ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
Τμήμα Πληροφορικής

ΤΙΤΛΟΣ	**ΕΠΙΒΙΩΣΗ ΣΕ ΕΝΑ ΕΡΗΜΟ ΝΗΣΙ**
ΤΙΤΛΟΣ ΣΤΑ ΑΓΓΛΙΚΑ	**SURVIVE IN A DESERT ISLAND**
ΟΝΟΜΑΤΕΠΩΝΥΜΟ	**ΦΥΤΡΟΣ ΕΥΑΓΓΕΛΟΣ**
ΠΑΤΡΩΝΥΜΟ	**ΔΗΜΗΤΡΙΟΣ**
ΑΡΙΘΜΟΣ ΜΗΤΡΩΟΥ	**Π18220**
ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ	**ΠΑΝΑΓΙΩΤΟΠΟΥΛΟΣ ΘΕΜΙΣΤΟΚΛΗΣ**

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

1. Περίληψη

Σε αυτήν την εργασία παρουσιάζεται η διαδικασία ανάπτυξης ενός 3D FPS παιχνιδιού με τίτλο "**SURVIVE IN A DESERT ISLAND**" στην πλατφόρμα Unity. Η εργασία ξεκινά με μια σύντομη εισαγωγή και μια περιγραφή του παιχνιδιού, ενώ ακολουθεί μια ανάλυση της πλοκής που υποκρύπτεται πίσω από αυτό. Εξετάζεται ο τρόπος σχεδιασμού του παιχνιδιού, συμπεριλαμβανομένων των scripts που δίνουν λειτουργικότητα στο παιχνίδι. Το έργο περιλαμβάνει επίσης χρονοδιαγράμματα, εισαγωγική οθόνη, in-game μενού και σχεδίαση mini-map. Επιπλέον, συνοδεύεται από ένα video-walkthrough που παρέχει μια περαιτέρω εικόνα του παιχνιδιού.

2. Εισαγωγή

Η τεχνολογία των υπολογιστών και των παιχνιδιών έχει εξελιχθεί με εντυπωσιακό τρόπο τις τελευταίες δεκαετίες, δημιουργώντας έναν κόσμο όπου η εικονική πραγματικότητα συγκρούεται με την πραγματική. Σε αυτόν τον δυναμικό και ανεπτυγμένο χώρο, η παρούσα πτυχιακή εργασία προσφέρει μια εισαγωγή σε έναν από τους σημαντικότερους τομείς της πληροφορικής και της ψυχαγωγίας: τη δημιουργία 3D First-Person Shooter (FPS) παιχνιδιών μέσω της πλατφόρμας Unity.

Τα 3D FPS παιχνίδια αποτελούν έναν συναρπαστικό κόσμο, όπου οι παίκτες εισέρχονται σε εντυπωσιακά περιβάλλοντα και ζουν εμπειρίες που αντανακλούν την φαντασία και τη δημιουργικότητα των δημιουργών τους. Μέσα από αυτήν την εργασία, θα αναδείξουμε τον συναρπαστικό κόσμο της ανάπτυξης παιχνιδιών με Unity και θα διερευνήσουμε τις τεχνολογίες, τις προσεγγίσεις και τις διαδικασίες που απαιτούνται για τη δημιουργία ενός επιτυχημένου 3D FPS παιχνιδιού.

Επίσης, θα αναλύσουμε την πλοκή, τον σχεδιασμό, την υλοποίηση, και την διαδικασία ανάπτυξης ενός τέτοιου παιχνιδιού, εξετάζοντας παράλληλα τις προκλήσεις και τις ευκαιρίες που αντιμετωπίζουν οι προγραμματιστές και οι δημιουργοί. Στο τέλος, θα παρουσιάσουμε το αποτέλεσμα της προσπάθειάς μας, πλαισιώνοντάς το με ένα video-walkthrough που αναδεικνύει την ουσία του παιχνιδιού μας.

Αυτή η εργασία αποτελεί ένα ταξίδι στον κόσμο των 3D FPS παιχνιδιών, προσφέροντας μια ευκαιρία να εξερευνήσουμε και να κατανοήσουμε τη μαγεία της δημιουργίας εντυπωσιακών εμπειριών για τους παίκτες.

3. Πρόβλημα και Πιθανές Προσεγγίσεις

Ανάλυση του προβλήματος

Το πρόβλημα που αντιμετωπίζεται είναι η δημιουργία ενός 3D First-Person Shooter (FPS) παιχνιδιού στην πλατφόρμα Unity. Αυτό ενέχει μια σειρά προκλήσεων και απαιτήσεων, καθώς η δημιουργία ενός επιτυχημένου παιχνιδιού απαιτεί συνεργασία μεταξύ πολλών τεχνικών, δημιουργικών και σχεδιαστικών πτυχών. Αναλύοντας περαιτέρω:

- **Σχεδιασμός του Παιχνιδιού**: Ένα από τα βασικά προβλήματα είναι πώς να δημιουργηθεί ένα ενδιαφέρον περιβάλλον παιχνιδιού που να ελκύει τους παίκτες και να τους παρέχει συναρπαστικές εμπειρίες.
- **Απόδοση και Γραφικά**: Πώς θα διατηρήσετε υψηλή απόδοση (performance) του παιχνιδιού, προσφέροντας ταυτόχρονα υψηλή ποιότητα γραφικών και φυσικής αλληλεπίδρασης για τους παίκτες.
- **NPC:** Ο τρόπος με τον οποίο θα λειτουργήσουν τα NPC και ποιες τεχνικές χρησιμοποιούνται για τον προγραμματισμό τους καθώς και οι αλληλεπιδράσεις με τον παίχτη.

Πιθανές Προσεγγίσεις

Για την αντιμετώπιση αυτών των προβλημάτων, υπάρχουν πολλές πιθανές προσεγγίσεις:

- **Σχεδιασμός Περιβάλλοντος**: Ο σχεδιασμός του περιβάλλοντος περιλαμβάνει την χρήση των κατάλληλων εργαλείων για την δημιουργία του τοπίου, των αντικειμένων και των χαρακτήρων που απαιτούνται για την δημιουργία μιας εντυπωσιακής και ανθρώπινης εμπειρίας.
- **Βελτιστοποίηση της απόδοσης**: Η βελτιστοποίηση της απόδοσης απαιτεί την χρήση διάφορων τεχνικών καθώς διάφορα assets τρώνε πολλούς πόρους και χρησιμοποιούν αρκετά την CPU και την GPU.
- **Λειτουργικά NPC:** "Για τον σχεδιασμό και την υλοποίηση των λειτουργικών NPC στο παιχνίδι μας, μπορούμε να χρησιμοποιήσουμε πολλές τεχνικές και προσεγγίσεις. Στην τεχνητή νοημοσύνη για τον έλεγχο της συμπεριφοράς των χαρακτήρων, μπορεί να υλοποιηθεί ένα σύστημα βασισμένο σε αλγόριθμους Finite State Machines (FSM) για την αντίδραση των

NPC σε διάφορες καταστάσεις ή αλλιώς κάποιο asset που τα ετοιμάζει όλα πχ Emerald AI.

Επιπλέον, μπορούν να χρησιμοποιηθούν διάφορα assets, όπως 3D μοντέλα χαρακτήρων και αντικειμένων, για να δημιουργήσουμε ένα πλήρως λειτουργικό περιβάλλον.

4. Προδιαγραφές, ανάλυση απαιτήσεων και ανάλυση της εφαρμογής

Απαιτήσεις Συστήματος:

- **Επεξεργαστής CPU:** Επεξεργαστής Intel Core i5 ή αντίστοιχος AMD. Το παιχνίδι έχει δοκιμαστεί και σε Intel Core i3-6100 με 40-60 fps.
- **Κάρτα γραφικών GPU:** NVIDIA GeForce GTX 960 ή AMD Radeon R9 280X με τουλάχιστον 2GB RAM. Προτιμότερο βέβαια να έχει 3GB RAM και πάνω.
- **Μνήμη RAM:** Τουλάχιστον 8GB RAM.
- **Σκληρός δίσκος:** Τουλάχιστον 2GB διαθέσιμου χώρου για την εγκατάσταση του παιχνιδιού και των αρχείων αποθήκευσης.
- **Ήχος:** Κάρτα ήχου συμβατή με το DirectX.
- Λειτουργικό σύστημα: Windows 10(64-bit).

Απαιτήσεις Λογισμικού:

- Λογισμικό Unity: Το παιχνίδι έχει αναπτυχθεί χρησιμοποιώντας το περιβάλλον ανάπτυξης Unity.
- Οδηγοί κάρτας γραφικών: Οι τελευταίοι οδηγοί για την κάρτα γραφικών πρέπει να είναι εγκατεστημένοι και ενημερωμένοι.

Ανάλυση Παιχνιδιού:

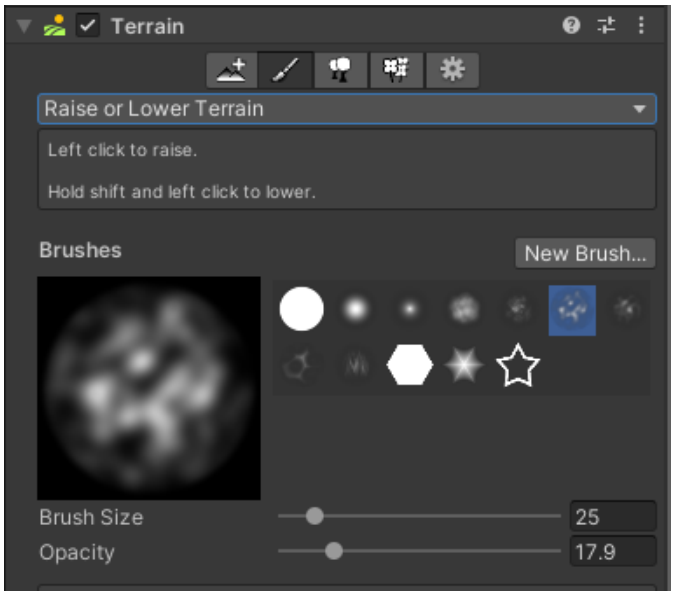
- **Τίτλος παιχνιδιού:** Survive in a desert island.
- **Είδος παιχνιδιού:** FPS(First-Person Shooter).
- **Περιγραφή του παιχνιδιού:** Το παιχνίδι "SURVIVE IN A DESERT ISLAND" διαδραματίζεται σε έναν απομονωμένο νησί, όπου ο πρωταγωνιστής βρίσκεται εκεί αφότου το αεροπλάνο του συντρίβεται. Ο κύριος στόχος του παίκτη είναι να επιβιώσει σε αυτό το απαιτητικό περιβάλλον, βρίσκοντας τροφή και νερό καθώς και να ανακαλύψει τα μυστικά του νησιού.

  Καθώς ο πρωταγωνιστής αναζητά τους πρώτους αναγκαίους πόρους, ανακαλύπτει ένα εγκαταλελειμμένο χωριό στο νησί. Εκεί, βρίσκει ένα βιβλίο ενός επιζώντα που αφηγείται την ιστορία του νησιού και το μυστικό των τριών διαμαντιών. Για να φύγει από το νησί, ο πρωταγωνιστής πρέπει να βρει και να συγκεντρώσει αυτά τα τρία διαμάντια, τα οποία βρίσκονται σε διάφορα μέρη του νησιού.

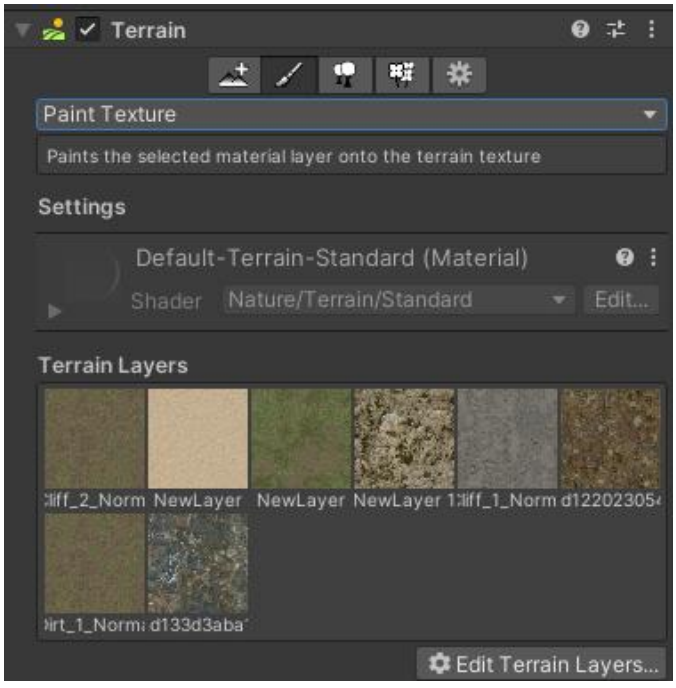
  Καθώς περιηγείται στο νησί, ο παίκτης θα αντιμετωπίσει πολλές προκλήσεις, όπως επικίνδυνα ζώα, και θα χρειαστεί να χρησιμοποιήσει τη φαντασία του και τις δεξιότητές του για να επιβιώσει και να προχωρήσει στην αναζήτηση των διαμαντιών.

  Αυτή η περιπέτεια διαθέτει συναρπαστική δράση και ανακαλύψεις που περιμένουν τον παίκτη σε κάθε γωνιά του νησιού. Θα χρειαστεί να επιδείξει τον συντονισμό, την επιδεξιότητά του και την ικανότητά του να λύνει προβλήματα για να ολοκληρώσει αυτήν την αποστολή και να ανακαλύψει το μυστικό του νησιού.

