφή, πατέρας δεν γίνεται κανένας εφαρμόζοντας την τιμή null σαν πατέρα.

```
void OnCollisionEnter2D(Collision2D collision){
    if(collision.gameObject.name=="Mr.JackCat"){
        collision.gameObject.transform.SetParent(transform);
    }


void OnCollisionExit2D(Collision2D collision){
    if(collision.gameObject.name=="Mr.JackCat"){
        collision.gameObject.transform.SetParent(null);
    }
```

Εικόνα 42: Ορισμός πατέρα αντικειμένου

4.6 Διεπαφή χρήστη

4.6.1 Διάλογος

Στην 2η πίστα δημιουργήθηκε και χρησιμοποιήθηκε ένας βασικός διάλογος μεταξύ του εχθρού και του πρωταγωνιστή μας αναπαριστώντας την διαμάχη που υπάρχει μεταξύ τους. Είναι ένα κομμάτι στο παιχνίδι το οποίο μας λέει πολλά πράγματα για την εξέλιξη της ιστορίας.

Εικόνα 43: Διάλογος πρωταγωνιστών

Αρχικά, ο κώδικας αρχίζει με το κάλεσμα μιας ρουτίνας, της Type, η οποία γράφει στην οθόνη κάθε γράμμα κάθε μιας πρότασης του διαλόγου. Χρησιμοποιείται μια επαναληπτική εντολή για την αναπαράσταση της πρότασης καθορίζοντας ταυτόχρονα την ταχύτητα που θα εμφανίζεται το κάθε γράμμα. Αφού ολοκληρωθεί η εμφάνιση της πρώτης πρότασης, θα γίνει ενεργό ένα κουμπί συνέχειας, το Continue και ο χρήστης μπορεί να το πατήσει. Στην ίδια συνάρτηση αναφέρεται πώς στα πέντε πατήματα του κουμπιού, δηλαδή στην απενεργοποίηση του διαλόγου, θα μπορεί πλέον να λειτουργεί το μενού παύσης μέσω του Escape πλήκτρου.

```
IEnumerator Type(){
    foreach(char letter in sentences[index].ToCharArray()){
        textDisplay.text +=letter;
        yield return new WaitForSeconds(typingSpeed);
    }
}
```

Εικόνα 44: Ρουτίνα εμφάνισης προτάσεων

Τέλος, καλείται η συνάρτηση “NextSentence” η οποία μας μεταβιβάζει στην επόμενη πρόταση. Συγκεκριμένα, απενεργοποιείται το Continue κουμπί, ο δείκτης δείχνει στην καινούργια πρόταση και το παλιό κείμενο στην οθόνη, σβήνει. Αυτή η διαδικασία επαναλαμβάνεται για πέντε φορές μέχρι να ολοκληρωθεί ο διάλογος και να αρχίσει η δεύτερη πίστα.

```
void Start()
{
    StartCoroutine(Type());
    Time.timeScale=0f;
}

void Update(){
    if(textDisplay.text == sentences[index]){
        continueButton.SetActive(true);
    }
    if(count==5){
        if(!next){
            can_pause.SetActive(true);
            Time.timeScale=1f;
            next=true;
        }
    }
}
public void NextSentence(){
    count=count+1;
    continueButton.SetActive(false);
    if(index< sentences.Length - 1){
        index++;
        textDisplay.text = "";
        StartCoroutine(Type());
    }
    else{
        textDisplay.text="";
    }
}
```

Εικόνα 45: Επιλογή επόμενης πρότασης

4.6.2 Συλλογή αντικειμένων και απεικόνιση του σκορ

Σε αυτήν την ενότητα θα εξηγήσουμε αναλυτικά πώς λειτουργεί η εμφάνιση του σκορ, καθώς ο παίκτης συγκεντρώνει τα αντίστοιχα συλλεκτικά αντικείμενα. Όπως, έχουμε αναφέρει, αυτά αποτελούνται από τα ψάρια και τα χρυσά νομίσματα τα οποία κατά την συλλογή τους ενεργοποιούν τον πηγαίο κώδικα. Εδώ, σημαντική σημείωση είναι ότι ο κώδικας του σκορ αποτελεί το πρώτο αρχείο στο οποίο χρησιμοποιείται αποθήκευση δεδομένων σε δυαδικό αρχείο, αλλά και φόρτωση δεδομένων. Περισσότερες λεπτομέρειες για την αποθήκευση θα αναλύσουμε σε επόμενη ενότητα.

