.

ΑΛΕΞΑΝΔΡΕΙΟ Τ.Ε.Ι. ΘΕΣΣΑΛΟΝΙΚΗΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

<< Δημιουργία 3D παιχνιδιού σε πρώτο πρόσωπο για PC σε Unreal 4 με χρήση Blueprints>>

Του φοιτητή Τσιτηρίδη Ιωάννη

Επιβλέπων καθηγητής: Ευκλείδης Κεραμόπουλος

Αρ. Μητρώου: 123882

Θεσσαλονίκη 2018

ΠΡΟΛΟΓΟΣ

Βασικός στόχος της πτυχιακής εργασίας είναι η δημιουργία ενός ατμοσφαιρικού παιχνιδιού 3 διαστάσεων (3D) που διαδραματίζεται σε πρώτο πρόσωπο, το οποίο θα αναπτυχθεί στην πλατφόρμα ανάπτυξης παιχνιδιών Unreal Engine 4 κυρίως με την χρήση Blueprints. Το παιχνίδι αποτελείται από τον βασικό μας παίχτη τον οποίο χειριζόμαστε και βλέπουμε μέσα από τα μάτια του (πρώτο πρόσωπο) τον κόσμο του παιχνιδιού. Η υλοποίηση έχει γίνει με την χρήση σεναρίου και εξελίσσεται σειριακά (sequence). Έχει δοθεί μεγάλη βάση στο δράμα και την αγωνία και αυτό έχει επιτευχθεί με το συνολικό περιεχόμενο του παιχνιδιού όπως οι διάλογοι, οι ήχοι από το περιβάλλον η διάδρασή με τα αντικείμενα. Μέσα από το ταξίδι της υλοποίησης του παιχνιδιού θα ανακαλύψουμε όσες περισσότερες από τις πτυχές της συγκεκριμένης πλατφόρμας μπορούμε, όπως εργαλεία, πλεονεκτήματα κλπ. Θα προσπαθήσουμε να δώσουμε απάντηση και σε ένα πολύ σημαντικό ερώτημα, εάν τελικά η Unreal Engine είναι το κατάλληλο εργαλείο, το οποίο συγκαταλέγεται στις δυνατότερες πλατφόρμες του είδους της, για την ανάπτυξη παιχνιδιών. Το μέγεθος και οι δυνατότητες είναι σχεδόν απεριόριστες με μόνο γνώμονα την δημιουργικότητα και την φαντασία.

Προσωπικός στόχος είναι η απόκτηση γνώσης, εμπειρίας πάνω στην ανάπτυξη παιχνιδιών και η μελλοντική ενασχόληση σε επαγγελματικό επίπεδο. Επίσης αποτελούσε και προσωπικό στοίχημα η δημιουργία ενός τέτοιου παιχνιδιού δίνοντας μου την ώθηση και την επιβεβαίωση που χρειαζόμουν για αυτά που έπονται.

ΠΕΡΙΛΗΨΗ

Σε αυτήν την πτυχιακή εργασία θα αναφερθούμε στις μηχανές παιχνιδιών και ειδικότερα στην ιστορία της Unreal Engine και την πορεία της από αρχή της δημιουργίας μέχρι το σήμερα. Επίσης θα αναφερθούμε στις δυνατότητες και στα εργαλεία που παρέχονται, δίνοντας βάση κυρίως σε αυτά που χρησιμοποιήθηκαν με πρωταγωνιστή την χρήση Blueprints για τα οποία γίνεται εκτεταμένη αναφορά. Μικρή αναφορά θα γίνει και στα βοηθητικά προγράμματα (Blender, World Machine) που χρησιμοποιήθηκαν για την δημιουργία 3D μοντέλων, texture, Terrain κλπ. Υπάρχει ολόκληρο υποκεφάλαιο στο οποίο αναλύουμε την ορολογία της Unreal Engine 4.

Έπειτα συνεχίζουμε με την Ανάπτυξη του παιχνιδιού, από την δημιουργία του project, του terrain και της φύσης με το εργαλείο foliage, την επεξεργασία του χαρακτήρα πρώτου προσώπου, δημιουργία των μενού έως και τους στόχους για τους οποίους γίνεται εκτεταμένη αναφορά, τα κεντρικά στοιχεία όπως το σπίτι αλλά και διάφορα ακόμη όπως components, cinematics και player dialogues.

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΡΟΛΟΓΟΣ 2
ΠΕΡΙΛΗΨΗ 3
ΠΕΡΙΕΧΟΜΕΝΑ 4
ΕΥΡΕΤΗΡΙΟ ΕΙΚΟΝΩΝ 7
ΚΕΦΑΛΑΙΟ 1: ΕΙΣΑΓΩΓΗ 9
ΚΕΦΑΛΑΙΟ 2: GAME ENGINE & UNREAL ENGINE 10
2.1 Τι είναι μια μηχανή παιχνιδιού; 10
2.2 Τι είναι η Unreal Engine 14
2.3 Η ιστορία της Unreal Engine 15
2.3.1 Unreal Engine 1 15
2.3.2 Unreal Engine 2 16
2.3.3 Unreal Engine 3 17
2.3.3 Unreal Engine 4 19
ΕΠΙΛΟΓΟΣ 21
ΚΕΦΑΛΑΙΟ 3: ΠΛΑΤΦΌΡΜΑ UNREAL ENGINE 4 22
ΕΙΣΑΓΩΓΗ 22
3.1 Περιήγηση στη πλατφόρμα (UE4) 22
3.2 Ορολογία της Unreal Engine 4 24
3.2.1 Έργα “Projects” 24
3.2.2 Αντικείμενα “Objects” 25
3.2.3 Κλάσεις “Classes” 25
3.2.4 Ηθοποιοί “Actors” 25
3.2.5 Συστατικά “Components” 25
3.2.6 Πιόνια “Pawns” 25
3.2.7 Χαρακτήρες “Characters” 26
3.2.8 PlayerController 26
3.2.9 AIController 26
3.2.10 Brushes 26
3.2.11 Επίπεδα “Levels” 27
3.2.12 Κόσμος “World” 27
3.2.13 GameModes 27

3.2.14 Σύστημα Blueprints Visual Scripting 27
3.2.15 GameStates 28
3.2.16 PlayerStates 28
3.3 Εργαλεία και Συντάκτες “Tools and Editors” 28
3.3.1 Level Editor 29
3.3.2 Material Editor 29
3.3.3 Blueprint Editor 30
3.3.4 UMG UI Editor 31
3.3.5 Static Mesh Editor 31
3.3.6 Behavior Tree Editor 32
3.3.7 Persona Editor 33
3.3.8 Cascade Editor 33
3.3.9 Matinee Editor 34
3.3.10 Sound Cue Editor 35
3.3.11 Paper2D Sprite Editor 35
3.3.12 Paper2D Flipbook Editor 36
3.3.13 Physics Asset Tool Editor 37
3.3.14 Media Player Editor 37
3.4 Blueprints 38
3.4.1 Πώς λειτουργούν τα Blueprints; 38
3.4.2 Οι πιο σύνηθες τύποι Blueprint 39
3.4.3 Τι άλλο μπορεί να κάνει ένα Blueprint; 40
ΕΠΙΛΟΓΟΣ 41
ΚΕΦΑΛΑΙΟ 4: ΑΝΆΠΤΥΞΗ ΠΑΙΧΝΙΔΙΟΎ 42
ΕΙΣΑΓΩΓΗ 42
4.1 Δημιουργία Project 42
4.2 Έδαφος “Terrain” 42
4.3 Foliage 45
4.4 First Person Character Blueprint 46
4.4.1 First Person Character Custom Variables. 48
4.4.2 Intro Sequence 48
4.4.3 Note Widget 48
4.4.4 Pause Widget 50

4.4.5 Interaction Line trace 51
4.5 Main Menu Level 52
4.5.1 MainMenu HUD 52
4.5.2 MainMenuWidget Widget 52
4.6 Objectives Sequence 54
4.6.1 Door_BP 54
4.6.2 key1 55
4.6.3 Look_for_dad 56
4.6.4 Notebook 57
4.6.5 Book_Page 58
4.6.6 Magic_History 59
4.6.7 to_the_cave1 60
4.6.8 to_the_cave2 61
4.6.9 to_the_cave3 62
4.6.10 into_the_cave1 62
4.6.11 into_the_cave2 63
4.6.12 end_1 63
4.7 Main Assets 65
4.7.1 House 66
4.7.2 windmill 69
4.7.3 Boat_bp 69
4.7.4 wood_plank 70
4.7.5 rip_bp 71
4.8 Components 72
4.8.1 Exponential Height Fog 72
4.8.2 Directional Light 72
4.8.3 Sky Sphere Blueprint 72
4.8.4 Post Process Volume 72
4.8.5 Ambient Sound Actor 72
4.8.6 Decal Actor 73
4.8.7 Έτοιμα Components 73
4.9 Cinematics 76
4.10 Player Dialogues 77

ΕΠΙΛΟΓΟΣ 77
ΚΕΦΑΛΑΙΟ 5: ΣΥΜΠΕΡΑΣΜΑΤΑ 79
ΒΙΒΛΙΟΓΡΑΦΙΑ 80

ΕΥΡΕΤΗΡΙΟ ΕΙΚΟΝΩΝ

Εικόνα 1 “Το λογότυπο της Unreal Engine” 15
Εικόνα 2 “Το Unreal ήταν το πρώτο παιχνίδι με την Unreal Engine” 15
Εικόνα 3 “Το Killing Floor κατασκευάστηκε στην Unreal Engine 2” 16
Εικόνα 4 “Το BioShock Infinite κατασκευάστηκε στην Unreal Engine 3” 17
Εικόνα 5 “Το Unreal Tournament υλοποιείται με την Unreal Engine 4” 19
Εικόνα 6 “Παράθυρο Έργου” 22
Εικόνα 7 “Ρυθμίσεις Έργου” 23
Εικόνα 8 “Περιηγητής Περιεχομένου” 24
Εικόνα 9 “Σύστημα Blueprints Visual Scripting” 28
Εικόνα 10 "Level Editor" 29
Εικόνα 11 "Material Editor" 30
Εικόνα 12 "Blueprint Editor" 30
Εικόνα 13 "UMG UI Editor" 31
Εικόνα 14 "Static Mesh Editor" 32
Εικόνα 15 "Behavior Tree Editor" 32
Εικόνα 16 "Persona Editor" 33
Εικόνα 17 "Cascade Editor" 34
Εικόνα 18 "Matinee Editor" 34
Εικόνα 19 "Sound Cue Editor" 35
Εικόνα 20 "Paper2D Sprite Editor" 36
Εικόνα 21 "Flipbook Editor" 36
Εικόνα 22 "Physics Asset Tool Editor" 37
Εικόνα 23 "Media Player Editor" 38
Εικόνα 24 "Level Blueprint example" 39
Εικόνα 25 "Blueprint Class Example" 39
Εικόνα 26 "World Machine" 42
Εικόνα 27 "Landscape Import" 43
Εικόνα 28 "Terrain μετά την εισαγωγή" 44
Εικόνα 29 "Material Terrain_new" 44
Εικόνα 30 "Τελικό Landscape" 45
Εικόνα 31 "Foliage Demostration" 45
Εικόνα 32 "First Person Character 1" 47

Εικόνα 33 "First Person Character 2” 47
Εικόνα 34 "Intro Sequence Playback” 48
Εικόνα 35 "Notes Widget Activation/Deactivation" 49
Εικόνα 36 "Notes Widget In-Game"..... 49
Εικόνα 37 "Pause Widget Activation" 50
Εικόνα 38 "Pause Widget In-Game"..... 51
Εικόνα 39 "Line Trace Workflow" 52
Εικόνα 40 "Main Menu" 53
Εικόνα 41 "Main Menu Options” 54
Εικόνα 42 " Door Blueprint" 55
Εικόνα 43 "Key Blueprint"..... 56
Εικόνα 44 " Look For your Dad Blueprint" 56
Εικόνα 45 "Notebook Blueprint"..... 58
Εικόνα 46 "Book_Page Blueprint" 59
Εικόνα 47 "Magic_History Blueprint" 60
Εικόνα 48 "To the cave 1 Blueprint" 61
Εικόνα 49 "To the cave 2 Blueprint" 61
Εικόνα 50 "To the cave 3 Blueprint" 62
Εικόνα 51 "Into the cave 1 Blueprint" 63
Εικόνα 52 "Into the cave 2 Blueprint" 63
Εικόνα 53 "end_1 Blueprint"..... 64
Εικόνα 54 "End Game Widget"..... 65
Εικόνα 55 "House Blueprint" 66
Εικόνα 56 "switch Blueprint"..... 67
Εικόνα 57 "old_lamp Blueprint" 68
Εικόνα 58 "cabinet Blueprint" 69
Εικόνα 59 "windmill Blueprint" 69
Εικόνα 60 "Boat_bp Blueprint"..... 70
Εικόνα 61 "Wood plank Blueprint Nodes"..... 70
Εικόνα 62 "Wood Plank Blueprint Preview"..... 71
Εικόνα 63 "RIP Tomb Blueprint"..... 71
Εικόνα 64 "Open World Demo Collection"..... 74
Εικόνα 65 "Landscape Mountains"..... 75
Εικόνα 66 "Water Planes" 75
Εικόνα 67 "Luos Caves" 76
Εικόνα 68 "Intro Sequence"..... 77

ΚΕΦΑΛΑΙΟ 1: ΕΙΣΑΓΩΓΗ

Η ιστορία του παιχνιδιού λαμβάνει χώρα στο πατρικό σπίτι του πρωταγωνιστή το οποίο βρίσκεται σε μία ορεινή περιοχή ανάμεσα σε βουνά, δάση και λίμνες. Το τελευταίο διάστημα δεν μπορεί να επικοινωνήσει με τον πατέρα του και ανησυχεί για εκείνον παίρνοντας έτσι την απόφαση να μάθει τι συμβαίνει και να τον επισκεφτεί. Φτάνοντας αρχίζει να αναπολεί τα παιδικά του χρόνια θαυμάζοντας παράλληλα το τοπίο. Αρχικά αναζητεί τρόπο να μπει στο σπίτι διότι δεν έχει ο ίδιος κλειδί, θυμάται όμως ότι ο πατέρας του είχε ένα δεύτερο κλειδί μέσα στην βάρκα του. Αφού βρει το κλειδί και μπει στο σπίτι ψάχνει στοιχεία για το τι μπορεί να έχει συμβεί στον πατέρα του. Βρίσκει το ημερολόγιο του, μια σελίδα από ένα βιβλίο και σιγά σιγά καθοδηγούμενος από ότι έχει ανακαλύψει φτάνει πιο κοντά στην αλήθεια και το μυστήριο που κυριεύει την σπηλιά στο βουνό. Κατά την διάρκεια της ανάβασης προς την σπηλιά μονολογεί ιστορίες που του είχε πει ο πατέρας του σχετικά με την σπηλιά κατά την διάρκεια του δεύτερου παγκοσμίου πολέμου και πως ο στρατός με πρόφαση την προστασία των πολιτών σφράγισε την είσοδο της. Μπαίνοντας στην σπηλιά ανακαλύπτει ότι κάτι ζωντανό υπάρχει και ότι κάτι πολύ κακό έχει συμβεί. Φτάνοντας προς το τέλος του παιχνιδιού ανακαλύπτει το σώμα του νεκρού πατέρα του και ολοκληρώνεται το πρώτο κεφάλαιο την ιστορίας.

Για την υλοποίηση της πτυχιακής εργασίας χρησιμοποιήσαμε την μηχανή γραφικών Unreal Engine 4 στην έκδοση 4.18.2, το Blender 3D Creation Software το οποίο είναι δωρεάν και ανοιχτού κώδικά και το World Machine 3D Terrain Generator.

Η εργασία για την καλύτερη κατανόηση της έχει χωριστεί σε 5 κεφάλαια. Το πρώτο κεφάλαιο περιέχει τα εισαγωγικά, στο δεύτερο γίνεται μια εισαγωγή στην γενικότερη έννοιας μιας μηχανής γραφικών και αναλύει την ιστορία της Unreal Engine. Το τρίτο κεφάλαιο κάνει λόγο για την πλατφόρμα ανάπτυξης της Unreal Engine 4, αναλύει όρους και περιγράφει το γραφικό περιβάλλον εργασίας. Στο τέταρτο κεφάλαιο γίνεται εκτενής ανάλυση της ανάπτυξης του παιχνιδιού λεπτομερώς για κάθε συστατικό ξεχωριστά. Τέλος στο πέμπτο κεφάλαιο υπάρχουν τα συμπεράσματα.

Επιπλέον στόχος της συγκεκριμένης πτυχιακής εργασίας είναι να δώσει κίνητρα σε επόμενους φοιτητές να ασχοληθούν με το αντικείμενο της ανάπτυξης παιχνιδιών, κλάδος που βρίσκεται σε χαμηλό επίπεδο στην χώρα μας και ανθίζει σε χώρες του εξωτερικού.

ΚΕΦΑΛΑΙΟ 2: GAME ENGINE & UNREAL ENGINE

2.1 Τι είναι μια μηχανή παιχνιδιού;

Μια μηχανή παιχνιδιού είναι ένα σύστημα λογισμικού σχεδιασμένο για τη δημιουργία και την ανάπτυξη βιντεοπαιχνιδιών. Υπάρχουν πολλές μηχανές παιχνιδιών οι οποίες είναι σχεδιασμένες να δουλεύουν σε κονσόλες βιντεοπαιχνιδιών και λειτουργικά συστήματα επιτραπέζιων υπολογιστών όπως τα Microsoft Windows, το Linux, και το Mac OS X. Η κεντρική λειτουργικότητα που παρέχεται τυπικά από μια μηχανή παιχνιδιού περιλαμβάνει μια μηχανή φωτοαπόδοσης ("renderer") για 2D ή 3D γραφικά, μια μηχανή φυσικής ή εντοπισμού συγκρούσεων (collision detection, καθώς και collision response), ήχο, scripting, animation, τεχνητή νοημοσύνη, δικτύωση, streaming, διαχείριση μνήμης, νήματα (threading), υποστήριξη τοπικοποίησης, και ένα γράφο σκηνής (scene graph). Η διαδικασία της ανάπτυξης παιχνιδιού συχνά οικονομικοποιείται με το ότι σε μεγάλο μέρος η ίδια μηχανή παιχνιδιού επαναχρησιμοποιείται για να δημιουργηθούν διαφορετικά παιχνίδια.

Οι μηχανές παιχνιδιών παρέχουν μια σουίτα οπτικών εργαλείων ανάπτυξης εκτός από επαναχρησιμοποιήσιμα στοιχεία λογισμικού. Αυτά τα εργαλεία γενικά παρέχονται σε ένα ολοκληρωμένο περιβάλλον ανάπτυξης ώστε να καθιστούν ικανή την απλή, γρήγορη ανάπτυξη παιχνιδιών με ένα οδηγούμενο από δεδομένα τρόπο. Αυτές οι μηχανές παιχνιδιών συχνά καλούνται "game middleware" επειδή, όπως με την εμπορική έννοια του όρου, παρέχουν μια ευέλικτη και επαναχρησιμοποιήσιμη πλατφόρμα λογισμικού η οποία παρέχει όλη την κεντρική λειτουργικότητα που απαιτείται, αμέσως έξω από το κουτί, για την ανάπτυξη μιας εφαρμογής παιχνιδιού ενώ ταυτόχρονα μειώνει τα κόστη, τις πολυπλοκότητες, και το χρόνο μέχρι την αγορά (time-to-market) — όλοι κρίσιμοι παράγοντες στην υψηλά ανταγωνιστική βιομηχανία βιντεοπαιχνιδιών.

Όπως άλλες λύσεις middleware, οι μηχανές παιχνιδιών συνήθως παρέχουν μια αφαίρεση της πλατφόρμας, επιτρέποντας στο ίδιο παιχνίδι να τρέχει σε διάφορες πλατφόρμες συμπεριλαμβανομένων των κόνσολών παιχνιδιών και των προσωπικών υπολογιστών με λίγες, αν όχι καθόλου, αλλαγές στον πηγαίο κώδικα του παιχνιδιού. Συχνά, το middleware παιχνιδιού σχεδιάζεται με μια βασισμένη σε στοιχεία αρχιτεκτονική η οποία επιτρέπει σε συγκεκριμένα συστήματα στη μηχανή να αντικατασταθούν ή να επεκταθούν με πιο εξειδικευμένα (και συχνά πιο ακριβά) στοιχεία middleware όπως το Havok για φυσική, το FMOD για ήχο, ή το Scaleform για UI και Βίντεο. Μερικές μηχανές παιχνιδιών όπως η RenderWare είναι και ακόμη σχεδιασμένες ως μια σειρά χαλαρά συνδεδεμένων στοιχείων middleware τα οποία μπορούν να συνδυαστούν κατά βούληση για τη δημιουργία μιας προσαρμοσμένης μηχανής, αντί για την πιο κοινή προσέγγιση της επέκτασης ή της προσαρμογής μια ευέλικτης ενσωματωμένης λύσης. Με οποιοδήποτε τρόπο

και αν η επεκτασιμότητα επιτυγχάνεται, παραμένει μια υψηλή προτεραιότητα στις μηχανές παιχνιδιών λόγω της ευρείας ποικιλίας χρήσεων για την οποίες αυτές εφαρμόζονται. Παρά την συγκεκριμένη έννοια του ονόματος, οι μηχανές παιχνιδιών συχνά χρησιμοποιούνται για άλλα είδη διαδραστικών εφαρμογών με πραγματικού χρόνου γραφικές απαιτήσεις όπως επιδείξεις μάρκετινγκ, αρχιτεκτονικές οπτικοποιήσεις, εκπαιδευτικές εξομοιώσεις, και περιβάλλοντα μοντελοποίησης.

Μερικές μηχανές παιχνιδιών παρέχουν μόνο ικανότητες φωτοαπόδοσης 3D πραγματικού χρόνου (real time 3D rendering) αντί της ευρείας γκάμας λειτουργικότητας που απαιτείται από τα παιχνίδια. Αυτές οι μηχανές βασίζονται στον δημιουργό του παιχνιδιού να εφαρμόσει το υπόλοιπο αυτής της λειτουργικότητας ή να το συνθέσει από άλλα στοιχεία middleware παιχνιδιού. Αυτού του τύπου οι μηχανές γενικά αναφέρονται ως «μηχανή γραφικών», «μηχανή rendering», ή «μηχανή 3D» αντί για τον πιο περιληπτικό όρο «μηχανή παιχνιδιού». Ωστόσο, αυτή η ορολογία χρησιμοποιείται ανακόλουθα καθώς πολλές πλήρεις σε χαρακτηριστικά μηχανές 3D παιχνιδιών αναφέρονται απλά ως «μηχανές 3D». Μερικά παραδείγματα μηχανών γραφικών είναι οι εξής: RealmForge, Truevision3D, OGRE, Crystal Space, Genesis3D, Irrlicht και JMonkey Engine. Οι μοντέρνες μηχανές παιχνιδιών ή γραφικών γενικά παρέχουν ένα scene graph, ο οποίος είναι μια αντικειμενοστραφής αναπαράσταση του 3D κόσμου του παιχνιδιού η οποία συχνά απλοποιεί τον σχεδιασμό του παιχνιδιού και μπορεί να χρησιμοποιηθεί για πιο αποδοτικό rendering τεράστιων εικονικών κόσμων.

Αφαίρεση υλικού

Πιο συχνά, οι 3D μηχανές ή τα συστήματα rendering στις μηχανές παιχνιδιών κατασκευάζονται πάνω σε ένα API γραφικών, όπως τα Direct3D ή OpenGL, το οποίο παρέχει μια αφαιρετικότητα λογισμικού της GPU ή της κάρτας βίντεο. Βιβλιοθήκες χαμηλού επιπέδου όπως τα DirectX, SDL και OpenAL επίσης χρησιμοποιούνται συχνά σε παιχνίδια καθώς παρέχουν πρόσβαση ανεξάρτητη από το υλικό σε άλλο υλικό του υπολογιστή όπως συσκευές εισόδου (ποντίκι (υπολογιστές), πληκτρολόγιο και joystick), κάρτες δικτύου και κάρτες ήχου. Πριν από τα επιταχυνόμενα από το υλικό 3D γραφικά, είχαν χρησιμοποιηθεί renderers λογισμικού. Το rendering λογισμικού ακόμα χρησιμοποιείται σε κάποια εργαλεία μοντελοποίησης ή για εικόνες ακίνητα rendered όταν η οπτική ακρίβεια αποτιμάται πάνω από την επίδοση πραγματικού χρόνου (καρέ ανά δευτερόλεπτο) ή όταν το υλικό του υπολογιστή δεν συναντά απαιτήσεις όπως υποστήριξη shader ή, στην περίπτωση των Windows Vista, υποστήριξη για το Direct3D 10.

Με την ανατολή της επιταχυνόμενης από το υλικό επεξεργασίας φυσικής, διάφορα API φυσικής όπως το PAL και οι επεκτάσεις φυσικής του COLLADA (μιας ανταλλακτικής μορφής για 3D στοιχεία) έγιναν διαθέσιμα για να παρέχουν μια

αφαιρετικότητα λογισμικού της μονάδας επεξεργασίας φυσικής διαφορετικών παρόχων middleware και πλατφορμών κόνσολών.

Ιστορία

Πριν από τις μηχανές παιχνιδιών, τα παιχνίδια τυπικά γράφονταν ως μοναδικές οντότητες: ένα παιχνίδι για το Atari 2600, για παράδειγμα, έπρεπε να σχεδιαστεί από το μηδέν για να κάνει βέλτιστη χρήση του υλικού εμφάνισης – αυτή η κεντρική ρουτίνα εμφάνισης σήμερα καλείται πυρήνας από τους δημιουργούς retro. Άλλες πλατφόρμες είχαν περισσότερα περιθώρια, αλλά ακόμα και όταν η εμφάνιση δεν ήταν μέριμνα, οι περιορισμοί μνήμης συνήθως σαμπόταραν προσπάθειες για τη δημιουργία της βαριάς σε δεδομένα σχεδίασης που μια μηχανή απαιτούσε. Ακόμα και σε πιο φιλικές πλατφόρμες, πολύ λίγα μπορούσαν να χρησιμοποιηθούν ανάμεσα σε παιχνίδια. Η γοργή ανάπτυξη του arcade υλικού – ο ακρογωνιαίος λίθος της αγοράς – σήμαινε ότι το μεγαλύτερο μέρος του κώδικα θα έπρεπε να πεταχτεί μετά έτσι και αλλιώς, καθώς οι μεταγενέστερες γενιές παιχνιδιών θα χρησιμοποιούσαν τελείως διαφορετικές σχεδιάσεις παιχνιδιών οι οποίες εκμεταλλευόντουσαν επιπλέον πόρους. Παρομοίως, οι περισσότερες σχεδιάσεις παιχνιδιών τη δεκαετία του 1980 σχεδιάζονταν διαμέσου ενός σκληρά κωδικοποιημένου συνόλου κανόνων με μια μικρή ποσότητα δεδομένων επιπέδου και γραφικών.

Η πρώτη γενιά μηχανών γραφικών τρίτων μερών ή renderers κυριαρχούνταν από τρεις παίκτες: την BRender από την Argonaut Software, την Renderware από την Criterion Software Limited και την Reality Lab της RenderMorphics. Η Reality Lab ήταν η ταχύτερη από τις τρεις και ήταν η πρώτη που αναλήφθηκε σε μια επιθετική κίνηση της Microsoft. Η ομάδα της RenderMorphics, οι Servan Keondjian, Kate Seekings και Doug Rabson, ακολούθως προσχώρησαν στο εγχείρημα της Microsoft το οποίο μετέτρεψε την Reality Lab στο Direct3D, πριν οι Keondjian και Rabson φύγουν για να ξεκινήσουν μια άλλη εταιρεία middleware Qube Software. Η Renderware τελικά αγοράστηκε από την EA.

Ο όρος «μηχανή παιχνιδιού» ήρθε στην επιφάνεια στα μέσα της δεκαετίας του 1990, ειδικά συσχετισμένος με τα παιχνίδια τριών διαστάσεων, όπως τα First Person Shooter (FPS). (Δείτε επίσης: μηχανή βολών πρώτου προσώπου). Τόση ήταν η δημοτικότητα των σειρών παιχνιδιών Doom και Quake που, αντί να δουλέψουν από μηδενική βάση, άλλοι δημιουργοί πήραν άδεια για τα κεντρικά κομμάτια του λογισμικού και σχεδίασαν τα δικά τους γραφικά, χαρακτήρες, όπλα και επίπεδα – το «περιεχόμενο του παιχνιδιού» ή τα «στοιχεία του παιχνιδιού». Ο διαχωρισμός των ειδικών για το παιχνίδι κανόνων και δεδομένων από βασικές έννοιες όπως ο έλεγχος σύγκρουσης και οι οντότητες του παιχνιδιού σήμαινε ότι οι ομάδες μπορούσαν να μεγαλώσουν και να ειδικευτούν.

Μεταγενέστερα παιχνίδια, όπως το Quake III Arena και το Unreal του 1998 της Epic Games σχεδιάζονταν με αυτή την προσέγγιση, μηχανή και περιεχόμενο αναπτυγμένα ξεχωριστά. Η πρακτική της αδειοδότησης τέτοιας τεχνολογίας έχει αποδειχθεί ότι είναι μια χρήσιμη βοηθητική πηγή εσόδων για κάποιους δημιουργούς παιχνιδιών, καθώς μια μόνο άδεια για μια υψηλής ποιότητας εμπορική μηχανή παιχνιδιού μπορεί να κυμανθεί από $10,000 μέχρι εκατομμύρια δολάρια, και ο αριθμός των αποδεκτών αδειών μπορεί να φτάσει αρκετές ντουζίνες εταιρειών (όπως έγινε με την Unreal Engine. Τουλάχιστον, οι επαναχρησιμοποιήσιμες μηχανές κάνουν την ανάπτυξη συνεχειών παιχνιδιών γρηγορότερη και ευκολότερη, το οποίο είναι ένα πλεονέκτημα αξίας στην ανταγωνιστική βιομηχανία βιντεοπαιχνιδιών.

Οι μοντέρνες μηχανές παιχνιδιών αποτελούν μερικές από τις πιο σύνθετες εφαρμογές που έχουν αναπτυχθεί, με συχνά να χαρακτηρίζονται από τελειοποιημένα συστήματα που αλληλεπιδρούν για να εγγυώνται μια ακριβώς ελεγχόμενη εμπειρία χρήστη. Η συνεχής εξέλιξη των μηχανών παιχνιδιών έχει δημιουργήσει ένα ισχυρό διαχωρισμό ανάμεσα στο rendering, το scripting, την τέχνη, και το σχεδιασμό επιπέδων. Είναι τώρα κοινό, για παράδειγμα, για μια τυπική ομάδα ανάπτυξης παιχνιδιών να έχει πολλές φορές περισσότερους καλλιτέχνες απ' ότι προγραμματιστές.

Τα παιχνίδια First Person Shooter παραμένουν οι κυρίαρχοι χρήστες third-party μηχανών παιχνιδιών, αλλά πλέον χρησιμοποιούνται και σε άλλα είδη. Για παράδειγμα, το βιντεοπαιχνίδι ρόλων The Elder Scrolls III: Morrowind και το MMORPG Dark Age of Camelot βασίζονται στη μηχανή Gamebryo, ενώ το επίσης MMORPG,Lineage II, στην Unreal Engine. Οι μηχανές παιχνιδιών χρησιμοποιούνται επίσης για παιχνίδια αρχικά ανεπτυγμένα για οικιακές κονσόλες: για παράδειγμα η μηχανή RenderWare χρησιμοποιείται στα franchises Grand Theft Auto και Burnout.

Το Threading αποκτά μεγαλύτερη σημασία λόγω των σύγχρονων συστημάτων πολλαπλών πυρήνων (π.χ. Cell) και των αυξημένων απαιτήσεων στον ρεαλισμό. Τυπικά τα threads περιλαμβάνουν το rendering, το streaming, τον ήχο, και τα Physics. Τα Racing Games έχουν τυπικά βρεθεί στην πρώτη γραμμή του προγραμματισμού με threads, με την Physics Engine να τρέχει σε ένα ξεχωριστό νήμα πολύ πιο πριν συγκριτικά με άλλα υποσυστήματα του πυρήνα, εν μέρη επειδή το rendering και σχετικές εργασίες απαιτούν ενημέρωση στα 30-60 Hz μόνο. Για παράδειγμα, το Need For Speed στο Playstation τρέχει τα Physics του στα 100 Hz συγκρινόμενα με το Forza Motorsport 2 που τρέχει τα Physics του στα 360 Hz.

Αν και ο όρος πρωτοχρησιμοποιήθηκε στη δεκαετία του 1990, υπάρχουν κάποια προγενέστερα συστήματα στη δεκαετία του 1980 τα οποία επίσης θεωρούνται μηχανές παιχνιδιών, όπως τα συστήματα AGI και SCI της Sierra, το σύστημα SCUMM της LucasArts και η Μηχανή Freescape της Incentive Software.

Ωστόσο, αντίθετα με τις μοντέρνες μηχανές παιχνιδιών, αυτά τα συστήματα δεν χρησιμοποιήθηκαν ποτέ σε προϊόντα τρίτων μερών (εκτός του συστήματος SCUMM το οποίο πήρε άδεια και χρησιμοποιήθηκε από την Humongous Entertainment).

Πρόσφατες τάσεις

Καθώς η τεχνολογία των μηχανών παιχνιδιών ωριμάζει και γίνεται πιο φιλική προς το χρήστη, η εφαρμογή των μηχανών παιχνιδιών έχει διευρυνθεί σε έκταση, και τώρα χρησιμοποιείται για σοβαρά παιχνίδια: οπτικοποίησης, εκπαίδευσης, ιατρικής και εφαρμογές στρατιωτικής εξομοίωσης. Για να διευκολύνουν αυτή την προσβασιμότητα, νέες πλατφόρμες υλικού στοχεύονται τώρα από μηχανές παιχνιδιών, περιλαμβανομένου κινητών τηλεφώνων (π.χ. το iPhone) και φυλλομετρητών (π.χ. Shockwave, Flash, Silverlight, Unity Web Player, O3D).

