ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Συγκριτική Μελέτη Unreal Engine και Unity: Δυνατότητες, Υλοποίηση Παιχνιδιών και Κατεύθυνση για Νέους Developers

Του φοιτητή

Λεοπάρντι Μάρκου-Αντώνιου

Αρ. Μητρώου: 154484

Επιβλέπων Καθηγητής

Κοκκώνης Γεώργιος

Επίκουρος Καθηγητής

Τίτλος Π.Ε. Συγκριτική Μελέτη Unreal Engine και Unity: Δυνατότητες, Υλοποίηση Παιχνιδιών και Κατεύθυνση για Νέους Developers

Κωδικός Π.Ε. 24143

Ονοματεπώνυμο φοιτητή/των Λεοπάρντι Μάρκος-Αντώνιος

Ονοματεπώνυμο εισηγητή Κοκκώνης Γεώργιος

Ημερομηνία ανάληψης Π.Ε. 08/03/2024

Ημερομηνία περάτωσης Π.Ε. 08/09/2025

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.

Πρόλογος

Από αρκετά μικρή ηλικία, κατάλαβα ότι πέρα από το απλά να παίζω, ασυνείδητα σκεφτόμουν την λογική που μπορούσε να υπάρχει πίσω από αυτό. Με το πέρασμα του χρόνου και τα βιντεοπαιχνίδια να γίνονται ολοένα και πιο όμορφα, γοητευόμουν στην ιδέα του να δημιουργήσω και εγώ ένα δικό μου.

Αποφάσισα έτσι να σπουδάσω πληροφορική και να εκπονήσω μια πτυχιακή που θα μου δώσει την ευκαιρία να εμβαθύνω στην ανάπτυξη βιντεοπαιχνιδιών.

Το πρώτο πρόβλημα που αντιμετώπισα ήταν η επιλογή της μηχανής, με την οποία ήθελα να ασχοληθώ. Ψάχνοντας κατέληξα στις 2 επιλογές Unity και Unreal Engine. Οι 2 πιο δυνατές και δωρεάν μηχανές για πολλά χρόνια. Και πάλι όμως μεταξύ των 2, η επιλογή ήταν δύσκολη, χωρίς να υπάρχει κάποιος ξεκάθαρος νικητής. Στην αναζήτηση μου, παρατήρησα ότι οι απόψεις διίστανται. Άλλοι χρησιμοποιούσαν την μια, ενώ άλλοι την άλλη.

Εμβαθύνοντας, κατάλαβα ότι πολλοί παράγοντες παίζουν ρόλο στην επιλογή της μηχανής που θα χρησιμοποιήσει κάποιος. Από την πλατφόρμα που στοχεύει, την ποιότητα τον γραφικών, την ευχέρεια των δημιουργών σε κώδικα, το μέγεθος της ομάδας και πολλούς ακόμα λόγους.

Η επιλογή του θέματος έγινε και για την προσωπική ανάπτυξη των δεξιοτήτων μου στις 2 μηχανές αλλά και στην συγκέντρωση πληροφοριών και ανάλυση τους, ώστε να μπορεί να γίνει πιο εύκολη η επιλογή μηχανής σε κάποιον που θέλει να ενταχθεί στον κόσμο του game development, και δεν ξέρει από που να ξεκινήσει.

Περίληψη

Αυτή η εργασία έχει ως σκοπό της, την ανάλυση και έπειτα την σύγκριση της Unreal Engine και της Unity με βάση τις παρακάτω πτυχές, με στόχο την βοήθεια κάποιου νέου Developer να κατανοήσει τις δυνατότητες τους στην καλύτερη λήψη απόφασης του, ως προς την επιλογή μηχανής που θα κάνει, για την ανάπτυξη ενός δικού του project:

- **Ευκολία Χρήσης και Εκμάθησης**: Ανάλυση και εξέταση της καμπύλης εκμάθησης των 2 μηχανών, της χρηστικότητας των τεχνολογιών και εργαλείων και των βιβλιοθηκών που προσφέρουν για την ανάπτυξη μιας τέτοιας εφαρμογής.
- **Ποιότητα και Ρεαλιστικότατα Γραφικών:** Αξιολόγηση της ποιότητας γραφικών και του ρεαλισμού του δημιουργημένου περιβάλλοντος και των μοντέλων οδήγησης που μπορούν να παραχθούν στην κάθε μηχανή.
- **Απόδοση και Αποτελεσματικότητα:** Σύγκριση της απόδοσης των παιχνιδιών οδήγησης, έχοντας υπόψη παράγοντες όπως ρυθμός καρέ και την χρήση πόρων.
- **Υποστήριξη και Κοινότητα:** Αξιολόγηση της βοήθειας που υπάρχει από την κοινότητα και τους ίδιους τους κατασκευαστές ως προς τους διαθέσιμους πόρους, εκπαιδευτικό υλικό και βοηθημάτων.

Η παραπάνω ανάλυση και σύγκριση, θα συνοδευτεί από την ανάπτυξη ενός παιχνιδιού και στις 2 πλατφόρμες. Τα τελικά προϊόντα θα χρησιμοποιηθούν για την εκτίμηση των προαναφερθέντων παραγόντων και την καλύτερη και σωστότερη εξαγωγή συμπερασμάτων σχετικά με την καταλληλόλητα κάθε μηχανής για την ανάπτυξη τέτοιων έργων.

Abstract

Comparative Study of Unreal Engine and Unity: Capabilities, Game Implementation, and Guidance for New Developers This project aims to analyze and subsequently compare Unreal Engine and Unity based on the following aspects, with the goal of helping a new developer understand their capabilities and make a more informed decision regarding the engine to choose for developing their own project:

- **Ease of Use and Learning Curve**: An analysis and evaluation of each engine's learning curve, usability of tools and technologies, and the libraries provided for the development of such an application.
- **Graphics Quality and Realism**: An assessment of the visual quality and the realism of the created environments and driving models that can be produced within each engine.
- **Performance and Efficiency**: A comparison of the performance of driving games developed with each engine, considering factors such as frame rate and resource usage.
- **Support and Community**: An evaluation of the available assistance from both the community and the engine developers, including resources, documentation, tutorials, and learning materials.

This analysis and comparison will be accompanied by the development of a game in both platforms. The final products will be used to assess the aforementioned factors and to draw more accurate conclusions regarding the suitability of each engine for developing such projects.

Περιεχόμενα

Κατάλογος Εικόνων

Κατάλογος Διαγραμμάτων

Κατάλογος Πινάκων

Κεφάλαιο 1ο Εισαγωγή

Η βιομηχανία της ανάπτυξης ψηφιακών παιχνιδιών τα τελευταία χρόνια γνωρίζει αλματώδη πρόοδο και αποτελεί έναν από τους ταχύτερα αναπτυσσόμενους τομείς στον χώρο της τεχνολογίας και της ψυχαγωγίας. Τα ψηφιακά παιχνίδια δεν αποτελούν πλέον απλώς μέσο διασκέδασης, αλλά συνιστούν ένα πολυσύνθετο προϊόν που συνδυάζει τέχνες, επιστήμες υπολογιστών, τεχνητή νοημοσύνη, σχεδίαση αλληλεπιδραστικών περιβαλλόντων και αφήγηση ιστοριών. Στο πλαίσιο αυτό, οι σύγχρονες μηχανές ανάπτυξης παιχνιδιών (game engines) αποτελούν τον ακρογωνιαίο λίθο της δημιουργικής διαδικασίας, καθώς προσφέρουν στον προγραμματιστή και τον σχεδιαστή μια ολοκληρωμένη εργαλειοθήκη για την υλοποίηση σύνθετων εικονικών κόσμων με υψηλό βαθμό ρεαλισμού και αλληλεπίδρασης.

Μεταξύ των διαθέσιμων game engines, δύο πλατφόρμες έχουν κυριαρχήσει τα τελευταία χρόνια: η Unreal Engine, που αναπτύχθηκε από την Epic Games, και η Unity, που αποτελεί το κύριο προϊόν της Unity Technologies. Και οι δύο μηχανές προσφέρουν εξαιρετικά προηγμένες δυνατότητες και υποστηρίζονται από πολυπληθείς κοινότητες χρηστών και προγραμματιστών, καθιστώντας τες εργαλεία πρώτης επιλογής για επαγγελματίες της βιομηχανίας αλλά και για νέους δημιουργούς. Η επιλογή μεταξύ Unreal Engine και Unity δεν είναι πάντοτε εύκολη, καθώς η καθεμία διαθέτει πλεονεκτήματα και ιδιαιτερότητες που ανταποκρίνονται σε διαφορετικές ανάγκες και προτεραιότητες ανάπτυξης.

Η παρούσα πτυχιακή εργασία έχει ως κεντρικό σκοπό τη συγκριτική μελέτη των Unreal Engine και Unity, με έμφαση τόσο στις τεχνικές τους δυνατότητες όσο και στις πρακτικές εφαρμογές τους. Ειδικότερα, η εργασία στοχεύει στην αναλυτική παρουσίαση των δύο μηχανών, στη συστηματική διερεύνηση των χαρακτηριστικών τους, στη μελέτη των μεθόδων εγκατάστασης και ρύθμισης του περιβάλλοντός τους, καθώς και στην πρακτική υλοποίηση ενός έργου σε καθεμία από αυτές. Μέσα από αυτή τη διαδικασία επιχειρείται η εξαγωγή ασφαλών συμπερασμάτων αναφορικά με τα πλεονεκτήματα και τα μειονεκτήματα κάθε μηχανής, ενώ παράλληλα διατυπώνονται κατευθύνσεις και προτάσεις που μπορούν να φανούν χρήσιμες σε νέους developers οι οποίοι κάνουν τα πρώτα τους βήματα στον χώρο της ανάπτυξης παιχνιδιών.

Τα παραδοτέα της εργασίας συνίστανται σε τρία διακριτά επίπεδα. Πρώτον, περιλαμβάνεται μία θεωρητική ανάλυση των δυνατοτήτων, των τεχνολογιών και των επεκτάσεων που προσφέρει η κάθε μηχανή. Δεύτερον, παρουσιάζεται η πρακτική εφαρμογή μέσω της δημιουργίας δύο έργων με αντίστοιχα χαρακτηριστικά, ένα στην Unreal Engine και ένα στην Unity, τα οποία υλοποιήθηκαν με στόχο την ανάδειξη των διαφορών στην ανάπτυξη. Τρίτον, επιχειρείται η τελική συγκριτική αξιολόγηση των αποτελεσμάτων, ώστε να εξαχθούν χρήσιμα και τεκμηριωμένα συμπεράσματα που μπορούν να αποτελέσουν οδηγό για τη μελλοντική ενασχόληση νέων δημιουργών με τις συγκεκριμένες πλατφόρμες.

Η δομή της εργασίας είναι οργανωμένη με τρόπο που να εξυπηρετεί τη σταδιακή παρουσίαση της θεωρίας και την πρακτική της εφαρμογή. Στο δεύτερο κεφάλαιο παρουσιάζονται συνοπτικά οι μηχανές ανάπτυξης και η σημασία τους στη σύγχρονη βιομηχανία ψηφιακών παιχνιδιών. Στο τρίτο κεφάλαιο αναλύεται εκτενώς η Unreal Engine, με αναφορά στην ιστορική της πορεία, τις τεχνικές προδιαγραφές, το περιβάλλον χρήσης και τις βασικές τεχνολογίες που τη χαρακτηρίζουν, καθώς και στα πλεονεκτήματα και μειονεκτήματά της. Αντίστοιχα, το τέταρτο κεφάλαιο είναι αφιερωμένο στην Unity, η οποία εξετάζεται με βάση την ιστορική της εξέλιξη, τα τεχνικά της χαρακτηριστικά, το περιβάλλον ανάπτυξης και τις επιμέρους τεχνολογίες που την καθιστούν ιδιαίτερα δημοφιλή.

Στο πέμπτο κεφάλαιο παρουσιάζεται αναλυτικά η διαδικασία υλοποίησης ενός έργου στην Unreal Engine, ξεκινώντας από την εισαγωγή των assets και την οργάνωση του project και καταλήγοντας στη δημιουργία του εικονικού κόσμου, στη ρύθμιση της πλοήγησης, στην ανάπτυξη της διεπαφής χρήστη και στην εκτέλεση του τελικού build. Στο έκτο κεφάλαιο ακολουθεί η αντίστοιχη υλοποίηση έργου στην Unity, ώστε να καταστεί εφικτή η συγκριτική μελέτη υπό παρόμοιες συνθήκες. Το έβδομο και τελευταίο κεφάλαιο συγκεντρώνει τα συμπεράσματα της έρευνας και επιχειρεί τη συγκριτική αποτίμηση των δύο μηχανών, αξιολογώντας την αποδοτικότητα, τη λειτουργικότητα, την ευκολία χρήσης και την προοπτική τους για μελλοντική ανάπτυξη.

Εν κατακλείδι, η εργασία αυτή φιλοδοξεί να προσφέρει μία ολοκληρωμένη εικόνα των δύο κυριότερων game engines που κυριαρχούν στην παγκόσμια βιομηχανία ανάπτυξης παιχνιδιών. Μέσα από τη θεωρητική και πρακτική προσέγγιση, επιδιώκει όχι μόνο να συγκρίνει αντικειμενικά τις δυνατότητες και τις αδυναμίες των Unreal Engine και Unity, αλλά και να αποτελέσει έναν χρήσιμο οδηγό για νέους developers, βοηθώντας τους να επιλέξουν το κατάλληλο εργαλείο για την ανάπτυξη των μελλοντικών τους έργων.

Λέξεις-κλειδιά: Unity, Unreal Engine, Προσομοιωτής Οδήγησης, Ανάπτυξη Παιχνιδιών, Σύγκριση Μηχανών Παιχνιδιών.

Κεφάλαιο 2ο Διαδικασία Ανάπτυξης Παιχνιδιών και Ρόλος των Game Engines

Όλοι μας έχουμε παίξει κάποια στιγμή στην ζωή μας κάποιο παιχνίδι, είτε αυτό ήταν σε υπολογιστή, σε κονσόλα ή στο κινητό. Όμως πίσω από αυτό το πολύ όμορφο και διασκεδαστικό παιχνίδι, βρίσκεται ένας πολύ πολύπλοκος κώδικας, που συνδέει την τέχνη όπως μουσική και γραφικά, με την λογική και την πληροφορική. Με το πέρασμα του χρόνου, έχουμε καταφέρει να φτάσουμε μερικές πτυχές της ανάπτυξης παιχνιδιών σε αυτοματοποιημένα επίπεδα ή σε σημείο που η ίδια η μηχανή ανάπτυξης που χρησιμοποιείται να σου δίνει έτοιμες λύσεις, μειώνοντας τον χρόνο για την ολοκλήρωση ενός παιχνιδιού.

Η ανάπτυξη (Production) όμως έρχεται αργότερα και δεν είναι το πρώτο βήμα που ακολουθείται. Πριν από αυτή υπάρχει η σύλληψη της ιδέας (Concept) και η προεργασία (Pre-Production), ενώ υπάρχει και το «μετά» (Post-Production).

2.1 Concept

Η σύλληψη της ιδέας είναι το πρώτο βήμα δημιουργίας παιχνιδιών. Είναι η απόφαση του είδους παιχνιδιού (όπως Racing, First-Person Shooter), αν θα είναι για κονσόλα, για υπολογιστή, για κινητό ή για όλα, η μηχανή που θα χρησιμοποιηθεί, καθώς επίσης, σε αυτό το στάδιο θα αποφασιστούν και πράγματα όπως το αν θα είναι βασισμένα σε αληθινά γεγονότα ή σε κάποιο βιβλίο ή ταινία, αν θα συνεχίζει κάποιον ήδη υπάρχων τίτλο ή αν θα προσομοιώνει με κάποιον τρόπο την πραγματικότητα. [1]

2.2 Pre-Production

Το επόμενο βήμα είναι και το πιο σημαντικό. Σε αυτό, θα συμμετάσχουν όλοι όσοι θα πάρουν μέρος στην δημιουργία του παιχνιδιού, μεταξύ άλλων προγραμματιστές (Programmers), σεναριογράφοι, καλλιτέχνες (Artists) και παραγωγοί. Όλοι μαζί θα κατασκευάσουν στο χαρτί το παιχνίδι. Θα γραφτεί η ιστορία (αν έχει μια), τα βασικά σημεία αυτής, θα διαμορφωθούν τα κουμπιά (inputs) από τους προγραμματιστές και θα δημιουργηθεί και το εικαστικό προσχέδιο του παιχνιδιού (Concept Art), πάντα με γνώμονα το concept που έχει αποφασιστεί στο προηγούμενο βήμα. [1]

2.3 Production

Το τρίτο και προτελευταίο στάδιο είναι η ανάπτυξη. Σε αυτό το στάδιο, αναφερόμαστε πλέον στη δημιουργία του ίδιου του παιχνιδιού καθαυτού. Οι προγραμματιστές χτίζουν τα βασικά στοιχεία του παιχνιδιού, όπως την βιβλιοθήκη και την τεχνητή νοημοσύνη (AI), ενώ άλλοι χτίζουν ότι χρειάζεται για τα γραφικά, ειδικά εφέ, φωτισμό. Οι artists ασχολούνται με τα καλλιτεχνικά κομμάτια, όπως η μουσική, η εικονογράφηση και την κίνηση (animation).

Αφού ολοκληρωθούν και ενοποιηθούν όλα μαζί θα γίνει ένας ποιοτικός έλεγχος του τελικού προϊόντος, για τυχόν αλλαγές που πρέπει να γίνουν. Για παράδειγμα, μπορεί ο στόχος να είναι τα 60 καρέ ανά δευτερόλεπτο (fps), αλλά να υπάρχουν στιγμές που το παιχνίδι να πέφτει κάτω από αυτά, χαλώντας την

εμπειρία. Οπότε θα πρέπει σε αυτό το στάδιο να αποφασιστούν τυχών αλλαγές για την βελτιστοποίηση (optimization) του παιχνιδιού προς διανομή. [1]

2.4 Post-Production

Έπειτα του optimization σε ένα σημείο που να είναι ικανοποιημένοι οι δημιουργοί, έρχεται το 4ο και τελευταίο στάδιο. Ξεκινά με την διανομή του παιχνιδιού σε πρώιμη εκδοχή (Alpha Version) σε περιορισμένο κοινό, με σκοπό να βρεθούν και να διορθωθούν λάθη (errors) και σφάλματα (bugs), πριν την επίσημη κυκλοφορία.

Με την ολοκλήρωση και του 4ου σταδίου, το παιχνίδι κυκλοφορεί επίσημα. Στην περίπτωση των κονσόλων, το παιχνίδι θα σταλεί και στον κατασκευαστή της εκάστοτέ κονσόλας για την ποιοτικό έλεγχο, πριν την κυκλοφορία. [1]

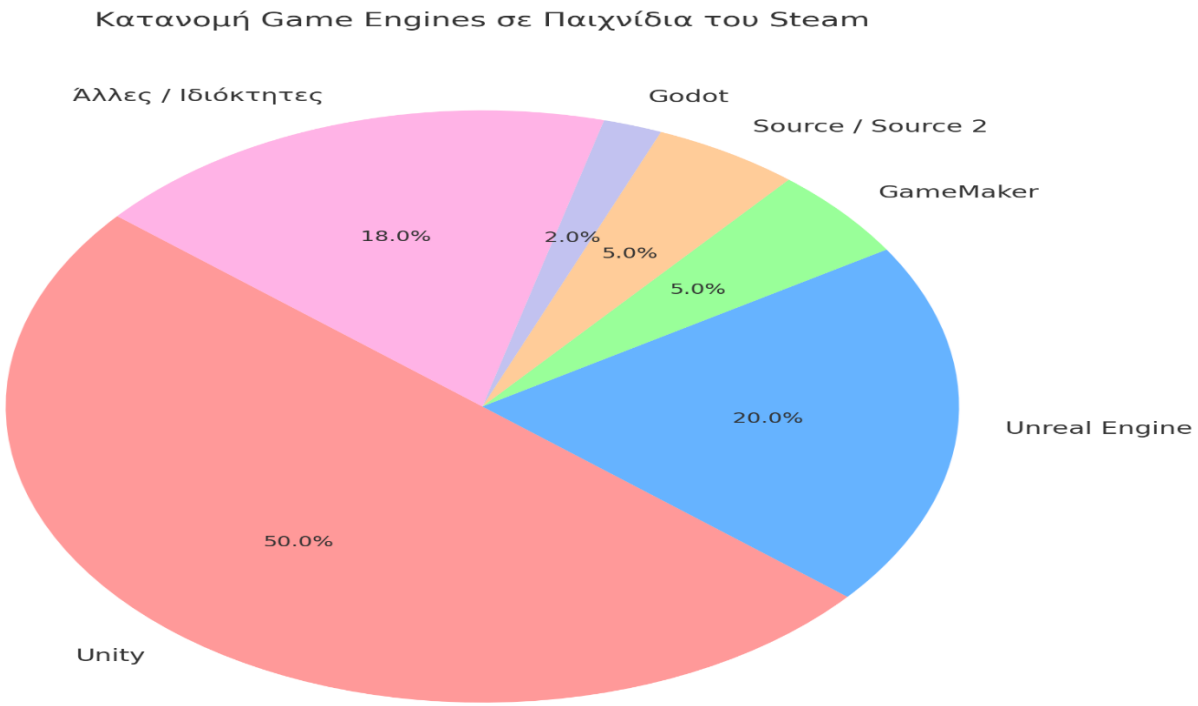
Στο πρώτο στάδιο, η επιλογή της μηχανής ανάπτυξης είναι απο τις πιο σημαντικές. Με τον όρο **μηχανή ανάπτυξης παιχνιδιών (game engines)** εννοούμε ένα προγραμματιστικό περιβάλλον, γνωστό και ως **«αρχιτεκτονική παιχνιδιού»**, με το οποίο κάποιος μπορεί να δημιουργήσει δισδιάστατα (2D) ή τρισδιάστατα (3D) περιβάλλοντα, βασιζόμενος σε ρυθμίσεις και τεχνικές που απλουστεύουν τη διαδικασία, χρησιμοποιώντας διάφορες γλώσσες προγραμματισμού. [2] Πρόκειται ουσιαστικά για ένα σύνολο από εργαλεία και υποσυστήματα όπως η απόδοση γραφικών (rendering), η φυσική προσομοίωση (physics), η διαχείριση ήχου και εφέ, η τεχνητή νοημοσύνη, το animation και η υποστήριξη πολλαπλών πλατφορμών, που συγκεντρώνονται σε μία ενιαία και οργανωμένη μορφή.