Αρχικά, δημιουργήθηκε ένα βασικό κείμενο, το “MyscoreText” που αναπαριστά στην οθόνη το σκορ. Αμέσως μετά δηλώθηκε μια ακέραια μεταβλητή, η “ScoreNum” την οποία θα προσθέτουμε κάθε φορά στο σκορ σε περίπτωση συλλογής αντικειμένου. Αν ο παίκτης έρθει σε επαφή με το ψάρι, αυξάνεται κατά ένα η τιμή της “ScoreNum” και προστίθεται όπως είπαμε στο συνολικό σκορ.

Αν το αντικείμενο που έρθει σε επαφή είναι το χρυσό νόμισμα, τότε ακολουθείτε η ίδια διαδικασία απλά με αύξηση του σκορ κατά δέκα. Επίσης, κάποιος μπορεί να παρατηρήσει ότι στην διάρκεια της συλλογής, χρησιμοποιούνται δύο μετρητές, ο c και ο c1. Ο λόγος είναι πώς όταν ο παίκτης τελικά τερματίσει την πρώτη πίστα, θα εμφανιστούν στατιστικά δεδομένα για το ποσοστό συλλογής σε αυτήν την πίστα. Τα ποσοστά αυτά θα προκύψουν από τις τιμές των συγκεκριμένων μετρητών.

```
private void OnTriggerEnter2D(Collider2D other){
    if(other.gameObject.name == "checkpoint1" || other.gameObject.name == "checkpoint2"){
        SaveManager.instance.score=ScoreNum;
        SaveManager.instance.Save();
    }
    if(other.tag =="Fish"){
        Collect_Sound.PlayOneShot(collect,1);
        ScoreNum+=1;
        Destroy(other.gameObject);
        MyscoreText.text = "" + ScoreNum;
        c+=1;
    }
    if(other.tag =="Coin"){
        Collect_Sound.PlayOneShot(collect_coin,1);
        ScoreNum+=10;
        Destroy(other.gameObject);
        MyscoreText.text = "" + ScoreNum;
        c1+=1;
    }
}
```

Εικόνα 46: Συλλογή συλλεκτικών αντικειμένων

Πριν τροποποιηθούν οι αντίστοιχες τιμές των μετρητών, θα δημιουργηθούν άλλα δύο κείμενα του Unity, το ένα για τα ψάρια και το άλλο για τα χρυσά κέρματα. Τα δύο κείμενα απενεργοποιούνται από το οπτικό πεδίο του παίκτη προσωρινά όσο παίζει την πίστα, αλλά ταυτόχρονα ενημερώνονται με την πρόσθεση των μετρητών σε αυτά ανά τακτά χρονικά διαστήματα. Μετά, όπως αναφέραμε, αν τερματιστεί η πίστα, απλά αναπαρίστανται τα δύο αυτά κείμενα. Επίσης, όταν συναντήσουμε κάποιο από τα σημεία αποθήκευσης προόδου, δηλαδή τα πολύχρωμα διαμάντια, τότε αποθηκεύεται το σκορ σε δυαδικό αρχείο για όσο είναι ενεργή μόνο η πρώτη πίστα. Έτσι, στην περίπτωση που ο παίκτης χάσει, στην επαναφορά, το σκορ παραμένει όσο ήταν στο τελευταίο σημείο αποθήκευσης προόδου.

```
void Start()
{
    SaveManager.instance.Load();
    ScoreNum=SaveManager.instance.score;
    MyscoreText.text = "" + ScoreNum;
    fish.text="" + c ;
    coin.text="" + c1 ;
    Collect_Sound = GetComponent<AudioSource>();
}

void Update(){
    fish.text="" + c ;
    coin.text="" + c1 ;
}
```

