ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Η ΑΝΑΠΤΥΞΗ ΒΙΝΤΕΟΠΑΙΧΝΙΔΙΩΝ ΣΤΗΝ UNREAL ENGINE 5 ΚΑΙ Ο ΟΠΤΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ»

Του φοιτητή Κωνσταντίνου Κουμαράκη
Αρ. Μητρώου: 185395

Επιβλέπων
Χριστίνα Βολιώτη
Έκτακτο Εκπαιδευτικό Προσωπικό

Ημερομηνία 10/09/2023

Η Ανάπτυξη Βιντεοπαιχνιδιών στην Unreal Engine 5 και ο Οπτικός Προγραμματισμός

Κωδικός Π.Ε. 23107

Κωνσταντίνος Κουμαράκης

ΧΡΙΣΤΙΝΑ ΒΟΛΙΩΤΗ

Ημερομηνία ανάληψης Π.Ε. 14-02-2023

Ημερομηνία περάτωσης Π.Ε. 10-09-2023

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως πτυχιακή εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Η έγκριση της πτυχιακής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

«Για όσους πίστεψαν σε έμενα»

Πρόλογος

Επέλεξα αυτό το θέμα της εργασίας καθώς η δημιουργία βιντεοπαιχνιδιών πάντα ήταν στο μυαλό μου ως ένα μελλοντικό επάγγελμα. Από την ηλικία των δεκατεσσάρων, ασχολήθηκα με την Unreal Engine και πειραματιζόμουν με μικρά projects έως ότου έφτασα σε σημείο να μπορώ να δημιουργήσω μια ολοκληρωμένη ιστορία και να κυκλοφορεί στο διαδίκτυο, παίρνοντας αρκετά θετικά σχόλια από αυτούς που το δοκίμασαν.
Είναι κάτι το οποίο αγαπώ, και κάτι που πολλοί θα ήθελαν να μάθουν και να εξερευνήσουν. Η Unreal Engine 5 μου έδωσε ακόμα περισσότερο το έναυσμα να πραγματοποιήσω αυτή την πτυχιακή, με τις πολύ ενδιαφέρουσες νέες τεχνολογίες της.

Περίληψη

Η ανάπτυξη παιχνιδιών υφίσταται εδώ και αρκετά χρόνια, και από το πρώτο παιχνίδι που πρωτοεμφανίστηκε το 1958 μέχρι και σήμερα, αυτή η βιομηχανία εξελίσσεται ραγδαία. Έμφαση όμως δινόταν στη φύση του βιντεοπαιχνιδιού και όχι στον πυρήνα του, όπου είναι η μηχανή παιχνιδιών. Σήμερα υπάρχουν πολύ δυνατές μηχανές παιχνιδιών, όπου μια εξ αυτών είναι η Unreal Engine 5. Η Unreal Engine 5, μετά από πολλές εκδόσεις που κυκλοφόρησε η Epic Games, είναι πλέον μια πανίσχυρη μηχανή παιχνιδιών η οποία δίνει τη δυνατότητα στους προγραμματιστές να δημιουργούν εντυπωσιακά παιχνίδια. Και ενώ για πολλά χρόνια οι προγραμματιστές βασίζονταν στον παραδοσιακό προγραμματισμό και περιορίζονταν από άποψη εργαλείων, η Unreal Engine 5 εξελίσσει πλέον το οπτικό σύστημα προγραμματισμού του, ενώ παράλληλα παρουσιάζει δύο νέα εργαλεία τα οποία πρόκειται να βοηθήσουν σε μεγάλο βαθμό στην απόδοση και στην ρεαλιστικότητα του παιχνιδιού αντίστοιχα. Σε αυτή την πτυχιακή, θα αναλυθούν αυτά τα εργαλεία και ο οπτικός προγραμματισμός της Unreal Engine 5. Η μηχανή προσφέρει εξαιρετικά εργαλεία για τα physics και την απόδοση του παιχνιδιού ώστε να επιτρέπουν το ρεαλιστικό φωτισμό, τις σκιές και άλλα οπτικά εφέ τα οποία είναι απαραίτητα σε ένα παιχνίδι. Τέλος, Unreal Engine 5 είναι ένα πολύ ισχυρό εργαλείο που βοηθά τους ανθρώπους να δημιουργούν εντυπωσιακά βιντεοπαιχνίδια αλλά και αρχιτεκτονικές κατασκευές.

Λέξεις – Κλειδιά

unreal engine 5, οπτικός προγραμματισμός, blueprints, βιντεοπαιχνίδι

Video Game Development in Unreal Engine 5 and

Visual Programming

Konstantinos Koumarakis

Abstract

Game development exists for several years, and since the first game was released in 1958 until today, the industry has evolved rapidly. Emphasis was given on the nature of the game rather than its core, which is the game engine. Nowadays, there are very powerful game engines, where one of them is Unreal Engine 5. Unreal Engine 5, after several versions released by Epic Games, is now a powerful game engine that enables developers to create impressive games. And while for many years developers had relied on traditional programming and were limited in terms of tools, Unreal Engine 5 now evolves its visual programming system, while introducing two new tools that are going to greatly help with the performance and realism of the game respectively. In this thesis, these tools and the visual programming of Unreal Engine 5 will be analyzed. The game engine offers great tools for physics and game performance to allow realistic lighting, shadows and other visual effects which are essential in a game. Unreal Engine 5 is a very powerful tool that helps people to create great games and architectural designs.

Περιεχόμενα

Πρόλογος 5

Περίληψη 6

Abstract 7

Περιεχόμενα 8

Συντομογραφίες 11

Κεφάλαιο 1: Εισαγωγή 13

Κεφάλαιο 3: Οι λειτουργίες της Unreal Engine 17

3.1. Blueprints 18

3.2. Enums 20

3.3. Nanite 23

3.3.1. Η σχέση του Nanite Mesh με το Static Mesh 26

3.4. Real-time global illumination 26

3.4.1. Lumen 29

3.4.1.1. Voxelization (Voxel) 30

Κεφάλαιο 4: Υλοποίηση του project 33

4.1. Δημιουργία χαρακτήρα 33

4.1.1. Character Blueprint 33

4.1.2. Flashlight 34

4.1.3. Movement 35

4.1.4. Camera System 40

4.2. Δημιουργία περιβάλλοντος 41

4.2.1. Foliage 41

4.2.2. Blueprint ουρανού 43

4.2.3. Σπίτι 46

Κεφάλαιο 5: Συγκριτική μελέτη λειτουργιών της Unreal Engine 49

5.1. Σύγκριση με και χωρίς χρήση Nanite στην Unreal Engine 5 49

5.2. Χρήση του Lumen 52

5.3. Σύγκριση των Blueprints 55

Κεφάλαιο 6: Συμπεράσματα 57

ΒΙΒΛΙΟΓΡΑΦΙΑ 59

Κατάλογος Εικόνων / Σχημάτων

Εικόνα 3.1: Φωτορεαλιστική σκηνή 17
Εικόνα 3.2: Debugging εργαλεία στα blueprints 19
Εικόνα 3.3: Οπτικό σύστημα σεναρίων 19
Εικόνα 3.4: custom nodes 20
Εικόνα 3.5: Δημιουργία των enums 21
Εικόνα 3.6: Παράδειγμα ανακριβών τιμών 21
Εικόνα 3.7: Λειτουργία Enum με κώδικα 22
Εικόνα 3.8: Σχηματική αναπαράσταση του Cluster 23
Εικόνα 3.9: Σχηματική αναπαράσταση του Cluster 25
Εικόνα 3.10: Nanite, ανάλυση του περιβάλλοντος σε complexity view 25
Εικόνα 3.11: Η ενεργοποίηση του Nanite 26
Εικόνα 3.12: Direct Lighting 27
Εικόνα 3.13: Indirect Lighting 27
Εικόνα 3.14: Indirect Lighting με lightmass 27
Εικόνα 3.15: Volumetric Lighting 28
Εικόνα 3.16: Reflection στην Unreal Engine 5 28
Εικόνα 3.17: Η ανάλυση του φωτισμού μέχρι και το φωτορεαλιστικό 29
Εικόνα 3.18: Η ρεαλιστικότητα του Lumen 30
Εικόνα 3.19: Το Voxelization σε 4 διαφορετικά στάδια 30
Εικόνα 4.1: Όλα τα Functions από το Character Blueprint 34
Εικόνα 4.2: Τοποθέτηση του φακού στο χέρι του χαρακτήρα 34
Εικόνα 4.3: Τα blueprints για την λειτουργικότητα του φακού 35
Εικόνα 4.4: Τα functions του animation blueprint 36
Εικόνα 4.5: Transitions του χαρακτήρας 37
Εικόνα 4.6: Animation layers 38
Εικόνα 4.7: Υπολογισμός βάρους ποδιών 38
Εικόνα 4.8: Διασύνδεση λεκάνης με την τιμή-στόχο 39
Εικόνα 4.9: Locomotion state. Δίνει εντολή για το πια κίνηση ζητάει ο παίκτης 39
Εικόνα 4.10: Η κάμερα ενεργοποιεί το input 40
Εικόνα 4.11: Updater camera animation blueprint 41
Εικόνα 4.12: ρυθμίσεις foliage 42
Εικόνα 4.13: material blueprint (1) 42
Εικόνα 4.14 material blueprint (2) 43
Εικόνα 4.15: Ο ουρανός του περιβάλλοντος μας 44
Εικόνα 4.16: Πυκνότητα και ταχύτητα σύννεφων 44
Εικόνα 4.17: Κύκλος Ημέρας/Νύχτας 45
Εικόνα 4.18: Ηλιοβασίλεμα, ανατολή 45
Εικόνα 4.19: πανσέληνος 46
Εικόνα 4.20: το blueprint για τον τοίχο 47
Εικόνα 4.21: Enum switcher και branches 47
Εικόνα 4.22: Οι ρυθμίσεις για τον φωτισμό των παραθύρων ενώ ενσωματώνονται στον τοίχο 48
Εικόνα 5.1: Τα Vertices και τα Triangles της σφαίρας όπως δημιουργήθηκε στο Blender 49
Εικόνα 5.2: Τα FPS πριν την ενεργοποίηση του Nanite 50
Εικόνα 5.3: Η ενεργοποίηση του Nanite 50
Εικόνα 5.4: Μετά την ενεργοποίηση του Nanite 51

Εικόνα 5.5: Το Nanite ενεργοποιημένο..... 51
Εικόνα 5.6: Το Nanite απενεργοποιημένο..... 52
Εικόνα 5.7: Αυτό είναι το Nanite view σε Triangles 52
Εικόνα 5.8: Σκοτεινό μέρος με εξωτερικό φως από το παράθυρο στην Unreal Engine 5 53
Εικόνα 5.9: Σκοτεινό μέρος με εξωτερικό φως από το παράθυρο στην Unreal Engine 4 53
Εικόνα 5.10: Σκοτεινό μέρος με εξωτερικό φως από το παράθυρο από άλλη οπτική γωνία στην Unreal Engine 5..... 54
Εικόνα 5.11: Σκοτεινό μέρος με εξωτερικό φως από το παράθυρο από άλλη οπτική γωνία στην Unreal Engine 4..... 54
Εικόνα 5.12: Συγκριτικά οι χρόνοι φόρτωσης ανάμεσα στις δύο μηχανές 55
Εικόνα 5.13: Render data σε Unreal Engine 4 56
Εικόνα 5.14: Render data σε Unreal Engine 5 56

Συντομογραφίες

ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
Π.Ε.	Πτυχιακή Εργασία
LOD	detail level system
GPU	Graphics Processing Unit
CPU	Central Processing Unit
CGI	Computer-generated imagery
VAF	Vertex Attribute Fetch

Κεφάλαιο 1: Εισαγωγή

Σε μια εποχή που τα ψηφιακά μέσα εξελίσσονται συνεχώς, στα γραφικά έχει γίνει ένα μεγάλο βήμα όσον αφορά τον ρεαλισμό και την καινοτομία. Η Unreal Engine 5 είναι η πιο πρόσφατη μηχανή γραφικών, η οποία έρχεται με μια σημαντική αναβάθμιση σε υπάρχοντες μηχανισμούς αλλά μας παρουσιάζει και μερικούς νέους όπου θα συμβάλλουν εξίσου σημαντικά στην ανάπτυξη ενός παιχνιδιού.

Ο σκοπός αυτής της εργασίας είναι να εμβαθύνουμε περισσότερο στα τεχνικά χαρακτηριστικά της ίδιας της μηχανής παρά στο πως να δημιουργήσουμε ένα παιχνίδι. Και να διαφωτίσει τις θεμελιώδεις αλλαγές και ευκαιρίες που παρουσιάζονται από τις εξελίξεις της Unreal Engine 5.

Με αυτόν τον τρόπο θα γνωρίσουμε καλύτερα ορισμένα εργαλεία, όπως το Lumen, το οποίο είναι ένα σύστημα που πρόκειται να φέρει μεγάλες αλλαγές στα projects των developers όσον αφορά τον φωτισμό και το Nanite, που παρέχει η Unreal Engine 5, η οποία φέρνει την επανάσταση ουσιαστικά στην απόδοση των παιχνιδιών καθώς πλέον ορισμένα πράγματα τα οποία θα αναλυθούν παρακάτω, δεν χρειάζεται να τα υπολογίζουμε όπως κάναμε μέχρι τώρα.

Με αυτή την εργασία, δίνεται η δυνατότητα σε κάποιον να εμπλουτίσει και να εκσυγχρονίσει τις γνώσεις του αλλά και να απελευθερώσει πλήρως το δημιουργικό κομμάτι που πάντα περιοριζόταν από τις δυνατότητες του εκάστοτε συστήματος λόγω της απόδοσης του παιχνιδιού. Θα παρασχεθεί μια ολοκληρωμένη εικόνα σχετικά με την Unreal Engine 5 και το σύστημα οπτικού προγραμματισμού, τα blueprints και θα αποδείξουμε την εξέλιξη τους σε αντίθεση με τον παραδοσιακό κώδικα, την C++. Αυτό θα γίνει και μέσα από μια συγκριτική μελέτη η οποία θα παρουσιάσει με καλύτερο τρόπο τα οφέλη της νέας μηχανής και των συστημάτων της.

Η απόφαση να αναλάβω αυτή την εργασία απορρέει από μια περιέργεια για τη δύναμη αυτής της μηχανής και τις τεχνολογίες που περιλαμβάνει. Επιπρόσθετα ένας άλλος λόγος είναι να δοθεί η ευκαιρία σε κάποιον να μάθει την κάθε σημαντική λεπτομέρεια της μηχανής, που θα χρειαστεί για ένα παιχνίδι, ώστε στην συνέχεια να έχει ακόμα περισσότερες γνώσεις πάνω στο αντικείμενο και να μπορέσει να στηρίξει ένα project μεγάλης κλίμακας.

Μέσω αυτής της εργασίας, στοχεύουμε να αναλύσουμε την λειτουργία των νέων συστημάτων της Unreal Engine 5. Η εργασία αυτή ξεκινάει με μια σχετικά σύντομη ιστορική αναδρομή για το πως ξεκίνησε η Unreal Engine από την αρχή της δημιουργίας του μέχρι και το σήμερα (**Κεφάλαιο 2**). Έπειτα θα παρουσιαστούν ορισμένα, ζωτικής σημασίας, εργαλεία και λειτουργίες της Unreal Engine 5 όπως τα δύο παραπάνω που αναφέρθηκαν (**Κεφάλαιο 3**). Στη συνέχεια θα αναλυθεί το project που έχει αναπτυχθεί χρησιμοποιώντας τα εργαλεία και τις λειτουργίες που περιγράφηκαν στο προηγούμενο κεφάλαιο (**Κεφάλαιο 4**), ενώ μετά θα παρουσιαστεί μια συγκριτική μελέτη με απώτερο σκοπό την ανάδειξη της αποτελεσματικότητας των εργαλείων αυτών (**Κεφάλαιο 5**). Τέλος, η εργασία θα κλείσει με τα συμπεράσματα που καταγράφηκαν από τη συγκεκριμένη εργασία (**Κεφάλαιο 6**).

Καθώς η τεχνολογία συνεχίζει να διαμορφώνει τον κόσμο μας, η Unreal Engine 5 αναδύεται ως μια τρομερή δύναμη και η παρούσα εργασία αποτελεί απόδειξη της επιτακτικής ανάγκης της κατανόησης των δυνατοτήτων της να αναδιαμορφώσει το τοπίο της διαδραστικής ψυχαγωγίας και όχι μόνο.

Κεφάλαιο 2: Η ιστορία της Unreal Engine

Για την κατανόηση του πως φτάσαμε στο σήμερα με την Unreal Engine 5, παρακάτω θα δούμε λίγο την ιστορία της μηχανής παιχνιδιών, το πως ξεκίνησε, και τι εξέλιξη είχε ανά τα χρόνια.

Η Unreal Engine αρχικά αναπτύχθηκε από την Epic Games [2], μια εταιρεία που ξεκίνησε το 1991 στην Καλιφόρνια και το 1998 κατάφερε να δημιουργήσει μια μηχανή παιχνιδιών η οποία αρχικά ήταν ένα απλό shooter παιχνίδι, ή τουλάχιστον το brand της Unreal Engine ξεκίνησε έτσι. Η Unreal δημιουργήθηκε αρχικά από προγραμματιστές που ήταν διασκορπισμένοι σε όλο τον κόσμο, αλλά τελικά η ομάδα συγκεντρώθηκε για να ολοκληρώσει το παιχνίδι.

Η εταιρεία ξεκίνησε στον διαγωνισμό Make Something Unreal το 2004, με στόχο να επιβραβεύσει τους προγραμματιστές που δημιουργούν τροποποιήσεις (mods) βιντεοπαιχνιδιών χρησιμοποιώντας την Unreal Engine. Έπειτα επικεντρώθηκε στο να εξελίξει την Unreal ως μηχανή παιχνιδιών.

Η πρώτη έκδοση της μηχανής, η Unreal Engine 1 [3], χρησιμοποιήθηκε για να τροφοδοτήσει το παιχνίδι Unreal, το οποίο κυκλοφόρησε το 1998. Το παιχνίδι σημείωσε κριτική και εμπορική επιτυχία και η μηχανή κέρδισε γρήγορα την προσοχή για τα προηγμένα γραφικά και τα χαρακτηριστικά παιχνιδιού της.

Την επόμενη χρονιά η Epic Games παρουσίασε και κυκλοφόρησε το Unreal Tournament, το οποίο αναπτύχθηκε με την Unreal Engine 1.5, μια αναβαθμισμένη μηχανή παιχνιδιών από την πρώτη. Το παιχνίδι κατάφερε και αυτό να πετύχει τους εμπορικούς του στόχους και εδραίωσε περαιτέρω τη φήμη της μηχανής ως ένα ισχυρό και ευέλικτο εργαλείο για την ανάπτυξη παιχνιδιών.

Το 2002, κυκλοφόρησε η Unreal Engine 2, και αμέσως χρησιμοποιήθηκε από ομάδες προγραμματιστών όπως των America's Army και Tom Clancy's Rainbow Six 3. Η μηχανή είχε βελτιωμένα γραφικά και physics, με τους χαρακτήρες να είναι ακόμα πιο ρεαλιστικοί. Επίσης υποστήριζε αρκετές πλατφόρμες πέρα από το PC, όπως κονσόλες και κινητές συσκευές.

Το 2006, η Epic Games έκανε το μεγάλο βήμα με την Unreal Engine 3, η οποία χρησιμοποιήθηκε για τα Gears of War, BioShock και Mass Effect στα οποία έδωσε μια άλλη ματιά στην ανάπτυξη παιχνιδιών και έδειξε τον δρόμο για την εξέλιξη των γραφικών, με τα physics να είναι πιο ρεαλιστικά από ποτέ και τα animations. Όλοι οι υπολογισμοί του φωτισμού και της σκίασης γίνονταν ανά pixel, αντί ανά vertex. [4]

Η Unreal Engine 4 ήρθε το 2011 και χρησιμοποιήθηκε σε παιχνίδια όπως τα Fortnite, Street Fighter V και Player Unknown's Battlegrounds, φέρνοντας έναν νέο αέρα στην ανάπτυξη βιντεοπαιχνιδιών με την υποστήριξη της εικονικής πραγματικότητα (Virtual Reality – VR) και της επαυξημένης πραγματικότητας (Augmented Reality – AR), δύο είδη παιχνιδιών που τα επόμενα χρόνια αυξήθηκαν ραγδαία. Μάλιστα η τέταρτη αυτή έκδοση της σειράς μηχανών παιχνιδιών της Epic Games παρουσίασε για πρώτη φορά ένα ολοκληρωμένο σύστημα οπτικού προγραμματισμού, τα blueprints. Υποσχόταν την μείωση του παραδοσιακού προγραμματισμού και την πιο διασκεδαστική ανάπτυξη ενός παιχνιδιού μέσα από αυτό.[5]

Το 2020, η Unreal Engine 5 ήταν αυτή που έφερε επιτέλους στο προσκήνιο δύο από τα σημαντικότερα εργαλεία που μπορούσαν ποτέ να υπάρξουν, το Nanite και το Lumen [6]. Όπου το πρώτο αφορά ένα νέο virtual geometry system, το οποίο συμβάλει στην πολύ καλύτερη απόδοση του παιχνιδιού χωρίς να χάνεται η ποιότητα του, ενώ το Lumen αφορά τον ρεαλιστικό φωτισμό και τις σκιάσεις. Η μηχανή περιλαμβάνει επίσης βελτιώσεις στη ροή εργασίας, τα εργαλεία και τις δυνατότητες συνεργασίας [7].

Πέρα από τα παιχνίδια η Unreal Engine έχει την δυνατότητα να χρησιμοποιηθεί και σε διάφορες εφαρμογές, όπως αρχιτεκτονικές απεικονίσεις [8] και διαδραστικές εγκαταστάσεις. Οι ανεπτυγμένες δυνατότητες των physics και των γραφικών αυτής της μηχανής την καθιστούν κατάλληλη για τέτοιου είδους εφαρμογές και μάλιστα ξεπερνάει πολλές φορές και προγράμματα όπως το Cinema 4D ή το 3DS Max.

Η Unreal Engine έχει επίσης χρησιμοποιηθεί πολλές φορές και από παραγωγές του κινηματογράφου και της τηλεόρασης, ώστε να απεικονίσουν τα σκηνικά τους μέσω του CGI (Computer-generated imagery). Οι δυνατότητες απόδοσης σε πραγματικό χρόνο και η υποστήριξη εικονικών καμερών της μηχανής την έχουν καταστήσει δημοφιλή επιλογή για τη δημιουργία οπτικών εφέ.

Μάλιστα, η Unreal Engine έχει γίνει αρκετά γνωστή πλέον και για εφαρμογές και παιχνίδια που υποστηρίζουν το AR αλλά και από το VR. [9]

Ένα από τα σημαντικότερα πλεονεκτήματα της Unreal Engine είναι η ευελιξία και η επεκτασιμότητάα της, η οποία επιτρέπει στους προγραμματιστές να δημιουργούν παιχνίδια όλων των μεγεθών, από μικρά indie παιχνίδια έως και μεγάλους AAA τίτλους. Η μηχανή έχει αρκετούς πόρους ώστε να δώσει αυτή την δυνατότητα στους προγραμματιστές, με τα αρκετά εργαλεία που διαθέτει και τις λειτουργίες της.

Τα τελευταία χρόνια, η Epic Games, η εταιρεία που βρίσκεται πίσω από την Unreal Engine, έχει επίσης επικεντρωθεί στο να κάνει προσβάσιμη την Unreal Engine σε περισσότερους προγραμματιστές παρέχοντας δωρεάν την ίδια την μηχανή αλλά και με ένα μεγάλο marketplace από assets όπου μπορούν να αγοράσουν και να ξεκινήσουν τα project τους με την βοήθεια αυτών.

Συνοψίζοντας, η ιστορία της Unreal Engine ξεκινάει από το μακρινό 1998 και καταλήγει στο σήμερα με μια μεγάλη εξέλιξη και με πολλές καινοτομίες στον χώρο ανάπτυξης παιχνιδιών και εφαρμογών. Έχει μια ισχυρή κοινότητα από φιλόδοξους προγραμματιστές που αλληλοβοηθάνε και στηρίζονται ώστε να λύσουν απορίες, να θέσουν τα πιθανά προβλήματα και να συνεργαστούν.