Οι μηχανές αυτές επιτρέπουν στους προγραμματιστές, σχεδιαστές και καλλιτέχνες να συνεργάζονται αποτελεσματικά σε ένα κοινό ψηφιακό περιβάλλον, καθιστώντας τη διαδικασία παραγωγής ενός παιχνιδιού πιο αποδοτική, προβλέψιμη και συστηματική. Οι περισσότερες σύγχρονες μηχανές ανάπτυξης προσφέρουν γραφικό περιβάλλον εργασίας, οπτικό προγραμματισμό, επεκτάσεις τρίτων, καθώς και υποστήριξη για πλήθος πλατφορμών όπως PC, κονσόλες, κινητές συσκευές και VR/AR τεχνολογίες.

Ερχόμενοι στο 2ο στάδιο, πρέπει να αποφασιστεί ποια μηχανή θα χρησιμοποιηθεί για την παραγωγή του παιχνιδιού. Μια πολύ κρίσιμη απόφαση, καθώς θα πρέπει να αξιολογηθούν όλες οι πιθανές για να επιλεχθεί η καλύτερη με βάση διαφόρων παραγόντων, μεταξύ άλλων τις τεχνικές γνώσεις των developer, των τεχνικών προδιαγραφών που θέλουμε να έχει το τελικό προϊόν. Δεν μπορεί να γίνει όμως μια σωστή και αντικειμενική σύγκριση αν δεν έχουμε την κατάλληλη προσωπική εμπειρία και γνώση από τη χρήση των μηχανών που έχουμε διαθέσιμες σαν επιλογές.

Όσον αφορά τις υπάρχουσες μηχανές, αυτές είναι πολλές. Είτε σαν υπηρεσίες τρίτων (Unity, Unreal), είτε ιδιόκτητες μηχανές (Frostbite της EA, Rage της Rockstar). Με τη 2η κατηγορία να είναι κλειστού κώδικα, συνεπώς να χρειάζονται άδεια χρήσης ή και σε κάποιες περιπτώσεις να είναι και μόνο για προσωπική χρήση της ίδιας της εταιρείας που την κατασκεύασε, οι περισσότερες μικρές εταιρείες και ανεξάρτητοι developers στρέφουν το βλέμμα τους στις δωρεάν μηχανές τρίτων.

Σε καμία περίπτωση όμως, όντας δωρεάν δεν τις καθιστά και κατώτερες. Μάλιστα, οι πιο διαδεδομένες είναι η Unreal και η Unity. Κυριαρχούν στον χώρο της ανάπτυξης παιχνιδιών αλλά και όχι μόνο, καθώς προσφέρουν εξαιρετικά εργαλεία και δυνατότητες. Όπως είναι γνωστό, και οι 2 μηχανές έχουν την ικανότητα να δημιουργήσουν μια υψηλής ποιότητας εφαρμογή, προσφέροντας διαφορετική εμπειρία στον χρήστη η κάθε μια.



2.1) Κατανομή Game Engines σε παιχνίδια του Steam μέχρι και τον Ιανουάριο 2025 [5]

Η Unity παρέχοντας στον χρήστη ένα ευρύ φάσμα έτοιμων βιβλιοθηκών και assets, μια φιλική προς τον χρήστη διεπαφή, πολλά δωρεάν online μαθήματα και μια μεγάλη κοινότητα, θεωρείται από πολλούς στον χώρο ως την ιδανική μηχανή για αρχάριους προγραμματιστές ή και μικρότερες ομάδες ανάπτυξης. [3] Υποστηρίζει γλώσσες όπως C#, επιτρέπει την ανάπτυξη σε πλήθος πλατφορμών και δίνει μεγάλη ευελιξία στον σχεδιασμό 2D ή 3D παιχνιδιών. Είναι επίσης ιδιαίτερα χρήσιμη σε εφαρμογές εκπαίδευσης, προσομοίωσης και αρχιτεκτονικής απεικόνισης.

Η Unreal, σε αντίθεση, θεωρείται δυσκολότερη στην εκμάθησή της, αλλά προσφέρει μεγαλύτερες δυνατότητες. Παρέχει στον χρήστη μια βιβλιοθήκη με έτοιμα εργαλεία, assets υψηλής ευκρίνειας και πολλές τεχνολογίες για τη διευκόλυνση δημιουργίας παιχνιδιών. Ενσωματώνει το σύστημα Blueprint, ένα οπτικό σύστημα scripting που επιτρέπει τον προγραμματισμό χωρίς γραφή κώδικα, και υποστηρίζει προηγμένες τεχνολογίες όπως Lumen (δυναμικός φωτισμός), Nanite (εικονική γεωμετρία υψηλής πιστότητας) και real-time rendering, καθιστώντας την ιδανική για AAA παραγωγές ή έργα που απαιτούν υψηλό επίπεδο ρεαλισμού.

Αξίζει να σημειωθεί ότι τόσο η Unity όσο και η Unreal χρησιμοποιούνται σήμερα πέρα από τη βιομηχανία του gaming – σε τομείς όπως η **τηλεόραση και ο κινηματογράφος**, η **αρχιτεκτονική**, η **αυτοκινητοβιομηχανία** και η **εκπαίδευση**, αποδεικνύοντας την ευελιξία και τη δύναμή τους ως εργαλεία ψηφιακής δημιουργίας.

Παρακάτω θα αναλυθούν όλες οι τεχνολογίες που προσφέρει η κάθε μηχανή, καθώς και ένας μικρός οδηγός για την κάθε μια, εξηγώντας πώς να χρησιμοποιηθούν σε βασικό επίπεδο με γνώμονα τη χρήση τους στις εφαρμογές που συνοδεύουν το κείμενο.

Κεφάλαιο 3ο Unreal Engine

Η **Unreal Engine**, μία από τις πιο ισχυρές και διαδεδομένες μηχανές γραφικών παγκοσμίως, δημιουργήθηκε από την **Epic Games** το 1995 για να καλύψει την ανάγκη των δημιουργών παιχνιδιών για ρεαλιστική απεικόνιση και πλούσιο περιεχόμενο. Ξεκινώντας ως εργαλείο για τη δημιουργία τρισδιάστατων βιντεοπαιχνιδιών, εξελίχθηκε με τα χρόνια σε μια πολυμορφική πλατφόρμα που υποστηρίζει την ανάπτυξη **διαδραστικών εφαρμογών**, **ταινιών**, **ψηφιακών τρέιλερ**, **VR/AR εμπειριών**, ακόμα και **προσομοιώσεων για αρχιτεκτονική και εκπαίδευση**. Σήμερα, τίτλοι όπως το *Fortnite* και το *Rocket League*, αλλά και δημοφιλείς τηλεοπτικές παραγωγές όπως *The Mandalorian*, αξιοποιούν τις δυνατότητές της για τη δημιουργία εντυπωσιακών γραφικών και περιβαλλόντων υψηλής πιστότητας [4] [5].

Η Unreal Engine διατίθεται δωρεάν για χρήση, με **ευέλικτους όρους άδειας** ανάλογα με το είδος του έργου και το εμπορικό του μέγεθος. Παρόλο που ο **πηγαίος κώδικάς της είναι προσβάσιμος** μέσω του GitHub για σκοπούς κατανόησης και απασφαλμάτωσης, η άδεια χρήσης της παραμένει **κλειστού τύπου** [30]. Υποστηρίζει **λειτουργικά συστήματα Windows, Mac και Linux**, ενώ συνεργάζεται άριστα με εργαλεία ανάπτυξης όπως το **Visual Studio** (2015 και μεταγενέστερα), με επιπλέον υποστήριξη μέσω των **Unreal Engine Visual Studio Tools**, που επιτρέπουν την καλύτερη διαχείριση του κώδικα και την αποδοτική μεταγλώττιση [5] [6].

Η Unreal βασίζεται κατά κύριο λόγο στη **γλώσσα C++**, αλλά υποστηρίζει και **οπτικό προγραμματισμό** μέσω του συστήματος **Blueprints** – ένα εργαλείο που επιτρέπει την ανάπτυξη λογικής χωρίς γραμμή κώδικα. Αυτό το σύστημα χρησιμοποιείται ευρέως για τη δημιουργία πρωτοτύπων, αλλά ο συνδυασμός **Blueprints και C++** είναι η προτεινόμενη προσέγγιση για ολοκληρωμένα έργα, καθώς εκμεταλλεύεται τα πλεονεκτήματα και των δύο τεχνολογιών [7].

Η **Epic Games παρέχει εκτενή τεκμηρίωση**, οδηγούς, βιντεομαθήματα (δωρεάν και επί πληρωμή), και βασικά assets (μοντέλα, υλικά, εφέ), καθιστώντας εφικτή την άμεση έναρξη δημιουργίας. Επιπλέον, με την πλατφόρμα **Unreal Marketplace**, οι χρήστες μπορούν να αποκτήσουν έτοιμο περιεχόμενο (ήχους, scripts, animations, τοπία, ακόμα και ολοκληρωμένα έργα), το οποίο μπορούν να ενσωματώσουν στις δικές τους δημιουργίες, μειώνοντας το χρόνο ανάπτυξης [8].

3.1 Ιστορία

Το 1995 ο Tim Sweeney, ιδρυτής της Epic Games, δημιούργησε την μηχανή για δική του χρήση, ενώ λίγα χρόνια αργότερα (1998) την διέθεσε και σε άλλα στούντιο που αναγνώρισαν τις δυνατότητες της. Εκμεταλλευόταν μόνο τον επεξεργαστή καθώς χρησιμοποιούσε αποκλειστικά απόδοση μέσω λογισμικού (*Software Rendering*). Έφερε στην επιφάνεια τεχνολογίες όπως το *Σύστημα Ανίχνευσης Συγκρούσεων (Collision Detection)*. [9]

Με τις συνεχές αναβαθμίσεις που δέχτηκε τα επόμενα χρόνια, η 2^{η} έκδοση της μηχανής έκανε την εμφάνισή της (2002). Μαζί με την μηχανή έκανε την εμφάνιση του και το παιχνίδι «America's Army», ένα παιχνίδι σε συνεργασία με τον αμερικανικό στρατό για την πληροφόρηση, εκπαίδευση και την στρατολόγηση πιθανών στρατιωτών. Η μηχανή πλέον δεν χρησιμοποιούσε Software Rendering, αλλά πλέον χρησιμοποιούσε και τις δυνατότητες της κάρτας γραφικών, αυξάνοντας την ποιότητα των γραφικών

κατακόρυφα. Πρόσθεσε λειτουργίες όπως το *Σύστημα Απεικόνισης/Προσομοίωσης Σωματιδίων (Particle System)*. [9]

Το 2006 κυκλοφόρησε η 3η έκδοση της μηχανής (UE3). Λειτουργίες όπως το *Σύστημα Προσομοίωσης Φυσικής (Physics System)* και το *Σύστημα Ήχου (Sound System)* έκανα την εμφάνιση τους. Ίσως η μεγαλύτερη αλλαγή όμως που έφερε, ήταν ότι για τους υπολογισμούς πλέον δεν χρησιμοποιούνταν κορυφές (Vertexes) αλλά εικονοστοιχεία (Pixels), αναβαθμίζοντας ακόμα περισσότερο τα γραφικά. Το 2010, η μηχανή κάλυψε τις ανάγκες των developer στον τομέα των κινητών, δίνοντας τους την δυνατότητα για ανάπτυξη παιχνιδιών iOS και Android. Την UE3 υιοθέτησαν στα παιχνίδια τους μεγάλες εταιρείες, όπως η Disney, η Atari και η Ubisoft. [9]

Για 8 χρόνια μέχρι το 2014, ήταν κυρίαρχη στον χώρο, έως ότου την θέση της πήρε η καινούργια έκδοση (UE4). Ενώ η μηχανή έφερε πολλές νέες καινοτομίες, άλλαξε και ριζικά τον τρόπο με τον οποίο οι χρήστες της μπορούσαν να δημιουργήσουν τα παιχνίδια τους. Η μηχανή πέρασε από απλό προγραμματισμό C++ σε *Οπτικό Προγραμματισμό (Blueprint Visual Scripting)*. Φυσικά, ένας developer μπορούσε ακόμα να γράψει την λογική που ήθελε με την παλιά μέθοδο, με το visual scripting όμως απλουστευόταν κατά πολύ η διαδικασία δημιουργίας βασικών λειτουργιών ενός παιχνιδιού. Σε αυτή την έκδοση, πρωτοεμφανίστηκε και το *Σύστημα Παγκόσμιου Φωτισμού (Global Illumination System)*. [9]

Το 2020 ανακοινώθηκε και κυκλοφόρησε η UE5. Αυτή η έκδοση αναβάθμισε παλιές τεχνολογίες που υπήρχαν στην μηχανή (το Global Illumination System, πλέον Lumen) και έφερε καινούργιες, μερικές από αυτές να είναι το *Σύστημα Υψηλής Πιστότητας Μικρογεωμετρίας (Nanites)* και το *Σύστημα Δυναμικής Κατανομής και Φόρτωσης του Παγκόσμιου Χάρτη (World Partition)*. [10]

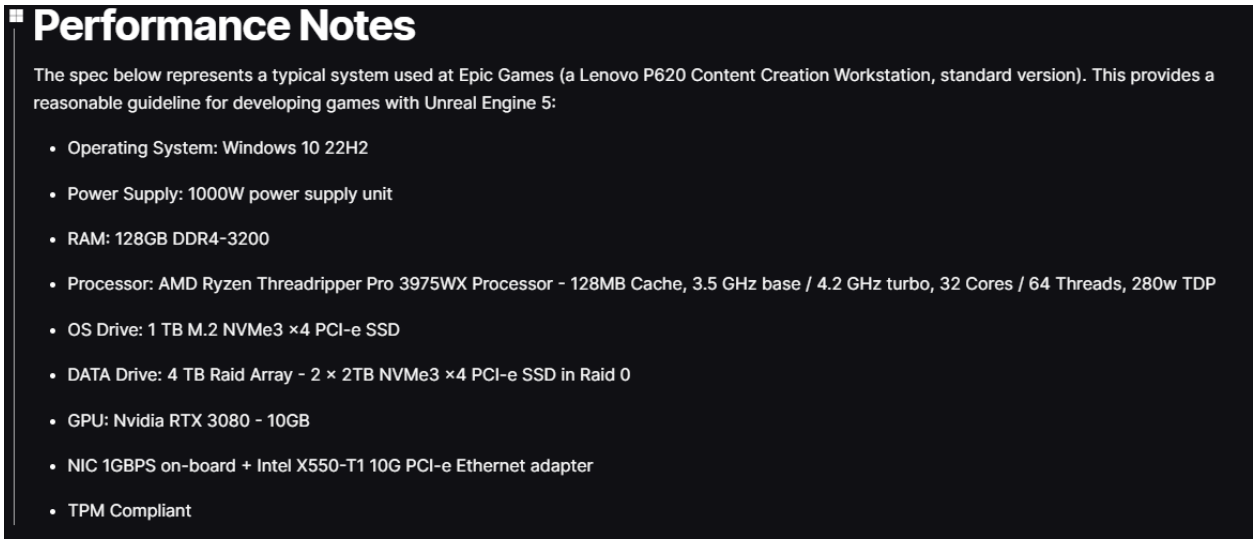
3.2 Τεχνικές Προδιαγραφές

Η Unreal Engine 5.6 υποστηρίζεται από λειτουργικά συστήματα Windows, ξεκινώντας από την έκδοση 10 και μεταγενέστερα, με προτεινόμενη τη χρήση των Windows 11 για βέλτιστη απόδοση. Όσον αφορά τον επεξεργαστή, προτείνεται η χρήση ενός εξαπύρηνου (ή καλύτερου) μοντέλου της Intel ή της AMD, με ελάχιστη συχνότητα λειτουργίας τα 3.4 GHz, προκειμένου να αξιοποιηθούν πλήρως οι δυνατότητες της μηχανής και των πολυνηματικών λειτουργιών της.

Για τα γραφικά, απαιτείται κάρτα γραφικών συμβατή με DirectX 12, με ελάχιστη VRAM τα 8 GB και υποστήριξη για Hardware Ray Tracing, ώστε να είναι δυνατή η χρήση του Lumen και του Nanite σε πραγματικό χρόνο. Ενδεικτικά, κάρτες όπως η NVIDIA RTX 2080 SUPER ή νεότερες αποτελούν τη συνιστώμενη επιλογή για σταθερή απόδοση σε απαιτητικά έργα.

Σε ό,τι αφορά τον αποθηκευτικό χώρο, η Unreal Engine 5.6 απαιτεί σημαντική χωρητικότητα τόσο για την αρχική εγκατάσταση όσο και για την αποθήκευση πρόσθετου περιεχομένου. Μια βασική εγκατάσταση καταλαμβάνει περίπου 40 GB, ενώ έργα που περιλαμβάνουν μεγάλο όγκο τρισδιάστατου περιεχομένου — όπως αρχιτεκτονικές μακέτες, μοντέλα χαρακτήρων ή ηχητικά/φωτιστικά εφέ — μπορεί να αυξήσουν τις ανάγκες αποθήκευσης σε 100 GB ή και περισσότερο. Ειδικά στην ανάπτυξη με χρήση C++, τα σύμβολα απασφαλμάτωσης, τα οποία είναι απαραίτητα για την ορθή παρακολούθηση και διάγνωση του κώδικα, μπορεί να καταλαμβάνουν επιπλέον 25–30 GB.

Σύμφωνα με δοκιμές που πραγματοποιήθηκαν στο πλαίσιο της παρούσας εργασίας, αλλά και βάσει των επίσημων συστάσεων της Epic Games, η χρήση ενός γρήγορου SSD δίσκου τύπου NVMe είναι καθοριστικής σημασίας. Αυτός εξασφαλίζει γρήγορη φόρτωση σκηνών, αποτελεσματικό streaming δεδομένων και ομαλή λειτουργία του περιβάλλοντος ανάπτυξης. [11]

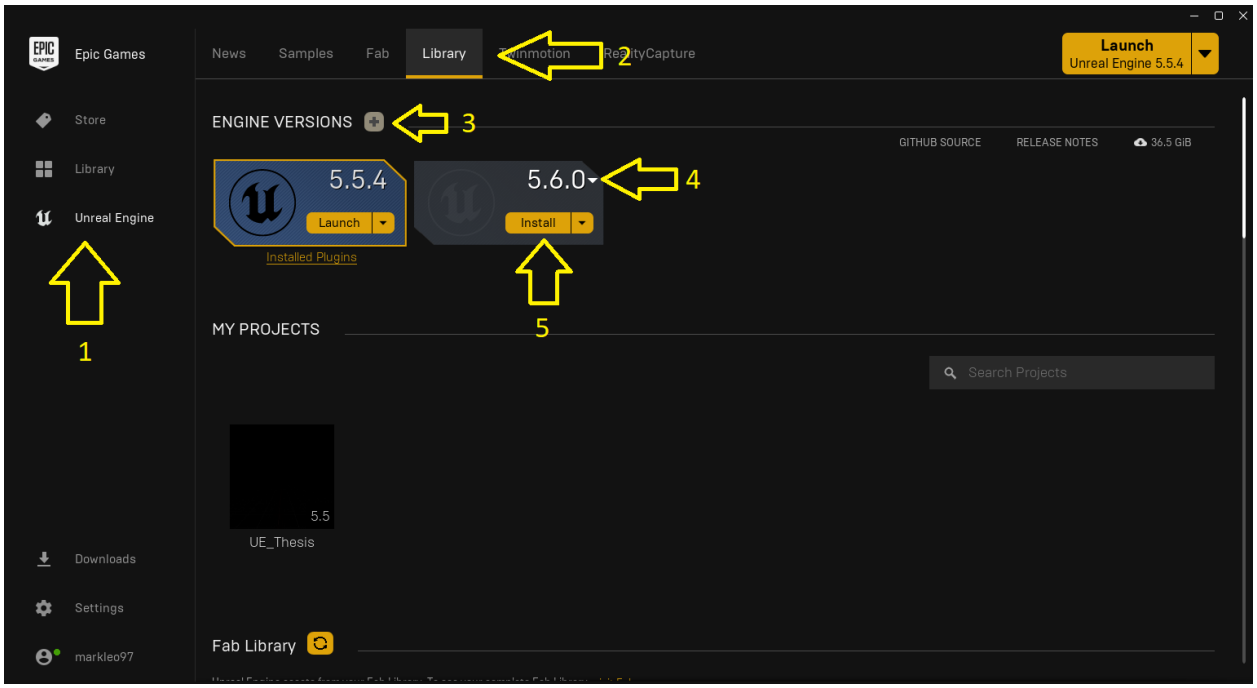


3.1) Specs Υπολογιστή της Epic

3.3 Εγκατάσταση και Περιβάλλον

3.3.1 Εγκατάστασή

Για την εγκατάσταση της Unreal, χρειάζεται να κατεβάσουμε το Epic Games Launcher, το οποίο μπορούμε να βρούμε στο διαδίκτυο, στην επίσημη σελίδα της Epic. Αφού κανείς κατεβάσει το Epic Games Launcher και κάνει σύνδεση στον λογαριασμό του, στα αριστερά θα βρει ένα tab με τίτλο “Unreal Engine”. Πατώντας το, και πηγαίνοντας στο tab “Βιβλιοθήκη (Library)” στο επάνω μέρος του launcher, θα του εμφανιστεί μια περιοχή με όνομα “Engine Versions”. Εκεί κάνοντας κλικ στο “+”, θα του εμφανιστεί ένα γκριζαρισμένο κουτάκι με την έκδοση της μηχανής και ένα κουμπί “Εγκατάσταση”. Πατώντας πάνω στον αριθμό της έκδοσης, ένα dropdown με όλες τις εκδόσεις που είναι διαθέσιμες θα εμφανιστεί, δίνοντας στον χρήστη την επιλογή να επιλέξει. Μετά την εγκατάσταση της μηχανής, το κουτάκι θα γίνει μπλε και το κουμπί πλέον θα γράφει “Εκκίνηση (Launch)”.



3.2)Βήματα Εγκατάστασης Μηχανής

3.3.2 Ρυθμίσεις Μηχανής

Αν το παιχνίδι προς δημιουργία θα στοχεύει Android, iOS, Linux ή λογισμικό τηλεόρασης, θα χρειαστεί μέσα από της ρυθμίσεις της εγκατεστημένης μηχανής, να εγκαταστήσουμε το αντίστοιχο πακέτο. Για να καταφέρουμε να εμπλουτίσουμε την μηχανή με τα αναγκαία αυτά πακέτα, πατάμε το βελάκι δίπλα από το κουμπί Εκκίνηση, και πλοηγούμαστε στην επιλογή Ρυθμίσεις (Options). Εκεί θα βρούμε όλες τις επιλογές που ενδεχομένως να χρειαζόμαστε. Θα παρατηρήσουμε ότι θα έχει προεπιλεγμένες κάποιες επιλογές. Το "Starter Content" και το Templates and Feature Pack" είναι τα δυο πακέτα που κάνουν την εκκίνηση δημιουργίας πολύ εύκολη, προσθέτοντας έτοιμο προς χρήση περιεχόμενο στο έργο. Το "Engine Source" δείχνει την προέλευση της μηχανής (Epic Games Launcher ή πηγαίο κώδικα) και καθορίζει τις αλλαγές που μπορούν να γίνουν στον πυρήνα της μηχανής. Επίσης, υπάρχει και η επιλογή "Editor Symbols for Debugging", η οποία δεν είναι προεπιλεγμένη, και η χρησιμότητά της είναι για καταστήσει πιο εύκολη την απασφαλμάτωση του κώδικα C++. Σε περίπτωση που το έργο είναι εξολοκλήρου φτιαγμένο με Blueprints, η τελευταία επιλογή μπορεί να μην ενεργοποιηθεί.