Εικόνα 47: Αποθήκευση του σκορ

Τέλος, κάποιος μπορεί να παρατηρήσει ότι στην πρώτη πίστα υπάρχει ένας μοχλός στο έδαφος, ο οποίος μάλιστα έχει μια ενδιαφέρουσα λειτουργία και ένα κρυφό συλλεκτικό αντικείμενο. Με την χρήση κώδικα, λοιπόν, ελέγχεται αν η θέση του παίκτη βρίσκεται κοντά στον μοχλό. Αν ναι, τότε ενεργοποιείται ένα σύννεφο κειμένου δίνοντας οδηγίες σε αυτόν, αλλιώς δεν εμφανίζεται τίποτα. Το σύννεφο κειμένου αναφέρει ότι ο χρήστης πρέπει να πατήσει το πλήκτρο D για να ανοίξει η μπροστινή πύλη, αλλά ταυτόχρονα κρύβεται μια μυστική λειτουργία πίσω από αυτήν την ενέργεια.

Εικόνα 48: Μυστικός μοχλός

Πιο συγκεκριμένα, αν πατηθεί, μια μεταβλητή αληθείας, η "isTurned", μετατρέπεται σε αληθείς. Έτσι, ανιχνεύεται από το πρόγραμμα εφαρμόζοντας μερικούς μετασχηματισμούς. Πρώτον, η θέση της πύλης, μετακινείται προς τα πάνω στο σημείο, point1, επιτρέποντας στον παίκτη να προχωρήσει. Δεύτερον, ανοίγει ένα μυστικό πέρασμα που αφήνει τον παίκτη να συλλέξει ένα κρυφό χρυσό κέρμα.

```
void Update()
{
    if(Player.transform.position.x>65.126 && Player.transform.position.x<65.652 && push==false){
        bubble.SetActive(true);
    }
    if(Player.transform.position.x<65.126 || Player.transform.position.x>65.652){
        bubble.SetActive(false);
    }
    if(Input.GetKeyDown(KeyCode.D) && Player.transform.position.x>=65.13 && Player.transform.position.x<=65.62){
        this.gameObject.GetComponent<SpriteRenderer>().sprite=on;
        isTurned=true;
        push=true;
        bubble.SetActive(false);
    }
    if(isTurned==true){
        var step=speedMultiplier* speed*Time.deltaTime;
        wall.transform.position=Vector3.MoveTowards(wall.transform.position,point1.transform.position,step);
        wall2.transform.position=Vector3.MoveTowards(wall2.transform.position,point2.transform.position,step);
        if(wall.transform.position == point1.transform.position && wall2.transform.position == point2.transform.position)
            isTurned=false;
    }
}
```

Εικόνα 49: Κώδικας λειτουργίας μοχλού

4.6.3 NPC χαρακτήρας

Στο παιχνίδι μας δημιουργήθηκε ένας μη χειριστικός χαρακτήρας ο οποίος απλά αλληλοεπιδράει με τον παίκτη μας στην διάρκεια της πίστας χωρίς όμως να μπορεί να κινηθεί από τον χρήστη. Αρχικά, ο κώδικας ανιχνεύει οποιαδήποτε σύγκρουση του παίκτη με τον NPC χαρακτήρα ενεργοποιώντας στην περίπτωση αυτή ένα πάνελ κειμένου. Σε αυτό εκφράζονται σκέψεις του χαρακτήρα, αλλά και προειδοποιήσεις για την συνέχεια της πίστας. Αφού ενεργοποιήθηκε το πάνελ, ο χρήστης πατώντας το πλήκτρο D, θα μπορεί αντίστοιχα να το απενεργοποιήσει. Στην εναλλαγή αυτή παίζει σημαντικό ρόλο η μεταβλητή, enter, η οποία δεν αφήνει να εμφανιστεί δεύτερη φορά το πάνελ.

```
void Update()
{
    if(Input.GetKeyDown(KeyCode.D)&&npc_p.activeSelf){
        npc_p.SetActive(false);
        Time.timeScale = 1f;
    }
}
void OnCollisionEnter2D(Collision2D collision){
    if(collision.gameObject.name == "NPC" && enter==false){
        enter=true;
        npc_p.SetActive(true);
        Time.timeScale = 0f;
    }
}
```