Επιπρόσθετα, περισσότερες μηχανές παιχνιδιών κατασκευάζονται πάνω σε Υψηλού επιπέδου γλώσσες προγραμματισμού όπως η Java και η C#/.NET (π.χ. TorqueX, Blade3D, και Visual3D.NET) ή Python (Panda3D). Καθώς τα περισσότερα από τα πλούσια σε 3D παιχνίδια είναι περισσότερο περιορισμένα από την GPU (δηλαδή περιορισμένα από την ισχύ της κάρτας γραφικών), οι ενδεχόμενες καθυστερήσεις των υψηλού επιπέδου γλωσσών προγραμματισμού γίνονται αμελητέες, ενώ τα οφέλη παραγωγικότητας που προσφέρονται από αυτές τις γλώσσες στους δημιουργούς μηχανών παιχνιδιών γίνονται κέρδος. Αυτές οι πρόσφατες τάσεις προωθούνται από εταιρείες όπως η Microsoft για να υποστηρίξουν ανεξάρτητη ανάπτυξη παιχνιδιών σε περισσότερες πλατφόρμες, όπως το Xbox360 και το Zune χρησιμοποιώντας το .NET Framework και το XNA για γραφικά και rendering ήχου. Γίνεται ευκολότερο και φθηνότερο από ποτέ να δημιουργηθούν μηχανές παιχνιδιών για πλατφόρμες οι οποίες υποστηρίζουν διαχειριζόμενα πλαίσια.

2.2 Τι είναι η Unreal Engine

Η Unreal Engine είναι μια μηχανή παιχνιδιών που αναπτύχθηκε από την Epic Games. Παρουσιάστηκε για πρώτη φορά στο first-person shooter παιχνίδι Unreal του 1998. Παρόλο που αναπτύχθηκε κυρίως για shooters πρώτου προσώπου, έχει χρησιμοποιηθεί με επιτυχία σε διάφορα άλλα είδη, όπως stealth, παιχνίδια μάχης, MMORPGs και άλλα RPGs. Με τον κώδικα γραμμένο σε C ++, η Unreal Engine διαθέτει υψηλό βαθμό φορητότητας και είναι ένα εργαλείο που χρησιμοποιείται από πολλούς προγραμματιστές παιχνιδιών σήμερα. Έχει κερδίσει πολλά βραβεία, συμπεριλαμβανομένου του βραβείου Guinness World Records για την "πιο επιτυχημένη μηχανή βιντεοπαιχνιδιών". Η τρέχουσα έκδοση είναι η Unreal Engine 4, σχεδιασμένη για τα Microsoft Windows, MacOS, Linux, SteamOS, HTML5, iOS, Android, Nintendo Switch, PlayStation 4, Xbox One, Magic Leap

One και εικονική πραγματικότητα (SteamVR / HTC Vive, Oculus Rift, PlayStation VR, Google Daydream, OSVR και Samsung Gear VR).

Εικόνα 1 “Το λογότυπο της Unreal Engine”

2.3 Η ιστορία της Unreal Engine

2.3.1 Unreal Engine 1

Εικόνα 2 “Το Unreal ήταν το πρώτο παιχνίδι με την Unreal Engine”

Η ανάπτυξη της πρώτης γενιάς Unreal Engine πραγματοποιήθηκε από τον ιδρυτή των Epic Games, Tim Sweeney. Εμπνευσμένος από το πρωτοποριακό πρόγραμμα προγραμματισμού του John Carmack για το Doom και στη συνέχεια για το Quake, ο Sweeney ξεκίνησε την μηχανή το 1995 για την παραγωγή ενός παιχνιδιού που αργότερα θα αποκαλούσε Unreal, ένα shooter πρώτου προσώπου σε έναν μεσαιωνικό κόσμο με ξένα στοιχεία. Μετά από τρία χρόνια ανάπτυξης, ξεκίνησε με την δημοσίευση του παιχνιδιού το 1998, παρόλο που οι κάτοχοι άδειας όπως η MicroProse και η Legend Entertainment είχαν την τεχνολογία πολύ νωρίτερα, με την πρώτη συμφωνία αδειοδότησης να πραγματοποιείται το 1996. Τόσο η απόδοση λογισμικού (software render) όσο και η απόδοση υλικού (hardware render) υπήρχαν στο βασικό λογισμικό, καθώς και ανίχνευση

σύγκρουσης (collision detection), έγχρωμος φωτισμός (colored lighting) και μια στοιχειώδης έκδοση φιλτραρίσματος υφής (texture filtering). Η μηχανή επίσης παρείχε την δυνατότητα επεξεργασίας επιπέδου (level editor), που ονομάζεται UnrealEd, που είχε υποστήριξη για λειτουργικές κατασκευές στερεάς γεωμετρίας σε πραγματικό χρόνο ήδη από το 1996, επιτρέποντας στους χαρτογράφους να αλλάξουν τη διάταξη επιπέδων "εν κινήσει". Άλλα χαρακτηριστικά που υλοποιήθηκαν κατά τη διάρκεια της ανάπτυξης περιελάμβαναν άμεσο φωτισμό σε πραγματικό χρόνο και πηγές φωτός (real-time direct illumination and light sourcing), τα οποία ενσωματώθηκαν αντιστοίχως το 1995 και το 1997. Εκτός από την υποστήριξη των Microsoft Windows, Linux και Mac, το Unreal Tournament άνοιξε επίσης την πλατφόρμα στο PlayStation 2 και, με τη βοήθεια του Secret Level, στο Dreamcast. Το 2000, η Epic ενημέρωσε την μηχανή με νέες βελτιώσεις, συμπεριλαμβανομένων μοντέλων και αρχιτεκτονικής υψηλότερου πολύγωνου, σκελετικού συστήματος κινούμενων σχεδίων και μεγάλης κλίμακας υποστήριξης εδάφους.

Στα τέλη του 1999, οι The New York Times ανέφεραν ότι ο αριθμός των έργων που χρησιμοποίησαν την τεχνολογία της Epic ήταν 16, ονομάζοντας τα ονόματα Deus Ex, Nerf Arena Blast και Duke Nukem Forever, ο τίτλος από τα 3D Realms που είχε προγραμματιστεί να κάνει ντεμπούτο στην κονσόλα του GameCube. Ενώ κόστισε περίπου 3 εκατομμύρια δολάρια για να υλοποιηθεί και με το κόστος αδειοδότησης να φτάνει μέχρι και τα 350.000 δολάρια, η Epic έδωσε στους Modders τη δυνατότητα να δημιουργήσουν τους δικούς τους κόσμους με την ενσωμάτωση του UnrealEd και μιας γλώσσας scripting που ονομάζεται UnrealScript, προκαλώντας μια κοινότητα enthusiasts γύρω από μια μηχανή παιχνιδιών που χτίστηκε για να είναι επεκτάσιμη και βελτιωμένη σε πολλαπλές γενιές παιχνιδιών.

2.3.2 Unreal Engine 2

Εικόνα 3 “Το Killing Floor κατασκευάστηκε στην Unreal Engine 2”

Η δεύτερη έκδοση έκανε το ντεμπούτο της το 2002 με το “America’s Army”, ένα δωρεάν shooter παιχνίδι για πολλούς παίκτες που ανέπτυξε ο στρατός των Η.Π.Α. ως εργαλείο πρόσληψης. Αν και βασισμένη σε προηγούμενη τεχνολογία, αυτή η γενιά είδε την απόδοση γραφικών να ξαναγράφεται εντελώς και να συμπεριλαμβάνει μια ποικιλία χαρακτηριστικών όπως το κινηματογραφικό εργαλείο επεξεργασίας Matinee, εξαγωγή plug-ins για το 3D Studio Max και Maya και τη Karma physics engine ένα εργαλείο από το Math Engine που τροφοδοτούσε τη φυσική του ragdoll στο Unreal Tournament 2003. Επιπλέον, παρουσιάστηκε το UnrealEd 2, το οποίο έκανε το ντεμπούτο του με την προηγούμενη γενιά του κινητήρα και λίγο αργότερα ακολούθησε το UnrealEd 3. Άλλα στοιχεία της μηχανής ενημερώθηκαν επίσης, με βελτιωμένα στοιχεία ενεργητικού καθώς και με την προσθήκη υποστήριξης για το Xbox.

UE2.5, μία ενημερωμένη έκδοση της UE2, βελτίωσε την απόδοση γραφικών και διέθεται φυσική οχημάτων (vehicles physics), επεξεργαστή συστήματος σωματιδίων (particle system editor) για το UnrealEd και υποστήριξη 64 bit στο Unreal Tournament 2004. Μια εξειδικευμένη έκδοση της UE2 που ονομάζεται UE2X χρησιμοποιήθηκε για το “Unreal Championship 2: The Liandri Conflict” στην αρχική πλατφόρμα Xbox, που περιλάμβανε βελτιστοποιήσεις που αφορούσαν την συγκεκριμένη κονσόλα.

Τον Μάρτιο του 2011, η Ubisoft του Μόντρεαλ αποκάλυψε ότι η UE2 λειτούργησε με επιτυχία στο Nintendo 3DS.

2.3.3 Unreal Engine 3

Εικόνα 4 “Το BioShock Infinite κατασκευάστηκε στην Unreal Engine 3”

Στιγμιότυπα οθόνης της Unreal Engine 3 παρουσιάστηκαν το 2004, οπότε η μηχανή είχε ήδη αναπτυχθεί για πάνω από 18 μήνες. Σε αντίθεση με την Unreal Engine 2, η Unreal Engine 3 σχεδιάστηκε για να εκμεταλλευτεί πλήρως προγραμματιζόμενο shader hardware. Όλοι οι υπολογισμοί φωτισμού γινόντουσαν ανά pixel, αντί για vertex. Στην πλευρά απόδοσης, η Unreal Engine 3 παρείχε υποστήριξη για gamma-correct high-dynamic range renderer.

Αρχικά, η Unreal Engine 3 υποστήριζε μόνο τις πλατφόρμες Windows, PlayStation 3 και Xbox 360, ενώ το iOS (που παρουσιάστηκε για πρώτη φορά με το Epic Citadel) και το Android προστέθηκαν αργότερα το 2010, με το Infinity Blade να είναι ο πρώτος τίτλος iOS και το Dungeon Defenders ο πρώτος τίτλος Android. Η υποστήριξη OS X προστέθηκε το 2011. Την ίδια χρονιά ανακοινώθηκε ότι η μηχανή θα υποστηρίζει το Adobe Flash Player 11 και ότι χρησιμοποιείται σε δύο παιχνίδια Wii U, Batman: Arkham City και Aliens: Colonial Marines. Υποστήριξη Windows 8 και Windows RT προστέθηκε το 2012. Το 2013, η Epic συνεργάστηκε με το Mozilla για να φέρει την Unreal Engine 3 σε HTML5. Χρησιμοποιώντας την υπογλώσσα asm.js και τον μεταγλωττιστή Emscripten, ήταν σε θέση να μεταφέρουν την μηχανή σε JavaScript και WebGL σε τέσσερις ημέρες.

Καθ 'όλη τη διάρκεια ζωής της UE3, έχουν ενσωματωθεί σημαντικές ενημερώσεις, συμπεριλαμβανομένων των βελτιωμένων καταστρεφόμενων περιβάλλοντων (destructible environments), της μαλακής δυναμικής του σώματος (soft body dynamics), της προσομοίωσης μεγάλου πλήθους (large crowd simulation), της λειτουργικότητας του iPod Touch, της ενσωμάτωσης του Steamworks, μιας παγκόσμιας λύσης φωτισμού σε πραγματικό χρόνο, και το στερεοσκοπικό 3D στο Xbox 360 μέσω της τεχνολογίας TriOviz for Games Η υποστήριξη του DirectX 11 παρουσιάστηκε με το demo του Samaritan, το οποίο αποκαλύφθηκε στο συνέδριο Developers Game 2011 και χτίστηκε από την Epic Games σε στενή συνεργασία με τη NVIDIA, με μηχανικούς που εργάζονταν σε όλη τη χώρα για να ωθήσουν τα γραφικά σε πραγματικό χρόνο σε ένα νέο υψηλό σημείο.

Unreal Development Kit

Ενώ η Unreal Engine 3 ήταν αρκετά open source για να συνεργαστεί με τους Modders, η δυνατότητα δημοσίευσης και πώλησης παιχνιδιών που έγιναν με χρήση UE3 περιορίστηκε στους αδειούχους της μηχανής. Ωστόσο, το Νοέμβριο του 2009, η Epic κυκλοφόρησε μια δωρεάν έκδοση του SDK του UE3, που ονομάζεται Unreal Development Kit (UDK), που διατίθεται στο ευρύ κοινό.

Τον Δεκέμβριο του 2010, ανανεώθηκε για να συμπεριλάβει υποστήριξη για τη δημιουργία παιχνιδιών και εφαρμογών iOS.

2.3.3 Unreal Engine 4

Εικόνα 5 "Το Unreal Tournament υλοποιείται με την Unreal Engine 4"

Τον Αύγουστο του 2005, ο Mark Rein, αντιπρόεδρος της Epic Games, αποκάλυψε ότι η Unreal Engine 4 βρισκόταν σε εξέλιξη από το 2003. Μέχρι το 2008, η ανάπτυξη "βασικά" έγινε από τον Tim Sweeney, Διευθύνοντα Σύμβουλο και ιδρυτή της Epic Games . Η μηχανή στοχεύει στην όγδοη γενιά κονσολών, υπολογιστών και συσκευών με τεχνολογία Tegra K1 συσκευές Android, που ανακοινώθηκαν τον Ιανουάριο του 2014 στο CES.

Τον Φεβρουάριο του 2012, ο Mark Rein δήλωσε ότι "οι άνθρωποι θα συγκλονιστούν αργότερα φέτος όταν θα δουν την Unreal Engine 4" . Η Unreal Engine 4 παρουσιάστηκε σε περιορισμένους συμμετέχοντες στο συνέδριο προγραμματιστών παιχνιδιών για το 2012, και το video της μηχανής που παρουσιάζεται από τον τεχνικό καλλιτέχνη Alan "Talisman" Willard κυκλοφόρησε στο κοινό στις 7 Ιουνίου 2012 μέσω της GameTrailers TV . Αυτό το demo δημιουργήθηκε σε έναν υπολογιστή με τρεις GeForce GTX 580 (tri SLI) και μπορεί να τρέξει σε έναν υπολογιστή με GeForce GTX 680.

Ένα από τα κύρια χαρακτηριστικά που σχεδιάστηκαν για την UE4 ήταν ο φωτισμός global illumination σε πραγματικό χρόνο χρησιμοποιώντας τον εντοπισμό κώνου voxel, εξαλείφοντας τον προ-υπολογισμένο φωτισμό. Ωστόσο, αυτή η δυνατότητα έχει αντικατασταθεί από έναν παρόμοιο αλλά λιγότερο computationally-expensive αλγόριθμο πριν από την απελευθέρωση για όλες τις πλατφόρμες, συμπεριλαμβανομένου του Η/Υ, λόγω προβλημάτων απόδοσης . Η

UE4 περιλαμβάνει επίσης νέες λειτουργίες προγραμματιστών για τη μείωση του χρόνου επανάληψης και επιτρέπει την ενημέρωση του κώδικα C++ κατά τη λειτουργία της μηχανής. Το νέο οπτικό σύστημα scripting "Blueprint" (διάδοχος του Kismet της UE3) επιτρέπει την ταχεία ανάπτυξη της λογικής του παιχνιδιού χωρίς τη χρήση της C ++ και περιλαμβάνει ζωντανή αποσφαλμάτωση. Το αποτέλεσμα είναι ο μειωμένος χρόνος επανάληψης και λιγότερη διαφορά μεταξύ τεχνικών καλλιτεχνών, σχεδιαστών και προγραμματιστών.

Στις 19 Μαρτίου 2014, στο συνέδριο προγραμματιστών παιχνιδιών, η Epic Games κυκλοφόρησε στην dev κοινότητα μέσω ενός νέου μοντέλου συνδρομής, την Unreal Engine 4, τα χαρακτηριστικά, τον πλήρη πηγαίο κώδικα C++ και όλα τα εργαλεία της. Ο Tim Sweeney, δήλωσε ότι το νέο επιχειρηματικό μοντέλο αντικατοπτρίζει τις αλλαγές στη βιομηχανία. Η Epic Games διέθετε την Unreal Engine σε μεγάλες ομάδες ανάπτυξης παιχνιδιών AAA με κόστος εκατομμυρίων δολαρίων, αλλά καθώς η βιομηχανία έχει εξελιχθεί, η Epic αναγκάστηκε να «Επανεξετάσει πραγματικά όλη την βιομηχανία μας για το πώς διαθέτουμε την μηχανή γραφικών. Κοιτάζοντας τη νέα μορφή της βιομηχανίας τώρα, συνειδητοποιούμε ότι αυτό είναι ένα ξεπερασμένο εργαλείο ", ανέφερε ο Sweeney. "Μελετώντας τις δυνατότητες της μηχανής, ξεκινήσαμε από το μηδέν και σκεφτήκαμε πώς μπορούμε να κάνουμε την μηχανή διαθέσιμη σε περισσότερους ανθρώπους;"

Στις 4 Σεπτεμβρίου 2014, η Epic κυκλοφόρησε δωρεάν την Unreal Engine 4 σε σχολεία και πανεπιστήμια, συμπεριλαμβανομένων προσωπικών αντιγράφων για φοιτητές εγγεγραμμένους σε αναγνωρισμένα προγράμματα βιντεοπαιχνιδιών, επιστήμης υπολογιστών, τέχνης, αρχιτεκτονικής, προσομοίωσης και οπτικοποίησης.

Στις 19 Φεβρουαρίου 2015, η Epic εγκαινίασε το Unreal Dev Grants, ένα αναπτυξιακό ταμείο ύψους 5 εκατομμυρίων δολαρίων που σχεδιάστηκε για να παρέχει χρηματοοικονομικές επιχορηγήσεις σε καινοτόμα έργα που κατασκευάστηκαν με UE4.

Από τις 2 Μαρτίου 2015, η Unreal Engine 4 είναι διαθέσιμη δωρεάν για όλους, μαζί με όλες τις μελλοντικές ενημερώσεις, με επιλεκτικό χρονοδιάγραμμα δικαιωμάτων Η Oculus VR ανακοίνωσε τον Οκτώβριο του 2016 ότι θα καλύψει τα τέλη δικαιωμάτων για όλους τους τίτλους Unreal Engine που μεταφέρονται στο Oculus Store μέχρι τα πρώτα ακαθάριστα έσοδα ανά παιχνίδι των 5 εκατομμυρίων δολαρίων.

Προκειμένου να προετοιμαστεί για την κυκλοφορία του free-to-play τύπου "Battle Royale" mode στο Fortnite τον Σεπτέμβριο του 2017, η Epic χρειάστηκε να κάνει μια σειρά τροποποιήσεων της Unreal Engine που την βοήθησαν να χειριστεί έναν μεγάλο αριθμό (μέχρι 100) συνδέσεων με τον ίδιο server διατηρώντας ταυτόχρονα υψηλό εύρος ζώνης και βελτιώνοντας την απόδοση ενός μεγάλου

ανοιχτού κόσμου στο παιχνίδι. Η Epic θα ενσωματώσει αυτές τις αλλαγές σε μελλοντικές ενημερώσεις του Unreal Engine.

Οι πλατφόρμες που υποστηρίζονται αυτήν την περίοδο είναι Microsoft Windows. macOS, Linux, SteamOS, HTML5, iOS, Android, Nintendo Switch, PlayStation 4, Xbox One, Magic Leap One and virtual reality (SteamVR/HTC Vive, Oculus Rift, PlayStation VR, Google Daydream, OSVR and Samsung Gear VR).

ΕΠΙΛΟΓΟΣ

Κλείνοντας το συγκεκριμένο κεφάλαιο έχουμε καλύψει τα εισαγωγικά θέματα όπως το τι είναι μια μηχανή γραφικών, πιο συγκεκριμένα μάθαμε πληροφορίες για την μηχανή γραφικών Unreal Engine και της ιστορική της διαδρομή από την ίδρυση και την δημιουργία μέχρι το σήμερα.

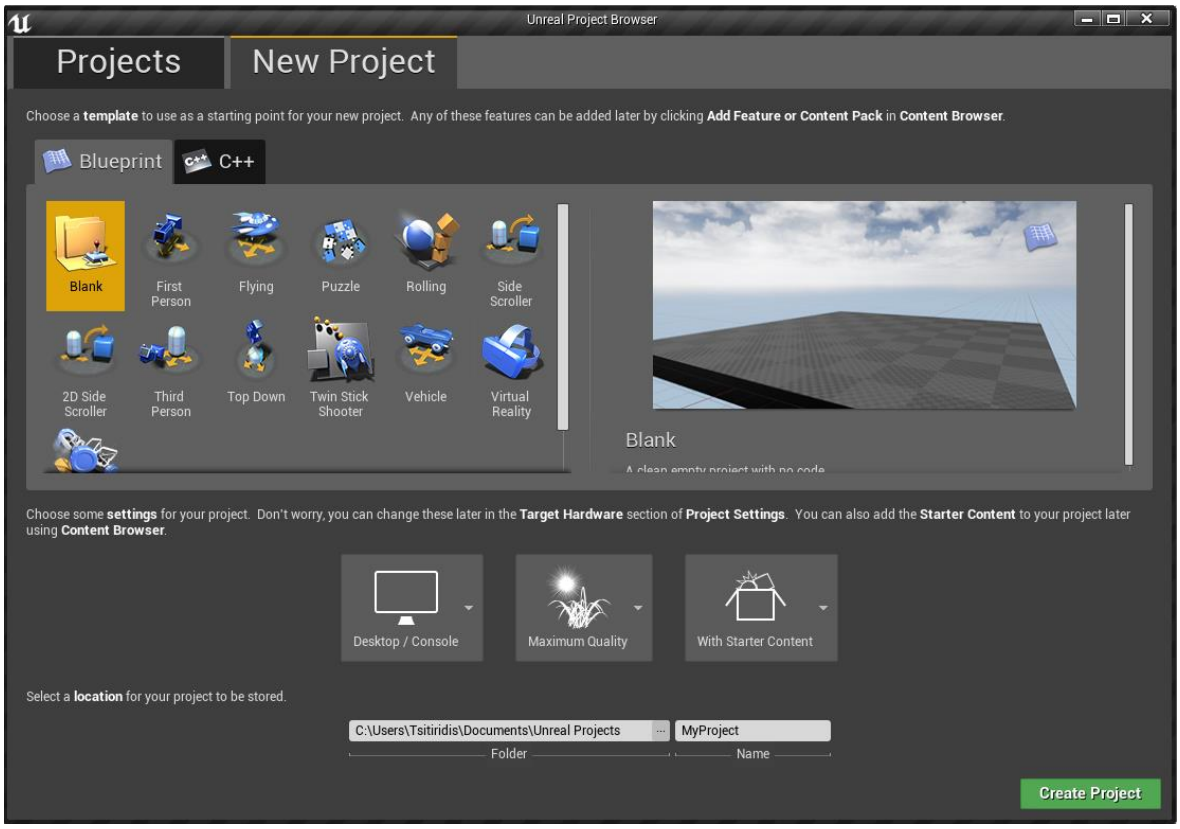
ΚΕΦΑΛΑΙΟ 3: ΠΛΑΤΦΌΡΜΑ UNREAL ENGINE 4

ΕΙΣΑΓΩΓΗ

Υπάρχουν πολλές διαθέσιμες εκδόσεις της πλατφόρμας καθώς η Epic Games την ενημερώνει και αναβαθμίζει συνεχώς. Η έκδοση της πλατφόρμας που χρησιμοποιήσαμε είναι η 14.8.2.

3.1 Περιήγηση στη πλατφόρμα (UE4)

Όταν εκκινούμε την πλατφόρμα εμφανίζεται το παράθυρο περιήγησης. Το παράθυρο περιήγησης project μας δίνει τη δυνατότητα να επιλέξουμε ένα υπάρχον έργο ή να δημιουργήσουμε ένα νέο, το οποίο μπορεί να είναι ένα έργο Blueprint ή C ++.



Εικόνα 6 "Παράθυρο Έργου"

Η καρτέλα "New Project" μας δίνει αρχικά πρότυπα για τα project μας. Το έργο Blank δημιουργεί ένα εντελώς άδειο έργο. Τα άλλα πρότυπα εμπίπτουν σε δύο κατηγορίες: Μόνο Blueprints και C ++.

Για παράδειγμα, υπάρχει ένα πρότυπο Side Scroller (Μόνο Blueprints) και ένα πρότυπο Side Scroller (C ++). Τα παιχνίδια που παράγονται από αυτά τα δύο πρότυπα παίζουν με τον ίδιο τρόπο, με το ίδιο επίπεδο σχεδιασμού, συμπεριφορά

χαρακτήρων και διάταξη κάμερας. Η διαφορά είναι πώς δημιουργείται το αρχικό πλαίσιο για τα έργα:

Μόνο Blueprints - Το αρχικό gameplay ορίζεται με Blueprints. Τα Blueprints είναι τα στοιχεία περιγραφής της Unreal Engine 4. Με τα Blueprints, είναι δυνατό να δημιουργήσουμε συμπεριφορά παιχνιδιού στην Unreal χωρίς να χρειαστεί να γράψουμε κώδικα C ++. Ωστόσο, ξεκινώντας από ένα πρότυπο Blueprints Only δεν σημαίνει ότι ποτέ δεν μπορούμε να συμπεριλάβουμε τον κώδικα C ++ στο έργο μας. Απλώς σημαίνει ότι τα αρχικά παραδείγματα που παρέχονται θα είναι σε Blueprints.

C ++ - Το αρχικό gameplay ορίζεται με τον κώδικα C ++. Τα έργα που δημιουργούνται με τα πρότυπα C ++ μπορούν να διαχειριστούν μέσω του editor της Unreal, αλλά μπορούν επίσης να επεξεργαστούν στο Visual Studio. Ακριβώς όπως τα έργα Blueprints Only μπορούν να επεκταθούν με τον κώδικα C ++, είναι εξίσου απλό να προσθέσουμε Blueprints σε ένα έργο που ξεκίνησε ως πρότυπο C ++.

Για να δημιουργήσουμε ένα νέο έργο:

1. Επιλέγουμε ένα πρότυπο από τη λίστα.
2. Επιλέγουμε αν θέλουμε ή όχι να περιέχει Περιεχόμενο εκκίνησης (Starter Content) στο έργο μας.
3. Καταχωρούμε ένα όνομα για το έργο μας.
4. Κάνουμε κλικ στην επιλογή Δημιουργία.

Για να αποθηκεύσουμε το έργο μας έξω από την προεπιλεγμένη τοποθεσία αποθήκευσης, μπορούμε να κάνουμε κλικ στο εικονίδιο με το βέλος για να αλλάξουμε την εμφάνιση της προεπισκόπησης πλήρους διαδρομής και δημιουργίας αρχείου.

Ρυθμίσεις έργου

Εικόνα 7 “Ρυθμίσεις Έργου”

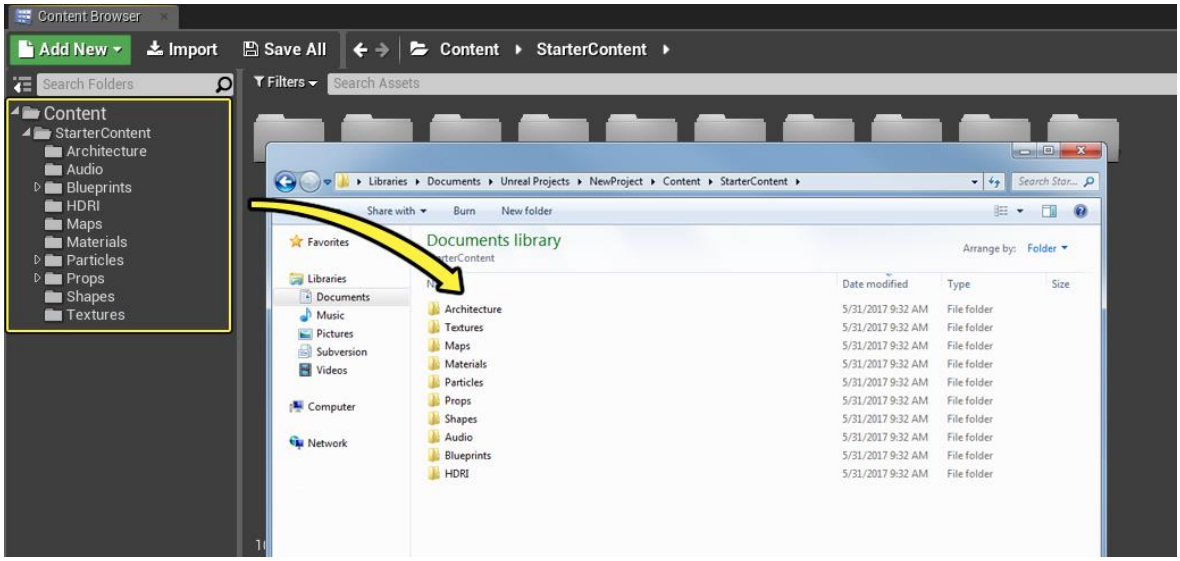
Οι Ρυθμίσεις έργου μας επιτρέπουν να ορίσουμε διαφορετικές επιλογές απόδοσης για το έργο μας ανάλογα με το είδος του υλικού που στοχεύουμε.

3.2 Ορολογία της Unreal Engine 4

Αυτό το σημείο είναι αφιερωμένο στην περιγραφή των συνήθως χρησιμοποιούμενων όρων κατά την εργασία με την Unreal Engine 4. Για παράδειγμα, αν βρεθούμε να ρωτάμε ερωτήσεις όπως "Τι είναι ένας Actor", "Τι είναι Component" ή "Τι είναι Pawn" εδώ δίνονται απαντήσεις σε αυτούς τους τύπους ερωτήσεων και πολλά άλλα.

3.2.1 Έργα "Projects"

Ένα έργο είναι μια αυτόνομη μονάδα που περιέχει όλο το περιεχόμενο και τον κώδικα που συνθέτουν ένα μεμονωμένο παιχνίδι και συμπίπτει με ένα σύνολο καταλόγων στο δίσκο μας. Για παράδειγμα, στην εικόνα κάτω από το Δέντρο ιεραρχίας του Περιηγητή Περιεχομένου περιέχει την ίδια δομή καταλόγου που βρέθηκε στο φάκελο του Έργου στο δίσκο μας.



Εικόνα 8 "Περιηγητής Περιεχομένου"

Παρόλο που ένα Έργο συχνά αναφέρεται από το αρχείο .uproject που σχετίζεται με αυτό, είναι δύο ξεχωριστά αρχεία που υπάρχουν παράλληλα. Το .uproject είναι ένα αρχείο αναφοράς που χρησιμοποιείται για τη δημιουργία, το άνοιγμα ή την αποθήκευση ενός αρχείου, ενώ το Project περιέχει όλα τα αρχεία και τους φακέλους που σχετίζονται με αυτό.

Μπορούμε να δημιουργήσουμε οποιοδήποτε αριθμό διαφορετικών έργων τα οποία μπορούν να διατηρηθούν και να αναπτυχθούν παράλληλα. Τόσο η Μηχανή Γραφικών (όσο και ο Editor) μπορούν εύκολα να εναλλάσσονται μεταξύ τους, πράγμα που μας επιτρέπει να εργαζόματε σε πολλαπλά παιχνίδια

ταυτόχρονα ή να έχουμε πολλά δοκιμαστικά προγράμματα εκτός από το κύριο Project μας.

3.2.2 Αντικείμενα "Objects"

Τα βασικά δομικά στοιχεία της Unreal Engine ονομάζονται Αντικείμενα και περιέχουν πολλές ουσιαστικές λειτουργίες για τα περιουσιακά στοιχεία του παιχνιδιού μας (game assets). Σχεδόν τα πάντα στην Unreal Engine 4 κληρονομούν (ή παίρνουν κάποια λειτουργικότητα) από ένα αντικείμενο. Στην C ++, το UObject είναι η βασική κλάση όλων των αντικειμένων. ενσωματώνει χαρακτηριστικά όπως garbage collections, υποστήριξη μεταδεδομένων (UProperty) για την έκθεση μεταβλητών στον επεξεργαστή Unreal και σειριοποίηση για φόρτωση και αποθήκευση.

3.2.3 Κλάσεις "Classes"

Μια κλάση ορίζει τις συμπεριφορές και τις ιδιότητες ενός συγκεκριμένου ηθοποιού (actor) ή αντικειμένου (object) που χρησιμοποιείται στη δημιουργία ενός παιχνιδιού στην Unreal Engine. Οι κλάσεις είναι ιεραρχικές, που σημαίνει ότι μια τάξη κληρονομεί πληροφορίες από τις μητρικές της κλάσεις (τις κατηγορίες που προήλθε ή από τις υποκλάσεις) και διαβιβάζει αυτές τις πληροφορίες στα παιδιά της. Οι κλάσεις μπορούν να δημιουργηθούν σε κώδικα C ++ ή σε Blueprints.

3.2.4 Ηθοποιοί "Actors"

Ένας ηθοποιός είναι οποιοδήποτε αντικείμενο που μπορεί να τοποθετηθεί σε ένα επίπεδο. Οι ηθοποιοί είναι μια γενική κλάση που υποστηρίζει τρισδιάστατους μετασχηματισμούς όπως μετάφραση, περιστροφή και κλίμακα. Οι ηθοποιοί μπορούν να δημιουργηθούν (spawned) και να καταστραφούν μέσω κώδικα παιχνιδιού (C ++ ή Blueprints). Στην C ++, ο AActor είναι η βασική κατηγορία όλων των Ηθοποιών.

Υπάρχουν διάφοροι τύποι ηθοποιών, μερικά παραδείγματα περιλαμβάνουν τα εξής: StaticMeshActor, CameraActor και PlayerStartActor.

3.2.5 Συστατικά "Components"

Ένα συστατικό είναι ένα κομμάτι λειτουργικότητας που μπορεί να προστεθεί σε έναν ηθοποιό. Τα συστατικά στοιχεία δεν μπορούν να υπάρχουν από μόνα τους, ωστόσο όταν προστεθούν σε έναν Ηθοποιό, ο Ηθοποιός θα έχει πρόσβαση και μπορεί να χρησιμοποιήσει τη λειτουργικότητα που παρέχεται από αυτά.

Για παράδειγμα, ένα συστατικό Spot Light θα επιτρέψει στον Ηθοποιό μας να εκπέμπει φως όπως ένα spot light, ένα στοιχείο περιστροφής κίνησης θα κάνει τον Ηθοποιό μας να περιστραφεί ή ένα στοιχείο ήχου θα κάνει τον Ηθοποιό μας να παίζει ήχους.

3.2.6 Πιόνια "Pawns"

Τα πιόνια είναι μια υποκλάση του ηθοποιού(Actor) και χρησιμεύουν ως avatar εντός του παιχνιδιού ή persona, για παράδειγμα οι χαρακτήρες σε ένα

παιχνίδι. Τα πιόνια μπορούν να ελέγχονται από έναν παίκτη ή από το AI του παιχνιδιού, με τη μορφή χαρακτήρων που δεν είναι παίκτες (NPCs).

3.2.7 Χαρακτήρες "Characters"

Ο Χαρακτήρας είναι μια υποκλάση ενός Pawn Actor που προορίζεται να χρησιμοποιηθεί ως χαρακτήρας παίκτη. Η υποκλάση χαρακτήρων περιλαμβάνει μια ρύθμιση σύγκρουσης (collision setup), συνδέσεις εισόδου (input bindings) για μετακίνηση δυαδικών ψηφίων και πρόσθετο κώδικα για κίνηση που ελέγχεται από τον παίχτη.

