A blue and red shield with a white and blue logo

Description automatically generated

**ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ**

**ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ**

**Τελική Εργασία Μαθήματος**

**ΕΙΚΟΝΙΚΗ ΠΡΑΓΜΑΤΙΚΟΤΗΤΑ**

**Τεχνικό Εγχειρίδιο Εφαρμογής**

**Ουρανία Αργύρα P20023**

**Γεώργιος Χριστόπουλος P20206**

**Επιβλέποντες Καθηγητές : Παναγιωτόπουλος Θεμιστοκλής**

**Τηλέφωνο επικοινωνίας : 6981059764**

**ΠΕΙΡΑΙΑΣ**

**ΙΑΝΟΥΑΡΙΟΣ 2024**

**Εισαγωγή**

Το αντικείμενο της εργασίας μας με βάση την εκφώνηση του μαθήματος Εικονική Πραγματικότητα του 7ου εξαμήνου είναι η δημιουργία ενός 3D χώρου με ένα χωριό της δικής μας έμπνευσης. Αποφασίσαμε να αναπαραστήσουμε ένα δασικό χωριό τρισδιάστατης αναπαράστασης, όπου θα υπάρχει η δυνατότητα διαβίωσης στην απομακρυσμένη δασική περιοχή. Μέσα από αυτόν τον χώρο, αναπαρίσταται η ζωή σε ένα χωριό δικής μας φαντασίας, με πλούσια δάση και βλάστηση, λίμνες ανάμεσα από τα βουνά, απόκρυφα μονοπάτια και εκπληκτικά βουνά που απλώνονται σχεδόν σε κάθε άκρη του αρχικού μας terrain. Το δάσος προσπαθεί να μεταφέρει στον παίκτη ένα συναίσθημα γαλήνης και ηρεμίας, εξερευνώντας κάθε πτυχή του terrain, ενώ παράλληλα μπορεί να μετατραπεί και σε παιχνίδι τρόμου, καθώς, σβήνοντας τα φώτα, ο παίκτης θα νιώθει χαμένος σε ένα εικονικό δάσος. Σκοπός της εργασίας είναι η δημιουργία κάποιων σπιτιών με αρχαίο στυλ αρχιτεκτονικής, καθώς ενσωματώνεται πιο ομαλά στις ανάγκες της εργασίας. Στο δάσος, μπορούν να υπάρχουν, επίσης, στοχευμένες εγκαταστάσεις που θα παράγουν έργο με σεβασμό στο περιβάλλον, διατηρώντας την οικολογική ισορροπία του δάσους. Για παράδειγμα, μπορούν να ενταχθούν εργοστάσια που δραστηριοποιούνται στον τομέα της δασοκομίας και της διαχείρισης των δασών, προσφέροντας την απαραίτητη συντήρηση για την υγιή ανάπτυξη του περιβάλλοντος.

Να σημειωθεί ότι για τη δημιουργία του word και για τη δημιουργία του 3D περιβάλλοντος χρησιμοποιήθηκε η ιστοσελίδα <https://sites.google.com/view/unipivr2023/%CE%BC%CE%B1%CE%B8%CE%AE%CE%BC%CE%B1%CF%84%CE%B1/%CE%BC%CE%AC%CE%B8%CE%B7%CE%BC%CE%B1-1> και οι σημειώσεις του μαθήματος.

**Δημιουργία Εικονικού Κόσμου**

Οι εικονικοί κόσμοι, όπως το Metaverse, είναι περιβάλλοντα προσομοίωσης που βασίζονται σε υπολογιστή, όπου οι χρήστες μπορούν να αλληλεπιδρούν με αυτόν χρησιμοποιώντας κάποιο avatar ή έναν χαρακτήρα με δυνατότητα αναπαραγωγής. Η δημιουργία εικονικών κόσμων γίνονται όλο και πιο εξελιγμένοι με καλύτερες και σαφέστερες εικόνες και πιο ζωντανούς avatar. Αυτό μπορεί εύκολα να διαπιστωθεί συγκρίνοντας παλιές εκδόσεις παιχνιδιών με πιο νέες, όπου τα γραφικά και η εμπιερία είναι σαφώς καλύτερα και πιο κοντά στην πραγματικότητα. (Για παράδειγμα, σε αυτό το βίντεο βλέπουμε την εξέλιξη των γραφικών στα βιντεοπαιχνίδια μεταξύ του 1958 και του 2020 <https://www.youtube.com/watch?v=IsPPWWlV-T8&ab_channel=Cussan> ) Για τη δημιουργία ενός εικονικού κόσμου, μπορούμε να εγγραφούμε σε πλατφόρμα εικονικού κόσμου και να δημιουργήσουμε τον δικό μας χώρο, ακόμη και εάν δεν έχουμε εμπειρία κωδικοποίησης.