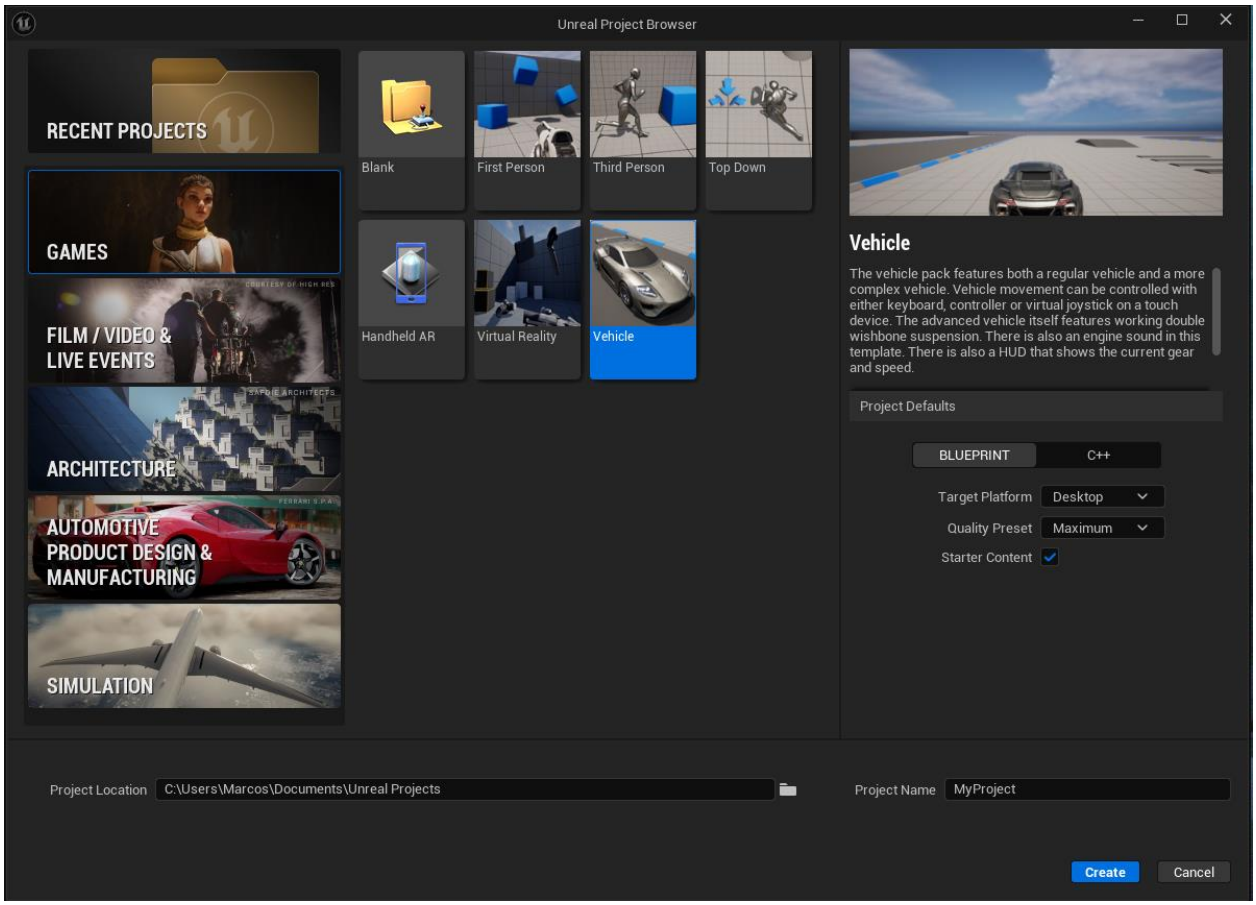
3.3.3 Δημιουργία Έργου

Με την δημιουργία ενός έργου και την επιλογή του κατάλληλου template, σύμφωνα με τις ανάγκες μας, η μηχανή θα ξεκινήσει να μεταγλωττίζει τα αρχεία κώδικα που θα προϋπάρχουν στο έργο μας. Ο χρόνος μεταγλώττισης είναι ανάλογος της ισχύος του επεξεργαστή και της ταχύτητας αναγνώρισης του σκληρού δίσκου. Ενδείκνυται ο δίσκος να είναι τουλάχιστον τύπου Solid State Drive (SSD), ενώ ένας δίσκος τύπου SSD M.2 NVMe θα μείωνε ακόμα περισσότερο τον χρόνο μεταγλώττισης.

3.3)Παράθυρο Μεταγλώττισης

Αφού ολοκληρωθεί η μεταγλώττιση, η Unreal θα προτρέψει τον χρήστη να επιλέξει και να χρησιμοποιήσει την χρήση κάποιου *Πρότυπου Έργου (Template)*. Το κάθε template είναι εμπλουτισμένο, με βασικά πράγματα που θα χρειαστούν στον developer ενώ επίσης με το άνοιγμα του κατάλληλου, μπορεί να έχει και έτοιμα components για την χρήση τους, έτσι ώστε να μην χρειάζεται να δημιουργηθούν από την αρχή.

Με την εκκίνηση της εγκατεστημένης έκδοσης της Unreal, εμφανίζεται ένα παράθυρο για την επιλογή του project που θέλουμε να χρησιμοποιήσουμε.



3.4)Παράθυρο Δημιουργίας Project Unreal

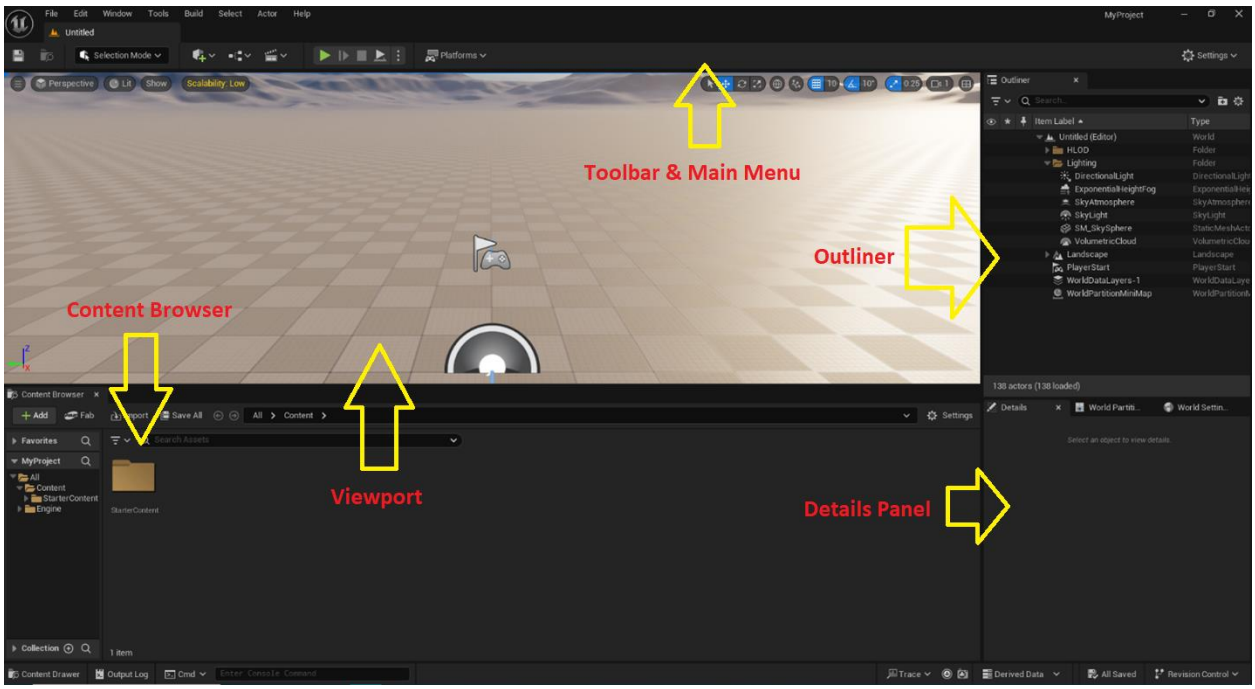
Στα αριστερά υπάρχουν οι κατηγορίες των έργων, για τις οποίες χρησιμοποιείται η Unreal. Κάθε κατηγορία, έχει τα δικά της templates, όπως φαίνεται στο κέντρο του παραθύρου, όπως για παράδειγμα η κατηγορία Games, έχει templates για την δημιουργία παιχνιδιών *Πρώτου/Τρίτου Προσώπου (First/Third Person), Εικονικής Πραγματικής (Virtual Reality), Αμαξιού (Vehicle)* και άλλα. Στα δεξιά, υπάρχει μια περίληψη για το τι προσφέρει το template, ώστε να καταλάβει ο developer πότε να το χρησιμοποιήσει και αν τον εξυπηρετεί για το τελικό προϊόν που θέλει να δημιουργήσει. Κάτω από την περιγραφή, υπάρχουν μερικές ρυθμίσεις για το project, όπως το αν θες να χρησιμοποιεί Blueprints ή C++ (η επιλογή κάποιου, δεν σημαίνει ότι δεν μπορεί να χρησιμοποιηθεί το άλλο) και αν στοχεύει specs κινητού ή υπολογιστή. Τέλος στο κάτω μέρος του παραθύρου, μπορείς να διαλέξεις την τοποθεσία αποθήκευσης του project καθώς και το όνομα του.

Για το δικό μου έργο, επιλέχθηκε το Vehicle Template.

Μετά την επιλογή του κατάλληλου template, θα γίνει ακόμα μια μεταγλώττιση από την μηχανή. Ο χρόνος της εξαρτάται πάλι από τις ταχύτητες επεξεργαστή και δίσκου, καθώς και από τα αρχεία που πρέπει να μεταγλωττιστούν. Για παράδειγμα, ένα άδειο (blank) έργο, θα χρειαστεί λιγότερο χρόνο από το First Person template.

3.3.4 Περιβάλλον – Editor

Με το τέλος της μεταγλώττισης, ο χρήστης θα είναι σε θέση να ξεκινήσει να δίνει ζωή στο παιχνίδι του. Αυτό θα μπορέσει να το κάνει μέσω του καταλλήλως διαμορφωμένου περιβάλλοντος (Editor).



3.5) Δομή Editor Unreal

Στο κάτω μέρος του παραθύρου βλέπουμε το παράθυρο διαχειριστή αρχείων (Content Browser), στο οποίο μπορούμε να δούμε όλα τα αρχεία που υπάρχουν στο έργο (Scripts, Blueprints, Γραφικά, Αρχεία Ήχου και ότι θα χρειαστούμε για την δημιουργία του παιχνιδιού).

Στο μέση βλέπουμε το παράθυρο προβολής τρισδιάστατων (Viewport), μέσα στο οποίο θα μπορούμε τόσο να τοποθετούμε τα στοιχεία που χρειαζόμαστε με απλό Drag and Drop, να βλέπουμε πώς θα μοιάζει το τελικό προϊόν αλλά και να επεξεργαζόμαστε τα τοποθετημένα στοιχεία με βάση της εικόνας του τελικού προϊόντος που έχουμε στο μυαλό μας.

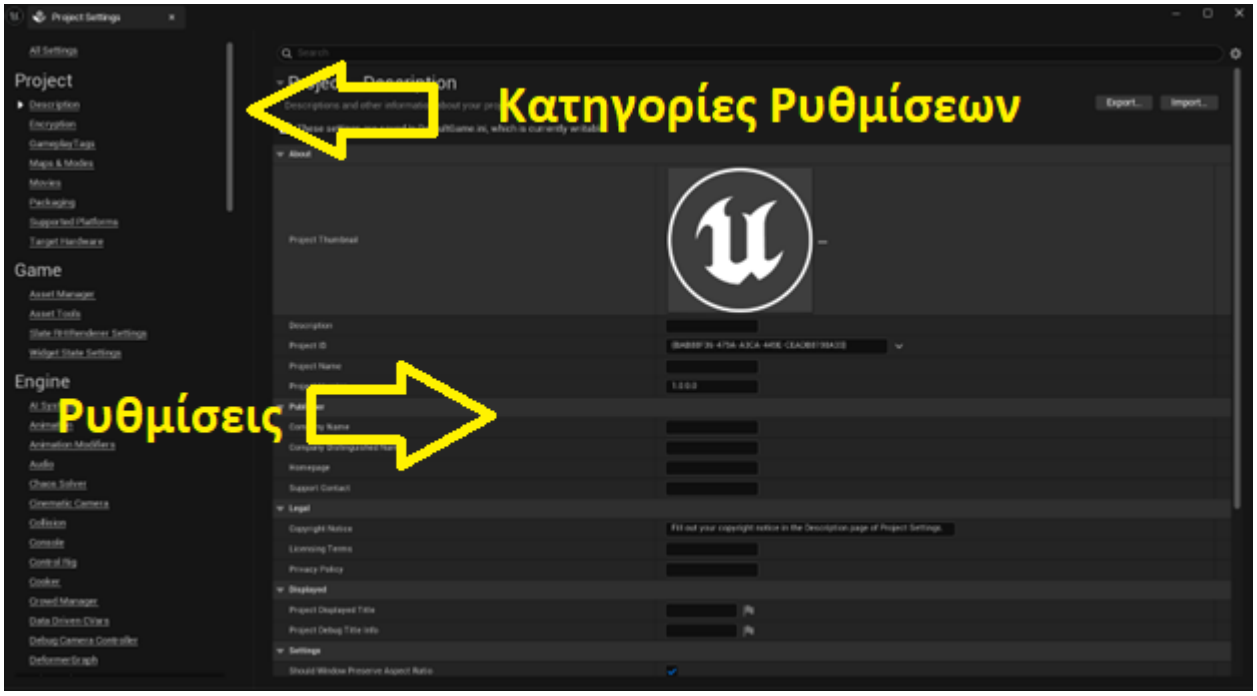
Στα δεξιά θα δούμε δυο παράθυρα. Το επάνω είναι το παράθυρο δομής σκηνής (Outliner), και περιέχει όλα τα στοιχεία που έχουμε τοποθετήσει στο Viewport σε ιεραρχική διάταξη. Στην κορυφή της ιεραρχίας πάντα θα βρίσκεται το Level αρχείο και από κάτω μπορούμε να έχουμε χωρίσουμε το περιεχόμενο σε φακέλους, ανάλογα την χρήση τους. Για παράδειγμα, θα μπορούσαμε να έχουμε δυο φακέλους, έναν για όλα τα αρχεία που χρησιμοποιεί ο παίκτης (Player Component) και έναν με όλο το περιεχόμενου που χρησιμοποιήσαμε για την δημιουργία του κόσμου (Landscape Component). Μέσα σε αυτούς μπορούμε να έχουμε και υποφακέλους για την ακόμη καλύτερη ιεράρχηση και δομή του περιεχομένου. Ακριβώς κάτω από το Outliner, βρίσκεται ένα μενού τριών επιλογών, λεπτομέρειες (Details), World partition και ρυθμίσεις κόσμου (World Settings). Στις λεπτομέρειες θα εμφανίζονται ρυθμίσεις και διάφορα επεξεργάσιμα στοιχεία του περιεχομένου που θα έχουμε επιλέξει είτε από το Viewport είτε από το Outliner. Για παράδειγμα, μέσα από το συγκεκριμένο μενού θα μπορούμε να προσθέσουμε γραφικά στο επιλεγμένο περιεχόμενο ή να του αλλάξουμε διαστάσεις (γίνεται και μέσω του Viewport). Το 3ο μενού, World Settings, επιτρέπει την

παγκόσμια διαμόρφωση παραμέτρων του επιπέδου Level και του κόσμου. Για παράδειγμα, μπορούμε να διαμορφώσουμε τα γενικά physics, όπως την βαρύτητα, και να επιλέξεις αν στον κόσμο θα είναι συνέχεια όλο περιεχόμενο φορτωμένο ή θα φορτώνεται δυναμικά με βάση την απόσταση από τον παίκτη. Το 2ο μενού, World Partition, θα αναλυθεί σε επόμενο κεφάλαιο.

Τέλος, στο επάνω μέρος του Editor θα δούμε το κεντρικό μενού, από κάτω το επίπεδο που έχουμε ανοιχτό (δεν μπορούμε να έχουμε παραπάνω από ένα) και ύστερα το μενού εργαλείων (Toolbar). Το toolbar περιέχει κουμπί αποθήκευσης, "Selection Mode" για την εναλλαγή μεταξύ διαφορετικών εργαλείων και λειτουργιών ("Select", "Landscape", "Foliage", "Geometry" και άλλα), προσθήκη αντικειμένου στο επίπεδο, δυο κουμπιά για την γρήγορη προσθήκη Blueprint και Cinematic, ένα μενού "Play / Pause / Stop", για έναρξη και τέλος προεπισκόπησης παιχνιδιού, το "Platforms", για το πακετάρισμα (Packaging) του παιχνιδιού στις διάφορες πλατφόρμες, και στην άκρη δεξιά το "Settings", απ' όπου μπορούμε να ενεργοποιήσουμε και να απενεργοποιήσουμε μερικές γενικές ρυθμίσεις στο έργο.

3.4 Ρυθμίσεις Έργου – Project Settings

Κατά την δημιουργία ενός έργου, μπορεί να χρειαστεί να αλλάξουμε μερικές ρυθμίσεις του. Μπορούμε να ανοίξουμε τις ρυθμίσεις από το κεντρικό μενού (Edit – Project Settings) ή από toolbar (Settings – Project Settings).



3.6) Παράθυρο Project Settings Unity

Οι ρυθμίσεις αυτές, χωρίζονται σε έξι βασικές κατηγορίες Project, Game, Engine, Editor, Platforms και Plugins.

Η κατηγορία Project περιέχει γενικές ρυθμίσεις. Μέσα από αυτές μπορείς να αλλάξεις όνομα στο έργο, να προσθέσεις περιγραφή, όνομα εταιρείας (αν υπάρχει), δικαιώματα (Copyrights), επίπεδο που θα

εμφανίζεται κατά την εκκίνηση του Editor, ρυθμίσεις που έχουν να κάνουν με την πλατφόρμα που προορίζεται το παιχνίδι, ρυθμίσεις ταινιών και πακεταρίσματος.

Η Game περιλαμβάνει ρυθμίσεις για τον τρόπο φόρτωσης των διάφορων αρχείων, τους καταλόγους που θα γίνεται αναζήτηση για αρχεία και τις κλάσεις που θα χρησιμοποιηθούν για τα πρωτεύοντα τύπου αρχείων. Συνίσταται αυτές τις ρυθμίσεις να μην τις αλλάζουμε και να χρησιμοποιούνται οι προκαθορισμένες, εκτός από πολύ ειδικών περιπτώσεων.

Η Engine ρυθμίζει την συμπεριφορά την μηχανής σύμφωνα με το πρόβλημα που έχει να επιλύσει. Χωρίζεται σε υποκατηγορίες που έχουν να κάνουν με την τεχνητή νοημοσύνη, με τον ήχο, με τις συγκρούσεις (Collisions), με τον ίδιο τον Editor και άλλα. Για ακόμη μια φορά, συνιστάται η χρήση των προκαθορισμένων ρυθμίσεων, με την πιθανή τροποποίηση τους κατά την διάρκεια δημιουργίας του παιχνιδιού.

Η Editor αφορά ρυθμίσεις που έχουν να κάνουν με την διεπαφή. Αλλάζει τον τρόπο που εμφανίζονται ή λειτουργούν κάποια συστατικά της μηχανής, καθώς και την απλοποίηση των αντικειμένων που εισάγονται.

Μέσα από την Platform, μπορούμε να παραμετροποίησουμε τις τεχνικές λεπτομέρειες της εφαρμογής ανά πλατφόρμα. Μπορούμε για παράδειγμα να επιλέξουμε αν θα χρειάζεται DirectX και ποιας έκδοσης, αν θα απαιτεί προσομοίωση του DirectX σε κινητά, αν θα κάνει χρήση του Shader Model 5 ακόμα και το εικονίδιο της εφαρμογής.

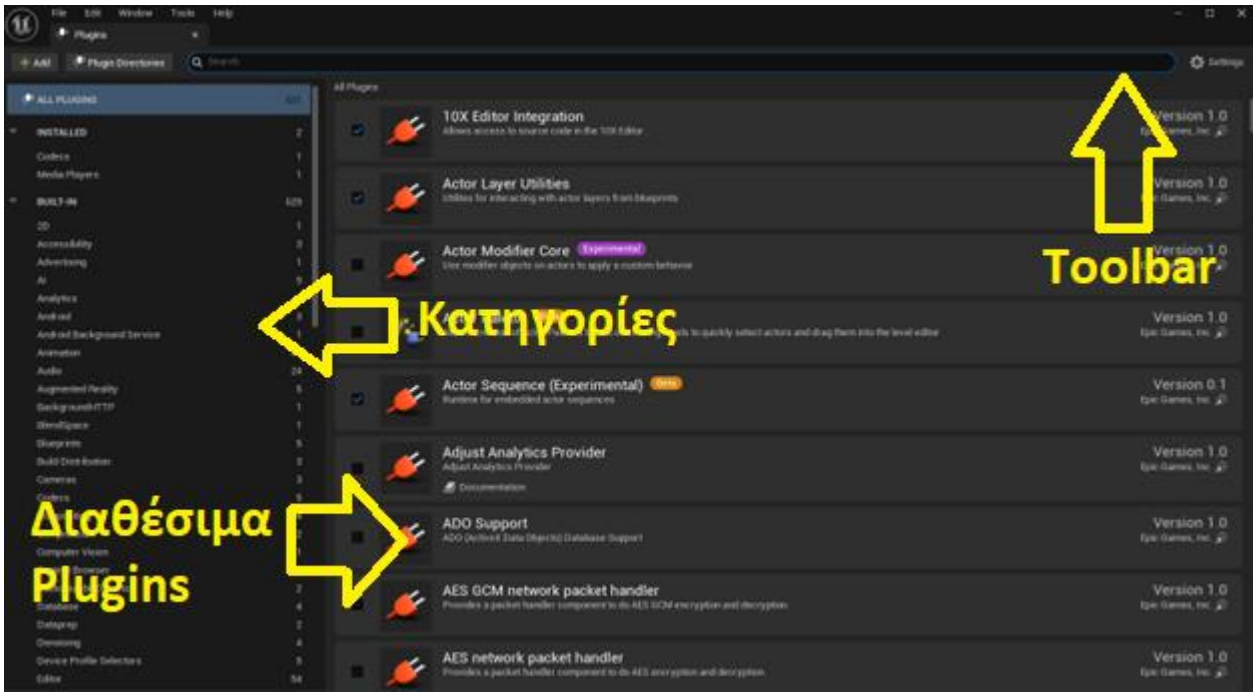
Τελευταία κατηγορία είναι τα Plugins. Περιέχει ρυθμίσεις για βασικά επεκτάσιμα που χρησιμοποιεί η μηχανή. Για παράδειγμα μπορούμε να ρυθμίσουμε την μηχανή για την χρήση script σε γλώσσα Python.