3.2.8 PlayerController

Η κλάση PlayerController χρησιμοποιείται για να πάρει την είσοδο του παίκτη και να μεταφράσει αυτή σε αλληλεπιδράσεις στο παιχνίδι και κάθε παιχνίδι έχει τουλάχιστον ένα PlayerController σε αυτό. Ένας παίκτης ελέγχου συχνά διαθέτει ένα πιόνι ή χαρακτήρα ως αναπαράσταση του παίκτη σε ένα παιχνίδι.

Ο PlayerController είναι επίσης το κύριο σημείο αλληλεπίδρασης του δικτύου παιχνιδιών για πολλούς παίκτες. Κατά τη διάρκεια του παιχνιδιού για πολλούς παίκτες, ο διακομιστής έχει μία εμφάνιση του PlayerController για κάθε παίκτη του παιχνιδιού, καθώς πρέπει να είναι σε θέση να κάνει κλήσεις λειτουργίας δικτύου σε κάθε παίκτη. Κάθε πελάτης έχει μόνο το PlayerController που αντιστοιχεί στη συσκευή αναπαραγωγής και μπορεί να χρησιμοποιεί μόνο το PlayerController για να επικοινωνήσει με το διακομιστή.

3.2.9 AIController

Ακριβώς όπως ο PlayerController διαθέτει ένα Πιόνι ως αναπαράσταση του παίκτη σε ένα παιχνίδι, ένας AIController διαθέτει ένα Πιόνι για να εκπροσωπεί έναν χαρακτήρα μη παίκτη (NPC) σε ένα παιχνίδι. Από προεπιλογή, τα Πιόνια και οι Χαρακτήρες θα καταλήξουν σε έναν βασικό AIController, εκτός εάν έχουν συγκεκριμένα κατοχυρωθεί από έναν PlayerController ή έχουν εντολή να μην δημιουργήσουν έναν AIController για τον εαυτό τους.

3.2.10 Brushes

Το Brush είναι ένας actor που περιγράφει έναν τρισδιάστατο τόμο που τοποθετείται σε ένα επίπεδο για να καθορίσει τη γεωμετρία του επιπέδου (που αναφέρεται ως BSP) και τους τόμους παιχνιδιού. Συνήθως θα χρησιμοποιούμε Βούρτσες BSP για να προτυποποιήσουμε ή να αποκλείσουμε τα επίπεδα για δοκιμές παιχνιδιού.

Οι τόμοι (Volumes) από την άλλη πλευρά έχουν πολλές χρήσεις ανάλογα με τα effects που συνδέονται με αυτούς, όπως: Blocking Volumes (που είναι αόρατοι και χρησιμοποιούνται για να αποτρέψουν τους Ηθοποιούς να περάσουν από αυτά), Pain Causing Volumes (που προκαλούν βλάβη στην διάρκεια του χρόνου σε οποιονδήποτε Ηθοποιό που τους επικαλύπτει) ή Trigger Volumes (οι οποίοι χρησιμοποιούνται ως ένας τρόπος για να προκαλέσουν γεγονότα όταν ένας Ηθοποιός εισέρχεται ή εξέρχεται από αυτά).

3.2.11 Επίπεδα "Levels"

Το Επίπεδο είναι μια περιοχή παιχνιδιού καθορισμένη από το χρήστη. Τα επίπεδα δημιουργούνται, προβάλλονται και τροποποιούνται κυρίως με τοποθέτηση, μετατροπή και επεξεργασία των ιδιοτήτων των Ηθοποιών που περιέχουν. Στην Unreal, κάθε επίπεδο αποθηκεύεται ως ξεχωριστό αρχείο .umap, γι 'αυτό και μερικές φορές θα τα βλέπουμε ως Χάρτες.

3.2.12 Κόσμος "World"

Ένας κόσμος περιέχει μια λίστα με Επίπεδα που φορτώνονται. Διαχειρίζεται τη ροή των επιπέδων και την αναπαραγωγή (δημιουργία) δυναμικών ηθοποιών.

Η άμεση αλληλεπίδραση με έναν κόσμο δεν είναι απαραίτητη, αλλά βοηθά στην παροχή ενός συγκεκριμένου σημείου αναφοράς στη δομή του παιχνιδιού (δηλ .: αναφέροντας την λέξη Κόσμος σημαίνει ότι δεν αναφερόμαστε απλά σε Επίπεδα, Χάρτες ή παιχνίδι).

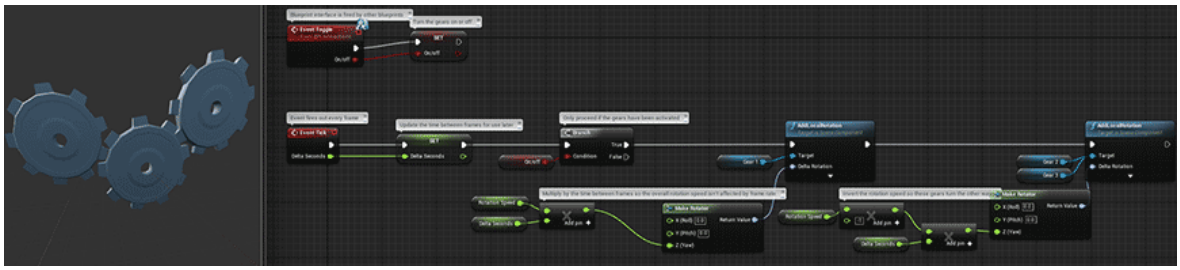
3.2.13 GameModes

Η Κλάση GameMode είναι υπεύθυνη για τον καθορισμό των κανόνων του παιχνιδιού που παίζεται. Οι κανόνες μπορούν να περιλαμβάνουν τον τρόπο με τον οποίο οι παίκτες εντάσσονται στο παιχνίδι, ανεξάρτητα από το εάν ένα παιχνίδι μπορεί να τεθεί σε παύση και μεταβάσεις επιπέδου, καθώς και οποιαδήποτε συμπεριφορά συγκεκριμένου παιχνιδιού, όπως οι συνθήκες νίκης.

Μπορούμε να ορίσουμε το προεπιλεγμένο GameMode στις Ρυθμίσεις του Έργου, αλλά μπορούμε να το παρακάμψουμε ανά Επίπεδο. Ανεξάρτητα από το πώς επιλέγουμε να εφαρμόσουμε το GameMode, υπάρχει πάντα μόνο ένα GameMode παρών ανά επίπεδο. Σε ένα παιχνίδι για πολλούς παίκτες, το GameMode υπάρχει μόνο στο διακομιστή και οι κανόνες αντιγράφονται (αποστέλλονται) σε κάθε συνδεδεμένο πελάτη.

3.2.14 Σύστημα Blueprints Visual Scripting

Το Σύστημα Visual Scripting Blueprints στο Unreal Engine είναι ένα πλήρες σενάριο παιχνιδιού που βασίζεται στην ιδέα της χρήσης μιας διασύνδεσης που βασίζεται σε κόμβους για να δημιουργήσουμε στοιχεία παιχνιδιού μέσα από τον Unreal Editor. Όπως συμβαίνει με πολλές κοινές γλώσσες scripting, χρησιμοποιούνται για τον ορισμό αντικειμενοστρεφών (OO) κλάσεων ή αντικειμένων στην μηχανή γραφικών. Καθώς χρησιμοποιούμε την UE4, συχνά διαπιστώνουμε ότι τα αντικείμενα που ορίζονται με χρήση του Blueprint αναφέρονται συνοπτικά ως απλά "Blueprints". Το σύστημα αυτό είναι εξαιρετικά ευέλικτο και ισχυρό, καθώς παρέχει τη δυνατότητα στους σχεδιαστές να χρησιμοποιούν σχεδόν το πλήρες φάσμα των εννοιών και των εργαλείων που είναι γενικά διαθέσιμα μόνο στους προγραμματιστές. Επιπλέον, η σχεδίαση Blueprint που είναι διαθέσιμη στην εφαρμογή C ++ της Unreal Engine επιτρέπει στους προγραμματιστές να δημιουργήσουν συστήματα βάσης που μπορούν να επεκταθούν από τους σχεδιαστές. Περισσότερα για τα Blueprints στην παράγραφο 3.4.



Εικόνα 9 “Σύστημα Blueprints Visual Scripting”

3.2.15 GameStates

Το GameState περιέχει τις πληροφορίες που θέλουμε να αναπαραχθούν σε κάθε πελάτη σε ένα παιχνίδι, ή απλά είναι "Η κατάσταση του παιχνιδιού" για όλους τους συνδεδεμένους.

Συχνά περιέχει πληροφορίες σχετικά με το σκορ παιχνιδιού, κατά πόσον ένας αγώνας έχει ξεκινήσει ή όχι, πόσα AI θα γεννήσει με βάση τον αριθμό των παικτών στον κόσμο και άλλες πληροφορίες για το παιχνίδι.

Για τα παιχνίδια mutliplayer, υπάρχει ένα στιγμιότυπο του GameState στη μηχανή κάθε παίκτη, με το στιγμιότυπο του διακομιστή να είναι το αυθεντικό (ή αυτό από το οποίο οι πελάτες λαμβάνουν ενημερωμένες πληροφορίες).

3.2.16 PlayerStates

Το PlayerState είναι η κατάσταση ενός συμμετέχοντος στο παιχνίδι, όπως ένας άνθρωπος παίκτης ή ένα bot που προσομοιώνει έναν παίκτη. Το AI που δεν είναι παίκτης που υπάρχει ως μέρος του κόσμου παιχνιδιών δεν θα είχε PlayerState.

Παράδειγμα δεδομένων που θα ήταν κατάλληλα σε ένα PlayerState περιλαμβάνουν το όνομα ή το σκορ του παίκτη, το τρέχον επίπεδο ή την υγεία τους, ή αν φέρουν αυτή τη σημαία σε ένα παιχνίδι Capture the Flag.

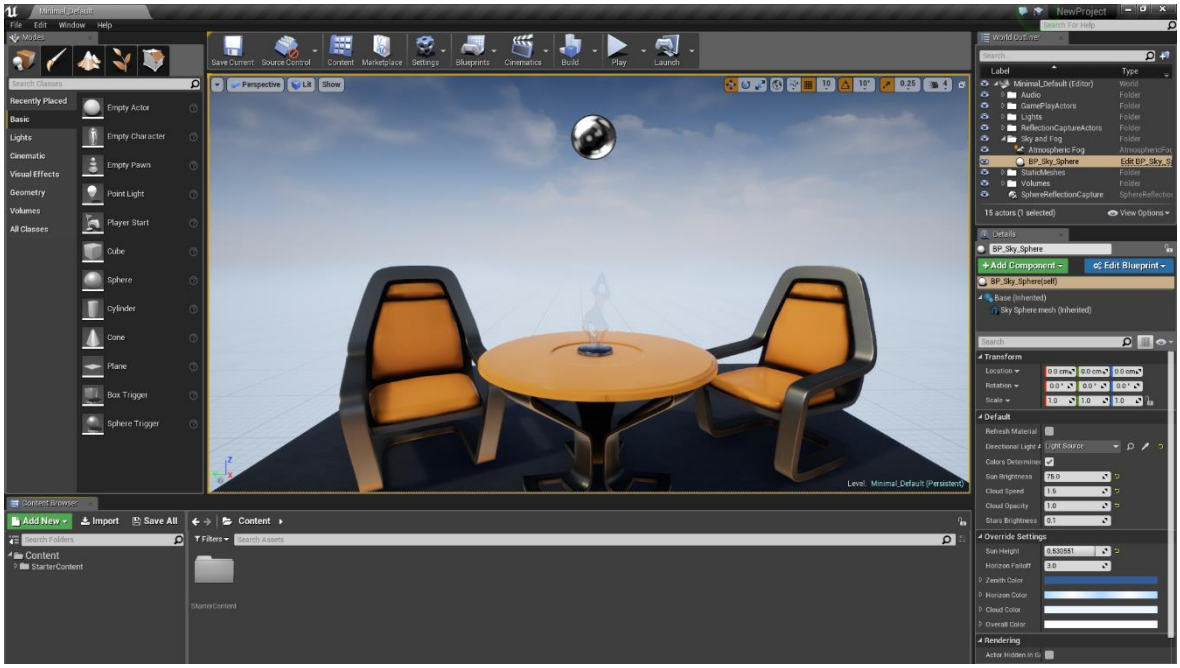
Για τα παιχνίδια για πολλούς παίκτες, οι PlayerStates για όλους τους παίκτες υπάρχουν σε όλα τα μηχανήματα (αντίθετα με τους PlayerControllers) και μπορούν να αναπαράγουν δεδομένα από το διακομιστή στον πελάτη για να κρατούν τα πράγματα συγχρονισμένα.

3.3 Εργαλεία και Συντάκτες “Tools and Editors”

Αυτό το σημείο έχει σκοπό να δώσει μια γενική εικόνα για κάποιους από τους διαφορετικούς τύπους εργαλείων που θα χρησιμοποιούμε μέσα στην Unreal Engine 4. Είτε βρισκόμαστε μέσα στον Level Editor, δημιουργώντας το επίπεδο του παιχνιδιού μας, είτε εργαζόμαστε με τον Blueprint Editor σε συμπεριφορές Ηθοποιών στο επίπεδο μας, μια καλή κατανόηση του τι μπορεί να κάνει ο κάθε επεξεργαστής μπορεί να βελτιώσει τη ροή εργασίας μας και να βοηθήσει να αποφευχθούν τυχόν εμπόδια κατά τη διάρκεια της ανάπτυξης.

3.3.1 Level Editor

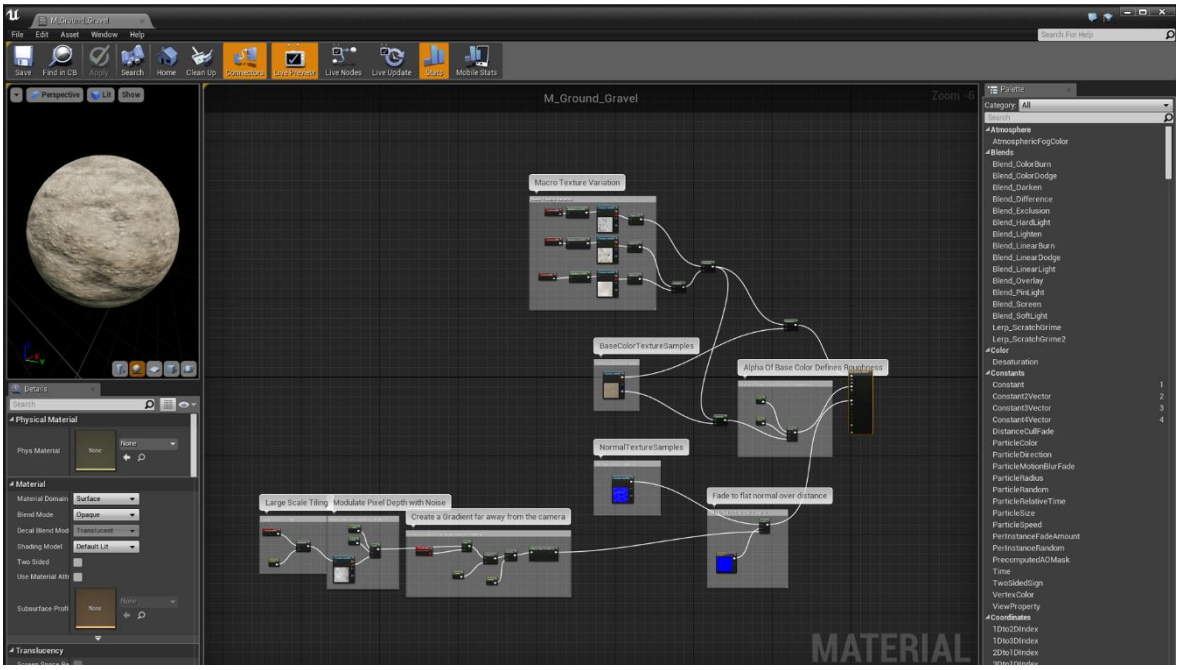
Ο Level Editor είναι ο κύριος Editor που χρησιμοποιείται για την κατασκευή των επιπέδων παιχνιδιού μας. Σε γενικές γραμμές, ορίζουμε τον χώρο παιχνιδιού προσθέτοντας διαφορετικούς τύπους Ηθοποιών και Γεωμετρίας, Blueprints, Συστήματα Cascade Particle ή οτιδήποτε άλλο θέλουμε να προσθέσουμε στο επίπεδο. Από προεπιλογή, όταν δημιουργούμε ή ανοίγουμε ένα έργο, θα ανοίξει ο Level Editor.



Εικόνα 10 "Level Editor"

3.3.2 Material Editor

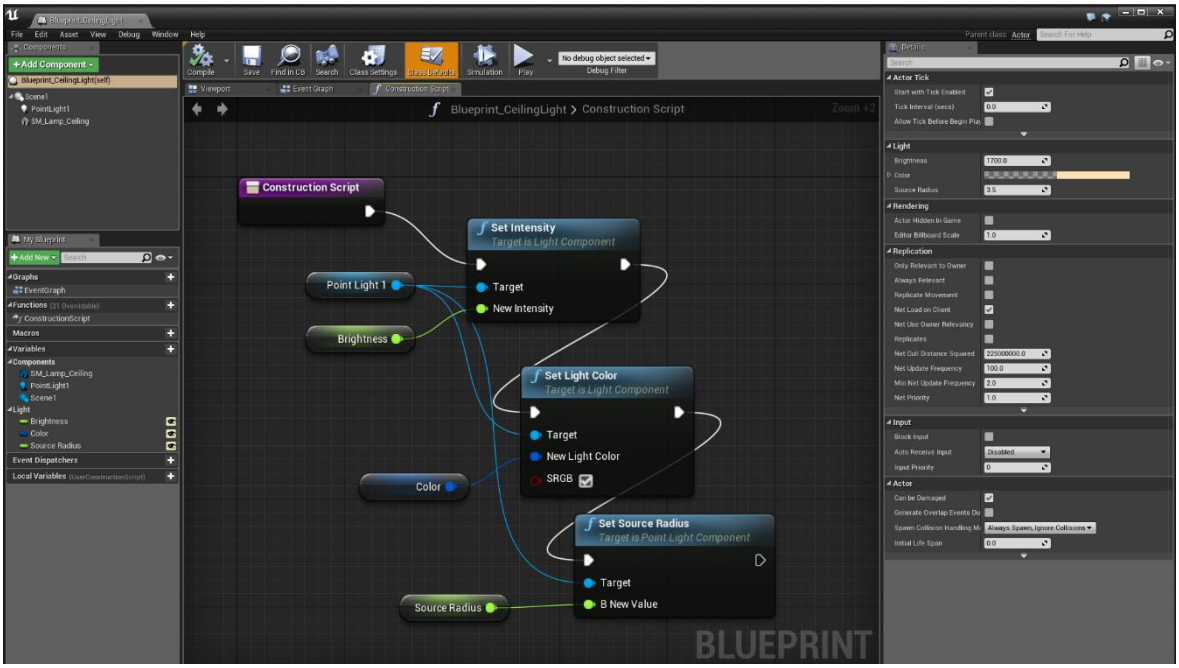
Ο Material Editor είναι το παράθυρο όπου μπορούμε να δημιουργήσουμε (ή να επεξεργαστούμε υπάρχοντα) Υλικά (Materials) τα οποία μπορούν να εφαρμοστούν σε ένα πλέγμα (mesh) για τον έλεγχο της οπτικής του εμφάνισης. Για παράδειγμα, μπορούμε να δημιουργήσουμε ένα υλικό "βρωμιάς" και να το εφαρμόσουμε στα δάπεδα στο επίπεδο ή στο έδαφός για να δημιουργήσουμε μια επιφάνεια που φαίνεται βρώμικη.



Εικόνα 11 "Material Editor"

3.3.3 Blueprint Editor

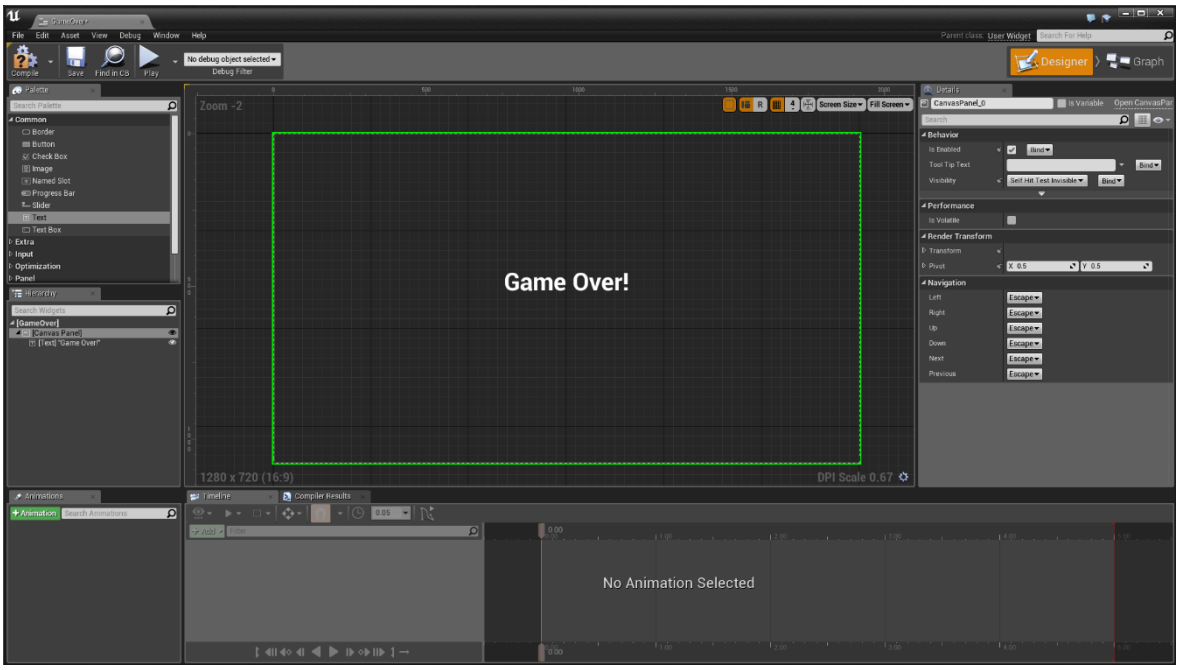
Ο Blueprint Editor είναι το παράθυρο όπου μπορούμε να επεξεργαστούμε και να τροποποιήσουμε τα Blueprints τα οποία είναι ειδικά στοιχεία που μπορούν να χρησιμοποιηθούν για τη δημιουργία νέων τύπων συμβάντων σε επίπεδο ηθοποιών και σεναρίων, χωρίς να χρειάζεται να γράψουμε οποιαδήποτε μορφή κώδικα C ++.



Εικόνα 12 "Blueprint Editor"

3.3.4 UMG UI Editor

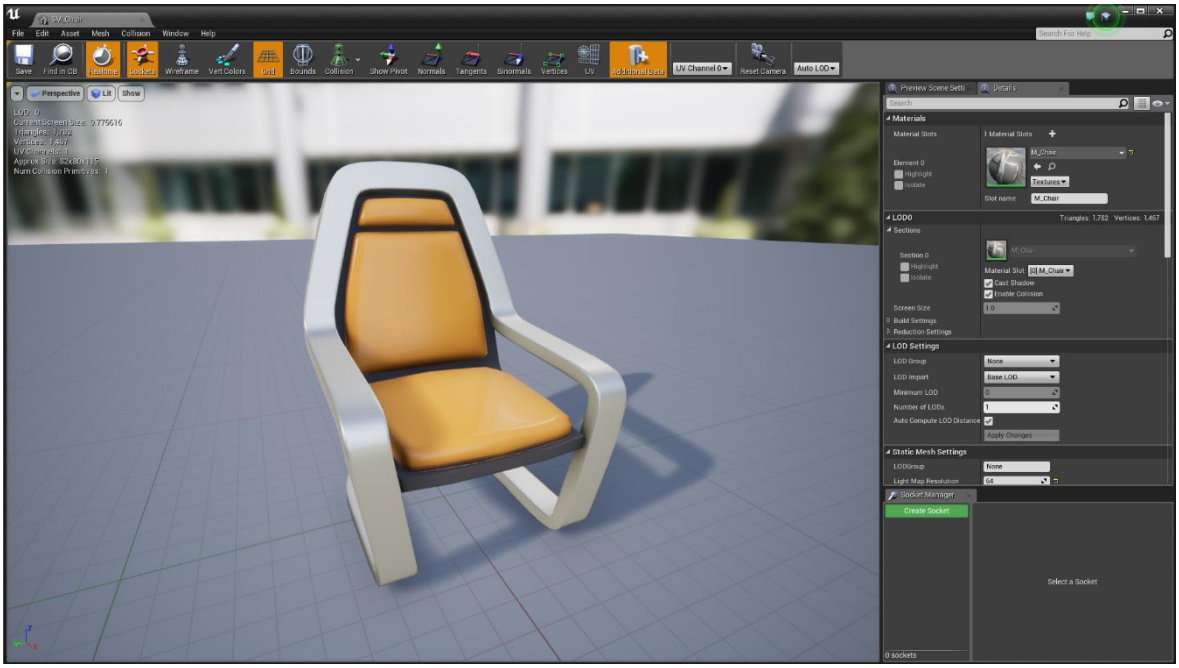
Ο UMG UI Editor είναι ένα εργαλείο δημιουργίας οπτικών UI που μπορεί να χρησιμοποιηθεί για τη δημιουργία στοιχείων UI όπως HUDs, μενού ή άλλα γραφικά που σχετίζονται με τη διεπαφή που θέλουμε να παρουσιάσουμε στους χρήστες μας.



Εικόνα 13 "UMG UI Editor"

3.3.5 Static Mesh Editor

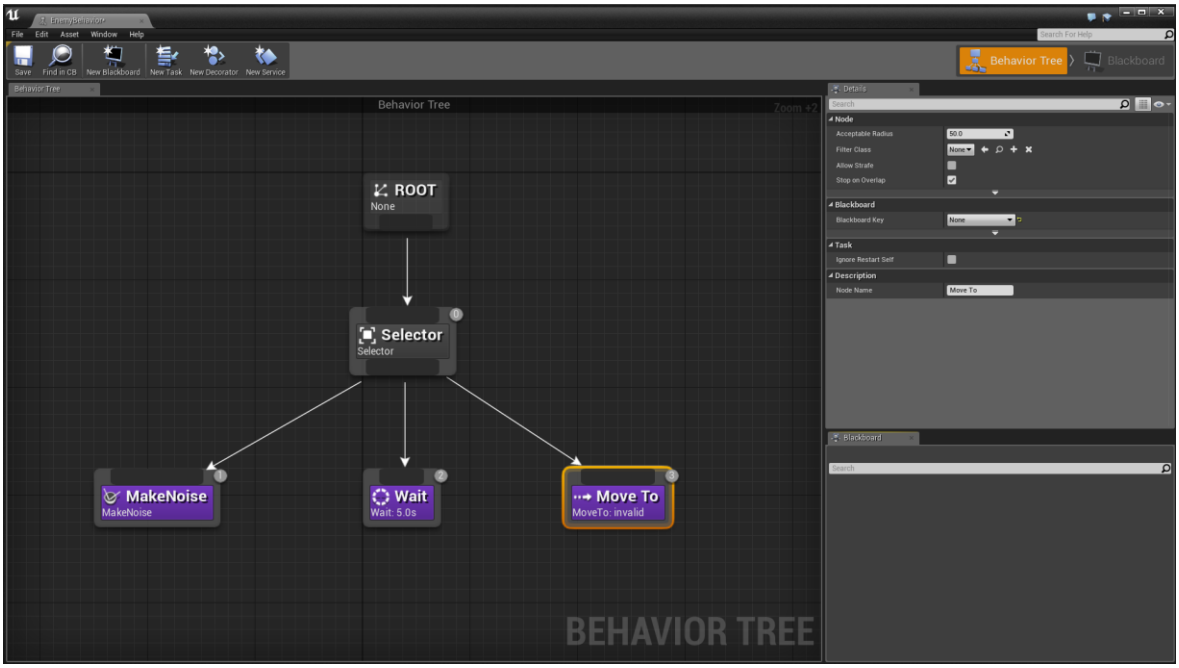
Ο Static Mesh Editor χρησιμοποιείται για την προεπισκόπηση της εμφάνισης, της σύγκρουσης και των UVs καθώς και για τον ορισμό και τον χειρισμό των ιδιοτήτων των Στατικών Πλεγμάτων (Static Meshes). Μέσα από τον Static Mesh Editor μπορούμε επίσης να ρυθμίσουμε τα LODs (τις ρυθμίσεις Επιπέδου λεπτομέρειας) για τα στοιχεία τύπου Static Mesh.



Εικόνα 14 "Static Mesh Editor"

3.3.6 Behavior Tree Editor

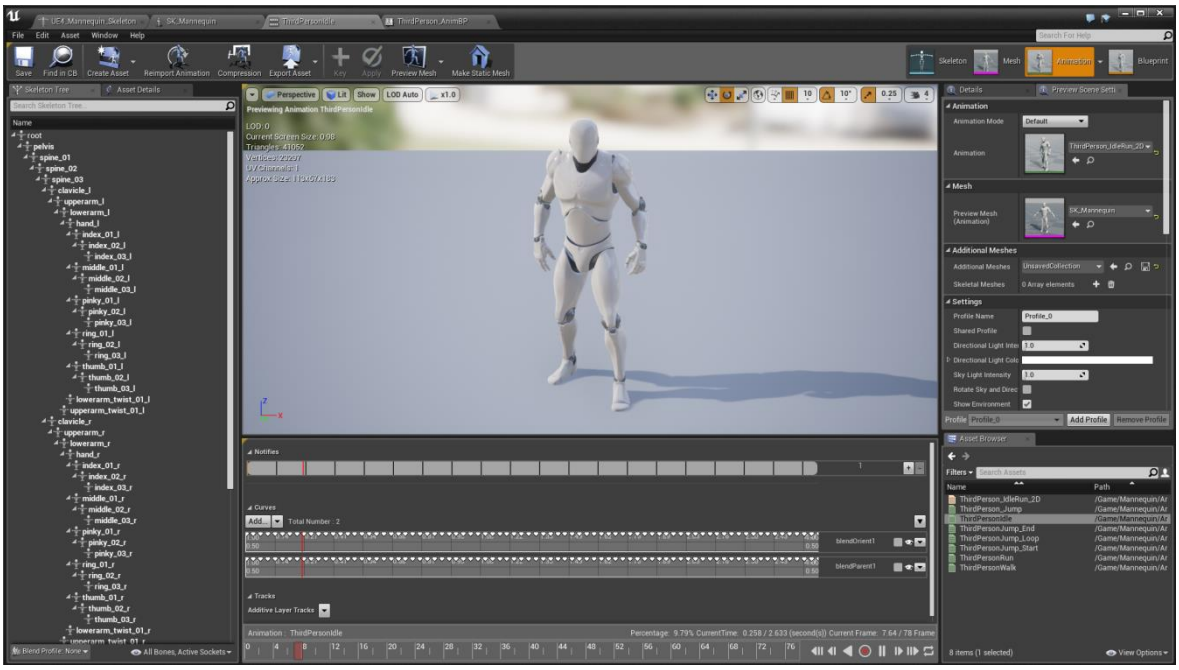
Μέσα από τον επεξεργαστή Tree Behavior μπορούμε να γράψουμε τεχνητή νοημοσύνη μέσω ενός οπτικού συστήματος βασισμένου σε κόμβους (παρόμοιο με Blueprints) για τους Ηθοποιούς στα επίπεδα μας (αυτό θα μπορούσε να είναι οποιοσδήποτε αριθμός διαφορετικών συμπεριφορών για εχθρούς, χαρακτήρες NPC, οχήματα κ.λπ.)



Εικόνα 15 "Behavior Tree Editor"

3.3.7 Persona Editor

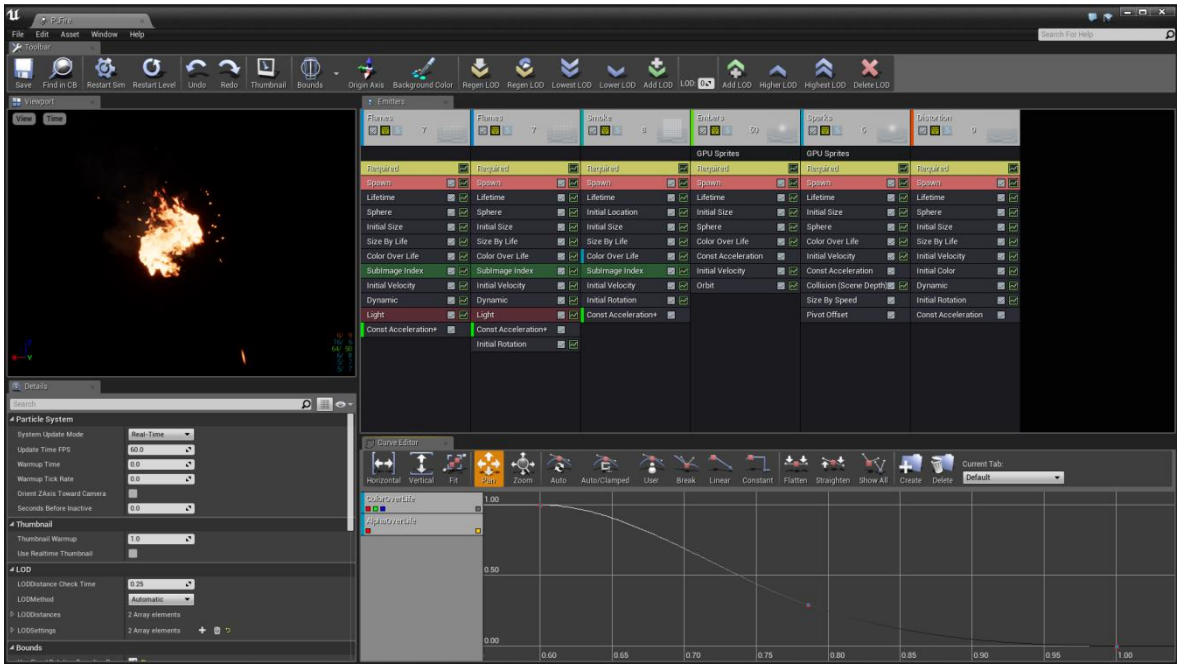
Ο επεξεργαστής Persona είναι το σύνολο εργαλείων επεξεργασίας κινούμενων σχεδίων στο Unreal Engine 4 και χρησιμοποιείται για την επεξεργασία στοιχείων σκελετού, σκελετικών ματιών, Blueprints κινούμενων εικόνων και άλλων. Τα περισσότερα (αν όχι όλα) έργα του animation στο Unreal Engine 4 θα πραγματοποιηθούν σε αυτόν τον επεξεργαστή.