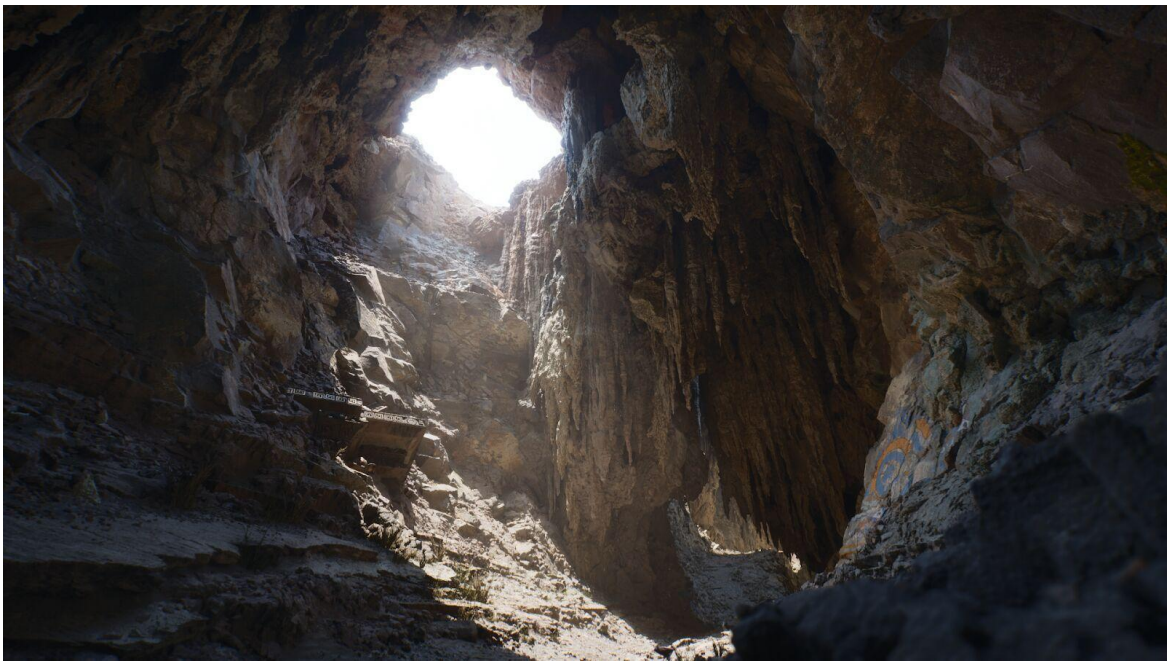
Κεφάλαιο 3: Οι λειτουργίες της Unreal Engine

Η Unreal Engine 5 διαθέτει αμέτρητα χαρακτηριστικά που δεν είναι εφικτό να αναφερθούν όλα εδώ, παρόλα αυτά θα υπάρξει εστίαση στα πιο σημαντικά από αυτά.

Τα **Blueprints** είναι ο οπτικός προγραμματισμός της μηχανής παιχνιδιών και επιτρέπει τη δημιουργία ορισμένων interactions μέσα στα παιχνίδια, που είναι απαραίτητα, χωρίς την ανάγκη του παραδοσιακού κώδικα, δηλαδή της C++. Ο **Level editor** [16] από την άλλη, τους βοηθάει στην μορφοποίηση του εικονικού περιβάλλοντος και τον χειρισμό των αντικειμένων μέσα σε αυτό. Φυσικά, πάντα δίνεται η δυνατότητα στον προγραμματιστή να κάνει μετατροπή των blueprints σε C++.

Το **Nanite** [11] είναι ένα virtual geometry system, μια τεχνολογία της Unreal Engine 5 που επιτρέπει τη δημιουργία ιδιαίτερα λεπτομερών μοντέλων με ελάχιστο βάρος στην μνήμη.

Ένα από τα πολλά χαρακτηριστικά που διαθέτει η Unreal Engine είναι ο παγκόσμιος φωτισμός σε πραγματικό χρόνο (**Real-Time Global Illumination**). Το σύστημα αυτό [10] εξασφαλίζει ότι τα αντικείμενα σε μια σκηνή φωτίζονται με ακρίβεια βάσει της αλληλεπίδρασης του φωτός με τις επιφάνειες. Επίσης το **Dynamic Global Lighting** όπου η Epic Games το ονομάζει Lumen, είναι ένα νέο σύστημα φωτισμού που παρέχει δυναμικό φωτισμό στη σκηνή ενός παιχνιδιού. Αυτό σημαίνει ότι ο φωτισμός μπορεί να αλλάζει σε πραγματικό χρόνο καθώς τα αντικείμενα στη σκηνή κινούνται.



Εικόνα 3.1: Φωτορεαλιστική σκηνή

Επίσης υπάρχουν μερικές εξίσου σημαντικές λειτουργίες που είναι εκτός του ερευνητικού πεδίου της εργασίας και είναι άξιες αναφοράς, όπως στο κομμάτι του ήχου όπου η Unreal Engine παρέχει τους προγραμματιστές και τους sound designers το **MetaSounds** [12], το οποίο είναι ένα εργαλείο όπου δίνει τον πλήρη έλεγχο σε ένα γράφημα ψηφιακής επεξεργασίας σήματος [13] για τη δημιουργία ηχητικών εμπειριών σε παιχνίδια και όχι μόνο. Ενώ το **δυναμικό σύστημα καιρού** επιτρέπει τη δημιουργία ρεαλιστικών εφέ, όπως την βροχή, το χιόνι, την ομίχλη και άλλα, Από την άλλη το **φωτορεαλιστικό rendering** των χαρακτήρων γίνεται με διάφορα εργαλεία και τεχνικές όπου συμβάλλουν στην εξαιρετική λεπτομέρεια αυτών. Το σύστημα **Niagara VFX** [14] είναι ένα σύστημα

για τα particle effects, και δίνει την δυνατότητα δημιουργίας σύνθετων visual effects, όπως εκρήξεις, φωτιά και καπνό.

Η Unreal Engine έχει επίσης **Physics Simulation** [15] όπου αποδίδει ρεαλιστικότητα στην σκηνή, δημιουργείται η φυσική των αντικειμένων μέσα σε αυτή, όπως για παράδειγμα η σύγκρουση δύο αντικειμένων ή η πτώση ενός από αυτών. Πέρα από αυτό, το physics simulation αφορά και τα animations των χαρακτήρων όπου τους δίνει πραγματικό βάρος και έχουν τις ανάλογες αντιδράσεις, όπως πτώση, περπάτημα στο έδαφος, πήδημα μερικών εκατοστών και άλλα. Επειδή όμως κάποια από αυτά χρειάζονται περισσότερη εξήγηση ή κάποια δεν αναφέρθηκαν, παρακάτω θα αναλυθούν λειτουργίες, όπως τα Enums που συνδέονται στενά με τα blueprints.

3.1. Blueprints

Στην αρχή τα blueprints χρησιμοποιήθηκαν ως ένα pre-production εργαλείο, όπου οι developers απλώς έκαναν σχέδια για το παιχνίδι και μετέπειτα τα μετέφεραν στην παραγωγή σε C++. Πλέον αυτό δεν ισχύει καθώς τα blueprints πλησιάζουν στο να αντικαταστήσουν πλήρως την γλώσσα προγραμματισμού δίχως καμία απώλεια, κάτι το οποίο θα συζητηθεί και παρακάτω, στην σύγκριση της προηγούμενης έκδοσης (Unreal Engine 4) με την τρέχουσα (Unreal Engine 5).

Τα Blueprints [17] αφορούν τον οπτικό προγραμματισμό της Unreal Engine 5 τα οποία επιτρέπουν στους developers να δημιουργούν και να χειρίζονται assets του παιχνιδιού χωρίς την ανάγκη γραφής παραδοσιακού κώδικα, δηλαδή C++. Τα Blueprints είναι ουσιαστικά ότι είναι και η C++ απλά γίνονται όλα οπτικά και έχουν εξίσου δυνατότητες για να δημιουργήσουν οι developers σύνθετους μηχανισμούς για το παιχνίδι, το πως θα συμπεριφέρεται η AI, για παράδειγμα οι NPCs πως θα κινούνται μέσα στο περιβάλλον, και άλλα συστήματα του παιχνιδιού συνδέοντας μεταξύ τους «κουτάκια», τα οποία ονομάζονται nodes, σε μια οπτική δομή που μοιάζει με διάγραμμα ροής, και μοιάζει σαν ένα προσχέδιο (blueprint).

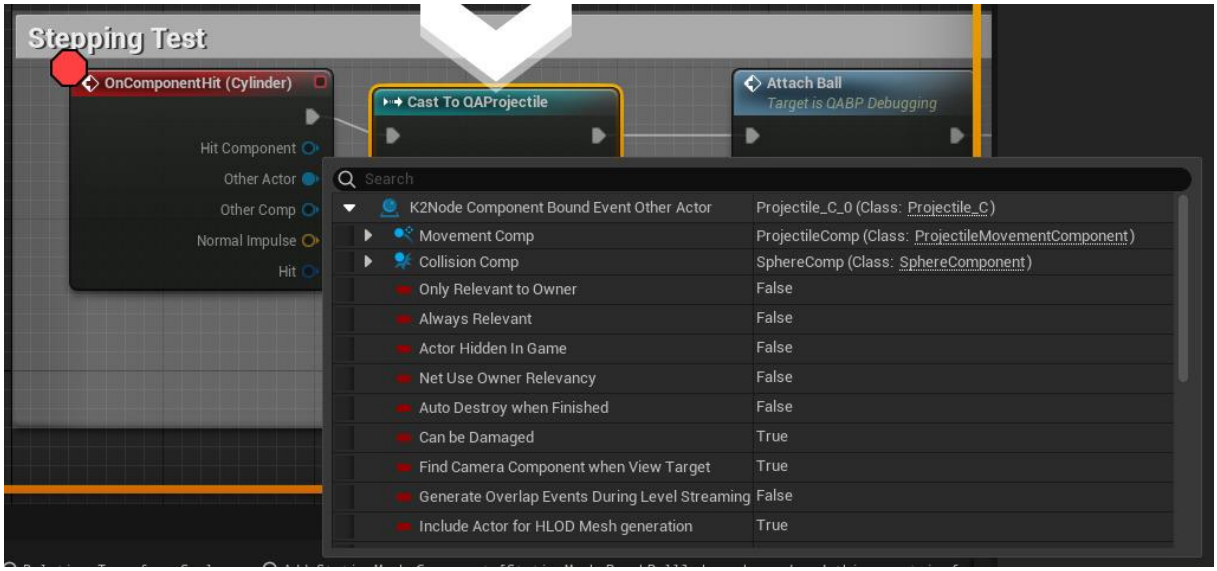
Αυτό που καθιστά την χρήση των Blueprints ελκυστική είναι το drag-and-drop, το οποίο κάνει πιο εύκολη τη χρήση τους από τους developers, και καθιστά δυνατή την δημιουργία παιχνιδιών ακόμα και από αρχάριους. Η γρήγορη δημιουργία των materials [18], των assets αλλά και των μηχανισμών του παιχνιδιού είναι ένα κύριο ζήτημα για τους developers και με το drag-and-drop των nodes, χωρίς να χρειαστεί να γράψουν κώδικα, δίνεται μια καλή λύση σε αυτό.

Στην συνέχεια έχουμε την δυνατότητα πρόσβασης και χειρισμού των ενσωματωμένων λειτουργιών της μηχανής, όπως τα physics, όπου όπως αναφέρθηκε και στο προηγούμενο κεφάλαιο, αφορά την προσομοίωση της συμπεριφοράς των αντικειμένων στον κόσμο του παιχνιδιού, συμπεριλαμβανομένου του collision [19]. Το collision είναι η ανίχνευση σύγκρουσης δύο αντικειμένων και θέτει τα όρια αυτών ως προς την σύγκρουση, τα rigid body dynamics, όπου είναι το RigidBody Animation Blueprint [20], στο οποίο μπορεί να χρησιμοποιηθεί το physics asset ενός χαρακτήρα, για να εκτελεστεί μια κίνηση στο σώμα του χαρακτήρα, και πολλά άλλα.

Τα Blueprints έρχονται με αρκετά pre-built nodes[21] τα οποία είναι αρκετά χρήσιμα στο να χρησιμοποιηθούν σε complex μηχανισμούς όπου δύο blueprints αλληλοσυνδέονται. Για παράδειγμα, η κίνηση των χαρακτήρων, οι αλληλεπιδράσεις που βασίζονται στα physics και το πως συμπεριφέρεται η AI είναι ένα αποτέλεσμα, τριών blueprints που τροφοδοτούν το ένα το άλλο με κανόνες για να δώσει το αποτέλεσμα που επιθυμούμε. Με τα παραπάνω, οι developers διευκολύνονται να δημιουργήσουν τα παιχνίδια καθώς ο κώδικας είναι μηδαμινός και η διαχείριση,

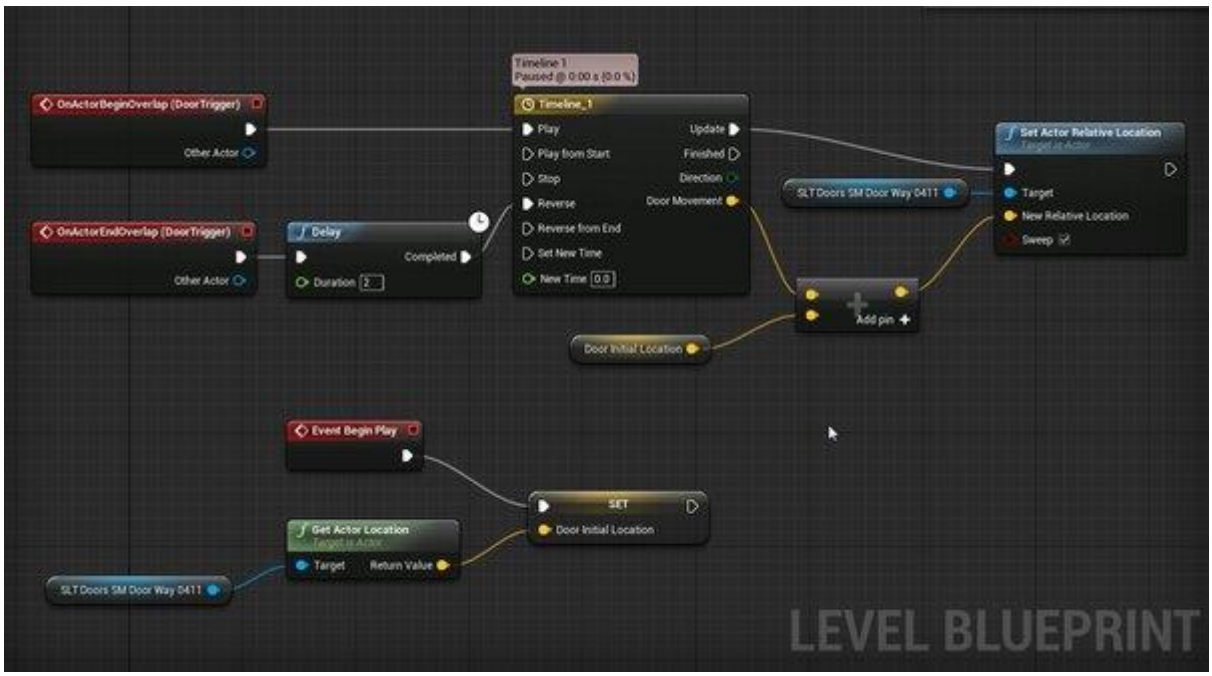
καθώς όλα είναι οπτικά, είναι πολύ καλύτερη, συνεπώς έχουν και γρηγορότερα ένα ολοκληρωμένο παιχνίδι.

Το debugging [22] είναι ένα σημαντικό μέρος της ανάπτυξης ενός παιχνιδιού καθώς με αυτό εντοπίζονται πιθανά σφάλματα στο παιχνίδι και τους μηχανισμούς του, βλέποντας την ροή εκτέλεσης των blueprints σε πραγματικό χρόνο. Έπειτα, μετά την ολοκλήρωση εντοπισμού, κατατάσσει τα σφάλματα σε κατηγορίες και με αυτό το τρόπο ο developer γνωρίζει τι χρειάζεται να διορθώσει και σε πιο ακριβώς σημείο.



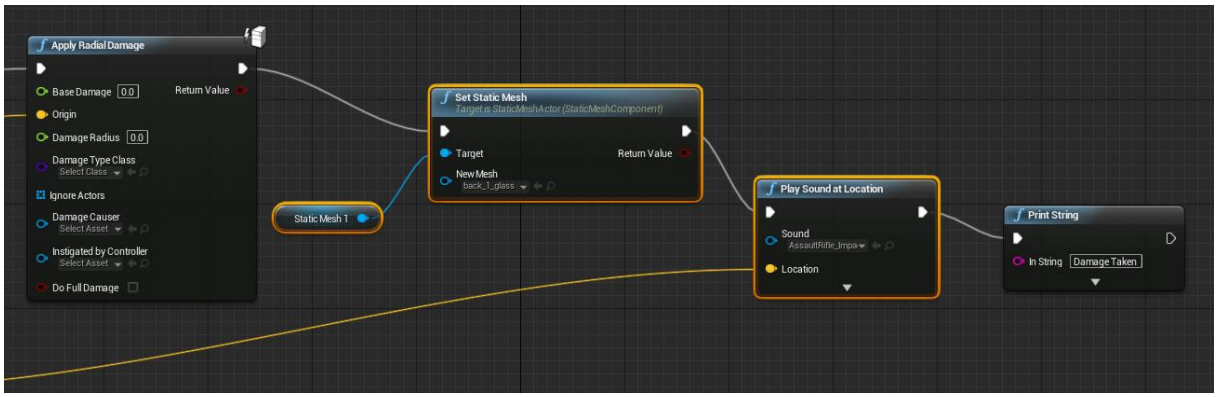
Εικόνα 3.2: Debugging εργαλεία στα blueprints

Τα Blueprints έχουν ένα θετικό, το ότι μπορούν να δημιουργήσουν ένα πρωτότυπο και να το μεταφέρουν ολόκληρο ή ένα τμήμα του, σε ένα άλλο blueprint απλά με ένα copy-paste. Οι developers έτσι, μπορούν εύκολα να δοκιμάσουν διαφορετικούς μηχανισμούς και να κάνουν αλλαγές χωρίς να χρειάζεται να γράψουν εκ νέου κώδικα, εξοικονομώντας πολύ χρόνο και προσπάθεια στη διαδικασία ανάπτυξης.



Εικόνα 3.3: Οπτικό σύστημα σεναρίων

Η δημιουργία custom nodes [23] μέσα στα blueprints είναι επίσης εφικτή, δίνοντας την δυνατότητα να χρησιμοποιηθούν για την επέκταση της λειτουργικότητας της μηχανής ή και τη δημιουργία ιδιαίτερων μηχανισμών για το παιχνίδι.



Εικόνα 3.4: custom nodes

Τα Blueprints της Unreal Engine 5 διαθέτουν επίσης ένα embedded network node για τη λειτουργικότητα του Multiplayer [24], το οποίο καθιστά πολύ πιο εύκολη την δημιουργία multiplayer συστήματος για ένα παιχνίδι.

Τα Blueprints γίνονται επίσης όλο και πιο διαδεδομένα στο χώρο της εικονικής πραγματικότητας (VR) και της επαυξημένης πραγματικότητας (AR), με αποτέλεσμα οι developers να δημιουργούν διαδραστικές εμπειρίες ακόμα πιο γρήγορα και εύκολα με όλες τις δυνατότητες που τους παρέχει η Unreal Engine. Τα blueprints στην Unreal Engine 5 θα φανούν πολύ χρήσιμα και στην δημιουργία εφαρμογών που δεν αφορούν τα παιχνίδια, όπως προσομοιώσεις εκπαίδευσης, αρχιτεκτονικές απεικονίσεις και διαδραστικές εγκαταστάσεις. Σύντομα θα δούμε πιο προηγμένες δυνατότητες τεχνητής νοημοσύνης και μηχανικής μάθησης να ενσωματώνονται σε οπτικά συστήματα όπως τα Blueprints, με τα οποία θα καταστήσει δυνατό να εξελιχθούν οι NPC χαρακτήρες που ελέγχονται από την AI, καθώς και πιο προηγμένων δυνατοτήτων στα physics και την απόδοση.

3.2. Enums

Τα enums [25], χρησιμοποιούνται για να δώσουν ονόματα σε ακέραιες τιμές. Για παράδειγμα, η κατάσταση ενός βασικού μενού μπορεί να αναπαρασταθεί ως ονόματα και να προγραμματιστεί ως ακέραιοι αριθμοί:

0 = Mainmenu
1 = Game
2 = Pause
3 = Gameover

Οι developers μπορούν με αυτά να δημιουργήσουν ένα σύνολο διακριτών επιλογών και να αντιστοιχίσουν μια μοναδική ακέραια τιμή σε κάθε επιλογή όπως βλέπουμε και στο παραπάνω παράδειγμα, κάνοντας πιο εύκολη την αναφορά στις επιλογές.

Τα Enums χρησιμοποιούνται συνήθως για τον ορισμό ενός εύρους τιμών που χρησιμοποιούνται ως είσοδοι ή έξοδοι για συναρτήσεις ή ως παράμετροι για διάφορα χαρακτηριστικά της μηχανής. Για παράδειγμα, μπορούν να χρησιμοποιηθούν για τον καθορισμό του τύπου των αντικειμένων σε μια σκηνή, των διαφορετικών καταστάσεων ενός χαρακτήρα ή των διαφόρων ηχητικών εφέ σε ένα παιχνίδι. Ορίζοντας αυτές τις επιλογές ως enums, γίνεται να διασφαλίσουν ότι ο κώδικάς τους είναι

εύκολα αναγνώσιμος και συντηρήσιμος και ότι οι τιμές που χρησιμοποιούνται στον κώδικα είναι συνεπείς και σωστές.

Με την λέξη-κλειδί enum γίνεται να δημιουργήσουν ένα enum, και στην συνέχεια να γράψουν ένα όνομα γι' αυτό και μια λίστα με ονομαστικές τιμές, κάθε μία από τις οποίες χωρίζεται με κόμμα. Στις ονομαστικές τιμές ανατίθεται μια ακέραια τιμή από προεπιλογή, ξεκινώντας από το 0, αλλά οι τιμές αυτές μπορούν επίσης να οριστούν από τον developer, αν αυτό είναι επιθυμητό. Αφού οριστούν, τα enums μπορούν να χρησιμοποιηθούν με τον ίδιο τρόπο όπως οποιοσδήποτε άλλος τύπος δεδομένων, και οι ονομαστικές τιμές μπορούν να αναφέρονται με τα ονόματα τους ή με τις ακέραιες τιμές που τους έχουν εκχωρηθεί.

Εικόνα 3.5: Δημιουργία των enums

Έχουν πολλά πλεονεκτήματα έναντι άλλων τύπων δεδομένων, όπως βελτιωμένη αναγνωσιμότητα και συντηρησιμότητα του κώδικα, λιγότερα σφάλματα και σε ορισμένες περιπτώσεις καλύτερες επιδόσεις. Για παράδειγμα, όταν χρησιμοποιούνται τα enums, η μηχανή μπορεί να χρησιμοποιήσει δείκτες για να αναζητήσει γρήγορα μια ονομαστική τιμή, πράγμα που είναι ταχύτερο από την αναζήτηση μιας συγκεκριμένης τιμής σε μια λίστα ή έναν πίνακα. Φυσικά, τα enums επιβάλλουν μόνο συγκεκριμένα ονόματα και βοηθούν στην αποφυγή της χρήσης ανακριβών τιμών, οι οποίες μπορεί να οδηγήσουν σε σφάλματα και άλλα προβλήματα.

```
UENUM(BlueprintType)
enum class EDayOfWeek : uint8
{
    Monday = 0,
    Tuesday = 1,
    Wednesday = 2,
    Thursday = 3,
    Friday = 4,
    Saturday = 5,
    Sunday = 6,
    InvalidDay = 7 // Inaccurate value
};
```

Εικόνα 3.6: Παράδειγμα ανακριβών τιμών

Στο παραπάνω παράδειγμα, το enum «EdayOfWeek» έχει σχεδιαστεί για να αντιπροσωπεύει την ημέρα της εβδομάδας. Ωστόσο, περιέχει τη λανθασμένη τιμή «InvalidDay», η οποία δεν αντιστοιχεί

στην πραγματική ημέρα της εβδομάδας. Αυτό σημαίνει ότι η λανθασμένη χρήση αυτής της τιμής σε συγκεκριμένη λογική μπορεί να προκαλέσει σύγχυση στον κώδικά. Αυτό είναι μια ανακριβής τιμή.

Τα enums στον οπτικό προγραμματισμό μπορούν να κάνουν πολύ εύκολη την ζωή του προγραμματιστή, του σχεδιαστή και γενικότερα της ομάδας ανάπτυξης ενός παιχνιδιού καθώς μέσα από μερικά κουτάκια μπορούν να δημιουργήσουν τις διάφορες λειτουργίες του παιχνιδιού. Αντιθέτως στην Εικόνα 3.7 βλέπουμε πόσες γραμμές κώδικα χρειάζεται να γράψει ο developer προκειμένου να δημιουργήσει μια λειτουργία με enum όπου θα περιλαμβάνει τρεις επιλογές.

```
// First, we declare the enum:
UENUM(BlueprintType)
enum class EMyEnum : uint8
{
    OptionA,
    OptionB,
    OptionC
};

// Then, we can use it as a type for a variable in a class:
UCLASS()
class MYPROJECT_API AMyActor : public AActor
{
    GENERATED_BODY()

public:
    UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "My Enum")
    EMyEnum MyEnumValue;
};

// Finally, we can use it in Blueprint graphs or in C++ code to control logic or behavior:
void AMyActor::MyFunction()
{
    if (MyEnumValue == EMyEnum::OptionA)
    {
        // Do something if MyEnumValue is set to OptionA
    }
    else if (MyEnumValue == EMyEnum::OptionB)
    {
        // Do something if MyEnumValue is set to OptionB
    }
    else
    {
        // Do something if MyEnumValue is set to OptionC
    }
}
```

Εικόνα 3.7: Λειτουργία Enum με κώδικα

Τα enums μπορούν να χρησιμοποιηθούν για τον ορισμό των διαφορετικών επιπέδων σε ένα παιχνίδι και τα blueprints μπορούν να χρησιμοποιηθούν για τη δημιουργία λογικής για το πώς συμπεριφέρεται το παιχνίδι με βάση το τρέχον επίπεδο. Με τη χρήση των enums σε συνδυασμό με τα blueprints, οι developers μπορούν να δημιουργήσουν συστήματα που μπορούν εύκολα να τροποποιηθούν από άλλα μέλη της ομάδας ανάπτυξης, χωρίς να απαιτούνται αλλαγές στον κώδικα από πίσω.