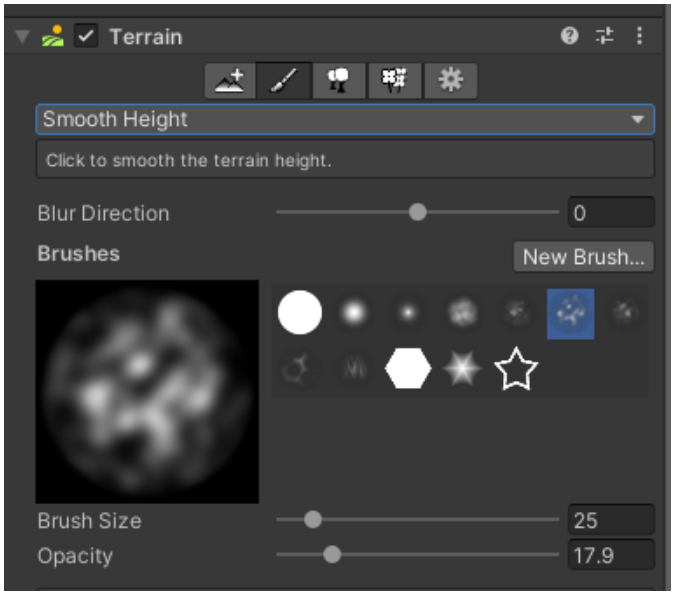
- **Σχεδίαση περιβάλλοντος:** Το περιβάλλον το παιχνιδιού σχεδιάστηκε μέσω των εργαλείων που σου παρέχει το unity.
    - **Raise or Lower Terrain:** Χρησιμοποιήθηκε το συγκεκριμένο εργαλείο για να διαμορφωθεί το terrain με τέτοιο τρόπο ώστε να θυμίζει νησί με λόφους παραλίες και βουνά.



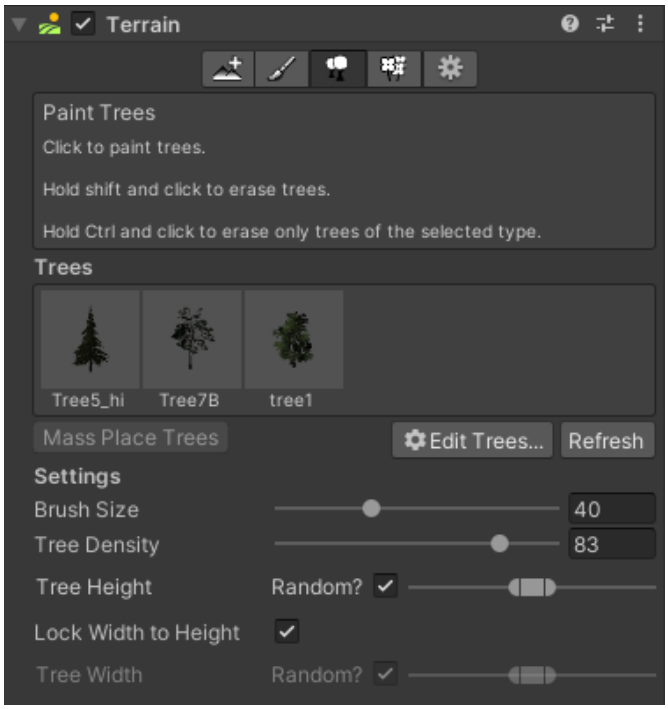
- **Paint Textures:** Με συγκεκριμένο εργαλείο ζωγραφίσαμε το terrain έτσι ώστε να έχουμε τα κατάλληλα χρώματα στο έδαφος, στα βουνά και στις παραλίες.



- **Smooth Height**: Με το εργαλείο αυτό φτιάξαμε τους λόφους έτσι ώστε να φαίνονται ομοιομόρφοι όπου θεωρήθηκαν κατάλληλοι.



- **Paint trees:** Έπειτα προσθέσαμε στο παιχνίδι μας δέντρα ώστε να σχεδιαστεί ένα αληθοφανές δάσος χρησιμοποιώντας τρια assets δέντρων.



- **Κύριος χαρακτήρας:** Ο κυριος χαρακτηρας του παιχνιδιου, ο Remi, είναι ένας ρεαλιστικός χαρακτήρας που δημιουργήθηκε με την χρήση ενός asset από το MIxamo.com, συμπεριλαμβανομένων και των κατάλληλων animations. Ο Remi εχει έναν αρχικό εξοπλισμό περιορισμένων πόρων και πρέπει να επιβιώσει στο απομονωμένο νησί.
    - **Κίνηση:** Ο Remi μπορεί να περπατήσει, να τρέξει, να πηδήξει, να ρίξει με τόξο, με τσεκουρι και να κολυμπησει.
    - **Εξοπλισμός:** Αρχικά, διαθέτει ένα backpack και ένα μπουκάλι με νερό και μπορεί να μαζέψει ένα απόθεμα τροφίμων(μήλων και αχλαδιών) για την επιβίωση του.
    - **Στόχος:** Ο στόχος του χαρακτήρα μας είναι να βρεί τα κατάλληλα αντικείμενα ώστε να επιβιώσει και να προχωρήσει στο παιχνίδι.

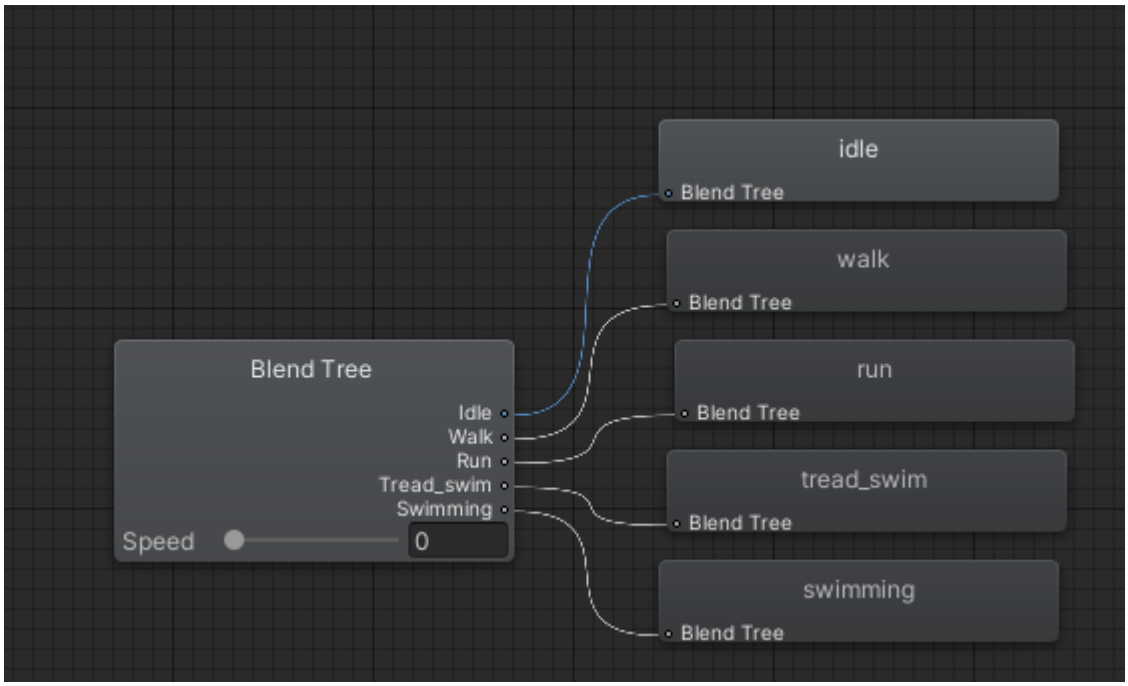
- **NPC:** Τα NPC του παιχνιδιού, είναι οι λύκοι τους οποίους μπορεί να συναντήσει ο χαρακτήρας κατά την διάρκεια της εξερεύνησης του νησιου. Οι λύκοι είναι απειλητικοί και εάν ο χαρακτήρας μας δεν προσέξει θα του επιτεθούν.

- **Σχεδιασμός περιβάλλοντος:**
  - **Τοπογραφία του νησιού:** Το παιχνίδι διαδραματίζεται σε ένα ανεξερεύνητο νησί. Ο σχεδιασμός του νησιού ήταν κρίσιμης σημασίας καθώς ήταν μεγάλης σημασίας η δημιουργία ενός εντυπωσιακού περιβάλλοντος.
  - **Φυσικά στοιχεία:** Το περιβάλλον, είναι αληθοφανές με ποικιλία σε δάση, παραλίες, βουνά και σπηλιές.
  - **Κτήρια:** Στο παιχνίδι έχουν προστεθεί σπίτια τα οποία ο χαρακτήρας μπορεί να χρησιμοποιήσει για την επιβίωση του.

➢ Ανάλυση χαρακτήρων και NPC:

Κύριος χαρακτήρας – Remi

- **Χαρακτηριστικά:** Ο κύριος χαρακτήρας μας, ο Remi, αποτελείται από τα κατάλληλα animation ώστε να κινείται σωστά τα οποία προγραμματίστηκαν μέσω blend Trees.



- **Εξοπλισμός:** Ο χαρακτήρας μας ξεκινάει το παιχνίδι στην παραλία του νησιού και πρέπει να βρει ένα σακίδιο, ένα μπουκάλι για νερό, ένα τόξο με βέλη και ένα τσεκούρι ώστε να μπορέσει να επιβιώσει.
- **Αλληλεπίδραση:** Ο Remi, μπορεί να αλληλοεπιδράει με αντικείμενα μέσω colliders, ώστε να μαζέψει τροφή και τα υπόλοιπα αντικείμενα που προαναφέραμε.

➢ **NPC – ΛΥΚΟΙ:**

- **Χαρακτηριστικά:** Οι λύκοι, αποτελούνται από τα κατάλληλα animation ώστε να κινείται σωστά τα οποία προγραμματίστηκαν μέσω του Emerald AI για την κατάλληλη κίνηση των λύκων.

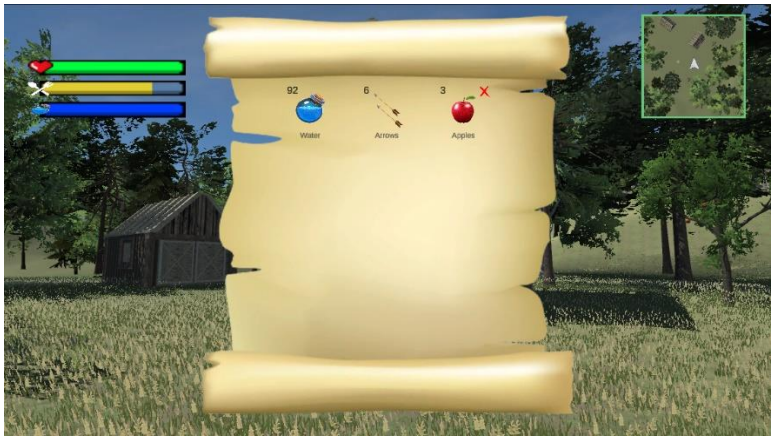
- **Συμπεριφορά:** Οι λύκοι μας επιτίθονται στον χαρακτήρας εάν αυτός βρεθεί σε κοντινή απόσταση.

➢ **Ανάλυση Διασύνδεσης του χρήστη:**

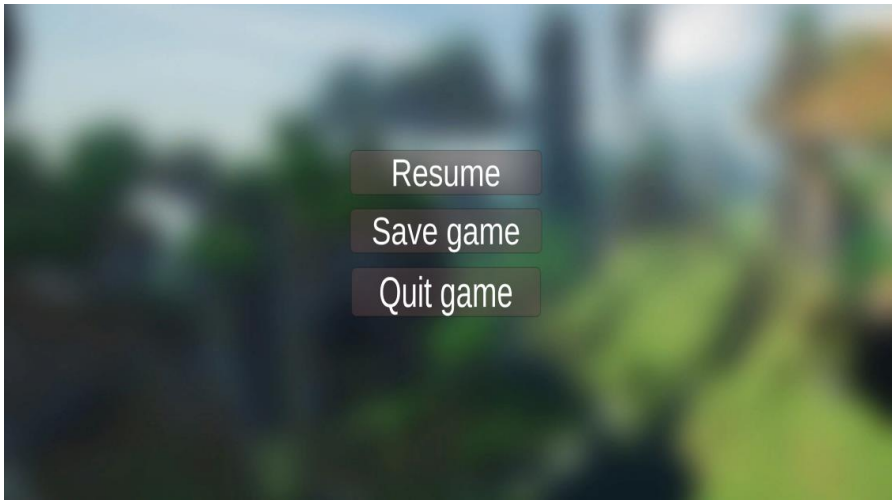
Η διασύνδεση του χρήστη(User Interface) στο παιχνίδι είναι κρίσημη για την καλύτερη εμπειρία του παίχτη.

- **Κύριο Μενού**: Το κύριο μενού είναι το πρώτο πράγμα που βλέπει ο χρήστης όταν ανοίξει το παιχνίδι. Οι επιλογές που περιλαμβάνει είναι το καινούργιο παιχνίδι, η φόρτωση ενός αποθηκευμένου παιχνιδιού και η έξοδος από το παιχνίδι.

- **Οθόνη παιχνιδιού:** Η οθόνη του παιχνιδιού προβάλλει τον κόσμο του παιχνιδιού όπως τον βλέπει ο Remi. Περιλαμβάνει πληροφορίες όπως τον αριθμό των τροφίμων και του νερού καθώς και τον εξοπλισμό του παίχτη μας και τυχόν αντικείμενα που έχει βρει.

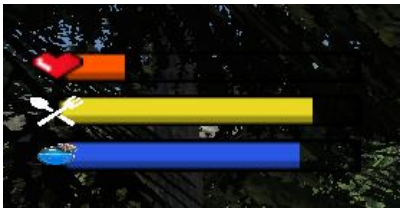


- **In-Game menu:** Το In-Game menu επιτρέπει στον παίχτη να κάνει ένα διάλλειμα από το παιχνίδι, να το κλείσει ή να κάνει Save στο σημείο που έχει φτάσει.