Εικόνα 16 "Persona Editor"

3.3.8 Cascade Editor

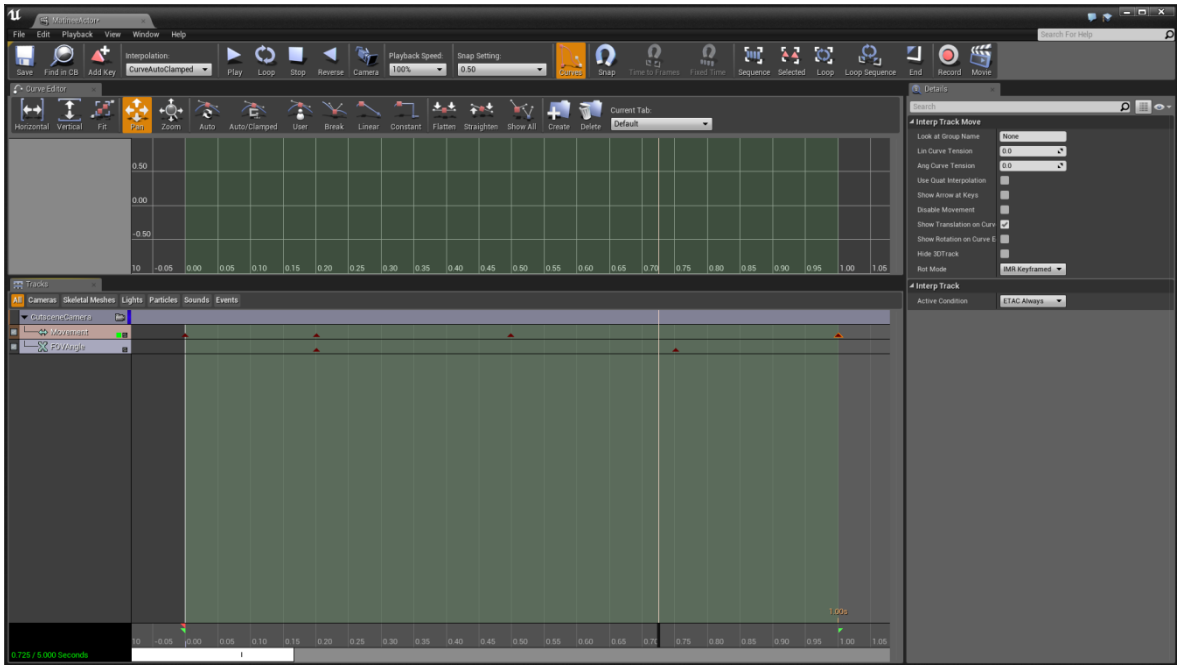
Cascade Particle Systems σε Unreal Engine 4 επεξεργάζονται μέσα στον επεξεργαστή Cascade, ο οποίος είναι ένας πλήρως ενσωματωμένος και διαμορφωμένος επεξεργαστής εφέ σωματιδίων. Το Cascade προσφέρει ανατροφοδότηση σε πραγματικό χρόνο και επεξεργασία αρθρωτών εφέ, επιτρέποντας την γρήγορη και εύκολη δημιουργία ακόμα και των πιο περίπλοκων εφέ.



Εικόνα 17 "Cascade Editor"

3.3.9 Matinee Editor

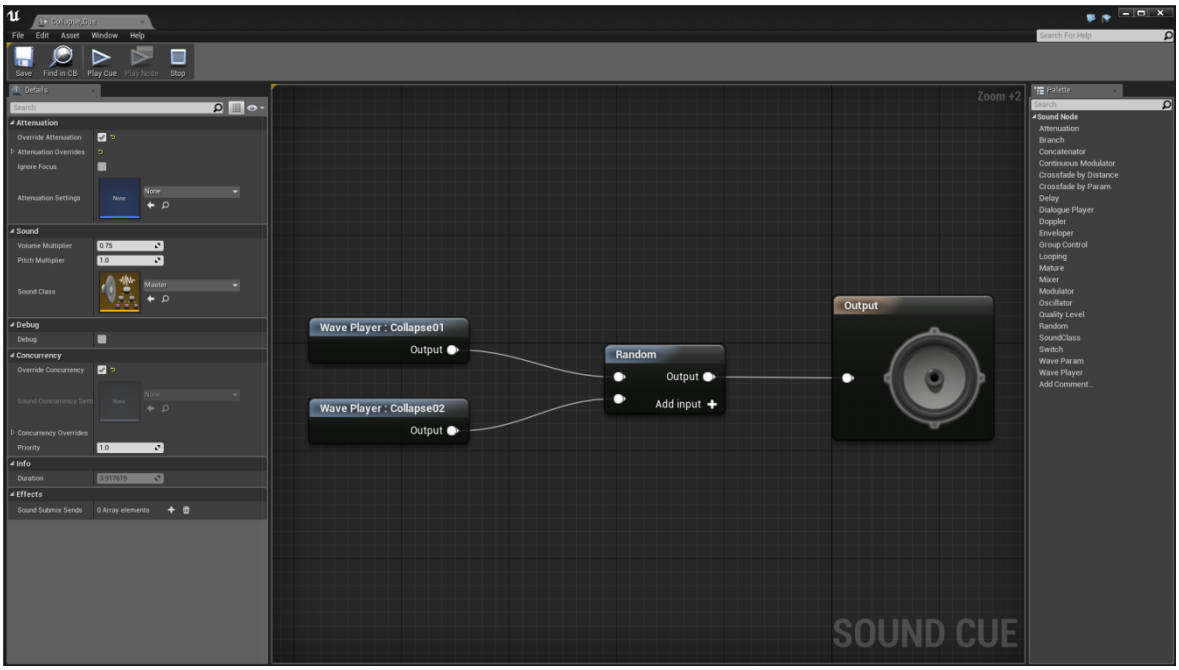
Ο επεξεργαστής Matinee βασίζεται στη χρήση εξειδικευμένων κομματιών κινούμενων εικόνων, στις οποίες μπορούμε να τοποθετήσουμε τα βασικά καρέ για να ορίσουμε τις τιμές ορισμένων ιδιοτήτων των Actor στο επίπεδο μας. Αυτό μας επιτρέπει να δημιουργήσουμε κινηματογραφικές κινήσεις παιχνιδιών, δυναμικά γεγονότα παιχνιδιού ή ακόμα και να ζωντανέψουμε τις ιδιότητες των Ηθοποιών με την πάροδο του χρόνου (όπως η αύξηση της φωτεινότητας μίας πηγής φωτός).



Εικόνα 18 "Matinee Editor"

3.3.10 Sound Cue Editor

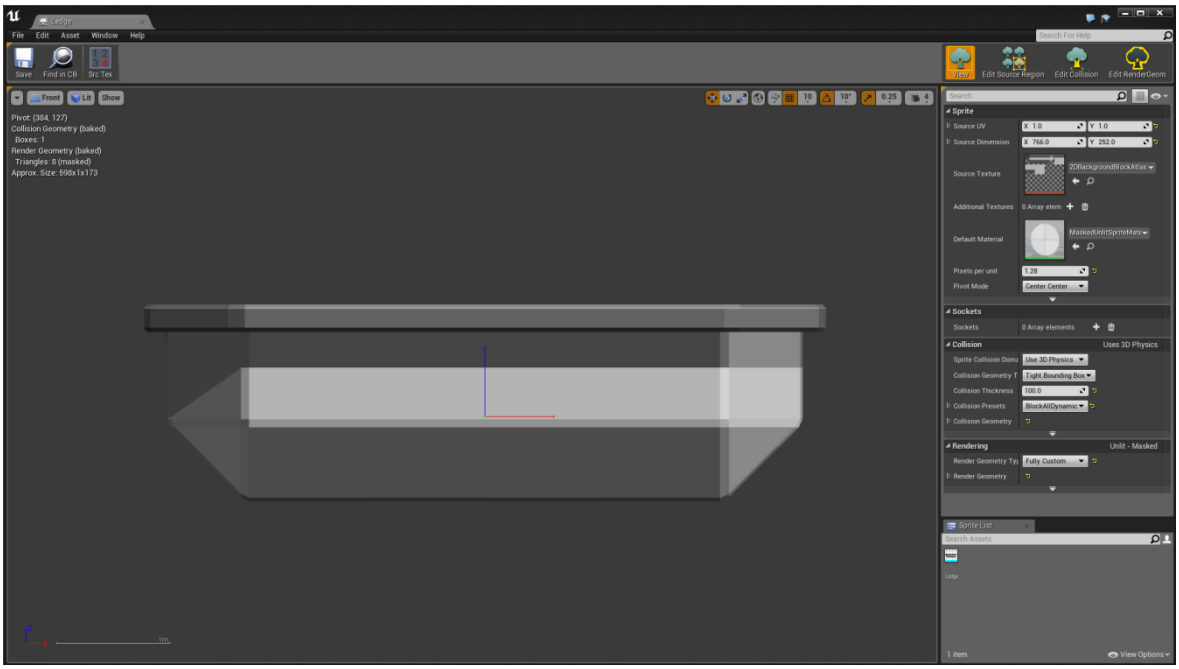
Η συμπεριφορά της αναπαραγωγής ήχου στο Unreal Engine 4 ορίζεται στο Sound Cues, το οποίο μπορεί να επεξεργαστεί χρησιμοποιώντας το πρόγραμμα επεξεργασίας Sound Cue. Μέσα από τον επεξεργαστή Sound Cue Editor, μπορούμε να συνδυάσουμε και να αναμίξουμε πολλά στοιχεία ήχου για να δημιουργήσουμε μια ενιαία μικτή "έξοδο" που είναι αποθηκευμένη ως Sound Cue.



Εικόνα 19 "Sound Cue Editor"

3.3.11 Paper2D Sprite Editor

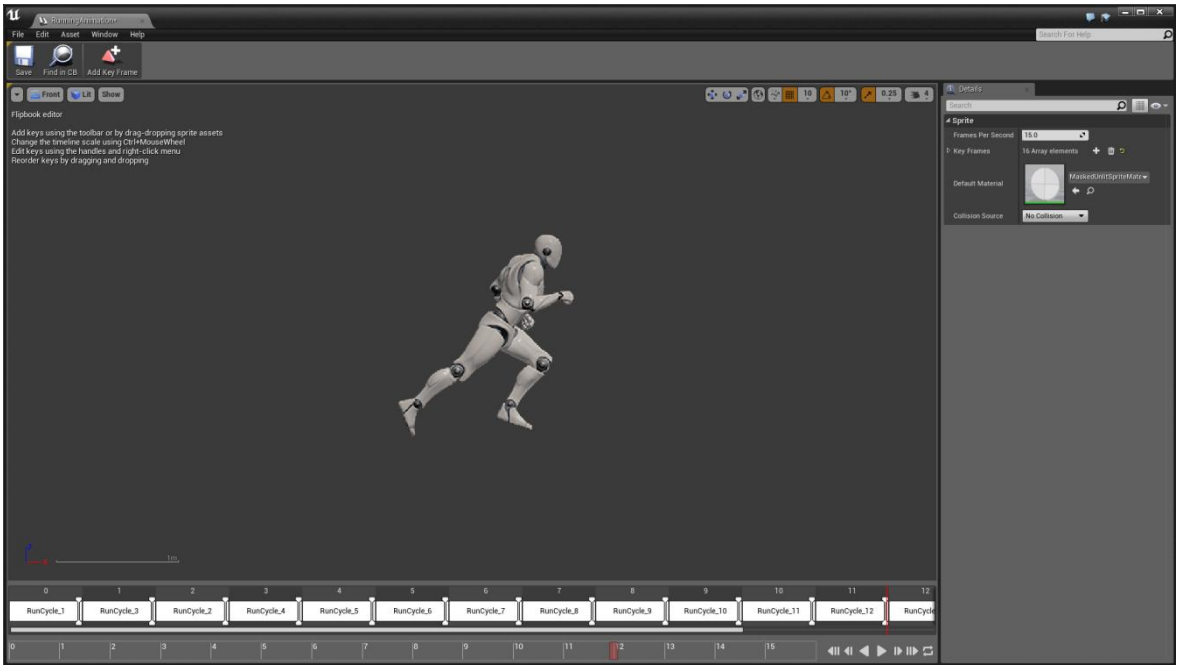
Ο επεξεργαστής Paper2D Sprite επιτρέπει την εγκατάσταση και την επεξεργασία μεμονωμένων 2D Sprites (που είναι ουσιαστικά ένας γρήγορος και εύκολος τρόπος για να σχεδιάσουμε 2D εικόνες στην Unreal Engine 4).



Εικόνα 20 "Paper2D Sprite Editor"

3.3.12 Paper2D Flipbook Editor

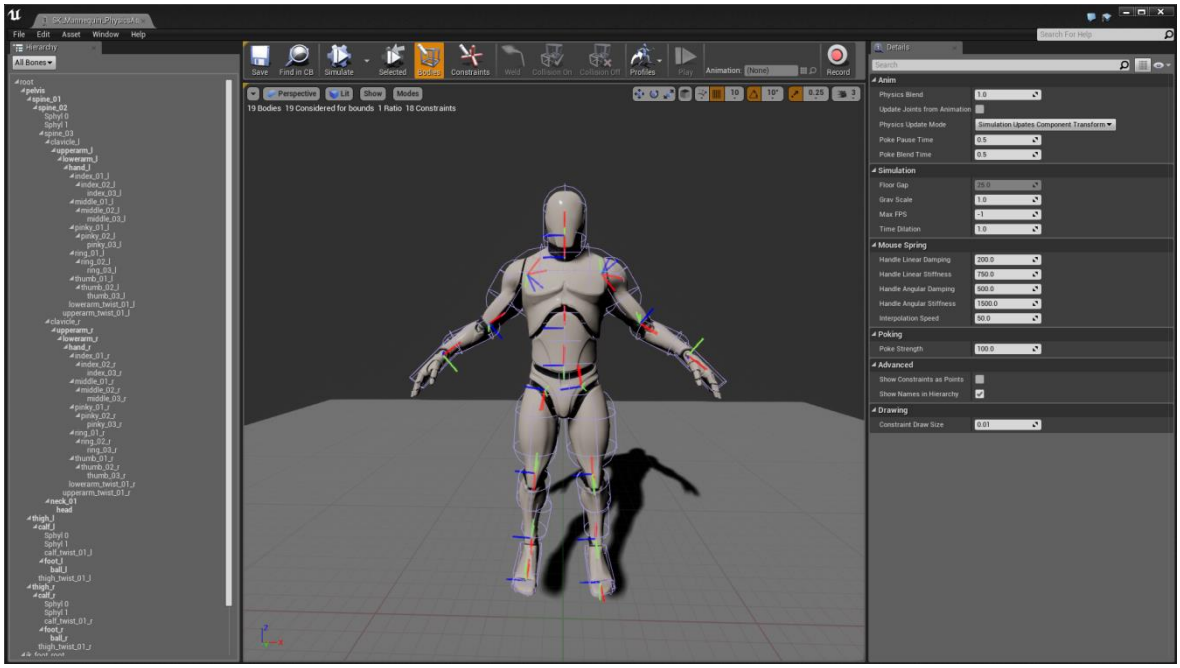
Με το Paper2D Flipbook Editor μπορούμε να δημιουργήσουμε 2D κινούμενα σχέδια που ονομάζονται Flipbooks. Καθορίζοντας μια σειρά από Sprites κατά μήκος ορισμένων πλαισίων κλειδιών μέσα στο Paper2D Flipbook Editor, αυτά τα πλαίσια "μετακινούνται" για να δημιουργήσουν μια κινούμενη εικόνα. Ο καλύτερος τρόπος να σκεφτούμε τα Flipbooks είναι το παλιό στυλ κινουμένων σχεδίων.



Εικόνα 21 "Flipbook Editor"

3.3.13 Physics Asset Tool Editor

Το Εργαλείο Physics Asset Tool Editor (ή PhAT εν συντομία) χρησιμοποιείται για τη δημιουργία στοιχείων φυσικής για χρήση με σκελετικά meshes. Μπορούμε να ξεκινήσουμε από το τίποτα και να χτίσουμε σε μια πλήρη σύνθεση ragdoll ή να χρησιμοποιήσουμε τα εργαλεία αυτοματισμού για να δημιουργήσουμε ένα βασικό σύνολο Φυσικών Ορίων και Φυσικών Περιορισμών.



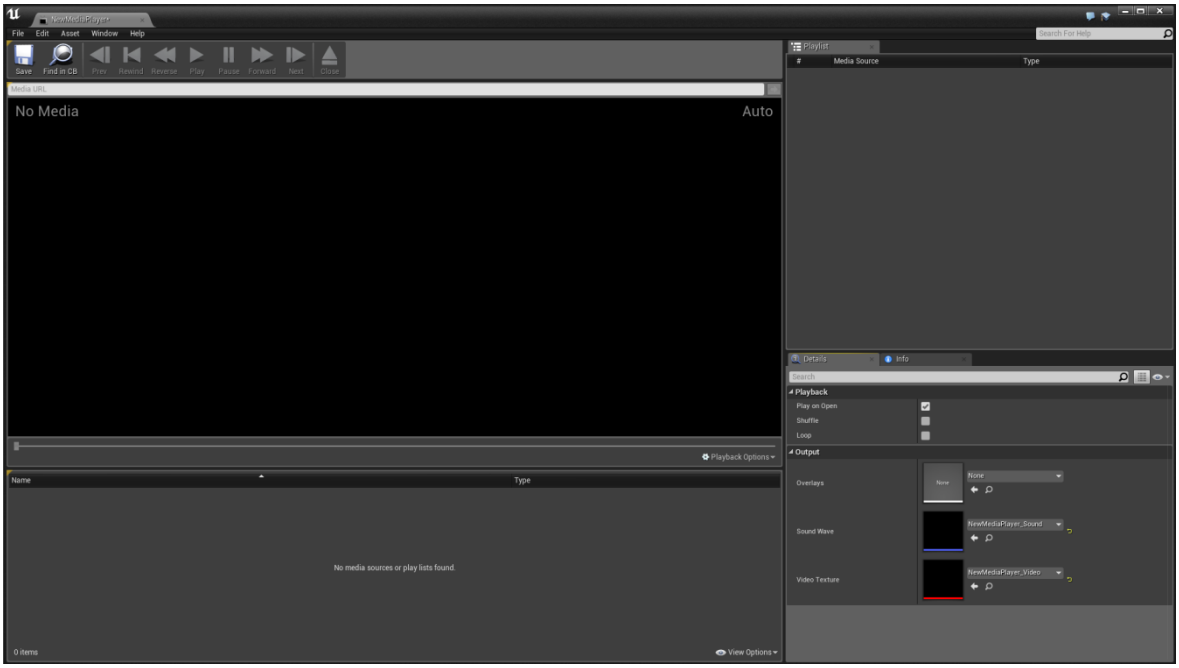
Εικόνα 22 "Physics Asset Tool Editor"

3.3.14 Media Player Editor

Ο Επεξεργαστής Media Player μας επιτρέπει να ορίσουμε αρχεία πολυμέσων ή διευθύνσεις URL που θα χρησιμοποιηθούν ως μέσο πηγής για αναπαραγωγή στην Unreal Engine 4.

Παρόλο που δεν είναι απαραιτήτως ένας "επεξεργαστής" όσον αφορά στο ότι δεν μπορούμε να επεξεργαστούμε αρχεία πολυμέσων, αλλά καθορίζουμε τις ρυθμίσεις για τον τρόπο αναπαραγωγής του πηγαίου μέσου, όπως για παράδειγμα αυτόματη αναπαραγωγή, ρυθμός αναπαραγωγής ή βρόχος.

Μπορούμε επίσης να δούμε πληροφορίες σχετικά με τα μέσα καθώς και να χρησιμοποιήσουμε τυπικά στοιχεία ελέγχου αναπαραγωγής για την προβολή αυτών.



Εικόνα 23 "Media Player Editor"

3.4 Blueprints

Τα Blueprints είναι η προσέγγιση της Unreal Engine 4 στον οπτικό προγραμματισμό. Αυτό σημαίνει ότι οι εργασίες που γενικά προορίζονται για προγραμματισμό scripts δημιουργούνται αντ 'αυτού μέσω ενός γραφήματος των κόμβων και των συνδέσεων, αντί να χρειάζεται να πληκτρολογήσουμε οποιοδήποτε πραγματικό κώδικα. Αυτό επιτρέπει στους καλλιτέχνες και άλλους χρήστες που δεν χρησιμοποιούν κώδικα να δημιουργούν περίπλοκα και εξελιγμένα συστήματα παιχνιδιού που ήταν προηγουμένως διαθέσιμα μόνο στους προγραμματιστές.

Ο σκοπός του παραδείγματος των Blueprints είναι να επιδείξει ορισμένους από τους πολλούς τρόπους με τους οποίους μπορούν να χρησιμοποιηθούν Blueprints στα σχέδια του επιπέδου μας. Το συμπεριλαμβανόμενο επίπεδο περιέχει μια ποικιλία δυνατοτήτων εφέ που βασίζονται σε Blueprint. Ορισμένα από αυτά χρησιμοποιούνται ως στοιχεία σχεδίασης επιπέδου για την προσθήκη οπτικών ή περιβαλλοντικών εφέ, όπως φύλλα ομίχλης και "ακτίνες θεού", ενώ άλλα είναι διαλογικά στοιχεία επιπέδου όπως εξελιγμένα συστήματα κάμερας ασφαλείας. Ο στόχος είναι να δοθεί μια αίσθηση των τεράστιων δυνατοτήτων που μπορούν να προσφέρουν τα Blueprints, παρέχοντας παράλληλα χρήσιμα παραδείγματα για να βοηθήσουν τους χρήστες να μάθουν πώς να δημιουργήσουν οι ίδιοι τα Blueprints.

3.4.1 Πώς λειτουργούν τα Blueprints;

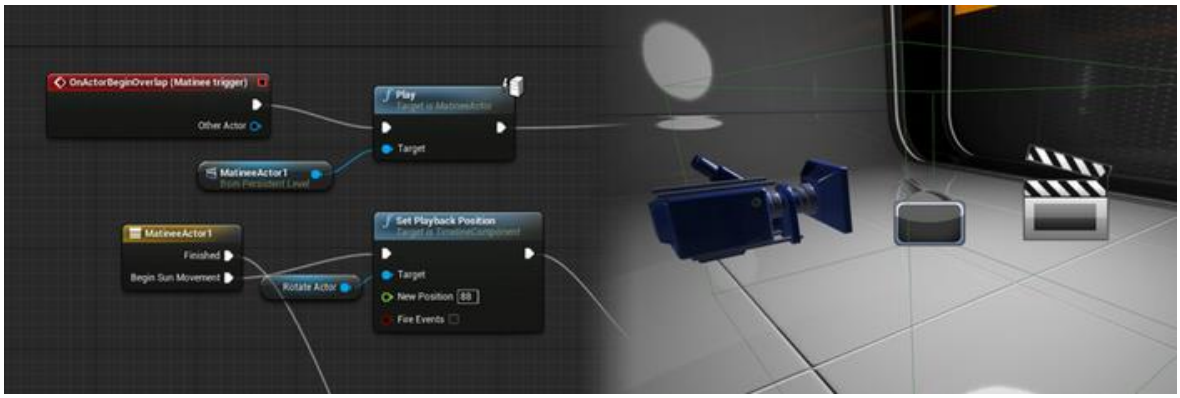
Στη βασική τους μορφή, τα Blueprints είναι visual scripting προσθήκες στο παιχνίδι μας. Συνδέοντας κόμβους, συμβάντα, λειτουργίες και μεταβλητές με καλώδια, είναι δυνατή η δημιουργία σύνθετων στοιχείων παιχνιδιού.

Τα Blueprints λειτουργούν με τη χρήση γραφημάτων κόμβων για διάφορους σκοπούς - κατασκευή αντικειμένων, μεμονωμένες λειτουργίες και γενικά συμβάντα παιχνιδιού - τα οποία είναι συγκεκριμένα για κάθε όψη του Blueprint για την υλοποίηση συμπεριφοράς και άλλων λειτουργιών.

3.4.2 Οι πιο σύνηθες τύποι Blueprint

Οι πιο συνηθισμένοι τύποι Blueprint τους οποίους θα συναντήσουμε συχνότερα είναι τα Level Blueprint και Blueprint Classes.

Αυτοί είναι μόνο δύο από τους τύπους Blueprints, οι οποίοι περιλαμβάνουν επίσης μακροεντολές Blueprint και Blueprint Interfaces.



Εικόνα 24 "Level Blueprint example"

- Το Level Blueprint πληρεί τον ίδιο ρόλο που έκανε το Kismet στο Unreal Engine 3 και έχει τις ίδιες δυνατότητες. Κάθε επίπεδο έχει το δικό του Level Blueprint και αυτό μπορεί να παραπέμπει και να χειρίζεται τους Actors στο επίπεδο, να ελέγχει τα cinematics χρησιμοποιώντας τους Matinee Actors και να διαχειρίζεται πράγματα όπως το level streaming, τα σημεία ελέγχου και άλλα συστήματα που σχετίζονται με το Level. Το Level Blueprint μπορεί επίσης να αλληλεπιδράσει με τις κλάσεις Blueprint (δείτε την επόμενη ενότητα για παραδείγματα αυτών) που βρίσκονται στο επίπεδο, όπως ανάγνωση / ρύθμιση οποιωνδήποτε μεταβλητών ή ενεργοποίηση προσαρμοσμένων συμβάντων που ενδέχεται να περιέχουν.

Εικόνα 25 "Blueprint Class Example"

- Οι Blueprint Classes είναι ιδανικές για τη δημιουργία διαδραστικών στοιχείων όπως πόρτες, διακόπτες, συλλεκτικά αντικείμενα και καταστροφικά τοπία. Στην παραπάνω εικόνα, το κουμπί και το σύνολο των θυρών είναι ξεχωριστά Blueprints που περιέχουν το απαραίτητο σενάριο για να απαντήσουν στα γεγονότα επικαλύψεων παικτών, να τα ζωντανέψουν, να παίξουν ηχητικά εφέ και να αλλάξουν τα υλικά τους (το κουμπί ανάβει όταν πατηθεί, για παράδειγμα). Σε αυτήν την περίπτωση, το πάτημα του κουμπιού ενεργοποιεί ένα συμβάν μέσα στο Blueprint της πόρτας, προκαλώντας το άνοιγμα - αλλά οι πόρτες θα μπορούσαν να ενεργοποιηθούν εξίσου εύκολα με άλλο τύπο Blueprint ή με μια ακολουθία Blueprint Level. Λόγω της αυτοτελούς φύσης των Blueprints, μπορούν να κατασκευαστούν με τέτοιο τρόπο ώστε να μπορούμε να τα αφήσουμε σε ένα επίπεδο και θα λειτουργήσουν απλά, με ελάχιστη απαιτούμενη ρύθμιση. Αυτό σημαίνει επίσης ότι η επεξεργασία ενός Blueprint που χρησιμοποιείται καθ 'όλη τη διάρκεια ενός έργου θα ενημερώνει κάθε εμφάνιση του.

3.4.3 Τι άλλο μπορεί να κάνει ένα Blueprint;

Δημιουργία ενός Χαρακτήρα παιχνιδιού

Τα πιόνια (Pawn) είναι επίσης ένας τύπος κατηγορίας Blueprint και είναι δυνατό να συγκεντρώσουμε όλα τα στοιχεία που χρειάζεστε για ένα χαρακτήρα μέσα σε ένα γράφημα Blueprint. Μπορούμε να χειριστούμε τη συμπεριφορά της κάμερας, να ρυθμίσουμε τα συμβάντα εισόδου για τις οθόνες αφής, controllers και mouse και να δημιουργήσουμε ένα περιουσιακό στοιχείο Blueprint Animation για το χειρισμό των κινούμενων σχεδίων σκελετικών mesh .

Όταν δημιουργούμε ένα νέο Blueprint χαρακτήρων, αυτό εμπεριέχει ένα μεγάλο μέρος της συμπεριφοράς που απαιτείται για τη μετακίνηση, το άλμα, το κολύμπι και την πτώση, και το μόνο που απαιτείται είναι να προσθέσουμε ορισμένα συμβάντα εισόδου σύμφωνα με τον τρόπο που θέλουμε να ελέγχεται ο χαρακτήρα μας.

Δημιουργία ενός HUD

Μπορούμε να χρησιμοποιήσουμε ένα Blueprint για να δημιουργήσουμε ένα HUD, το οποίο είναι παρόμοιο με τις Classes Blueprint, καθώς μπορεί να περιέχει ακολουθίες συμβάντων και μεταβλητών, αλλά έχει ανατεθεί στο περιουσιακό στοιχείο GameMode του project αντί να προστεθεί απευθείας σε ένα επίπεδο.

Μπορούμε να ρυθμίσουμε ένα HUD για να διαβάζουμε τις μεταβλητές από άλλα Blueprints και να τις χρησιμοποιήσουμε για να εμφανίσουμε μια μπάρα υγείας, να ενημερώσουμε μια τιμή σκορ, να εμφανίσουμε αντικειμενικούς δείκτες κλπ. Είναι επίσης δυνατό να χρησιμοποιήσουμε το HUD για να προσθέσουμε hit-

boxes για στοιχεία όπως κουμπιά στα οποία μπορούμε να κάνουμε κλικ ή, στην περίπτωση παιχνιδιών για κινητά, να ανταποκρίνονται στην είσοδο αφής.

ΕΠΙΛΟΓΟΣ

Κλείνοντας αυτό το κεφάλαιο έχουμε πλέον εξοικειωθεί με τις έννοιες και τα εργαλεία που παρέχονται στη πλατφόρμα της Unreal Engine 4. Έχει γίνει εκτενής ανάπτυξη των εννοιών, εργαλείων και δυνατοτήτων της Unreal για την ομαλή κατανόηση της υλοποίησης. Μεγάλη βαρύτητα με αποκλειστική παράγραφο έχει δοθεί στα Blueprints τα οποία αποτελούν το κύριο εργαλείο της UE4. Συνεχίζουμε με το κεφάλαιο της ανάπτυξης.

ΚΕΦΑΛΑΙΟ 4: ΑΝΆΠΤΥΞΗ ΠΑΙΧΝΙΔΙΟΎ

ΕΙΣΑΓΩΓΗ

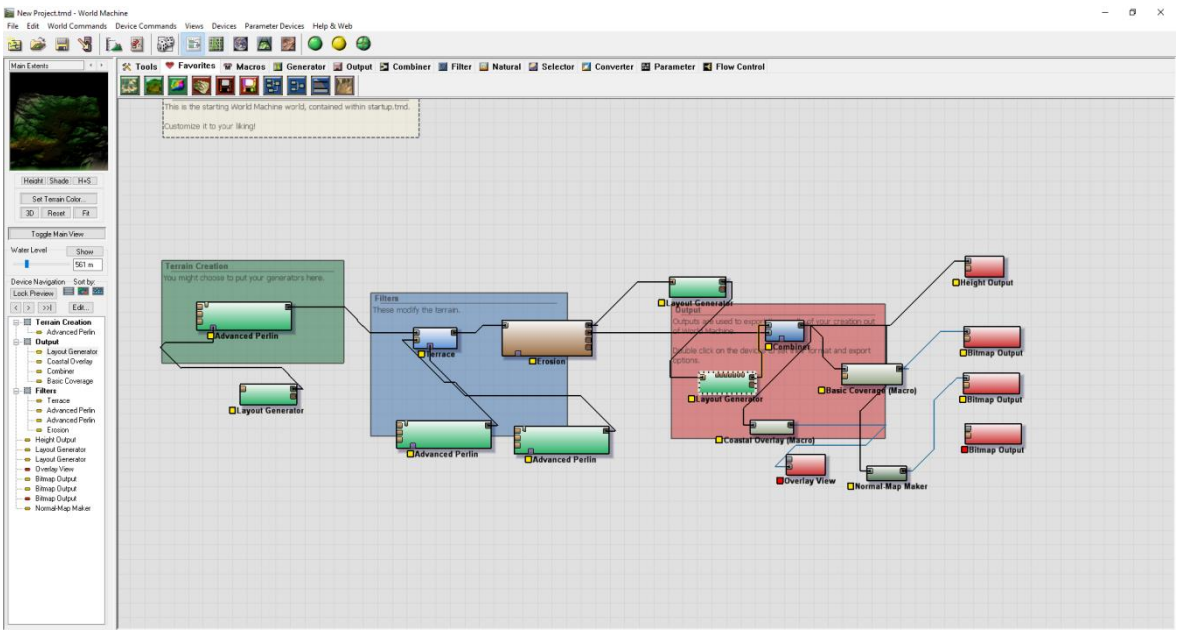
Ξεκινώντας το ταξίδι της δημιουργίας πρέπει να αναφερθεί ότι ακολουθείται μια προσέγγιση από έξω προς τα μέσα. Αυτό σημαίνει ότι περιγράφεται η δημιουργία του κόσμου από το terrain (μορφολογία του εδάφους) και τα foliage(δέντρα, γρασίδι, πέτρες) και καθώς προχωράει προς το εσωτερικού του κόσμου και τις μικρότερες λεπτομέρειες όπως μικροαντικείμενα και λειτουργίες διάδρασης με το περιβάλλον.

4.1 Δημιουργία Project

Το παιχνίδι δημιουργήθηκε με την χρήση του προτύπου First Person με βάση τα Blueprints. Ως ρυθμίσεις του project επιλέχθηκαν Desktop/Console και Maximum Quality ως το υλικό στόχευσης και συμπεριλάβαμε και το Starter Content διότι περιλαμβάνονται πολύ εύχρηστα και σημαντικά για την υλοποίηση μας assets.