3.5 Επεκτάσιμα - Plugins

Εκτός από τις ρυθμίσεις ενός έργου, κανείς πρέπει να μπορεί να ενεργοποιήσει ή και υποπεριπτώσεις να απενεργοποιήσει τα plugins. Με τον όρο plugins εννοούμε κομμάτια λογισμικού που όταν προστεθούν σε ένα βασικό πρόγραμμα, επεκτείνουν τις δυνατότητες του, δίνοντας του επιπλέον λειτουργίες και χωρίς να αλλάζει ο πυρήνας του προγράμματος. Για να ανοίξουμε το παράθυρο των plugins, μπορούμε να το κάνουμε με τους ίδιους δυο τρόπους όπως το άνοιγμα των ρυθμίσεων έργου. Είτε μέσω του κεντρικού μενού (Edit – Plugins) είτε μέσω του toolbar (Settings – Plugins).

Η Unreal διαθέτει από μόνη της 631 διαφορετικά plugins. Επειδή ο όγκος είναι πολύ μεγάλος, στο δεξιά του παράθυρου, χωρίζονται ανά κατηγορίες. Για παράδειγμα, η κατηγορία τεχνητής νοημοσύνης AI περιέχει όλα τα plugins που έχουν να κάνουν με αυτήν, ενώ η κατηγορία εικονικής πραγματικότητας (Virtual Reality) περιέχει όλα τα plugins που πιθανώς να χρειαστεί κάποιος για την δημιουργία ενός τέτοιο παιχνιδιού.

Αν ξέρουμε ποιο plugin θέλουμε να χρησιμοποιήσουμε, τότε από το επάνω μέρος μπορούμε να ψάξουμε αυτό που θέλουμε. Αριστερά από την γραμμή αναζήτησης θα βρούμε 2 κουμπιά για προσθήκη δικού μας, κατά παραγγελία plugin, και για προσθήκη νέου φακέλου, στον οποίο η Unreal θα μπορεί να ψάχνει επιπλέον plugin. Στα δεξιά της γραμμής αναζήτησης, υπάρχει το κουμπί Setting, από το οποίο μπορούμε να φιλτράρουμε τα plugin ώστε να εμφανίζονται μόνο τα ενεργοποιημένα ή μόνο τα απενεργοποιημένα.



3.7) Παράθυρο Plugins Unreal

3.6 Ανάλυση Τεχνολογιών και Επεκτάσιμων (Plugin)

Όπως ήδη αναφέρθηκε, η Unreal προσφέρει μια πληθώρα τεχνολογιών για την διευκόλυνση των developer στην δημιουργία παιχνιδιών. Παρακάτω θα αναλυθούν μερικές από αυτές που, κατά την γνώμη μου, είτε ήδη παίζουν σημαντικό ρόλο στην δημιουργία παιχνιδιών, είτε θα αποτελέσουν στο μέλλον αναπόσπαστο κομμάτι.

Μετά την ανάλυση της κάθε τεχνολογίας, θα παρατεθούν εικόνες καθώς και επεξήγηση στην χρήση όσων χρησιμοποιήθηκαν, με βάση την προσωπική μου εμπειρία σε αυτές κατά την δημιουργία του παιχνιδιού που συνοδεύει το κείμενο,

3.6.1 Blueprints

Το 2014 η Epic παρουσίασε τα Blueprints. Αμέσως αποτέλεσαν ένα από τα πιο ισχυρά κομμάτια της σύγχρονης ανάπτυξης παιχνιδιού με Unreal, αφού δίνει την δυνατότητα ανάπτυξης λογικής παιχνιδιού χωρίς την χρήση κώδικα. Μέσω αυτών, developers και όχι (Game Designers ή Artists) μπορούν να δημιουργήσουν διάφορες λειτουργίες, να κατασκευάσουν μηχανισμούς ή και να ορίσουν αλληλεπιδράσεις μεταξύ component, οπτικά, συνδέοντας μεταξύ τους τους Κόμβους (Nodes).

Τα Blueprints θεωρούνται κλάσεις αντικειμένων εμπλουτισμένα με γραφικά χαρακτηριστικά, μεταβλητές και λειτουργίες, για την διεύρυνση των ικανοτήτων των αντικειμένων του παιχνιδιού. Μέσω αυτών,

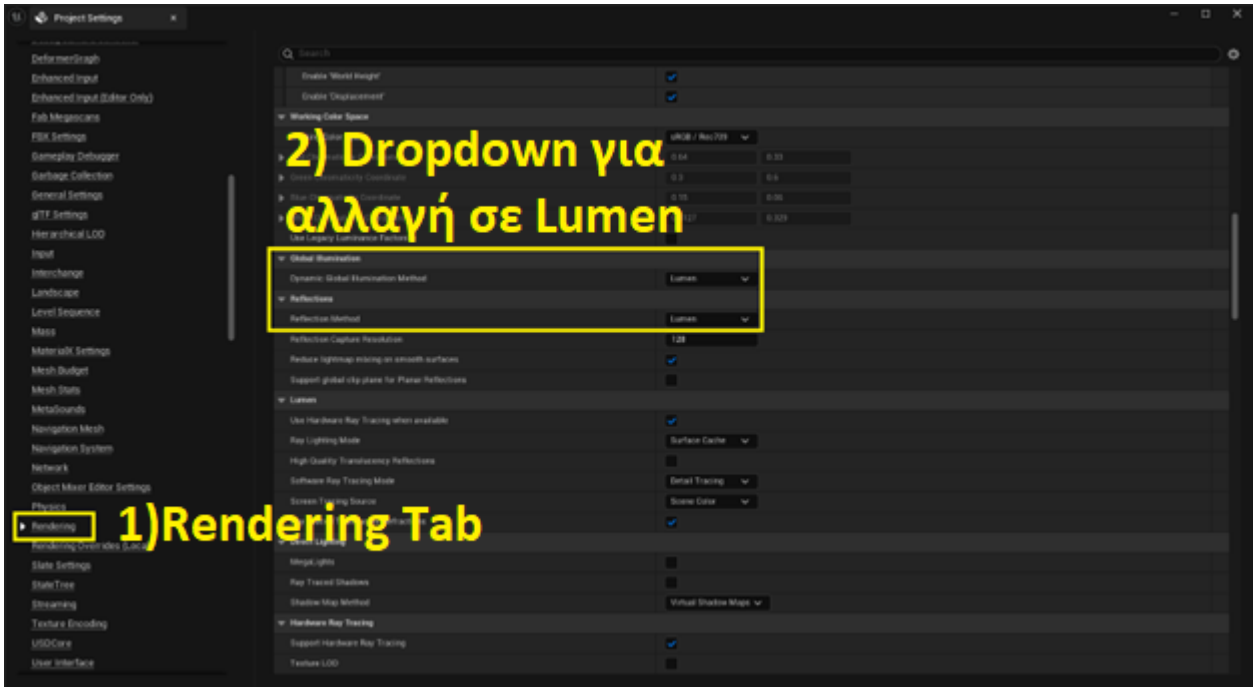
μπορούμε να έχουμε την καταχώρηση εντολών από τον παίκτη, αλληλεπίδραση με άλλα blueprint, αλλά και την αλληλεπίδραση μεταξύ 2 ή περισσότερών άλλων κόμβων μέσα στο ίδιο το Blueprint.

Παρότι όμως, η χρήση τους, προσφέρει μια γρήγορη ανάπτυξη και δοκιμή λογικής, διευκόλυνση στην Απασφαλμάτωση (Debugging), ευκολία στην συνεργασία μεταξύ διαφόρων ρόλων στο έργο, χωρίς αυτοί να χρειάζεται να έχουν καλή γνώση κώδικα, και την ευκολία επαναχρησιμοποίησης τους, η ίδια η Epic ενθαρρύνει την υβριδική ανάπτυξη (συνδυασμό Blueprints με C++). [12]

3.6.2 Global Illumination System - Lumen

Το Lumen είναι το σύστημα φωτισμού που έχει η UE5. Προσδίδει ένα νέο επίπεδο ρεαλισμού, αφαιρώντας την ανάγκη, κάποιος να έχει αποθηκευμένο προκαταβολικά τον φωτισμό σε *Υφές (Lightmaps)*. Με τον *έμμεσο φωτισμό (Global Illumination)* και την, βασισμένη στην *Ιχνηλάτηση Ακτινοβολίας (Ray-tracing)*, *Αντανάκλαση Χώρου Οθόνης (Screen Space Reflection (SSR))*, το σύστημα αναλύει τις μεταβολές που γίνονται και αναπροσαρμόζει τον φωτισμό και της αντανακλάσεις του παιχνιδιού καταλληλά σε πραγματικό χρόνο.

Για την ενεργοποίηση του, ο χρήστης πρέπει να μεταφερθεί στο παράθυρο ρυθμίσεων του έργου και να ψάξει τις ρυθμίσεις Global Illumination στο tab, Rendering (Project Settings – Engine – Rendering – Global Illumination) και να επιλέξει Lumen από το Dropdown Menu. [13]



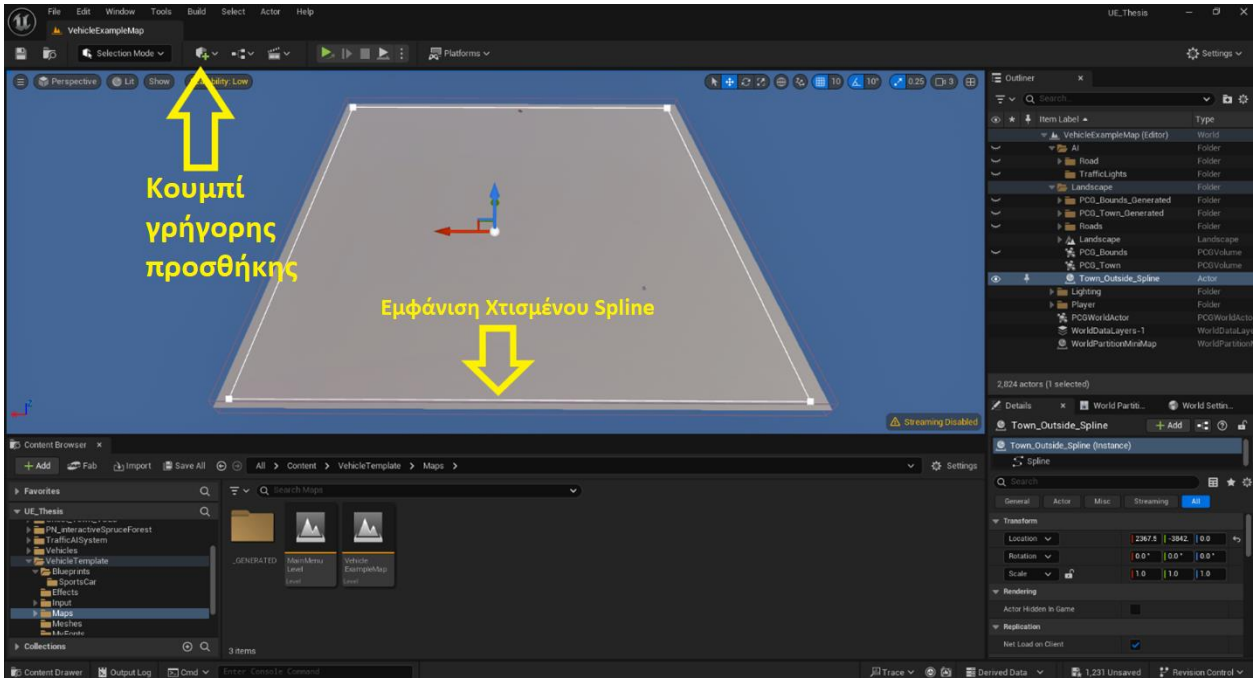
3.8)Ρυθμίσεις Ενεργοποίησης Lumen

3.6.3 Splines – Καμπύλες Ελέγχου Περιβάλλοντος

Τα Splines είναι καμπύλες, τοποθετημένες με τέτοιο τρόπο που επιτρέπουν την τοποθέτηση διάφορων αντικειμένων με παραμετρικό τρόπο. Κανείς μπορεί να τα χρησιμοποιήσει για να δημιουργήσει δρόμους, ποτάμια, φράκτες ή και animated μονοπάτια. Πέρα από το απλό Spline που μπορεί να δημιουργηθεί, υπάρχει και το Landscape Spline, το οποίο εξυπηρετεί περισσότερο για τοπογραφική μοντελοποίηση. Οποιοδήποτε είδος Spline και να επιλεχθεί, μπορούν να χρησιμοποιηθούν ώστε ένα αντικείμενο να κινείτε κατά μήκος τους, σαν να ακολουθεί μια διαδρομή. [14]

Συνήθως δεν χρειάζονται ενεργοποίηση. Στην περίπτωση όμως που δεν είναι διαθέσιμα για χρήση από την αρχή, κανείς μπορεί να ενεργοποιήσει το plugin από το αντίστοιχο παράθυρο. Για την δημιουργία ενός απλού Spline, θα χρειαστεί να τοποθετήσουμε ένα *Spline Mesh Actor*. Αυτό θα το βρούμε εύκολα στο κουμπί γρήγορης προσθήκης (αριστερό βελάκι στην 3.5). Με την προσθήκη αυτή θα εμφανιστούν 2 Points ενωμένα με μια γραμμή. Ο χρήστης μπορεί να προσθέσει επιπλέον κορυφές, κάνοντας ότι σχήμα έχει ανάγκη για το έργο του, ενώ υπάρχει και η επιλογή *Βρόχος (Loop)* ώστε να ενωθούν με γραμμή το τελευταίο με το πρώτο point του Spline.

Προσωπικά, χρειάστηκα τα Splines για 2 λειτουργίες. Η 1η ήταν για να το χρησιμοποιήσω σαν σημείο αναφοράς στο *PCG Graph* (θα αναλυθεί στο επόμενο κεφάλαιο) του δάσους γύρω γύρω από το χάρτη. Η 2η περίπτωση χρήση τους, ήταν για να οριστεί η κατεύθυνση και η τοποθεσία κίνησης των αυτοκινήτων μη ελεγχόμενα από παίκτη (NPC – Non-Playable Character). Θα αναλυθεί ο τρόπος χρήσης τους στο κεφάλαιο 5.



3.9) Προσθήκη Απλού Spline

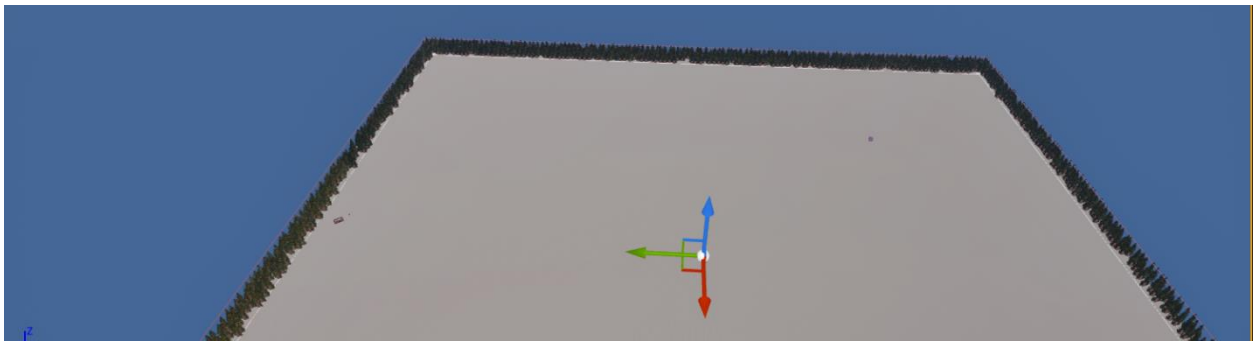
3.10) Spline Blueprint

3.6.4 Προγραμματισμένη Δημιουργία Περιεχομένου (PCG - Procedural Content Generation)

Το PCG είναι μια καινοτόμος τεχνολογία για την δυναμική δημιουργία περιεχομένου. Επιτρέπει το γρήγορη δημιουργία περιβάλλοντος με την χρήση παραμετροποιημένων κανόνων και αλγορίθμων. Σκοπός είναι να αυτοματοποιεί ένα μεγάλο μέρος της καλλιτεχνικής παραγωγής ενός παιχνιδιού. Στο γράφημα δημιουργίας λογικής του, θα βρούμε nodes για συγχρονισμό με διάφορα στοιχεία ενσωματωμένα στο παιχνίδι, μεταξύ άλλων και τα Splines. Η χρήση του προτιμάται για παιχνίδια *Ανοικτού Κόσμου (Open World Games)*, των οποίων η χειροκίνητη δημιουργία θα ήταν χρονοβόρα ή και αδύνατη ανάλογα με το μέγεθος του κόσμου. [15]

Στο κεφάλαιο 5 θα αναλυθεί ο τρόπος χρήσης του plugin μέσα από την χρήση που έκανα στην ανάπτυξη της δικιάς μου εφαρμογής. Χρησιμοποιήθηκε 2 φορές στην δημιουργία κόσμου.

3.11) Nodes Δημιουργίας PCG Ορίων

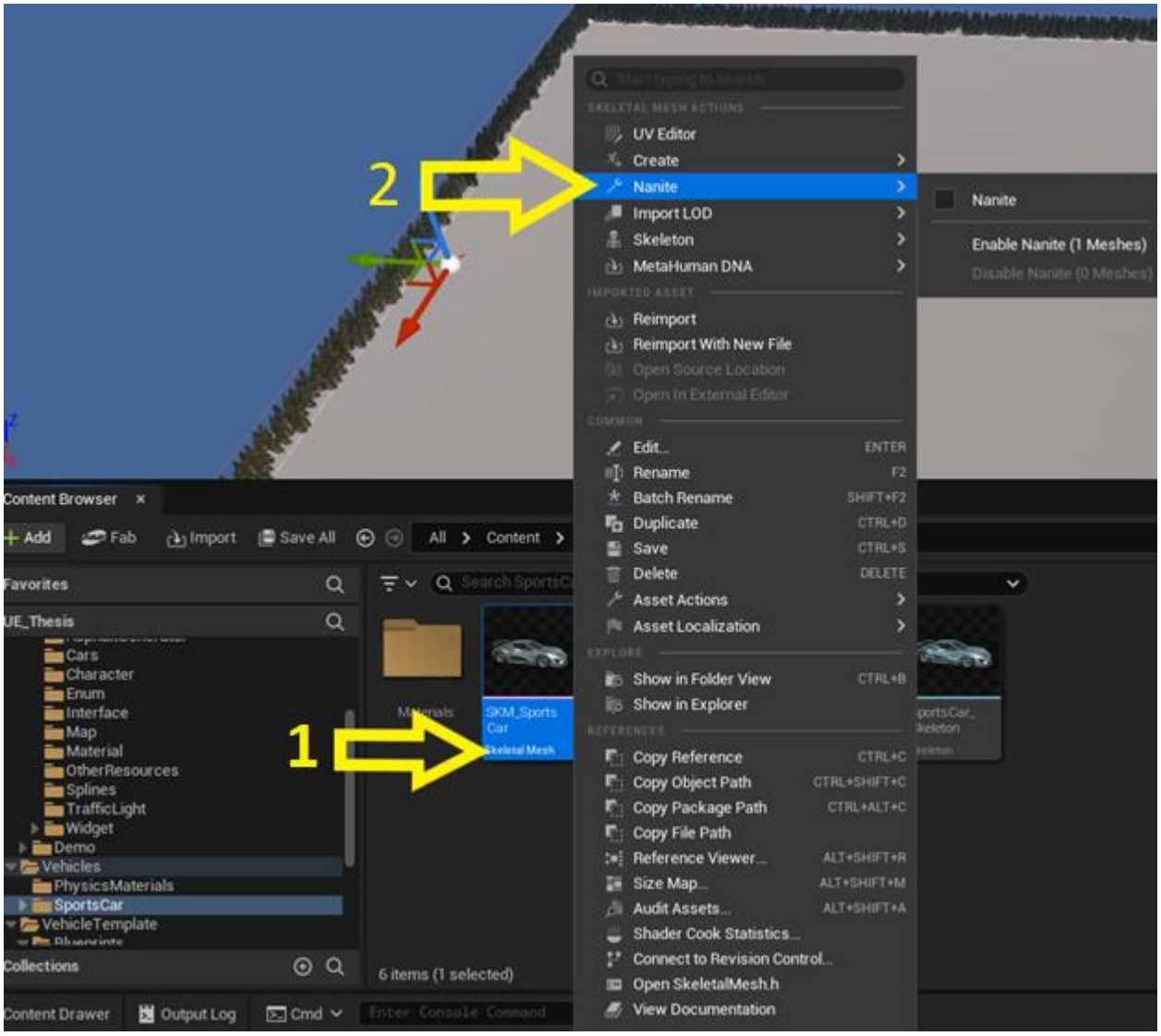


3.12)Αποτέλεσμα PCG εικόνας 3.7

3.6.5 Nanite – Εικονική Γεωμετρία Υψηλής Πιστότητας

Nanites είναι η τεχνολογία για την αντιμετώπιση του περιορισμού στον αριθμό πολυγώνων που μπορούν να αποδοθούν σε πραγματικό χρόνο. Αντικαθιστούν το στατικό Level of Detail (LOD), με χρήση εικονικής γεωμετρίας, και βασιζόμενο στην απόσταση και την γωνία θέασης, προσαρμόζουν τον δυναμικά προβληθέντα αριθμό πολυγώνων. Διευκολύνεται με αυτό τον τρόπο, η εισαγωγή κινηματογραφικής ποιότητας περιεχομένου, και επιτρέπει ρεαλιστικά περιβάλλοντα μεγάλης κλίμακας. [16]

Για να χτίσουμε τα Nanites κάποιου περιεχομένου, πρέπει να κάνουμε δεξί κλικ στο *Μοντέλο (Skeletal Mesh)* στον Περιηγητής Περιεχομένου (Content Browser) που θέλουμε και να τα ενεργοποιήσουμε. Προτείνεται, αφού χτιστούν τα Nanites για όλο το περιεχόμενο που θέλουμε, να γίνει αποθήκευση του έργου και επανεκκίνηση της μηχανής.