Εικόνα 50: Κώδικας NPC χαρακτήρα

4.6.4 Ηχητικά εφέ

Για την χρήση ηχητικών εφέ προστέθηκαν πηγές ήχου του Unity στα αντικείμενα του παιχνιδιού. Μετά με την χρήση κώδικα, δημιουργήθηκαν μεταβλητές η οποίες χρησίμεψαν σαν κλιπ ήχου. Τέλος, εφαρμόστηκαν διάφορες συναρτήσεις σε αυτές τις μεταβλητές για αναπαραγωγή του ήχου προσπαθώντας να ακούγονται συγχρονισμένα με το παιχνίδι. Ένα απλό παράδειγμα αποτελεί το αρχείο κώδικα του Game Over. Σε αυτό δηλώθηκε η μεταβλητή GameOver_Sound στην οποία εφαρμόσαμε την συνάρτηση “PlayOneShot” για αναπαραγωγή του ηχητικού εφέ όταν ο παίκτης χάσει.

```
void Start()
{
    GameOver_Sound = GetComponent<AudioSource>();
}

if(playerview.transform.position.y<-2.7){
    if(Sound==false){
    GameOver_Sound.PlayOneShot(gameOver,0.5F);
    GameOverUI.SetActive(true);
    Time.timeScale = 0f;
    Sound=true;
    }
```

Εικόνα 51: Ηχητικά εφέ

4.7 Σημεία αποθήκευσης προόδου (Checkpoints)

Μια κρίσιμη λειτουργία σε ένα παιχνίδι πλατφόρμας σίγουρα είναι η συλλογή αντικειμένων κατά την διάρκεια της πίστας όπου αποθηκεύουν προσωρινά την πρόοδο του παίκτη. Στο παιχνίδι μας, λοιπόν, δημιουργήθηκαν τρία αρχεία κώδικα που συμβάλουν στην σωστή λειτουργία της προσωρινής αποθήκευσης. Στο πρώτο αρχείο δηλώνεται μια τοπική μεταβλητή, η instance η οποία με το κλείσιμο της σκηνής, κρατάει ενεργό το αρχείο στην μορφή αντικειμένου. Αυτό συμβαίνει για να μπορεί να ανιχνεύεται οποιαδήποτε συλλογή χρωματιστού διαμαντιού αλλάζοντας την θέση επαναφοράς του παίκτη αν χάσει.

```
void Awake(){
    if(instance == null){
        instance=this;
        DontDestroyOnLoad(instance);
    }
    else{
        Destroy(gameObject);
    }
}
```

Εικόνα 52: Κώδικας ενεργού αντικειμένου αποθήκευσης

Το δεύτερο αρχείο χρησιμεύει στην περίπτωση που ο παίκτης χάσει και απλά το παιχνίδι επαναφέρεται με την θέση του παίκτη να έχει αλλάξει. Όλα αυτά με την προϋπόθεση ότι έχει συλλέξει το αντίστοιχο συλλεκτικό αντικείμενο αποθήκευσης, δηλαδή το χρωματιστό διαμάντι, ώστε να γίνει η μετακίνηση εκεί.

```
void Start()
{
    cp=GameObject.FindGameObjectWithTag("check").GetComponent<CheckSystem>();
    transform.position=cp.lastCheckPointPos;
}
```

Εικόνα 53: Αλλαγή θέσης παίκτη

Το τρίτο αρχείο απλά ανιχνεύει την επαφή του παίκτη με το διαμάντι και κάθε φορά ορίζεται η καινούργια προσωρινή θέση αποθήκευσης μέσω της μεταβλητής “lastCheckPointPos”. Αμέσως μετά, καλείται η συνάρτηση Disappear η οποία μετά από ένα δευτερόλεπτο, απενεργοποιεί το διαμάντι και ταυτόχρονα ακούγεται ένας χαρακτηριστικός ήχος συλλογής.

```
void Start()
{
    cp=GameObject.FindGameObjectWithTag("check").GetComponent<CheckSystem>();
    Checkp_music=GetComponent<AudioSource>();
}

void OnTriggerEnter2D(Collider2D other){
    if(other.CompareTag("Player")){
        Checkp_music.PlayOneShot(check,0.5F);
        cp.lastCheckPointPos=transform.position;
        Invoke("Disappear",1);
    }
}
void Disappear(){
    gameObject.SetActive(false);
}
```