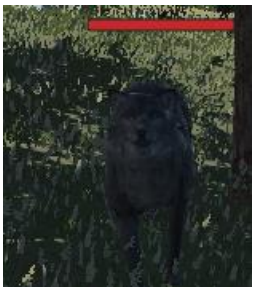


➢ **Ειδοποιήσεις:**

- **Παίχτης:** Εαν η κατάσταση της υγείας ή του νερού ή της τροφής το παίχτη φτάσει σε κρίσιμο σημείο για εκείνον τότε το χρώμα της μπάρας αλλάζει ειδοποιόντας τον ότι πρέπει να κάνει κάτι για να τις γεμίσει.



➢ **NPC- ΛΥΚΟΙ:** Εάν ο παίχτης πλησιάσει του λύκους τότε πάνω από τους λύκους εμφανίζεται μια μπάρα ειδοποιόντας τον ότι ο λύκος τον έχει καταλάβει καθώς δείχνει και την ζωή του λύκου.



5. Σχεδίαση της εφαρμογής

Στο στάδιο της σχεδίασης, επικεντρωθήκαμε κυρίως στα παρακάτω:

- **Σχεδίαση χάρτη**: Δημιουργήσαμε τον χάρτη του παιχνιδιού, όπως προείπαμε, περιλαμβάνοντας τη διαμόρφωση του νησιού, τη θέση των αντικειμένων και τον τρόπο που συνδέονται μεταξύ τους.
    - **Σχεδίαση νησιού:** Το παιχνίδι επικεντρώνεται σε ένα απομονωμένο νησί. Ο χάρτης σχεδιάστηκε με προσοχή ώστε να δημιουργήσει μια αίσθηση απομόνωσης και εξερεύνησης. Το νησί περιλαμβάνει δάση, βουνά, παραλίες και κρυφές σπηλιές για να προσφέρει ποικιλία και ενδιαφέρον στον παίχτη.
    - **Τοποθεσία αντικειμένων:** Τα αντικείμενα που απαιτούνται για την επιβιώση του παίχτη, όπως φαγητό και νερό, βρίσκονται στον χάρτη.
    - **Χωριό:** Στον χάρτη υπάρχει ένα χωρίο, όπου ο παίχτης μπορεί να μάθει για τον κύριο στόχο του παιχνιδιού. Το χωριό είναι ένα κεντρικό σημείο του χάρτη και αποτελεί ένα καλό σημείο αναφοράς.
    - **Μονοπάτια:** Για να διευκολύνεται η κίνηση του παίχτη στον χάρτη, υπάρχουν μονοπάτια που συνδέουν διάφορες περιοχές του νησιού.

- **Σχεδίαση διεπαφής χρήστη:** Δημιουργήσαμε τη διεπαφή του χρήστη που παρουσιάζει στον παίχτη τις πληροφορίες που χρειάζεται, όπως το υπόλοιπο φαγητό και νερό καθώς και άλλα στοιχεία που μπορεί να χρειαστεί.
  - **Κεντρικό Μενού:** Η διεπαφή ξεκινά με ένα κεντρικό μενού, όπου ο παίκτης μπορεί να ξεκινήσει ένα νέο παιχνίδι, να φορτώσει ένα αποθηκευμένο παιχνίδι, ή να βγει από το παιχνίδι.
  - **Mini-map**: Στην γωνία της οθόνης, υπάρχει ένας μικρός χάρτης που παρουσλιαζει το σημείο όπου βρίσκεται ο παίχτης σε σχέση με τον περίγυρο του.

    
  - **Εισαγωγική Οθόνη:** Πριν από την έναρξη του παιχνιδιού, υπάρχει μια εισαγωγική οθόνη που παρουσιάζει τον τίτλο του παιχνιδιού και ένα σύντομο βίντεο με την συντιβή του αεροσκάφους του χαρακτήρα μας.
  - **Στοιχεία Επιβίωσης:** Κατά τη διάρκεια του παιχνιδιού, ο παίκτης βλέπει πληροφορίες σχετικά με την κατάσταση του χαρακτήρα του, όπως το επίπεδο υγείας, τη δίψα και την πείνα. Αυτά τα στοιχεία αναπαρίστανται ως μπάρες.
  - **Πληροφορίες Αντικειμένων:** Όταν ο παίκτης συλλέγει αντικείμενα, όπως φαγητό ή νερό, εμφανίζονται πληροφορίες σχετικά με τον τύπο και την ποσότητα των αντικειμένων που κατέχει.
  - **Χάρτης:** Ένας χάρτης του νησιού εμφανίζεται στην οθόνη, με τη θέση του παίχτη.

- **Κουμπιά και Ενέργειες:** Κουμπιά εμφανίζονται στην οθόνη για να επιτρέψουν στον παίκτη να εκτελέσει διάφορες ενέργειες, όπως να ανοίξει το αποθηκευμένο μενού, να συλλέξει αντικείμενα ή να ρίξει αντικείμενα από τον εξοπλισμό του πχ. βέλη.

- **Σχεδίαση περιβάλλοντος:** Δημιουργήσαμε το περιβάλλον του παιχνιδιού περιλαμβάνοντας το νησί, τα αντικείμενα, του χαρακτήρες και τα animation για να δώσουμε ζωή στο παιχνίδι μας.
  - **Φυσικά Χαρακτηριστικά:** Στο νησί υπάρχουν διάφορα φυσικά χαρακτηριστικά, όπως ποτάμια, λίμνες, σπηλιές, καταρράκτες και κορυφές βουνών. Αυτά τα χαρακτηριστικά προσφέρουν ευκαιρίες για αλληλεπιδράσεις και εξερεύνηση.
  - **Εχθροί και Κίνδυνοι:** Το περιβάλλον είναι γεμάτο με εχθρούς και κινδύνους. Οι λύκοι μπορούν να γίνουν αρκετά επικίνδυνοι και να τραυματίσουν τον παίχτη εάν δεν προσέξει αρκετά.
  - **Πόροι:** Ο παίκτης πρέπει να συλλέγει πόρους από το περιβάλλον για την επιβίωσή του, όπως φρούτα και νερό.
  - **Κατανομή Πόρων:** Η κατανομή των πόρων είναι λογική και ρεαλιστική, προκειμένου να απαιτείται προσπάθεια για την εύρεση τους.
  - **Εικόνα και Ήχος:** Η γραφική αισθητική και ο ήχος του περιβάλλοντος πρέπει να δημιουργούν μια ατμόσφαιρα επιβίωσης.

Screenshot για την σχεδίαση της εφαρμογής υπάρχουν στο προηγούμενο κεφάλαιο (**Προδιαγραφές, ανάλυση απαιτήσεων και ανάλυση της εφαρμογής**)

6. Υλοποίηση της εφαρμογής

Για την υλοποίηση χρειάστηκε η σωστή δόμηση του παιχνιδιού έτσι ώστε να υπάρχει η σωστή λειτουργία όλων των μερών του προγράμματος.

- **Αρχιτεκτονική:** Αρχικά, αφιερώσαμε χρόνο στον σχεδιασμό του παιχνιδιού, χρησιμοποιώντας κατάλληλα Asset για να δημιουργήσουμε έναν μαγευτικό κόσμο στο νησί μας. Στη συνέχεια, δημιουργήσαμε προσαρμοσμένα scripts (που βρίσκονται στον φάκελο 'scripts'), επιτρέποντας στον παίκτη να εξερευνήσει ελεύθερα τον χώρο του νησιού, να αποθηκεύει και να φορτώνει το παιχνίδι του, να το εγκαταλείπει και να αλληλοεπιδρά με τα αντικείμενα που βρίσκονται σκόρπια σε όλο το νησί. Τέλος, προσθέσαμε τους κατάλληλους ήχους, όπως οι βηματισμοί του χαρακτήρα μας και ο ήχος του καταρράκτη, προσθέτοντας μια επιπλέον διάσταση στην ατμόσφαιρα του παιχνιδιού μας
- **Scripts:**

loadGameMenu.cs	Αυτό το script χρησιμοποιείται έτσι ώστε ο χρήστης να μπορεί να κάνει load ένα από τα τρια save που μπορεί να υπάρχουν
newGameMenu.cs	Με αυτό το script ο χρήστης μπορεί να κάνει νέο save στο παιχνίδι.
playerData.cs	Έδω υπάρχουν τα δεδομένα που θα αποθηκευτούν όταν ο χρήστης πατήσει save ή φτάσει σε κάποιο checkpoint.
saveSystem.cs	Αυτό το script χρησιμοποιείται για να δημιουργηθεί το save ή να πάρει τον φάκελο του load του παιχνιδιού.

AxeCollider.cs	Όταν ο χρήστης πατήσει το αριστερό click στο ποντίκι για να κάνει επίθεση τότε το script ελέγχει εάν υπάρχει σε κοντινή απόσταση λύκος και του κάνει damage.
Backpack.cs	Εάν ο χρήστης βρίσκεται μέσα στο collider του σακίδιου τότε αυτό το script εμφανίζει στον χρήστη το κατάλληλο μήνυμα για να το μαζέψει.
Checkpoint.cs	Άυτο το script λειτουργεί όταν ο χρήστης πατήσει το “m” έτσι ώστε να του ανοίξει ο χάρτης και να του δείξει το επόμενο σημείο του χάρτη που πρέπει να πάει.
checkpointController.cs	Αυτό το script ελέγχει σε ποιο checkpoint έχει φτάσει ο χρήστης
collectRemovedItem.cs	Εάν ο χρήστης βγάλει ένα αντικείμενο από το inventory του τότε μέσω αυτού του script, του δίνεται η δυνατότητα να μαζέψει το πεταμένο αντικείμενο.
healthBar.cs-energyBar.cs-waterBar.cs	Τα τρια αυτά script ελέγχουν τα slider για το νερο, την τροφή και την ζωή του χαρακτήρα.
EnergyController.cs	Αυτό το script παιρνει ενέργεια από τον χρήστη κάθε λίγα δευτερόλεπτα και δίνει ενέργεια στον χρήστη εάν εκείνος καταναλώσει τροφή.
FallDamage.cs	Εάν ο χρήστης πέσει από κάποιο μεγάλο ύψωμα τότε το script αυτό μειώνει την ζωή του χρήστη αναλόγως το ύψος.

getGems.cs	Εάν ο χρήστης έχει μαζέψει κάποιο από τα διαμάντια τότε το script κάνει ανενεργό το διαμάντι αυτό και ενεργοποιεί το επόμενο.
HealthController.cs	Εάν ο χρήστης έχει δεχτεί ζημιά τότε το script αυτό επιστρέφει την ζώη του κάθε λίγα δευτερόλεπτα.
holdingItems.cs	Ελέγχει εάν ο χρήστης έχει το τσεκούρι ή το τόξο στην κατοχή του.
Items.cs	Εάν ο χρήστης είναι μέσα στην εμβέλεια του collider των δέντρων με φρούτα, του δίνεται η δυατότητα να τα προσθέσει το inventory του.
LoadScene.cs	Όταν ο χρήστης πατήσει να ξεκινήσει το παιχνίδι τότε το script αυτό του εμφανίζει ένα sclider με το ποσοστό ετοιμότητας του παιχνιδιού.
Map.cs	Το script αυτό κινεί τον χαρακτήρα πάνω στον χάρτη ώστε όταν ο παίχτης πατήσει το “m” να του δείχνει την τοποθεσία του.
MyPlayerAnimations.cs	Στο scipt αυτό ελέγχονται οι κινήσεις το παίχτη έτσι ώστε να γίνουν τα κατάλληλα animation πχ. Walk ή jump.
Opener.cs	Εάν ο χρήστης βρίσκεται κοντά στην πόρτα του ενός σπιτιού μέσα στο χωριό τότε του εμφανίζεται το κατάλληλο μήνυμα και εάν πατήσει το “e” τότε οι πόρτες του σπιτιού κάνουν το κατάλληλο animation.

openInventory.cs	Εάν ο χρήστης πατήσει το “tab” τότε του ανοίγει το inventory έτσι ώστε να μπορεί να πιεί νερό, να φάει ή να δει πόσα τόξα έχει στην κατοχή του.
Pausemenu.cs	Εάν ο χρήστης πατήσει το “escape” τότε του εμφανίζει ένα menu στο οποίο μπορεί να κάνει save ή resume ή να βγεί από το παιχνίδι.
PlayerSwimInLake.cs-PlayerSwimInSea.cs	Εάν ο χαρακτήρας μπεί μέσα στο ποτάμι ή στην θάλασσα τότε η μεταβλητή swimming γίνεται true και κάνει τα κατάλληλα animation.
ReadBook.cs	Το script αυτό χρησιμοποιείται όταν ο χρήστης πατήσει να διαβάσει το βιβλίο για να μπορεί να αλλάζει σελίδα ή να το κλείσει.
startOfGameCinematic.cs	Εάν ο χρήστης πατήσει new game, τοτε μέσω ενός timeline, ξεκινάει το παιχνίδι με τον παίχτη να ξυπνάει στην παραλία του νησιού.
Timeline.cs	Όταν ο χαρακτήρας αποκτήσει το τελευταίο διαμάντι, ξεκινάει ένα timeline δείχνοντας τον χαρακτήρα να φεύγει από νησί με μια βάρκα.
waterBottle.cs	Εάν ο παίχτης έχει πάρει το μπουκάλει τότε μπορεί μέσω του inventory να πίει νερό. Μέσω αυτου το script, όταν ο παίχτης πατήσει το νερο στο inventory, τότε το waterBar ανεβαίνει.
waterController.cs	Μέσω του script αυτού, το νερό του παίχτη πέφτει κάθε λίγα δευτερόλεπτα.

wolvesEnabled.cs	Όταν ο χαρακτήρας βρεθεί σε κοντινή απόσταση από τους λύκους, τότε οι λύκοι γίνονται ενεργοί. Το script αυτό υλοποιήθηκε για να μην χρειάζονται πολλοι πόροι.
Aim.cs	Εάν ο χρήστης έχει συλλέξει το τόξο τότε αυτό το script χρησιμοποιείται έτσι ώστε ο χρήστης να μπορεί να συμαδέψει.
Arrow.cs	Εάν ο χρήστης έχει ρίξει ένα βέλος τότε μπορει να το συλλέξει ξανά μέσω αυτού του script.
ArrowCollider.cs	Εάν το βέλος που ρίξει ο χρήστης πετύχει κάποιο λύκο τότε ο λύκος πεθαινει και το βέλος μένει πάνω στον λύκο. Το script αυτό χρησιμέυει για να γίνει αυτή η λειτουργία.
ArrowCollision.cs	Το script αυτό κοιτάει ένα ένα βέλος κάνει collide με τους λύκους ή κάποιο δέντρο και σταματάει την κίνηση του.
ArrowHolder.cs	Το script αυτό χρησιμοποιείται όταν ο χρήστης βρεί βέλοι έτσι ώστε να μπορεί να τα συλλέξει.
holdingBow.cs	Το script αυτό χρησιμοποιείται όταν ο χρήστης βρεί το τόξο έτσι ώστε να μπορεί να το συλλέξει.