Τα custom events [26] είναι επίσης μέρος των enums και μπορούν να ενεργοποιηθούν στη μηχανή, επιτρέποντας τη δημιουργία πιο σύνθετων συστημάτων. Για παράδειγμα, τα enums μπορούν να χρησιμοποιηθούν για τον ορισμό διαφορετικών events που συμβαίνουν κατά τη διάρκεια ενός παιχνιδιού, όπως ο θάνατος του παίκτη, η επίθεση του εχθρού ή η παραλαβή αντικειμένων. Τα σχέδια μπορούν να χρησιμοποιηθούν για τη δημιουργία λογικής, για τον τρόπο με τον οποίο το παιχνίδι αντιδρά σε αυτά τα συμβάντα.

3.3. Nanite

Το Nanite [35], είναι το νέο χαρακτηριστικό της Unreal Engine 5 και αυτό που κάνει είναι να εμφανίζει με λεπτομερή και υψηλή ανάλυση τα assets, και αυτό με ελάχιστο αντίκτυπο στην απόδοση. Ο παραδοσιακός τρόπος δημιουργίας assets, μέχρι στιγμής, είναι η χρήση μιας διαδικασίας που ονομάζεται "μείωση του αριθμού των πολυγώνων" [27], η οποία μειώνει τον αριθμό των πολυγώνων σε ένα μοντέλο προκειμένου να βελτιωθεί η απόδοση, με αντίκτυπο την ποιότητα και την ανάλυση αυτού του asset.

Εικόνα 3.8: Σχηματική αναπαράσταση του Cluster

Τα τελευταία δέκα χρόνια, η ανάπτυξη των μεγάλων παιχνιδιών τείνει σταδιακά σε δύο βασικά σημεία: το interactive storytelling και το open-world [28]. Για να είναι ρεαλιστικές οι σκηνές, το μοντέλο του χαρακτήρα πρέπει να είναι άψογα σχεδιασμένο, το μέγεθος του κόσμου που επεξεργαζόμαστε και ο αριθμός των αντικειμένων αυξάνονται πλέον κατά πολύ, και αυτά τα δύο έχουν αυξήσει σημαντικά την πολυπλοκότητα σε μια σκηνή του παιχνιδιού. Ο αριθμός των αντικειμένων πρέπει να είναι μεγάλος και κάθε μοντέλο πρέπει να είναι επαρκώς λεπτομερές.

Υπάρχουν συνήθως δύο σημεία που δημιουργείται ζήτημα στην απόδοση πολύπλοκων σκηνών:

- **Περιορισμοί στην GPU (Graphics Processing Unit)**: Η GPU είναι υπεύθυνη για την απόδοση των στοιχείων σε πραγματικό χρόνο και σε τέτοιες περιπτώσεις δυσκολεύονται να διατηρήσουν σταθερό frame-rate, όπου frame-rate είναι ο αριθμός των εικόνων (καρέ) που μας δίνει η οθόνη ανά δευτερόλεπτο. Αυτό μπορεί να οδηγήσει σε προβλήματα, όπως πτώσεις του frame-rate, stuttering, όπου είναι ένα πρόβλημα που προκαλείται από ακανόνιστες καθυστερήσεις μεταξύ της μονάδας επεξεργασίας γραφικών (GPU) και των καρέ στην οθόνη σας, καθώς και πολλά άλλα.
- **Bottlenecks στην CPU (Central Processing Unit) και στην μνήμη**: Η CPU διαχειρίζεται διάφορες άλλες πτυχές του παιχνιδιού, συμπεριλαμβανομένων των physics simulation, της συμπεριφοράς της τεχνητής νοημοσύνης και της λογικής του παιχνιδιού. Σε πολύπλοκες σκηνές, οι περιορισμοί της CPU και της μνήμης μπορεί να δημιουργήσουν ζήτημα, όπως αργούς χρόνους φόρτωσης, προβλήματα στην AI ή την ξαφνική διακοπή σωστής λειτουργίας των physics simulation [29].

Τα τελευταία χρόνια, η βελτιστοποίηση της τεχνολογίας απόδοσης έχει περιστραφεί γύρω από αυτά τα δύο προβλήματα και έχει διαμορφωθεί κάποια τεχνική συναίνεση στον κλάδο.

Οι developers προκειμένου να αντιμετωπίσουν τα παραπάνω δύο ζητήματα χρειάζεται αρχικά να βελτιώσουν το rendering του παιχνιδιού όταν τρέχει σε πραγματικό χρόνο και αυτό θα γίνει με το LOD που αναφέρθηκε προηγουμένως αλλά και με το occlusion culling [30], όπου είναι μια διαδικασία προσωρινής εξαφάνισης των αντικειμένων στη σκηνή που δεν χρειάζεται να βλέπει ο παίκτης, καθώς και το dynamic resolution scaling [31], το οποίο αφορά την προσαρμογή στο ποσοστό της κύριας οθόνης ανάλογα με τον φόρτο εργασίας της GPU των προηγούμενων καρέ. Η ανάλυση προσαρμόζεται, ανάλογα με τις ανάγκες, για παράδειγμα, όταν υπάρχουν πάρα πολλά αντικείμενα στην οθόνη ή όταν υπάρχει ένα μεγάλο εφέ που μπαίνει ξαφνικά στο καρέ. Ο χρόνος απόδοσης της GPU θα αυξηθεί και η ανάλυση της οθόνης θα μειωθεί για να διατηρηθεί το frame-rate που έχει ορίσει ότι επιθυμεί ο παίκτης.

Το μέγεθος της κρυφής μνήμης και οι διάφοροι τύποι που αναφέρθηκαν παραπάνω έχουν γίνει σταδιακά η βέλτιστη πρακτική για τη βελτιστοποίηση σύνθετων σκηνών και οι προμηθευτές των GPU έχουν σταδιακά αναγνωρίσει αυτή τη νέα ροή επεξεργασίας των vertex.

Για την ιστορία, το Driven Rendering [32] βασίζεται στο Compute Shader culling. Τα δεδομένα που έχουν αφαιρεθεί πρέπει να αποθηκευτούν στο Buffer της GPU μέσω από τα APIs όπως το Execute Indirect, το οποίο είναι για να επαναφέρετε όλες τις δεσμεύσεις που επηρεάζονται από αυτό σε γνωστές τιμές, και το αφαιρεθέν Vertex/Index Buffer που τροφοδοτείται πίσω στο Pipeline της GPU και το οποίο έχει αυξήσει την αόρατη επιβάρυνση της ανάγνωσης και της εγγραφής.

Με βάση το παραπάνω, προκειμένου να βελτιώσει περαιτέρω την ευελιξία της επεξεργασίας vertex, η NVidia εισήγαγε για πρώτη φορά την έννοια του Mesh Shader[33], ελπίζοντας να καταργήσει σταδιακά ορισμένες σταθερές μονάδες (το VAF (Vertex Attribute Fetch), όπου διαβάζει τις τιμές των χαρακτηριστικών από τη μνήμη και τις στέλνει στο vertex shade, το οποίο αντλεί δείκτες από το index buffer και στέλνει triangles στον vertex shader)[34]. Παρακάτω η Εικόνα δείχνει πως είναι ένα Mesh Shader.

Εικόνα 3.9: Σχηματική αναπαράσταση του Cluster

Πλέον, όμως με το Nanite, αυτό παύει να είναι πρόβλημα, και θα δώσει αρκετό χώρο στην επεξεργαστική ισχύ και στην GPU να αξιοποιήσουν αυτή την δύναμη σε σημεία που είναι πραγματικά απαραίτητο, όπως το ακόμα καλύτερο rendering στα materials, στον φωτισμό και φυσικά στην δημιουργία πολύ μεγαλύτερων κόσμων καθώς και σε άλλα. Πρόκειται για ένα virtualized geometry system [36] που επιτρέπει στη μηχανή να εργάζεται με πολύ λεπτομερή models, χωρίς να χρειάζεται να μειωθεί ο αριθμός των πολυγώνων.

Εικόνα 3.10: Nanite, ανάλυση του περιβάλλοντος σε complexity view

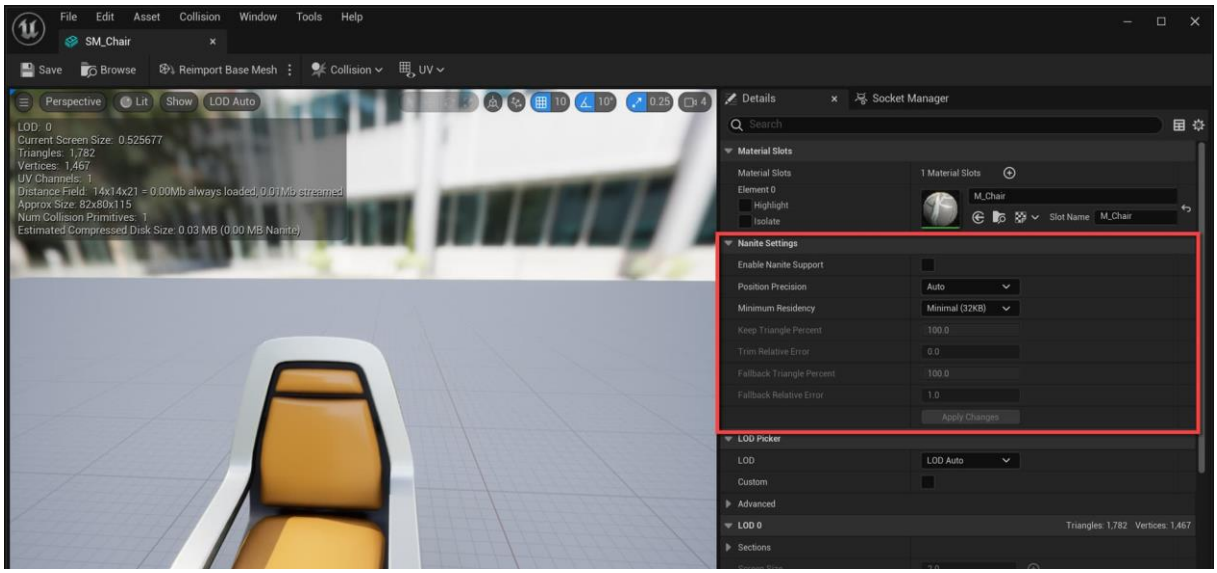
Η μηχανή μπορεί και δείχνει εκατομμύρια polygons σε πραγματικό χρόνο χωρίς να χρειάζεται να χρησιμοποιήσει αρκετή μνήμη, και πιο συγκεκριμένα, χρησιμοποίει ένα πολύ μικρό ποσοστό αυτής, και του επεξεργαστή, σε αντίθεση με άλλες μεθόδους όπου βασίζονται κυρίως στην επεξεργαστική ισχύ και την μνήμη, περιορίζοντας έτσι την δημιουργικότητα των developers.

Πιο συγκεκριμένα, το μοντέλο διασπάται σε πολύ μικρά κομμάτια, και εμφανίζονται τα polygons όταν είμαστε στο nanite mode. Αυτά τα μικρά κομμάτια, μπορούμε να τα πούμε «nanoparticles». Η μηχανή δείχνει ότι μπορεί να είναι ορατό στον παίκτη και με αποτέλεσμα να μειώνει την χρήση της ισχύς και της μνήμης σε σημαντικό βαθμό. Για να μπορεί η μηχανή φυσικά να αποδώσει έξυπνα τον αριθμό των polygons του αντικειμένου, το Nanite έχει τη δυνατότητα να δημιουργεί αυτόματα ένα LOD (Level of Detail) [37] σύστημα, όπου ανάλογα με το πόσο κοντά βρίσκεται ο παίκτης στο αντικείμενο, θα αλλάζει και το επίπεδο λεπτομέρειας. Αν απομακρύνεται, το επίπεδο λεπτομέρειας θα πέφτει.

3.3.1. Η σχέση του Nanite Mesh με το Static Mesh

Ένα Nanite mesh είναι ένα static mesh [38] με ενεργοποιημένο το Nanite. Ένα Nanite mesh εξακολουθεί να είναι ουσιαστικά ένα triangle mesh στον πυρήνα του με πολλά επίπεδα λεπτομέρειας που εφαρμόζονται στα δεδομένα του. Μάλιστα, το Nanite χρησιμοποιεί ένα εντελώς νέο σύστημα για την απόδοση αυτής της μορφής δεδομένων με εξαιρετικά αποτελεσματικό τρόπο.

Το μόνο που απαιτείται για να εκμεταλλευτεί ένα static mesh τα πλεονεκτήματα του Nanite, είναι ένα κουμπί για την ενεργοποίηση του. Το Nanite δεν διαφέρει από τα παραδοσιακά meshes, με τη διαφορά ότι το Nanite μπορεί να χειριστεί πολλά περισσότερα triangles από ό,τι είναι δυνατό για το παραδοσιακό geometry. Ο developer το μόνο που έχει να κάνει είναι να μετακινήσει την κάμερα αρκετά κοντά και το Nanite θα σχεδιάσει τα αρχικά triangles της πηγής που εισήχθησαν.



Εικόνα 3.11: Η ενεργοποίηση του Nanite

Τα Nanite meshes υποστηρίζουν πολλαπλά UV και vertex χρώματα. Τα materials αντιστοιχίζονται σε τμήματα του mesh έτσι ώστε τα materials αυτά, να μπορούν να χρησιμοποιούν διαφορετικά μοντέλα σκίασης και dynamic effects που μπορούν να γίνουν στα shaders. Η ανάθεση των materials μπορεί να εναλλάσσεται δυναμικά, όπως και κάθε άλλο static mesh, και το Nanite δεν απαιτεί καμία διαδικασία για το «ψήσιμο» των materials ή αλλιώς το Bake down, όπου είναι ένας τρόπος εφαρμογής ενός συνόλου από textures σε ένα τρισδιάστατο μοντέλο, έτσι ώστε να μην χρειάζεται πλέον να υπολογίζονται κατά το rendering.[39]

3.4. Real-time global illumination

Ο παγκόσμιος φωτισμός σε πραγματικό χρόνο (Real-time global illumination) αφορά την υψηλή ποιότητα και το dynamic lighting [40] στα περιβάλλοντα των παιχνιδιών. Αυτή η τεχνολογία επιτρέπει στη μηχανή να προσομοιώνει τον τρόπο με τον οποίο συμπεριφέρεται το φως στον πραγματικό κόσμο, λαμβάνοντας υπόψη τα materials, τα σχήματα και τα textures των αντικειμένων σε μια σκηνή για τη δημιουργία πιο ρεαλιστικού φωτισμού.

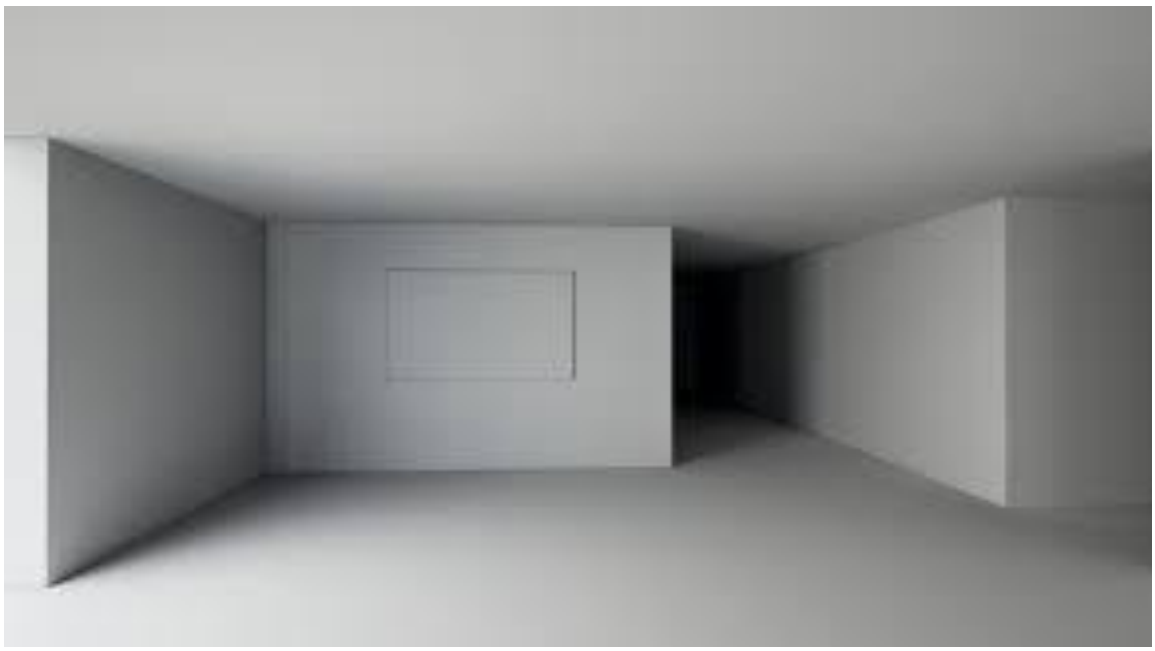
Ο παγκόσμιος φωτισμός χρησιμοποιεί προηγμένες τεχνικές για να υπολογίζει τον τρόπο με τον οποίο το φως αναπηδά και διασκορπίζεται σε μια σκηνή, λαμβάνοντας υπόψη τόσο το Direct[41] όσο και το Indirect Lighting[42]. Το Direct Lighting είναι το φως που φτάνει σε μια επιφάνεια απευθείας από μια

πηγή φωτός, ενώ το Indirect Lighting είναι το φως που έχει διασκορπιστεί από άλλα αντικείμενα στη σκηνή και φτάνει σε μια επιφάνεια έμμεσα. Αυτό δημιουργεί ένα πιο φυσικό και πιστευτό περιβάλλον φωτισμού, με soft σκιές, διακριτικές αντανακλάσεις και πολλά άλλα.

Εικόνα 3.12: Direct Lighting

Εικόνα 3.13: Indirect Lighting

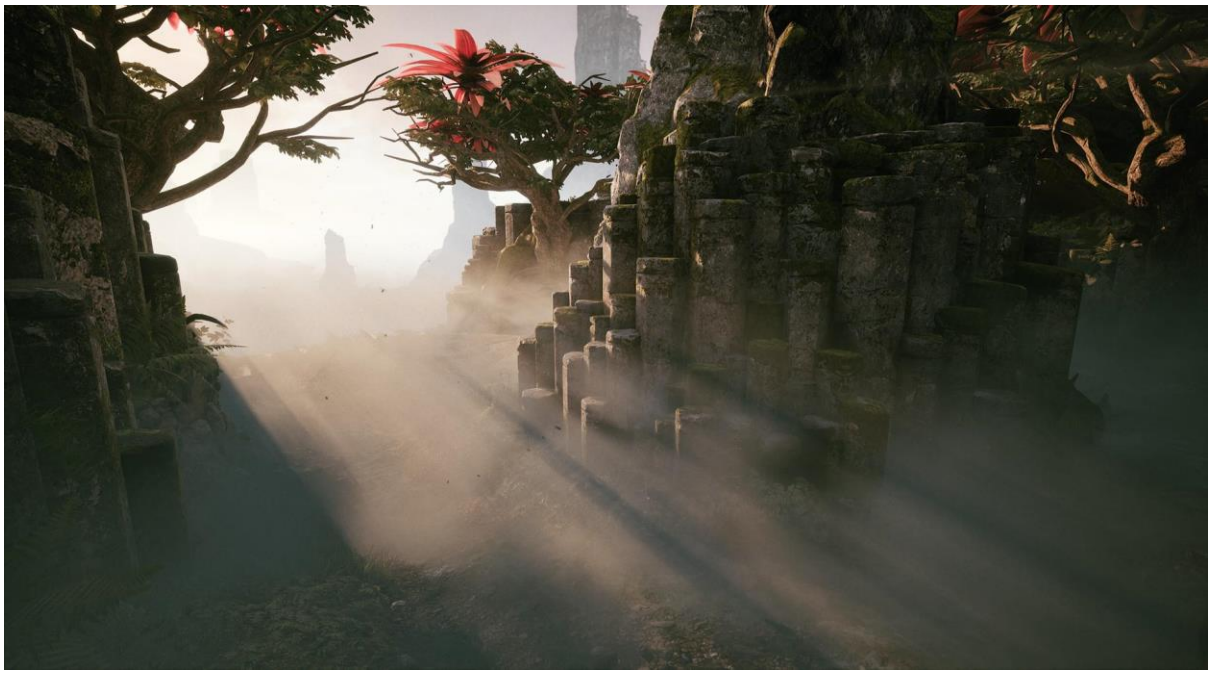
Το lightmass[43], είναι και αυτό μέρος του παγκόσμιου φωτισμού το οποίο χρησιμοποιείται για το pre-calculation του Indirect Lighting σε μια σκηνή. Το lightmass καταφέρνει να υπολογίσει το indirect lighting, λαμβάνοντας υπόψη τα materials, τα σχήματα και τα χρώματα των αντικειμένων στη σκηνή και αποθηκεύοντας αυτές τις πληροφορίες σε έναν lightmap[44] που μπορεί να χρησιμοποιηθεί κατά τον χρόνο εκτέλεσης για την απόδοση της σκηνής. Αυτό το lightmap περιέχει πληροφορίες σχετικά με το Indirect Lighting σε μια σκηνή και μπορεί να χρησιμοποιηθεί για τον γρήγορο και ακριβή υπολογισμό του φωτισμού, ακόμη και σε μεγάλα και πολύπλοκα περιβάλλοντα.



Εικόνα 3.14: Indirect Lighting με lightmass

Το volumetric lighting[45] είναι μια ακόμα τεχνική φωτισμού, το οποίο αλληλεπιδρά με το περιβάλλον με πιο φυσικό τρόπο και αυτό γίνεται εντός του Lightmass Importance Volume, όπου

ουσιαστικά είναι η περιοχή η οποία καλύπτει το lightmass, δημιουργώντας πιο ρεαλιστικές σκιές και αντανακλάσεις και προσθέτοντας βάθος και διάσταση σε μια σκηνή. Το volumetric lighting μπορεί να χρησιμοποιηθεί για τη δημιουργία εφέ όπως η ομίχλη.



Εικόνα 3.15: Volumetric Lighting

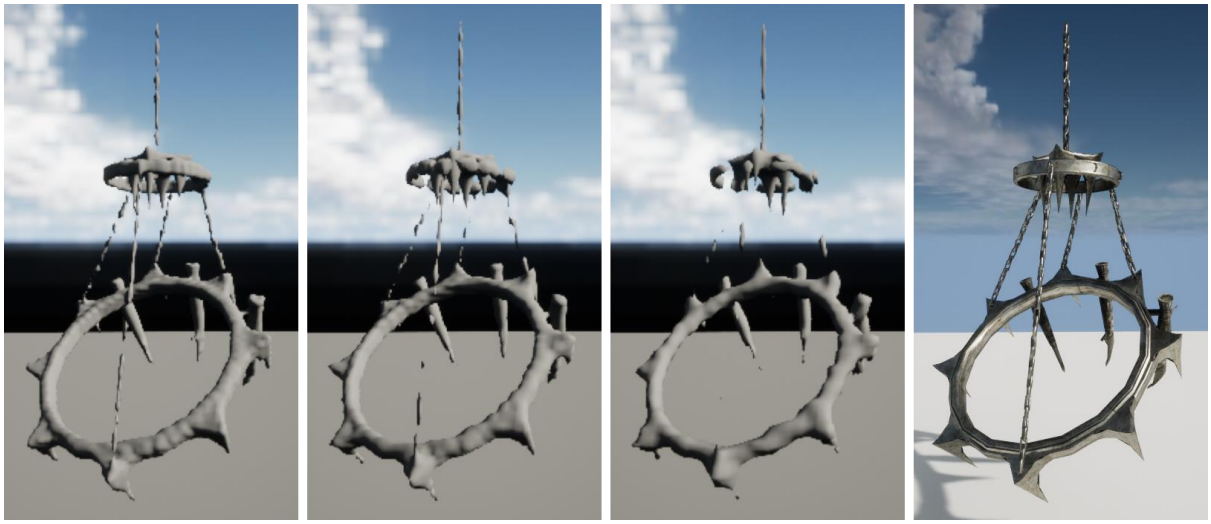
Έπειτα έχουμε τα reflections[46] όπου είναι μια σημαντική πτυχή για να ζωντανέψουν τα περιβάλλοντα, επιτρέποντας στο φως να κάνει αντανάκλαση στα αντικείμενα στη σκηνή. Οι αντανακλάσεις ξεκινούν από τον τρόπο με τον οποίο έχουν ρυθμιστεί τα materials. Τα materials που στις ιδιότητες τους δεν έχουν υψηλό νούμερο στο roughness, κάνουν τις επιφάνειες λιγότερο ανακλαστικές. Αυτό μπορεί να είναι η διαφορά μεταξύ της δημιουργίας μιας επιφάνειας που μοιάζει με καθρέφτη ή σαν βουρτσισμένο μέταλλο.

Εικόνα 3.16: Reflection στην Unreal Engine 5

3.4.1. Lumen

Το Lumen[47] είναι ένα νέο σύστημα παγκόσμιου φωτισμού που εισήχθη στην Unreal Engine 5, το οποίο στοχεύει στον ρεαλιστικό φωτισμό και την σκίαση σε πραγματικό χρόνο. Αποτελεί σημαντική βελτίωση σε σχέση με το προηγούμενο σύστημα φωτισμού της Unreal Engine 4, το οποίο χρησιμοποιούσε λύσεις pre-computed φωτισμού που ήταν περιορισμένες ως προς την ικανότητα τους να προσομοιώνουν δυναμικό φωτισμό και σκιές, όπως το Distance Field Soft Shadows.