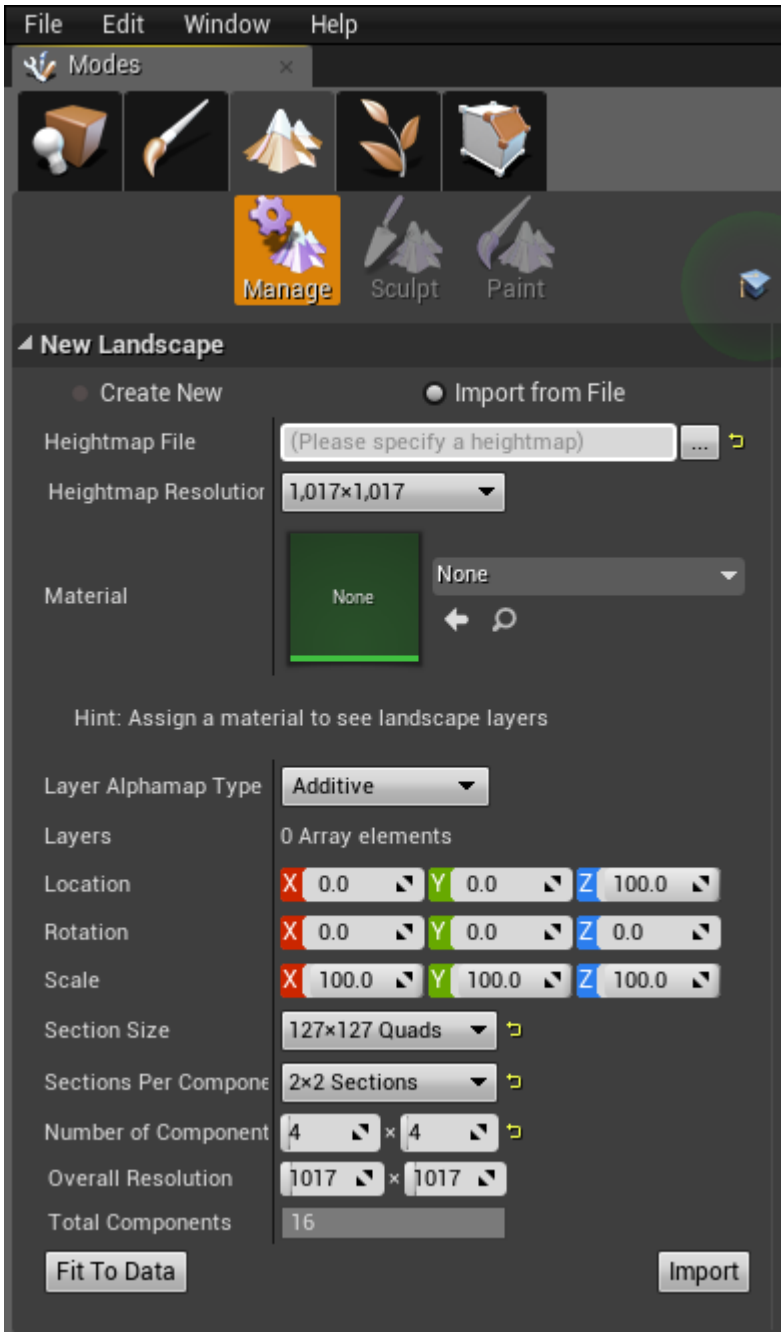
4.2 Έδαφος “Terrain”

Το έδαφος που χρησιμοποιήθηκε δημιουργήθηκε με την χρήση του προγράμματος World Machine. Εδώ δεν μπορεί να γίνει εκτενής αναφορά για το συγκεκριμένο εργαλείο αλλά θα αρκεστούμε στην περιγραφή του και στο τι μας εξυπηρετεί. Μέσα από το παραπάνω εργαλείο έχουμε την δυνατότητα να δημιουργήσουμε, μέσο διαδικαστικής προσέγγισης, ρεαλιστικά εδάφη με μεγάλη ευκολία και σε ελάχιστο χρόνο. Στην εικόνα 26 βλέπουμε το project που χρησιμοποιήθηκε ως βάση για την δημιουργία του εδάφους στο παιχνίδι.



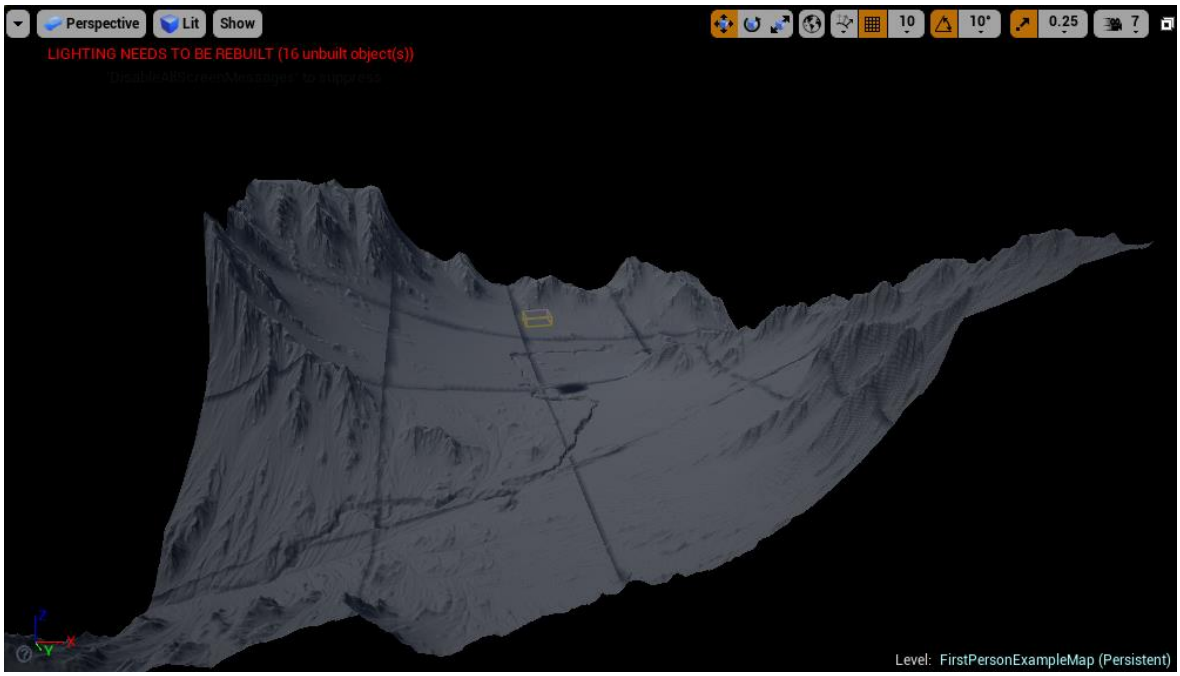
Εικόνα 26 "World Machine"

Έπειτα έγινε η εισαγωγή του terrain στο Project μας όπως φαίνεται και στην εικόνα 27 και πλέον βλέπουμε και το αρχικό terrain στον editor εικόνα 28.



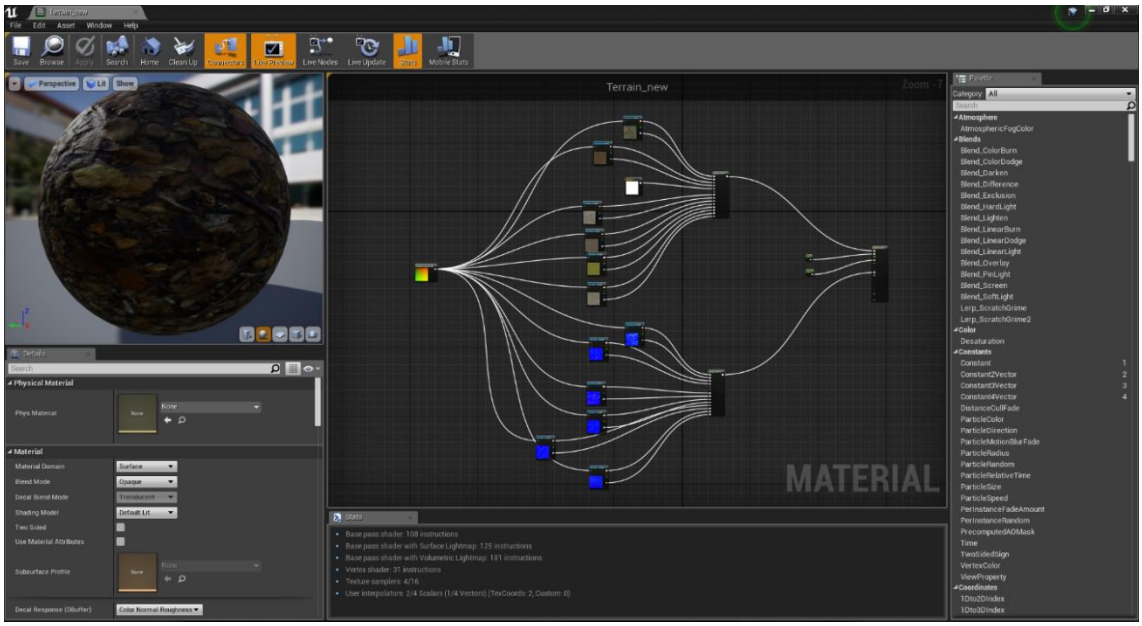
Εικόνα 27 "Landscape Import"

Εικόνα 15 "Import Terrain"

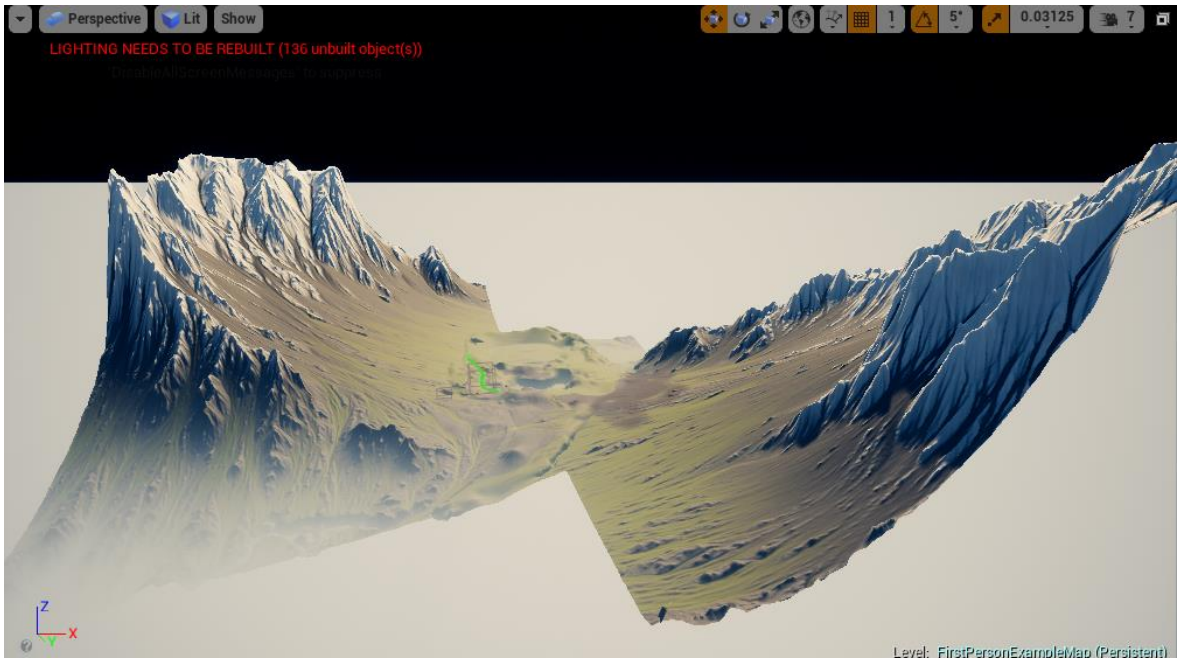


Εικόνα 28 "Terrain μετά την εισαγωγή"

Το επόμενο βήμα ήταν ο «χρωματισμός» του εδάφους και έτσι δημιουργήθηκε το αντίστοιχο material το οποίο περιέχει όλα εκείνα τα texture που χρειαστήκαμε. Για τον «χρωματισμό» χρησιμοποιήθηκαν Bitmaps από το World Machine έτσι η διαδικασία ήταν πιο γρήγορη και το αποτέλεσμα πιο ρεαλιστικό. Στην εικόνα 29 βλέπουμε το material “terrain_new” και στην εικόνα 30 το αποτέλεσμα με την χρήση των bitmap. Έχει γίνει και παρέμβαση στο αρχικό terrain με την χρήση των εργαλείων sculpt και paint για την τελική διαμόρφωση του εδάφους.



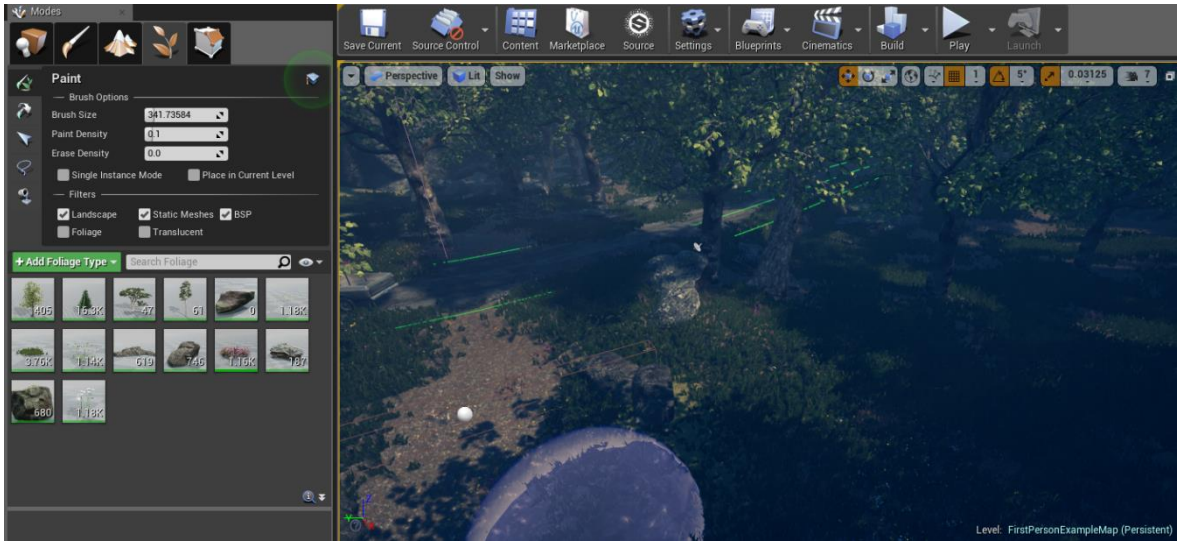
Εικόνα 29 "Material Terrain_new"



Εικόνα 30 "Τελικό Landscape"

4.3 Foliage

Το σύστημα Foliage Instanced Mesh μας επιτρέπει να ζωγραφίσουμε γρήγορα ή να διαγράψουμε σύνολα από Static Meshes. Για παράδειγμα εισάγουμε ένα μοντέλο δέντρου και μπορούμε να «Ζωγραφίσουμε» και να διαχειριζόμαστε πολλά ίδια δέντρα πάνω το terrain με μεγάλη ευκολία και ταχύτητα χωρίς να χρειάζεται να τα προσθέτουμε ένα ένα. Με την χρήση του συγκεκριμένου εργαλείου προστέθηκαν στο terrain δέντρα, γρασίδι, λουλούδια και πέτρες.



Εικόνα 31 "Foliage Demostration"

Τα στοιχεία που χρησιμοποιήσαμε μέσα από το σύστημα Foliage είναι τα παρακάτω:

- HillTree_02
- Low_pine
- ScotsPine_01
- ScotsPineTall_01
- SM_Boulder05a
- SM_Buttercup_Patch_01
- SM_FieldGrass_01
- SM_FieldScabious_01
- SM_GroundRevealRock001
- SM_GroundRevealRock002
- SM_Heather_Mesh_Clumps2
- SM_LargePlainsBoulder002
- SM_River_Rock_01
- SM_Yarrow

4.4 First Person Character Blueprint

Όπως έχει αναφερθεί και στο κεφάλαιο 3.1 κατά την δημιουργία του project χρησιμοποιήθηκε το πρότυπο First Person. Αυτό το πρότυπο περιέχει τις βασικές λειτουργίες που χρειάζεται ένα παιχνίδι τύπου «Πρώτου Προσώπου». Ένα πολύ σημαντικό κομμάτι του είναι το First Person Character Blueprint. Έχουν γίνει οι απαραίτητες παρεμβάσεις στο συγκεκριμένο αρχείο Blueprint ώστε να επιτευχθούν οι στόχοι του παιχνιδιού. Το συγκεκριμένο Blueprint εμπεριέχει όλες τις λειτουργίες που έχουν να κάνουν με την κίνηση του χαρακτήρα μας, την εισαγωγή δηλαδή της κίνησης από το πληκτρολόγιο, ποντίκι κλπ. όπως και λειτουργίες που προσθέσαμε εμείς Line Trace, Pause Widget, Notes Widget και Intro Sequence όπως θα δούμε και θα αναπτύξουμε παρακάτω. Αφαιρέθηκαν στοιχεία όπως η εκτόξευση μπάλας με το κλικ και τα χέρια με το όπλο βλέπε εικόνα 32,33. Αλλαγές έγιναν και το FirstPersonHUD ώστε να γίνει μια διάφανη τελεία ο κεντρικός, αρχικά κόκκινος, στόχος. Με αυτήν τη τελεία θα μπορεί ο παίχτης να στοχεύει τα αντικείμενα με τα οποία επιθυμεί να αλληλοεπιδράσει.

Εικόνα 32 "First Person Character 1"

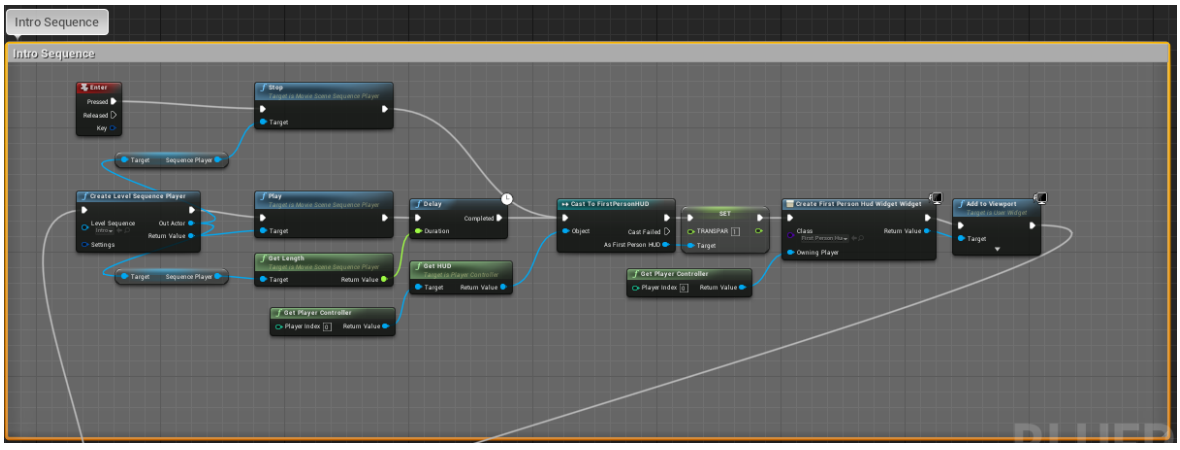
Εικόνα 33 "First Person Character 2”

4.4.1 First Person Character Custom Variables.

- PlayerObjective τύπου Text στην οποία κρατάμε και εμφανίζουμε τον επόμενο στόχο του παίχτη.
- Pause Menu Reference τύπου User WIdget στην οποία κρατάμε το Pause Menu WIdget μετά την δημιουργία του.
- has_key τύπου Boolean η οποία καθορίζει αν ο παίχτης διαθέτει το κλειδί για την πόρτα του σπιτιού.
- Notes_Text τύπου Text η οποία περιέχει το κείμενο από το σημειωματάριο.
- Story_sequence τύπου Integer την οποία την χρησιμοποιούμε για να καθορίσουμε την σειρά που μπορεί ο παίχτης να λαμβάνει τα objectives.

4.4.2 Intro Sequence

Κατά την δημιουργία του First Person Character και του Event BeginPlay δημιουργούμε έναν Level Sequence Player με την χρήση του κόμβου Create Level Sequence Player και στην είσοδο του δίνουμε το αντικείμενο Level Sequence “Intro”. Έπειτα η επιστρεφόμενη τιμή συνδέεται με τον κόμβο play ώστε να ξεκινήσει η αναπαραγωγή της εισαγωγής η οποία θα εκτελείται μέχρι να διακοπεί. Στους κόμβους παρατηρούμε ότι μετά τον κόμβο Play υπάρχει ο κόμβος Delay ο οποίος λαμβάνει σαν παράμετρο την διάρκεια του Level Sequence αντικειμένου Intro και η χρήση του είναι να καθυστερήσει την εμφάνιση του FirstPersonHUD μέχρι να ολοκληρωθεί η αναπαραγωγή. Παράλληλα υπάρχει και ο κόμβος enter που κατά το event pressed ενεργοποιείται ο κόμβος Stop έτσι ώστε να παρακάμψουμε την εισαγωγή. Αφού ολοκληρωθεί η αναπαραγωγή τότε με την χρήση του κόμβου Cast to First Person HUD έχουμε πρόσβαση και αλλάζουμε την custom μεταβλητή που έχουμε δημιουργήσει TRANSPAR τύπου Float από μηδέν σε 1, δημιουργούμε το First Person Hud Widget και με την χρήση του κόμβου Add to Viewpoint το εμφανίζουμε. Στην εικόνα 34 βλέπουμε το τελικό αποτέλεσμα.



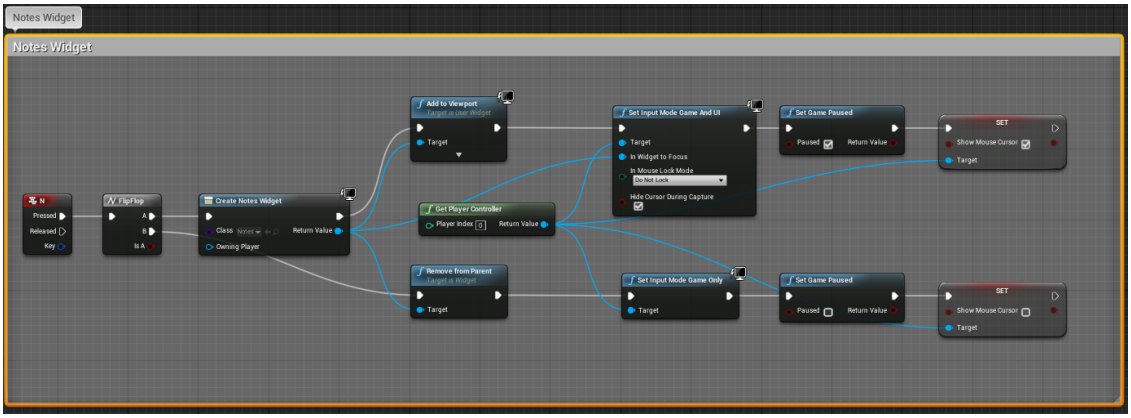
Εικόνα 34 "Intro Sequence Playback”

4.4.3 Note Widget

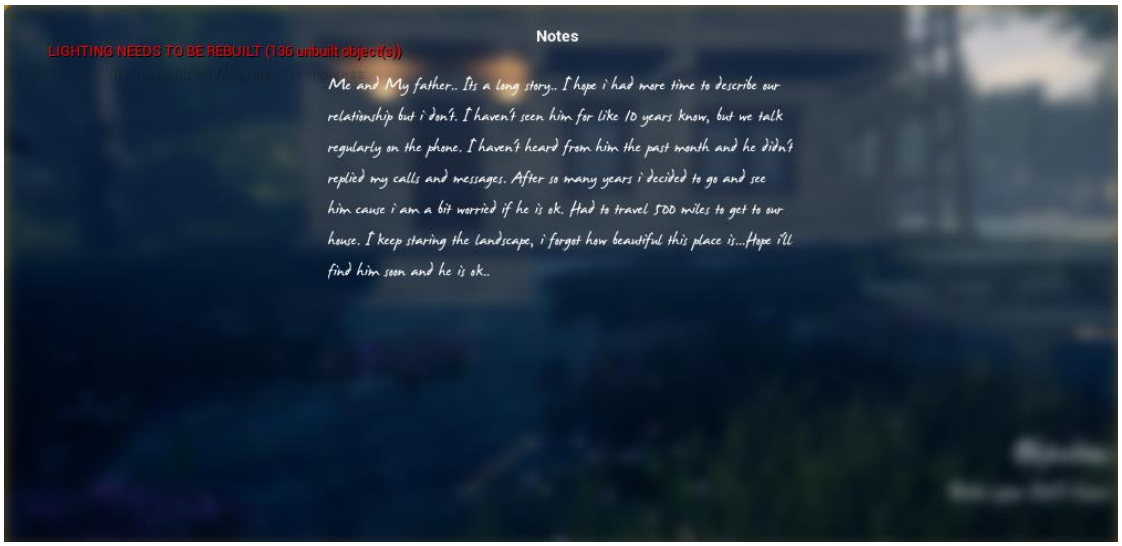
Ένα χαρακτηριστικό του παιχνιδιού είναι ότι καταγράφεται στο «σημειωματάριο» του παίχτη ότι συμβαίνει στο παιχνίδι ώστε να μπορεί να ανατρέξει στο παρελθόν βοηθώντας τον έτσι να φτάσει πιο γρήγορα στην λύση

του μυστηρίου. Έτσι με την χρήση κόμβων με την χρήση του πλήκτρου «N» μπορεί ο παίχτης να ενεργοποιήσει και να απενεργοποιήσει το Σημειωματάριο. Αρχικά έχουμε τον κόμβο τύπου event N και κατά το event pressed ενεργοποιείται ο κόμβος FlipFlop ο οποίος έχει δύο ενέργειες A και B τις οποίες εκτελεί εναλλάξ με κάθε πληκτρολόγηση του πλήκτρου «N».

- Η A επιλογή μας οδηγεί στον κόμβο Create Notes Widget στο οποίο δημιουργούμε το Widget Notes και έπειτα με τον κόμβο Add to Viewport το εμφανίζουμε. Έπειτα αλλάζουμε τον τρόπο εισαγωγής σε Game and UI και αυτό ώστε να μπορούμε να κινηθούμε με τα πλήκτρα και το ποντίκι μέσα στο Note Widget. Με τον επόμενο κόμβο κάνουμε Pause το Game και τέλος εμφανίζουμε τον κέρσορα με τον κόμβο SET.
- Η B επιλογή μας οδηγεί στον κόμβο Remove from Parent και λαμβάνουμε ως τιμή εισόδου την επιστρεφόμενη τιμή από τον κόμβο Create Notes Widget. Έπειτα αλλάζουμε ξανά τον τρόπο εισόδου αυτήν την φορά μόνο σε Game, αλλάζουμε την τιμή του Game Paused σε false και αποκρύπτουμε τον κέρσορα.



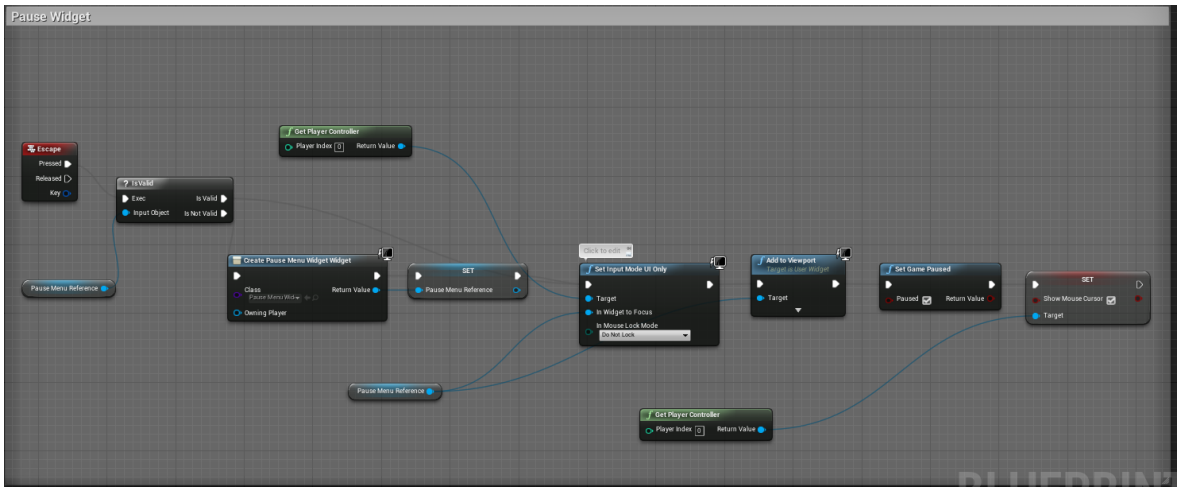
Εικόνα 35 "Notes Widget Activation/Deactivation"



Εικόνα 36 "Notes Widget In-Game"

4.4.4 Pause Widget

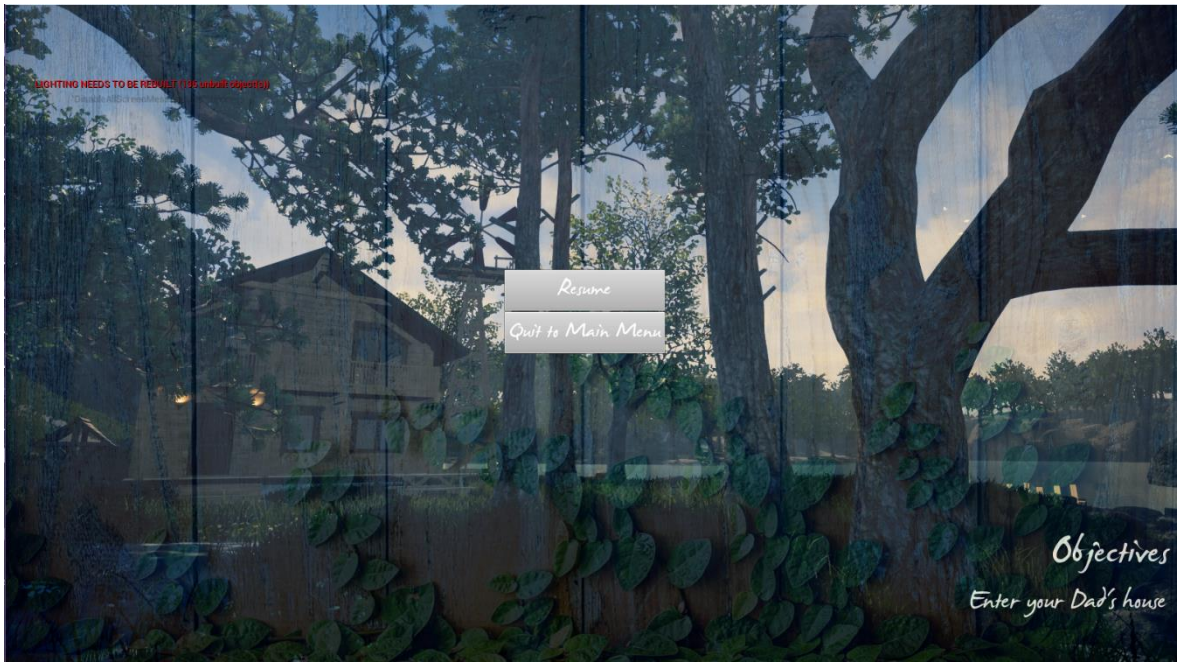
Ο παίχτης έχει την δυνατότητα να κάνει παύση κατά την διάρκεια που παίζει με την χρήση του πλήκτρου Escape. Έτσι ξεκινώντας από το event on escape key pressed ελέγχουμε αν η μεταβλητή τύπου User Widget “Pause Menu Reference” που έχουμε δημιουργήσει είναι valid. Εάν δεν είναι τότε με τον κόμβο “Create Pause Menu Widget Widget” δημιουργούμε το Pause Menu Widget και το θέτουμε στην μεταβλητή “Pause Menu Reference”. Έπειτα ή στην περίπτωση που είναι valid συνεχίζουμε με την χρήση του κόμβου “Set Input Mode UI Only” με στόχο τον Player Controller και Widget to Focus το περιεχόμενο της μεταβλητής “Pause Menu Reference” και το εμφανίζουμε με την χρήση του κόμβου Add to Viewport. Τελειώνοντας θέτουμε το παιχνίδι σε παύση με την χρήση του κόμβου Set Game Paused θέτοντας σε True την boolean μεταβλητή Paused και εμφανίζουμε τον κέρσορα.



Εικόνα 37 "Pause Widget Activation"

Pause Menu Widget

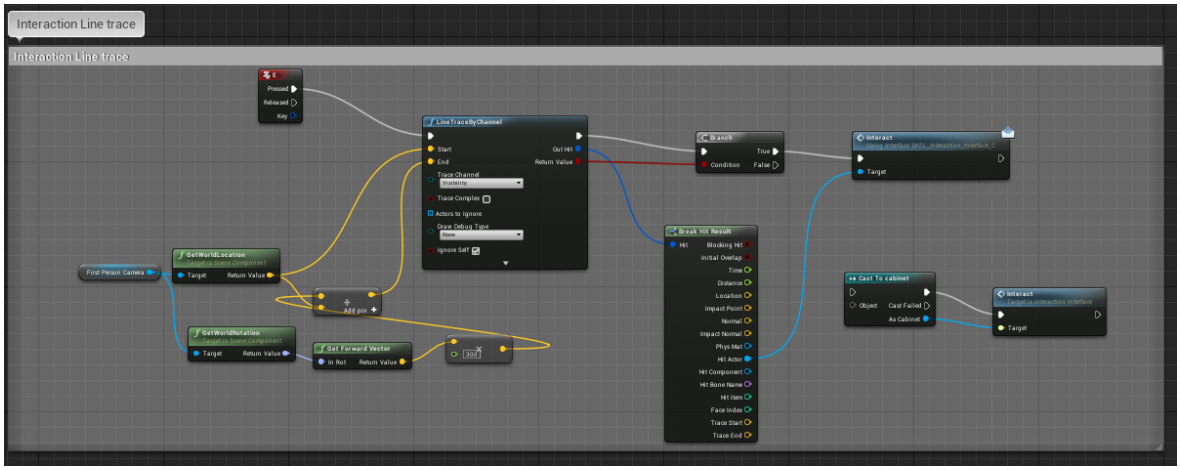
Ο σχεδιασμός του συγκεκριμένου widget είναι πολύ απλός. Διαθέτει ένα background, δύο buttons, Resume και Quit to Main Menu. Κάνοντας κλικ στο Resume αποκρύπτουμε τον κέρσορα, θέτουμε την μέθοδο εισόδου σε Game Mode Only, αφαιρούμε το widget με την χρήση του κόμβου Remove from Parent και δίνουμε συνέχεια στο παιχνίδι κάνοντας false την μεταβλητή Paused του κόμβου Set Game Paused. Κάνοντας κλικ στο Quit to Main Menu αρχικά εμφανίζουμε τον κέρσορα και καλούμε την μέθοδο Remove HUD που έχουμε δημιουργήσει. Έπειτα ανοίγουμε το αρχικό μενού το οποίο βρίσκεται σε διαφορετικό Level. Αυτό το καταφέρνουμε με την χρήση του κόμβου Open Level και δίνοντας σαν εισαγωγή Level Name το “MainMenu”. Για το αρχικό μενού θα μιλήσουμε σε επόμενη παράγραφο.