➢ Αναλυτικά τα σημαντικότερα scripts:

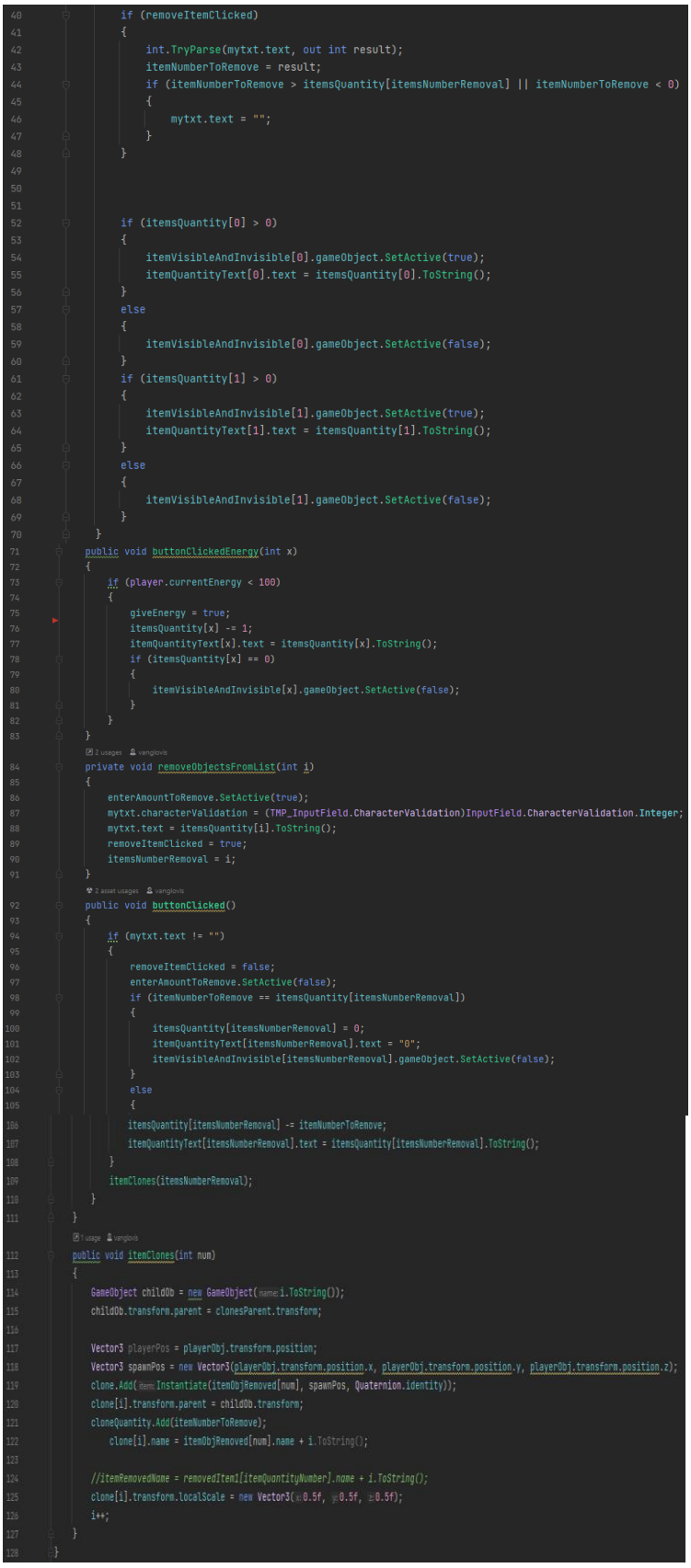
- **itemController:**

  Το script αυτό, ασχολείται με την συλλογή τροφής και την εισαγωγή τους στο inventory. Αρχικά αποθηκεύουμε σε λίστες τα απαραίτητα στοιχεία πχ. την ποσότητα των μήλων που έχει ο χρήστης.

  Επίσης, εάν ο χρήστης χρησιμοποιήσει τροφή από το inventory, τότε στο script μας μείωνεται η ποσότητα της τροφής και η μεταβλητή giveEnergy=true ώστε να μπεί στο energyController και να ανέβει η μπάρα της ενέργειας του παίχτη.

  Τέλος, εάν ο χαρακτήρας θελήσει να ρίξει τροφή από το inventory, τοτε μέσω αυτού του script μειώνεται η ποσότητα της τροφης που έριξε ο χρήστης και δημιουργούνται κλώνοι της τροφής. Πχ εάν ο χαρακτήρας ριξει τρια μήλα, τότε θα εμφανιστεί ένα μήλο στον χάρτη, με ποσότητα τρια, το οποίο θα μπορεί να το επιλέξει και να το ξαναβάλει στο inventory.

```
public class itemController : MonoBehaviour
{
    int i = 0;
    public int itemsNumberRemoval,itemNumberToRemove;
    public TMP_InputField mytxt;
    public GameObject enterAmountToRemove,clonesParent,playerObj;
    public static bool giveEnergy = false,removeItemClicked=false;
    public static List<int> itemsQuantity = new List<int>();
    public List<TMP_Text> itemQuantityText= new List<TMP_Text>();
    public List<Button> itemVisibleAndInvisible = new List<Button>();
    public List<Button> removeItem = new List<Button>();
    static List<GameObject> clone = new List<GameObject>();
    public  List<GameObject> itemObjRemoved = new List<GameObject>();
    public static List<int>  cloneQuantity = new List<int>();
    // Start is called before the first frame update
    void Start()
    {
        itemsQuantity.Add(0);
        itemsQuantity.Add(0);
        for(int x = 0; x < 2; x++)
        {
            itemVisibleAndInvisible[x].gameObject.SetActive(false);
        }
        itemVisibleAndInvisible[0].onClick.AddListener(() => buttonClickedEnergy(0));
        itemVisibleAndInvisible[1].onClick.AddListener(() => buttonClickedEnergy(1));
        removeItem[0].onClick.AddListener(() => removeObjectsFromList(0));
        removeItem[1].onClick.AddListener(() => removeObjectsFromList(1));
    }

    // Update is called once per frame
    void Update()
    {
```



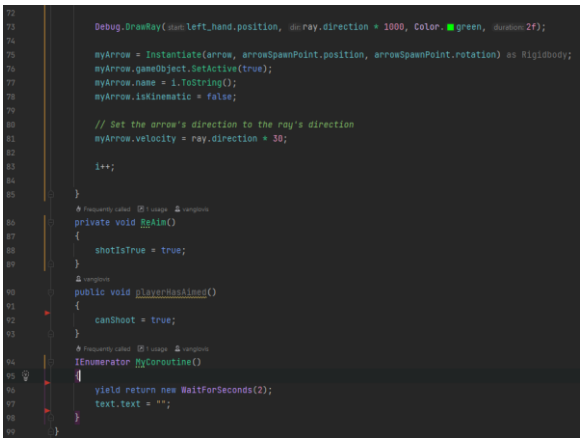
```
        if (removeItemClicked)
        {
            int.TryParse(mytxt.text, out int result);
            itemNumberToRemove = result;
            if (itemNumberToRemove > itemsQuantity[itemsNumberRemoval] || itemNumberToRemove < 0)
            {
                mytxt.text = "";
            }
        }




        if (itemsQuantity[0] > 0)
        {
            itemVisibleAndInvisible[0].gameObject.SetActive(true);
            itemQuantityText[0].text = itemsQuantity[0].ToString();
        }
        else
        {
            itemVisibleAndInvisible[0].gameObject.SetActive(false);
        }
        if (itemsQuantity[1] > 0)
        {
            itemVisibleAndInvisible[1].gameObject.SetActive(true);
            itemQuantityText[1].text = itemsQuantity[1].ToString();
        }
        else
        {
            itemVisibleAndInvisible[1].gameObject.SetActive(false);
        }
    }
    public void buttonClickedEnergy(int x)
    {
        if (player.currentEnergy < 100)
        {
            giveEnergy = true;
            itemsQuantity[x] -= 1;
            itemQuantityText[x].text = itemsQuantity[x].ToString();
            if (itemsQuantity[x] == 0)
            {
                itemVisibleAndInvisible[x].gameObject.SetActive(false);
            }
        }
    }
    2 usages  vanglovis
    private void removeObjectsFromList(int i)
    {
        enterAmountToRemove.SetActive(true);
        mytxt.characterValidation = (TMP_InputField.CharacterValidation)InputField.CharacterValidation.Integer;
        mytxt.text = itemsQuantity[i].ToString();
        removeItemClicked = true;
        itemsNumberRemoval = i;
    }
    2 asset usages  vanglovis
    public void buttonClicked()
    {
        if (mytxt.text != "")
        {
            removeItemClicked = false;
            enterAmountToRemove.SetActive(false);
            if (itemNumberToRemove == itemsQuantity[itemsNumberRemoval])
            {
                itemsQuantity[itemsNumberRemoval] = 0;
                itemQuantityText[itemsNumberRemoval].text = "0";
                itemVisibleAndInvisible[itemsNumberRemoval].gameObject.SetActive(false);
            }
            else
            {
                itemsQuantity[itemsNumberRemoval] -= itemNumberToRemove;
                itemQuantityText[itemsNumberRemoval].text = itemsQuantity[itemsNumberRemoval].ToString();
            }
            itemClones(itemsNumberRemoval);
        }
    }
    1 usage  vanglovis
    public void itemClones(int num)
    {
        GameObject childOb = new GameObject(name:i.ToString());
        childOb.transform.parent = clonesParent.transform;

        Vector3 playerPos = playerObj.transform.position;
        Vector3 spawnPos = new Vector3(playerObj.transform.position.x, playerObj.transform.position.y, playerObj.transform.position.z);
        clone.Add(item:Instantiate(itemObjRemoved[num], spawnPos, Quaternion.identity));
        clone[i].transform.parent = childOb.transform;
        cloneQuantity.Add(itemNumberToRemove);
            clone[i].name = itemObjRemoved[num].name + i.ToString();

        //itemRemovedName = removedItem1[itemQuantityNumber].name + i.ToString();
        clone[i].transform.localScale = new Vector3(x:0.5f, y:0.5f, z:0.5f);
        i++;
    }
}
```

- **bowController**

  Το bowController έχει ως σκοπό, να ετοιμάζει τον χαρακτήρα έτσι ώστε να ρίχνει βέλη. Στις πρώτες γραμμές βλέπουμε να ελέγχει, εάν ο παίχτης πατάει το δεξί κλικ και ένα ο χαρακτήρας κρατάει το τόξο ώστε να συμαδέψει. Εάν αυτά είναι true και ο χαρακτήρας έχει βέλη και πατήσει το αριστερό κλικ τότε γίνεται το animation και ρίχνει το βέλος.

  Όταν μπεί στο ShootArrow(), τότε δημιουργείται κλώνος του βέλους το οποίο ταξιδεύει προς το μέρος που έχει σημαδέψει ο παίχτης.

```
public class bowController : MonoBehaviour
{
    public Text text;  Changed in 2+ assets
    public GameObject arrowsCanvas,aimImage,rigAim;  Changed in 2 assets
    public MonoBehaviour rigAimEnabled;  Changed in 2 assets
    public Rigidbody arrow;  Changed in 2 assets
    public Transform left_hand;  Changed in 2 assets
    public Transform arrowSpawnPoint;  Changed in 2 assets
    public static int arrowCounter=0;
    public static Rigidbody myArrow;
    private float horzMovement;
    private float prevSpineX = 64.7f;
    private int i = 0;
    public static bool shotIsTrue = false;
    private bool canShoot = false;

    Event function  vanglovis
    void Start()
    {
        prevSpineX = 64.7f;
    }
    Event function  vanglovis
    void Awake() {
        prevSpineX = 64.7f;
    }
    Event function  vanglovis *
    void Update()
    {
        if (arrowCounter > 0)
        {
            arrowsCanvas.gameObject.SetActive(true);
        }
        horzMovement = -Input.GetAxis("Mouse Y")* Time.deltaTime * 50f;
        if (Input.GetKey(KeyCode.Mouse1) &&  holdingItems.holdBowIsActive)
        {
          rigAimEnabled.enabled = true;
          aimImage.SetActive(true);
          shotIsTrue = false;
          if (Input.GetKeyDown(KeyCode.Mouse0) && canShoot)
          {
              canShoot = false;
              Debug.Log(arrowCounter);
              if (arrowCounter > 0)
              {
                  ShootArrow();
                  ReAim();
                  arrowCounter--;
              }
              else
              {
                  text.text = "No ammo";
                  StartCoroutine(routine:MyCoroutine());
              }
          }
        }
        else
        {
            canShoot = false;
            rigAimEnabled.enabled = false;
            aimImage.SetActive(false);
        }
    }
    Frequently called  1 usage  vanglovis
    private void ShootArrow()
    {
        // Get the center of the screen in world coordinates
        Vector3 screenCenter = new Vector3(x:Screen.width / 2f, y:Screen.height / 2f, z:0f);

        // Convert the screen center point to a ray
        Ray ray = Camera.main.ScreenPointToRay(screenCenter);
```