Το Lumen χρησιμοποιεί έναν συνδυασμό τεχνικών του ray-tracing[48], το οποίο προσπαθεί να μιμηθεί τον τρόπο με τον οποίο λειτουργεί το φως στον πραγματικό κόσμο και το φως θα μπορούσε να αναγνωριστεί ως μια ροή φωτονίων, και του voxelization [49], το οποίο είναι η διαδικασία μετατροπής δομών δεδομένων που αποθηκεύουν γεωμετρικές πληροφορίες σε μια εικόνα σε rasterized μορφή. Εντοπίζει τις ακτίνες φωτός στη σκηνή για να προσομοιώσει τον τρόπο με τον οποίο το φως ανακλάται στις επιφάνειες, δημιουργώντας σκιές και αντανακλάσεις.



Εικόνα 3.17: Η ανάλυση του φωτισμού μέχρι και το φωτορεαλιστικό

Ένα από τα βασικά πλεονεκτήματα του Lumen είναι ότι επιτρέπει το dynamic lighting και τις σκιές σε πραγματικό χρόνο, πράγμα που σημαίνει ότι ο φωτισμός και οι σκιές μπορούν να αλλάζουν με βάση τη θέση της φωτεινής πηγής ή την κίνηση των αντικειμένων στη σκηνή.

Το Lumen επιτρέπει πιο ευέλικτες ροές εργασίας στην ανάπτυξη παιχνιδιών. Επειδή πρόκειται για ένα σύστημα φωτισμού πραγματικού χρόνου, οι developers μπορούν να δουν τις αλλαγές φωτισμού που πραγματοποιούν αμέσως, χωρίς να χρειάζεται να περιμένουν να δημιουργηθούν pre-computed λύσεις φωτισμού. Για παράδειγμα, την στιγμή που τοποθετούν ένα αντικείμενο μέσα στο περιβάλλον αυτό προσαρμόζεται αυτόματα στον φωτισμό χάρη στο Lumen και δεν χρειάζεται να κάνει bake out ώστε να δούνε τον πραγματικό φωτισμό.

Παρόλα αυτά υπάρχουν και περιορισμοί. Μία από τις μεγαλύτερες προκλήσεις είναι η απόδοση. Επειδή το Lumen είναι μια εντατική υπολογιστική διαδικασία, απαιτεί σημαντική ισχύ της GPU. Φυσικά, για να το υποστηρίξει αυτό, χρειάζεται μια κάρτα γραφικών από την NVIDIA από την RTX 2080 και νεότερη, ενώ από την AMD από την Radeon RX 6800 XT και νεότερη, για την επίτευξη των ικανοποιητικών αποτελεσμάτων. Αυτό σημαίνει ότι μπορεί να μην είναι κατάλληλο ακόμα για συσκευές που δεν έχουν μεγάλη υπολογιστική ισχύ.

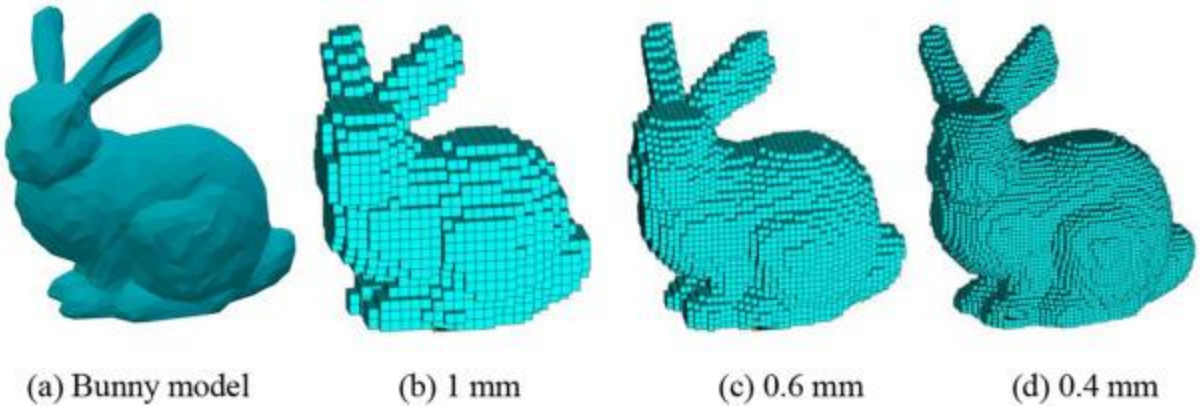
Εικόνα 3.18: Η ρεαλιστικότητα του Lumen

Η φύση του Lumen σημαίνει ότι μπορεί να μην είναι κατάλληλη για όλους τους τύπους παιχνιδιών. Τα παιχνίδια που απαιτούν υψηλό βαθμό ακρίβειας και ελέγχου του φωτισμού, όπως τα παιχνίδια παζλ ή τα παιχνίδια στρατηγικής, ενδέχεται να εξακολουθούν να επωφελούνται από λύσεις φωτισμού που έχουν υπολογιστεί εκ των προτέρων.[50]

3.4.1.1. Voxelization (Voxel)

Προηγουμένως αναφερθήκαμε στο ότι το Lumen χρησιμοποιεί Voxelization. Το Voxelization είναι μια διαδικασία στην Unreal Engine (και σε άλλες μηχανές παιχνιδιών) που μετατρέπει την πολυγωνική γεωμετρία των τρισδιάστατων αντικειμένων σε όγκο voxels, τα οποία είναι τρισδιάστατα pixel ή "volume pixels". Το Voxelization χρησιμοποιείται συχνά σε συνδυασμό με το ray-tracing και άλλες τεχνικές σε πραγματικό χρόνο για τη βελτίωση της απόδοσης και της οπτικής ποιότητας.

Το voxelization χρησιμοποιείται με διάφορους τρόπους. Ένας από τους πιο συνηθισμένους είναι η υποστήριξη του παγκόσμιου φωτισμού σε πραγματικό χρόνο όπως αναφερθήκαμε παραπάνω, ο οποίος χρησιμοποιείται για τον υπολογισμό του τρόπου με τον οποίο το φως ανακλάται στις επιφάνειες της σκηνής. Με το voxelization της γεωμετρίας, η μηχανή μπορεί να δημιουργήσει μια αναπαράσταση της σκηνής που μπορεί να ανιχνευθεί γρήγορα και αποτελεσματικά από τις ακτίνες φωτός, επιτρέποντας πιο ρεαλιστικό φωτισμό και σκιές.



Εικόνα 3.19: Το Voxelization σε 4 διαφορετικά στάδια

Μια άλλη χρήση του voxelization στην Unreal Engine είναι η υποστήριξη των physics simulation. Με το voxelization της γεωμετρίας των αντικειμένων στη σκηνή, η μηχανή μπορεί να δημιουργήσει μια φυσική αναπαράσταση αυτών των αντικειμένων που μπορεί να προσομοιωθεί ευκολότερα και να αλληλεπιδράσει με άλλα αντικείμενα στη σκηνή, κάτι που βοηθάει τα παραμορφωμένα αντικείμενα ή την δημιουργία ενός destructible περιβάλλοντος, όπου δηλαδή τα assets μπορούν να διασπαστούν σε τυχαία κομμάτια, όπως για παράδειγμα το σπάσιμο ενός πιάτου.

Για παράδειγμα, οι developers έχουν ένα περιβάλλον με έναν χαρακτήρα, μια μπάλα, και κάποια ξύλα. Σε αυτό το περιβάλλον, θέλουν να δημιουργήσουν ένα ρεαλιστικό physics simulation όπου τα αντικείμενα αλληλεπιδράνε μεταξύ τους.

Voxelization: Αρχικά, το περιβάλλον αποτελείται από πολύπλοκα τρισδιάστατα μοντέλα για κάθε αντικείμενο, όπως λεπτομερή πολυγωνικά meshes για τον χαρακτήρα, τα ξύλα και ένα σφαιρικό mesh για τη μπάλα. Για την απλοποίηση των υπολογισμών των physics, κάνουν voxelize τα αντικείμενα αυτά:

- Το μοντέλο του χαρακτήρα διαμορφώνεται σε ένα πλέγμα από cubic voxels, όπου κάθε voxel αντιπροσωπεύει ένα μικρό τμήμα του όγκου του χαρακτήρα.
- Τα ξύλα επίσης έγιναν voxelized με παρόμοιο τρόπο, μετατρέποντας τα πολύπλοκα σχήματα τους σε voxel πλέγματα.
- Τέλος, η μπάλα αναπαρίσταται από μια voxelized σφαίρα.

Physics Simulation: Η μηχανή μπορεί πλέον να προσομοιώνει εύκολα τις αλληλεπιδράσεις μεταξύ αντικειμένων:

- Όταν ο χαρακτήρας πηδά και ακουμπάει σε ένα voxelized ξύλο, η μηχανή μπορεί να υπολογίσει πως τα voxels του χαρακτήρα αλληλοεπιδρούν με τα voxels του ξύλου, επιτρέποντας ρεαλιστικές αντιδράσεις σύγκρουσης.
- Εάν ο χαρακτήρας σπρώξει ένα ξύλο, η μηχανή μπορεί να υπολογίσει τις δυνάμεις που ασκούνται στα voxelized ξύλα και την κίνηση τους αναλόγως.
- Όταν η μπάλα αναπηδά σε επιφάνειες ή συγκρούεται με άλλα voxelized αντικείμενα, η μηχανή μπορεί να χειριστεί τα physics των αλληλεπιδράσεων της voxelized μπάλας.

Το Voxelization, εκτός από τον παγκόσμιο φωτισμό και τα physics, μπορεί επίσης να χρησιμοποιηθεί και για άλλους σκοπούς, όπως για τη δημιουργία volumetric effects, όπως η ομίχλη ή ο καπνός, ή για τη δημιουργία λεπτομερών textures.[51]

Κεφάλαιο 4: Υλοποίηση του project

Στα πλαίσια της υλοποίησης του project θα δείξουμε βήμα-βήμα το πως υλοποιήθηκε με σκοπό να χρησιμοποιήσουμε τα νέα τεχνικά χαρακτηριστικά όπου αναφέρθηκαν παραπάνω με σκοπό να αποδείξουμε την σημαντική συνεισφορά των νέων μηχανισμών της Unreal Engine 5, Nanite και Lumen, αλλά και την εξέλιξη των blueprints. Παράλληλα θα παρουσιάσουμε τα blueprints, πως δημιουργήθηκαν για την κάθε λειτουργία που υπάρχει μέσα στο project και το πόσο εύκολα είναι στην χρήση και τις ατελείωτες δυνατότητες του. Τα assets που χρησιμοποιήθηκαν ήταν από το marketplace της Unreal Engine και υλοποιήσαμε πάνω σε αυτά τα blueprints. Το marketplace έχει πληθώρα assets ώστε να βοηθήσουν τους επίδοξους προγραμματιστές να ξεκινήσουν με μια βάση αλλά και όχι μόνο, καθώς ο αριθμός τους είναι μεγάλος και μπορεί κάποιος να δημιουργήσει ένα παιχνίδι εξ' ολοκλήρου με αυτά.

4.1. Δημιουργία χαρακτήρα

Η δημιουργία του χαρακτήρα στην Unreal Engine με blueprints μπορεί να φαίνεται εκ πρώτης όψεως εύκολη αλλά με λίγο περισσότερη προσοχή καταλαβαίνουμε πως για να έχουμε έναν πλήρως λειτουργικό χαρακτήρα όπου να μπορεί να κάνει διάφορες κινήσεις, ρεαλιστικά και για να μπορεί να αλληλοεπιδρά με το περιβάλλον, χρειάζεται αρκετά, και πολλές φορές περίπλοκα, blueprints για να πετύχουμε ένα καλό αποτέλεσμα.

Για να γίνουν όλα αυτά μάλιστα, χρειάζεται και την συνδρομή των Enums όπου εξηγούμε παραπάνω τι είναι και πως χρησιμοποιούνται ενώ φυσικά απαιτείται και ο animation editor της Unreal Engine όπου θα δώσει κίνηση στον χαρακτήρα μας.

Παρακάτω θα δούμε λίγο πιο αναλυτικά, πως δημιουργείται το κάθε κομμάτι του χαρακτήρα μας, πέρα από το γραφιστικό όπου αυτό μπορεί να γίνει μέσα από το Blender ή κάποιο παρόμοιο πρόγραμμα δημιουργίας 3d model.

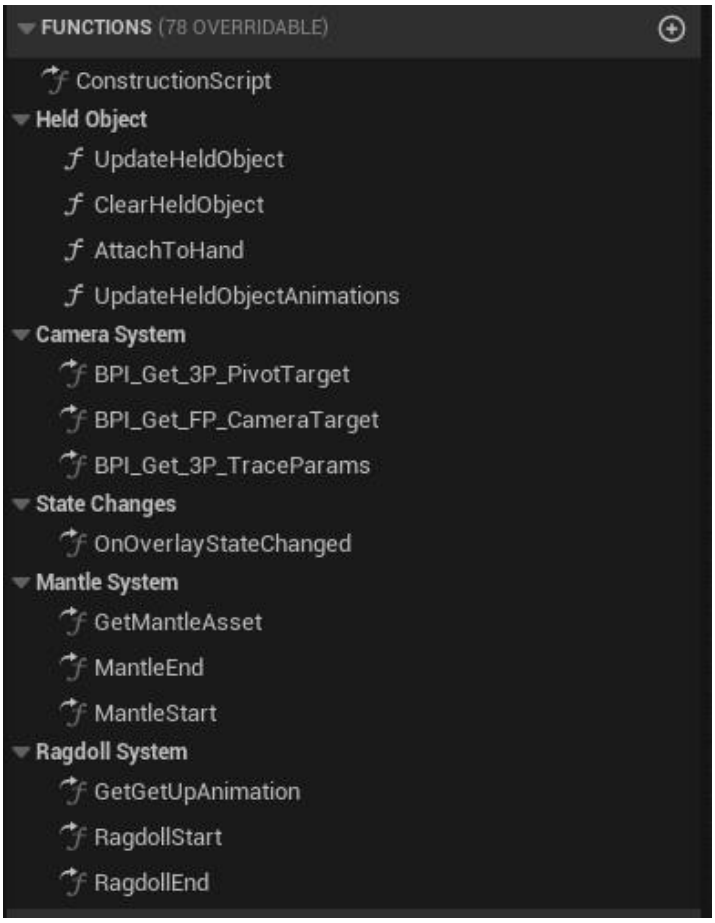
4.1.1. Character Blueprint

Το Character Blueprint συνδυάζει το τρισδιάστατο μοντέλο, την κίνηση του χαρακτήρα και ένα σύνολο συμπεριφορών και αλληλεπιδράσεων σε ένα ενιαίο επαναχρησιμοποιήσιμο asset. Θα μπορούσαμε να πούμε δηλαδή ότι είναι ο πυρήνας του χαρακτήρα και από εκεί μπορούμε να ελέγξουμε όλες τις λειτουργίες που μπορεί να δημιουργήσαμε σε ξεχωριστά blueprints.

Στην παρακάτω Εικόνα βλέπουμε όλα τα functions του χαρακτήρα όπου περνάνε μέσα από αυτό το κεντρικό (πυρήνας) blueprint. Σε αυτό βλέπουμε τις κλάσεις όπως του held object, όπου ο παίκτης χάρη αυτής μπορεί να συλλέγει αντικείμενα και να αποθηκεύονται σε ένα inventory για να τα χρησιμοποιήσει μελλοντικά.

Έπειτα είναι το σύστημα κάμερας όπου αφορά το τι βλέπει ο παίκτης κατά την διάρκεια του παιχνιδιού και ανάλογα που κοιτάει ο χαρακτήρας, μετατοπίζεται και το σώμα του και ότι είναι συνδεδεμένο πάνω σε αυτό.

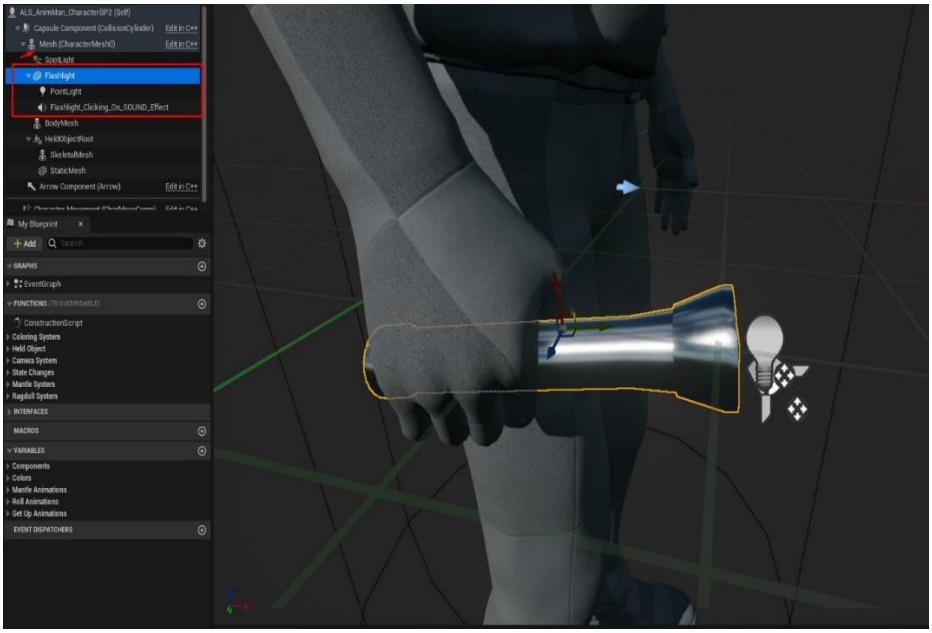
Αυτά είναι μερικές από τις λειτουργίες όπου μπορεί να έχει ένας χαρακτήρας και παρακάτω θα δούμε λίγο πιο αναλυτικά τα blueprints αυτά μεμονωμένα.



Εικόνα 4.1: Όλα τα Functions από το Character Blueprint

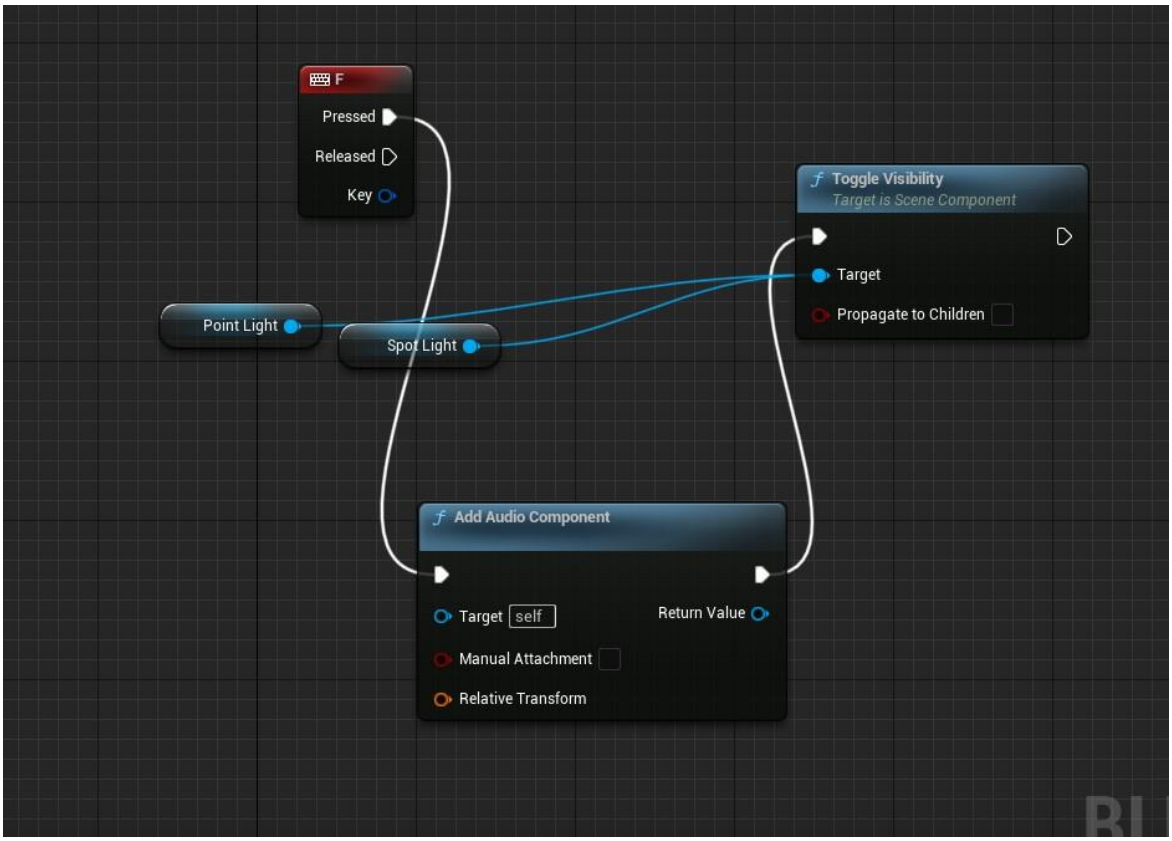
4.1.2. Flashlight

Η δημιουργία του φακού είναι ένα από τα πιο απλά πράγματα, καθώς ο κώδικας του είναι μικρός. Αφού τοποθετήσουμε το asset μέσα στο blueprint και το κάνουμε attach στο mesh του χαρακτήρα τότε είμαστε έτοιμοι για να του δώσουμε την λειτουργία ώστε να δουλεύει κανονικά.



Εικόνα 4.2: Τοποθέτηση του φακού στο χέρι του χαρακτήρα

Αυτό γίνεται με το να συνδέσουμε το πλήκτρο που θέλουμε για να ανάβουμε τον φακό, και μετά να το συνδέσουμε με το toggle visibility το οποίο αφορά την φωτεινότητα του φακού, καθώς και την τοποθέτηση του «κλικ» ήχου μέσα από ένα audio component.



Εικόνα 4.3: Τα blueprints για την λειτουργικότητα του φακού

Με αυτό, έχουμε πλέον τον φακό μας όπου μπορούμε να τον ανοίγουμε όποτε θέλουμε και σε σημεία που είναι σκοτεινά. Το άνοιγμα του φακού ουσιαστικά γίνεται πατώντας το κουμπί F όπως φαίνεται και στην εικόνα 4.3 και από εκεί παίρνει εντολή το Audio Component ώστε να παίξει ο ήχος ανοίγματος του φακού, και έπειτα πάει στο toggle visibility για να ενεργοποιηθεί το spot light και το point light. Φυσικά, το φως πρέπει να είναι δυναμικό ώστε να μπορεί να προσαρμόζεται κατάλληλα με το περιβάλλον.

4.1.3. Movement

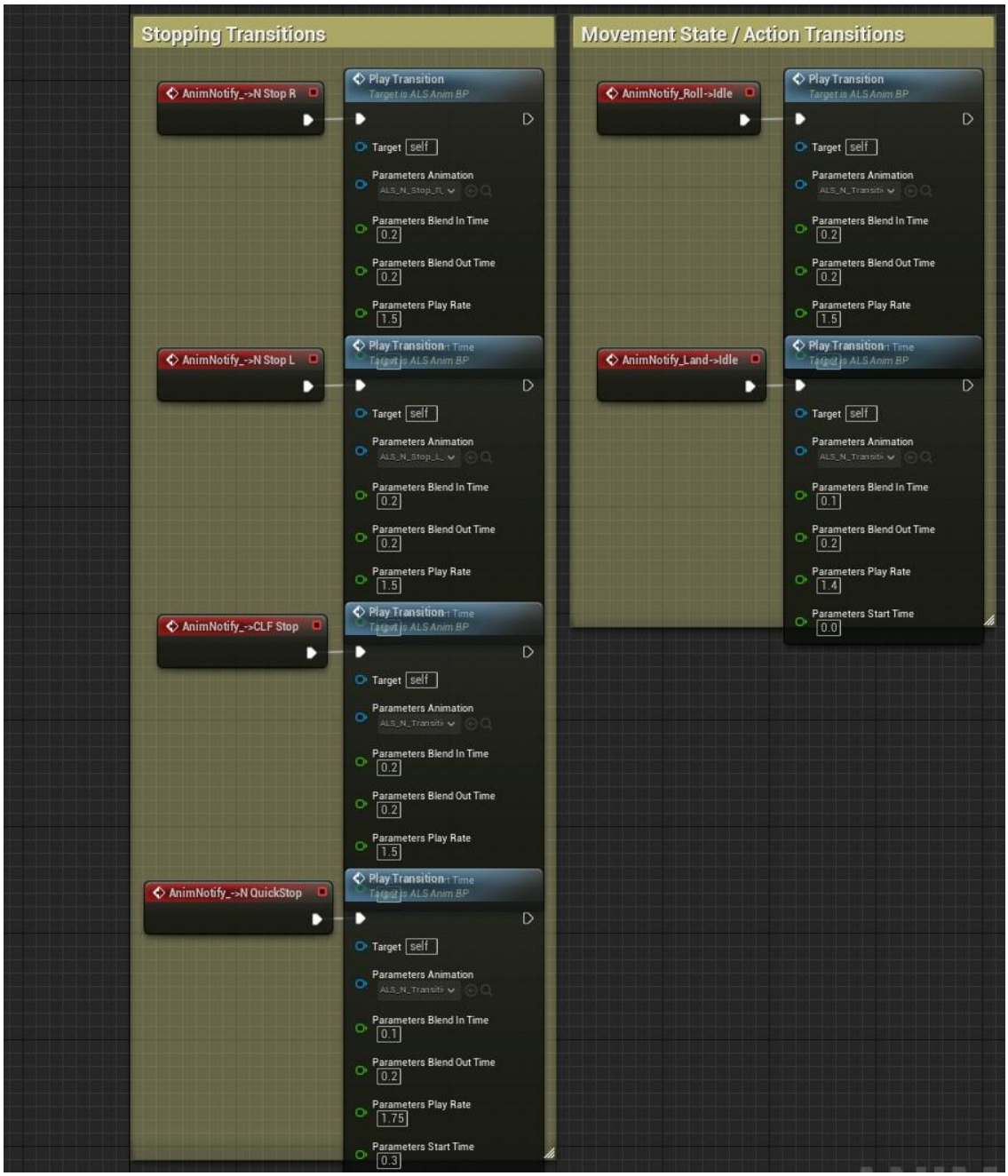
Στην κίνηση του χαρακτήρα εμβαθύνουμε πιο πολύ πλέον στις λειτουργίες της μηχανής καθώς χρειάστηκε να κάνουμε το animation για την κάθε κίνηση που ο χαρακτήρας μας θα έκανε. Αυτό πραγματοποιείται μέσα από ένα animation blueprint το οποίο είναι ο πυρήνας της κίνησης του χαρακτήρα μας και ό,τι κίνηση κάνει ενεργοποιείται από εκεί.