Εικόνα 38 "Pause Widget In-Game"

4.4.5 Interaction Line trace

Μπορεί να υπάρχουν περιπτώσεις στο παιχνίδι όπου θέλουμε να προσδιορίσουμε αν ο χαρακτήρας του παίκτη κοιτάζει κάτι και αν ναι να δημιουργείται κάποιο συμβάν. Αυτό μπορούμε να το υλοποιήσουμε χρησιμοποιώντας Raycasts (ή Tracing) για να "πυροβολήσουμε" μια αόρατη ακτίνα που θα ανιχνεύσει τη γεωμετρία μεταξύ δύο σημείων και αν χτυπήσει γεωμετρία, επιστρέφει αυτό που χτυπήθηκε, ώστε να μπορούμε να κάνουμε κάτι με αυτό. Το πιο σημαντικό χαρακτηριστικό του παιχνιδιού είναι η παραπάνω δυνατότητα του παίχτη να αλληλοεπιδρά με τα αντικείμενα τα οποία στοχεύει με το στόχο πατώντας παράλληλα το πλήκτρο “E”. Ξεκινώντας με το event on “E” key pressed καλούμε την μέθοδο LineTraceByChannel δίνοντας παραμέτρους Start την θέση της κάμερας “First Person Camera” με την χρήση του GetWorldLocation και End μέχρι το σημείο που θέλουμε να λειτουργεί το linetrace με απόσταση 300 από την θέση της κάμερας ανεξαρτήτου περιστροφής. Με την χρήση του κόμβου Break Hit Result αναλύουμε το χτύπημα της αόρατης ακτίνας και λαμβάνουμε τον Hit Actor αυτό φυσικά μόνο εάν έχει γίνει χτύπημα δηλαδή η return value από την LineTraceByChannel είναι true. Σε αυτό το σημείο πρέπει να αναφέρουμε ότι έχουμε δημιουργήσει ένα interface το “Interaction_Interface” το οποίο περιέχει την μέθοδο Interact. Στο τέλος του Line Trace καλείται αυτή η μέθοδος με στόχο τον Hit Actor που “χτύπησε” το Line Trace. Οπότε κάθε αντικείμενο που έχει γίνει impliment το συγκεκριμένο interface και διαθέτει υλοποιημένο το Interact event, στοχεύοντας το με τον στόχο και πατώντας το πλήκτρο “E”, θα εκτελείται ότι ακολουθεί μετά την ενεργοποίηση του event. Αυτή η υλοποίηση μας έδωσε την δυνατότητα να αλληλοεπιδρούμε με όποιον Actor επιθυμούμε, εισάγοντας τις ενέργειες που θέλουμε να λαμβάνουν χώρα εντός του αντίστοιχου Blueprint.



Εικόνα 39 "Line Trace Workflow"

4.5 Main Menu Level

Το συγκεκριμένο Level το χρησιμοποιούμε μόνο για το κεντρικό μενού του παιχνιδιού. Σε επίπεδο ρυθμίσεων κόσμου χρησιμοποιούμε το “FirstPersonGameModeMenu” αλλάζοντας συγκριτικά με το “FirstPersonGameMode” από το “FirstPersonExampleMap” level την HUD Class σε “MainMenu” και την Default Pawn Class σε “None”.

4.5.1 MainMenu HUD

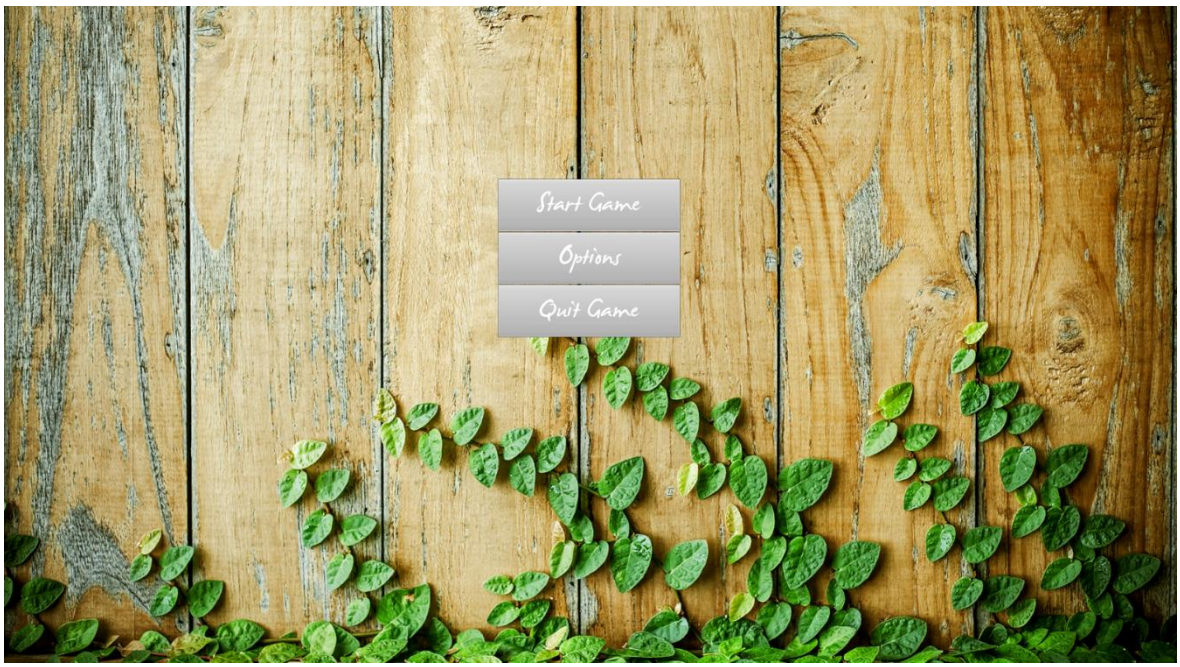
Κατά την εκκίνηση του Level και παράλληλα την δημιουργία του MainMenu HUD ενεργοποιείται το event BeginPlay που βρίσκεται μέσα σε αυτό. Έπειτα δημιουργούμε το MainMenuWidget Widget και το εμφανίζουμε με την μέθοδο Add to Viewport εμφανίζοντας και τον κέρσορα.

4.5.2 MainMenuWidget Widget

Στο σχεδιαστικό κομμάτι το κεντρικό μενού αποτελείται από μια εικόνα ως background και τρία buttons Start Game, Options και Quit Game. Εντός της επιλογής Options υπάρχουν 4 κουμπιά με 3 επιλογές ανάλυσης (1920*1080, 1280*800, 800*600) και μία επιλογή Return για επιστροφή στο κεντρικό μενού. Στο λειτουργικό κομμάτι υλοποιούνται τα παρακάτω:

- On Clicked (StartGame): Καλούμε την μέθοδο Open Level με στόχο το Level “FirstPersonExampleMap”, θέτουμε ώς μέθοδο εισαγωγής Game Only και εξαφανίζουμε τον κέρσορα.
- On Clicked (QuitGame): Καλούμε την μέθοδο Quit Game και τερματίζει το παιχνίδι.
- On Clicked (Options): Όταν επιλέγουμε την επιλογή options δια της μεθόδου Set Visibility εμφανίζουμε το μενού Options και εξαφανίζουμε το μενού Main.

- On Clicked (Return): Κάνει ακριβώς το αντίθετο, δια της μεθόδου Set Visibility εμφανίζουμε το μενού Main και εξαφανίζουμε το μενού Options.
- On Clicked (1920*1080): Με την χρήση της μεθόδου Execute Console Command εκτελούμε την εντολή “setres 1920x1080” και αλλάζουμε την ανάλυση του παιχνιδιού σε 1920*1080 pixel.
- On Clicked (1280*800): Με την χρήση της μεθόδου Execute Console Command εκτελούμε την εντολή “setres 1280x800w” και αλλάζουμε την ανάλυση του παιχνιδιού σε 1280*800 pixel.
- On Clicked (800*600): Με την χρήση της μεθόδου Execute Console Command εκτελούμε την εντολή “setres 800x600w” και αλλάζουμε την ανάλυση του παιχνιδιού σε 800*600 pixel.



Εικόνα 40 "Main Menu"

Εικόνα 41 "Main Menu Options"

4.6 Objectives Sequence

Σε αυτό το σημείο θα αναλύσουμε τους στόχους που πρέπει να ολοκληρώνει ο παίχτης και τον τρόπο που υλοποιούνται ώστε να φτάσει στον τερματισμό. Ο έλεγχος της σειράς εκτέλεσης των στόχων γίνεται με την χρήση της μεταβλητής Story_sequence που βρίσκεται μέσα στο Blueprint “First Person Character”.

4.6.1 Door_BP

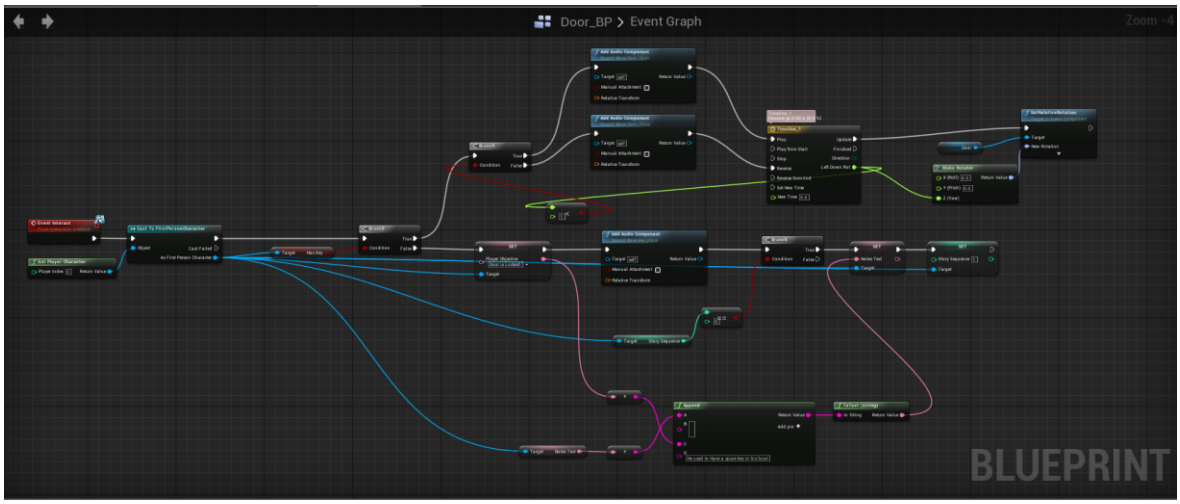
Το Blueprint Door_BP εκτελεί τις λειτουργίες μιας κανονικής πόρτας. Χρειάζεται κλειδί για να την ανοίξουμε, αναπαράγεται ήχος κάθε φορά που ανοίγει ή κλείνει και στην περίπτωση μας ενημερώνει και κάποιες μεταβλητές για την εξέλιξη της ιστορίας.

Αρχικά στις ρυθμίσεις κλάσης (Class Settings) του Blueprint έχουμε κάνει implement το Interaction Interface και έτσι υλοποιήσαμε το *Event Interact*. Όταν ενεργοποιείται το *Event Interact* με την χρήση του κόμβου Cast το *FirstPersonCharacter* έχουμε πρόσβαση στις μεταβλητές του *FirstPersonCharacter*. Ελέγχουμε αν ο χαρακτήρας μας κατέχει το κλειδί με την boolean μεταβλητή Has Key. Με την χρήση του κόμβου Branch ανάλογα με το αν κατέχει το κλειδί (True) ή όχι (False) γίνεται διαχωρισμός στις ενέργειες που έπονται.

- Εάν κατέχει το κλειδί ελέγχουμε αν η πόρτα είναι ανοιχτή ή κλειστή ώστε κατά την εκτέλεση να ανοίξει ή να κλείσει. Με την χρήση του κόμβου Add Audio Component αναπαράγουμε τον αντίστοιχο ήχο όταν ανοίξει η πόρτα (Door_Open) ή όταν κλείσει (Door_Close). Έπειτα χρησιμοποιούμε τον κόμβο Timeline. Εάν η πόρτα ήταν κλειστή εκτελείται κανονικά το

χρονοδιάγραμμα (Play) timeline διαφορετικά εάν ήταν ανοιχτή εκτελείται αντίστροφα (Reverse). Εσωτερικά στο Timeline έχουμε δημιουργήσει ένα Float Track (left_down_rot) με μήκος 2.0 για να εκτελέσουμε ομαλά την περιστροφή της πόρτας από ανοιχτή σε κλειστή και αντίστροφα. Ξεκινάμε έχοντας το πρώτο key στην τιμή 0 στον χρόνο 0.0 και το δεύτερο key στην τιμή 100 και χρόνο 2.0. Τέλος με την χρήση του κόμβου SetRelativeRotation, με στόχο την Πόρτα (Door) και ως New Rotation τη τιμή του κόμβου Make Rotator σύμφωνα με την τιμή του Left Down Rot από το timeline στην θέση Z (του κάθετου άξονα), εκτελούμε την περιστροφή της πόρτας.

- Εάν δεν έχει το κλειδί τότε αλλάζουμε την τιμή της μεταβλητής Player Objective σε “Door is Locked!” και αναπαράγουμε τον ήχο “key_place”. Τέλος ενημερώνουμε την τιμή της μεταβλητής Notes Text προσθέτοντας τα νέα στοιχεία στο σημειωματάριο και την τιμή της μεταβλητής Story Sequence σε 1 δείχνονταν ότι τότε ολοκληρώθηκε ο πρώτος στόχος.



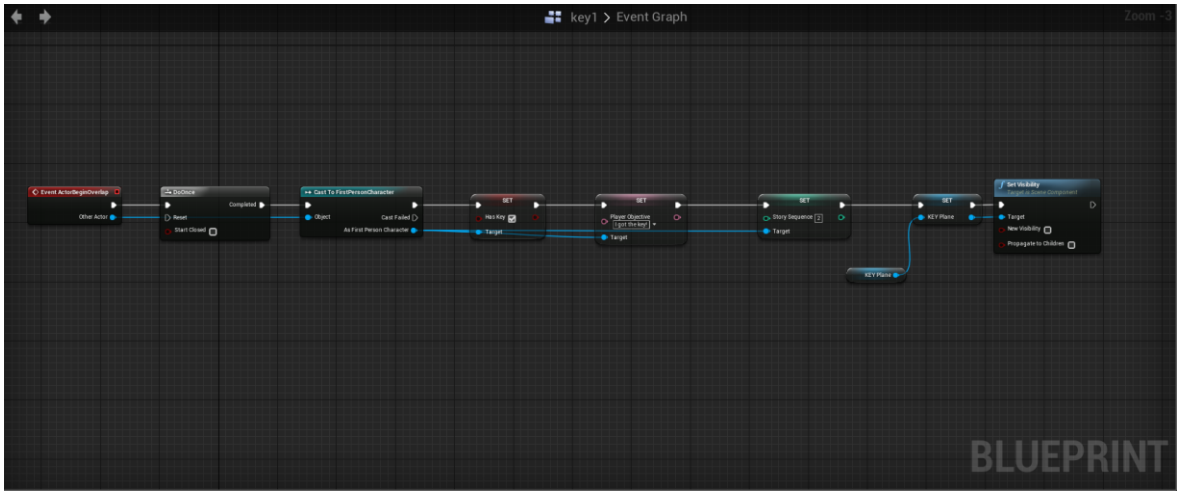
Εικόνα 42 " Door Blueprint"

4.6.2 key1

Το Blueprint key περιέχει το κλειδί το οποίο είναι απαραίτητο για να ανοίξει η πόρτα Door_BP.

Για την υλοποίηση του συγκεκριμένου objective χρησιμοποιήθηκε το event ActorBeginOverlap και η εκτέλεση γίνεται μια φορά με την χρήση του κόμβου DoOnce. Με την χρήση του παραπάνω event και τον κόμβο Cast Το *FirstPersonCharacter* έχουμε πρόσβαση στις μεταβλητές που υπάρχουν στο *FirstPersonCharacter* και ενημερώνουμε την μεταβλητή Has Key σε True, ενημερώνουμε την μεταβλητή Player Objective σε “I got the key!” και την

μεταβλητή Story Sequence σε 2. Τέλος με την χρήση του κόμβου Set Visibility αποκρύπτουμε το αντικείμενο του κλειδιού.

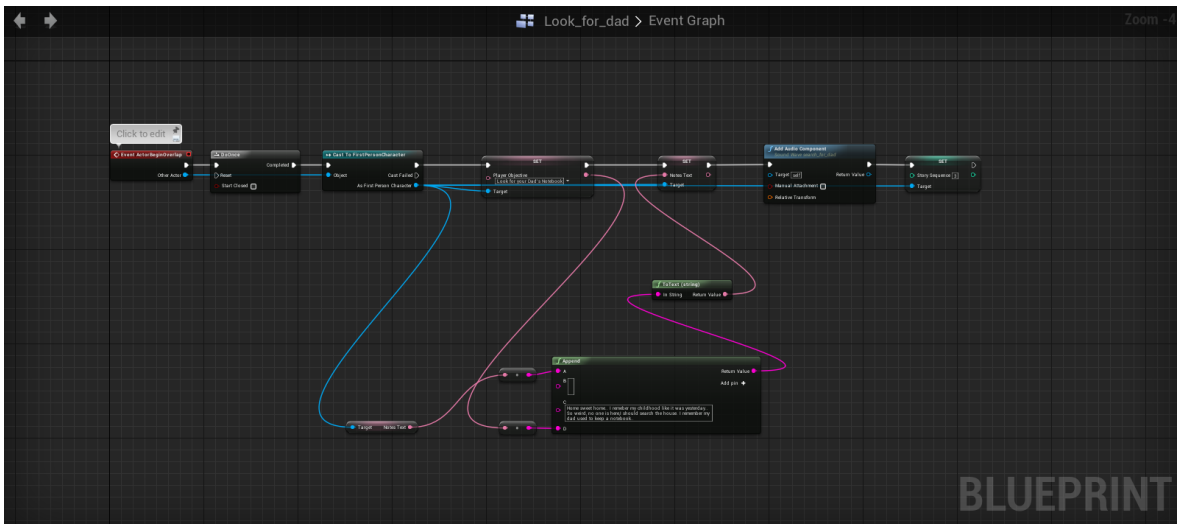


Εικόνα 43 "Key Blueprint"

4.6.3 Look_for_dad

Το Blueprint Look_for_dad αναπαράγει έναν μονόλογο του πρωταγωνιστή και ενημερώνει και κάποιες μεταβλητές για την εξέλιξη της ιστορίας

Όπως και στο προηγούμενο Objective για την υλοποίηση χρησιμοποιήθηκε το event ActorBeginOverlap και η εκτέλεση γίνεται μια φορά με την χρήση του κόμβου DoOnce. Με την χρήση του παραπάνω event και τον κόμβο Cast To *FirstPersonCharacter* έχουμε πρόσβαση στις μεταβλητές που υπάρχουν στο *FirstPersonCharacter* και ενημερώνουμε την τιμή της μεταβλητής Player Objective σε “Look for your Dad’s Notebook. Έπειτα ενημερώνουμε το σημειωματάριο δηλαδή την μεταβλητή Notes Text και αναπαράγουμε τον ήχο “search_for_dad”. Τέλος ενημερώνουμε την μεταβλητή Story Sequence στην τιμή 3.



Εικόνα 44 " Look For your Dad Blueprint"

4.6.4 Notebook

Το Blueprint Notebook αναπαράγει έναν μονόλογο του πρωταγωνιστή, ανοίγει το widget “Notebook Widget” και ενημερώνει και κάποιες μεταβλητές για την εξέλιξη της ιστορίας.

Το συγκεκριμένο Blueprint αποτελείται από δύο event που υλοποιούν διαφορετικές λειτουργίες. Στις ρυθμίσεις κλάσης (Class Settings) του Blueprint έχουμε κάνει implement το Interaction Interface και έτσι υλοποιήσαμε το *Event Interact*. Επίσης έχουμε προσθέσει το event ActorEndOverlap. Έχουμε δημιουργήσει και μία μεταβλητή τύπου Boolean την BookVis για να ελέγχουμε εάν έχουμε ανοιχτό το ημερολόγιο.

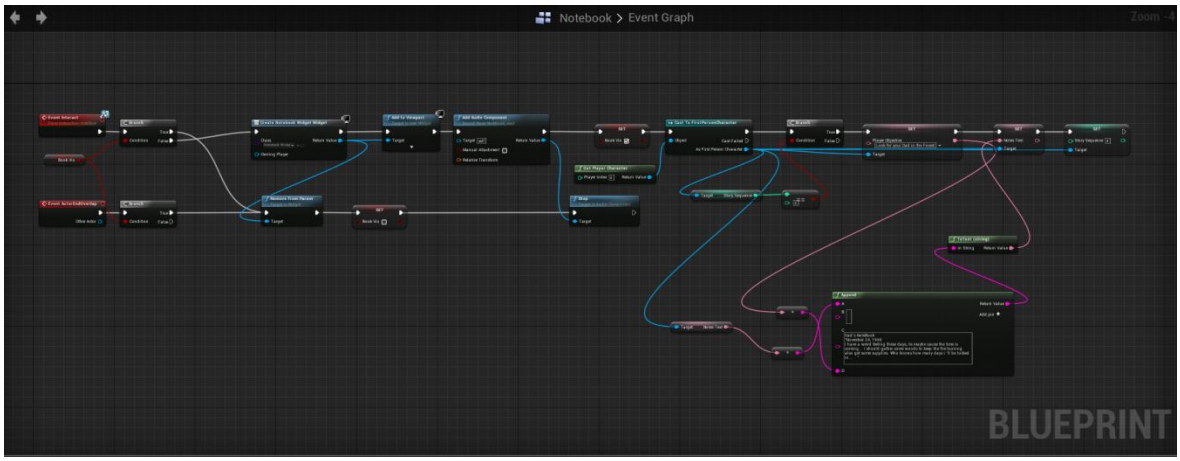
Event Interact

Ξεκινώντας της εκτέλεση χρησιμοποιούμε τον κόμβο Branch με συνθήκη την τιμή της μεταβλητής Book Vis.

- Εάν η τιμή της είναι False δηλαδή δεν έχουμε ανοίξει το ημερολόγιο οπότε δημιουργούμε το Widget “Notebook Widget” και το εμφανίζουμε με την χρήση του κόμβου Add to Viewport και αναπαράγουμε τον ήχο “Notebook_read” με την χρήση του κόμβου Add Audio Component. Έπειτα αλλάζουμε την τιμή της μεταβλητής Book Vis σε true και με την χρήση του κόμβου Cast Το *FirstPersonCharacter* έχουμε πρόσβαση στις μεταβλητές που υπάρχουν στο *FirstPersonCharacter* και ενημερώνουμε την τιμή της μεταβλητής Player Objective σε “Look for your Dad in the Forest”, την τιμή της μεταβλητής Notes Text και το Story Sequence σε 4.
- Εάν η τιμή της είναι True αφαιρούμε το Widget με την χρήση του κόμβου Remove from Parent, θέτουμε την τιμή της μεταβλητής Book Vis σε False και διακόπτουμε την αναπαραγωγή του ήχου “Notebook_read” με την χρήση του κόμβου Stop.

Event ActorEndOverlap

Ξεκινώντας της εκτέλεση χρησιμοποιούμε τον κόμβο Branch με συνθήκη την τιμή της μεταβλητής Book Vis. Εάν είναι αληθής, με την χρήση του κόμβου Remove from Parent, θέτουμε την τιμή της μεταβλητής Book Vis σε False και διακόπτουμε την αναπαραγωγή του ήχου “Notebook_read” με την χρήση του κόμβου Stop.



Εικόνα 45 "Notebook Blueprint"

4.6.5 Book_Page

Το Blueprint Book_Page αναπαράγει έναν μονόλογο του πρωταγωνιστή, ανοίγει το widget “Page Widget” και ενημερώνει και κάποιες μεταβλητές για την εξέλιξη της ιστορίας.

Όπως και το προηγούμενο και αυτό το Blueprint αποτελείται από δύο event που υλοποιούν διαφορετικές λειτουργίες. Στις ρυθμίσεις κλάσης (Class Settings) του Blueprint έχουμε κάνει implement το Interaction Interface και έτσι υλοποιήσαμε το *Event Interact*. Επίσης έχουμε προσθέσει το event ActorEndOverlap. Έχουμε δημιουργήσει και μία μεταβλητή τύπου Boolean την BookVis για να ελέγχουμε εάν έχουμε ανοιχτή την σελίδα.

Event Interact

Ξεκινώντας της εκτέλεση χρησιμοποιούμε τον κόμβο Branch με συνθήκη την τιμή της μεταβλητής Book Vis.

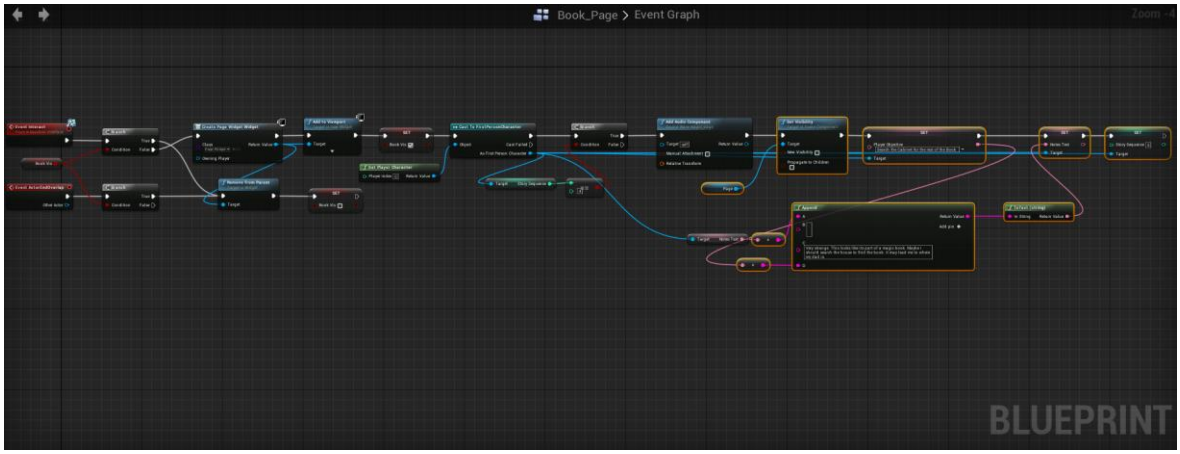
- Εάν η τιμή της είναι False δηλαδή δεν έχουμε ανοίξει την σελίδα οπότε δημιουργούμε το Widget “Page Widget” και την εμφανίζουμε με την χρήση του κόμβου Add to Viewport. Έπειτα αλλάζουμε την τιμή της μεταβλητής Book Vis σε true και με την χρήση του κόμβου Cast To *FirstPersonCharacter* έχουμε πρόσβαση στις μεταβλητές που υπάρχουν στο *FirstPersonCharacter*. Με την χρήση του κόμβου branch ελέγχουμε εάν η τιμή της μεταβλητής Story Sequence είναι ίση με 4 ώστε να μην μπορεί να εκτελεστεί η συνέχεια εάν δεν έχουμε ολοκληρώσει τα προηγούμενα objective. Εάν η συνθήκη είναι αληθής τότε αναπαράγουμε τον ήχο “magic_page” με την χρήση του κόμβου Add Audio Component και έπειτα με την χρήση του κόμβου Set Visibility και με στόχο του Static Mesh “Page” αποκρύπτουμε την σελίδα. Τέλος ενημερώνουμε την τιμή της μεταβλητής

Player Objective σε “ Search the Cabinet for the rest of the Book ”, την τιμή της μεταβλητής Notes Text και το Story Sequence σε 5.

- Εάν η τιμή της είναι True αφαιρούμε το Widget με την χρήση του κόμβου Remove from Parent και θέτουμε την τιμή της μεταβλητής Book Vis σε False.

Event ActorEndOverlap

Ξεκινώντας της εκτέλεση χρησιμοποιούμε τον κόμβο Branch με συνθήκη την τιμή της μεταβλητής Book Vis. Εάν είναι αληθής, με την χρήση του κόμβου Remove from Parent και θέτουμε την τιμή της μεταβλητής Book Vis σε False.



Εικόνα 46 "Book_Page Blueprint"

4.6.6 Magic_History

Το Blueprint Magic_History αναπαράγει έναν μονόλογο του πρωταγωνιστή, ανοίγει το widget “Magic Book Widget” και ενημερώνει και κάποιες μεταβλητές για την εξέλιξη της ιστορίας.

Όπως και το προηγούμενο και αυτό το Blueprint αποτελείται από δύο event που υλοποιούν διαφορετικές λειτουργίες. Στις ρυθμίσεις κλάσης (Class Settings) του Blueprint έχουμε κάνει implement το Interaction Interface και έτσι υλοποιήσαμε το *Event Interact*. Επίσης έχουμε προσθέσει το event ActorEndOverlap. Έχουμε δημιουργήσει και μία μεταβλητή τύπου Boolean την BookVis για να ελέγχουμε εάν έχουμε ανοιχτό το βιβλίο.

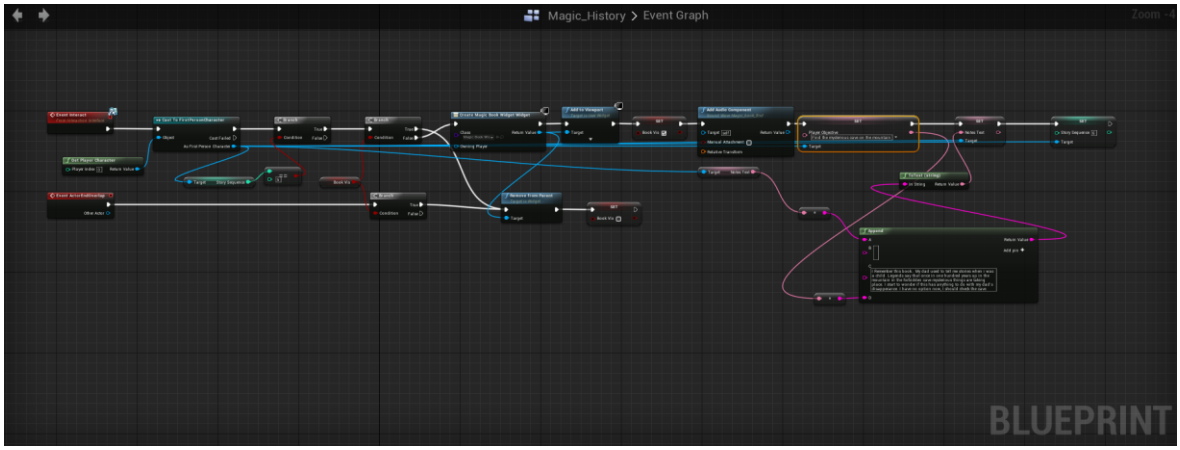
Event Interact

Ξεκινώντας της εκτέλεση χρησιμοποιούμε τον κόμβο Cast Το *FirstPersonCharacter* ώστε έχουμε πρόσβαση στις μεταβλητές που υπάρχουν στο *FirstPersonCharacter*. Με την χρήση του κόμβου branch ελέγχουμε εάν η τιμή της μεταβλητής Story Sequence είναι ίση με 5 ώστε να μην μπορεί να εκτελεστεί η συνέχεια εάν δεν έχουμε ολοκληρώσει τα προηγούμενα objective. Έπειτα χρησιμοποιούμε τον κόμβο Branch με συνθήκη την τιμή της μεταβλητής Book Vis.

- Εάν η τιμή της είναι False δηλαδή δεν έχουμε ανοίξει το μαγικό βιβλίο οπότε δημιουργούμε το Widget “Magic Book Widget” και το εμφανίζουμε με την χρήση του κόμβου Add to Viewport. Έπειτα αλλάζουμε την τιμή της μεταβλητής Book Vis σε true και αναπαράγουμε τον ήχο “Magic_book_find” με την χρήση του κόμβου Add Audio Component. Τέλος ενημερώνουμε την τιμή της μεταβλητής Player Objective σε “Find the mysterious cave on the mountain ”, την τιμή της μεταβλητής Notes Text και το Story Sequence σε 6.
- Εάν η τιμή της είναι True αφαιρούμε το Widget με την χρήση του κόμβου Remove from Parent και θέτουμε την τιμή της μεταβλητής Book Vis σε False.

Event ActorEndOverlap

Ξεκινώντας της εκτέλεση χρησιμοποιούμε τον κόμβο Branch με συνθήκη την τιμή της μεταβλητής Book Vis. Εάν είναι αληθής, με την χρήση του κόμβου Remove from Parent και θέτουμε την τιμή της μεταβλητής Book Vis σε False.

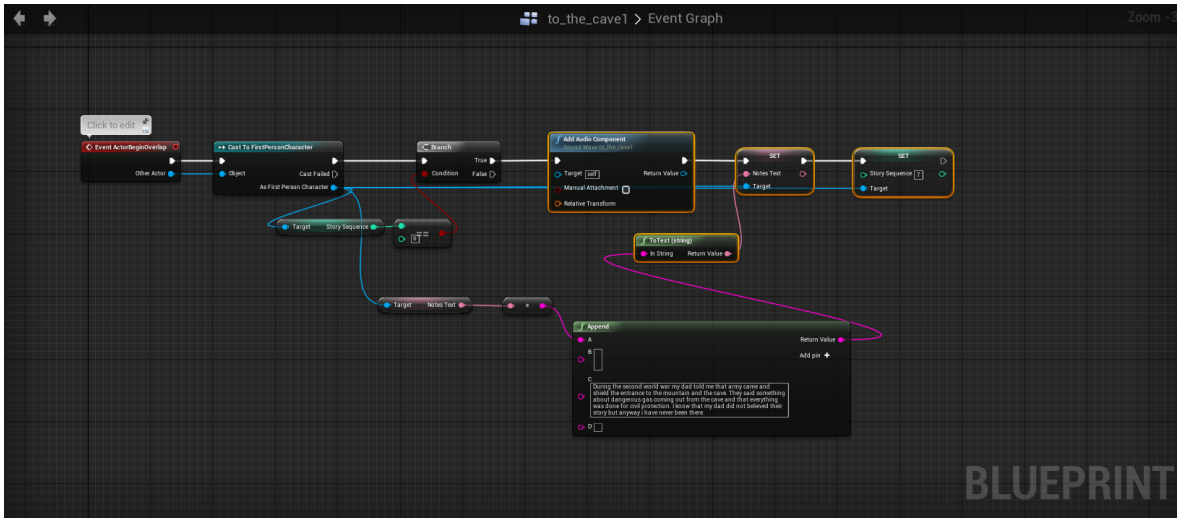


Εικόνα 47 "Magic_History Blueprint"

4.6.7 to_the_cave1

Το συγκεκριμένο Blueprint αναπαράγει έναν μονόλογο του πρωταγωνιστή, και ενημερώνει και κάποιες μεταβλητές για την εξέλιξη της ιστορίας.