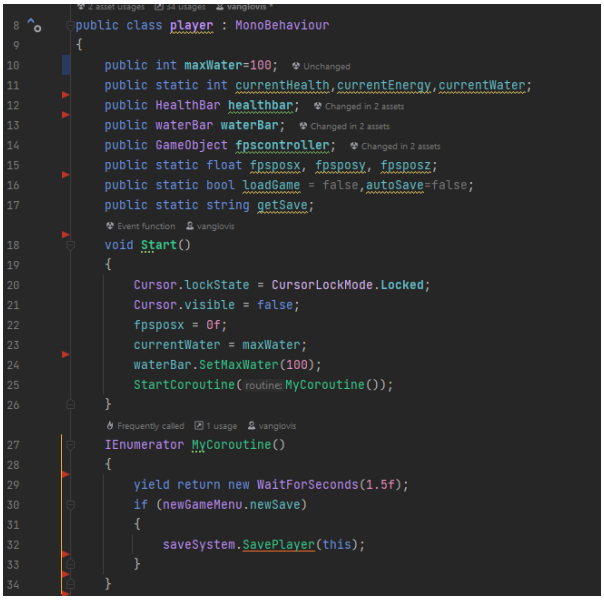


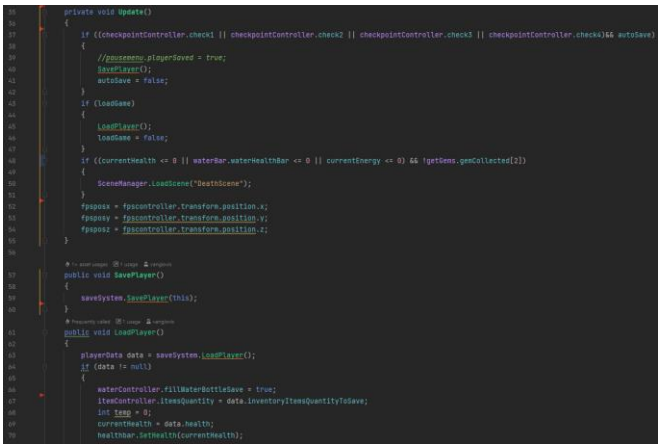
- **player.cs**

  Το συγκεκριμένο script και ίσως το σημαντικότερο στο παιχνίδι, κάνει το ποντίκι να μην φαίνεται όταν φορτωθεί ο χάρτης, ελέγχει εάν ο παίχτης φτάσει σε κάποιο από τα checkpoint και κάνει save το παιχνίδι και κάνει load όλα τα αντικείμενα στο παιχνίδι.

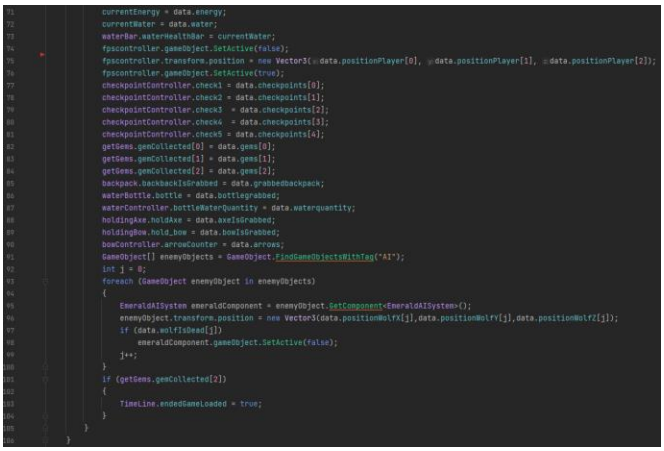
  Επίσης, εάν ο χρήστης ξεμείνει από ζωή ή ενέργεια ή νερό, φορτώνει το DeathScene.

  Μέσα στην LoadPlayer(), παίρνει όλες τις μεταβλητές που έχουν γίνει save και τις φορτώνει στον χάρτη. Πχ τον αριθμό των μήλων που έχει μαζέψει ο παίχτης, το checkpoint που έχει φτάσει και ποια διαμάντια έχει μαζέψει.





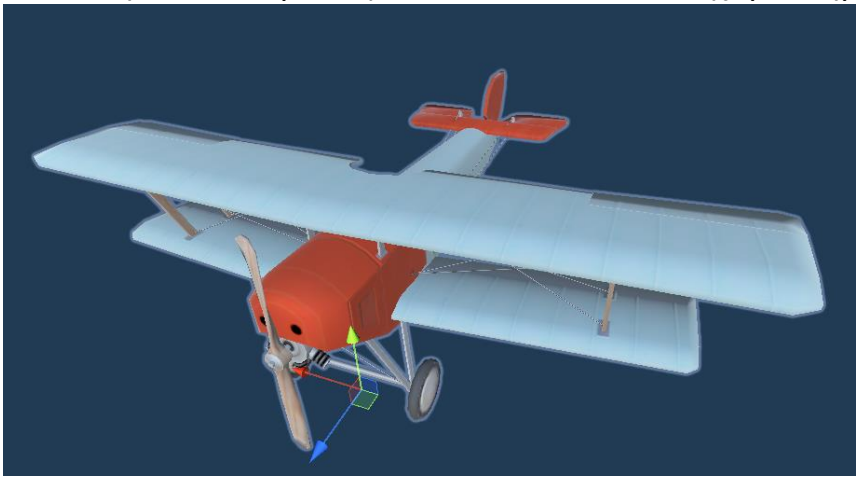
```
private void Update()
{
    if ((checkpointController.check1 || checkpointController.check2 || checkpointController.check3 || checkpointController.check4)&& autoSave)
    {
        //pauseenemy.playerSaved = true;
        SavePlayer();
        autoSave = false;
    }
    if (loadGame)
    {
        LoadPlayer();
        loadGame = false;
    }
    if ((currentHealth <= 0 || waterBar.waterHealthBar <= 0 || currentEnergy <= 0) && !getGems.gemCollected[2])
    {
        SceneManager.LoadScene("DeathScene");
    }
    fpsposx = fpscontroller.transform.position.x;
    fpsposy = fpscontroller.transform.position.y;
    fpsposz = fpscontroller.transform.position.z;
}

public void SavePlayer()
{
    saveSystem.SavePlayer(this);
}
public void LoadPlayer()
{
    playerData data = saveSystem.LoadPlayer();
    if (data != null)
    {
        waterController.fillWaterBottleSave = true;
        itemController.itemsQuantity = data.inventoryItemsQuantityToSave;
        int temp = 0;
        currentHealth = data.health;
        healthbar.SetHealth(currentHealth);
```



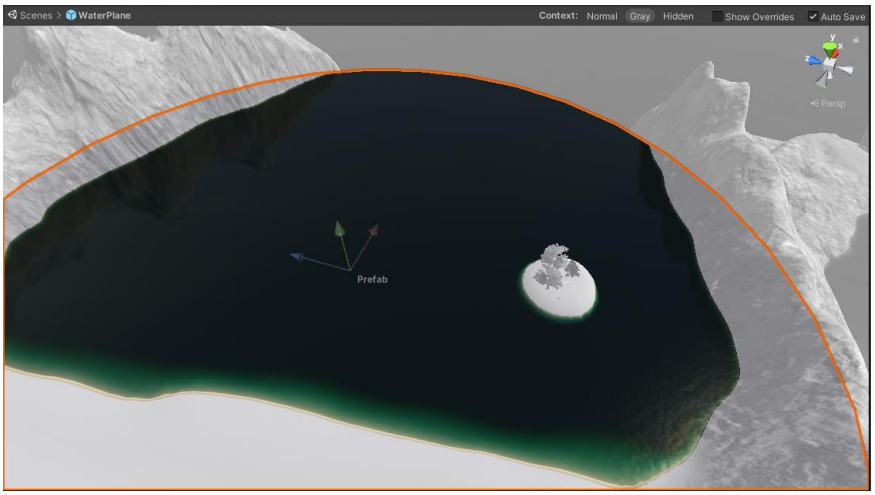
```
        currentEnergy = data.energy;
        currentWater = data.water;
        waterBar.waterHealthBar = currentWater;
        fpscontroller.gameObject.SetActive(false);
        fpscontroller.transform.position = new Vector3(x:data.positionPlayer[0], y:data.positionPlayer[1], z:data.positionPlayer[2]);
        fpscontroller.gameObject.SetActive(true);
        checkpointController.check1 = data.checkpoints[0];
        checkpointController.check2 = data.checkpoints[1];
        checkpointController.check3  = data.checkpoints[2];
        checkpointController.check4  = data.checkpoints[3];
        checkpointController.check5 = data.checkpoints[4];
        getGems.gemCollected[0] = data.gems[0];
        getGems.gemCollected[1] = data.gems[1];
        getGems.gemCollected[2] = data.gems[2];
        backpack.backpackIsGrabbed = data.grabbedbackpack;
        waterBottle.bottle = data.bottlegrabbed;
        waterController.bottleWaterQuantity = data.waterquantity;
        holdingAxe.holdAxe = data.axeIsGrabbed;
        holdingBow.hold_bow = data.bowIsGrabbed;
        bowController.arrowCounter = data.arrows;
        GameObject[] enemyObjects = GameObject.FindGameObjectsWithTag("AI");
        int j = 0;
        foreach (GameObject enemyObject in enemyObjects)
        {
            EmeraldAISystem emeraldComponent = enemyObject.GetComponent<EmeraldAISystem>();
            enemyObject.transform.position = new Vector3(data.positionWolfX[j],data.positionWolfY[j],data.positionWolfZ[j]);
            if (data.wolfIsDead[j])
                emeraldComponent.gameObject.SetActive(false);
            j++;
        }
        if (getGems.gemCollected[2])
        {
            TimeLine.endedGameLoaded = true;
        }
    }
}
```

➢ ASSETS:

- **Air_biplane_A:** Χρησιμοποιήθηκε το asset αυτό, για να φαίνεται στο νησί ο τρόπος που έπεσε ο χαρακτήρας μας.



- **WaterPlane:** Το συγκεκριμένο asset είναι η θάλλασα του χάρτη μας έτσι ώστε το νησί μας να προσφέρει μια αληθοφάνεια.

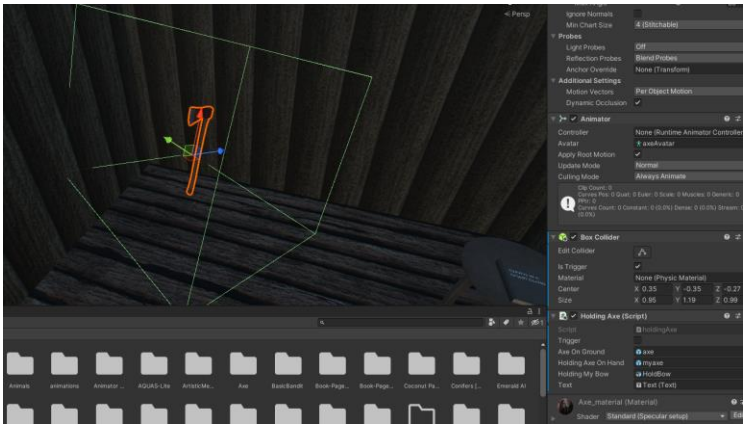


- **OldHouse:** Το σπίτι αυτό χρησιμοποιήθηκε έτσι ώστε ο παίχτης να μπορεί να ανοίγει τις πόρτες, να μπαίνει μέσα και να μπορεί να συλλέξει το τσεκούρι που υπάρχει στον χάρτη μας αλλά και να διαβάσει το βιβλίο που του εξηγεί τι πρέπει να κάνει για να ξεφύγει από το νησί.

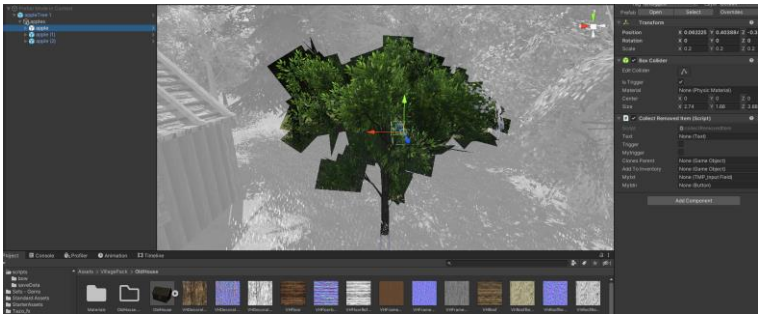
- **Γραφείο και βιβλίο:**

  Τα δύο αυτά asset υπάρχουν έτσι ο χρήστης να μπορεί να μάθει τι γίνεται στο παιχνίδι μας όπως έχουμε προαναφέρει.

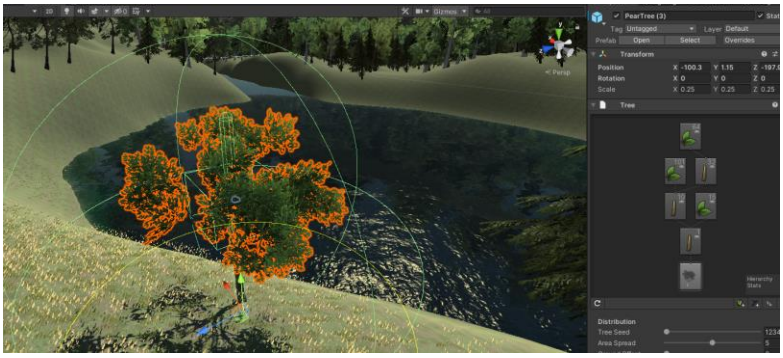
- **Τσεκούρι:** Το asset αυτό βρίσκεται στο σπίτι που περιγράψαμε από πάνω και έχει colliders έτσι ώστε ο χρήστης να μπορεί να το σηκώσει.