3.13) Ενεργοποίηση Nanite περιεχομένου

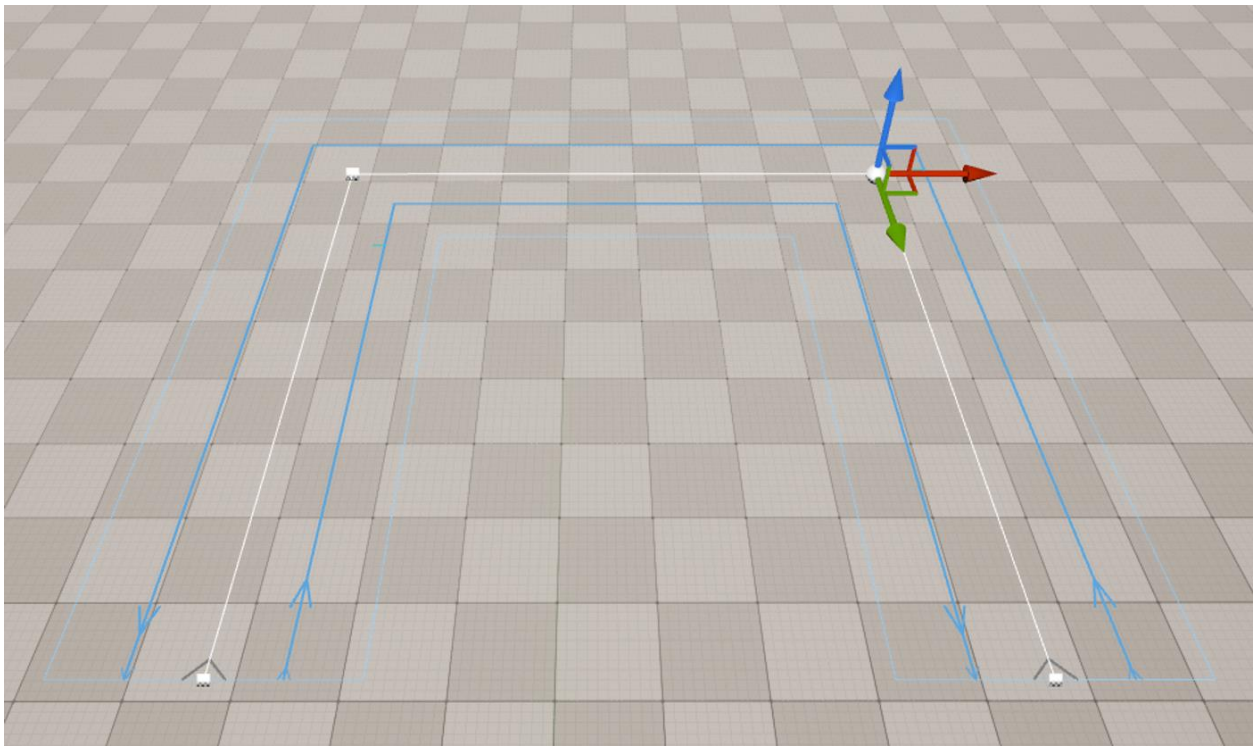
3.6.6 Zone Graph – Ζωνοποίηση Πλοήγησης

Το Zone Graph είναι το σύστημα με το οποίο μπορούμε να καθορίσουμε ζώνες πλοήγησης και κυκλοφορίας. Η κύρια χρήση του είναι για την δημιουργία συστημάτων κάθε είδους μεταφορών, για παράδειγμα οδικό δίκτυο και AI πλοήγηση. Επίσης ελέγχει το πλήθος των NPC χαρακτήρων και οχημάτων. Με τα Zone Graphs μπορούν να δημιουργηθούν σύνθετα πλέγματα μεταξύ κόμβων, επιτρέποντας την προσθήκη λογικής πλοήγησης σε μια μεγάλη ποικιλία σεναρίων.

Προσδιορίζονται με *Πολυγωνικούς Όγκους (Polygonal Volumes)* και οδηγήσεις με την χρήση Splines. [17]

Τα Zone Graph είναι ακόμα σε Experimental επίπεδο, οπότε κανείς μπορεί να βρε σφάλματα. Γι' αυτό κιόλας, κανείς πρέπει να ανοίξει το παράθυρο με τα Plugins και να τα ενεργοποιήσει, και είναι και ο λόγος

που δεν χρησιμοποιήθηκαν στο project μου. Ωστόσο, θα αποτελέσουν ένα πολύ δυνατό εργαλείο στην δημιουργία σύνθετων οδικών πλεγμάτων.



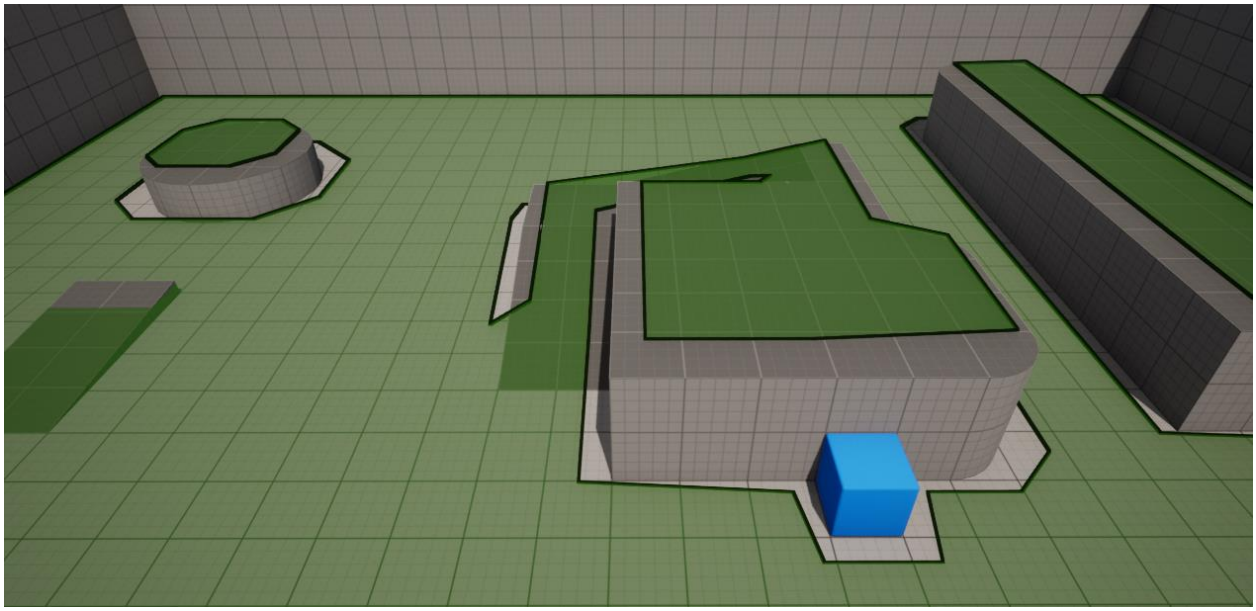
3.14) Απλό Zone Graph με γωνίες 90 Μοιρών

3.6.7 NavMesh – Πλοήγηση Τεχνητής Νοημοσύνης

Το NavMesh είναι ακόμα ένα εργαλείο για την πλοήγηση AI χαρακτήρων. Με το NavMesh καθορίζουμε ένα χώρο στον κόσμο μας μέσα στον οποίο μπορούν να κινηθούν οι NPC χαρακτήρες. Χρειάζεται να τοποθετηθεί ένα *NavMeshBoundsVolume*, και να το τοποθετήσουμε με τις κατάλληλες διαστάσεις, εκεί που θέλουμε. Με το χτίσιμο του, η μηχανή θα δημιουργήσει μια πράσινη επιφάνεια, μόνο μέσα στην οποία μπορούν να κινηθούν οι χαρακτήρες που θα θέσουμε. [18]

Το NavMesh είναι κατάλληλο για παιχνίδια όπου οι χαρακτήρες χρειάζεται να έχουν μια πιο "έξυπνη" πλοήγηση μέσα σε ένα περιβάλλον, που μπορεί να περιέχει εμπόδια και διαφορετικά επίπεδα εδάφους.

Όπως και στην προσθήκη απλού Spline, μέσα από το κουμπί γρήγορης προσθήκης (αριστερό βελάκι στην εικόνα 3.5) , θα πρέπει να βρεθεί και να τοποθετηθεί στον κόσμο μας ένα NavMeshBoundsVolume, και στην συνέχεια να προσαρμόσουμε τις διαστάσεις του κύβου, έτσι ώστε να καλύπτει την περιοχή που θέλουμε. Τέλος, πρέπει να γίνει Build («Build» Tab – Build Paths) και να προστεθεί η κατάλληλη λογική στον χαρακτήρα μας.



3.15) Όψη Χτισμένου NavMesh

3.6.8 Sequencer

Το Sequencer είναι ένα ισχυρό εργαλείο κινηματογραφικής σκηνοθεσίας της Unreal Engine το οποίο έχει σχεδιαστεί να εκχωρεί την οπτικοποίηση από πολλές πηγές σε πραγματικό χρόνο. Ο χρήστης μπορεί να οικοδομήσει και να επεξεργαστεί sequences “Level Sequences” με time lines που περιέχουν tracks για animation, κίνησης κάμερας, ηχητικά εφέ και scripted events. Το περιβάλλον exe Sequencer μοιάζει με ένα εργαλείο επεξεργασίας βίντεο, farming keyframing, nesting πολλαπλών scowls, και real-time προεπισκόπηση. Είναι πλήρως υποστηρίζεται από το Blueprint System και C ++,^ και η δυνατότητα σύνδεσης σε trigger events καθιστά πιο εύκολη την ενσωμάτωση της κινηματογραφικής διαδικασίας με το gameplay. Χρήσιμο στην κατάταξη cutscenes, διαδραστικά trailer, και scripted gameplay αλληλεπιδράσεις, το Sequencer prep Int ένα επίπεδο οδήγησης που απλώς εξουσιάζει την ανάγκη για ένα external cinematic το εργαλείο. [19]

3.6.9 World Partition

Το World Partition είναι ένα σύστημα που αντικαθιστά το παραδοσιακό World Composition της Unreal Engine, επιτρέποντας την αυτόματη διαχείριση και streaming μεγάλων ανοιχτών κόσμων. Αντί να δημιουργεί ο χρήστης χειροκίνητα sub-levels και να ορίζει visibility ranges, το World Partition διαιρεί τον χάρτη σε grid cells, που φορτώνονται δυναμικά ανάλογα με τη θέση του ήρωα. Κάθε cell περιλαμβάνει τα απαιτούμενα δεδομένα σχηματικής απεικόνισης, φωτισμού και scripting, προσφέροντας βελτιωμένη απόδοση και οργάνωση. Υποστηρίζει Data Layers, για εναλλακτικές εκδοχές του ίδιου κόσμου (π.χ. μέρα/νύχτα ή κατεστραμμένο/ακέραιο περιβάλλον), ενώ λειτουργεί και με One File Per Actor σύστημα, ευνοώντας τη συνεργασία σε ομάδες ανάπτυξης. [20]

3.7 Μειονεκτήματα Τεχνολογιών

Όπως είδαμε, υπάρχουν τεχνολογίες που μειώνουν δραματικά τον χρόνο δημιουργίας ενός παιχνιδιού, και άλλες που αυξάνουν την ποιότητα του. Όμως δεν έρχονται χωρίς τα δικά τους μειονεκτήματα. Όσο απλά και αν φαίνονται στην χρήση, ένας που τώρα μπαίνει στον χώρο και δεν γνωρίζει καλά ακόμα, μπορεί πολύ εύκολα να χαθεί. Η καμπύλη εκμάθησης τους, ποικίλει από την εμπειρία και τις ήδη υπάρχουσες τεχνικές γνώσεις του developer.

Για παράδειγμα, κατά την δημιουργία ενός Blueprint, θα προσέξει κανείς, ότι υπάρχουν πολλά nodes τα οποία μοιάζουν μεταξύ τους, με διαφορετικές λειτουργίες όμως. Όποτε καλό θα ήταν, να γίνει πρώτα μια στοιχειώδη εξοικειώσει με τα διαθέσιμα εργαλεία που προσφέρει η Unreal, για καλή χρήση τους

Ένα άλλο μειονέκτημα, είναι η χρήση πόρων της μηχανής και έπειτα του ίδιου του παιχνιδιού. Με την πάροδο του χρόνου και με την προσθήκη τεχνολογιών στο παιχνίδι, παρατηρούσα ότι η χρήση μνήμης που έκανε ο editor, απλά και μόνο σε idle κατάσταση, αυξανόταν. Στο τελικό στάδιο του project μου, με τον editor απλά ανοικτό και σε κατάσταση αδράνειας (idle), χρησιμοποιούνταν ~25% της μνήμης RAM. Σε αντίθεση, το άνοιγμα του παιχνιδιού ανέβαζε το ποσοστό χρήσης RAM κατά ~10% και έφτανε τα ποσοστά CPU και GPU στο 100%, τουλάχιστον μέχρι να φορτωθούν όλα τα γραφικά. Μετά σταθεροποιούνταν στο ~ 40% ο CPU, ενώ της GPU έπεφτε σχεδόν στο 0%. Στην περίπτωση επανεκκίνησης του κόσμου μετά από τρακάρισμα, το παιχνίδι κολλούσε για λίγα δευτερόλεπτα, έως ότου ξαναφορτωθούν τα γραφικά.

Θα μου επιτραπεί να επισημάνω, ότι κατά την διάρκεια της δημιουργίας του παιχνιδιού, έγινε αναβάθμιση του hardware του υπολογιστή μου. Με τα προηγούμενα, να δουλεύουν σχεδόν στο 100% ακόμα και σε idle κατάσταση ο editor. Το άνοιγμα του παιχνιδιού μέσα από τον editor για προεπισκόπηση (preview) χωρίς να γίνει Build και Packaging, εμφάνιζε μήνυμα λάθους στο Viewport του παίκτη, για την ανεπάρκεια μνήμης που διέθετα για την σωστή λειτουργία.

3.8 Υποστήριξη και Κοινότητα

Κατά την υλοποίηση του project μου, χρειάστηκε αρκετές φορές να ανατρέξω για κάποια βοήθεια. Το Documentation που παρέχει η Epic, δυστυχώς καλύπτει ένα μικρό κομμάτι, και δύσκολα κάποιος θα μπορέσει να βρει λύση εκεί.

Όταν δεν βρέθηκε λύση εκεί, στράφηκα στο Forum της Unreal. Ξεκίνησα έτσι να ψάχνω τις ήδη υπάρχουσες ερωτήσεις, με σκοπό να βρω αυτό που χρειαζόμουν ήδη απαντημένο. Κάποιες κατάφερα να τις βρω και να της απαντήσω. Άλλες όμως όχι.

Έτσι, τελευταία μου επιλογή, ήταν να κάνω ο ίδιος την ερώτηση, τόσο στο Forum όσο και σε έναν Discord Server για την Unreal. Με τις ενότητες του Forum να μην είναι απολύτως κατανοητές, αναρωτιόμουν κάθε φορά αν έθετα την ερώτηση στην σωστή ομάδα. Κάποιες φορές είχα βοήθεια, με αποτέλεσμα να είχα και λύση ή τουλάχιστον καθοδήγηση. Τις περισσότερες φορές όμως, δεν έπαιρνα απάντηση, συνεχίζοντας έτσι να είμαι μόνος μου, στο ψάξιμο για λύση του προβλήματος που αντιμετώπιζα. Στην περίπτωση που έπαιρνα απάντηση, αυτή ερχόταν αρκετά αργότερα, ίσως και μια μέρα μετά.

3.9 Κατάστημα – Marketplace (Fab)

Για την απόκτηση έτοιμου περιεχομένου, η Epic παρέχει στους χρήστες το *Marketplace (*ή Fab όπως μετονομάστηκε). Το Fab είναι ένας χώρος που μπορείς να επισκεφτείς μέσα από το project σου, δίνοντάς σου την δυνατότητα να κατεβάσεις και να ενσωματώσεις κατευθείαν το περιεχόμενο που χρειάζεσαι. Υπάρχει διαθέσιμο και στο διαδίκτυο. Εκεί μπορείς να δεις αναλυτικότερα κάποιο περιεχόμενο, αξιολογώντας πιθανώς καλύτερα αν σε ενδιαφέρει ή όχι. Κάθε μήνα η Epic Games, δίνει δωρεάν 3 περιεχόμενα από το marketplace.

Η διαφορά είναι ότι από το διαδίκτυο, θα συνδεθεί με τον λογαριασμό σου, και θα πρέπει να ανοιχτεί το project με τον ίδιο λογαριασμό. Και σε αυτή την περίπτωση, το παράθυρο του Fab πρέπει να ανοιχτεί, για να κατεβεί και να γίνει εισαγωγή (import) του plugin ή περιεχομένου στο project.

Το μεγαλύτερο μέρος του περιεχομένου που διαθέτει το Fab, είναι επί πληρωμή αλλά είναι υψηλής ποιότητας.

Κεφάλαιο 4ο Unity

Η Unity είναι και αυτή, μια από τις πιο δημοφιλής μηχανές ανάπτυξης παιχνιδιών και όχι μόνο. Δημιουργήθηκε ως την λύση στην ανάγκη που υπήρχε, για μια πιο προσιτή, σε προγραμματιστές ανεξαρτήτου υπόβαθρου, μηχανή ανάπτυξης. Ενώ η εταιρεία Unity Technologies ιδρύθηκε το 2004, η 1η έκδοση της μηχανής κυκλοφόρησε ένα χρόνο μετά, το 2005. Η ευελιξία που παρέχει στους χρήστες της, η υποστήριξη των 2D και 3D γραφικών, την καθιστά ιδανική για ανάπτυξη παιχνιδιών σε κάθε είδους πλατφόρμα (υπολογιστή, κινητά ,κονσόλα και συσκευές επαυξημένης/εικονικής πραγματικότητας). Εκτός από παιχνίδια μπορεί να χρησιμοποιηθεί σε έργα που έχουν να σχέση με τον κινηματογράφο, στην αυτοκίνητό-βιομηχανία και την αρχιτεκτονική. Πολλοί διάσημοι τίτλοι χρησιμοποιούν τις δυνατότητες αυτής της μηχανής, όπως το "Hollow Knight", το AR παιχνίδι κινητού "Pokémon Go", και ταινίες όπως "Ο Βασιλιάς των Λιονταριών" του 2019.

4.1 Ιστορία

Η ιδέα της Unity γεννήθηκε το 2004, όταν 3 Δανοί φίλοι, ξεκίνησαν την δημιουργία της ως την ανάγκη για μια οικονομική, εύκολη προς την χρήση της και πολυπλατφορμική (cross-platform) μηχανή ανάπτυξης παιχνιδιών. Ξεκίνησαν έτσι την κατασκευή της, και ένα χρόνο αργότερα, το 2005, λάνσαραν την πρώτη έκδοση της για Mac OS στο Apple Worldwide Developers Conference. [21]

Η επιτυχία της μηχανής βασίστηκε κατά κύριο λόγο στους ανεξάρτητους προγραμματιστές (indie developers) που δεν είχαν την οικονομική δυνατότητα να αποκτήσουν πιο εμπορικές και ακριβές μηχανές ανάπτυξης. Για την καλύτερη εξέταση των δυνατοτήτων αλλά και την απόδοση της μηχανής, η Over The Edge Entertainment (OTEE), η εταιρεία που ιδρύθηκε από τους δημιουργούς της Unity, ξόδεψε τους 5 τελευταίους μήνες πριν το λανσάρισμα της μηχανής, δημιουργώντας το παιχνίδι "GooBall". Μέσα από τα σχόλια και της παρατηρήσεις, αναβάθμισαν τις πτυχές της μηχανής που δεν δούλευαν όπως θα ήθελαν, και βελτίωσαν τις λειτουργίες της. Η Unity έγινε διάσημη σε indie developers, ειδικά για iOS και Android εφαρμογές. [21]

Το 2007, εκδόθηκε η Unity 2.0. Εισήχθησαν νέες λειτουργίες, μεταξύ αυτών *Δυναμικές Σκιές (Dynamic Shadow)* και το *Σύστημα Απεικόνισης Εδάφους (Terrain Engine)*. Η μηχανή απέκτησε μια μεγαλύτερη ευελιξία, όταν προστέθηκε υποστήριξη για Windows και προγράμματα περιήγησης (Web Browsers), στην ίδια έκδοση.