**Unity Engine**

Στα μαθήματα που πραγματοποιήθηκαν στο πλαίσιο της διδασκαλίας «Εικονική Πραγματικότητα» χρησιμοποιήθηκε η εφαρμογή Unity, η οποία δίνει τη δυνατότητα στους χρήστες να δημιουργούν παιχνίδια και εμπειρίες τόσο σε 2D όσο και σε 3D, προσφέροντας ένα κύριο API με σύνολο εντολών γραμμένων σε C#, βοηθώντας τον επεξεργαστή Unity με τη μορφή πρόσθετων, τα ίδια τα παιχνίδια, ενώ παράλληλα προσφέρει λειτουργικότητα. Η μηχανή ανάπτυξης 3D σε πραγματικό χρόνο της Unity επιτρέπει σε καλλιτέχνες, σχεδιαστές και προγραμματιστές να συνεργαστούν για να δημιουργήσουν εκπληκτικές και καθηλωτικές εμεπιρίες.

**Δημιουργία Project**

To Unity Hub είναι μία αυτόνομη εφαρμογή που βελτιστοποιεί τον τρόπο πλοήγησης, λήψης και διαχείρισης των έργων και εγκαταστάσεων του Unity. Μπορούμε να χρησιμοποιήσουμε υο Unity Hub για διαχείριση, λήψη και εγκατάσταση λειτουργικών μονάδων και εκδόσεων του Unity Editor. Το πρόγραμμα επεξεργασίας Unity Editor είναι το οπτικό στοιχείο που μας δίνει τη δυνατότητα να δημιουργήσουμε παιχνίδια. Επειδή το μεγαλύτερο μέρος τη αλληλεπίδρασης που έχουμε είναι στην πραγματικότητα με τον επεξεργαστή, πολλοι άνθρωποι το αναφέρουν και ως Unity.

Για το Unity Hub ζητείται να δημιουργηθεί προσωπικός λογαριασμός και personal license (άδεια).

Για τη δημιουργία ενός project, πατάμε στο New Project τoυ Unity Hub, και επιλέγουμε το προτζεκτ 3D URP (University Render Pipeline), και έπειτα download template. Τέλος δίνουμε ένα όνομα στο προτζεκτ, επιλέγουμε σε ποιον φάκελο θα αποθηκευτεί και πατάμε Create Project.

**Συνοπτική επεξήγηση της εφαρμογής**

Όταν ανοίξει το πρότζεκτ μας, μπορούμε πάνω αριστερά να παρατηρήσουμε την ιεραρχία της εφαρμογής **(hierarchy)**, όπου βρίσκονται τα αντικείμενα του χώρου σε μορφή λίστας (πιθανόν να έχουν και σχέση γονέα – παιδιού). Οι σχέσεις μεταξύ των αντικειμένων είναι ένα προς ένα (1-1), δηλαδή ό,τι υπάρχει στη λίστα, υπάρχει και στον τρισδιάστατο χώρο. Μπορούμε κάποια από αυτά να τα ορίσουμε και ως αόρατα, οπότε στο χώρο δε θα φαίνονται, αλλά στο hierarchy θα υπάρχουν κανονικά. Με δεξί κλικ πάνω σε ένα αντικείμενο, μπορούμε να δούμε τα properties (ιδιότητες του κάθε αντικειμένου), τα οποία εμφανίζονται στον inspector.

* Πώς μπορούμε να αλλάξουμε ιδιότητες σε ένα αντικείμενο και να αλλάξουν αυτόματα οι ίδιες ιδιότητες σε παρόμοια αντικείμενα;

Με τη διαδικασία **prefabricated** (πρωτότυπο), όπου σε έναν ειδικό τύπο αρχείου κρατάμε το πρωτότυπο αντικείμενο και όλα τα υπόλοιπα που πηγάζουν από αυτό, στην ουσία είναι σαν αντίγραφα. Αν αλλάξει η ιδιότητα στα αντίγραφα, δεν αλλάζει στο πρωτότυπο.

Το default στον inspector μας επιτρέπει να επιστρέψουμε στις αρχικές ιδιότητες του αντικειμένου.

Αντικείμενο **Game Object**: Είναι θεμελιώδη αντικείμενα που αντιπροσωπεύουν χαρακτήρες και σκηνικά. Συνήθως, λειτουργούν ως δοχεία για components, τα οποία υλοποιούν πραγματική λειτουργικότητα. Ο **inspector** διαθέτει **components**. Ένα από αυτά είναι το **transform**. Έχει διανύσματα-συντεταγμένες (οι default τιμές είναι 0) και την κλίμακα 1. Επίσης, το scale έχει lock που αλλάζουν όλα τα x,y και γίνονται ίδια. Με το **add component** μπορούμε να προσθέσουμε λογική στο αντικείμενο (ήδη φτιαγμένος κώδικας-κλάσεις). Κάτω στην εφαρμογή, υπάρχει το **Project**, το οποίο λειτουργεί σαν file explorer. Δίπλα, το παράθυρο **console** εμφανίζει τα errors της εφαρμογής.