Μέσα στο animation blueprint βλέπουμε όλες τις λειτουργίες των animations όπου η καθεμία έχει συνδεθεί με συγκεκριμένα σημεία του σκελετού ώστε να του δώσει την κίνηση που θέλουμε.

Εικόνα 4.4: Τα functions του animation blueprint

Αρχικά, έχουμε τα main events στο event graph του blueprint όπου αφορούν τις βασικές λειτουργίες όπως το να ξεκινήσει να προχωρά ή να σταματήσει αλλά και τα transitions όταν ο χαρακτήρας μας κρατάει κάποιο αντικείμενο όπως ένα όπλο. Τα transitions αυτά έχουν συνδεθεί με το κάθε animation, όπου το κάθε animation δημιουργείται σε ένα animation sequence.

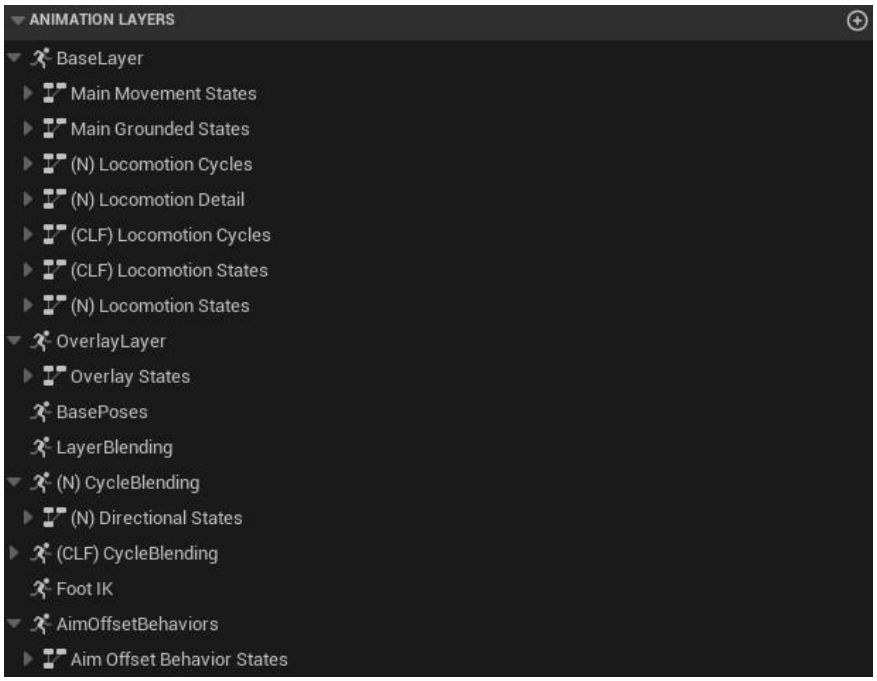
Αυτή η σύνδεση, επιτρέπει στο animation blueprint να έχει πρόσβαση στα sequences και επομένως στον σκελετό του χαρακτήρα.



Εικόνα 4.5: Transitions του χαρακτήρας

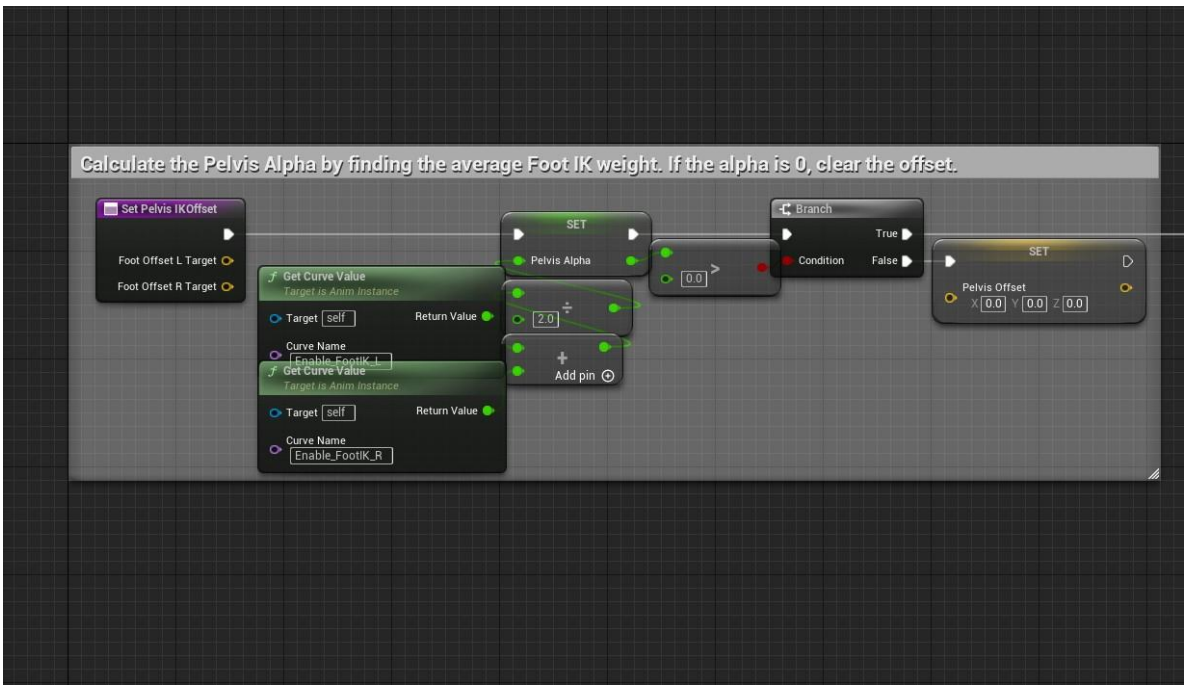
Αυτά τα events είναι αρκετά όπως φαίνεται και στην παρακάτω Εικόνα 4.5, και μπορούμε να πούμε ότι αυτός είναι ο πυρήνας του animation που θα δώσει ζωή στον χαρακτήρα μας. Έχουμε τα Stop R, Stop L, CLF Stop και Quick Stop, όπου το καθένα ανάλογα την κατεύθυνση μας, μπροστά, πίσω, δεξιά, αριστέρα, ενεργοποιεί το animation για όταν σταματάει ο χαρακτήρας να κινείται. Στα δεξιά έχουμε απλά τα idle animations για όταν δεν κινείται ο χαρακτήρας.

Το event graph του Animation blueprint έχει όλες τις λειτουργίες και για να δουλέψουν, χρειάζεται να περάσουν μέσα από τα animation layers όπου εκεί δίνουμε στην μηχανή μερικές συγκεκριμένες παραμέτρους ώστε να μπορέσει να συντονίσει τον σκελετό με τα animations.

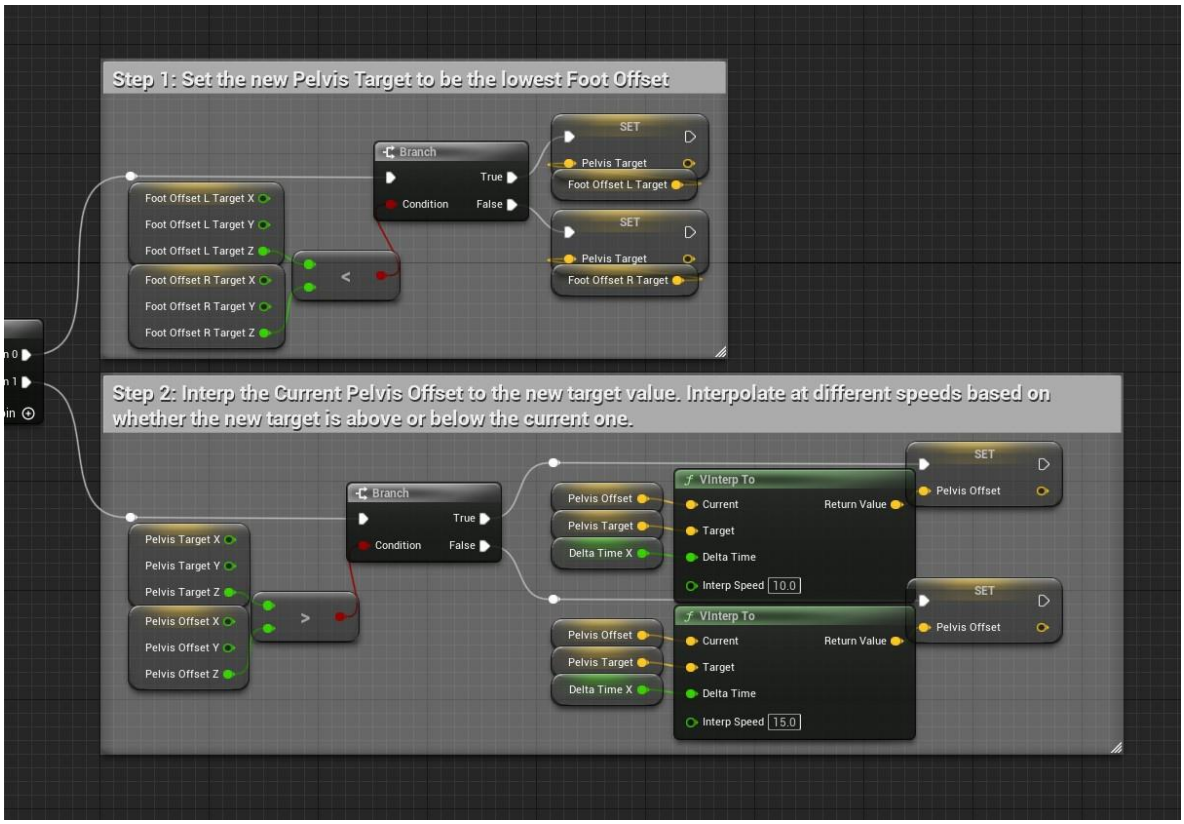


Εικόνα 4.6: Animation layers

Για παράδειγμα, στο Set Pelvis IKOffset υπολογίζει αρχικά το Pelvis Alpha βρίσκοντας το μέσο βάρος των ποδιών ώστε να δώσει την κατάλληλη ώθηση και ισορροπία στον χαρακτήρα όταν κινείται. Για να γίνει αυτό, πρώτα ορίσαμε ένα νέο στόχο της λεκάνης του χαρακτήρα ως τη χαμηλότερη μετατόπιση ποδιού. Έπειτα, διασυνδέουμε την τρέχουσα μετατόπιση της λεκάνης με τη νέα τιμή-στόχο. Κάνουμε παρεμβολή με διαφορετικές ταχύτητες ανάλογα με το αν ο νέος στόχος είναι πάνω ή κάτω από τον τρέχοντα.

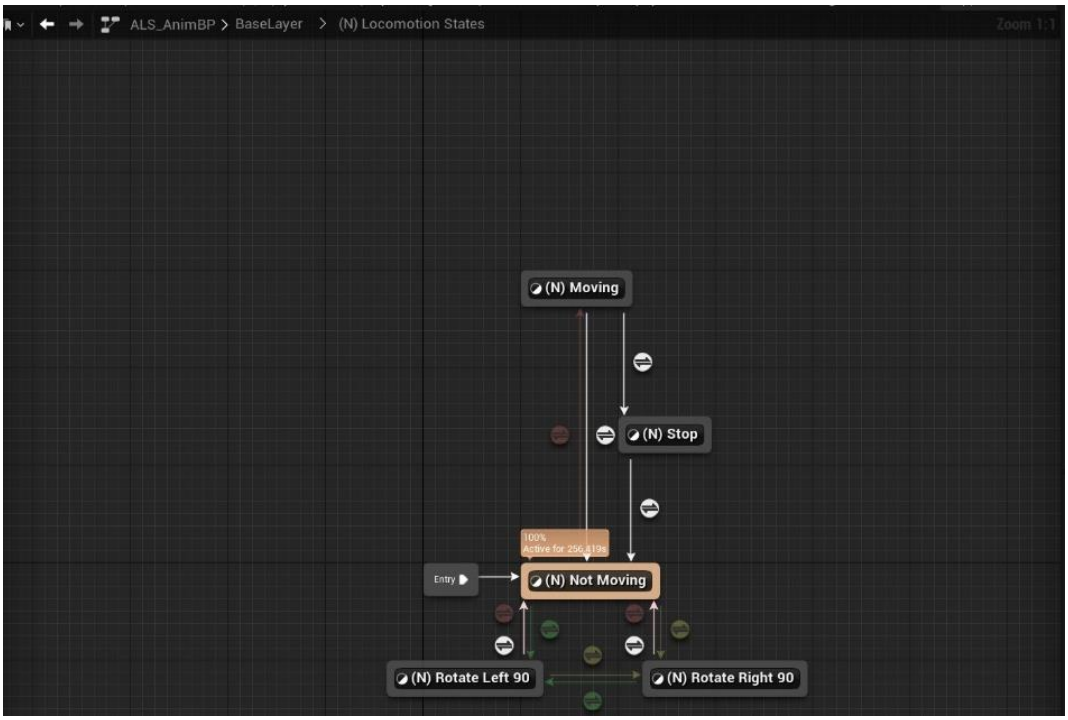


Εικόνα 4.7: Υπολογισμός βάρους ποδιών



Εικόνα 4.8: Διασύνδεση λεκάνης με την τιμή-στόχο

Εντός των animation layers, υπάρχουν τα animation state machines τα οποία είναι αρθρωτά συστήματα που μπορούμε να δημιουργήσουμε στα Animation Blueprints για να ορίσουμε animations για το πότε επιτρέπεται να αναπαραχθούν. Ένα από αυτά είναι το Locomotion state όπου ανάλογα την κίνηση που κάνουμε με το ποντίκι και το πληκτρολόγιο, αυτό δίνει εντολή για να πάει στο κατάλληλο Anim Graph, σαν αυτά που είδαμε παραπάνω, και να εκτελέσει τις εντολές που έχει εκεί.



Εικόνα 4.9: Locomotion state. Δίνει εντολή για το πια κίνηση ζητάει ο παίκτης

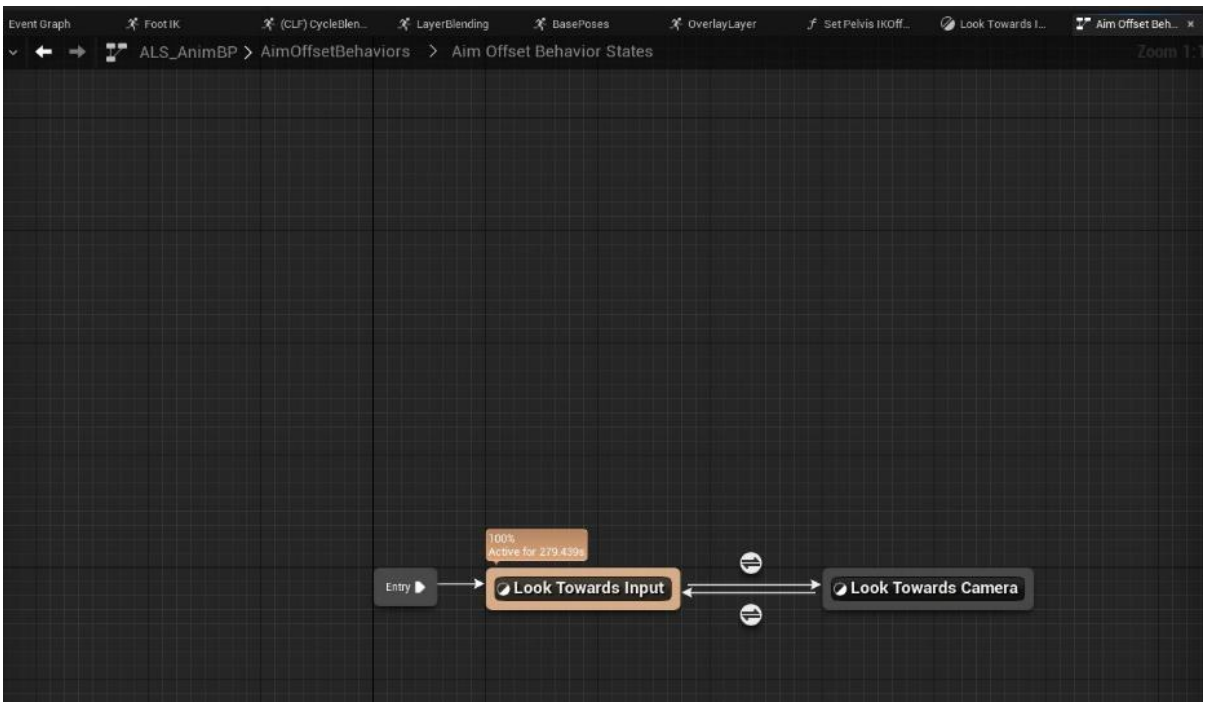
Με τον ίδιο τρόπο δημιουργούνται και τα υπόλοιπα states που βλέπουμε στην εικόνα 4.9, και κατ' επέκταση και τα animation layers. Ύστερα από αυτή την διαδικασία και εφόσον όλα έχουν εφαρμοστεί σωστά, ο χαρακτήρας θα πρέπει να έχει κίνηση σε κάθε μέρος του σώματος του και λειτουργίες, όπως να γυρίζει το κεφάλι ανάλογα εκεί που σημαδεύει το ποντίκι μας.

4.1.4. Camera System

Το σύστημα της κάμερας του χαρακτήρα είναι ένα κομμάτι των animations και με αυτό ολοκληρώνει την όλη διαδικασία υλοποίησης του χαρακτήρα. Πολλοί δημιουργοί επιλέγουν τον εύκολο δρόμο τοποθέτησης της κάμερας στο χαρακτήρα, απλά δημιουργώντας ένα βασικό animation blueprint της κάμερας και συνδέοντας το με το mesh του χαρακτήρα.

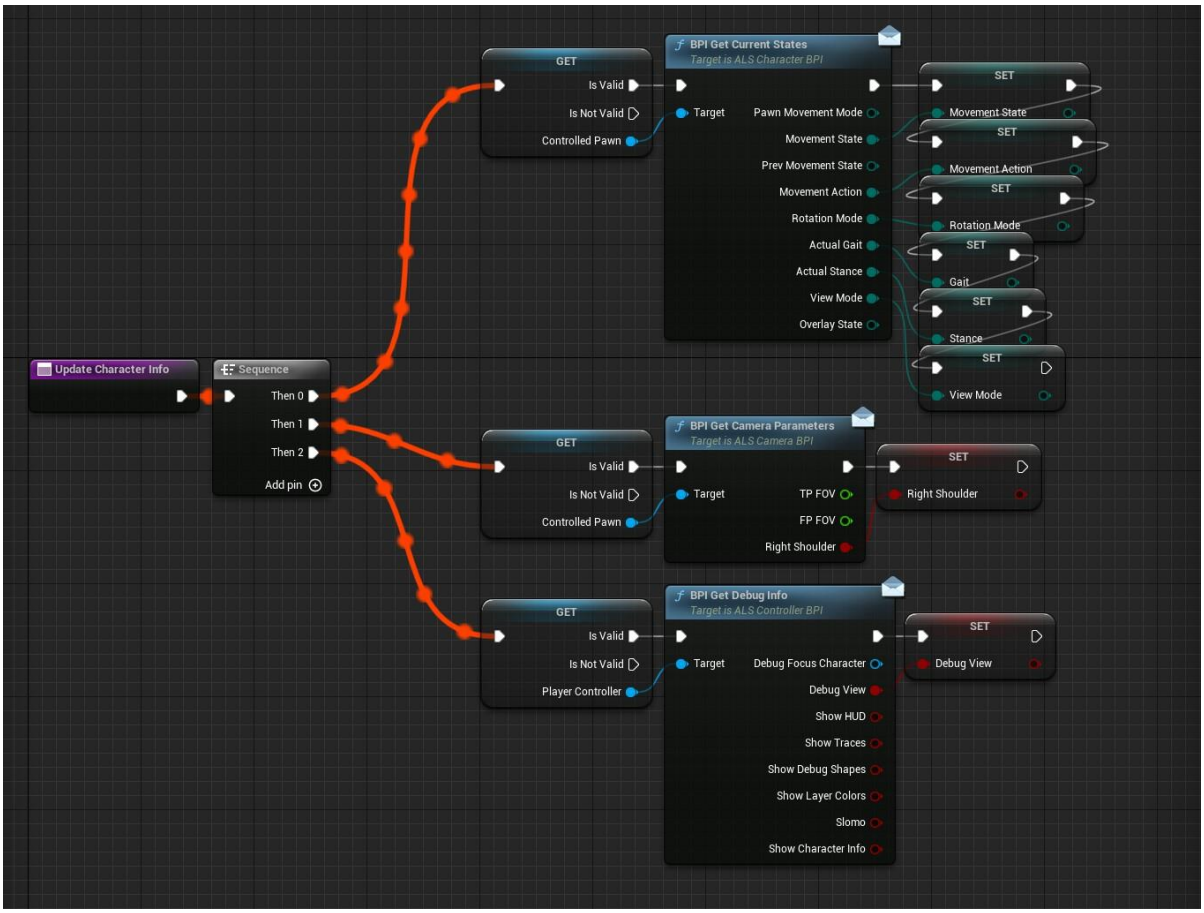
Στην περίπτωση μας υπάρχει περισσότερη προσοχή στην κάμερα του χαρακτήρα, καθώς έχει μια ρεαλιστικότητα στην κάμερα και ανάλογα την κίνηση που κάνουν τα πόδια η κάμερα έχει την δυνατότητα να πηγαίνει πάνω κάτω μαζί με το υπόλοιπο σώμα αλλά και όταν είναι στατικός, η κάμερα να κινείται ανάλογα με την αναπνοή του χαρακτήρα.

Στο παρακάτω παράδειγμα έχουμε το Aim Offset behaviour states όπου αφορά και πάλι τα animation state machines που αναφέραμε πριν, στο οποίο εάν ο χαρακτήρας μας κοιτάει σε ένα συγκεκριμένο trigger ενεργοποιεί την κάμερα του χαρακτήρα προκειμένου να δώσει εντολή για να μας δείξει τις πληροφορίες σχετικά με το trigger στην οθόνη μας. Για παράδειγμα, αν κοιτάμε μια πόρτα, δίνει εντολή στην κάμερα να μας εμφανίσει τον pointer και ένα κείμενο ότι αυτή η πόρτα ανοίγει.



Εικόνα 4.10: Η κάμερα ενεργοποιεί το input

Στο animation blueprint της κάμερας έχουμε πάλι κάποια states τα οποία είναι, Velocity Direction όπου αφορά την ταχύτητα του παίκτη, το Looking Direction όπου αφορά το που κοιτάει ο παίκτης, και το aiming όπου ο παίκτης μπορεί να σημαδέψει. Όλα αυτά περνάνε μέσα από έναν updater όπου ενημερώνει το main blueprint του χαρακτήρα για τις κινήσεις του.



Εικόνα 4.11: Updater camera animation blueprint

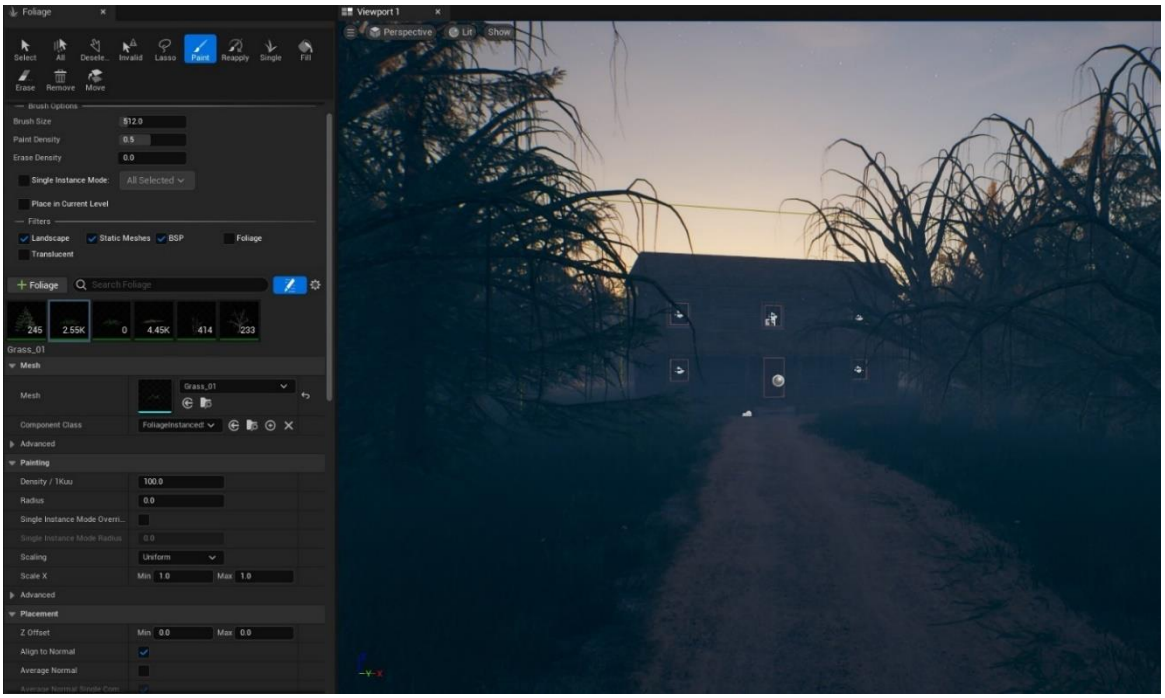
Στην εικόνα 4.11 ξεκινάει από τον updater του χαρακτήρα, πηγαίνει σε ένα sequence και έχει τρεις επιλογές οι οποίες ξεκινάνε μαζί. Στο πρώτο παίρνει την στάση του χαρακτήρα την στιγμή που ξεκινάει, έπειτα παίρνει τις παραμέτρους της κάμερας και τέλος δίνει debug πληροφορία.