Εικόνα 54: Αλλαγή σημείου επαναφοράς

4.8 Αποθήκευση παιχνιδιού

Η αποθήκευση δεν υπάρχει σαν επιλογή στο παιχνίδι μας, αλλά πραγματοποιείται αυτόματη αποθήκευση με τον τερματισμό της 1ης πίστας. Αρχικά, δημιουργήθηκε ένα αρχείο κώδικα για την διαχείριση της αποθήκευσης και το κάλεσμα διάφορων συναρτήσεων από εξωτερικά αρχεία το οποίο παραμένει ενεργό σε οποιαδήποτε εναλλαγή σκηνής. Η βασικές συναρτήσεις του είναι οι Save και Load όπως είναι λογικό οι οποίες χρησιμοποιούν μεταβλητές από serializable κλάση. Αυτό σημαίνει ότι οι μεταβλητές μετατρέπονται στην μορφή bytes, ώστε να μπορεί να γίνει αποθήκευση τους στον δίσκο.

Στην συνάρτηση Save δηλώνεται ένα δυαδικό αρχείο και ταυτόχρονα μια μεταβλητή της κλάσης μας, την data, μέσω της οποίας θα γίνει η βασική αποθήκευση. Αμέσως μετά, καθορίζουμε τα δεδομένα που θα αποθηκεύσουμε μέσω της data, τα μεταβιβάζουμε στο αρχείο και τέλος κλείνουμε το αντίστοιχο αρχείο.

```
public void Save(){
    BinaryFormatter bf=new BinaryFormatter();
    FileStream file = File.Create(Application.persistentDataPath + "/VariableInfo.dat");
    Variable_storage data= new Variable_storage();
    data.saved=saved;
    data.score=score;
    bf.Serialize(file,data);
    file.Close();
}

[Serializable]
class Variable_storage{
    public bool saved;
    public int score;
}
```

Εικόνα 55: Αποθήκευση παιχνιδιού

Στην συνάρτηση Load ελέγχεται πρώτα αν υπάρχει το αποθηκευμένο αρχείο στο αντίστοιχο μονοπάτι, αλλιώς δεν προχωράει σε κάποια ενέργεια. Αν υπάρχει, τότε τα αποθηκευμένα δεδομένα του φορτώνονται σε μια μεταβλητή της κλάσης, την data, αλλά πρώτα πραγματοποιείται η αντίθετη διαδικασία του Serialization των δεδομένων για να μπορεί να γίνει η ανάγνωση τους. Ύστερα, τα δεδομένα φορτώνονται μέσω της data σε μεταβλητές του αρχείου και τέλος αυτό κλείνει.

```
public void Load(){
    if(File.Exists(Application.persistentDataPath + "/VariableInfo.dat")){
        BinaryFormatter bf = new BinaryFormatter();
        FileStream file = File.Open(Application.persistentDataPath + "/VariableInfo.dat",FileMode.Open);
        Variable_storage data =(Variable_storage)bf.Deserialize(file);

        saved= data.saved;
        score=data.score;
        file.Close();
    }

}
```

Εικόνα 56: Φόρτωση παιχνιδιού

Όλα αρχίζουν, λοιπόν, μόλις ο παίκτης τερματίσει την πρώτη πίστα αγγίζοντας την πλατφόρμα τερματισμού. Έτσι, καλείται η συνάρτηση load_final η οποία αλλάζει την global μεταβλητή saved σε αληθείς κάνοντας αυτόματη αποθήκευση. Επίσης, γίνεται ενεργό και το πάνελ που αναφέρει τα στατιστικά συλλογής του παίκτη το οποίο θα γίνει ανενεργό μόνο αν ο χρήστης πατήσει το πλήκτρο C και θα φορτωθεί η 2η πίστα.

```
void Update()
{
    if(What_you_collect.activeSelf && sceneName=="Level_1" && Input.GetKeyDown(KeyCode.C)){
        SceneManager.LoadScene("Boss_lvl");
        What_you_collect.SetActive(false);
    }
}

void OnCollisionEnter2D(Collision2D collision){
    if(collision.gameObject.name == "Finish_line"){
        anim.SetTrigger("Finish");
        Invoke("load_final",4);
}}

public void load_final(){
    saved=true;
    SaveManager.instance.saved=saved;
    SaveManager.instance.Save();
    Time.timeScale=0f;
    What_you_collect.SetActive(true);
}
```