Τρία χρόνια μετά, το 2010, έφτασε η Unity 3.0. Αύξησε τις γραφικές δυνατότητες της, επιτρέποντας έτσι καλύτερα γραφικά για παιχνίδια υπολογιστή και κόνσολών. Ενσωμάτωσε λειτουργίες όπως *Εργαλεία Lightmap, Αναβαθμισμένη Καθυστερημένη Απόδοση (Deferred Rendering), Αυτόματη απεικόνιση Υπεριώδους Ακτινοβολίας (Automatic UV Mapping)*. Οι άδεις χρήσεις χωρίστηκαν σε 2 ειδών: δωρεάν άδεια για προσωπική ή εκπαιδευτική χρήση και επιπληρωμή άδεια για επαγγελματική χρήση. [21]

Το 2012, η μηχανή αναβαθμίστηκε στην 4η έκδοσή της, Unity 4.0. Έκδοση ορόσημο, αφού μαζί της ήρθε και το "Mecanim", ένα σύστημα προηγμένης απεικόνισης κίνησης, που επέτρεπε στον χαρακτήρα να πραγματοποιήσει πιο ρεαλιστικές και σύνθετες κινήσεις. Επιπλέον. Προστέθηκε υποστήριξη για Linux λειτουργικό, καθώς και για κονσόλες όπως Xbox360, Nintendo Wii U και PlayStation 3. Στην

συγκεκριμένη έκδοση, έκανε την πρώτη του εμφάνιση το κατάστημα της Unity (Unity Asset Store), επιτρέποντας την αγορά και την πώληση περιεχομένου. [21]

Το 2015, η Unity πέρασε στην επόμενη φάση της, Unity 5.0. Από αυτό το σημείο, η Unity σταθεροποίησε την θέση της ως μια από τις δυνατότερες μηχανές ανάπτυξης. Πλέον, παρέχει στους developers πρόσβαση σε μια ευρύτερη λίστα υποστηριζόμενων πλατφορμών, με κάποιες από αυτές να είναι τα PlayStation 4, Xbox One, Nintendo Switch καθώς και Hardware VR. Πρόσθεσε επίσης λειτουργίες όπως το *Global Illumination, Ανακλάσεις Πραγματικού Χρόνου (Real-Time Reflections)* και η *Απόδοση βάσει Φυσικών Ιδιοτήτων (Physically Based Rendering – PBR).* [21]

Από το 2017 μέχρι το 2021, κάθε χρόνο κυκλοφορούσε μια νέα έκδοση, κάθε μια παίρνοντας το όνομα της χρονιάς που κυκλοφόρησε, για παράδειγμα Unity 2019. Πατώντας πάνω στην Unity 5, η Unity 2017 και 2018 πρόσφερε πολλά εργαλεία κινηματογραφικού ενδιαφέροντος *(Cinemachine και Timeline)*, ανοίγοντας τον δρόμο για πιο όμορφες σκηνές αφήγησης (cutscenes). Χωρίς να σταματάει η προσθήκη καινούργιων τεχνολογιών, μέσα σε αυτά τα χρόνια προστέθηκαν πολλές νέες λειτουργίες για την δημιουργία τεχνητής νοημοσύνης (*Machine Learning Agents – ML Agents)*, για την αναβάθμιση της απόδοσης (*High Definition Render Pipeline – HDPR, Universal Render Pipeline – URP, Data Oriented Technology Stack - DOTS),* για την ανάλυση και βελτιστοποίηση του κώδικα (*Burst Compiler , Incremental Garbage Collection – IGC)* και πολλές ακόμα. [21]

Η τελευταία και τρέχουσα έκδοση, κυκλοφόρησε τον Οκτώβριο του 2024, Unity 6. Μεγάλες βελτιστοποιήσεις γίναν σε πολλές λειτουργίες της μηχανής. Ενισχύθηκε η διαδικασία απόδοσης, απλούστευση της δημιουργίας παιχνιδιών πολλών παικτών (multiplayer), δυνατότητα επίτευξης πιο ρεαλιστικών γραφικών (Adaptive Probe Volumes – APV) και πολλά ακόμα. [22]

4.2 Τεχνικές Προδιαγραφές

Όπως και στην Unreal, η χρήση της Unity προϋποθέτει κάποιες ελάχιστες απαιτήσεις συστήματος. Βέβαια, οι αναφορά αυτώ των απαιτήσεων είναι πιο γενικευμένες. Συγκεκριμένα αναφέρονται το λειτουργικό, η ελάχιστη αρχιτεκτονική του επεξεργαστή και τα υποστηριζόμενα API της κάρτας γραφικών. Όσο αναφορά την RAM, μια ελάχιστη μνήμη των 8GB είναι απαραίτητη μόνο για την χρήση του Editor. [23]

Unity Editor system requirements

This section lists the hardware and software requirements to run the Unity Editor. Actual performance and rendering quality might vary depending on the complexity of your project.

For all operating systems, the Unity Editor is supported on workstations or laptop form factors running without emulation, container or compatibility layer.

Operating system	Operating system version	CPU	Graphics API	Additional requirements
Windows	Windows 10 version 21H1 (build 19043) or newer (X64), Windows 11 21H2 (build 22000) or newer (Arm64)	X64 architecture with SSE2 instruction set support, Arm64	DX10, DX11, DX12 or Vulkan capable GPUs	Hardware vendor officially supported drivers
macOS	Big Sur 11 or newer	X64 architecture with SSE2 instruction set support (Intel processors) Apple M1 or above (Apple silicon-based processors)	Metal-capable Intel and AMD GPUs	Apple officially supported drivers (Intel processor) Rosetta 2 is required for Apple silicon devices running on either Apple silicon or Intel versions of the Unity Editor
Linux	Ubuntu 22.04, Ubuntu 24.04	X64 architecture with SSE2 instruction set support	OpenGL 3.2+ or Vulkan-capable, Nvidia and AMD GPUs	Gnome desktop environment running on top of X11 or Wayland windowing system, Nvidia official proprietary graphics driver, or AMD Mesa graphics driver. Other configuration and user environment as provided stock with the supported distribution (Kernel, Compositor, etc.) **Notes:** • **Ubuntu 22.04:** Wayland is supported with AMD graphics cards. • **Ubuntu 24.04:** Wayland is supported with AMD graphics cards and Nvidia graphics cards utilizing Nvidia proprietary graphics drivers 550 and above.

RAM recommendations for the Unity Editor

To run the Unity Editor on Windows, macOS, or Linux, a minimum of 8 GB RAM is recommended.

However, the amount of RAM required to load and run your project depends on your project's size and complexity. Larger and more complex projects require additional RAM.

4.1) Απαιτήσεις Συστήματος Unity Editor

Ενώ οι παραπάνω είναι οι ελάχιστες απαιτήσεις για να μπορεί κανείς να χρησιμοποιήσει τον Editor, δεν σημαίνει ότι και το παιχνίδι που θα δημιουργήσει θα απαιτεί τις ίδιες. Ανάλογα με την πλατφόρμα στην

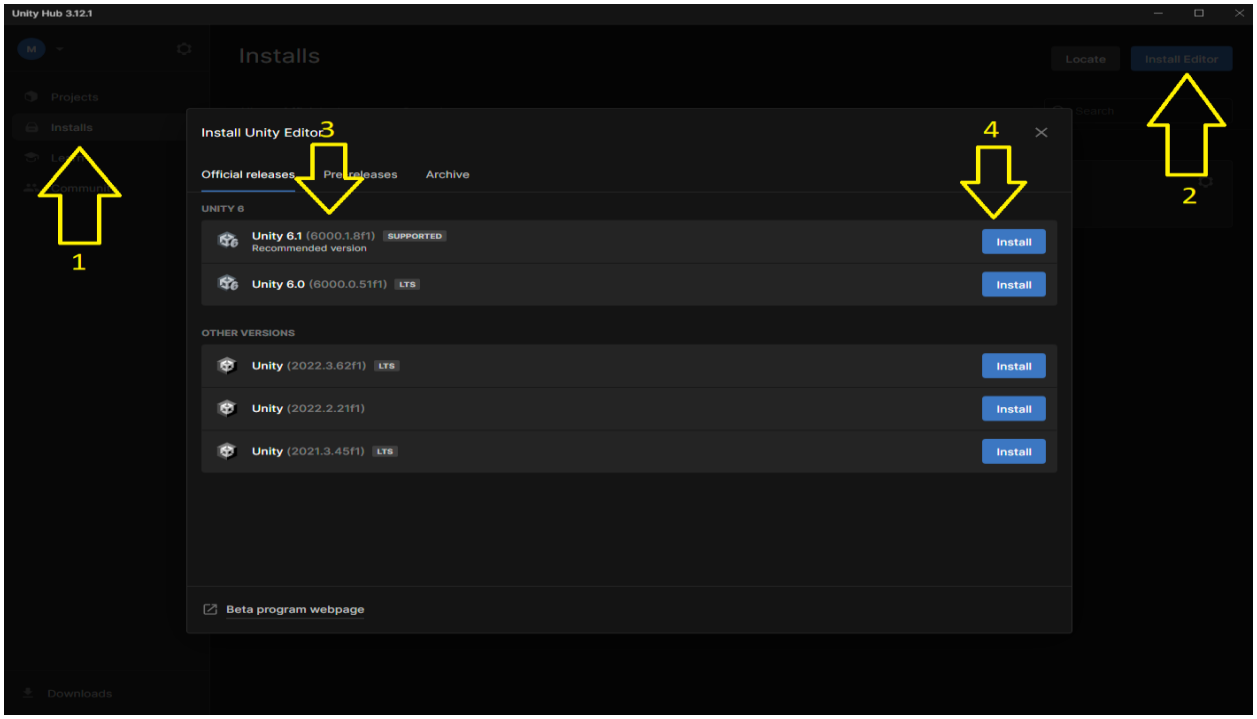
οποία θέλει να τρέξει το παιχνίδι, το μέγεθος, την ποιότητα και την πολυπλοκότητα του παιχνιδιού, υπάρχει πιθανότητα να χρειαστούν περισσότεροι πόροι ή και λιγότεροι.

4.3 Εγκατάσταση και Περιβάλλον

4.3.1 Εγκατάσταση

Για την εγκατάσταση της Unity, θα χρειαστεί να κατεβάσουμε από το διαδίκτυο το Unity Hub. Με το τέλος της εγκατάστασης του Unity Hub και αφού κάνουμε σύνδεση στον λογαριασμό μας, πρέπει να κατεβάσουμε και να εγκαταστήσουμε έναν editor, μέσα από τον οποίο θα δουλέψουμε. Για να το κάνουμε αυτό, από το μενού στα αριστερά θα πατήσουμε το "Εγκαταστάσεις (Installs)" και στο πάνω δεξιά σημείο του παραθύρου θα δούμε ένα κουμπί "Εγκατάσταση Περιβάλλοντος (Install Editor)". Πατώντας το ένα αναδυόμενο παράθυρο (Pop-Up) θα εμφανιστεί με όλες της διαθέσιμες εκδόσεις. Οι εκδόσεις εκτός από τον αριθμό έκδοσής τους έχουν και μια περιγραφή, "Supported", που σημαίνει ότι λαμβάνει επί του παρόντος ενημερώσεις, και "LTS (Long-Term Support)", για να δείξουν ότι μια έκδοση είναι πιο σταθερή αλλά δεν λαμβάνει νέες ενημερώσεις.

Αφού επιλέξουμε την έκδοση που θέλουμε να χρησιμοποιήσουμε, το περιεχόμενο του pop-up θα αλλάξει. Πλέον θα εμφανίζονται κάποιες ρυθμίσεις. Επιλέγουμε αυτές που χρειαζόμαστε και θα ξεκινήσει να γίνεται η εγκατάσταση του Editor.



4.2) Βήματα Εγκατάστασης Μηχανής

4.3.2 Ρυθμίσεις Μηχανής

Σε αντίθεση με την Unreal, οι ρυθμίσεις για την προσθήκη διάφορων πακέτων στην μηχανή γίνεται κατά την εγκατάσταση. Αφού επιλέξουμε την έκδοση της μηχανής που θέλουμε να εγκαταστήσουμε, το νέο

περιεχόμενο του pop-up, θα περιέχει τα πακέτα με τα οποία που μπορούμε να εμπλουτίσουμε την μηχανή. Αυτά περιέχουν την υποστήριξη κάποιας πλατφόρμας ή λειτουργικού, όπως για παράδειγμα iOS, Mac, Web και visionOS, την αλλαγή γλώσσας του Editor, όπως κινεζικά, ιαπωνικά, γερμανικά και ισπανικά, και το Documentation που παρέχει στον προγραμματιστή πλήρη τεκμηρίωση για τις κλάσεις, τις συναρτήσεις και τα συστήματα μηχανής.

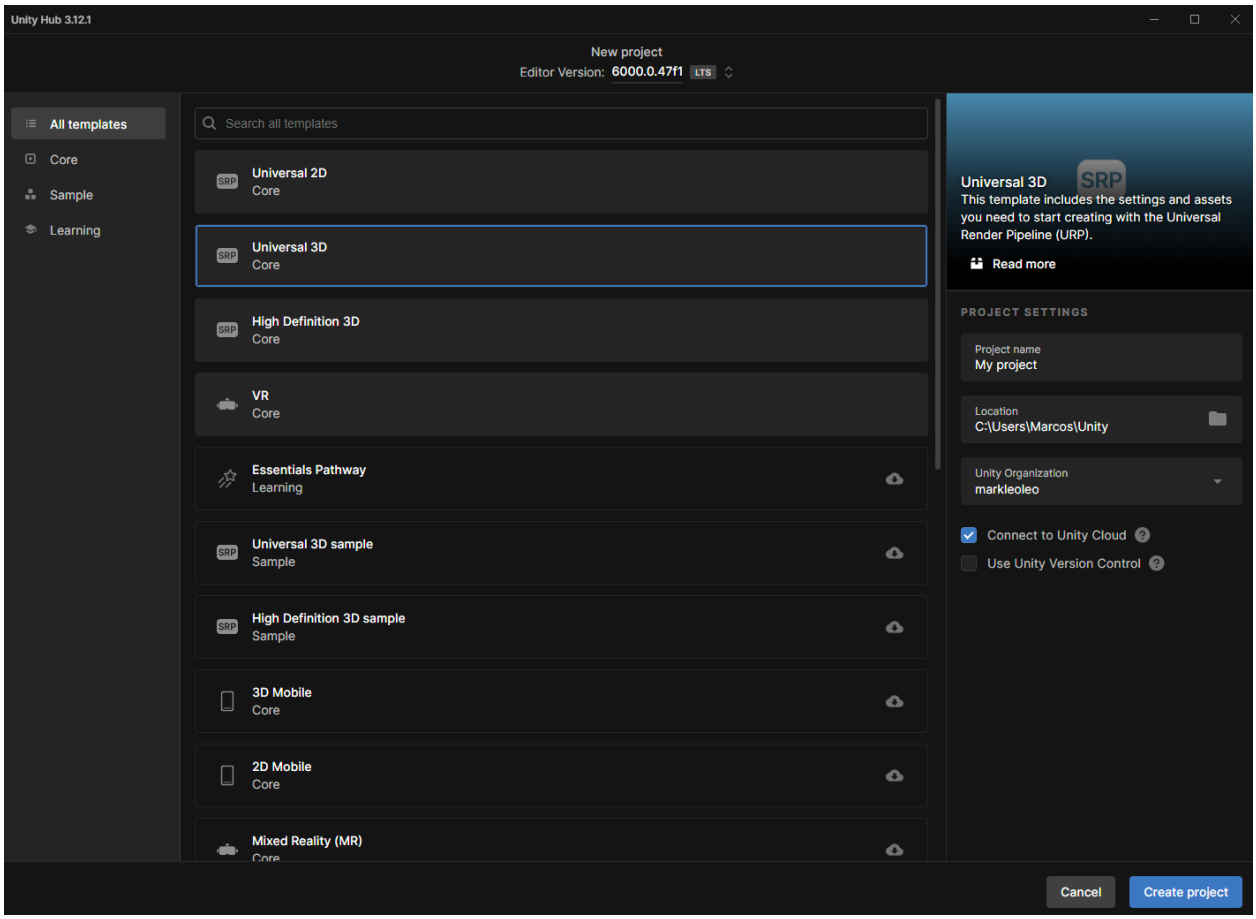
Αν στην πορεία της δημιουργίας του έργου, καταλάβουμε πως χρειαζόμαστε κάποιο από τα παραπάνω πακέτα, μπορούμε πολύ εύκολα να το προσθέσουμε. Υπάρχουν 2 τρόποι.

Ο πρώτος τρόπος είναι από το ίδιο tab "Installs". Εκεί θα εμφανίζονται όλες οι εγκατεστημένες εκδόσεις Unity. Βρίσκουμε αυτή που χρησιμοποιούμε και πατώντας το γρανάζι στο πάνω δεξιά μέρος της έκδοσης, θα δούμε ένα dropdown. Μέσα από αυτό μπορούμε να προσθέσουμε πακέτα με την επιλογή "Add Modules", να ανοίξουμε τον φάκελο με τα αρχεία της εγκατεστημένης μηχανής στον υπολογιστή μας ("Show in Explorer") ή να απεγκαταστήσουμε την μηχανή ("Uninstall").

Ο δεύτερος τρόπος είναι από το tab "Projects" στο μενού στα αριστερά. Στο τέλος της γραμμής του έργου μας, θα δούμε την έκδοση με την οποία είναι χτισμένο. Πατώντας τον αριθμό, ένα pop-up θα εμφανιστεί. Σε αυτό το pop-up θα εμφανίζονται όλες οι εκδόσεις Unity που έχουμε εγκατεστημένες. Μέσα από αυτό το pop-up μπορούμε να αλλάξουμε την έκδοση με την οποία θα ανοίγει το έργο ή πατώντας το κουμπί "Add platform" στο τέλος της έκδοσης, να δούμε το pop-up με όλα τα διαθέσιμα πακέτα.

4.3.3 Δημιουργία Έργου

Αφού εγκαταστήσουμε τον Editor, πρέπει να δημιουργήσουμε το έργο. Για να το κάνουμε αυτό, πρέπει να πάμε στο tab "Projects". Στο πάνω δεξιά μέρος του παραθύρου του Unity Hub, θα υπάρχει ένα κουμπί Νέο Έργο (New Project) και δίπλα του ένα κουμπί "Add". Με το κουμπί "Add" μπορούμε να προσθέσουμε ένα ήδη υπάρχων έργο είτε αυτό υπάρχει στον δίσκο μας ("Add from disk") είτε αυτό υπάρχει αποθηκευμένο σε κάποια repository, όπως για παράδειγμα στο GitHub. Στην περίπτωση που θέλουμε να δημιουργήσουμε ένα καινούργιο έργο, η διαδικασία είναι παρόμοια με την Unreal. Με το πάτημα του κουμπιού "New Project", το περιεχόμενο του παραθύρου θα αλλάξει. Πλέον θα υπάρχουν διάφορα templates.



4.3) Παράθυρο Δημιουργίας Project Unity

Στα αριστερά θα δούμε τις τέσσερις κατηγορίες φιλτραρίσματος template που υπάρχουν.

- Η “All templates” περιέχει όλα τα template. Με την επιλογή εμφανίζονται όλα τα διαθέσιμα templates.
- Η “Core” περιέχει templates τα οποία είναι άδεια ή με πολύ απλές λειτουργίες. Χρησιμοποιούνται για την δημιουργία παιχνιδιών από το μηδέν, αφού δεν έχουν πολλά προρυθμισμένα στοιχεία.
- Η “Sample” περιέχει αυτά με προκαθορισμένες λειτουργίες για την βοήθεια του προγραμματιστή να δημιουργήσει το παιχνίδι που επιθυμεί\ Περιλαμβάνουν έτοιμους μηχανισμούς και λειτουργίες.
- Η “Learning” περιέχει έτοιμα παραδείγματα, τα οποία καθοδηγούν τον προγραμματιστή βήμα-βήμα για την εκμάθηση της μηχανής.

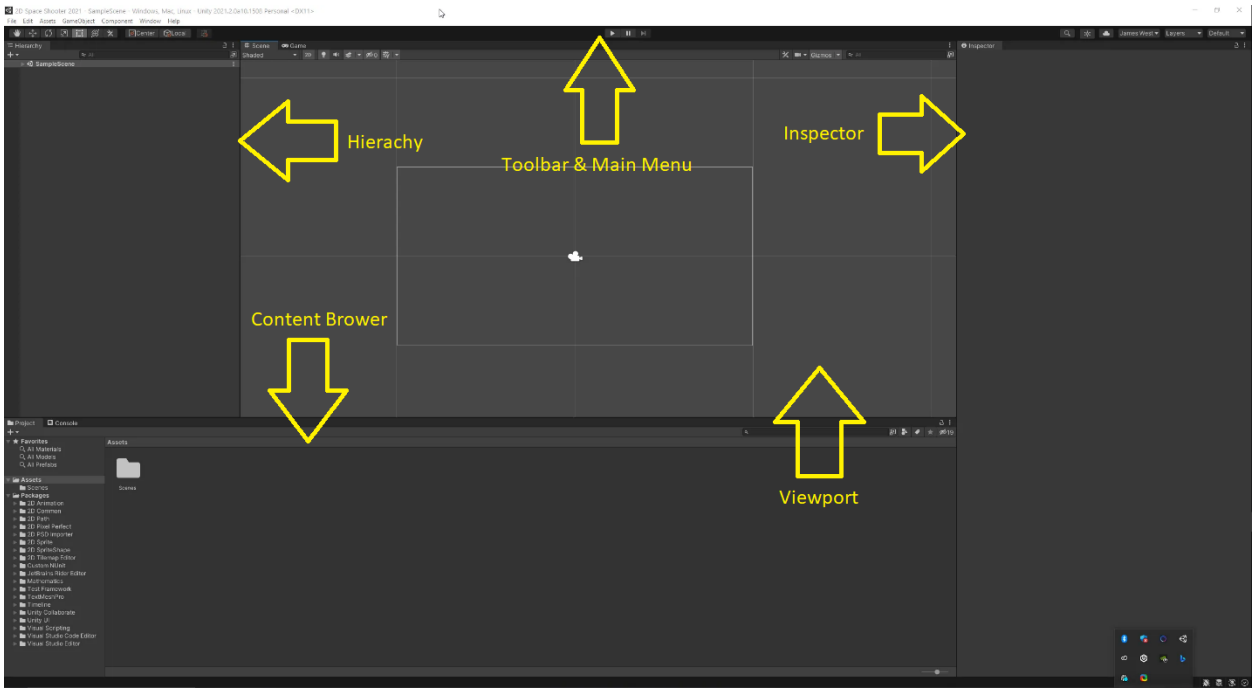
Στο κέντρο θα δούμε τα διαθέσιμα templates, σύμφωνα με την κατηγορία που είχαμε επιλέξει. Υπάρχουν templates για VR, για παιχνίδια κινητού, παιχνίδια πολλαπλών παικτών (multiplayer), παιχνίδια υψηλής ποιότητας (High Definition 3D) για επεξεργασία και δημιουργία ταινιών, σειρών και animated περιεχομένου.

Στα δεξιά θα δούμε στο πάνω μέρος μια μικρή επεξήγηση του περιεχομένου και των δυνατοτήτων του επιλεγμένου template, για την πιο εύκολη επιλογή του καταλληλότερου. Πιο κάτω είναι οι βασικές ρυθμίσεις του έργου, όπως η επιλογή ονόματος και η τοποθεσία αποθήκευσης του έργου.

Αρκετά template, χρειάζονται εγκατάσταση. Τα μόνα που είναι προ εγκατεστημένα είναι το VR, το Universal 2D, το Universal 3D και το High Definition 3D. Είναι “Core” templates για την δημιουργία έργου εικονικής πραγματικότητας, έργου 2D, ελαφριού έργου 3D και έργα προσομοίωσης, αρχιτεκτονικής ή παιχνίδια ρεαλιστικών γραφικών αντίστοιχα.

4.3.4 Περιβάλλον – Editor

Με την επιλογή του σωστού template θα ξεκινήσει η μεταγλώττισή των αρχείων που περιέχονται στο δημιουργημένο έργο. Η διαδικασία την πρώτη φορά, θα πάρει λίγη ώρα. Έπειτα θα ανοίξει ο editor στον οποίο θα δημιουργήσουμε το παιχνίδι.