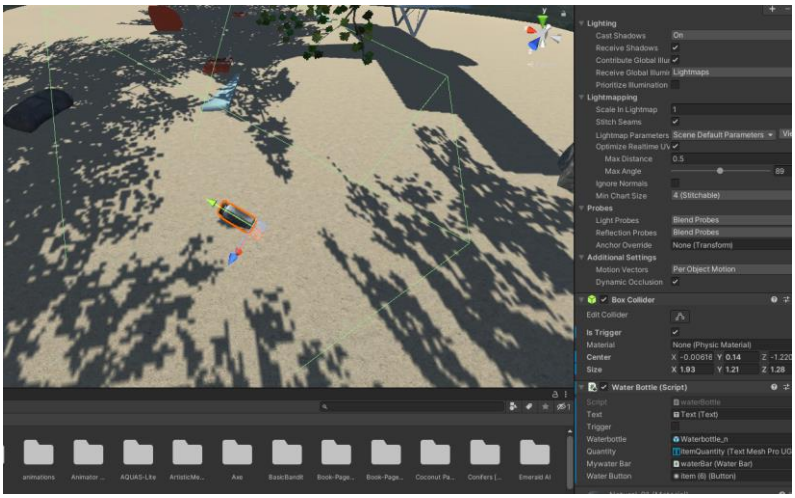
- **Apple Trees:** Στο συγκεκριμένο asset έχουν προστεθεί colliders τόσο στα δέντρα για να μπορεί ο χρήστης να μαζέψει μήλα, όσο και στα μήλα για να βλέπουμε την ποσότητα των μήλων που θα μαζέψει.



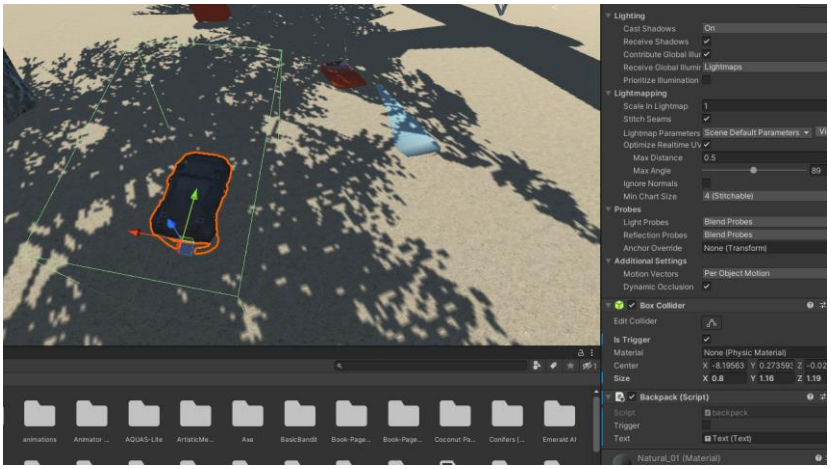
- **Pear Trees:** Το asset αυτό κάνει την ίδια δουλεία με τα Apple Trees με την μόνη διαφορά ότι στο δέντρο αυτό έχουν προστεθεί αχλάδια αντί για μήλα.



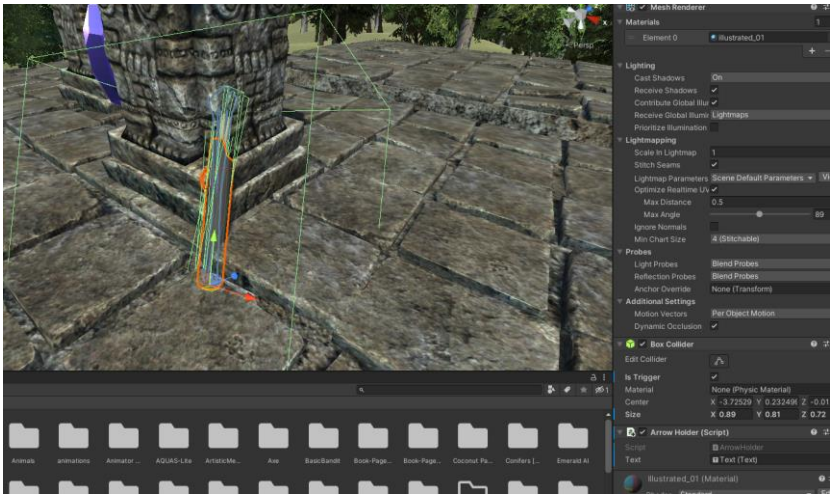
- **Μπουκάλι νερού:** Το asset αυτό είναι απαραίτητο για να μπορεί ο χρήστης να συλλέγει νερό. Υπάρχουν τα κατάλληλα colliders για να μπορεί ο χρήστης να το μαζέψει.



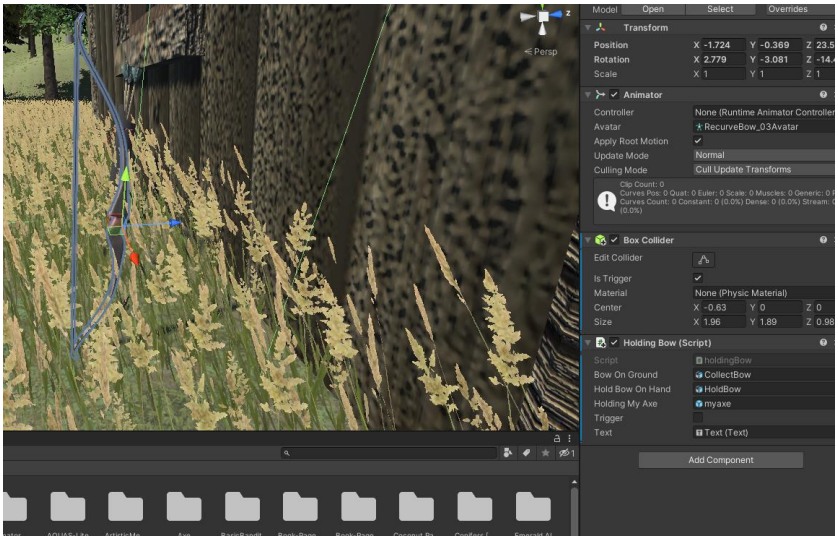
- **Τσάντα:** Η τσάντα χρησιμοποιήθηκε για να μπορεί ο χρήστης να κρατάει το μπουκάλι. Εχει τα κατάλληλα colliders έτσι ώστε ο χρήστης να μπορεί να την σηκώσει.



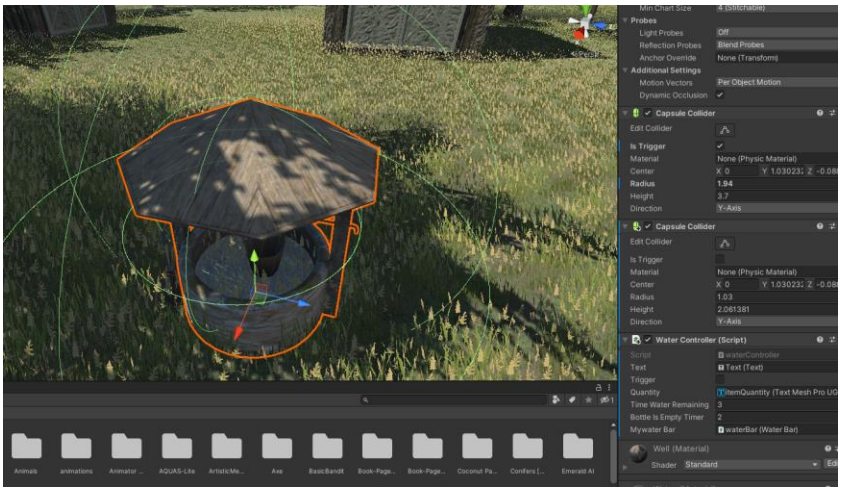
- **Βέλη:** Τα βέλη χρησιμεύουν έτσι ώστε ο χρήστης να μπορεί να πολεμήσει με τους λύκους για να πάρει τα διαμάντια. Όπως και στα προηγούμενα asset έχουν χρησιμοποιηθεί τα κατάλληλα colliders για να μπορεί ο χρήστης να τα μαζέψει.



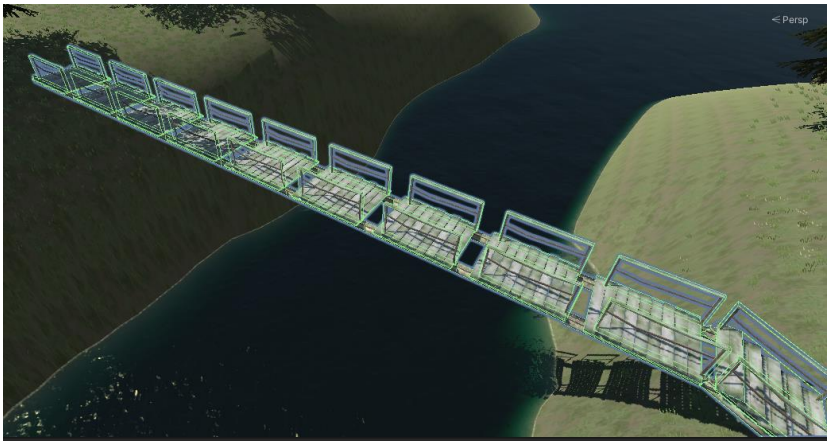
- **Τόξο:** Το τόξο υπάρχει για να μπορεί ο χρήστης να πολεμήσει τους λύκους από μακρινή απόσταση και να μην ρισκάρει να φάει damage.



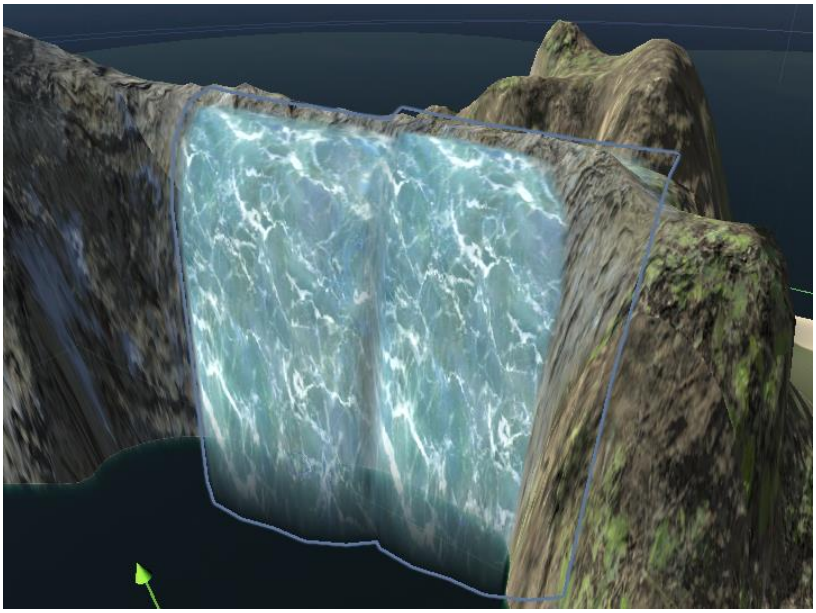
- **Πηγάδια:** Τα πηγάδια υπάρχουν έτσι ώστε ο χρήστης να μπορεί να πίνει νερό ή να γεμίζει το μπουκάλι του χρησιμοποιώντας τα κατάλληλα colliders.



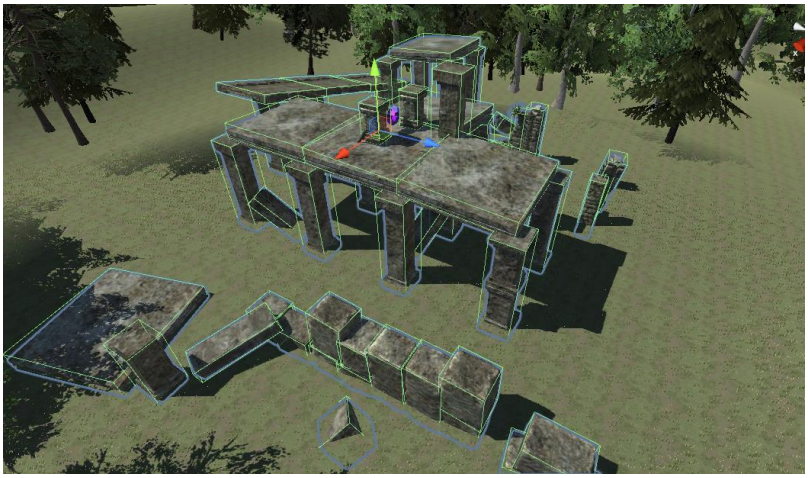
- **Γέφυρα:** Η γέφυρα χρησιμοποιήθηκε για να μπορεί ο χρήστης να πάει στην άλλη μεριά του νησιού χωρίς να κολυμπήσει καθώς σε πολλά σημεία του ποταμού είναι αδύνατον να ανέβει.



- **Καταρράκτης:** Χρησιμοποιήθηκε ο καταρράκτης για να ομορφύνει την λίμνη του νησιού.



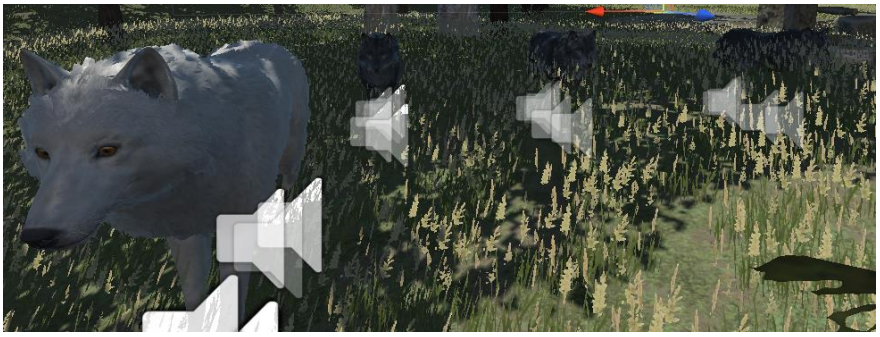
- **Ερείπια:** Τα ερείπια χρησιμοποιήθηκαν έτσι ώστε να τοποθετηθεί το δεύτερο διαμάντι του παιχνιδιού και να δείξει στον χαρακτήρα ότι το νησί κάποτε κατοικούνταν.