A screenshot of a computer program

Description automatically generatedA screenshot of a computer

Description automatically generated

Εργαλεία **querty**: Το πρώτο είναι απλά για να κοιτάμε το χώρο, το δεύτερο είναι το move (εργαλείο μετακίνησης - position), το τρίτο είναι το rotation (περιστροφή αντικειμένου), το τέταρτο είναι για την κλίμακα-scale, δηλαδή καθορίζει το μέγεθος το αντικειμένου, το πέμπτο χρησιμεύει κυρίως σε αντικείμενα δύο διαστάσεων, ενώ το τελευταίο τα διαμορφώνει όλα μαζί.

Κάνοντας δεξί κλικ στην ιεραρχία, βλέπουμε κάποια ήδη φτιαγμένα αντικείμενα, τα οποία ονομάζονται **primitive**. Στα components έχουμε τη φυσική υπόσταση του αντικειμένου (**capsule collider**) και το **mesh renderer – mesh filter**, το οποίο είναι η εμφάνιση του αντικειμένου.

Για να τρέξουμε την εφαρμογή πατάμε πάνω το βελάκι με το σήμα **play**.

**Freeze position**: Με αυτή την επιλογή, μπορούμε να «ακινητοποιούμε» τις συντεταγμένες της θέσης ενός αντικειμένου στο χώρο 3D, δηλαδή παραμένουν σταθερές και δε μπορούν να τροποποιηθούν κατά τη διάρκεια της εκτέλεσης του παιχνιδιού. Αυτό είναι χρήσιμο σε περιπτώσεις που θέλουμε να διατηρήσουμε ένα αντικείμενο σε μία συγκεκριμένη θέση στον κόσμο του παιχνιδιού, χωρίς να μπορεί να μετακινηθεί από τον παίκτη ή από άλλες δυνάμεις.

**Rigidbody**: Συστατικό που χρησιμοποιείται για την προσωμοίωση της φυσικής συμπεριφοράς αντικειμένων στο παιχνίδι. Το αντικείμενο αποκτά τη δυνατότητα να αλληλεπιδρά με την κίνηση και τις φυσικές δυνάμεις στον κόσμο του παιχνιδιού. Κύρις λειτουργίες του μπορεί να είναι η δυναμική κίνηση, η ανίχνευση συγκρούσεων και η προσομοίωση φυσικής.

**Object Parenting** : Τα γονικά αντικείμενα είναι ο τρόπος με τον οποίο μπορούμε να συσχετίσουμε προσωρινά αντικείμενα με άλλα αντικείμενα. Ένα αντικείμενο μπορεί να έχει ένα γονικό αντικείμενο, δηλαδή του έχει εκχωρηθεί ένας γονέας και μπορεί να μοιραστεί με αυτόν κώδικα, ενέργειες και συμβάντα (σύστημα ιεραρχίας). Αυτή η κοινή χρήση ονομάζεται «κληρονομι;a» και ένα αντικείμενο που έχει γονέα ονομάζεται αντικείμενο «παιδί».

**Nesting objects** : Το ένα αντικείμενο βρίσκεται μέσα στο άλλο.

**Πηγές των assets**

**Unity Asset Store**

Τα αντικείμενα που χρησιμοποιήθηκαν στην εργασία υπάρχουν στην ιστοσελίδα Unity Asset Store <https://assetstore.unity.com/> . Στην ιστοσελίδα, μπορούμε να βρούμε αρκετά δωρεάν assets και να τα προσθέσουμε στο λογαριασμό μας στο Unity. Κάποια από αυτά ίσως να περιέχουν και ExampleScenes, δηλαδή σκηνές που έχει φτιάξει ο δημιουργός με τα asset.

Ένας render pipeline εκτελεί μία σειρά λειτουργιών που λαμβάνουν το περιεχόμενο μιας σκηνής. Το render pipeline που έχει χρησιμοποιηθηκε για την εργασία είναι το URP (Universal Render Pipeline), το οποίο χρησιμοποιείται για τη βελτιστοποίηση της απόδοσης των γραφικών σε όλες τις πλατφόρμες. Κάποια από τα αντικείμενα που προσθέτουμε μπορεί να μη χρησιμοποιούν URP (εμφανίζονται με ροζ χρώμα στη σκηνή). Για να δούμε αν μπορούμε να το μετατρέψουμε σε URP πατάμε πάνω στο αντικείμενο και έπειτα στο Window -> Rendering -> Render Pipeline Converter, κάνουμε material update και Built-i to URP και μετά Convert.

A screenshot of a computer

Description automatically generated

Επίσης, στην εφαρμογή μας από το Unity Asset Store προσθέτουμε έναν προφτιαγμένο χαρακτήρα ThirdPersonController, με τον οποίο θα μπορούμε να δούμε τη σκηνή, μέσω της κάμερας που θα τον ακολουθεί.

**Mixamo**

Το mixamo δίνει τη δυνατότητα στους χρήστες να ζωντανέψουν τρισδιάστατους χαρακτήρες για παιχνίδια, ταινίες και άλλα (<https://www.mixamo.com/#/> ). Ενισχύει τη δημιουργηκότητα με κινούμενους τρισδιάστατους χαρακτήρες έτοιμους για χρήση.