4.4) Δομή Editor Unity

Στο κάτω μέρος του editor βλέπουμε το παράθυρο του Project (αντίστοιχο Content Browser της Unreal). Μέσα σε αυτό εμφανίζονται όλα τα αρχεία και οι φάκελοι που περιέχει το έργο. Βασικοί φάκελοι όπως Scripts, Materials, Prefabs και Maps, θα υπάρχουν ήδη δημιουργημένοι, ενώ μπορούν να αναδιοργανωθούν για ευκολία του προγραμματιστή. Η δομή των φακέλων είναι ιεραρχική, και στην κορυφή αυτής της ιεραρχίας υπάρχουν 2 βασικοί φάκελοι, ο “Assets” που περιέχει όλα τα αρχεία για την δημιουργία του παιχνιδιού, και ο “Packages” που περιέχει όλα τα αρχεία που χρειάζονται τα επεκτάσιμα και οι λειτουργίες για να δουλέψουν σωστά. Συνήθως τον φάκελο με “Packages” δεν τον αλλάζουμε. Μαζί με το “Project” υπάρχει και το “Console” στο οποίο θα εμφανίζονται λάθη και προειδοποιήσεις κατά την εκτέλεση του παιχνιδιού ή την εκτέλεση του κώδικα.

Στα αριστερά υπάρχει το παράθυρο ιεραρχίας (Hierarchy). Εκεί θα μπορούμε να δούμε όλα τα GameObjects που έχουμε τοποθετήσει στην σκηνή μας. Μπορούμε να δούμε την δομή της σκηνής και να προσθέσουμε καινούργια αντικείμενα. Στην περίπτωση που εισάγουμε ένα αντικείμενο κατευθείαν στην σκηνή, το

αντικείμενο αυτό θα εμφανιστεί αυτόματα και σε αυτό το παράθυρο. Μια καλή πρακτική, είναι να οργανώσουμε τα αντικείμενα που φαίνονται εδώ σε κατηγορίες βάζοντας ένα κενό GameObject (Separator). Για παράδειγμα, δημιουργούμε ένα κενό GameObject με το όνομα "Player" και ένα με το όνομα "World" και από κάτω έχουμε όλα τα αντίστοιχα αντικείμενα που έχουμε τοποθετήσει στην σκηνή μας. Υπάρχει πολύ έτοιμο περιεχόμενο στο κατάστημα για την καλύτερη οπτικοποίηση των "Separator".

Στο κεντρικό παράθυρο υπάρχουν 2 υποπαράθυρα, το "Scene" και το "Game". Το "Scene" λειτουργεί σαν το "Viewport" στην Unreal. Είναι το παράθυρο στο οποίο θα τοποθετούμε, μετακινούμε και επεξεργαστούμε τα αντικείμενα του παιχνιδιού. Στο "Game" φαίνεται το πως θα μοιάζει το παιχνίδι στον τελικό παίκτη όταν πατηθεί το "Play".

Στα δεξιά υπάρχει το "Inspector". Επιλέγοντας ένα GameObject, σε αυτό το παράθυρο θα εμφανίζονται όλες οι ιδιότητές του. Μπορούμε έτσι να προσθέσουμε ή να επεξεργαστούμε αυτές τις ιδιότητες. Για παράδειγμα, μπορούμε να ενσωματώσουμε ένα Script αρχείο, εμπλουτίζοντας το GameObject με λογική. Ή μπορούμε να αλλάξουμε την εμφάνιση του μέσω διαφόρων component. Χρησιμοποιείται για την παραμετροποίηση κάθε αντικειμένου.

Στην κορυφή του παραθύρου υπάρχει το κεντρικό μενού. Από εκεί έχουμε πρόσβαση σε βασικές λειτουργίες, όπως Edit, Assets, GameObject και Window, κάθε μια παρέχοντας βασικές λειτουργίες όπως άνοιγμα των Project Settings, αποθήκευση του project, δημιουργία GameObject και προσθήκη έτοιμου περιεχομένου στο έργο. Κάτω από αυτό υπάρχει ένα toolbar με κουμπιά Play/Pause/Stop, κουμπιά για άνοιγμα του παραθύρου των επεκτάσιμων, του λογαριασμού και του περιεχομένου, ενώ υπάρχουν και κουμπιά για αναζήτηση και διαμόρφωση των παραθύρων του Editor σύμφωνα με τις προτιμήσεις του προγραμματιστή.

4.4 Ρυθμίσεις Έργου – Project Settings

Συχνά μπορεί να παρουσιαστεί η ανάγκη για την αλλαγή βασικών ρυθμίσεων του έργου. Αυτό γίνεται μέσω των Project Settings, τις οποίες μπορούμε να ανοίξουμε από το κεντρικό μενού (Edit – Project Settings), είτε μέσω του toolbar.

Οι ρυθμίσεις αυτές οργανώνονται σε διάφορες ενότητες όπως **Player, Quality, Input Manager, Time, Physics, Tags and Layers, Graphics, Audio, Scripting** και άλλες.

Η κατηγορία **Player** περιλαμβάνει σημαντικές γενικές ρυθμίσεις του έργου όπως το όνομα της εφαρμογής, την εταιρεία, τις πλατφόρμες στόχευσης (π.χ. Android, iOS, Windows), καθώς και παραμέτρους όπως ανάλυση, εικονίδιο, splash screen, δικαιώματα, backend scripting και άλλα.

Η ενότητα **Quality** επιτρέπει την διαμόρφωση του επιπέδου ποιότητας ανά πλατφόρμα, π.χ. πόσα anti-aliasing επίπεδα να χρησιμοποιούνται, αν θα υπάρχουν shadows, LOD ρυθμίσεις κ.ά.

Η **Input Manager** επιτρέπει τη διαχείριση των πλήκτρων και εισόδων από το χρήστη (π.χ. ποντίκι, πληκτρολόγιο, χειριστήρια), ενώ η **Time** καθορίζει τις παραμέτρους του χρόνου παιχνιδιού όπως fixed timestep και time scale.

Στην **Physics** μπορούμε να ορίσουμε παραμέτρους όπως η βαρύτητα, το μέγιστο αριθμό iterations για τις φυσικές συγκρούσεις κ.λπ.

Οι **Tags and Layers** καθορίζουν τη δυνατότητα κατηγοριοποίησης αντικειμένων για φιλτράρισμα και collision detection.

Η κατηγορία **Graphics** ελέγχει την απόδοση και εμφάνιση του γραφικού συστήματος (π.χ. render pipeline settings), ενώ η **Audio** επιτρέπει τη διαχείριση ήχου, spatialization και output ρυθμίσεων.

Τέλος, το **Scripting** περιλαμβάνει ρυθμίσεις που σχετίζονται με την έκδοση του .NET framework, το backend scripting engine (π.χ. IL2CPP ή Mono), και άλλες επιλογές σχετικές με την μεταγλώττιση και εκτέλεση του κώδικα.

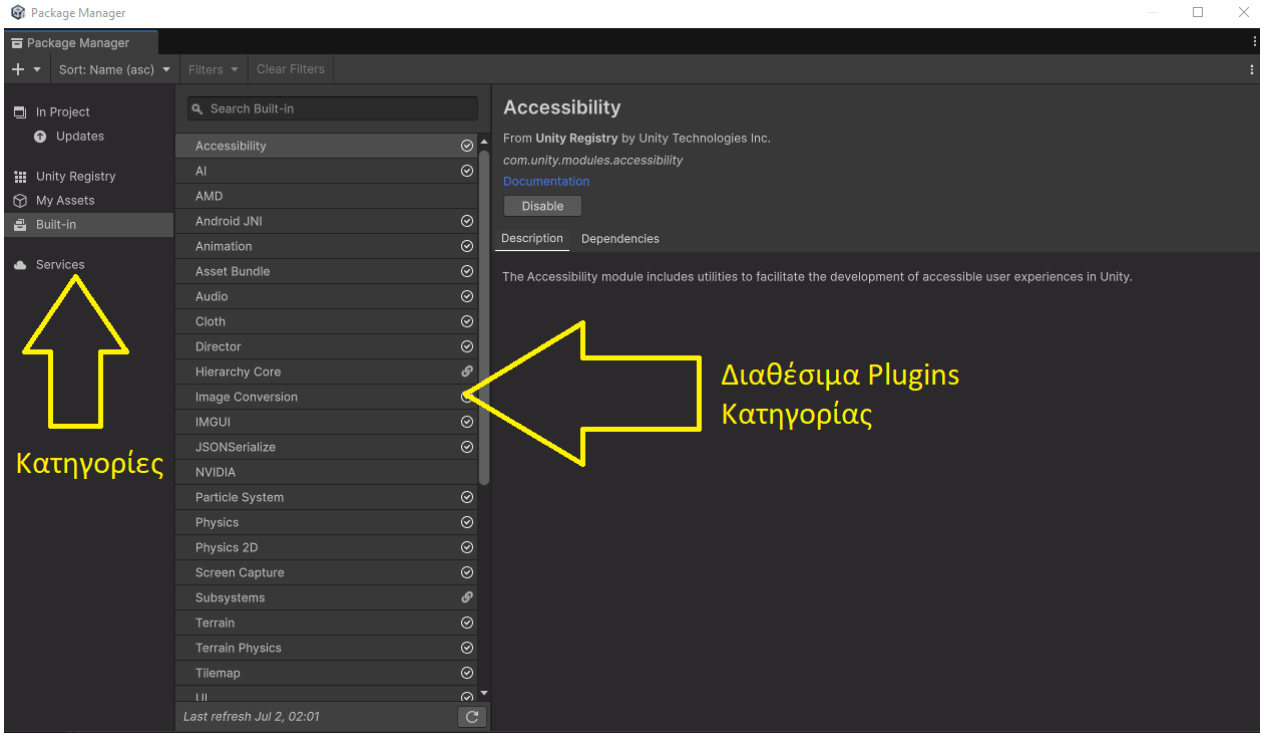
Οι περισσότερες από αυτές τις ρυθμίσεις έχουν προκαθορισμένες τιμές που λειτουργούν στις περισσότερες περιπτώσεις, αλλά είναι σημαντικό να γίνουν κατάλληλες τροποποιήσεις ανάλογα με τις απαιτήσεις του έργου και την πλατφόρμα στόχευσης.

4.5 Επεκτάσιμα – Plugins

Στην Unity, τα plugins (ή assemblies) συνήθως αναφέρονται σε βιβλιοθήκες (.dll αρχεία) ή εγγενείς (native) πόρους σε C++ που ενσωματώνονται στον φάκελο Assets/Plugins του project. Μπορεί να είναι 2 ειδών, Managed (δηλαδή assemblies που εκτελούνται σε .NET runtime της Unity), είτε native (απαιτείται συμβατότητα με την πλατφόρμα που στοχεύει το project)[27-24]. Η προσθήκη τους μπορεί να γίνει με drag and drop ή μέσω του Unity Package Manager. [24]

Μέσω του **Package Manager** (Window → Package Manager), η Unity επιτρέπει αναζήτηση, εγκατάσταση, ενημέρωση ή και αφαίρεση plugins. Εκεί διαχωρίζονται σε κατηγορίες: “Unity Registry” για τα επίσημα πακέτα της Unity Technologies, “In Project” για όσα χρησιμοποιούνται ήδη, και “My Assets” για όσα έχουν αγοραστεί ή αποθηκευτεί από το Asset Store [24]. Επιπλέον, ο χρήστης μπορεί να ενεργοποιήσει πειραματικά ή preview πακέτα μέσω του advanced filter. Για προχωρημένες χρήσεις, τα plugins μπορούν να ρυθμιστούν να είναι ορατά ή ενεργά μόνο για συγκεκριμένες πλατφόρμες ή αρχιτεκτονικές, μέσω των import settings στο Inspector. [25]

Επιπλέον, όταν χρησιμοποιούνται custom plugins, είναι σημαντική η σωστή διαχείριση της **εξάρτησης μεταξύ assemblies** (assembly definitions), ώστε να μειώνεται ο χρόνος compile και να διατηρείται modularity στον κώδικα. [25] Έτσι, τα plugins στην Unity εξυπηρετούν όχι μόνο την επέκταση της λειτουργικότητας ενός project, αλλά και την οργάνωση του κώδικα σε σαφώς οριοθετημένες μονάδες.



4.5) Unity Package Manager

4.6 Ανάλυση Τεχνολογιών και Επεκτάσιμων (Plugins)

Όπως και στο προηγούμενο κεφάλαιο, παρακάτω θα αναλυθούν μερικές βασικές τεχνολογίες που προσφέρει η Unity.

4.6.1 Scripting

Η Unity χρησιμοποιεί τη γλώσσα C# ως βασική πλατφόρμα scripting, επιτρέποντας στους δημιουργούς να προγραμματίζουν interactive συμπεριφορές για αντικείμενα, ελέγχους, animations, UI, φυσική και rendering μέσα στο περιβάλλον του game engine. Μέσω της δομής MonoBehaviour, κάθε script μπορεί να συνδεθεί με ένα GameObject, κάτι που υποστηρίζει modular, αντικειμενοστραφή σχεδιασμό με delegates, events και LINQ για υψηλή αναγνωσιμότητα και επεκτασιμότητα. Η C# είναι επιλογή με ολοκληρωμένη τεκμηρίωση από το Unity Manual και ενεργή υποστήριξη κοινότητας μέσω API Reference, tutorials και guides.

4.6.2 Prefabs

Τα Prefabs είναι επαναχρησιμοποιήσιμα πρότυπα GameObjects που περιλαμβάνουν component configuration, scripts, child objects και υλικά. Διευκολύνουν την επανατοποθέτηση και ενημέρωση πολλαπλών instances μέσα σε σκηνές μέσω μιας ενιαίας αλλαγής στην prefab asset. Το system των Prefab Variants επιτρέπει την κατασκευή παραλλαγών ενώ διατηρεί την αρχική δομή, υποστηρίζοντας modular ανάπτυξη και συνεργασία σε ομάδες με λιγότερα conflicts στο version control. [26]

4.6.3 Forward & Deferred Rendering

Η Unity υποστηρίζει δύο βασικά rendering paths: το **Forward Rendering** (μεμονωμένα passes για κάθε φωτιστικό σε κάθε αντικείμενο) και το **Deferred Rendering** (αποθήκευση δεδομένων γεωμετρίας σε G-buffers, με μεταγενέστερους υπολογισμούς φωτισμού για κάθε pixel). Το Forward είναι κατάλληλο για εφαρμογές με λίγα φώτα ή ανάγκη για transparency, ενώ το Deferred προσφέρει αποδοτικότερη διαχείριση πολλαπλών φωτιστικών πηγών και advanced shading. [27]

4.6.4 Render Pipelines

Η Unity προσφέρει 2 διαφορετικές ροές επεξεργασίας (pipelines). Την legacy Built-In pipeline, Universal Render Pipeline (URP) για βελτιστοποίηση πολλαπλών πλατφορμών, και την High Definition Render Pipeline (HDRP) για high-end φωτορεαλιστικά γραφικά με volumetric lighting, PBR, screen-space effects και advanced post-processing. Ο χρήστης μπορεί να επιλέξει την pipeline που ταιριάζει στο target hardware και την οπτική ποιότητα της εφαρμογής, ενώ η χρήση Scriptable Render Pipelines (SRP) επιτρέπει custom ρυθμίσεις και workflows. [28]

4.6.5 Data Oriented Technology Stack – DOTS

Η αρχιτεκτονική DOTS (Data-Oriented Technology Stack) της Unity αποτελεί ένα νέο μοντέλο ανάπτυξης λογισμικού που εισάγει έναν ριζικά διαφορετικό τρόπο προσέγγισης του σχεδιασμού και της εκτέλεσης game logic. [29] Ο πυρήνας του DOTS βασίζεται στο Entity Component System (ECS), το οποίο αντικαθιστά την παραδοσιακή αντικειμενοστραφή (object-oriented) προσέγγιση με μια δομή βασισμένη σε δεδομένα (data-oriented design). Αντί κάθε αντικείμενο (π.χ. χαρακτήρας, αντικείμενο στο περιβάλλον) να αναπαρίσταται ως μια πολύπλοκη ιεραρχία κλάσεων με ιδιότητες και μεθόδους, το ECS χωρίζει τα δεδομένα (components) από τη συμπεριφορά (systems), επιτρέποντας εξαιρετικά βελτιστοποιημένη πρόσβαση στη μνήμη. [30]

Συμπληρωματικά, το C# Job System επιτρέπει την αξιοποίηση πολυπύρηνων επεξεργαστών μέσω ασύγχρονων, παράλληλων διεργασιών. Αντί οι υπολογισμοί να εκτελούνται σειριακά μέσα στο main thread, μπορούν να κατανέμονται σε πολλαπλά threads με ασφάλεια και συγχρονισμό, μειώνοντας έτσι το computational load και τα frame drops. [31] Τέλος, ο Burst Compiler αποτελεί έναν compiler χαμηλού επιπέδου που μεταφράζει τον κώδικα των jobs σε βελτιστοποιημένο μηχανικό κώδικα (assembly) για εξαιρετικά υψηλές επιδόσεις, συγκρίσιμες με native C++ εκτέλεση. [32]

Ο συνδυασμός αυτών των τριών τεχνολογιών καθιστά το DOTS ιδανικό για real-time προσομοιώσεις, φυσικές αλληλεπιδράσεις, AI swarms, καθώς και για παιχνίδια με μεγάλο αριθμό entities, όπως MMOs ή RTS.

4.6.6 Unity ML-Agents

Το ML-Agents Toolkit (Machine Learning Agents Toolkit) της Unity είναι ένα ανοικτού κώδικα framework που γεφυρώνει τον κόσμο της τεχνητής νοημοσύνης με την ανάπτυξη διαδραστικών 3D περιβαλλόντων. Η βασική του λειτουργία είναι να μετατρέπει τις Unity σκηνές σε προσαρμόσιμα περιβάλλοντα εκπαίδευσης για agents οι οποίοι μπορούν να μάθουν μέσω ενισχυτικής μάθησης (reinforcement learning), μιμητικής μάθησης (imitation learning), καθώς και άλλων μορφών machine learning. Οι agents αυτοί δρουν μέσα στον εικονικό χώρο αλληλεπιδρώντας με το περιβάλλον τους, λαμβάνοντας παρατηρήσεις (observations), εκτελώντας ενέργειες και συλλέγοντας ανταμοιβές (rewards), στο πλαίσιο ενός διαρκούς κύκλου βελτιστοποίησης. [33]

Η σύνδεση του ML-Agents με Python μέσω API επιτρέπει τη χρήση ισχυρών βιβλιοθηκών όπως PyTorch ή TensorFlow για την εκπαίδευση νευρωνικών δικτύων που καθορίζουν τη συμπεριφορά των agents. Παράλληλα, προσφέρει υποστήριξη για training scripts, imitation trainers, curriculum learning (βαθμιαία εκπαίδευση) και multi-agent περιβάλλοντα. Ένα σημαντικό πλεονέκτημα είναι ότι όλη η εκπαιδευτική διαδικασία μπορεί να πραγματοποιηθεί εντός Unity, με δυνατότητα visual debugging και real-time αξιολόγησης της μάθησης.

Το ML-Agents είναι ιδανικό για ανάπτυξη έξυπνων NPCs (non-player characters) που προσαρμόζονται στη συμπεριφορά του παίκτη, έρευνα σε τομείς όπως η ρομποτική ή η οικονομική προσομοίωση, και εκπαιδευτικά simulations όπου η προσαρμοστικότητα είναι κρίσιμη. Η κοινότητα γύρω από το εργαλείο είναι ενεργή, με πλούσιο περιεχόμενο στο GitHub και επίσημη τεκμηρίωση, γεγονός που διευκολύνει την ενσωμάτωσή του σε ποικίλα έργα. [34]

4.6.7 Cinemachine & Timeline

Το Cinemachine είναι ένα προηγμένο σύστημα ελέγχου κάμερας που παρέχεται ως πακέτο από τη Unity και έχει σχεδιαστεί για να βελτιστοποιεί την κάλυψη σκηνών μέσω δυναμικού και αυτοματοποιημένου framing, παρακολούθησης (follow) και μετάβασης (transitioning). Χωρίς την ανάγκη εκτενούς scripting, προσφέρει λειτουργίες όπως auto-dolly tracking, procedural camera behaviors, composer-based framing και noise simulation, επιτρέποντας στον δημιουργό να επικεντρωθεί περισσότερο στη σκηνοθεσία και λιγότερο στον τεχνικό έλεγχο. [35] Επιπλέον, η λειτουργικότητα blending μεταξύ καμερών παρέχει φυσικές, ομαλές μεταβάσεις, κάτι που είναι ιδανικό για δράση σε πραγματικό χρόνο ή cutscenes.

Το Timeline, από την άλλη πλευρά, είναι ένα non-linear sequence editor που επιτρέπει την ενορχήστρωση animation, ήχου, ενεργειών και εναλλαγών σε μια ενιαία χρονολογική γραμμή. Μέσω του Timeline, οι δημιουργοί μπορούν να συντονίζουν διάλογο, κινήσεις χαρακτήρων, χειρονομίες και εναλλαγές κάμερας, χωρίς να γράψουν ούτε μια γραμμή κώδικα. Παρέχει multi-track control για animation clips, audio sources και signals, δίνοντας απόλυτο έλεγχο στο συγχρονισμό στοιχείων. [36]

Όταν αυτά τα δύο εργαλεία συνδυαστούν, δημιουργείται ένα ισχυρό σύστημα για κινηματογραφική αφήγηση και αλληλεπιδραστικά σκηνικά. Μέσω της χρήσης Virtual Cameras (VCams) και sequencing με το Timeline, μπορούν να δημιουργηθούν δυναμικές σκηνές με cutscenes υψηλής ποιότητας, που παλαιότερα απαιτούσαν custom scripts και animations. Ο συγχρονισμός μεταξύ ήχου, κινούμενων αντικειμένων και λήψεων κάμερας επιτυγχάνεται εξ ολοκλήρου μέσω του editor, διευκολύνοντας την εργασία των δημιουργικών ομάδων και ενισχύοντας τον κινηματογραφικό χαρακτήρα της παραγωγής.

Και οι 2 τεχνολογίες μπορούν να εγκατασταθούν από το παράθυρο επεκτάσιμων, Unity Package Manager.