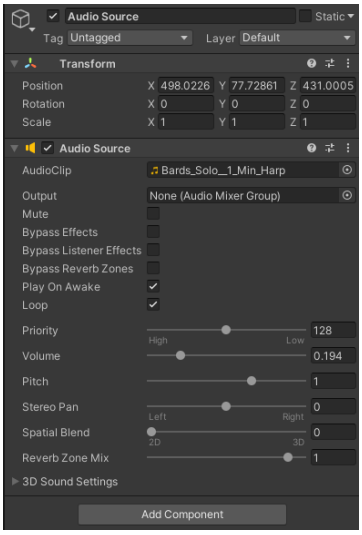
- **Χαρακτήρας:** Ο χαρακτήρας πάρθηκε από την σελίδα mixamo.com, βρίσκεται μέσα στον fpsController και εκτελεί όλα τα κατάλληλα animations τα οποία πάρθηκαν και αυτά από το mixamo.com.

- **Διαμάντια:** Τα διαμάντια είναι από τα σημαντικότερα asset του παιχνιδιού καθώς πρέπει να τα συλλέξει ο χρήστης για να τερματίσει το παιχνίδι.

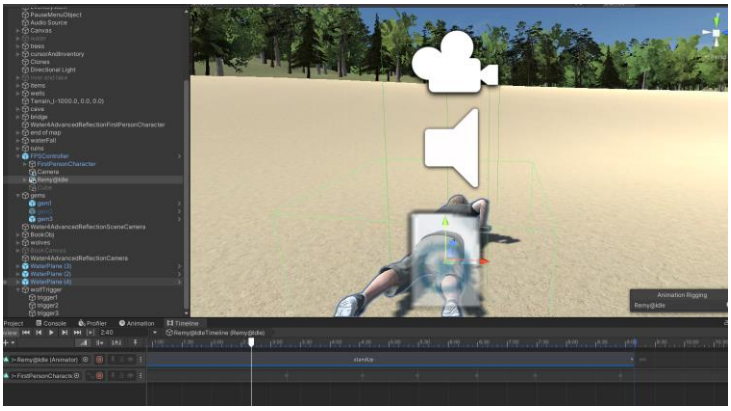
- **Λύκοι:** Οι λύκοι είναι διασκορπισμένοι σε όλο το map και λειτουργούν ως αμυντικός μηχανισμός των διαμαντιών έτσι ώστε να δυσκολέψουν τον χρήστη να τερματίσει το παιχνίδι.

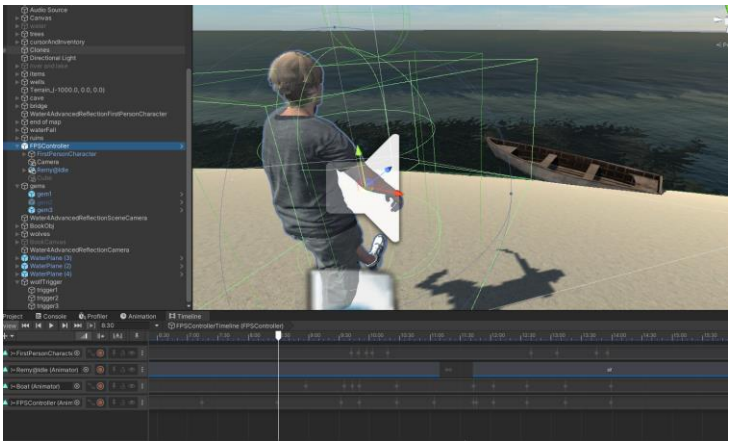


- **Μουσική:** Καθ' όλη την διάρκεια του παιχνιδιού, υπάρχει ένα τραγούδι που επιλέχθηκε από το asset store για να χαλαρώνει τον χρήστη ώστε να απολαύσει το παιχνίδι.

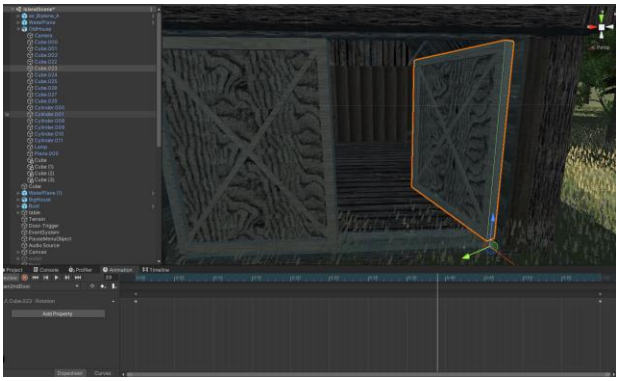


- **Βοηθητικά εργαλεία:** Χρησιμοποιήθηκαν αρκετά βοηθητικά εργαλεία όπως τα time-lines, animation editors και το Emerald AI.
    - **Time-lines:** Υπάρχουν δυο time-lines στην εφαρμογή μας. Το πρώτο βρίσκεται στην αρχή του παιχνιδιού, όπου ο παίχτης μας σηκώνεται στην παραλία και ένα στο τέλος, όπου ο παίχτης μας παίρνει το τελευταίο διαμάντι και φεύγει από το νησί.

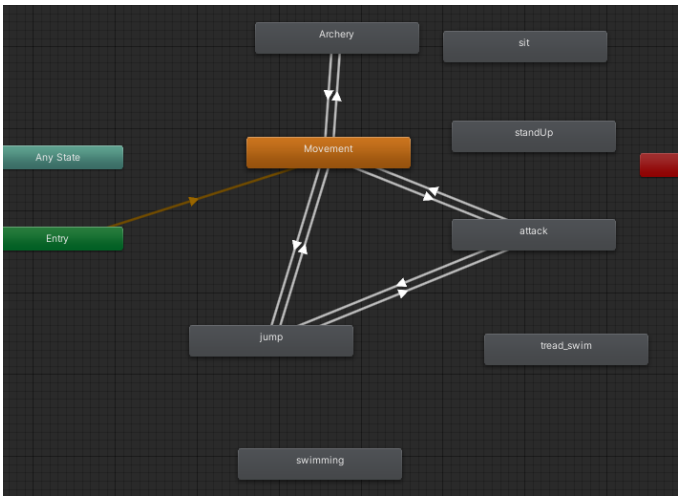


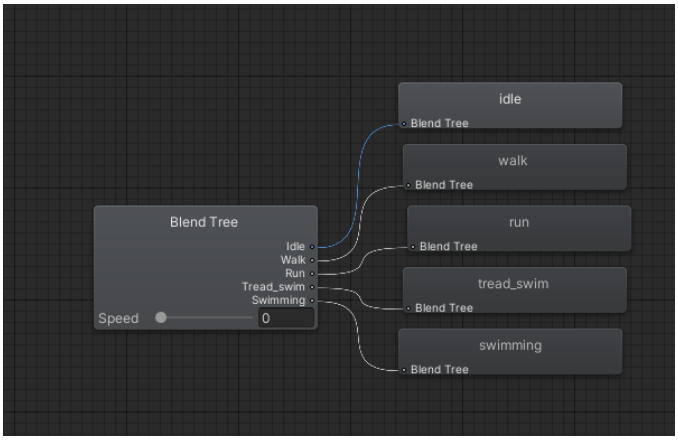


- **animation editors:** Χρησιμοποιήθηκε έτσι ώστε να μπορεί να ανοίγει η πόρτα όταν ο χρήστης κάνει interact με αυτήν αλλά και μέσα στα time-line για την κίνηση του παίχτη.

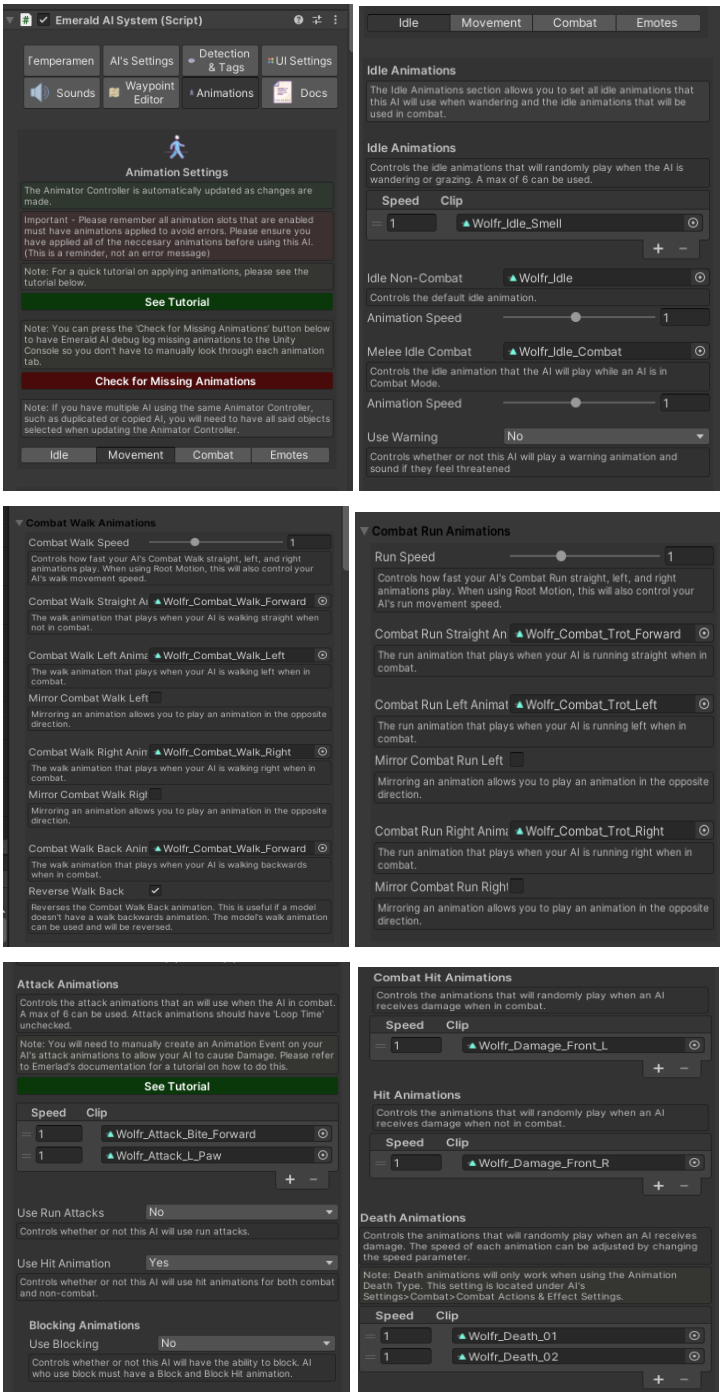


- **animator:** Μέσω των animation που πήραμε από το mixamo.com, και με την βοήθεια του animator καταφέραμε να κάνουμε τον παίχτη μας να κίνειται χρησιμοποιώντας τα κατάλληλα animations. Για την σωστή μετάβαση προτιμήθηκαν Blend Trees.





- **Emerald-AI:** Το συγκεκριμένο asset ασχολείται με την κίνηση των NPCs. Μέσω αυτού, τα NPCs του παιχνιδιού είναι πλήρως λειτουργικά με τα κατάλληλα animation. Επίσης, τα NPCs μας κάνουν επίθεση στον παίχτη μας εάν νιώσουν απειλή με την βοήθεια του asset αυτού.

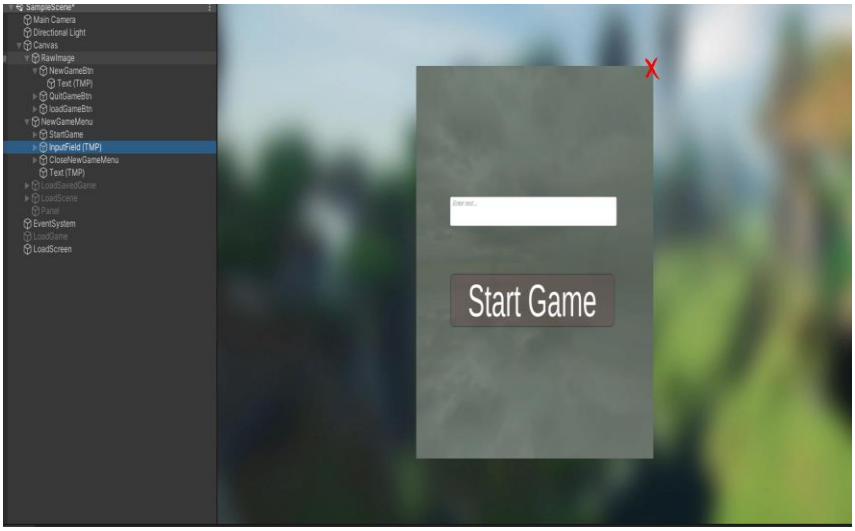


➢ Canvas

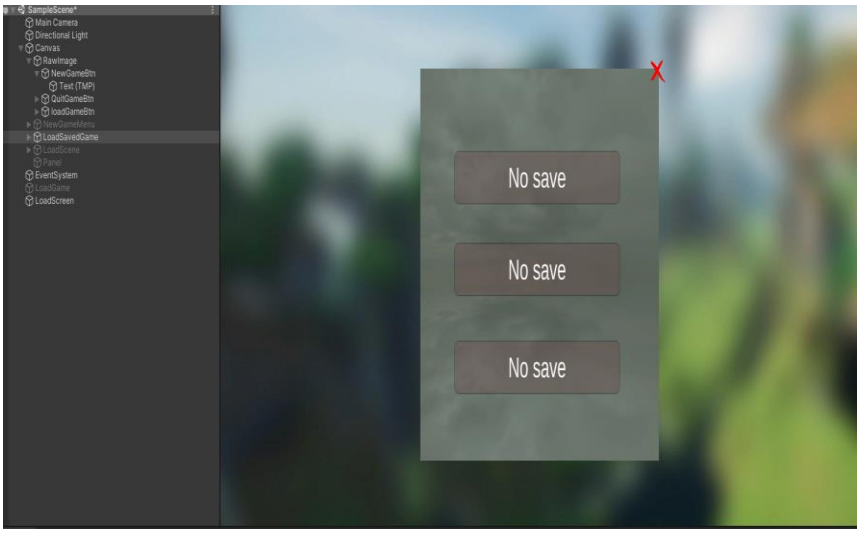
Για την δημιουργία εισαγωγικής οθόνης, εισαγωγικού μενού, in game menu, mini-map και map έγινε η χρήση του UI του unity.