A screenshot of a computer

Description automatically generated

Μέσω του mixamo μπορούμε να επιλέξουμε χαρακτήρες και κινήσεις που θέλουμε να κάνουν, καθώς επίσης και επιλογές για το πόσο θέλουμε να έχουν ανοιχτά τα χέρια τους (character arm space), αν θέλουμε την κίνηση ολόκληρη ή από κάποιο χρονικό διάστημα (trim) και μετά και πόσο γρήγροα θέλουμε να κινείται ο χαρακτήρας (overdrive). Κατεβάζουμε τους χαρακτήρες σε fbx for unity, with skin, 30 frames per second and none keyframe reduction. Κατεβαίνει αρχείο με κατάληξη .fbx.

A screenshot of a computer

Description automatically generated

Οι χαρακτήρες λειτουργούν με controllers. Ανάλογα με τον controller που θα βάλουμε σε κάθε φιγούρα θα κάνει και την αντίστοιχη κίνηση. Αν θέλουμε μία φιγούρα να κάνει διαφορετικές κινήσεις, χρησιμοποιούμε διαφορετικούς controllers.

**Περιγραφή του προβλήματος**

Για την περιγραφή του προβλήματος, αρχικά θα πρέπει να περιγράψουμε την έννοια της εικονικής πραγματικότητας. Η εικονική πραγματικότητα (Virtual Reality) είναι ένα περιβάλλον που δημιουργείται από τον υπολογιστή με σκηνές και αντικείμενα που φαίνονται αληθινά, παροτρύνοντας το χρήστη να ενταχθεί σε αυτό το περιβάλλον.

Η περιγραφή του προβλήματος είναι συχνά το πρώτο βήμα για την έναρξη ενός έργου. Στη δική μας περίπτωση, η περιγραφή μπορεί να μας βοηθήσει να κατανοήσουμε καλύτερα τη σημασία του προβλήματος.

Στο πλαίδιο της εργασίας, έχουμε το πρόβλημα της δημιουργίας ενός σύγχρονου χωριού μέσω της εφαρμογής Unity που θα προσφέρει στο χρήστη μοναδική, ρεαλιστική και απολαυστική εμπειρία εξερεύνησης και αλληλεπίδρασης. Αυτό απαιτεί αρκετές διαδικασίες, όπως η δημιουργία μοντέλων και αντικειμένων που θα συνθέτουν το χωριό, η εφαρμογή του φωτισμού, των υλικών και των κινήσεων που θα προσδώσουν στο παιχνίδι ρεαλισμό, αληθοφάνεια και λειτουργικότητα. Υπάρχει, δηλαδή, ανάγκη για ρεαλιστική απεικόνιση των αντικειμένων και του περιβάλλοντος, για τη διαχείριση του φωτισμού (για παράδειγμα μπορούμε να ορίσουμε τη διάρκεια της ημέρας 20 λεπτά και να διαχειριστούμε ανάλογα το φωτισμό μες σκοπό να δημιουργηθεί μία πιο φυσική αίσθηση) καθώς και για την αποδοτική αντιμετώπιση της αλληλεπίδρασης του χρήστη με το περιβάλλον (στο οποίο πιθανόν να απαιτείται κώδικας). Στο σχεδιασμό, θα πρέπει να υπάρχει ακρίβεια, δημιουργικότητα και λεπτομέρεια, όπως επίσης και καλή αισθητική και δομή.

Για το σχεδιασμό της εφαρμογής και εφόσον θα υλοποιηθεί σύγχρονο χωριό, θα πρέπει να χρησιμοποιηθούν προϋπάρχοντα ή να δημιουργηθούν μοντέλα και αντικείμενα, όπως ξύλινα σπιτάκια, λουλούδια, δένδρα, λιμνούλες, ζωάκια κτλ. Αυτά τα αντικείμενα, θα πρέπει να τοποθετηθούν στο χώρο με σωστή δομή και προστεθούν τα απαραίτητα υλικά και εφέ, καθώς και ο απαραίτητος φωτισμός. Θα πρέπει να ληφθεί υπόψη η αλληλεπίδραση του χρήστη με το χωριό, όπως για παράδειγμα η δυνατότητα περιηγήσεώς του σε αυτό, η δυνατότητα αλληλεπίδρασής του με αντικείμενα (όπως για παράδειγμα το άνοιγμα της πόρτας στα σπιτάκια) και η αλληλεπίδρασή του με άλλους χαρακτήρες (όπως για παράδειγμα χαιρετισμός όταν φτάνουμε σε συγκεκριμένη απόσταση από αυτόν). Είναι λογικό ότι η περιγραφή του προβλήματος δεν αναφέρεται μόνο στην υλοποίηση ενός περιβάλλοντος, αλλά και στις προκλήσεις που προκύπτουν κατά τη διαδικασία δημιουργίας του.