Εικόνα 57: Μετάβαση πίστας και αυτόματη αποθήκευση

Στην 2η πίστα όταν ο παίκτης νικήσει τον κακό επιστήμονα EvilFred, θα πλησιάσει την γυναίκα του με αποτέλεσμα να ανιχνεύεται η επαφή μαζί της. Τότε, θα γίνει ενεργή η κρυμμένη εικόνα της καρδιάς που θα εμφανιστεί πάνω από τα κεφάλια τους. Στην συνέχεια μέσω της invoke θα καλεστεί η end συνάρτηση μετατρέποντας την μεταβλητή saved σε ψευδές και εμφανίζοντας το πάνελ νίκης.

```
void OnCollisionEnter2D(Collision2D other){
    if(other.gameObject.name=="female_cat"){
        heart.SetActive(true);
        Invoke("end",2);
    }
     public void end(){
       saved=false;
       SaveManager.instance.saved=saved;
       SaveManager.instance.Save();
       Win_sound.PlayOneShot(win,1);
       win_panel.SetActive(true);
     }
```

Εικόνα 58: Εμφάνιση πάνελ νίκης

Τέλος, η φόρτωση του αποθηκευμένου παιχνιδιού πραγματοποιείται από το κύριο μενού μέσω της επιλογής load game μόνο αν η μεταβλητή saved είναι αληθείς άρα θα φορτωθεί η δεύτερη πίστα. Στην άλλη περίπτωση, σημαίνει ότι δεν υπάρχουν αποθηκευμένα δεδομένα άρα θα εμφανιστεί αντίστοιχο μήνυμα στην οθόνη το οποίο θα εξαφανιστεί μετά από δύο δευτερόλεπτα. Επιπλέον, η επιλογή του μενού New Game, ενεργοποιεί τον μηδενισμό του σκορ, με χρήση αποθήκευσης, για να μην αναγράφεται το προηγούμενο σκορ κατά την εκκίνηση όπως και αλλαγή στην τιμή της saved για ξεκίνημα νέου παιχνιδιού.

Εικόνα 59: Μήνυμα μη αποθηκευμένου παιχνιδιού

```
public void LoadGame(){
    SaveManager.instance.Load();
    SaveManager.instance.saved=saved;
    if(saved==true){
        SceneManager.LoadScene("Boss_lvl");
    }
    else if(saved==false){
        no_saved.SetActive(true);
        Invoke("Disappear",2);
    }
}

public void PlayGame(){
    SaveManager.instance.score=0;
    SaveManager.instance.saved=false;
    SaveManager.instance.Save();
    story_1.SetActive(true);
}
```

Εικόνα 60: Επιλογή φόρτωσης παιχνιδιού

5. Σύνοψη

5.1 Συμπεράσματα

Το συμπέρασμα αυτής της πτυχιακής εργασίας είναι ότι δημιουργήθηκε ένα δομημένο κλασσικό ρετρό παιχνίδι της δεκαετίας του 80 που περιλαμβάνει όλα τα απαραίτητα συστατικά για να φαίνεται απολαυστικό αποτελώντας ελκυστικό περιεχόμενο για μικρούς και μεγάλους. Δημιουργήθηκαν δύο πίστες η οποίες είναι αυξανόμενης δυσκολίας, καθώς ο παίκτης προχωράει σε αυτές με την πρώτη να είναι μια κλασσική πίστα πλατφόρμας. Σε αυτήν ο παίκτης θα συναντήσει διάφορους χαρακτήρες, εχθρούς, εμπόδια και παγίδες που ανεβάζουν το παιχνίδι σε ένα ανταγωνιστικό επίπεδο. Αντίστοιχα, στην δεύτερη πίστα το επίπεδο δυσκολίας είναι υψηλότερο, καθώς αποτελεί μια μάχη αρένας του παίκτη μας και του βασικού εχθρού. Επίσης, προστέθηκαν διάφορες μουσικές υποκρούσεις, animations και κινήσεις για πιο ποιοτική αίσθηση του παιχνιδιού.