Στο συγκεκριμένο Blueprint υλοποιούμε το event ActorBeginOverlap το οποίο ενεργοποιείται όταν ο παίχτης εισαχθεί στην περιοχή του blueprint. Ξεκινώντας την εκτέλεση χρησιμοποιούμε τον κόμβο Cast Το *FirstPersonCharacter* ώστε έχουμε πρόσβαση στις μεταβλητές που υπάρχουν στο *FirstPersonCharacter*. Με την χρήση του κόμβου branch ελέγχουμε εάν η τιμή της μεταβλητής Story Sequence είναι ίση με 6 ώστε να μην μπορεί να εκτελεστεί η συνέχεια εάν δεν έχουμε ολοκληρώσει τα προηγούμενα objective. Έπειτα αναπαράγουμε τον ήχο “to_the_cave1” με την χρήση του κόμβου Add Audio Component. Τέλος ενημερώνουμε την τιμή της μεταβλητής Notes Text και το Story Sequence σε 7.

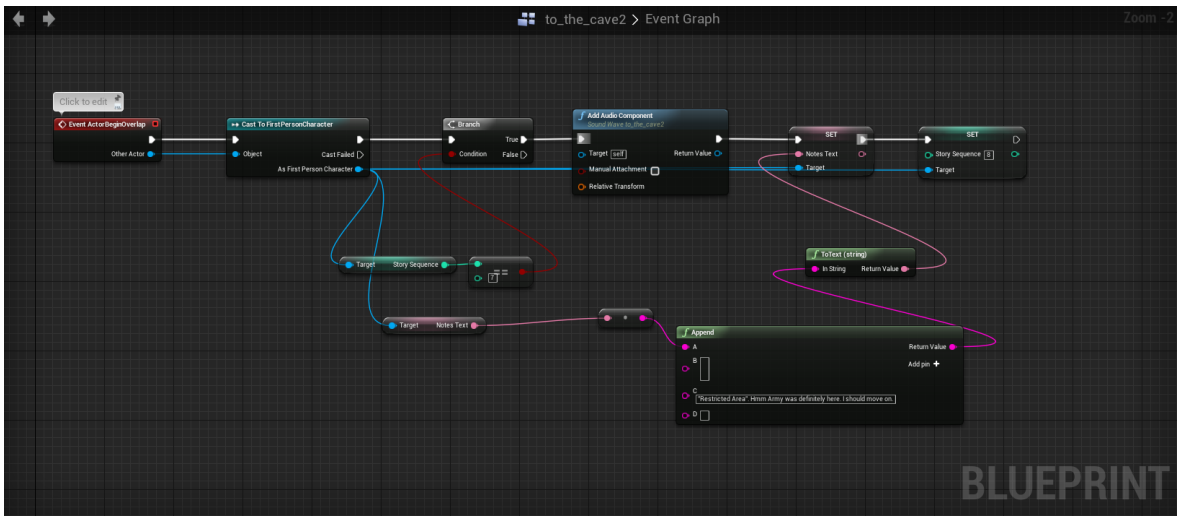


Εικόνα 48 "To the cave 1 Blueprint"

4.6.8 to_the_cave2

Το συγκεκριμένο Blueprint αναπαράγει έναν μονόλογο του πρωταγωνιστή, και ενημερώνει και κάποιες μεταβλητές για την εξέλιξη της ιστορίας.

Στο συγκεκριμένο Blueprint υλοποιούμε το event ActorBeginOverlap το οποίο ενεργοποιείται όταν ο παίχτης εισαχθεί στην περιοχή του blueprint. Ξεκινώντας την εκτέλεση χρησιμοποιούμε τον κόμβο Cast Το *FirstPersonCharacter* ώστε έχουμε πρόσβαση στις μεταβλητές που υπάρχουν στο *FirstPersonCharacter*. Με την χρήση του κόμβου branch ελέγχουμε εάν η τιμή της μεταβλητής Story Sequence είναι ίση με 7 ώστε να μην μπορεί να εκτελεστεί η συνέχεια εάν δεν έχουμε ολοκληρώσει τα προηγούμενα objective. Έπειτα αναπαράγουμε τον ήχο “to_the_cave2” με την χρήση του κόμβου Add Audio Component. Τέλος ενημερώνουμε την τιμή της μεταβλητής Notes Text και το Story Sequence σε 8.

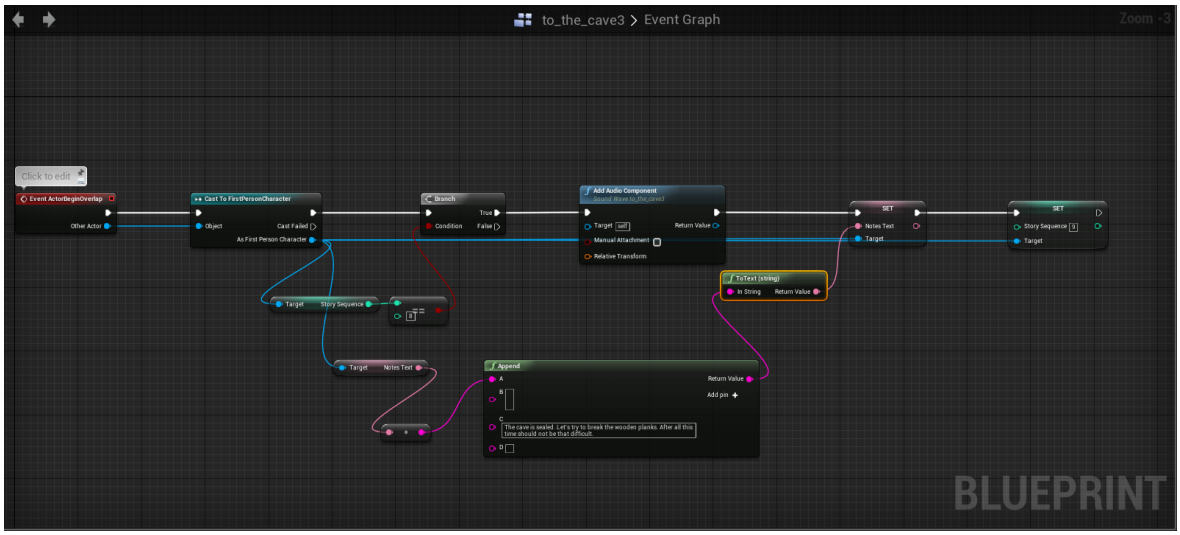


Εικόνα 49 "To the cave 2 Blueprint"

4.6.9 to_the_cave3

Το συγκεκριμένο Blueprint αναπαράγει έναν μονόλογο του πρωταγωνιστή, και ενημερώνει και κάποιες μεταβλητές για την εξέλιξη της ιστορίας.

Στο συγκεκριμένο Blueprint υλοποιούμε το event ActorBeginOverlap το οποίο ενεργοποιείται όταν ο παίχτης εισαχθεί στην περιοχή του blueprint. Ξεκινώντας την εκτέλεση χρησιμοποιούμε τον κόμβο Cast Το *FirstPersonCharacter* ώστε έχουμε πρόσβαση στις μεταβλητές που υπάρχουν στο *FirstPersonCharacter*. Με την χρήση του κόμβου branch ελέγχουμε εάν η τιμή της μεταβλητής Story Sequence είναι ίση με 8 ώστε να μην μπορεί να εκτελεστεί η συνέχεια εάν δεν έχουμε ολοκληρώσει τα προηγούμενα objective. Έπειτα αναπαράγουμε τον ήχο “to_the_cave3” με την χρήση του κόμβου Add Audio Component. Τέλος ενημερώνουμε την τιμή της μεταβλητής Notes Text και το Story Sequence σε 9.

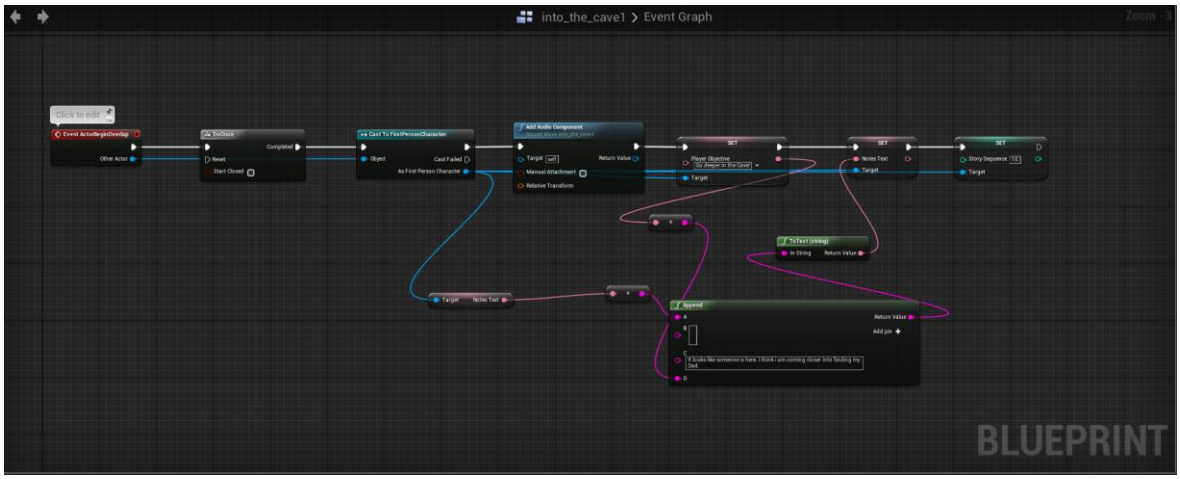


Εικόνα 50 "To the cave 3 Blueprint"

4.6.10 into_the_cave1

Το συγκεκριμένο Blueprint αναπαράγει έναν μονόλογο του πρωταγωνιστή, και ενημερώνει και κάποιες μεταβλητές για την εξέλιξη της ιστορίας.

Στο συγκεκριμένο Blueprint υλοποιούμε το event ActorBeginOverlap το οποίο ενεργοποιείται όταν ο παίχτης εισαχθεί στην περιοχή του blueprint. Ξεκινώντας την εκτέλεση χρησιμοποιούμε τον κόμβο DoOnce ώστε η επακόλουθη εκτέλεση να γίνει μια φορά και τον κόμβο Cast Το *FirstPersonCharacter* ώστε έχουμε πρόσβαση στις μεταβλητές που υπάρχουν στο *FirstPersonCharacter*. Έπειτα χρησιμοποιούμε αναπαράγουμε τον ήχο “into_the_cave1” με την χρήση του κόμβου Add Audio Component. Τέλος ενημερώνουμε την τιμή της μεταβλητής Player Objective σε “Go deeper in the Cave!”, Notes Text και το Story Sequence σε 10.

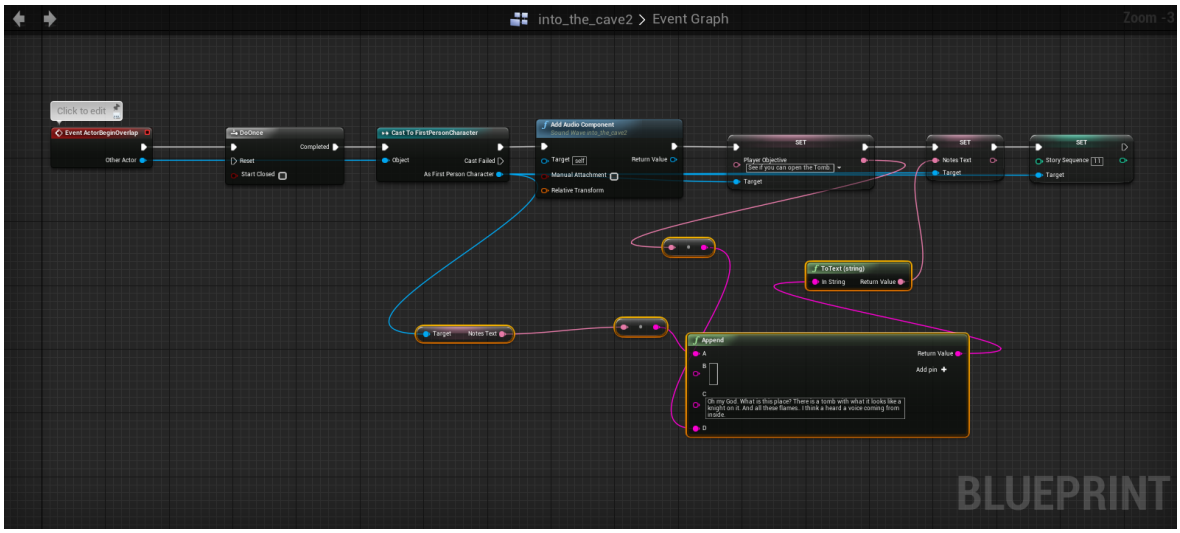


Εικόνα 51 "Into the cave 1 Blueprint"

4.6.11 into_the_cave2

Το συγκεκριμένο Blueprint αναπαράγει έναν μονόλογο του πρωταγωνιστή, και ενημερώνει και κάποιες μεταβλητές για την εξέλιξη της ιστορίας.

Στο συγκεκριμένο Blueprint υλοποιούμε το event ActorBeginOverlap το οποίο ενεργοποιείται όταν ο παίχτης εισαχθεί στην περιοχή του blueprint. Ξεκινώντας την εκτέλεση χρησιμοποιούμε τον κόμβο DoOnce ώστε η επακόλουθη εκτέλεση να γίνει μια φορά και τον κόμβο Cast To *FirstPersonCharacter* ώστε έχουμε πρόσβαση στις μεταβλητές που υπάρχουν στο *FirstPersonCharacter*. Έπειτα χρησιμοποιούμε αναπαράγουμε τον ήχο “into_the_cave2” με την χρήση του κόμβου Add Audio Component. Τέλος ενημερώνουμε την τιμή της μεταβλητής Player Objective σε “See if you can open the Tomb.”, Notes Text και το Story Sequence σε 11.

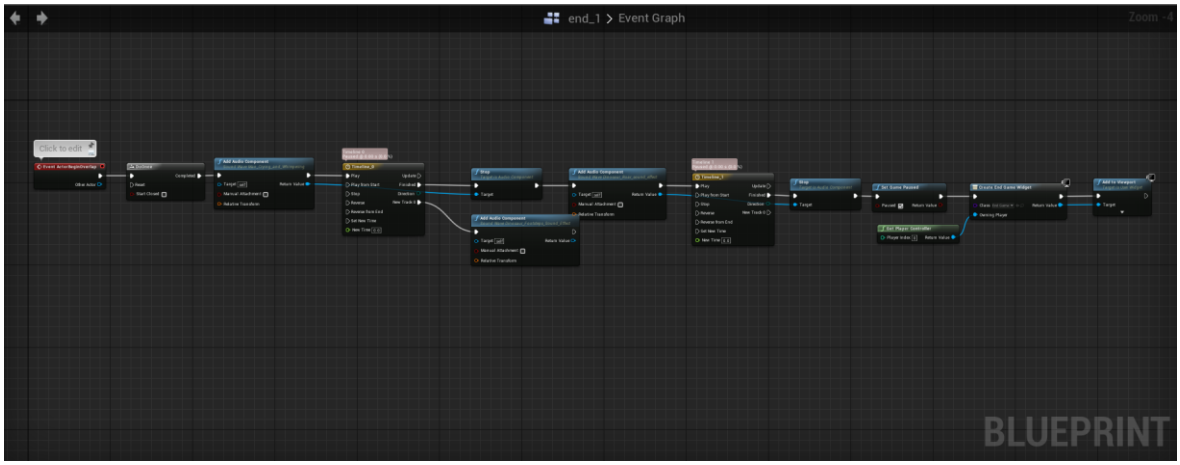


Εικόνα 52 "Into the cave 2 Blueprint"

4.6.12 end_1

Το Blueprint end_1 ουσιαστικά αναπαράγει μέσα από χρονοδιαγράμματα ήχους και με την ολοκλήρωση του ανοίγει το widget “End Game Widget”.

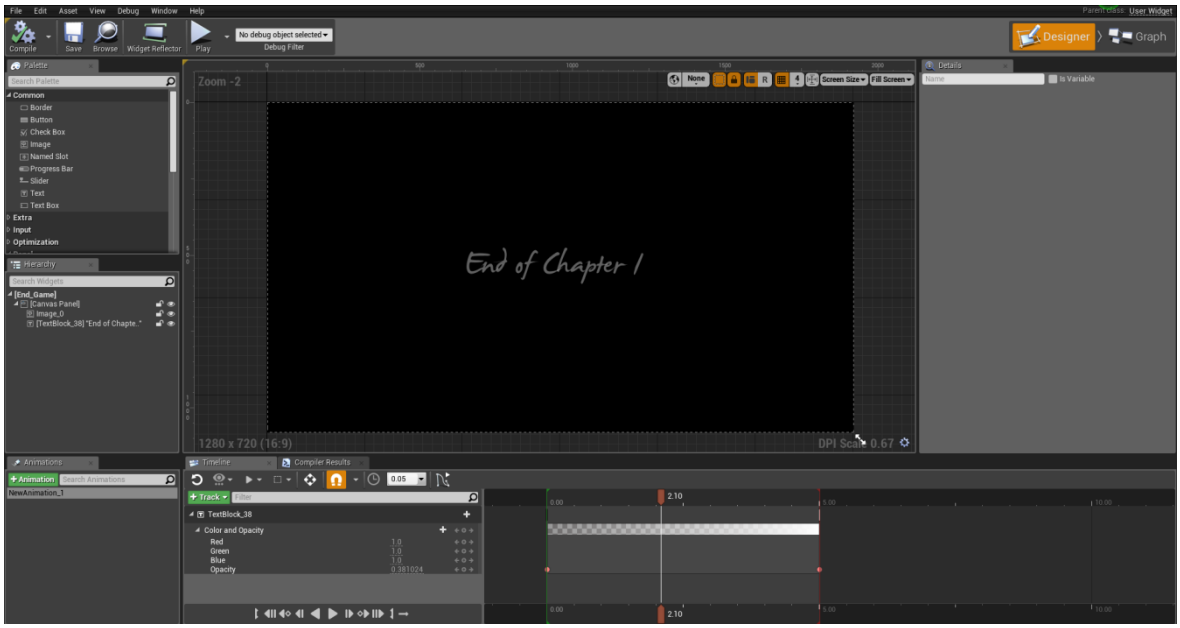
Το συγκεκριμένο objective αποτελεί το τελευταίο και μετά την εκτέλεση του ολοκληρώνεται το παιχνίδι. Αρχικά υλοποιούμε το event ActorBeginOverlap το οποίο ενεργοποιείται όταν ο παίχτης εισαχθεί στην περιοχή του blueprint. Ξεκινώντας την εκτέλεση χρησιμοποιούμε τον κόμβο DoOnce ώστε η επακόλουθη εκτέλεση να γίνει μια φορά και αναπαράγουμε τον ήχο “Man_Crying_and_Whimpering” με την χρήση του κόμβου Add Audio Component. Έπειτα με την χρήση του Timeline_0 όταν ολοκληρωθεί ο χρόνος εκτέλεσης του με την χρήση του κόμβου Stop και ως στόχο το προηγούμενο Audio Component διακόπτουμε την αναπαραγωγή. Παράλληλα κατά την εκτέλεση του προηγούμενου Timeline έχουμε προσθέσει και ένα float track στον οποίο πάνω αναπαράγουμε τον ήχο “Dinosaur_Footsteps_Sound_Effect” για 5 sec. Έπειτα αναπαράγουμε τον ήχο “Dinosaur_Roar_sound_effect” και με την χρήση του Timeline_1 και του κόμβου Stop διακόπτουμε την αναπαραγωγή μετά από 7 sec. Θέτουμε την κατάσταση του παιχνιδιού σε Paused με την χρήση του κόμβου Set Game Paused. Τέλος δημιουργούμε το Widget End Game και το εμφανίζουμε.



Εικόνα 53 "end_1 Blueprint"

End Game Widget

Όταν ανοίγουμε το συγκεκριμένο Widget εκτελείται το Event Construct μέσο του κόμβου Play Animation αναπαράγουμε το “New Animation 1” το οποίο έχει μέγεθος 5 sec όπου έχουμε δύο Key ένα στον χρόνο 0 όπου και το Opacity του TextBlock_38 είναι 0 και στον χρόνο 5 όπου το Opacity είναι 1. Με την χρήση του κόμβου Delay καθυστερούμε την συνέχεια της εκτέλεσης κατά τον χρόνο “Get End Time” του “New Animation 1” και αφού περάσει αυτός ο χρόνος με την χρήση του κόμβου Open Level ανοίγουμε το MainMenu Level επιστρέφοντας έτσι στο αρχικό μενού.

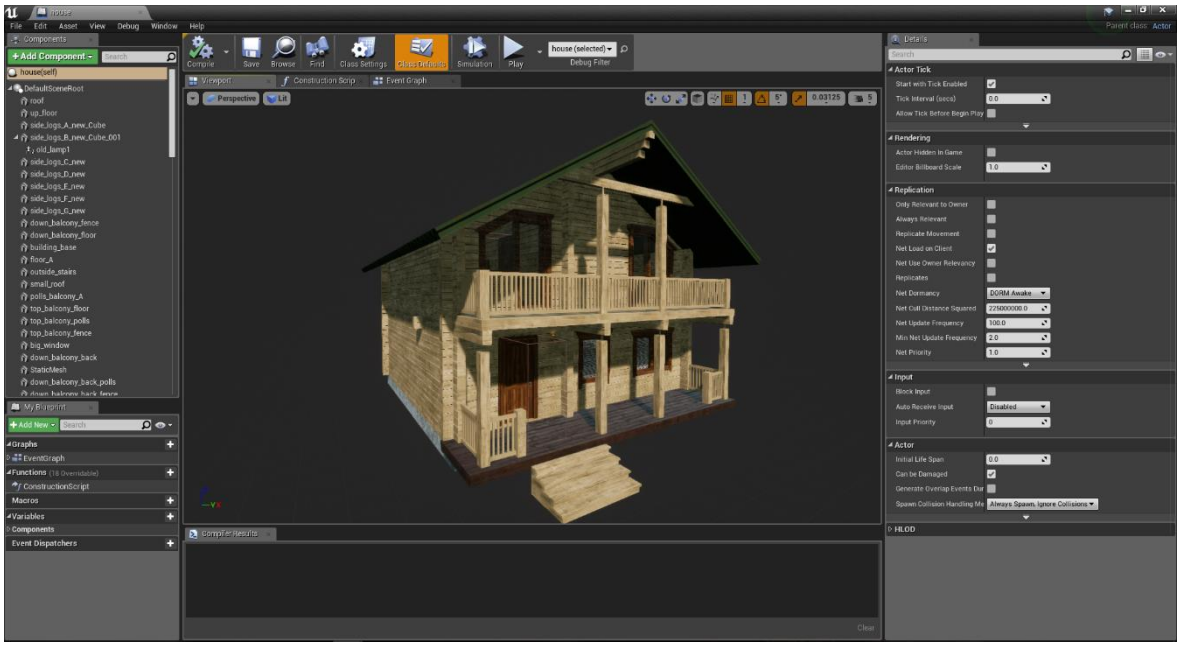


Εικόνα 54 "End Game Widget"

4.7 Main Assets

Για να μπορέσει να υλοποιηθεί σε ρεαλιστικό χρόνο το συγκεκριμένο project χρησιμοποιήθηκαν πολλά έτοιμα assets από τα οποία κάποια παρέχονται από την Epic Games, τον δημιουργό δηλαδή της Unreal Engine, αλλά και πολλά ακόμη τα οποία δημιουργήθηκαν από διάφορους δημιουργούς και παρέχονται δωρεάν προς χρήση. Μην ξεχνάμε ότι για την υλοποίηση ενός AAA (* A lot of time. * A lot of resources. * A lot of money.) παιχνιδιού διαφορετική προσέγγιση θα ήταν αδύνατη. Η εισαγωγή και η βελτιστοποίηση ενός τρισδιάστατου μοντέλου αποτελεί μία διαδικασία που απαιτεί αρκετή μελέτη και εκπαίδευση διότι μην δίνοντας σημασία σε παραμέτρους όπως το μέγεθος των texture και των triangles μπορεί να οδηγήσει σε ένα αρκετά ‘βαρύ’ για το hardware παιχνίδι. Παράλληλα αρκετά μοντέλα δημιουργήθηκαν από εμένα με την χρήση του εργαλείου Blender το οποίο αποτελεί δωρεάν και ανοιχτού κώδικα λογισμικό δημιουργίας τρισδιάστατου περιεχομένου. Παρακάτω παραθέτοντας μερικά από τα μοντέλα τα οποία χρησιμοποιήθηκαν και γίνεται πιο εκτενής ανάλυση σε αυτά που δημιουργήθηκαν από εμένα όπως και σε αυτά που εκτελούν κάποια λειτουργία.

4.7.1 House



Εικόνα 55 "House Blueprint"

Το σπίτι που αποτελεί σημαντικό κομμάτι του παιχνιδιού δημιουργήθηκε εξολοκλήρου από εμένα στο Blender 3D με τέτοιο τρόπο ώστε να αποτελείται από όσο το δυνατόν μικρότερο αριθμό πολύγωνων χωρίς να χάνεται ο ρεαλισμός. Αποτελείται από διαφορετικά κομμάτια (roof, up_floor, side_logs_A_new_Cube, side_logs_A_new_Cube_001, side_logs_C_new,…. και πολλά ακόμη) τα οποία συνθέτουν το Blueprint house το οποίο εικονίζεται στην εικόνα 55. Εφαρμόστηκαν material στο κάθε κομμάτι ανάλογα με την θέση που έχει στο σπίτι. Το σπίτι αποτελείται από 2 ορόφους. Ο κάτω όροφος αποτελείται από 2 δωμάτια (κουζίνα, σαλόνι) και έχει δύο εισόδους μια κεντρική είσοδο και μία στο πίσω μέρος. Μέσα από το σαλόνι με την χρήση εσωτερικής σκάλας ο παίχτης έχει πρόσβαση στον άνω όροφο ο οποίος αποτελείται και αυτός από δύο δωμάτια (υπνοδωμάτιο, αποθήκη-γραφείο). Όλα τα δωμάτια διαχωρίζονται με πόρτες τις οποίες όπως θα δούμε παρακάτω ο χρήστης μπορεί με την θέληση του να τις ανοίξει εφόσον διαθέτει το κλειδί. Περιμετρικά στους τοίχους υπάρχουν παράθυρα τα οποία αποτελούνται από material γυαλί (M_Glass) και ξύλο (M_Wood_Oak). Εσωτερικά του σπιτιού υπάρχουν «ηλεκτρικοί» διακόπτες που ελέγχουν το φως σε κάθε δωμάτιο ξεχωριστά σύμφωνα με την βούληση του παίχτη.

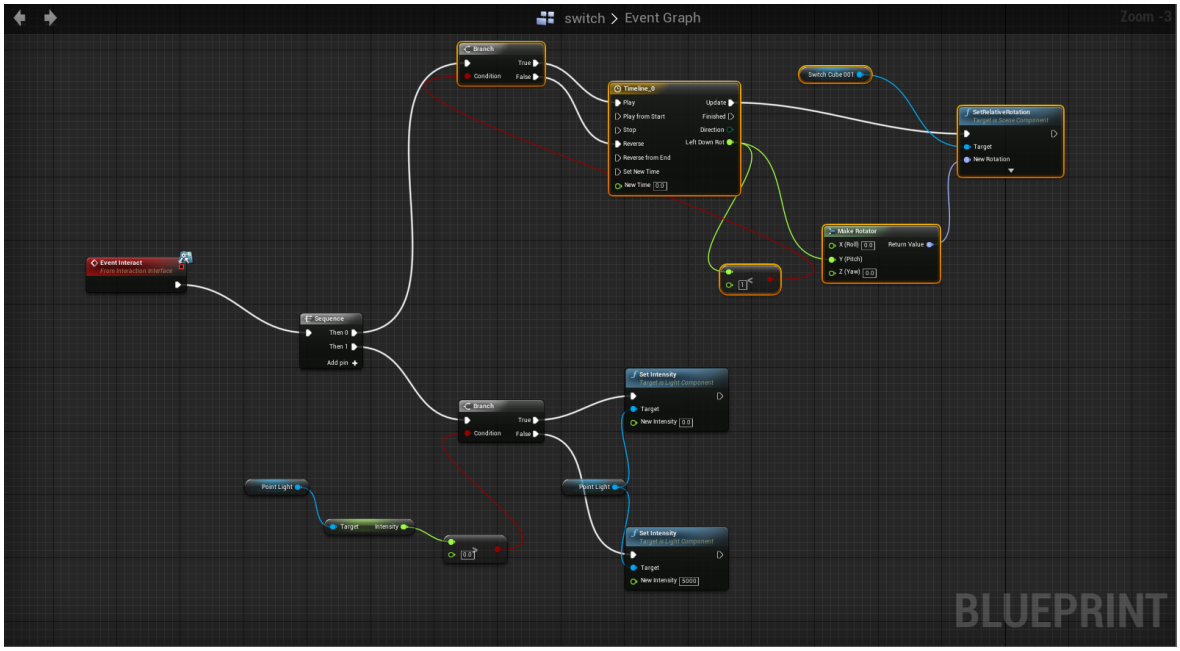
4.7.1.1 switch

Το συγκεκριμένο Blueprint αποτελείται από έναν διακόπτη και ένα PointLight και η χρήση του είναι όταν ο παίχτης στοχεύει τον διακόπτη με την χρήση του πλήκτρου “E” αλληλοεπιδρά με αυτόν και ανάβει ή σβήνει τα φώτα. Αρχικά μέσα στις ρυθμίσεις Κλάσης “Class Settings” έχουμε κάνει Implement το Interaction

Interface έτσι ώστε να μπορέσουμε να χρησιμοποιήσουμε το *Event Interact*. Υλοποιώντας το παραπάνω Event εκτελούμε τον κόμβο Sequence διότι κατά την ενεργοποίηση θέλουμε να εκτελέσουμε δύο ενέργειες. Πρώτον την αλλαγή θέσης του διακόπτη από ON σε OFF και αντίστροφα, Δεύτερον την αλλαγή της έντασης την πηγής του φωτός.

- Then 0: Με την χρήση του κόμβου Branch ελέγχουμε την τιμή του Left Down Rot ώστε αναγνωρίζοντας την θέση του διακόπτη να εκτελέσουμε την αντίστοιχη ενέργεια ώστε να αλλάξει θέση. Έπειτα με την χρήση του Timeline_0 εάν η συνθήκη είναι αληθής εκτελούμε την κανονική φορά του Timeline. Εσωτερικά του Timeline έχουμε δημιουργήσει ένα float track με δύο keyframes ένα στον χρόνο 0.0 με τιμή -30.0 και άλλο ένα στον χρόνο 0.35, που είναι και το μήκος του Timeline, και τιμή 30.0. Επειδή η κίνηση γίνεται στον άξονα Y δημιουργούμε έναν Rotator με την χρήση του κόμβου “Make Rotator” και θέτουμε την νέα περιστροφή με τον κόμβο SetRelativeRotation με στόχο τον διακόπτη “Switch Cube 001”. Εάν η συνθήκη είναι ψευδής εκτελείται το Timeline με αντίθετη φορά έτσι ώστε επαναφέρεται στην προηγούμενη θέση περιστροφής.

- Then 1: Με την χρήση του κόμβου Branch ελέγχουμε την τιμή Intensity του Point Light έτσι αν η τιμή του είναι μεγαλύτερη από 0.0 τότε θέτουμε την τιμή του Intesity σε 0.0 σβήνοντας έτσι το φως. Διαφορετικά θέτουμε την τιμή σε 5000 και έτσι ανάβουμε το φως.



Εικόνα 56 "switch Blueprint"

4.7.1.2 switch2

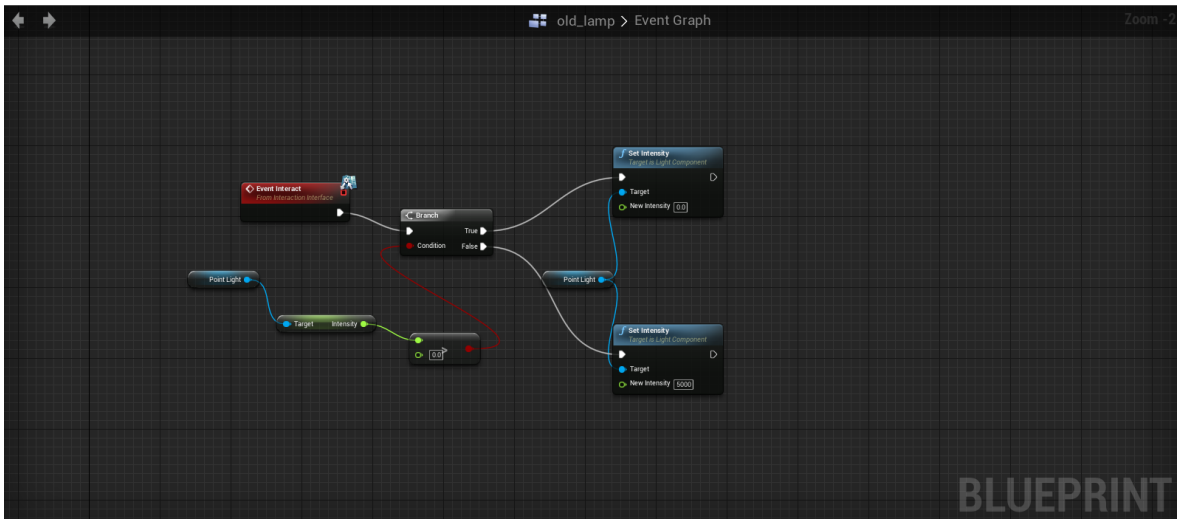
Ο τρόπος υλοποίησης είναι ίδιος με το switch με την διαφορά ότι έχουμε δύο πηγές φωτός.

4.7.1.3 switch3

Ο τρόπος υλοποίησης είναι ίδιος με το switch.