Παραπάνω αναφέρθηκαν και αναλύθηκαν κάποια από τα εργαλεία που χρησιμοποιούνται στην εφαρμογή Unity για τη δημιουργία ενός εικονικού περιβάλλοντος, καθώς και κάποιες τεχνικές οι οποίες εφαρμόζονται για την επίλυση συγκεκριμένων προβλημάτων. Για την δημιουργία του χώρου μας, θα πρέπει να αναητάμε συνεχώς πιθανές τεχνικές λύσεις για τις προκλήσεις που προκύπτουν (erros, render pipline converter κτλ.)

**Αναλυτική παρουσίαση όλων των φάσεων ανάπτυξης (Ανάλυση-Σχεδίαση-Υλοποίηση)**

Στο συγκεκριμένο βήμα της εργασίας, ζητείται να κάνουμε μία αναλυτική παρουσίαση όλων των φάσεων ανάπτυξης του κύκλου ζωής της εφαρμογής. Αυτό περιλαμβάνει την **ανάλυση** των απαιτήσεων του έργου, το **σχεδιασμό** της αρχιτεκτονικής της εφαρμογής, των δοκιμών στοιχείων, των διεπαφών και της λειτουργικότητας της εφαρμογής και τέλος την **υλοποίηση**, δηλαδή την πραγματική κατασκευή της εφαρμογής.

**Ανάλυση**

Οι απαιτήσεις ενός έργου αναφέρονται στις λειτουργικές και μη λειτουργικές ανάγκες και προδιαγραφές που πρέπει να πληροί η τελική εφαρμογή. Αυτές οι απαιτήσεις καθορίζουν το πεδίο του έργου και ορίζουν τις ανάγκες και τις προσδοκίες για την τελική εφαρμογή. Κατά τη διαδικασία ανάπτυξης ενός έργου, η κατανόηση και η καταγραφή αυτών των απαιτήσεων είναι κρίσιμη για την επιτυχή ολοκλήρωση του έργου.

Οι απαιτήσεις αυτές μπορούν να χωριστούν σε δύο βασικές κατηγορίες:

* Λειτουργικές απαιτήσεις : Σε αυτή την κατηγορία συμπεριλαμβάνονται οι λεπτομερείς περιγραφές των λειτουργιών και των δυνατοτήτων της εφαρμογής. Στο παιχνίδι που υλοποιούμε οι λειτουργικές απαιτήσεις περιλαμβάνουν τον τρόπο κίνησης των χαρακτήρων, τον τρόπο επικοινωνίας μεταξύ τους, τα επίπεδα και τους στόχους του παιχνιδιού, δυνατότητα αλληλεπίδρασης του χρήστη με το περιβάλλον, τη προβολή ρεαλιστικών γραφικών για τη δημιουργία ενός αληθοφανούς κόσμου κτλ.
* Μη λειτουργικές απαιτήσεις : Σε αυτή την κατηγορία αναφέρονται οι περιορισμοί ή οι προδιαγραφές που πρέπει να πληροί η εφαρμογή, αλλά δεν αφορούν απαραίτητα τις λειτουργίες της. Περιλαμβάνονται, δηλαδή, θέματα ασφαλείας, απόδοσης, χρηστικότητας, συμβατότητας με συγκεκριμένες πλατφόρμες, επεκτασιμότητας κτλ.

Για τις λειτουργικές απαιτήσεις, μπορούμε να δημιουργήσουμε διάφορους **shaders**, δηλαδή κώδικες που πάνω σε αυτούς μπορούμε να φτιάξουμε τα materials. Μπορεί να ορίζει το πώς θα απεικονίζεται ή θα επεξεργάζεται το material, καθώς επιτρέπει τη δημιουργία ποικίλων εφέ, όπως σκιές, φωτισμό, ανακλάσεις κτλ προσθέτοντας ρεαλισμό. Επίσης, μπορεί να περιέχει λειτουργίες και εντολές που καθορίζουν το πώς θα αλλάξει η εικόνα ή θα γίνει επεξεργασία της γραφικής πληροφορίας κάθε pixel στην οθόνη. Στη δική μας περίπτωση, μπορεί να χρησιμέψει για να κινηθούν τα αντικείμενά μας ή οι χαρακτήρες μας.

Στις απαιτήσεις έχουμε και το έδαφος, το οποίο είναι πολύ σημαντικό για την εφαρμογή, καθώς προσδίδει αληθοφάνεια και πιο ρεαλιστική εμπειρία στο χρήστη.

**Σχεδιασμός**

Ο σχεδιασμός αναφέρεται στη διαδικασία που προηγείται της υλοποίησης του έργου, κατά την οποία καθορίζονται οι λεπτομερείς προδιαγραφές και οι προσεγγίσεις για την επίλυση του προβλήματος. Ο σχεδιασμός περιλαμβάνει τη δημιουργία μοντέλων, αλγορίθμων και προγραμμάτων που θα χρησιμοποιηθούν στην υλοποίηση του έργου, καθώς και το σχεδιασμό του παιχνιδιού. Ο σχεδιασμός αποτελεί κρίσιμο στάδιο στη διαδικασία ανάπτυξης ενός έργου, καθώς καθορίζει τη βάση για την επόμενη φάση της υλοποίησης.