5.2 Μελλοντικές εργασίες και επεκτάσεις

Αυτή η πτυχιακή εργασία περιλαμβάνει διάφορους μηχανισμούς και αλγόριθμους οι οποίοι θα μπορούσαν να αποτελούν παράδειγμα για μελλοντική χρήση από άλλες πτυχιακές εργασίες δημιουργίας παιχνιδιού. Θα μπορούσε να αποτελέσει ένα πάτημα για την κατασκευή παιχνιδιού σε συνδυασμό με τους μηχανισμούς που παρέχει η τεχνολογία η οποία εξελίσσεται με ραγδαίους ρυθμούς. Η εξέλιξη των Game Engines, των αλγόριθμων και της τεχνητής νοημοσύνης δίνουν την δυνατότητα δημιουργίας ποιοτικότερων παιχνιδιών άρα αντίστοιχα και πιο ενδιαφέρων πτυχιακών εργασιών με αντίστοιχο θέμα.

5.3 Προκλήσεις κατά την υλοποίηση του παιχνιδιού

Στην διάρκεια δημιουργίας του παιχνιδιού συναντήθηκαν αρκετές δυσκολίες τόσο στο περιβάλλον υλοποίησης όσο και στον σχεδιασμό των χαρακτήρων. Όμως, η μεγαλύτερη πρόκληση αποδείχθηκε η υλοποίηση του πηγαίου κώδικα ο οποίος ξόδεψε αρκετό από τον διαθέσιμο χρόνο μου. Η αντιμετώπιση των σφαλμάτων του κώδικα και η επίλυση τους ήταν η δυσκολότερη διαδικασία όπως και η λογική σχεδίαση των ποικίλων αλγόριθμων. Παρ ’όλα αυτά, η προσπάθεια αυτή με βοήθησε να κατανοήσω καλύτερα τον τρόπο διαχείρισης κώδικα και την σημασία της προγραμματιστικής ικανότητας στην διάρκεια υλοποίησης της πτυχιακής εργασίας.

Βιβλιογραφία

1. **https://docs.unity3d.com/Manual/**
2. **https://free-game-assets.itch.io/pixel-art-animated-traps**
3. **https://assetstore.unity.com/packages/2d/characters/sunny-land-103349**
4. **https://www.youtube.com/watch?v=RuvfOl8HhhM**
5. **https://www.youtube.com/watch?v=zc8ac_qUXQY&t=359s**
6. **https://www.youtube.com/watch?v=QGDeafTx5ug&list=PLBIb_auVtBwBotxgdQXn2smO0Fvqqea4-&pp=iAQB**
7. **https://www.youtube.com/watch?v=0tDPxNB2JNs**
8. **https://www.youtube.com/watch?v=on9nwbZngyw&list=PLPV2KyIb3jR6TFcFuzI2bB7TMNIIBpKMQ&pp=iAQB**
9. **https://www.youtube.com/watch?v=_nRzoTzeyxU&t=35s**
10. **https://www.youtube.com/watch?v=VbZ9_C4-Qbo&list=PLPV2KyIb3jR53Jce9hP7G5xC4O9AgnOuL&index=9**
11. **https://www.youtube.com/watch?v=vZU51tbgMXk&t=392s**
12. **https://www.youtube.com/watch?v=XOjd_qU2Ido&t=688s**
13. **https://www.youtube.com/watch?v=qWCLauk9oNo**
14. **https://www.youtube.com/watch?v=uIfD2BKaD2k**
15. **https://www.youtube.com/watch?v=Wsw-86zjb8I&list=PLobY7vO0pgVKn4FRDgwXk5FUSiGS8_jA8&index=2**
16. **https://www.youtube.com/watch?v=7GDq2MKjyec**
17. **https://www.youtube.com/watch?v=mQSmVZU5EL4**
18. **https://www.youtube.com/watch?v=zR2m4NvohJQ**
19. **https://en.wikipedia.org/wiki/Finite-state_machine**