4.2. Δημιουργία περιβάλλοντος

Η δημιουργία του περιβάλλοντος είναι η πιο απαιτητική διαδικασία, καθώς είναι ο κόσμος στον οποίο ο παίκτης με τον χαρακτήρα του κάνει interaction. Όλο το παιχνίδι είναι το περιβάλλον που βλέπει ο παίκτης. Η διαδικασία ξεκινάει από τον σχεδιασμό του περιβάλλοντος και του χάρτη, έπειτα προχωράει στην αρχική υλοποίηση του προσθέτοντας το terrain (έδαφος), και έπειτα έρχεται ο σχεδιασμός των assets. Στην Unreal Engine αυτά συνδυάζονται και δημιουργείται το υπόλοιπο περιβάλλον, όπως η φύση, τα σπίτια και τα αντικείμενα που έχει μέσα αυτό.

4.2.1. Foliage

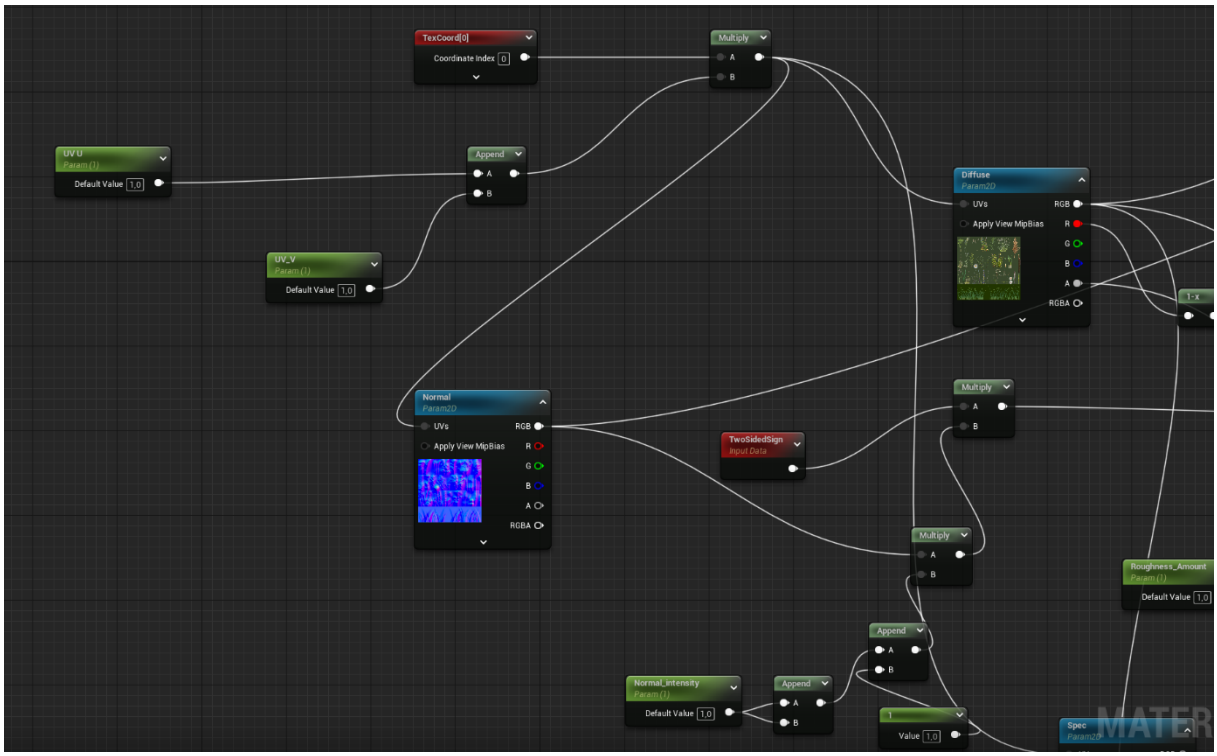
Αρχικά το foliage αφορά το χόρτο, τα δέντρα, τις πέτρες και οτιδήποτε άλλο χρειάζεται να έχει η φύση. Όλα αυτά έχουν το δικό τους animations και μπορεί να ρυθμιστεί ανάλογα με τον αέρα που θέλουμε να έχει ώστε να κουνιούνται πιο πολύ. Είναι εύκολη η τοποθέτηση του στον χάρτη μας με ένα απλό paint brush, στοχεύουμε εκεί που θέλουμε, ρυθμίζουμε την ποσότητα του και απλά κάνουμε ένα spray.



Εικόνα 4.12: ρυθμίσεις foliage

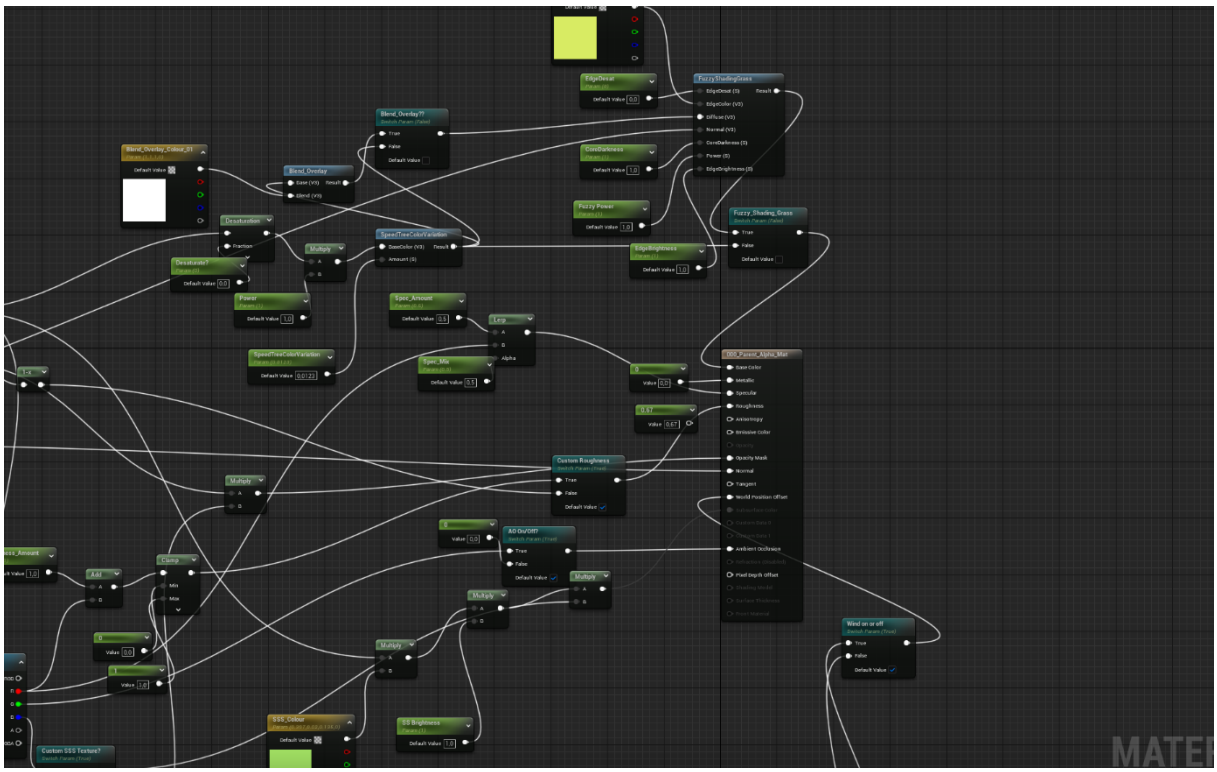
Κάθε αντικείμενο, όπως δέντρα ή χόρτο, έχουν το δικό τους texture και material, το οποίο material είναι ουσιαστικά ένα blueprint. Μέσα σε αυτό υπάρχει όλος ο κώδικας όπου του αναθέτουμε το texture, τους χρωματισμούς που ίσως θέλουμε να έχει αλλά και την κίνηση των φύλλων. Τα normal maps αναθέτονται επίσης από εκεί και μπορούμε να τα ρυθμίσουμε και αυτά ανάλογα με τις ανάγκες μας στο project.

Φυσικά εκεί μπορούμε να ρυθμίσουμε και την κίνηση των φύλλων, τους χρωματισμούς του και οτιδήποτε άλλο χρειάζεται να έχει ένα foliage material.



Εικόνα 4.13: material blueprint (1)

Το Blueprint του material στην εικόνα 4.13 ξεκινάει με την σύνδεση των normal map και diffuse map, και δίνουμε παραμέτρους για το πόσο έντονα θα εμφανίζονται στο τελικό material μέσα από τα UVU variables.



Εικόνα 4.14 material blueprint (2)

Για την τελικό αποτέλεσμα του όλου node, προσθέτουμε την παράμετρο για το roughness, την ταχύτητα που θα κινείται το material, καθώς προσθέτουμε και μερικά ακόμα χρώματα για να μπορούμε να το ρυθμίζουμε όσο πράσινο ή κίτρινο θέλουμε να είναι.

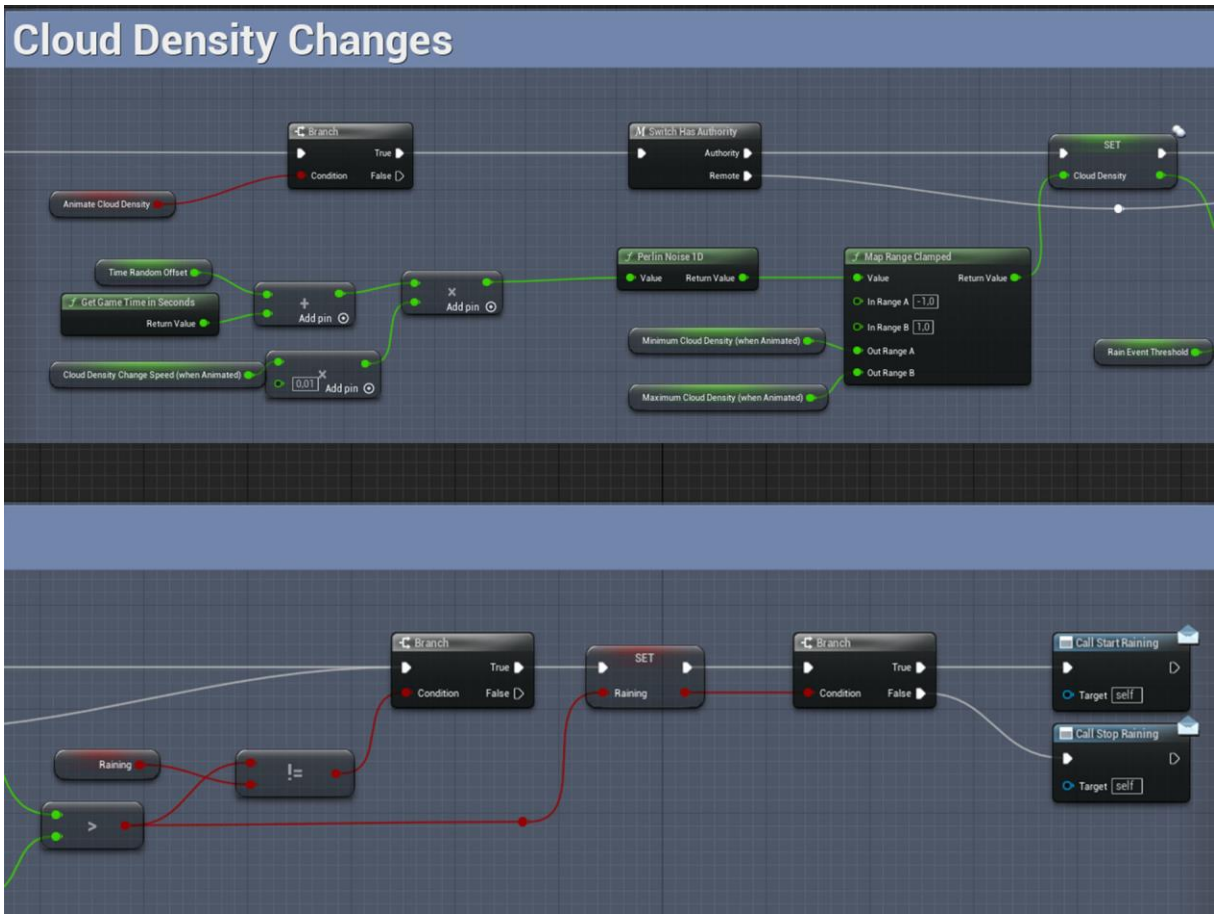
4.2.2. Blueprint ουρανού

Ένα από τα βασικά blueprints μέσα στο περιβάλλον μας είναι αυτό του ουρανού, το οποίο μάλιστα έχει και άμεση επίδραση φυσικά στον φωτισμό του περιβάλλοντος και επηρεάζει και το Lumen στο πως θα αποδώσει τον φωτισμό στον χώρο.

Για να πετύχουμε έναν ρεαλιστικό ουρανό με τα σύννεφα και τα αστέρια όπως φαίνονται στην παρακάτω Εικόνα, χρειάζεται να δημιουργήσουμε ένα blueprint όπου θα τοποθετήσουμε τα textures γι' αυτά, και θα δώσουμε την δυνατότητα στο blueprint να ελέγχει την κίνηση και την ορατότητα των σύννεφων, την ένταση του φωτός και το σημείο του ηλίου ώστε να νυχτώνει όταν κατεβαίνει ή να ξημερώνει όταν ανεβαίνει. Το τελευταίο επιτυγχάνεται δίνοντας την δυνατότητα στο blueprint να υπολογίζει την ένταση του ηλίου με βάση την ώρα ενός ρολογιού. Επομένως όταν είναι ακριβώς στο κέντρο του ουρανού αυτό μεταφράζεται σε δώδεκα η ώρα το μεσημέρι.

Εικόνα 4.15: Ο ουρανός του περιβάλλοντος μας

Για την πυκνότητα των σύννεφων ξεκινάμε με ένα event tick και από εκεί δίνουμε την δυνατότητα στο blueprint μέσα από branches να μπορεί να αλλάζει την πυκνότητα αλλά και την ταχύτητα των σύννεφων.

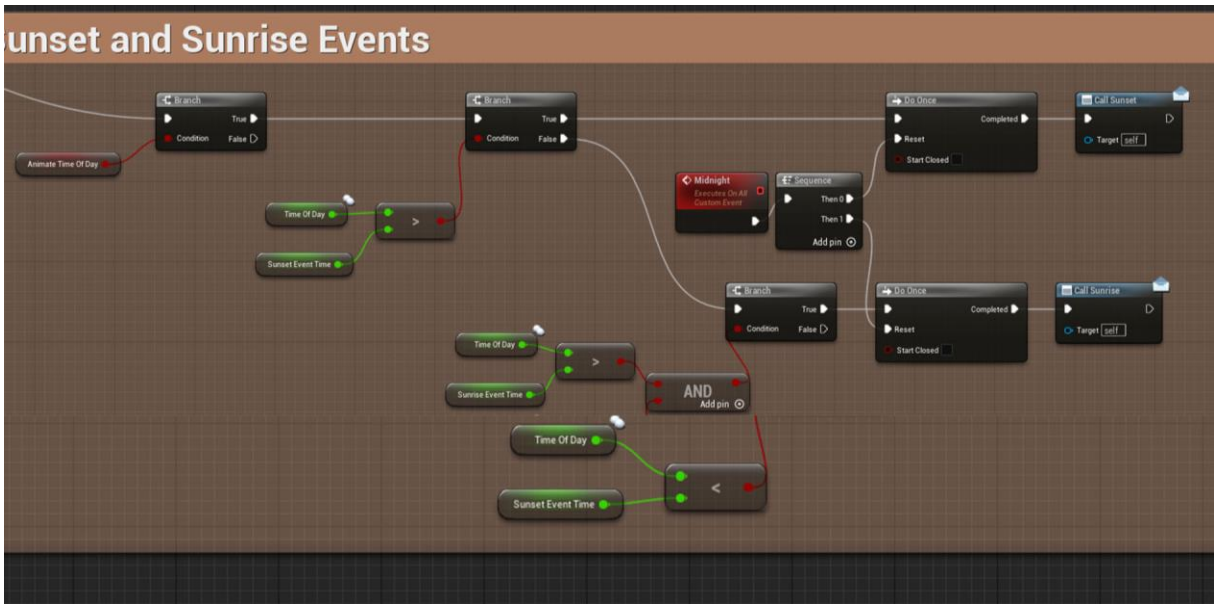


Εικόνα 4.16: Πυκνότητα και ταχύτητα σύννεφων

Έπειτα έχουμε τον κύκλο ημέρας και νύχτας όπου παίρνει την τοποθεσία του ηλίου και ανάλογα που θα τον γυρίσουμε, θα σκοτεινιάσει ή θα ξημερώσει. Ενώ παράλληλα υποστηρίζει και την πραγματική ώρα και κινείται από μόνο σε πραγματικό χρόνο.

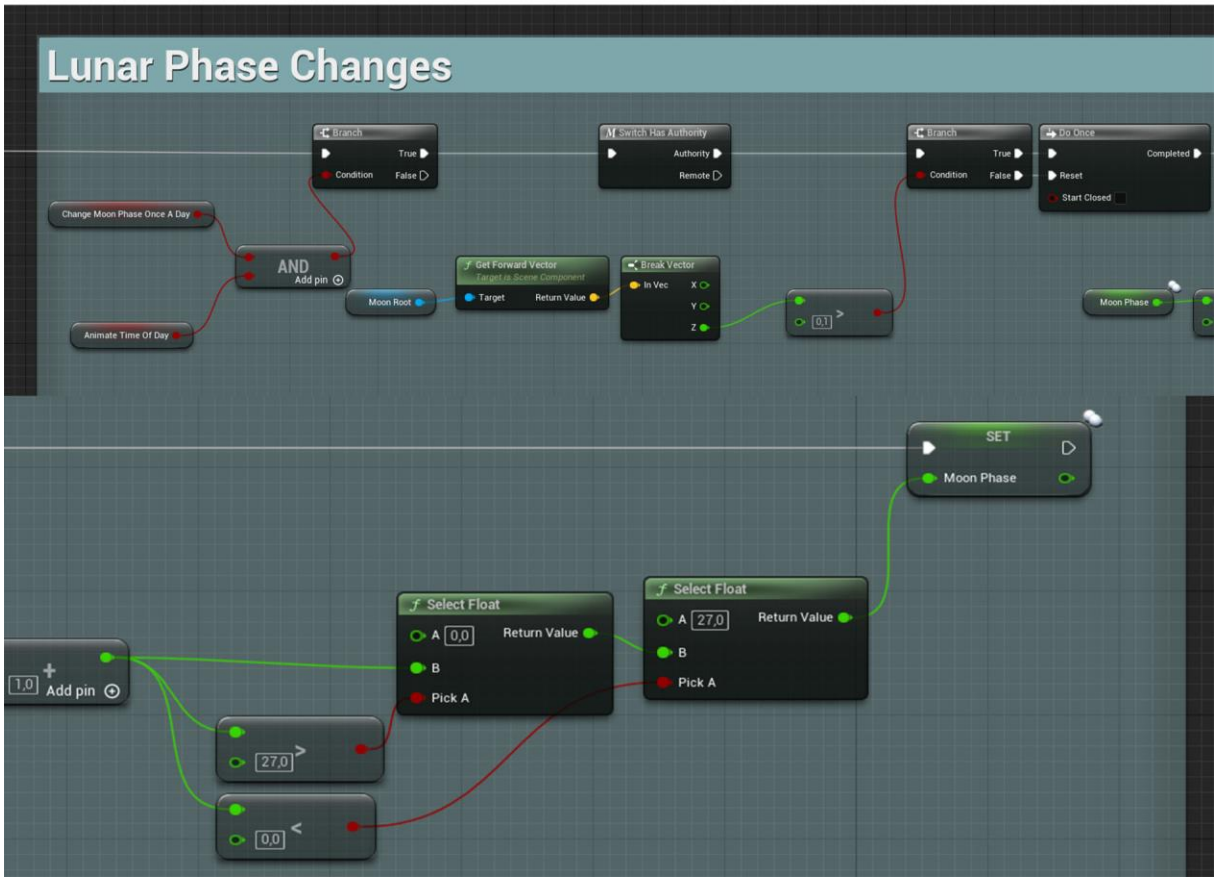
Εικόνα 4.17: Κύκλος Ημέρας/Νύχτας

Τέλος, έχουμε τα events για το ηλιοβασίλεμα και την ανατολή του ηλίου, για την πανσέληνο αλλά και την περιοδική φωτεινότητα του ουρανού.



Εικόνα 4.18: Ηλιοβασίλεμα, ανατολή

Εδώ μέσα από τα variables για την ώρα της ημέρας και του ηλιοβασιλέματος και της ανατολής, τα οποία παίρνουν την ώρα από τον υπολογιστή που χρησιμοποιούμε, το blueprint γνωρίζει πότε να δηλώσει αν είναι ηλιοβασίλεμα ή ανατολή.



Εικόνα 4.19: πανσέληνος

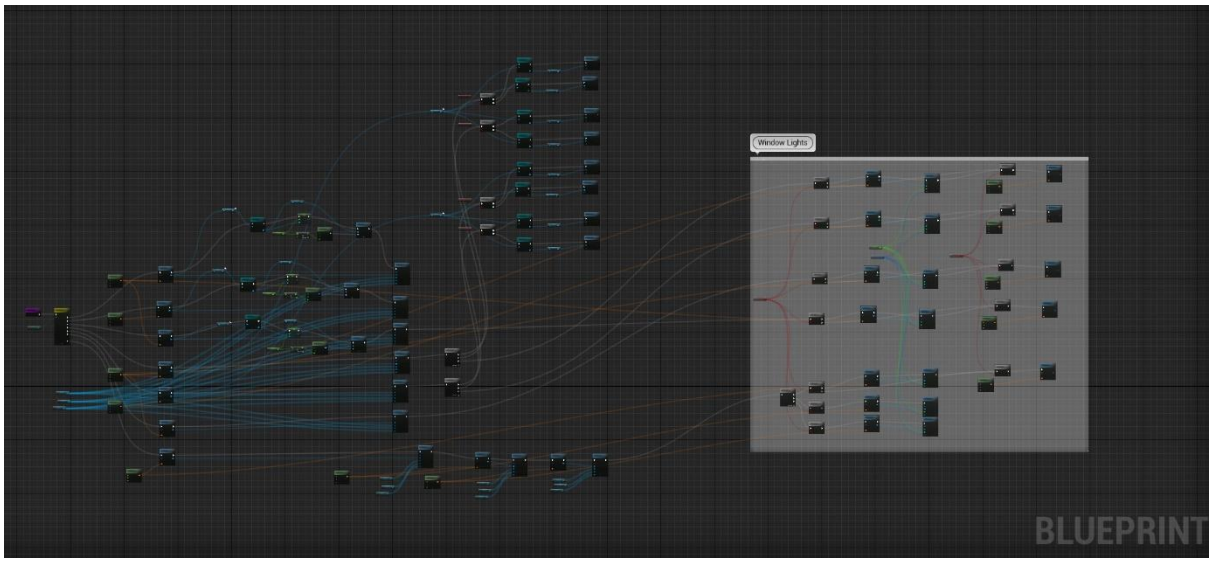
Για την πανσέληνο του δίνεται η τιμή 27 όπου ουσιαστικά υπολογίζεται σαν 27 ημέρες για να δώσει εντολή να εμφανίσει το φεγγάρι ολόκληρο για την πανσέληνο ενώ ενώνεται και με το variable της ημέρας και της νύχτας για να γνωρίζει πότε είναι βράδυ ώστε να εμφανίσει το φεγγάρι.