Οι βασικές φάσεις του σχεδιασμού είναι:

* Ανάλυση των απαιτήσεων : Σε αυτή τη φάση του σχεδιασμού, γίνεται μία λεπτομερής ανάλυση των απαιτήσεων του έργου, προκειμένου να κατανοηθεί πλήρως το πρόβλημα που πρέπει να επιλυθεί και οι λειτουργικότητες που πρέπει να παρέχονται. Στην ανάλυση των απαιτήσεων συμπεριλαμβάνονται και οι λειτουργικές και μη λειτουργικές απαιτήσεις που αναφέρθηκαν στην ανάλυση, η απαιτήσεις απόδοσης του παιχνιδιού, οι οποίες περιλαμβάνουν την ταχύτητα φόρτωσής του και την απόδοση των γραφικών του και τέλος οι απαιτήσεις ασφαλείας, οι οποίες αφορούν την προστασία των δεδομένων και την αποτροπή των απειλών.
* Σχεδιασμός αρχιτεκτονικής : Σε αυτή τη φάση, καθορίζεται η γενική δομή του συστήματος και ο τρόπος λειτουργίας των διάφορων συνιστωσών του.
* Σχεδιασμός διεπαφών χρήστη : Εάν το έργο περιλαμβάνει διεπαφή χρήστη, καθορίζονται οι λειτουργίες και ο σχεδιασμός των διάφορων ενεργει΄ψν που απαιτούνται.
* Σχεδίαση αλγορίθμων και δομών δεδομένων : Καθορίζονται οι αλγόριθμοι που θα χρησιμοποιηθούν για την επίλυση των προβλημάτων και οι δομές δεδομένων που θα χρησιμοποιηθούν για την οργάνωση και την αποθήκευση των δεδομένων.
* Σχεδιασμός ασφάλειας και απόδοσης : Λαμβάνονται υπόψη θέματα ασφαλείας και απόδοσης του συστήματος, προκειμένου να εξασφαλιστεί η ασφαλής και αποτελεσματική λειτουργία του.

Αρχικά, για το σχεδιασμό του παιχνιδιού, θα πρέπει να δημιουργήσουμε ένα terrain (έδαφος), το οποίο είναι ένα από τα βασικότερα βήματα της εφαρμογής (3D object -> Terrain). Για την εμφάνιση και το σχεδιασμό του παιχνιδιού, θα πρέπει να λάβουμε υπόψη τη γεωμορφία του χώρου. Στο παιχνίδι, αφού αποφασίστηκε να κάνουμε ένα χωριό, προστέθηκε στο έδαφος αρκετή βλάστηση και διάφορα κτίρια και σπιτάκια, καθώς και διάφορες λιμνούλες, πεδιάδες και κοιλάδες. Το έδαφος συμβάλλει αρκετά στην εμπειρία του παίκτη καθώς δημιουργεί το μεγαλύτερο μέρος της ατμόσφαιρας και του περιβάλλοντος του παιχνιδιού και στη δική μας περίπτωση, δημιουργείται μία φιλική και φιλόξενη ατμόσφαιρα. Επίσης, δίνει κίνητρο στο χρήστη για ποικιλία και εξερεύνης του περιβάλλοντος, καθώς στο παιχνίδι υπάρχει ποικιλία στο έδαφος, η οποία προσθέτει ενδιαφέρον και περιπέτεια στο χρήστη.

Στο έδαφος, πρέπει να προστεθούν όρια για να μη μπορεί ο παίκτης να βγει εκτός ορίων και να πέσει στο κενό. Επίσης, στο οπτικό πεδίο του χρήστη πρέπει να προστεθεί κάτι πιο αληθοφανές, όπως για παράδειγμα ομίχλη ή μία εικόνα.

Στο έδαφος, θα προσθέσουμε δένδρα με το paint trees στον inspector. Μπορούμε να κάνουμε add ενός prefab δένδρου και να ζωγραφίσουμε δένδρα μαζικά εκεί που θέλουμε. Στα δένδρα εμπεριέχεται και το level of detail, δηλαδή το πόση λεπτομέρεια θέλουμε να έχει το πλέγμα του αντικειμένου. Εάν θέλουμε δένδρα σε όλο το έδαφος, τότε πατάμε το number of trees και επιλέγουμε πόσα δένδρα θέλουμε να εμφανιστούν μαζικά σε όλο το έδαφος. Το ίδιο μπορούμε να κάνουμε και με το γρασίδι, τα μανιτάρια, τους βράχους κτλ. Να σημειωθεί ότι εάν το δένδρο δεν είναι αντικείμενο και το ζωγραφίζουμε με αυτόν τον τρόπο, τότε είναι δύσκολο να το επεξεργαστούμε. Μπορούμε να τα βάζουμε 1 προς 1 ή να κάνουμε duplicate, το οποίο προφανώς και είναι πιο χρονοβόρο.