4.7.1.4 old_lamp

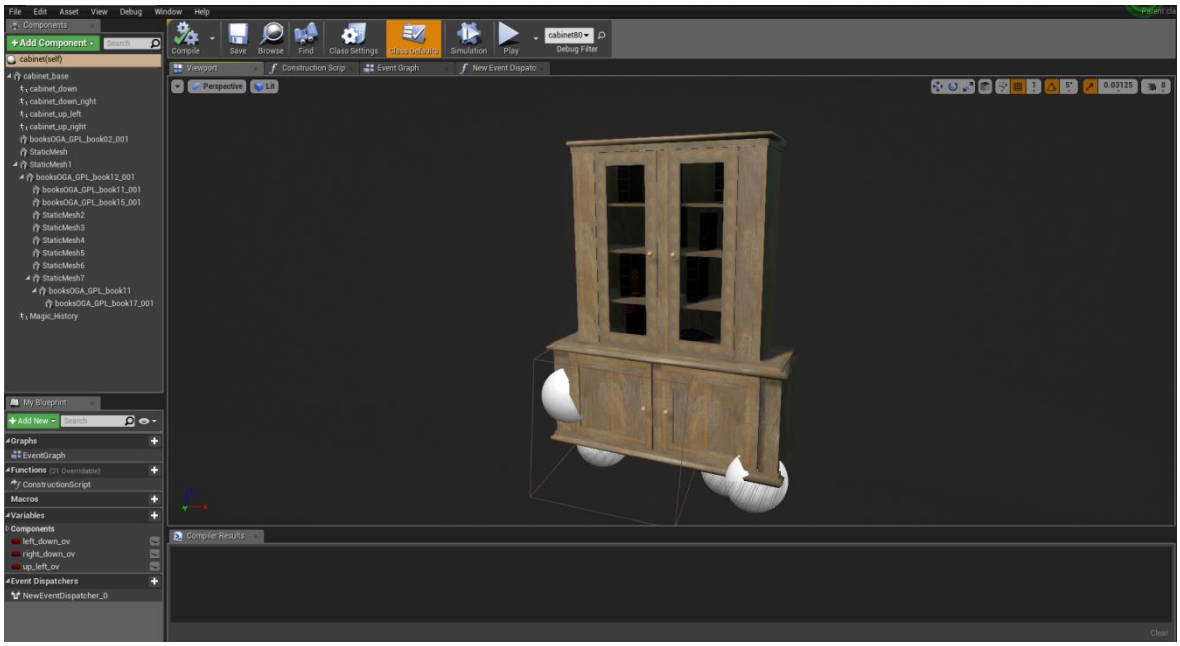
Το συγκεκριμένο Blueprint αποτελείται από το πορτατίφ και μία πηγή φωτός και η χρήση του είναι όταν ο παίχτης στοχεύει τον διακόπτη με την χρήση του πλήκτρου “E” αλληλοεπιδρά με αυτόν και ανάβει ή σβήνει το φως. Αρχικά μέσα στις ρυθμίσεις Κλάσης “Class Settings” έχουμε κάνει Implement το Interaction Interface έτσι ώστε να μπορέσουμε να χρησιμοποιήσουμε το *Event Interact*. Υλοποιώντας το παραπάνω Event και με την χρήση του κόμβου Branch ελέγχουμε την τιμή Intensity του Point Light ώστε να αλλάξουμε την τιμή του Intesity με την χρήση του κόμβου Set Intesity ανάβοντας ή σβήνοντας αντίστοιχα το φως.



Εικόνα 57 "old_lamp Blueprint"

4.7.1.5 cabinet

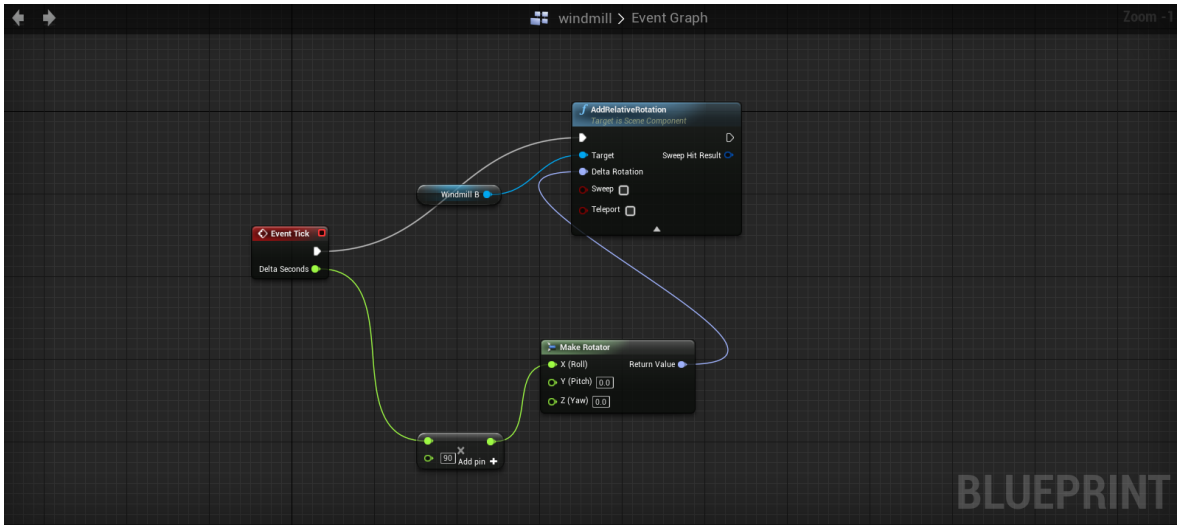
Στην βιβλιοθήκη-καμπίνα που υπάρχει στο σαλόνι έχουμε δώσει την δυνατότητα στα ντουλάπια της (cabinet_down, cabinet_down_right, cabinet_up_left, cabinet_up_right) με την χρήση του *Event Interact* να ανοίγουν και να κλείνουν. Μέσα σε κάθε ένα από αυτά στις ρυθμίσεις κλάσης έχει γίνει Implement το Interaction interface. Ο τρόπος που υλοποιείται το άνοιγμα και το κλείσιμο των ντουλαπιών είναι ίδιος με αυτόν της πόρτας και των διακοπτών. Εσωτερικά αυτής υπάρχει και το objective Magic_History.



Εικόνα 58 "cabinet Blueprint"

4.7.2 windmill

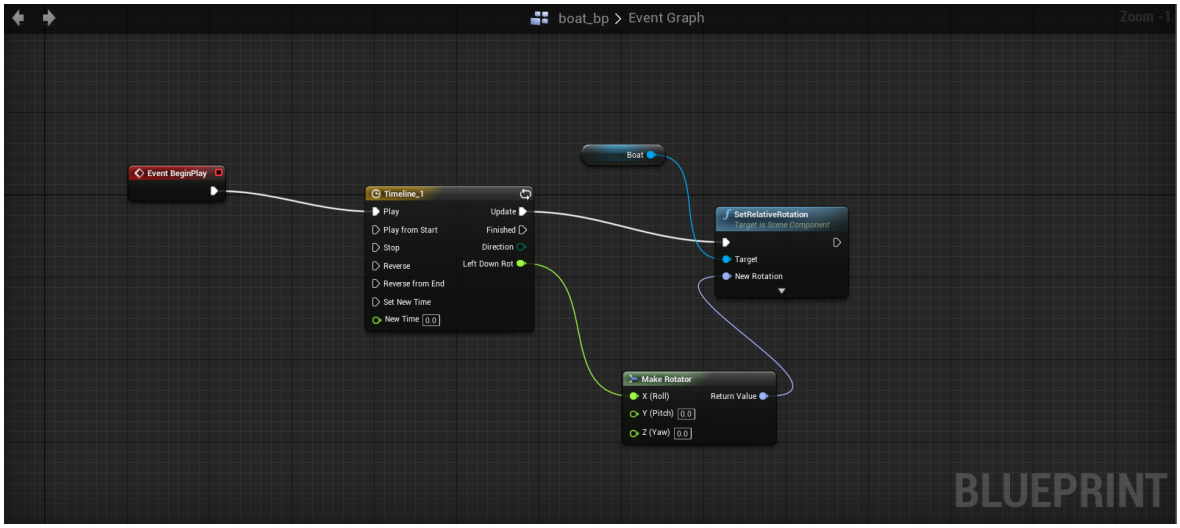
Το Blueprint του ανεμόμυλου αποτελείται από δύο κομμάτια, τον κορμό και τα πτερύγια στα οποία τα έχουμε δώσει κίνηση. Με την χρήση του event tick που καλείται σε κάθε frame εκτελούμε την περιστροφή των πτερυγίων στον άξονα X.



Εικόνα 59 "windmill Blueprint"

4.7.3 Boat_bp

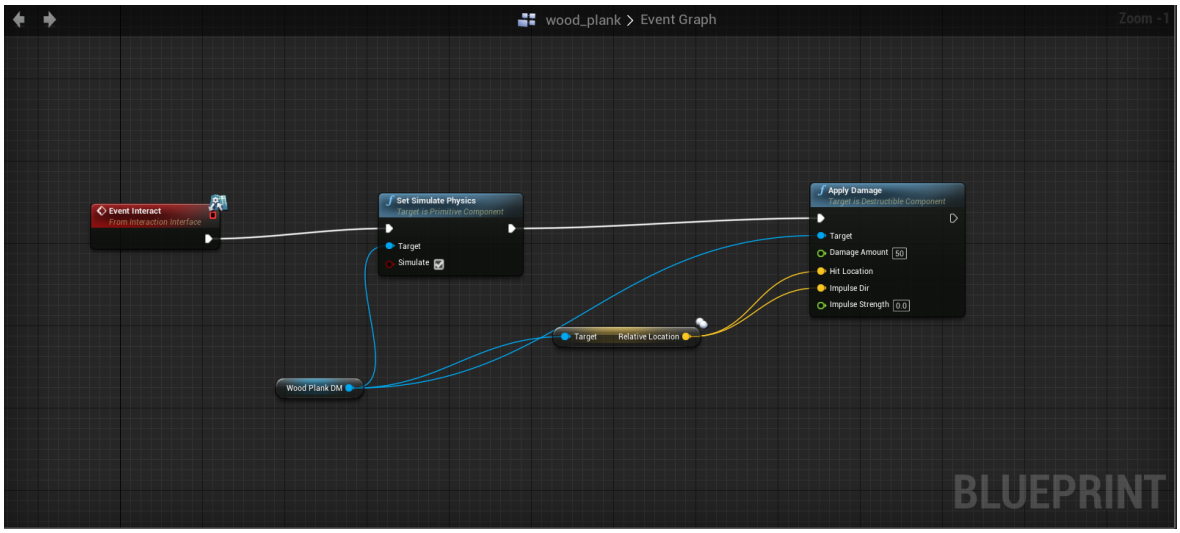
Με την χρήση του Event BeginPlay και του Timeline έχουμε δημιουργήσει ένα Loop το οποίο εκτελείται κατά την εκκίνηση του παιχνιδιού και ουσιαστικά αλλάζει την τιμή της περιστροφής της βάρκας στον άξονα X δίνοντας την ψευδαίσθηση της κίνησης επάνω στο νερό. Στο συγκεκριμένο blueprint εμπεριέχεται και το objective “key1”.



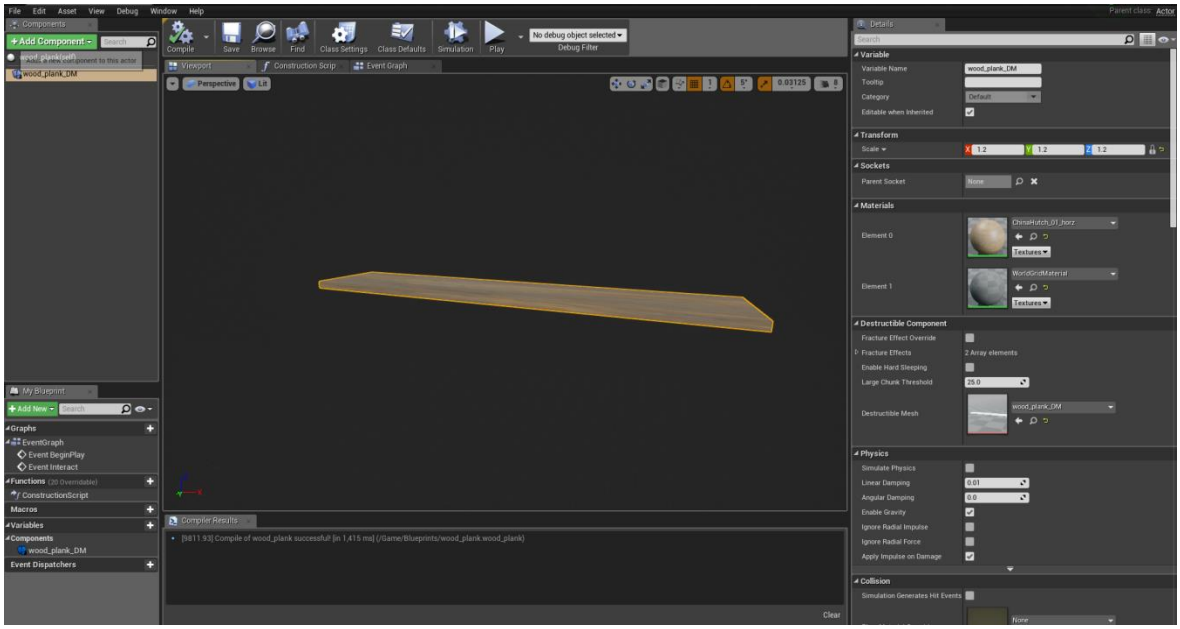
Εικόνα 60 "Boat_bp Blueprint"

4.7.4 wood_plank

Με τη ξύλινη σανίδα μπορεί ο χρήστης να αλληλοεπιδράσει και να την σπάσει. Για να είναι εφικτό αυτό έχουμε δημιουργήσει ένα destructible mesh από το “wood_plank” και έχουμε κάνει implement το interaction interface. Επομένως με την χρήση του *Event Interact* ενεργοποιούμε την λειτουργία Simulate physics έτσι ώστε κατά την καταστροφή τα κομμάτια να πέσουν στο έδαφος. Τέλος με την χρήση του κόμβου Apply Damage προκαλούμε την ζημιά στην σανίδα.



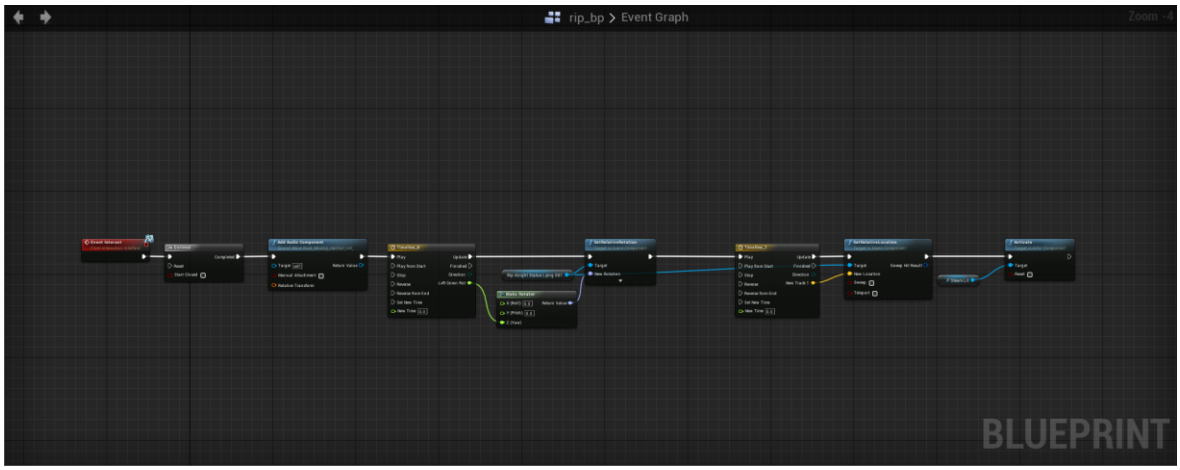
Εικόνα 61 "Wood plank Blueprint Nodes"



Εικόνα 62 "Wood Plank Blueprint Preview"

4.7.5 rip_bp

Το τελευταίο ουσιαστικά blueprint που συναντάει ο παίχτης στο παιχνίδι και αποτελείται από το τάφο που πρέπει να ανοίξει και να μπει μέσα. Αρχικά μέσα στις ρυθμίσεις Κλάσης “Class Settings” έχουμε κάνει Implement το Interaction Interface έτσι ώστε να μπορέσουμε να χρησιμοποιήσουμε το *Event Interact*. Υλοποιώντας το παραπάνω Event και με την χρήση του κόμβου DoOnce ο παίχτης πατώντας το “E” εκτελούνται τα παρακάτω μία μόνο φορά. Αναπαράγουμε τον ήχο “Rock_Moving_mp3cut_net_” με την χρήση του κόμβου Add Audio Component. Έπειτα έχουμε δύο Timeline με τα οποία αλλάζουμε την τιμή της περιστροφής στον άξονα Z και την τιμή της θέσης του “Rip Knight Statue Lying 001” ώστε να ανοίξει. Τέλος με την χρήση του κόμβου Activate ενεργοποιούμε τον ατμό “P Steam Lit”.



Εικόνα 63 "RIP Tomb Blueprint"

4.8 Components

4.8.1 Exponential Height Fog

Έχει περισσότερη πυκνότητα σε χαμηλές θέσεις ενός χάρτη και λιγότερη πυκνότητα σε ψηλά μέρη. Η μετάβαση είναι ομαλή, ώστε ποτέ να μην απότομη αλλαγή καθώς αυξάνεται το υψόμετρο. Το εκθετικό ύψος ομίχλης παρέχει επίσης δύο χρώματα ομίχλης, ένα για το ημισφαίριο που βλέπει το κυρίαρχο directional light (ή ευθεία επάνω αν δεν υπάρχει) και ένα άλλο χρώμα για το αντίθετο ημισφαίριο. Το συγκεκριμένο Component χρησιμοποιήθηκε για να προσθέσουμε ομίχλη στο σκηνικό μας. Κρατήθηκαν οι βασικές ρυθμίσεις, ενεργοποιήσαμε όμως την επιλογή Volumetric Fog ώστε να επιτύχουμε μεγαλύτερο ρεαλισμό σε σχέση με την ομίχλη και την πηγή φωτός του ήλιου. Στις τιμές Extinction Scale δώσαμε την τιμή 9.34 και View Distance την τιμή 1000.

4.8.2 Directional Light

Το συγκεκριμένο Component αποτελεί την πηγή φωτός μας. Έχουμε αλλάξει την τιμή του Intensity σε 4.5 όπως επίσης έχουμε ενεργοποιήσει τις επιλογές “Light Shaft Occlusion” και “Light Shaft Bloom” ώστε να δημιουργήσουμε το φαινόμενο των God Rays (θεϊκές ακτίνες) κάνοντας την εμπειρία του παιχνιδιού ατμοσφαιρική.

4.8.3 Sky Sphere Blueprint

Το συγκεκριμένο blueprint υπήρχε μέσα στο αρχικό πρότυπο και αποτελεί την σφαίρα του κόσμου μας. Οι αλλαγές που κάναμε σε σχέση με το αρχικό είναι η επιλογή του Directional Light Actor σε “DirectionalLight” από το 3.9.2, απενεργοποιήσαμε την επιλογή “Colors Determined by Sun’s Position”, αλλάξαμε τις τιμές “Sun Brightness” σε 5.0, “Cloud Speed” σε 5.0, “Cloud Opacity” σε 1.1 και “Stars Brightness” σε 0.1. Τέλος αλλάξαμε το χρώμα στις επιλογές “Zenith Color” και “Horizon Color”.

4.8.4 Post Process Volume

Τα Components τύπου Post Process Volume τα χρησιμοποιούμε για να αλλάξουμε τιμές όπως θερμοκρασία χρώματος, κορεσμός, αντίθεση, gamma κτλ. φτάνοντας έτσι στο επιθυμητό τελικό αποτέλεσμα. Είναι ένα πολύ σημαντικό κομμάτι όταν θέλουμε να επιτύχουμε ρεαλισμό και αποτελεί ισχυρό εργαλείο της μηχανή γραφικών. Στο παιχνίδι έχουμε χρησιμοποιήσει 2 τέτοια Components έχοντας το κάθε ένα τις ανάλογες ρυθμίσεις.

4.8.5 Ambient Sound Actor

Ο Ambient Sound Actor μπορεί να χρησιμοποιηθεί για πολλούς σκοπούς, όπως οι επαναλαμβανόμενοι ήχοι περιβάλλοντος καθώς και οι ήχοι χωρίς loop.

Γενικά, ο Ambient Sound Actor συμμορφώνεται με τον πραγματικό κόσμο, όπου όσο πιο κοντά στην πηγή ενός ήχου βρισκόμαστε, τόσο πιο δυνατά τον ακούμε. Συγκριτικά, ένας ήχος που είναι κανονικά δυνατός, μπορεί να ακούγεται πιο σιγά εάν είναι πιο μακριά.

Εάν ο Ambient Sound Actor έχει ρυθμιστεί σε Αυτόματη Ενεργοποίηση, θα αρχίσει αμέσως να αναπαράγεται μόλις δημιουργηθεί (στην αρχή του παιχνιδιού ή στην αναπαραγωγή), ακόμα και αν η συσκευή αναπαραγωγής δεν είναι σε θέση να την ακούσει.

Το στοιχείο ήχου που δείχνει το στοιχείο Ambient Sound Actor θα αναπαράγεται μόνο μία φορά ανά ενεργοποίηση, εκτός αν ορίζεται ως Looping στο Sound Wave ή ορίζεται ως μέρος ενός στοιχείου Sound Cue.

Εμείς έχουμε χρησιμοποιήσει 2 τέτοια Components, ένα για τον ήχο εντός της σπηλιάς και ένα για τον ήχο στο δάσος.

4.8.6 Decal Actor

Για την απεικόνιση του αίματος χρησιμοποιήσαμε το component Decal actor το οποίο προσφέρει καλύτερες επιδόσεις, συντήρηση αλλά και μεγαλύτερη ρεαλιστικότατα σε γωνίες και πολύπλοκα μοντέλα όπως ο άνθρωπος.

4.8.7 Έτοιμα Components

Παρακάτω παραθέτουμε μερικά πακέτα και μεμονωμένα asset τα οποία χρησιμοποιήσαμε και διατίθενται από την Epic Games ή ελεύθερα από διάφορους δημιουργούς προς χρήση.

4.8.7.1 Open World Demo Collection

Χορηγείται άδεια χρήσης μόνο για προϊόντα που βασίζονται σε UE4. Μια συλλογή ρεαλιστικών στοιχείων από το demo του Open World της Epic που παρουσιάστηκε στο GDC 2015.

Σχετικά με αυτό το πακέτο:

Κάθε στοιχείο έχει δημιουργηθεί χρησιμοποιώντας φωτογραφίες ενός αντικειμένου πραγματικού κόσμου.

Η Διαδικασία:

Για μεγαλύτερα στοιχεία όπως πετρώματα, έδαφος, γκρεμούς και δέντρα, η ανακατασκευή πραγματοποιήθηκε απευθείας από φωτογραφίες χρησιμοποιώντας μια διαδικασία που ονομάζεται φωτογραμμετρία. Αυτά τα περιουσιακά στοιχεία τέθηκαν στη συνέχεια μέσω μιας διαδικασίας που ονομάζεται "de-lighting" για να τα καταστήσουν κατάλληλα για χρήση σε οποιοδήποτε σενάριο φωτισμού. Specular και roughness maps, που δημιουργήθηκαν για να εκμεταλλευτούν πλήρως την αρχιτεκτονική του Physical Based Rendering της Unreal Engine 4. Τέλος, τα περιουσιακά στοιχεία που βελτιστοποιήθηκαν από το παιχνίδι

δημιουργήθηκαν με normal maps, LOD και πλέγματα σύγκρουσης (collision meshes).

Για τη βλάστηση, όπως λουλούδια, θάμνους και γρασίδι, χρησιμοποιήθηκε color calibrated reference photography που τραβήχτηκε στο πεδίο για ακριβέστερη μοντελοποίηση και υφή αυτών των στοιχείων.

Εικόνα 64 "Open World Demo Collection"

4.8.7.2 Landscape Mountains

Αυτό το χιονισμένο, ορεινό περιβάλλον δημιουργήθηκε για να παρουσιάσει τι μπορείτε να κάνετε με τα εργαλεία τοπίου landscape και φυλλώματος foliage στην Unreal Engine 4. Το έδαφος είναι περίπου 5χλμ έως 5χλμ σε μέγεθος, αποτελούμενο από ψηλές κορυφές, κοιλάδες, κατεψυγμένες λίμνες και δάση αποτελούμενα από χιλιάδες δέντρα.

Μέσα από αυτό το πακέτο χρησιμοποιήσαμε το Blueprint BP_Birds το οποίο αποτελεί ένα σμήνος από γλάρους όπως επίσης και SM_Path που το χρησιμοποιήσαμε για τον δρόμο με την χρήση του εργαλείου Splines.

Εικόνα 65 "Landscape Mountains"

4.8.7.3 Water Planes

Αυτή η συλλογή περιέχει επιφάνειες νερού και παραδείγματα από διαφορετικά water shaders και Blueprints για να τα επεξεργαστούμε σύμφωνα με τις ανάγκες μας. Οι επιφάνειες νερού σε αυτό το πακέτο καλύπτουν ένα εύρος από διαφορετικές περιπτώσεις εφαρμογής, από μεγάλης έκτασης επιφάνειες ωκεανού με κύματα και αφρό, έως διάφανες επιφάνειες μικρότερων περιοχών.

Εμείς χρησιμοποιήσαμε την επιφάνεια νερού BP_LakeWater για την δημιουργία της λίμνης που υπάρχει στον κόσμο μας.

Εικόνα 66 "Water Planes"

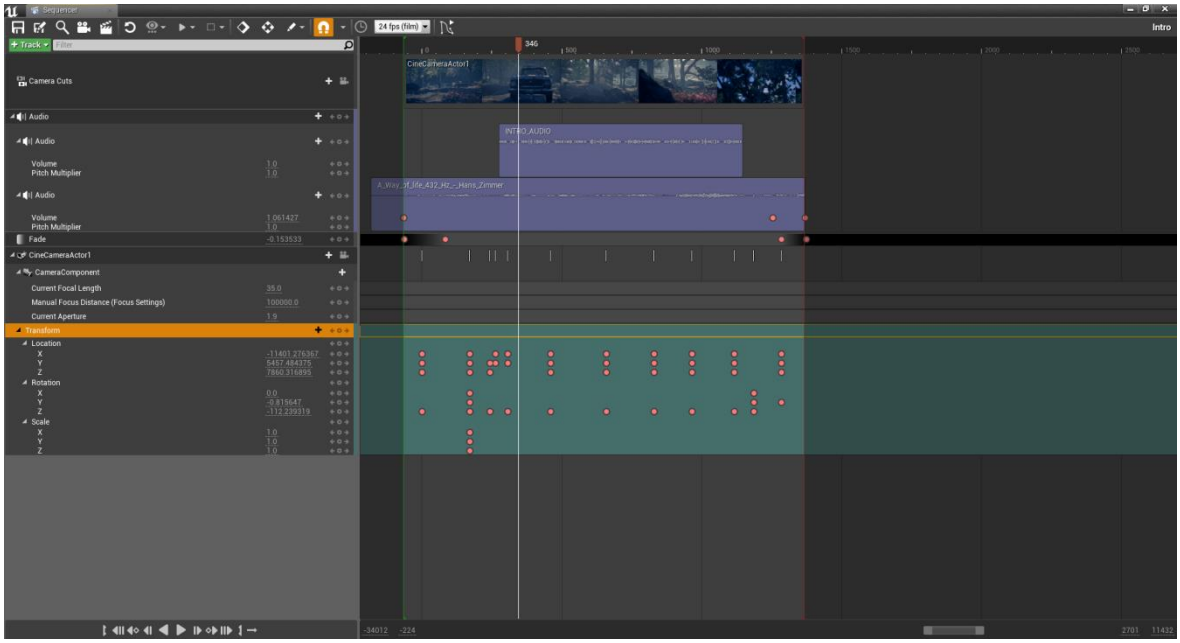
4.8.7.4 Luos's FREE Modular Caves/Rocks mini Package

Το συγκεκριμένο πακέτο αποτελεί μία δωρεάν έκδοση του “Luos's Modular Rocks & Caves” και περιέχει έτοιμα μοντέλα και materials τα οποία είναι modular και τα χρησιμοποιήσαμε για να συνθέσουμε την σπηλιά μας.

Εικόνα 67 "Luos Caves"

4.9 Cinematics

Σε αυτό το σημείο θα πούμε μερικά πράγματα για το πώς δημιουργήθηκε το εισαγωγικό βίντεο το οποίο αναπαράγεται σε πραγματικό χρόνο. Χρησιμοποιώντας την επιλογή Cinematics από τα το μενού εισάγουμε έναν νέο Level Sequence. Έπειτα εισάγουμε ένα Component τύπου Cine Camera Actor στον κόσμο μας και έπειτα εντός του Sequencer. Με την χρήση των Keyframes αλλάζουμε την θέση της κάμερας. Επίσης έχουμε εισάγει 2 στοιχεία ήχου (audio track), την φωνή του πρωταγωνιστή και background music όπως και ένα Fade track για να δημιουργήσουμε ομαλή εκκίνηση του intro και ομαλό κλείσιμο.



Εικόνα 68 "Intro Sequence"

4.10 Player Dialogues

Ο μονόλογος του παίχτη κατά την διάρκεια του παιχνιδιού έχει δημιουργηθεί μέσο της ιστοσελίδας http://www.fromtexttospeech.com/ η οποία μας δίνει την δυνατότητα να μετατρέψουμε κείμενο σε λόγια. Με την κατάλληλη μετατροπή του αρχείου σε format *.wav μπορούμε να εισάγουμε το αρχείο ήχου στην Unreal Engine 4.

ΕΠΙΛΟΓΟΣ

Κλείνοντας αυτό το κεφάλαιο έχουμε αναλύσει όλες τις πτυχές του παιχνιδιού έτσι ώστε κάποιος τελειώνοντας το να έχει πλήρη επίγνωση του τρόπου σκέψης και υλοποίησης δημιουργώντας παράλληλα και την επιθυμία για επιπλέον μελέτη και αναζήτηση.

Ξεκινήσαμε με την δημιουργία του project, φτιάξαμε το έδαφος την φύση, τα δέντρα, τον κεντρικό μας χαρακτήρα, το αρχικό βασικό μενού, τους στόχους που πρέπει να ολοκληρώσει ο παίχτης, τα βασικά στοιχεία που απαρτίζουν τον κόσμο, το σπίτι, η βάρκα, η σπηλιά όπως και όλα αυτά τα στοιχεία που αποτελούν το αλατοπίπερο στο παιχνίδι όπως ο φωτισμός η ομίχλη οι ακτίνες του ήλιου κτλ.

Η προσέγγιση όπως έχει αναφερθεί και στην αρχή του κεφαλαίου είναι από έξω προς τα μέσα αλλά πρέπει να γνωρίζουμε ότι ένα τέτοιο project δεν ακολουθεί συνεχώς σειριακά μοντέλα ανάπτυξης. Πολλές πτυχές του παιχνιδιού αναπτύσσονται παράλληλα και ο συνδυασμός τους μας δίνει ένα ολοκληρωμένο ατμοσφαιρικό, οπτικοακουστικό με πλοκή και νόημα αποτέλεσμα.

Σίγουρα υπάρχουν έννοιες μικρότερης σημασίας που δεν έχουν αναπτυχθεί όπως και μοντέλα ή τεχνικές στις οποίες δεν γίνεται κάποια αναφορά. Εάν γινόταν ανάλυση σε βάθος η συγκεκριμένη εργασία θα μπορούσε να γίνει ένα μικρό βιβλίο.

ΚΕΦΑΛΑΙΟ 5: ΣΥΜΠΕΡΑΣΜΑΤΑ

Συμπερασματικά, η UE4 αποτελεί ένα σύγχρονο εργαλείο ανάπτυξης παιχνιδιών, στο οποίο συνεχώς όλο και περισσότεροι προγραμματιστές επενδύουν. Η δωρεάν έκδοση που παρέχει η Epic Games παρέχει ένα αρκετά ικανοποιητικό αριθμό εργαλείων και δεν υστερεί σε τίποτα σε σχέση με άλλα παρόμοια προγράμματα. Τέλος η Unreal Engine 4 αποτελεί την πρώτη επιλογή για την ανάπτυξη 3D παιχνιδιών και καταλήγω στο συμπέρασμα αυτό, καθώς παρέχει το κατάλληλο περιβάλλον, τα εργαλεία αλλά και από το μεγάλο αριθμό βιβλιογραφίας και βίντεο εκμάθησης που κυκλοφορεί στο διαδίκτυο.

Η ανάπτυξη ενός παιχνιδιού τριών διαστάσεων απαιτεί πολύ χρόνο και πρωτοτυπία. Σε τεχνικό επίπεδο χωρίζεται σε διαφορετικά μέρη όπως ο σχεδιασμός των 3D μοντέλων, η δημιουργία σεναρίου, cinematic, material, Level Designer κτλ. Έτσι καταλαβαίνουμε ότι ο βαθμός δυσκολίας είναι υψηλός για μια μικρή ομάδα 2-3 ανθρώπων να υλοποιήσει ένα τέτοιο project. Τίποτα όμως δεν είναι αδύνατο αρκεί να υπάρχει μεθοδικότητα στο τρόπο εργασίας πάνω στο project. Υπάρχουν αρκετά έτοιμα στοιχεία από άλλους developers ανά τον κόσμο τα οποία μπορούμε να αποκτήσουμε με μικρό αντίτιμο και να γλιτώσουμε πολύ χρόνο και χρήματα.

Η συγκεκριμένη εργασία αποτελεί μια μικρογραφία ενός εμπορικού παιχνιδιού και σίγουρα επιδέχεται βελτιώσεων και επεκτάσεων. Καθώς ολοκληρώνεται το παιχνίδι ολοκληρώνεται ουσιαστικά το πρώτο κεφάλαιο της ιστορίας αφήνοντας τον παίχτη με την απορία του πως θα μπορούσε να είναι η συνέχεια. Θα λέγαμε ότι είναι demo από κάτι μεγαλύτερο. Επίσης λόγο της χρήσης του Interaction Interface θα μπορούσαμε να δώσουμε ζωή σε περισσότερα αντικείμενα όπως το ραδιόφωνο, η τηλεόραση, το τσεκούρι το οποίο θα μπορούσε να γίνεται και όπλο του παίχτη. Τεχνολογίες όπως η χρήση inventory ή Player Stats όπως ζωή και αντοχή είναι πράγματα που δεν θα έλειπαν από ένα ολοκληρωμένο, στο επίπεδο που απαιτεί η αγορά, παιχνίδι.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Brenden Sewell (2015), Blueprints Visual Scripting for Unreal Engine: Build professional 3D games with Unreal Engine 4's Visual Scripting system

Bjarne Stroustrup (2013), The C++ Programming Language, 4th Edition

Epic Games (2018) Epic Games Documentation https://docs.unrealengine.com

John O'Neill (2008),The Real Cost of Middleware, GameDaily Article.

João Paulo (2017) 3D Textures https://3dtextures.me/

Luos (2017) FREE Modular Caves/Rocks mini Package https://forums.unrealengine.com/community/community-content-tools-and-tutorials/35905-free-luos-s-free-modular-caves-rocks-mini-package

Nicola Valcasara (2015) Unreal Engine Game Development Blueprints

Ruslan Nazirov (2017) WorldMachine + UE4: Full Workflow https://80.lv/articles/worldmachine-ue4-full-workflow/

Steve Swink (2008) Game Feel: A Game Designer's Guide to Virtual Sensation (Morgan Kaufmann Game Design Books) 1st Edition

Various (2017) Free 3D Models https://free3d.com/

Various (2017) Free Game Files https://opengameart.org/

Various (2017) Free 3D Models https://archive3d.net/

Wikipedia (2017) Unreal Engine https://en.wikipedia.org/wiki/Unreal_Engine

Wikipedia (2017) Game Engine https://en.wikipedia.org/wiki/Game_engine