- **Εισαγωγική οθόνη:** Για την εισαγωγική οθόνη προσθέσαμε την εικόνα που θέλαμε και τρια κουμπια (NewGameBtn, QuitGameBtn, LoadGameBtn). Ανάλογα με το κουμπι γίνεται και η ανάλογη ενέργεια.

I. NewGameBtn: Πατόντας το “New game”, εμφανίζεται στον χρήστη μια νέα εικόνα με ένα inputField ώστε ο χρήστης να γράψει το όνομα του save που θέλει και να ξεκινήσει ένα καινούργιο παιχνίδι.

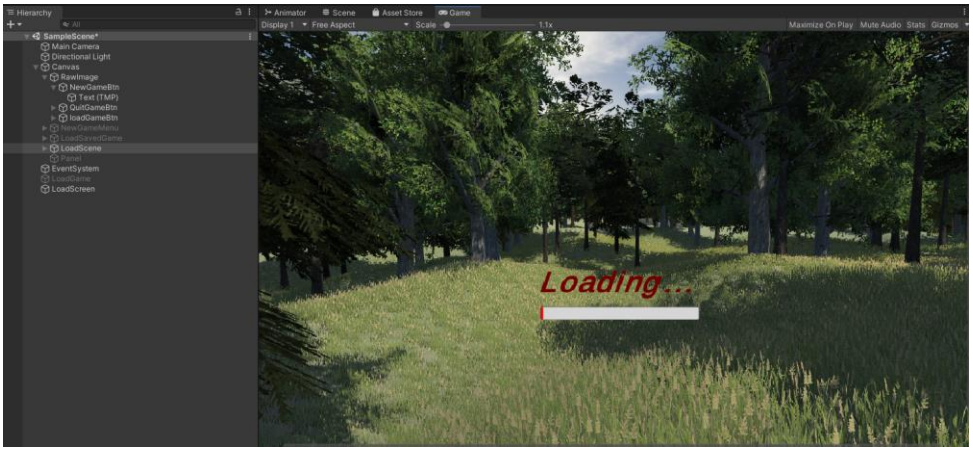


II. **LoadGameBtn:** Πατώντας το δεύτερο κουμπί, δείχνει στον χρήστη τα αποθηκευμένα saves, εάν υπάρχουν, μαζί με την δυνατότητα να κάνει καινούργιο save καθώς και να σβήσει τα ήδη υπάρχοντα.

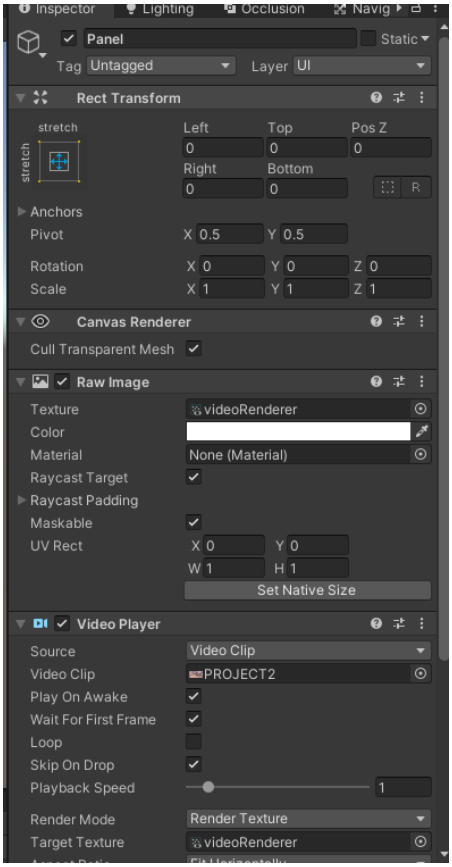


III. **QuitGameBtn:** Πατώντας το κουμπί αυτό κλείνει το παιχνίδι.

- **Οθόνη φόρτωσης:** Η οθόνη αυτή εμφανίζεται ο χρήστης ξεκινήσει το παιχνίδι (είτε new game, ειτε load game) και παραμένει μέχρι να φορτώσει ο χάρτης.

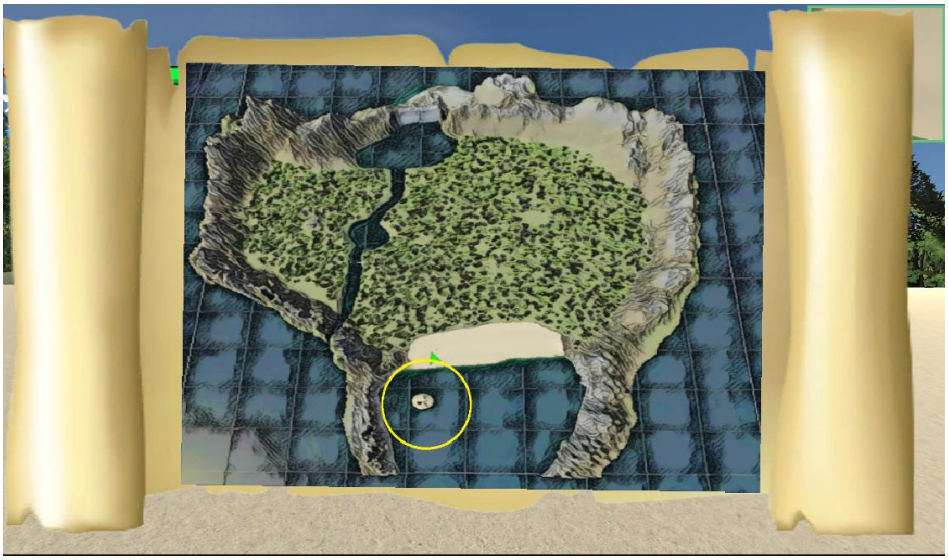


- **Βίντεο:** Εάν ο χρήστης πατήσει new game, τότε εμφανίζεται το panel και παίζει ένα μικρό βίντεο δείχνοντας τον χαρακτήρα να πέφτει με το αεροπλάνο.



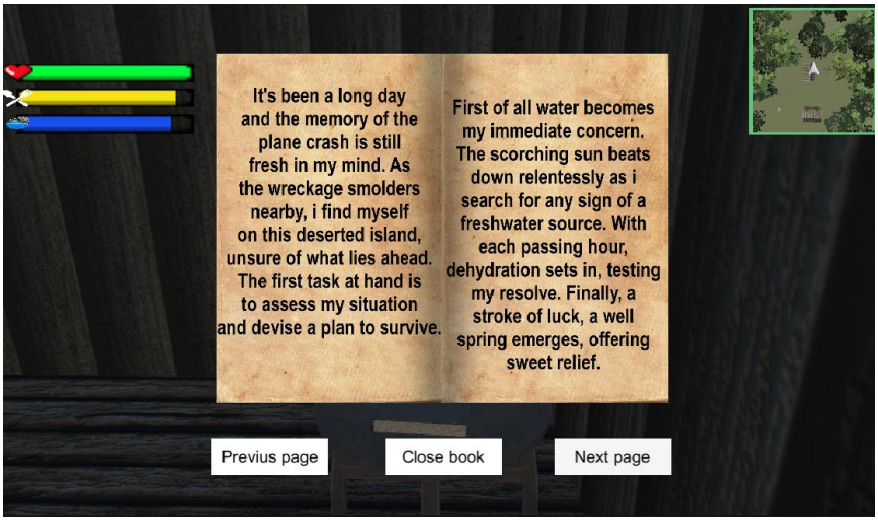
- **Οθόνη παιχνιδιού:** Αφού φορτώσει ο χάρτης, τοτε ο χρήστης μπορεί να δει τρεις μπάρεις με ζωή, νερό, και ενέργεια καθώς και το minimap του παιχνιδιού.

- **Χάρτης:** Στον χάρτη του παιχνιδιού έχουμε πάρει μια πανωραμική φωτογραφία του νησιού και έχουμε προσθέσει ένα εικονίδιο για να δείχνει την τοποθεσία του παίχτη και τις τοποθεσίες που πρέπει να πάει ο παίχτης για να συνεχίσει το παιχνίδι.

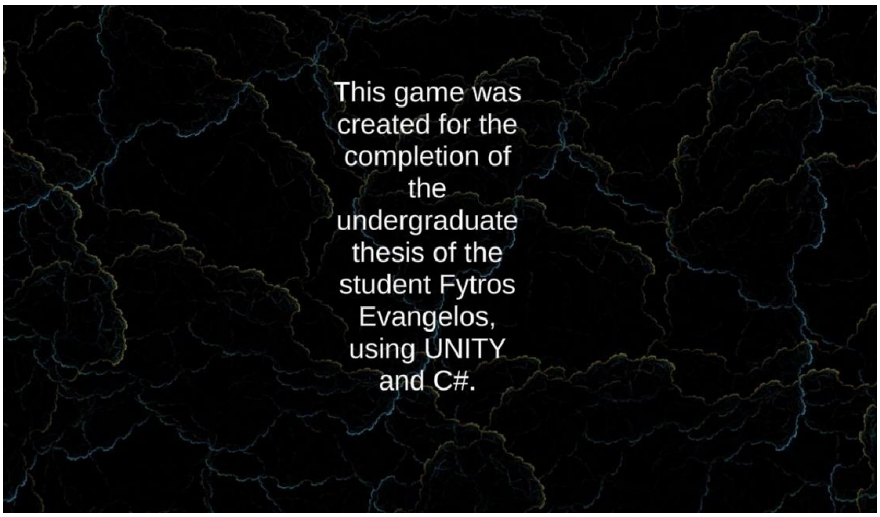


- **Στόχευση χαρακτήρα:** Όταν ο χαρακτήρας μας βρεί το τόξο μπορεί να στοχεύσει και να ρίξει με αυτό. Για αυτόν τον λόγο σχεδιάστηκε ένας στόχος στο κέντρο της όθονης για να δείχνει που θα πάει ο το βέλος.

- **Ανάγνωση βιβλίου:** Όταν ο χρήστης βρεί το βιβλίο για να μάθει τον τρόπο με τον οποίο θα επιβιώσει, ανοίγει ενας canvas με τα περιεχόμενα του βιβλίου τα οποία μπορεί ο χρήστης να διαβάσει.



- **Τέλος παιχνιδιού:** Όταν ο χρήστης βρεί και το τεελυταίο διαμάντι, τοτε παίζει ένα timeline με τον χαρακτήρα να φεύγει από το νησι. Αφου τελείωσει το timeline, εμφανίζεται ένα μήνυμα για το τέλος του παιχνιδιού.



7.Συμπεράσματα και μελλοντικές εργασίες

Κατά την διάρκεια της υλοποίησης της εργασίας καταφέραμε να δημιουργήσουμε ένα 3D First-Person shooter παιχνίδι με έναν ενδιαφέρον και σχεδιασμένο κόσμο. Η ανάπτυξη αυτού του παιχνιδιού μας έδωσε την ευκαιρία να εξοικειωθούμε με τη χρήση πολλών τεχνικών και εργαλείων για την δημιουργία ενός τόσο πολύπλοκου παιχνιδιού.

Αν και αντιμετωπίστηκαν αρκετές προκλήσεις κατά τη διάρκεια της ανάπτυξης , όπως τα προβλήματα με τον σωστό φωτισμό του νησιού και τα χαμηλά fps τα οποία διορθώθηκαν, αλλά και αστοχίες κώδικα, βοήθησε στην ανάπτυξη των ικανοτήτων σε μελλοντική σχεδίαση άλλου παιχνιδιού. Υπήρξαν βοήθειες από διάφορα tutorial, κυρίως για να σχεδιαστεί το UI με τέτοιο τρόπο ώστε να φαίνεται όμορφο στον χρήστη.

Σε μελλοντική εργασία, προτεραιότητα θα είναι η βελτιστοποίηση των γραφικών, η πρόσθεση περισσότερων αντικειμένων για να μπορεί να αλληλοεπιδράει ο χρήστης και τον εμπλουτισμό του σεναρίου.

8. Εκτέλεση

Υπάρχει βίντεο με walkthrough του παιχνιδιού ανεβασμένο στο OneDrive και μπορείτε να το βρείτε εδώ: PtuxiakhPanepisthmioPeiraiws

9. Βιβλιογραφικές Πηγές

Στη διάρκεια της ανάπτυξης του παιχνιδιού, αντλήθηκαν πολύτιμες πληροφορίες και πόροι OnLine. Οι πηγές αυτές περιλάμβαναν εκπαιδευτικά βίντεο στο YouTube που δημιουργήθηκαν από διαφορετικούς παραγωγούς, λύσεις από το Stack Overflow, καθώς και πολύτιμες συζητήσεις και μαθηματικές προσεγγίσεις από διάφορα OnLine forum που είναι αφιερωμένα στην ανάπτυξη παιχνιδιών.

https://www.youtube.com/watch?v=gHFQy_-2W20&t=473s

https://www.youtube.com/watch?v=28JTTXqMvOU

https://www.youtube.com/watch?v=YMj2qPq9CP8

https://www.youtube.com/watch?v=BLfNP4Sc_iA

https://www.youtube.com/watch?v=eoO2SzRi1D8

https://www.youtube.com/watch?v=1MNCG4bSSM4&t=110s

https://www.youtube.com/watch?v=vApG8aYD5aI

https://www.youtube.com/watch?v=tD4tR7zO8y0