A screenshot of a computer

Description automatically generated

Στο σχεδιασμό, θα πρέπει να ληφθεί υπόψη ότι το έδαφος θα πρέπει να γεμίσει με χαρακτήρες και αντικείμενα, τα οποία αναλύονται και παρακάτω στην αναφορά. Τα αντικείμενα και οι χαρακτήρες θα χρησιμοποιηθούν από το unity asset store και το mixamo, τα οποία έχουν αναλυθεί στις πηγές των assets.

**Υλοποίηση**

Η φάση της υλοποίηση κρίνεται απαραίτητη στη διαδικασία ανάπτυξης ενός έργου. Κατά τη διάρκεια αυτής της φάσης μεταφέρονται οι απαιτήσεις και ο σχεδιασμός σε λειτουργική και εκτελέσιμη μορφή. Κατά την υλοποίηση, λαμβάνονται υπόψη οι ακόλουθες αρχές και πρακτικές, οι οποίες συνεισφέρουν στην αποτελεσματική και αξιόπιστη υλοποίηση του παιχνιδιού μας, ενισχύοντας την απόδοση, την ασφάλεια και τη σταθερότητά του:

* Καλή πρακτική στον κώδικα : Ο κώδικας πρέπει να είναι κατανοητός, οργανωμένος και ευανάγνωστος για τους άλλους προγραμματιστές. Πρέπει να εμπεριέχει σωστά σχόλια, κατάλληλη ονοματοδοσία μεταβλητών και συναρτήσεων και βέλτιστες πρακτικές προγραμματισμού.
* Δοκιμές κώδικα : Πρέπει να γίνονται συχνές δοκιμές στον κώδικα με σκοπό να επιβεβαιωθεί η λειτουργικότητα και η ακρίβεια του κώδικα.
* Συνεχής ενημέρωση : Ενημέρωση για προβλήματα, σφάλματα, για την αντιμετώπισή τους, για ελλείψεις κτλ.
* Υλοποίηση του περιβάλλοντος με διάφορα αντικείμενα και χαρακτήρες
* Αλληλεπίδραση με χαρακτήρων με αντικείμενα ή χαρακτήρων με άλλους χαρακτήρες

Το ύψος του εδάφους μπορεί να ρυθμιστεί μέσω του inspector (πλέγμα από ακμές που δε

μπορεί να λάβει αρνητικές τιμές) με το raise or lower terrain. Με τα brushes μας δίνεται η δυνατότητα να σηκώσουμε βουνά. Μπορούμε να ορίσουμε στο transform x=-500, y=-200, z=-500 για να ανεβάσουμε το επίπεδο και να μπορούν οι ακμές να κατέβουν πιο κάτω. Μπορούμε, επίσης, να αλλάξουμε το χρώμα του με το paint texture, με το οποίο μπορούμε να δημιουργήσουμε και διαφόρων ειδών μονοπάτια στο χωριό. Για παραπάνω πολυπλοκότητα, πατάμε στο Windows -> Package Manager και αναζητάμε terrain tools (Πχ terrain sample assett pack), λειτουργία που μπορεί να κάνει τα εργαλεία που θα χρησιμοποιήσουμε πιο εξελιγμένα (advanced).

Για να μη μπορεί ο παίκτης να βγει εκτός ορίων, δημιουργούμε έναν αόρατο τοίχο με όνομα InvisibleWall στα αντικείμενα, ο οποίος αποτελείται από 4 κύβους διαμορφωμένους, έτσι ώστε να καλύπτουν τις 4 πλευρές του εδάφους, και να αποτρέπουν το χρήστη να πέσει στο κενό και να δημιουργηθεί λογικό σφάλμα.

A screenshot of a computer program

Description automatically generated

**Πίνακα με C# scripts και σύντομη περιγραφή**

Για τη συγγραφή κώδικα, πατάμε αριστερό κλικ στο φάκελο που θέλουμε να αποθηκευτεί και έπειτα δεξί κλικ -> Create -> C# script. Με το enter γίνεται compile και ανοίγει η εφαρμογή Visual Studio. Στον κώδικα, χρησιμοποιούμε κάποια libraries (κλάσεις που φτιάχνονται αυτόματα, μπορούμε και εμείς να κάνουμε αναφορά σε κάποια βιβλιοθήκη) και όταν ανοίγουμε το script έχουμε τις μεθόδους Start() και Update(). H Start() καλείται μία φορά στο πρώτο καρέ που το GameObject είναι ενεργό και πριν την πρώτη κλήση της Update. Η μέθοδος Update() καλείται μία φορά κάθε καρέ εάν το GameObject και το script είναι ενεργά. Τα scripts προσθέτονται σε ένα χαρακτήρα ή αντικείμενο με το add component.