4.2.3. Σπίτι

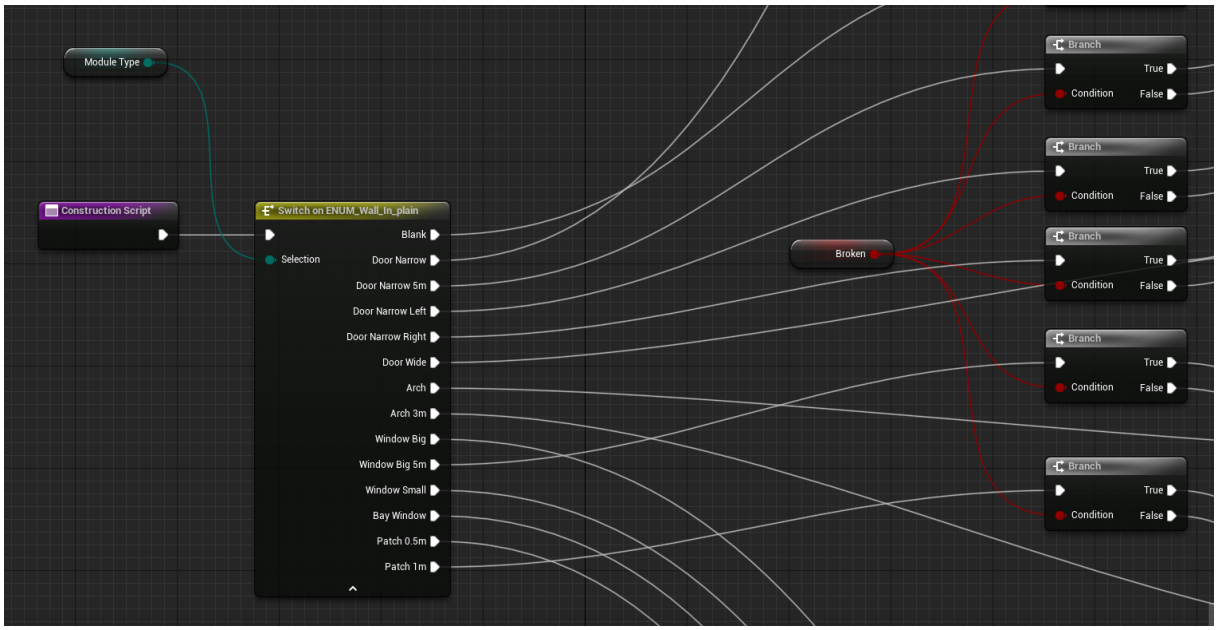
Για την δημιουργία του σπιτιού, χρησιμοποιήθηκαν αρκετά διαφορετικά assets όπου το καθένα έχει το δικό του custom blueprint ώστε να μπορεί να προσαρμόζεται ανάλογα όπως θέλουμε εμείς.

Για παράδειγμα, ένα κομμάτι τοίχου, με την βοήθεια των blueprints, έχει την δυνατότητα μέσα από της ρυθμίσεις του asset να επιλέξουμε αν θα έχει παράθυρο, αν θα είναι χαλασμένος τοίχος και άλλες τέτοιες παρόμοιες ρυθμίσεις. Αυτό επιτυγχάνεται με τα blueprints τα οποία δίνουν την δυνατότητα να δούμε οπτικά αυτές τις ρυθμίσεις μέσα από το UI της Unreal Engine και με μερικά απλά κλικ, να αποπλέξουμε ή να επιλέξουμε τις ρυθμίσεις που θέλουμε.

Παρακάτω βλέπουμε ένα αρκετά μεγάλο blueprint, όπου ξεκινάει από αριστερά, και το οποίο δίνει την δυνατότητα στον δημιουργό να μπορεί να επιλέξει αν θα προστεθεί στον τοίχο το asset του παραθύρου ή της πόρτας και έπειτα προς τα δεξία είναι τα blueprints όπου μπορούμε να ρυθμίσουμε τον φωτισμό που θα εισέρχεται από τα παράθυρα. Φυσικά, οι πόρτες και τα παράθυρα έχουν ξεχωριστά δικό τους blueprint για τις δικές τους λειτουργίες, όπως η πόρτα να ανοίγει και να κλείνει και το παράθυρο για το αν θα έχει στόρια ή όχι, ή αν θα έχει σπασμένα τζάμια.

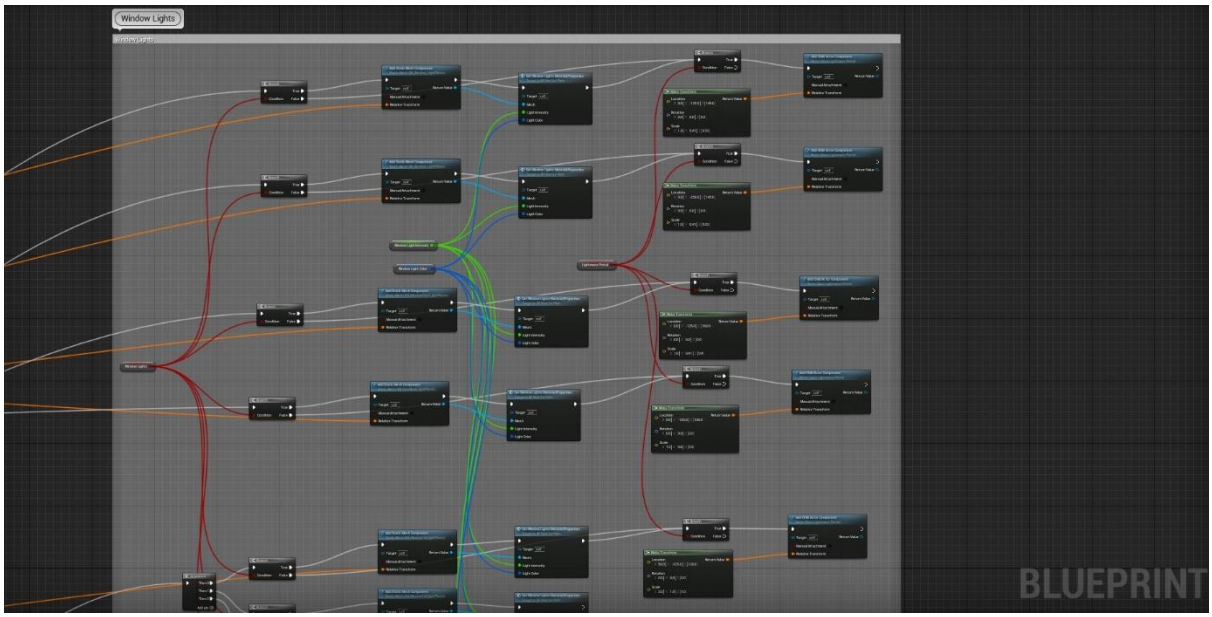


Εικόνα 4.20: το blueprint για τον τοίχο



Εικόνα 4.21: Enum switcher και branches

Για το script δίνεται ένα enum switcher το οποίο περνάει στα branches και από εκεί, ανάλογα τι ρυθμίσεις θα επιλέξει ο developer από το panel, αυτό θα ενεργοποιεί αναλόγως μέσα από τα branches την εκάστοτε λειτουργία. Για παράδειγμα, αν ο developer επιλέξει ο τοίχος να έχει παράθυρο, το enum θα περάσει στο branch όπου βρίσκεται η λειτουργία για την εμφάνιση του παραθύρου στον τοίχο.



Εικόνα 4.22: Οι ρυθμίσεις για τον φωτισμό των παραθύρων ενώ ενσωματώνονται στον τοίχο

Οι λειτουργίες που δίνονται στον τοίχο δεν είναι μόνο αυτές, αλλά το blueprint επιτρέπει, πάντα με μια επιλογή, να έχουμε έναν μεγαλύτερο τοίχο, ή ένα μικρότερο κομμάτι αυτού. Αυτό καθιστά πανεύκολη και γρήγορη την διαδικασία δημιουργίας του σπιτιού από τον προγραμματιστή, δίχως να καταναλώνει πολύτιμο χρόνο γι' αυτό, ο οποίος μπορεί να αξιοποιηθεί σε πολύ πιο σημαντικά σημεία του παιχνιδιού, όπως το optimization.

Με αυτό τον τρόπο, η δημιουργία του περιβάλλοντος γίνεται ακόμα πιο έξυπνα και πιο διασκεδαστικά, καθώς ουσιαστικά έχουμε δημιουργήσει έναν builder εντός της Unreal Engine.

Κεφάλαιο 5: Συγκριτική μελέτη λειτουργιών της Unreal Engine

Οι δύο τελευταίες εκδόσεις του Unreal Engine, η Unreal Engine 4 και η Unreal Engine 5, διαθέτουν αμφότερες εντυπωσιακές τεχνολογίες και δυνατότητες για τη δημιουργία video games και άλλων περιεχομένων. Παρόλα αυτά υπάρχουν κάποιες νέες λειτουργίες στην τελευταία έκδοση που αναμένεται να κάνουν την δημιουργία παιχνιδιών ακόμα πιο συναρπαστική και σίγουρα θα επωφελήσει πολύ τους προγραμματιστές.

Παρακάτω θα επιδιώξουμε αρχικά να δημιουργήσουμε μια σκηνή με πολλά triangles και polygons εντός της Unreal Engine 5, προκειμένου να γίνει μια δοκιμή χρήσης του Nanite και στην συνέχεια θα δούμε μια συγκριτική μελέτη μεταξύ της Unreal Engine 5 και της Unreal Engine 4, με πραγματικά αντικείμενα, όπως θα βλέπαμε και σε ένα παιχνίδι. Εκεί θα δούμε το Lumen και το Nanite σε πραγματικές συνθήκες πως ακριβώς δουλεύουν, και τα αποτελέσματα τους.

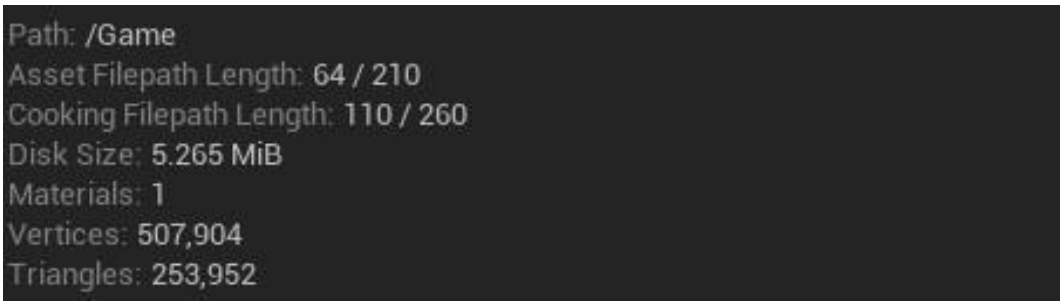
5.1. Σύγκριση με και χωρίς χρήση Nanite στην Unreal Engine 5

Η πρώτη σύγκριση αφορά το Nanite. Αυτό θα γίνει εντός της Unreal Engine 5 καθώς όταν το Nanite είναι απενεργοποιημένο, έχουμε το ίδιο αποτέλεσμα που θα είχαμε και στην Unreal Engine 4. Οι περιορισμοί της πολυγωνικής ανάλυσης δεν είναι πλέον πρόβλημα και όπως μπορούμε να δούμε παρακάτω στις εικόνες 5.2 και 5.4, η διαφορά στα FPS και γενικότερα στην απόδοση του project μας είναι εμφανής.

Τα frames δεν περιορίζονται πλέον από τον αριθμό των πολυμερών, όπως από το ZBrush και τα photogrammetry scans[52], τα οποία είναι η διαδικασία λήψης πολλών φωτογραφιών ενός αντικειμένου από διάφορες γωνίες και η ένωση τους τους για τη δημιουργία ενός τρισδιάστατου μοντέλου. Ενώ παράλληλα η απώλεια ποιότητας είναι σπάνια ή ανύπαρκτη, ειδικά με τις μεταβάσεις LOD.

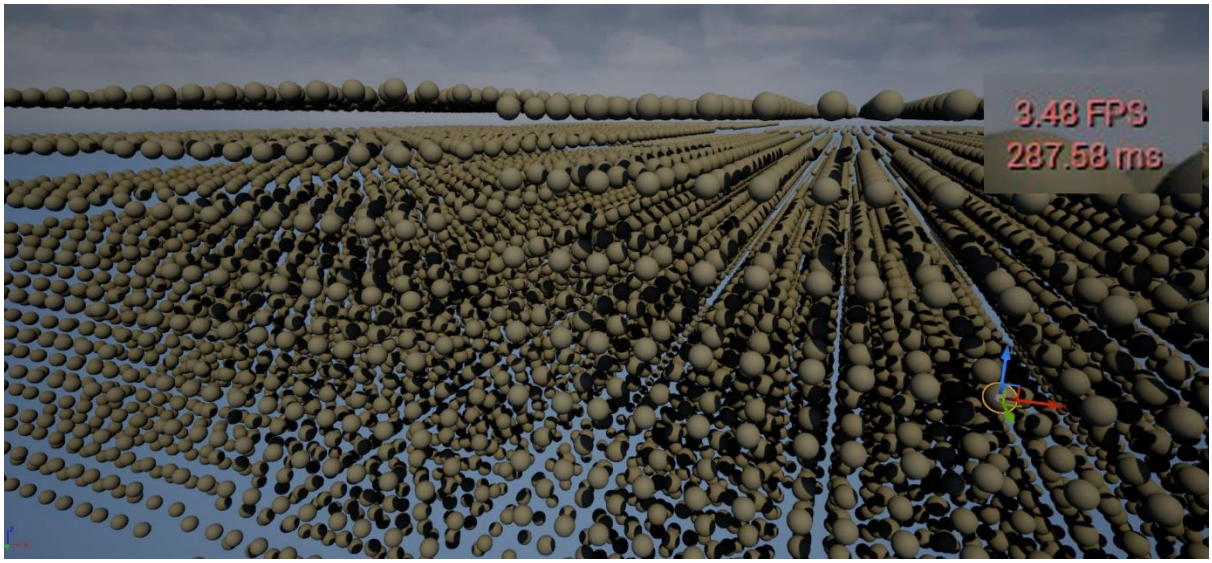
Όσο και να κινηθούμε στο περιβάλλον βλέπουμε ότι το Nanite καταφέρνει με επιτυχία να προσαρμόσει αυτόματα την γεωμετρία των αντικειμένων γύρω από τον χαρακτήρα μας ανάλογα με την απόσταση από την κάμερα που τον ακολουθεί, αυτό έχει ως αποτέλεσμα να διατηρεί σταθερά σε υψηλά επίπεδα τα FPS.

Για την ακόμα πιο ξεκάθαρη διαφορά στην απόδοση όταν χρησιμοποιούμε το Nanite, δημιουργήσαμε τυπικά μια σφαίρα μέσα από το Blender και του δώσαμε πολλές γωνίες και πλευρές προκειμένου τα triangles να είναι πάνω από 250 χιλιάδες και τα vertices πάνω από 500 χιλιάδες.



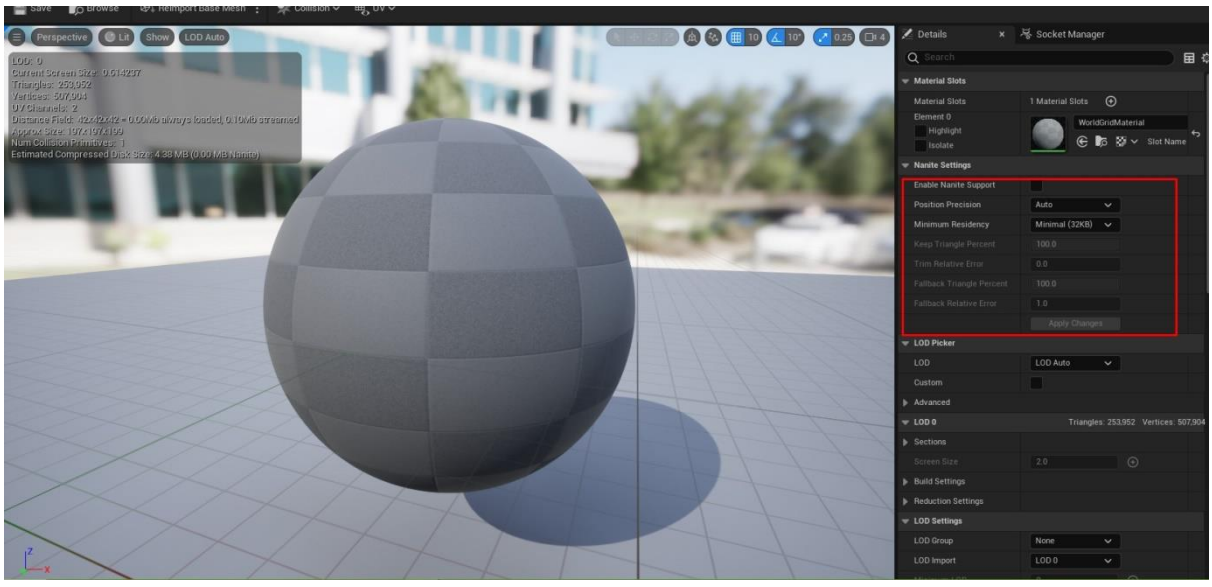
Εικόνα 5.1: Τα Vertices και τα Triangles της σφαίρας όπως δημιουργήθηκε στο Blender

Έπειτα δημιουργήσαμε ένα κενό περιβάλλον μέσα στην Unreal Engine 5 και πολλαπλασιάσαμε την σφαίρα μας πάνω από 20.000 φορές μέσα στο περιβάλλον. Αυτό είχε ως αποτέλεσμα την μείωση των fps στα 3.48, όπως θα δείτε παρακάτω. Το περιβάλλον μας αυτή τη στιγμή έχει εκατομμύρια δεδομένα για να επεξεργαστεί, επομένως τα FPS έχουν πέσει δραματικά, σε σημείο που ένα τέτοιο παιχνίδι δεν θα ήταν playable φυσικά.



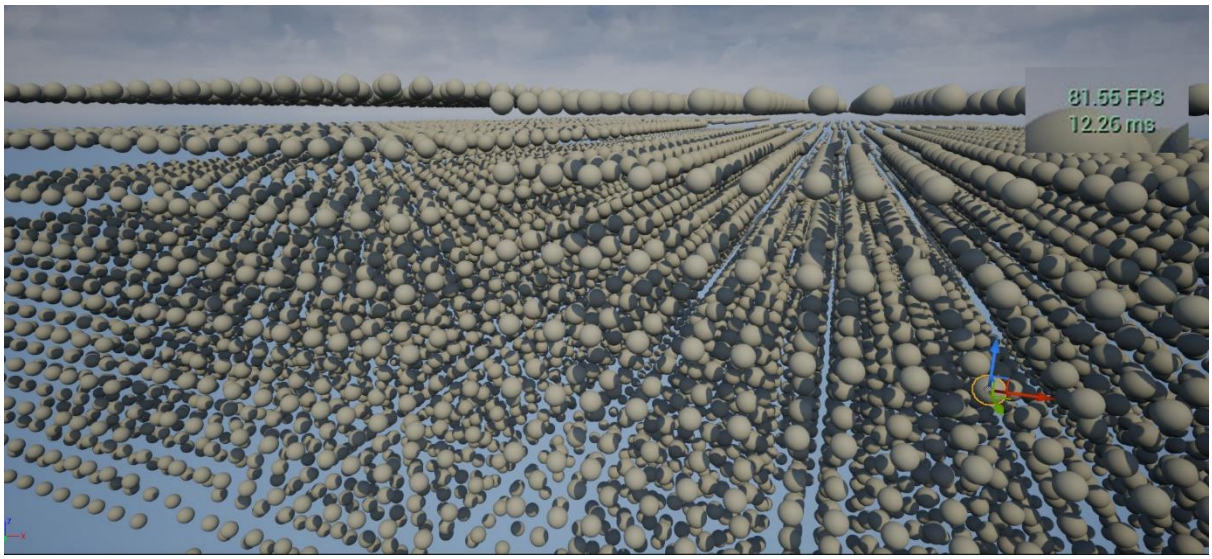
Εικόνα 5.2: Τα FPS πριν την ενεργοποίηση του Nanite

Με την ενεργοποίηση του Nanite, το οποίο γίνεται μέσα από την επεξεργασία του asset μας, εντός της Unreal Engine, και με ένα απλό κλικ, θα δούμε την εντυπωσιακή διαφορά που υπάρχει στα FPS από την ίδια ακριβώς οπτική γωνία, και με την ανάλυση να μην έχει μειωθεί καθόλου αλλά αντιθέτως να βελτιώνει και τον φωτισμό.



Εικόνα 5.3: Η ενεργοποίηση του Nanite

Όταν όμως ενεργοποιούμε το Nanite, ο πολλαπλασιασμός του αντικειμένου μας είναι σαν να μην έγινε ποτέ, και η απόδοση παραμένει σε υψηλά επίπεδα και ξεπερνάει τα 80 FPS.



Εικόνα 5.4: Μετά την ενεργοποίηση του Nanite

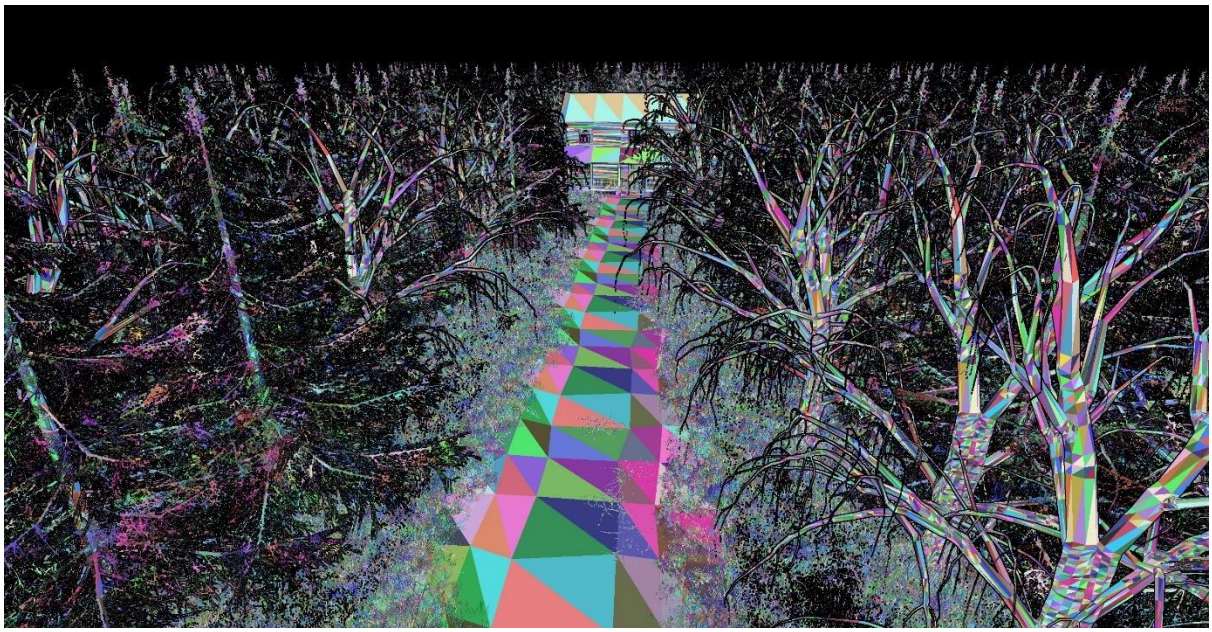
Χωρίς το Nanite μάλιστα, θα πρέπει να δημιουργήσουμε ένα low poly μοντέλο, να κάνουμε bake τις high poly λεπτομέρειες μέσα στον χάρτη και να τοποθετήσουμε το material. Έπειτα θα πρέπει να δημιουργήσουμε LODs, το οποίο είναι το επίπεδο λεπτομέρειας στο μοντέλο ανάλογα την απόσταση. Αντίθετα με το Nanite στην Unreal Engine 5 μπορούμε να χρησιμοποιήσουμε απευθείας το μεγάλης ανάλυσης μοντέλο μας και το εργαλείο θα κάνει την υπόλοιπη δουλειά για εμάς, μειώνοντας δραστικά τον χρόνο ανάπτυξης και διατηρώντας την ποιότητα του αντικειμένου μας σε υψηλά επίπεδα.

Ακόμα και σε ένα πραγματικό περιβάλλον όπως αυτό που δημιουργήσαμε παρακάτω, το nanite κάνει πραγματική δουλειά, αυξάνοντας αρκετά τα FPS χωρίς να βλέπουμε πραγματική διαφορά στην ποιότητα.

Εικόνα 5.5: Το Nanite ενεργοποιημένο

Εικόνα 5.6: Το Nanite απενεργοποιημένο

Στην παρακάτω Εικόνα βλέπουμε τα Triangles από τα assets τα οποία έχουν ενεργοποιημένο το nanite. Μάλιστα στις νέες εκδόσεις, το nanite υποστηρίζεται και στο foliage όπου εκεί τα φυτά, τα δέντρα κλπ έχουν και animation, παρόλα αυτά το nanite καταφέρνει να εφαρμοστεί και εκεί πλέον, μειώνοντας τον χρόνο φόρτωσης τους και αυξάνοντας συνεπώς τα FPS στο παιχνίδι.



Εικόνα 5.7: Αυτό είναι το Nanite view σε Triangles

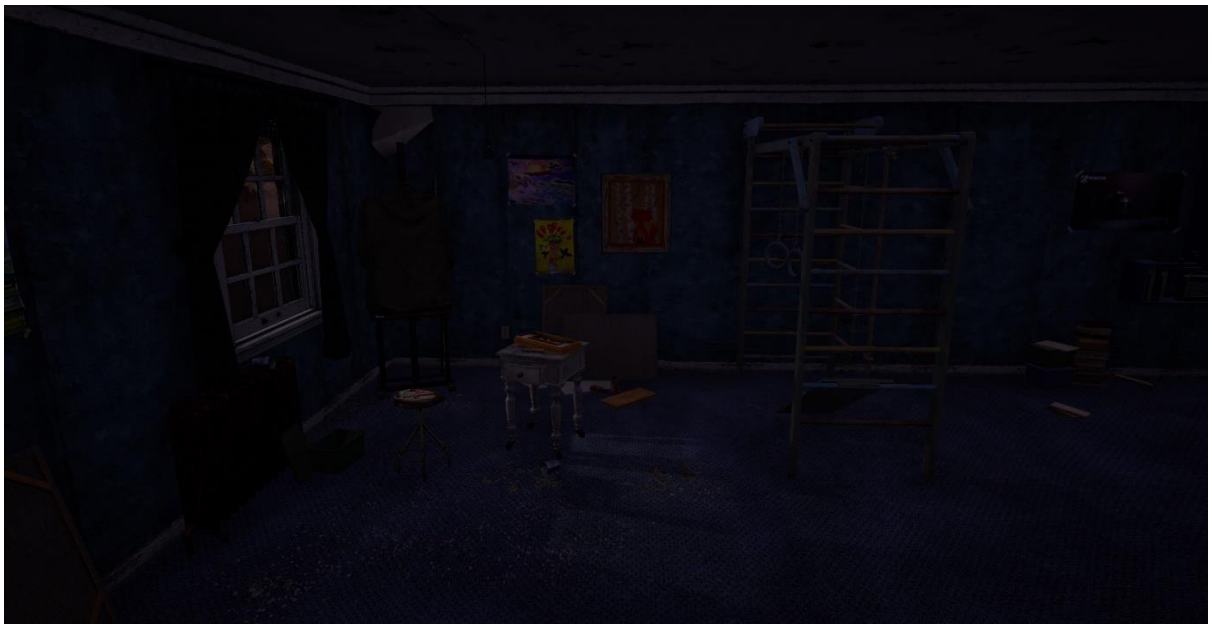
5.2. Χρήση του Lumen

Το Lumen πάλι, επιτρέπει στους προγραμματιστές να δημιουργήσουν πολύ πιο ρεαλιστικούς φωτισμούς και σκιές σε πραγματικό χρόνο. Στην τελευταία έκδοση βλέπουμε πως ο φωτισμός είναι πιο ρεαλιστικός, τα μαύρα ακόμα πιο βαθιά με αποτέλεσμα το σκοτάδι να αντικατοπτρίζεται πιο ρεαλιστικά πλέον.