Στα scripts μπορούμε να υλοποιήσουμε τη λειτουργικότητα των χαρακτήρων, τις κινήσεις τους, τη συμπεριφορά τους και τις δράσεις τους εντός του παιχνιδιου, με σκοπό να γίνει πιο ρεαλιστικό και ενδιαφέρον. Επίσης, υλοποιούμε τη λειτουργικότητα του παιχνιδιού γενικότερα, ορίζοντας τους κανόνες και τους μηχανισμούς του (ελέγχους, επικοινωνία μεταξύ χαρακτήρων κτλ.). Σημαντικά είναι τα scripts για την αλληλεπίδραση του παίκτη με το περιβάλλον γενικότερα που έχουμε δημιουργήσει. Τέλος, μπορούμε να διαχειριστούμε τους ήχους και τα εφέ, καθώς και τη διαχείριση των δεδομένων του παιχνιδιού.

Αναφέρθηκε και παραπάνω, ότι για να προστεθεί ένα πρόγραμμα σε ένα αντικείμενο, πρέπει να κάνουμε add component το script. Στα scripts χρησιμοποιήθηκε το OnCollisionEnter και το OnTriggerEnter. Το OnTriggerEnter καλείται 50 φορές το δευτερόλεπτο για κάθε αντικείμενο που βρίσκεται στο χώρο (κάθε fixedUpdate), δηλαδή μπορεί σε μία επανάληψη να κληθεί 3 φορές αν εκείνη τη στιγμή εισαχθούν δύο αντικείμενα στην περιοχή, αλλά μπορεί να κληθεί και 0. Στα Project Settings υπάρχουνε μεταβλητές πχ. Time με fixed timestep, όπου οι κινήσεις σχετίζονται με φυσική και γίνονται κάθε 50 φορές το δευτερόλεπτο. Μπορούμε, επίσης, να ορίσουμε time scale, δηλαδή το πόσο γρήγορα θα τρέχει η εφαρμογή μας (0.3 -> 30% ταχύτητα ή 10 για μέγιστη ή 0 για παύση του παιχνιδιού).

Στο πλαίσιο των παιχνιδιών και της φυσικής μηχανικ’ης, η ιδιότητα «kinematic» αφορά την κίνηση ενός αντικειμένου. Όταν ένα αντικείμενο ορίζεται ως kinematic, σημαίνει ΄οτι η κίνησή του δεν επηρεάζεται από δυνάμεις που δρουν σε άλλα αντικείμενα όπως η βαρύτητα, οι συγκρούσεις κτλ. Επίσης, ένα αντικείμενο kinematic ελέγχεται και κινείται από τον προγραμματιστή άμεσα, χωρίς να επηρεάζεται από τις φυσικές δυνάμεις του περιβάλλονοτος. Αυτό χρησιμοποιείται για να ελέγχεται η κίνηση του χαρακτήρα ή άλλων αντικειμένων από τον κώδικα του παιχνιδιού.

|  |
| --- |
| Στους χαρακτήρες mixamo χρησιμοποιήθηκε το Set IsTrigger to true. Το other είναι το αντικείμενο collider και με το gameObject παίρνω όλο το αντικείμενο. Όταν ένα άλλο αντικείμενο μπαίνει μέσα στον collider, η μέθοδος OnTriggerEnter ενεργοποιείται. Με το SetActive(false) απενεργοποιείται το gameObject που εισήλθε μέσα στον collider. Για να γίνει η σύγκρουση πρέπει και τα δύο αντικείμενα να έχουν collider. Επίσης, πρέπει τουλάχιστον ένα απ’ τα δύο αντικείμενα να έχει component rigidBody (component για βαρύτητα, kinematic όταν δε θέλω σύγκρουση με αντικείμενο).  Script onTrigger  private void OnTriggerEnter (Collider other)  {  other.gameObject.SetActive(false);  } |
| Με τον συγκεκριμένο κώδικα ελέγχουμε πότε ο παίκτης και μόνο (κάνουμε φιλτράρισμα) μπαίνει στην περιοχή που έχουμε ορίσει και άμα μπει γίνεται reference στον animator με σκοπο να αλλάξει η μετανλητή που λέγεται wave και σε αυτή την περίπτωση είναι trigger. Στα μιξαμο, υπάρχει συνήθως ήδη ένας collider trigger sphere (ο trigger καλεί οποιοδήποτε αντικείμενο βρίσκεται στην περιοχή). Ο Animator είναι public και σειριοποιήσιμος, άρα εκτεθειμένη μεταβλητή null, στην οποία κάνουμε drag and drop με αριστερό κλικ τον animator που θέλουμε να έχει ο μίξαμο χαρακτήρας.  Script npc (non playable character)  public class npc : MonoBehaviour  {  public Animator npcAnimator;  // Start is called before the first frame update  void Start()  {    }  // Update is called once per frame  void Update()  {    }  private void OnTriggerEnter(Collider other)  {  if (other.tag == "Player") //the player  {  Wave();  }  }  void Wave()  {  npcAnimator.SetTrigger("Wave");  }  } |
| Script messageDisplay  public GameObject messageDisplay; //can be text or 3d object  private void OnTriggerEnter(Collider other)  {  //filter only player  if (other.tag == "Player")  {  messageDisplay.gameObject.SetActive(true);  }  }  } |
|  |
|  |
|  |
|  |
|  |