Οι έλεγχοι βελτιστοποίησης για τη μείωση του μεγέθους των Nanite meshes στο δίσκο, επιτρέπουν τη βελτιστοποίηση των meshes κατά τη διάρκεια της ανάπτυξης, αφαιρώντας πρώτα τα λιγότερο σημαντικά δεδομένα, ώστε να μοιάζει περισσότερο με συμπίεση παρά με απώλεια.

Για παράδειγμα, στο συγκριτική μελέτη που έχουμε δημιουργήσει, μπορούμε να δούμε την διαφορά στον φωτισμό. Το σκοτάδι είναι πιο ρεαλιστικό και οι ακτίνες του ηλίου περνάνε ανάμεσα από τα δέντρα και τα πάραθυρα όπως θα γινόταν και στην πραγματικότητα.

Εικόνα 5.8: Σκοτεινό μέρος με εξωτερικό φως από το παράθυρο στην Unreal Engine 5



Εικόνα 5.9: Σκοτεινό μέρος με εξωτερικό φως από το παράθυρο στην Unreal Engine 4

Εικόνα 5.10: Σκοτεινό μέρος με εξωτερικό φως από το παράθυρο από άλλη οπτική γωνία στην Unreal Engine 5

Εικόνα 5.11: Σκοτεινό μέρος με εξωτερικό φως από το παράθυρο από άλλη οπτική γωνία στην Unreal Engine 4

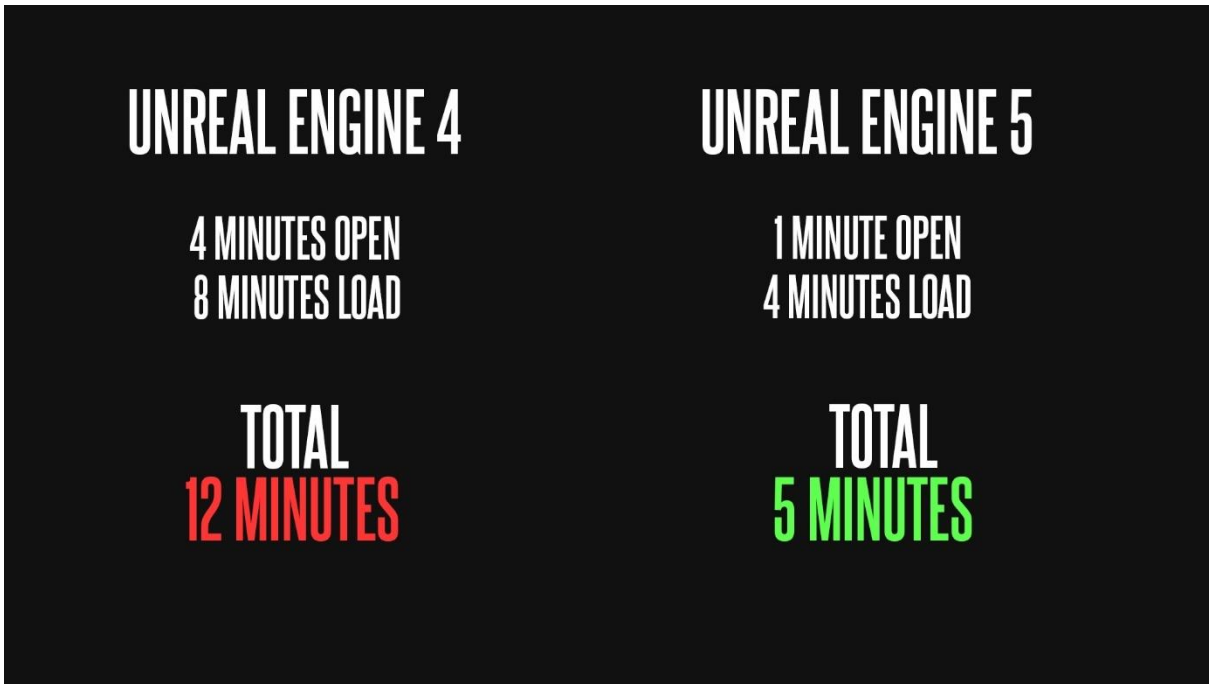
Στην Unreal Engine 4, είναι εμφανής η διαφορά του φωτισμού όπου εκεί δεν υπάρχει το Lumen συνεπώς δεν έχουμε και την φωτορεαλιστικότητα της νέας έκδοσης της μηχανής γραφικών.

5.3. Σύγκριση των Blueprints

Τα Blueprints επίσης έχουν γίνει ακόμα πιο γρήγορα. Στην 4η έκδοση της μηχανής γραφικών, το χάσμα μεταξύ των blueprints και της C++, όσον αφορά στην απόδοση, ήταν περίπου 10 δευτερόλεπτα διαφορά μεταξύ τους, και αυτό το αποτέλεσμα βγήκε από το πείραμα που πραγματοποιήσαμε, ανοίγοντας το ίδιο project και στην Unreal Engine 4 αλλά και στην Unreal Engine 5 και καταγράψαμε τους χρόνους φόρτωσης.

Δηλαδή, η C++ φόρτωνε πιο γρήγορα το περιβάλλον από τον κόσμο που δημιουργήσαμε και ολοκλήρωνε την εργασία του νωρίτερα. Πλέον στην 5η έκδοση της μηχανής, αυτό το χάσμα μειώθηκε στα περίπου 5 δευτερόλεπτα καθώς τα Blueprints έχουν γίνει πιο ευανάγνωστα, έχουν περισσότερες δυνατότητες με αποτέλεσμα να υπάρχουν ακόμα πιο εύκολοι τρόποι για να δημιουργήσουμε μια λειτουργία μέσα στα παιχνίδια και να μην χρειάζονται πολλά nodes, τα οποία επιβάρυναν την απόδοση του παιχνιδιού.

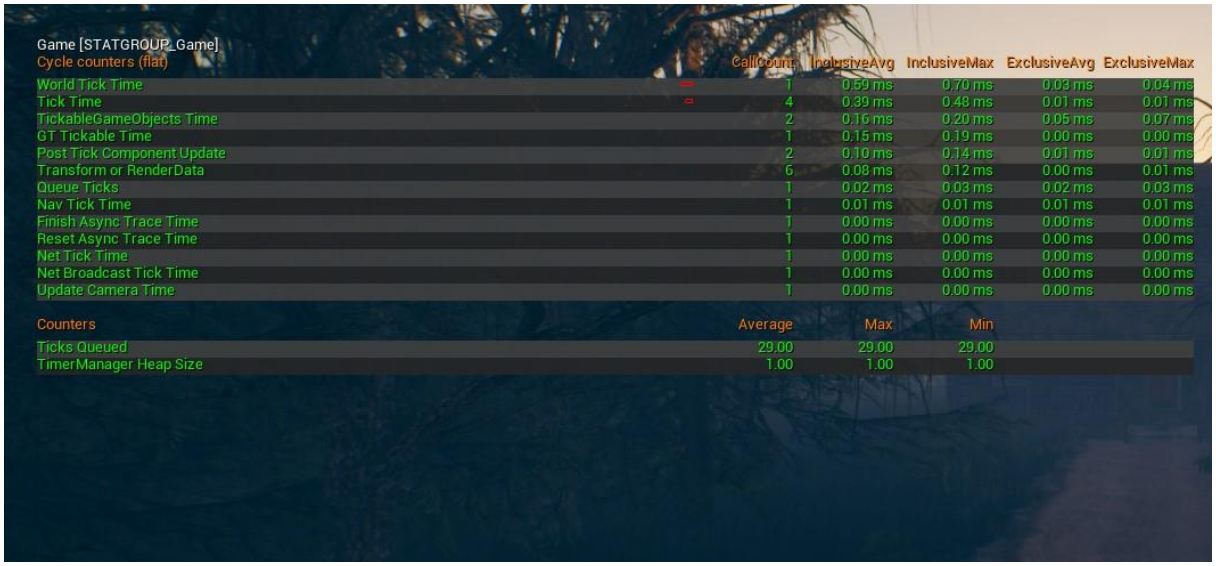
Επίσης η φόρτωση του project στην μηχανή γίνεται πολύ πιο γρήγορα, με ένα συγκριτικό ανάμεσα στην Unreal Engine 4 και στην Unreal Engine 5 παρακάτω, από το project μας, να δείχνει την τεράστια διαφορά στους χρόνους. Και αυτό επειδή κάνει καλύτερη κατανομή των δεδομένων από ότι η προηγούμενη έκδοση. Φυσικά μεγάλο ρόλο παίζει και το hardware που έχουμε, παρόλα αυτά τα νούμερα παρακάτω είναι από τον ίδιο υπολογιστή, ο οποίος είχε μια RTX 3080 και έναν Ryzen 9 5960 και 32GB ram.



Εικόνα 5.12: Συγκριτικά οι χρόνοι φόρτωσης ανάμεσα στις δύο μηχανές

Στην παρακάτω Εικόνα, βλέπουμε πως η Unreal Engine 4 κάνει render τα δεδομένα και πόσο καθυστερεί σε αντίθεση με την Unreal Engine 5. Η διαφορά και εδώ είναι εμφανής.

Εικόνα 5.13: Render data σε Unreal Engine 4



Εικόνα 5.14: Render data σε Unreal Engine 5

Λόγω του Nanite, η Unreal Engine 5 έχει πολύ λιγότερα RenderData, από 135 μειώθηκαν στα 6, ενώ όλοι οι χρόνοι είναι πολύ πιο κάτω από το μισό απ' ότι στην Unreal Engine 4. Αυτό έχει ως αποτέλεσμα το πολύ καλύτερο optimization στο παιχνίδι, χάρη στα βελτιωμένα blueprints με τους γρηγορότερους χρόνους ανάγνωσης και στο Nanite όπου συμμετέχει σε μεγάλο βαθμό σε αυτή την βελτιστοποίηση.

Κεφάλαιο 6: Συμπεράσματα

Η Unreal Engine 5 αποτελεί ένα πρωτοποριακό ορόσημο στη βιομηχανία ανάπτυξης παιχνιδιών, συνδυάζοντας τεχνολογίες όπως το Nanite και το Lumen με τη δύναμη και την προσβασιμότητα των Blueprints.

Το Nanite, με το virtual geometry system, επιτρέπει να υπάρχουν υψηλά επίπεδα λεπτομέρειας στα περιβάλλοντα των παιχνιδιών. Εισαγωγή των assets με υψηλή ανάλυση είναι πλέον πιο εφικτή και χωρίς να θυσιάζουν τις επιδόσεις, με αποτέλεσμα να δημιουργούνται οπτικά εντυπωσιακοί και ρεαλιστικοί κόσμοι που μέχρι πρότινος ήταν δυνατοί μόνο σε pre-rendered cinematics. Με το Nanite, τα όρια στενεύουν στο μεταξύ των γραφικών πραγματικού χρόνου και των εικόνων που έχουν σκηνοθετηθεί εκ των προτέρων.

Φυσικά, η τεράστια εξέλιξη των Blueprints λειτουργεί ως κρίσιμος καταλύτης για την καινοτομία και την προσβασιμότητα. Τα blueprints απλοποιούν πλέον ακόμα περισσότερο τη διαδικασία ανάπτυξης, και δίνεται ακόμα μεγαλύτερη ελευθερία στο να δημιουργηθούν σύνθετοι μηχανισμοί και interactions χωρίς να βασίζονται αποκλειστικά στον παραδοσιακό προγραμματισμό.

Συνδυάζοντας το Nanite, το Lumen και τα Blueprints, η Unreal Engine 5 όχι μόνο ενδυναμώνει τους έμπειρους developers, αλλά ανοίγει και τις πόρτες για τους νεοεισερχόμενους να συνεισφέρουν τις μοναδικές ιδέες και τα οράματα τους στο τοπίο των videogames.

Η έμφαση της μηχανής στην προσβασιμότητα και τα φιλικά προς τον χρήστη εργαλεία ανοίγουν τον δρόμο για ένα πιο συνεργατικό και δημιουργικό μέλλον, όπου οι developers μπορούν να δημιουργήσουν οπτικά εκπληκτικά παιχνίδια. Καθώς η βιομηχανία των videogames συνεχίζει να εξελίσσεται, η Unreal Engine 5 βρίσκεται στην πρώτη γραμμή, εμπνέοντας την καινοτομία και προωθώντας το μέσο σε νέα ύψη. Και με βάση την συγκριτική μελέτη που έγινε στην παρούσα εργασία αποδεικνύονται τα οφέλη των νέων τεχνολογιών Lumen και Nanite, αλλά και της εξέλιξης των blueprints καθιστώντας την Unreal Engine 5 μια από τις πιο δυναμικές μηχανές γραφικών έως σήμερα.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Brookhaven National laboratory, “The First Video Game?”, *Brookhaven Natioal Laboratory*. [Online]. Διαθέσιμο: https://www.bnl.gov/about/history/firstvideo.php
[2] Business Insider, “What is Epic Games?”, *Vivian McCall*, Nov 13, 2020. [Online]. Διαθέσιμο:https://www.businessinsider.com/guides/tech/what-is-epic-games
[3] Extern Labs, “Unreal Engine and its Evolution”, *Gaurav Meena*, April 12, 2023. [Online] Διαθέσιμο:https://externlabs.com/blogs/unreal-engine-and-its-evolution/
[4] UDK Documentation, “Per-object dynamic shadows”, *Epic Games*. [Online] Διαθέσιμο:https://docs.unrealengine.com/udk/Three/ShadowingReference.html
[5] Aalto University, School of Science, “How Virtual Reality Meets the Industrial IoT”, Chapter 2 “VR/AR engines”, *Alberto Vaccari*. [Online] Διαθέσιμο:https://wiki.aalto.fi/download/attachments/109392027/How-VR-meets-IIoT.pdf
[6] Logic Simplified, “Unreal 5: Nanite and Lumen Technology for Next-Gen Games”, *Logic Simplified*. [Online]. Διαθέσιμο: https://logicsimplified.com/newgames/unreal-5-nanite-and-lumen-technology-for-next-gen-games/
[7] Anchorpoint, “How to collaborate in Unreal Engine 5”, *Matthaus Niedoba*, January 12, 2023. [Online] Διαθέσιμο: https://www.anchorpoint.app/blog/how-to-collaborate-in-unreal-engine-5
[8] Unreal Engine Documentation, “Architecture”, *Epic Games*. [Online]. Διαθέσιμο:https://www.unrealengine.com/en-US/solutions/architecture
[9] Unreal Engine Documentation, “Augmented Reality Overview”, *Epic Games*. [Online]. Διαθέσιμο:https://docs.unrealengine.com/4.26/en-US/SharingAndReleasing/XRDevelopment/AR/HandheldAR/AROverview/
[10] Unreal Engine Documentation, “Global Illumination”, *Epic Games*. [Online]. Διαθέσιμο:https://docs.unrealengine.com/5.0/en-US/global-illumination-in-unreal-engine/
[11] Unreal Engine Documentation, “Nanite Virtualized Geometry”, *Epic Games*. [Online]. Διαθέσιμο:https://docs.unrealengine.com/5.0/en-US/nanite-virtualized-geometry-in-unreal-engine/
[12] Unreal Engine Documentation, “MetaSounds: The Next Generation Sound Sources”, *Epic Games*. [Online]. Διαθέσιμο: https://docs.unrealengine.com/5.1/en-US/metasounds-the-next-generation-sound-sources-in-unreal-engine/
[13] Unreal Engine Documentation, “Audio Engine Overview”, *Epic Games*. [Online]. Διαθέσιμο:https://docs.unrealengine.com/5.1/en-US/audio-engine-overview-in-unreal-engine/
[14] Unreal Engine Documentation, “Creating Visual Effects”, *Epic Games*. [Online]. Διαθέσιμο:https://docs.unrealengine.com/5.2/en-US/creating-visual-effects-in-niagara-for-unreal-engine/
[15] Unreal Engine Documentation, “Simulating Physics”, *Epic Games*. [Online]. Διαθέσιμο:https://docs.unrealengine.com/4.26/en-US/Resources/ContentExamples/Physics/1_1/
[16] Unreal Engine Documentation, “Level Editor”, *Epic Games*. [Online]. Διαθέσιμο:https://docs.unrealengine.com/5.1/en-US/level-editor-in-unreal-engine/
[17] Unreal Engine Documentation, “Blueprints Visual Scripting”, *Epic Games*. [Online]. Διαθέσιμο:https://docs.unrealengine.com/5.1/en-US/blueprints-visual-scripting-in-unreal-engine/
[18] Unreal Engine Documentation, “Materials”, *Epic Games*, [Online]. Διαθέσιμο:https://docs.unrealengine.com/5.1/en-US/unreal-engine-materials/
[19] Unreal Engine Documentation, “Collision Overview”, *Epic Games*. [Online]. Διαθέσιμο:https://docs.unrealengine.com/4.27/en-US/InteractiveExperiences/Physics/Collision/Overview/

[20] Unreal Engine Documentation, "RigidBody", *Epic Games.* [Online]. Διαθέσιμο:https://docs.unrealengine.com/5.0/en-US/animation-blueprint-rigid-body-in-unreal-engine/
[21] Unreal Engine Documentation, "Nodes", *Epic Games.* [Online]. Διαθέσιμο:https://docs.unrealengine.com/5.1/en-US/nodes-in-unreal-engine/
[22] Unreal Engine Documentation, "Blueprint Debugging", *Epic Games.* [Online]. Διαθέσιμο:https://docs.unrealengine.com/5.1/en-US/blueprint-debugging-in-unreal-engine/
[23] 1000 Forms of Bunnies(Victor's tech art blog), "Unreal Engine Custom Node", *Victor Li,* February 1, 2022. [Online]. Διαθέσιμο:https://viclw17.github.io/2022/02/01/unreal-engine-custom-node
[24] Unreal Engine Documentation, "Blueprint Debugging", *Epic Games".* [Online]. Διαθέσιμο:https://docs.unrealengine.com/5.1/en-US/blueprint-debugging-in-unreal-engine/
[25] Couch Learn, "Enums in Unreal Engine 4 Blueprints", *Matt,* July 14, 2020. [Online]. Διαθέσιμο:https://couchlearn.com/enums-in-unreal-engine-4-blueprints/
[26] Unreal Engine Documentation, "Custom Events", *Epic Games.* [Online]. Διαθέσιμο:https://docs.unrealengine.com/4.27/en-US/ProgrammingAndScripting/Blueprints/UserGuide/Events/Custom/
[27] Realtime Rendering, "Nanite: A Deep Dive", *Brian Karis, Rune Stubbe, Graham Wihlidal,* 2021. [Online].
Διαθέσιμο:https://advances.realtimerendering.com/s2021/Karis_Nanite_SIGGRAPH_Advances_2021_final.pdf
[28] Unreal Engine Blog, "Unreal Engine 5's new virtualized geometry system", *Epic Games,* June 5, 2021. [Online]. Διαθέσιμο: https://www.unrealengine.com/en-US/blog/understanding-nanite---unreal-engine-5-s-new-virtualized-geometry-system
[29] All3DP, "Blender: Reduce Polygons – Simply Explained", *Blu Siber,* August 5, 2022. [Online]. Διαθέσιμο: https://all3dp.com/2/blender-how-to-reduce-polygons/
[30] Unreal Engine Documentation, "Creating and Using LODs", *Epic Games.* [Online]. Διαθέσιμο: https://docs.unrealengine.com/5.1/en-US/creating-and-using-lods-in-unreal-engine/
[31] Unreal Engine Blog, "Large worlds in UE5: A whole new (open) world", *Arran Langmead,* September 28, 2021. [Online]. Διαθέσιμο: https://www.unrealengine.com/en-US/blog/large-worlds-in-ue5-a-whole-new-open-world
[32] Intel Blog, "How to Properly Balance Your Components", *Intel.* [Online]. Διαθέσιμο: https://www.intel.com/content/www/us/en/gaming/resources/what-is-bottlenecking-my-pc.html
[33] Chris McCole Blog, "Culling in UE4/UE5 (Precomputed Visibility Volumes, and Cull Distance Volumes)", *Chris McCole,* March 22, 2023. [Online]. Διαθέσιμο: https://www.chrismccole.com/blog/culling-in-ue4ue5
[34] Unreal Engine Documentation, "Dynamic Resolution", *Epic Games.* [Online]. Διαθέσιμο: https://docs.unrealengine.com/5.1/en-US/dynamic-resolution-in-unreal-engine/
[35] VK Guide Documentation, "GPU Driven Rendering", *Vulkan Guide.* [Online]. Διαθέσιμο: https://vkguide.dev/docs/gpudriven/gpu_driven_engines/
[36] NVIDIA Developer's Blog, "Introduction to Turing Mesh Shaders", *Christoph Kubisch,* September 17, 2018. [Online]. Διαθέσιμο: https://developer.nvidia.com/blog/introduction-turing-mesh-shaders/
[37] NVIDIA Documentation, "Nsight Graphics Activities - Advanced Learning", *NVIDIA.* [Online]. Διαθέσιμο: https://docs.nvidia.com/nsight-graphics/AdvancedLearning/index.html
[38] Unreal Engine Documentation, "Static Meshes", *Epic Games.* [Online]. Διαθέσιμο: https://docs.unrealengine.com/5.2/en-US/static-meshes/
[39] Unreal Engine Forums, "What is Baking?", *Unreal Engine Forums*. [Online]. Διαθέσιμο: https://forums.unrealengine.com/t/what-is-baking/110149

[40] Unreal Engine Documentation, “Movable Lights”, *Epic Games*. [Online]. Διαθέσιμο: https://docs.unrealengine.com/4.27/en-US/BuildingWorlds/LightingAndShadows/LightMobility/DynamicLights/
[41] Unreal Engine Documentation, “Directional Light”, *Epic Games*. [Online]. Διαθέσιμο: https://docs.unrealengine.com/4.27/en-US/Resources/ContentExamples/Lighting/1_3/
[42] Unreal Engine Documentation, “Indirect Lighting”, *Epic Games*. [Online]. Διαθέσιμο: https://docs.unrealengine.com/4.27/en-US/Resources/ContentExamples/Lighting/4_3/
[43] Unreal Engine Documentation, “Lightmass Basics”, *Epic Games*. [Online]. Διαθέσιμο: https://docs.unrealengine.com/5.0/en-US/lightmass-basics-in-unreal-engine/
[44] Unreal Engine Documentation, “Understanding Lightmapping in Unreal Engine”, *Epic Games*. [Online]. Διαθέσιμο: https://docs.unrealengine.com/5.2/en-US/understanding-lightmapping-in-unreal-engine/
[45] Unreal Engine Documentation, “Volumetric Lightmaps”, *Epic Games*. [Online]. Διαθέσιμο: https://docs.unrealengine.com/5.1/en-US/volumetric-lightmaps-in-unreal-engine/
[46] Unreal Engine Documentation, “Reflections Environment”, *Epic Games*. [Online]. Διαθέσιμο: https://docs.unrealengine.com/5.1/en-US/reflections-environment-in-unreal-engine/
[47] Unreal Engine Documentation, “Lumen Global Illumination and Reflections”, *Epic Games*. [Online]. Διαθέσιμο: https://docs.unrealengine.com/5.0/en-US/lumen-global-illumination-and-reflections-in-unreal-engine/
[48] Digital Trends, “What is ray tracing, and how will it change games?”, *Jon Martindale*, September 20, 2022. [Online]. Διαθέσιμο: https://www.digitaltrends.com/computing/what-is-ray-tracing/
[49] Imagej, “Voxelization”, *Imagej*. [Online]. Διαθέσιμο: https://imagej.net/imaging/voxelization
[50] Unreal Engine Documentation, “Lumen Technical Details”, *Epic Games*. [Online]. Διαθέσιμο: https://docs.unrealengine.com/5.0/en-US/lumen-technical-details-in-unreal-engine/
[51] UWA4D, “UE5 Lumen Implementation Analysis”, *UWA4D*, January 25, 2022. [Online]. Διαθέσιμο: https://blog.en.uwa4d.com/2022/01/25/ue5-lumen-implementation-analysis/
[52] Vntana, “3D Scanning And Photogrammetry Explained”, *Ben Conway*, December 16, 2020. [Online]. Διαθέσιμο: https://www.vntana.com/blog/3d-scanning-and-photogrammetry-explained/