4.6.8 Incremental Garbage Collection – IGC

Η Incremental Garbage Collection (IGC) της Unity αποτελεί μια σημαντική τεχνολογική προσθήκη στον τομέα της βελτιστοποίησης απόδοσης για real-time εφαρμογές, ειδικά σε παιχνίδια όπου η συνεχής, ομαλή ροή καρέ είναι κρίσιμη. Σε αντίθεση με την παραδοσιακή stop-the-world garbage collection, η οποία μπορεί να οδηγήσει σε μεγάλης διάρκειας παύσεις (spikes) κατά την εκκαθάριση της μη χρησιμοποιούμενης μνήμης, η IGC διαιρεί τη διαδικασία σε μικρότερες, κατανεμημένες φάσεις, οι οποίες εκτελούνται σε διαδοχικά frames. Αυτή η στρατηγική επιτρέπει τη μείωση των απρόβλεπτων καθυστερήσεων (GC spikes), διατηρώντας ένα πιο σταθερό frame rate, κάτι που είναι ιδιαίτερα σημαντικό για εμπειρίες σε VR, mobile gaming και προσομοιώσεις υψηλής ακρίβειας. [37]

Η IGC ενεργοποιείται από προεπιλογή από την έκδοση Unity 2023 και έπειτα, και η συμπεριφορά της μπορεί να ρυθμιστεί μέσω του μενού Player Settings, κάτω από το Memory Management tab (Unity Docs, *Player Settings*). [37] Εκεί, οι developers μπορούν να επιλέξουν μεταξύ incremental και non-incremental modes, ανάλογα με τις ανάγκες του έργου. Αν και η IGC ενδέχεται να παρατείνει συνολικά τη διάρκεια της συλλογής απορριμμάτων, το γεγονός ότι η επιβάρυνση κατανέμεται ομαλά μέσα στο χρόνο αποτελεί καθοριστικό πλεονέκτημα σε περιβάλλοντα όπου η responsiveness και η σταθερότητα είναι προτεραιότητα. [38] [39]

Επιπλέον, η χρήση της IGC ενδείκνυται σε σενάρια με συχνές δυναμικές δεσμεύσεις μνήμης, όπως π.χ. η runtime δημιουργία αντικειμένων, εναλλαγή σκηνών, ή γρήγορη χρήση scripts που διαχειρίζονται μεγάλο όγκο δεδομένων. Η τεκμηρίωση της Unity προσφέρει επιπλέον οδηγίες για βελτιστοποίηση των αντικειμένων ώστε να ελαχιστοποιείται η επιβάρυνση στον συλλέκτη, ενισχύοντας έτσι τη μακροπρόθεσμη αποδοτικότητα του συστήματος μνήμης. [37]

Ωστόσο, η χρήση της IGC εισάγει overhead από write barriers, καθώς όταν μεταβάλλονται αναφορές μεταξύ objects κατά τη διάρκεια της συλλογής, ο GC πρέπει να επανεξετάσει αυτά τα αντικείμενα. Σε περιπτώσεις όπου η marking phase δεν ολοκληρώνεται λόγω συχνών μεταβολών, ο GC μπορεί να επανέλθει σε non-incremental λειτουργία για να αποφευχθεί deadlock. [37]

4.6.9 Adaptive Probe Volumes – APV

Το σύστημα Adaptive Probe Volumes (APV) στη Unity αποτελεί ένα καινοτόμο εργαλείο για global illumination το οποίο αντικαθιστά τα παλαιά Light Probe Groups, προσφέροντας αυτοματοποιημένη τοποθέτηση light probes βάσει της πυκνότητας της γεωμετρίας σε μια σκηνή. Αυτό επιτρέπει την δημιουργία ενός 3D adaptive grid, όπου κάθε "brick" περιέχει 64 light probes σε διάταξη 4×4×4, με μεταβαλλόμενη πυκνότητα ανάλογα με την πολυπλοκότητα της περιοχής (URP). [40] Το αποτέλεσμα είναι per-pixel sampling φωτισμού αντί του προηγούμενου per-object μοντέλου, οδηγώντας σε πιο ομοιογενή φωτισμό και μείωση seam artifacts. [40]

Τα APV λειτουργούν αποκλειστικά εντός των Scriptable Render Pipelines (URP και HDRP), και προσφέρουν υποστήριξη για real-time streaming των δεδομένων φωτισμού, επιτρέποντας την αποτελεσματική διαχείριση μεγάλων, ανοικτών κόσμων χωρίς στατικό baking lightmaps. [41] Καθώς το

σύστημα προσαρμόζει δυναμικά την πυκνότητα των probes, επιτυγχάνεται ομαλότερη μετάβαση φωτισμού μεταξύ περιοχών και βελτιωμένη ποιότητα εικόνας χωρίς την ανάγκη χειροκίνητης διαμόρφωσης ή πρόσθετων reflection probes. [42]

Παρά τα πλεονεκτήματά τους, τα APV έχουν και ορισμένες περιοριστικές συνθήκες: δεν επιτρέπεται η χειροκίνητη τοποθέτηση των probes μέσα στον όγκο και η κάθε αλλαγή στην απόσταση ή την πυκνότητα απαιτεί νέα bake, κάτι που μπορεί να προκαλέσει errors αν δεν γίνει σωστή επαναφορά του editor. [43] Επιπλέον, σε εφαρμογές που χρησιμοποιούν SSGI ή SSAO σε συνδυασμό με APV σε HDRP, μπορεί να επηρεαστεί το framerate, ιδιαίτερα σε συστήματα με χαμηλότερο performance

4.7 Μειονεκτήματα Τεχνολογιών

Παρόλο που οι τεχνολογίες και τα συστήματα που προσφέρει η Unity –όπως τα DOTS, ML-Agents, Adaptive Probe Volumes, Cinemachine και Timeline– αποτελούν ισχυρά εργαλεία για την ανάπτυξη σύγχρονων και αποδοτικών παιχνιδιών, συνοδεύονται από σημαντικά μειονεκτήματα και προκλήσεις, ιδίως όσον αφορά τη μαθησιακή καμπύλη, τη διαχείριση πόρων και τη γενική πολυπλοκότητα. Ένα βασικό πρόβλημα είναι ότι πολλές από αυτές τις τεχνολογίες έχουν ακόμη ημιτελή ή πειραματική υλοποίηση, με συχνές αλλαγές μεταξύ εκδόσεων και περιορισμένη τεκμηρίωση, κάτι που καθιστά δύσκολη την εκμάθησή τους, ακόμη και για έμπειρους προγραμματιστές. Το DOTS, για παράδειγμα, απαιτεί εξοικείωση με το ECS paradigm που διαφέρει θεμελιωδώς από το αντικειμενοστραφές μοντέλο – αυτό συνεπάγεται πλήρη αλλαγή φιλοσοφίας στον σχεδιασμό του κώδικα, γεγονός που μπορεί να αποβεί αποτρεπτικό σε μικρές ή μεσαίες ομάδες.

Επιπλέον, αρκετά από τα συστήματα της Unity καταναλώνουν σημαντικούς υπολογιστικούς πόρους. Οι Adaptive Probe Volumes, αν και ευέλικτοι για dynamic lighting, αυξάνουν τη χρήση μνήμης και απαιτούν επεξεργαστική ισχύ για real-time streaming, ειδικά σε μεγάλες σκηνές. Ομοίως, το ML-Agents Toolkit απαιτεί εξωτερικά εργαλεία και Python περιβάλλοντα, ενώ η εκπαίδευση agents μπορεί να είναι χρονικά και υπολογιστικά κοστοβόρα. Παράλληλα, ακόμη και φαινομενικά απλά εργαλεία όπως το Timeline και το Cinemachine μπορεί να προκαλέσουν επιβάρυνση στο editor, ειδικά σε έργα με πολύπλοκα cutscenes και layers. Τέλος, το γεγονός ότι πολλές από αυτές τις τεχνολογίες είναι tightly coupled με συγκεκριμένες render pipelines (URP/HDRP) περιορίζει τη συμβατότητα και απαιτεί επιπλέον ρυθμίσεις και debugging σε multi-platform περιβάλλοντα.

4.8 Υποστήριξη και Κοινότητα

Σε αντίθεση με την Unreal, κατά την διάρκεια της υλοποίησης του project μου δεν χρειάστηκε να ανατρέξω τόσες φορές για βοήθεια. Τις περισσότερες φορές, το Documentation και τα Tutorial βίντεο που παρέχονται στην επίσημη σελίδα της Unity, ήταν επαρκή για να με βοηθήσουν να ξεπεράσω το πρόβλημα που είχα.

Και ξανά επόμενο βήμα μου ήταν να πάω στο Forum της Unity και να ψάξω ερωτήσεις χρηστών με παρόμοιο ή και το ίδιο ακριβώς πρόβλημα. Δυστυχώς για τα δικά μου προβλήματα, δεν βρήκα κάτι οπότε χρειάστηκε να κάνω ο ίδιος την ερώτηση. Σε καμία δυστυχώς δεν έλαβα απάντηση, αν και ήμουν σίγουρος εδώ ότι την έθετα στην σωστή ενότητα.

Τέλος, προσπάθησα και εδώ με κάποιο πιο ανεπίσημο τρόπο, όπως μέσω Discord. Ευτυχώς, εδώ υπήρχαν άτομα έμπειρα και μη. Με βοήθησαν ή με κατεύθυναν προς την σωστή πορεία, καταφέρνοντας έτσι να λύσω τα προβλήματα μου και να δημιουργήσω το project μου.

4.9 Κατάστημα – Marketplace

Για την απόκτηση έτοιμου περιεχομένου, η Unity παρέχει στους χρήστες το Asset Store. Το Asset Store είναι ένας ψηφιακός χώρος στον οποίο μπορείς να περιηγηθείς είτε μέσα από το Unity Editor είτε μέσω του διαδικτύου, ώστε να αναζητήσεις και να αποκτήσεις πακέτα περιεχομένου που καλύπτουν τις ανάγκες του project σου. Εκεί μπορείς να δεις αναλυτικές πληροφορίες, εικόνες ή και βίντεο, προκειμένου να αξιολογήσεις αν σε ενδιαφέρει κάποιο asset.

Πολλά από τα διαθέσιμα assets είναι επί πληρωμή, ωστόσο υπάρχει και δωρεάν περιεχόμενο, και η ποιότητά τους ποικίλει, με αρκετά να προσφέρουν επαγγελματικό αποτέλεσμα. Για να χρησιμοποιήσεις ένα asset, πρέπει να είσαι συνδεδεμένος με τον λογαριασμό Unity και να το κατεβάσεις μέσω του Unity Editor, ώστε να εισαχθεί απευθείας στο project σου.

Κεφάλαιο 5ο Υλοποίηση Έργου σε Unreal Engine

Πριν από την έναρξη δημιουργίας του κάθε έργου, έγινε μια εκτενής έρευνα στο marketplace για τα asset που θα χρησιμοποιηθούν, προσπαθώντας να είναι δωρεάν.

Αυτό που παρατήρησα ήταν ότι το Fab (Unreal Marketplace) διέθετε πολύ λίγα σε αριθμό asset απ' ότι αυτό. Επίσης, δεν διέθετε μεγάλο αριθμό δωρεάν asset ή τα δωρεάν asset ήταν δυσλειτουργικά. Μετά από αρκετό ψάξιμο, κατέληξα στα asset και μπορούσε πλέον να ξεκινήσει η δημιουργία του project. Τα asset αναφέρονται στο πρώτο υποκεφάλαιο.

Το project στην Unreal Engine ξεκίνησε σε έκδοση 5.3 και στην πορεία αναβαθμίστηκε στην 5.5.4. Μετά το τέλος ανάπτυξης και κατά την περίοδο συγγραφής του κειμένου, κυκλοφόρησε και η 5.6 στην οποία δεν έγινε αναβάθμιση.

5.1 Assets

Ο παρακάτω πίνακας παρουσιάζει τα assets που χρησιμοποιήθηκαν στην ανάπτυξη. Τα περισσότερα assets αποκτήθηκαν δωρεάν, είτε μέσω marketplace είτε μέσω του template που χρησιμοποιήθηκε, ενώ για την καλύτερη κατανόηση του τρόπου λειτουργίας και δημιουργίας, αγοράστηκε και ένα πακέτο περιεχομένου.

Όνομα	Πηγή	Περιγραφή	Τιμή Fab	Τιμή Απόκτησης
Modular Urban houses	Fab	Κτίρια	Δωρεάν	Δωρεάν
Traffic AI System	Fab	Δρόμος, AI	40€	40€
Crossroad Generator	Fab	Δρόμος	50€	Δωρεάν
Vehicle	UE Template	Αυτοκίνητο παίκτη	Δωρεάν	Δωρεάν
Temperate Vegetation: Spruce Forest	Fab	Φύση	Δωρεάν	Δωρεάν

1) Τα assets που χρησιμοποιήθηκαν στην Unreal.

5.2 Οργάνωση Project

Στην εικόνα 5.1, βλέπουμε την οργάνωση των αντικειμένων που είναι τοποθετημένα στην σκηνή μας. Δημιουργήθηκαν οι κατάλληλοι φάκελοι και υποφάκελοι για την καλύτερη διαχώριση των στοιχείων. Οι βασικοί φάκελοι είναι οι AI, Landscape, Lighting και Player, μέσα στους οποίους περιέχονται όλα το περιεχόμενο, χωρισμένο σε επιμέρους φακέλους, όπου αυτό κρίθηκε απαραίτητο. Ο φάκελος AI περιέχει όλα τα Splines που είναι υπεύθυνα για την κίνηση και την δημιουργία των NPC αυτοκινήτων. Ο Landscape περιέχει το αρχικό έδαφος (Landscape), τα PCG Blueprints και τα meshes που δημιουργήθηκαν με το PCG, τα Blueprints για τους δρόμους καθώς και το Spline για τα όρια του χάρτη. Ο Lighting περιέχει όλα τα στοιχεία για το φωτισμό του χάρτη, και ο Player το Blueprint του αυτοκινήτου του παίκτη. Τα 3 τελευταία στοιχεία είναι στοιχεία δημιουργημένα αυτόματα από την μηχανή και προτείνεται να μην πειραχτούν.

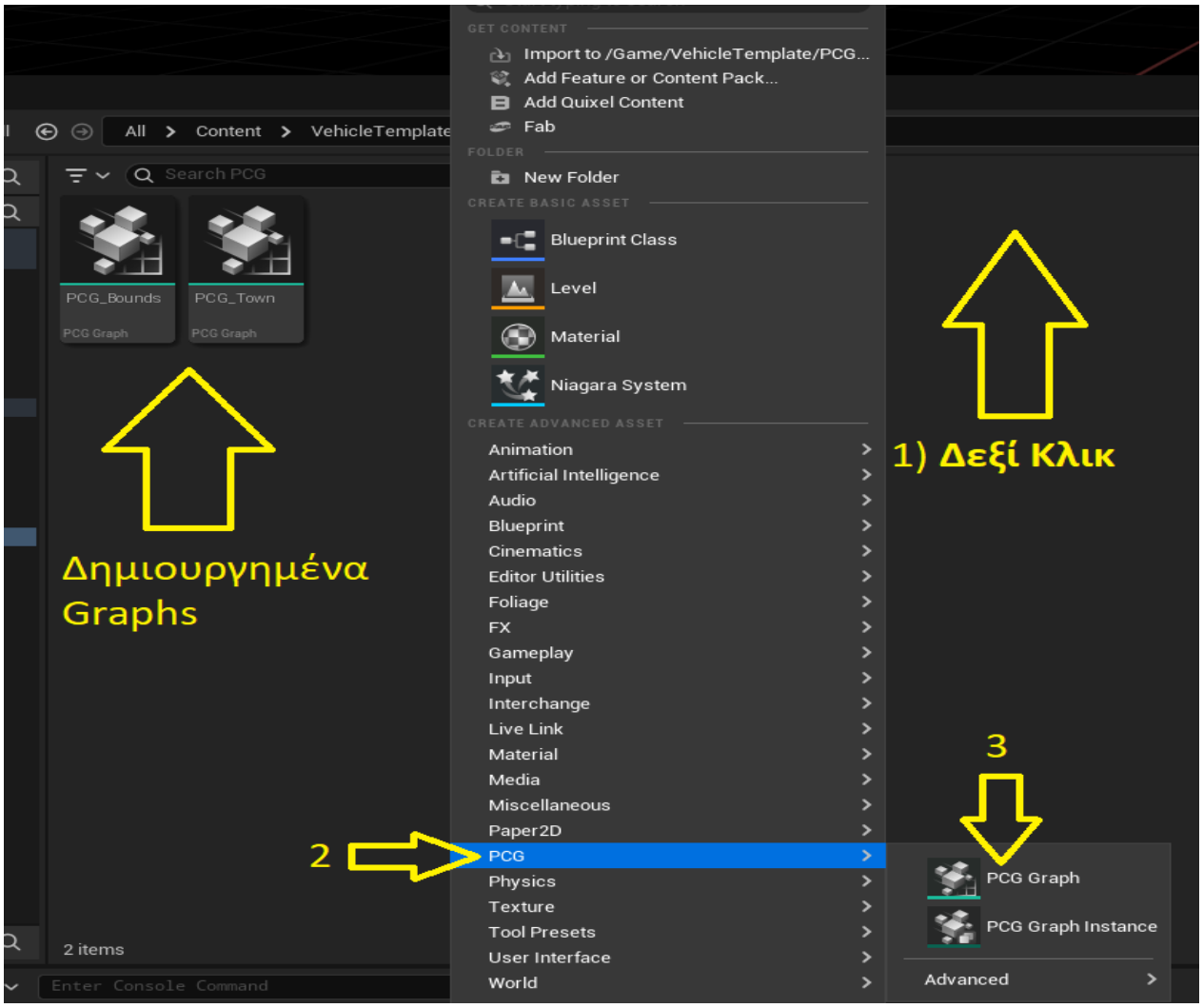
5.1) Οργάνωση Αντικειμένων Σκηνής Unreal

5.3 Δημιουργία κόσμου

Η δημιουργία του κόσμου ήταν το πρώτο κομμάτι στο οποίο ξεκίνησα να εργάζομαι. Ήταν το μεγαλύτερο το πιο απαιτητικό και το δυσκολότερο μέρος στην ανάπτυξη. Δυστυχώς το Vehicle template που χρησιμοποίησα, δεν είχε κάτι που να με βοήθησε στην δημιουργία του κόσμου.

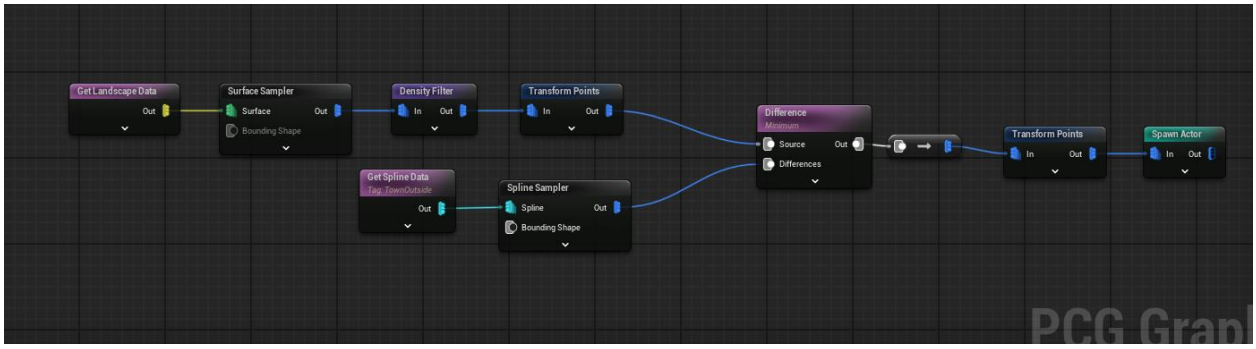
Ξεκινώντας διέγραψα όλα τα στοιχεία που δεν χρειαζόμουν από το αρχικό Level, αφήνοντας μόνο το αυτοκίνητο του παίκτη και το Landscape. Έπειτα μεγάλωσα το Landscape σε μέγεθος ώστε να είναι αρκετά μεγάλο για όλο το περιεχόμενο.

Έφτασα στο σημείο να αρχίζω να τοποθετώ το περιεχόμενο στον κόσμο μου. Ενεργοποίησα το PCG από το παράθυρο των plugin και αφού έγινε η επανεκκίνηση του editor που χρειάστηκε, δημιούργησα ένα PCG Graph (Δεξί κλικ στο Content Browser στον φάκελο που θέλουμε να δημιουργηθεί – PCG – PCG Graph).



5.2) Δημιουργία PCG Graph

Το πρώτο που δημιούργησα ήταν PCG_Bounds που ήταν υπεύθυνο για την δημιουργία του δάσους που θα υπήρχε σαν όρια χαρτιά για να μην μπορεί να βγει ο παίκτης. Τοποθέτησα το Graph με drag and drop στο Viewport μου, το μετακίνησα και το επεξεργάστηκα, έτσι ώστε να καλύπτει όλο το landscape. Άνοιξα το Graph και ήρθα αντιμέτωπος με ένα παρόμοιο παράθυρο σαν τα Blueprints. Αμέσως ξεκίνησα να ψάχνω και να τοποθετώ τα Nodes που θα έφτιαχναν το τελικό αποτέλεσμα που είχα στο μυαλό μου, με την βοήθεια διαφόρων tutorial που βρήκα στο διαδίκτυο.



5.3) Nodes PCG_Bounds

Το πρώτο Node ήταν το <Get Landscape Data>, με το οποίο διάβαζε το μέγεθος και την τοποθεσία του εδάφους που περιεχόταν μέσα στο Graph στο Viewport. Σε περίπτωση που σημεία του Landscape δεν ήταν μέσα στον κύβο (bounding box) που έδειχνε το Graph, τότε εκείνα τα σημεία δεν θα διαβάζονταν και δεν θα λαμβάνονταν υπόψη.

Αφού είχα όλα τα στοιχεία του εδάφους, έπρεπε να δημιουργήσω σε τυχαία σημεία στην επιφάνεια που είχα, σωματίδια (particles), ώστε να έχω μια οπτική εικόνα της κατανομής τους, καθώς και σωστής λειτουργίας του PCG. Γι' αυτό τον λόγο σύνδεσα το <Get Landscape Data> με ένα <Surface Sampler>. Στο συγκεκριμένο node μπορούμε να αλλάξουμε τα σωματίδια ανά τετραγωνικό (Points per Square Meter), δηλαδή τον αριθμό των συνολικών σωματιδίων και στην περίπτωση που η διάταξη των σημείων δεν μας αρέσει, υπάρχει η ρύθμιση "Seed" που ξανά δημιουργεί τα σημεία με άλλη τυχαία διάταξη. Τέλος, υπάρχει η επιλογή "Unbounded", με την οποία δεν περιορίζουμε την δημιουργία σωματιδίων στον bounding box του Graph. Στην δική μου εφαρμογή για μεγαλύτερη ασφάλεια, και ενεργοποίησα το Unbounded και κάλυψα το Landscape με το bounding box.

Επειδή όμως δημιουργούνταν πάρα πολλά σωματίδια στον κόσμο, έπρεπε να τα περιορίσω. Επομένως, χρησιμοποίησα το node <Density Filter> με το οποίο περιόρισα το αποτέλεσμα του <Surface Sampler> με βάση της πυκνότητας τους.

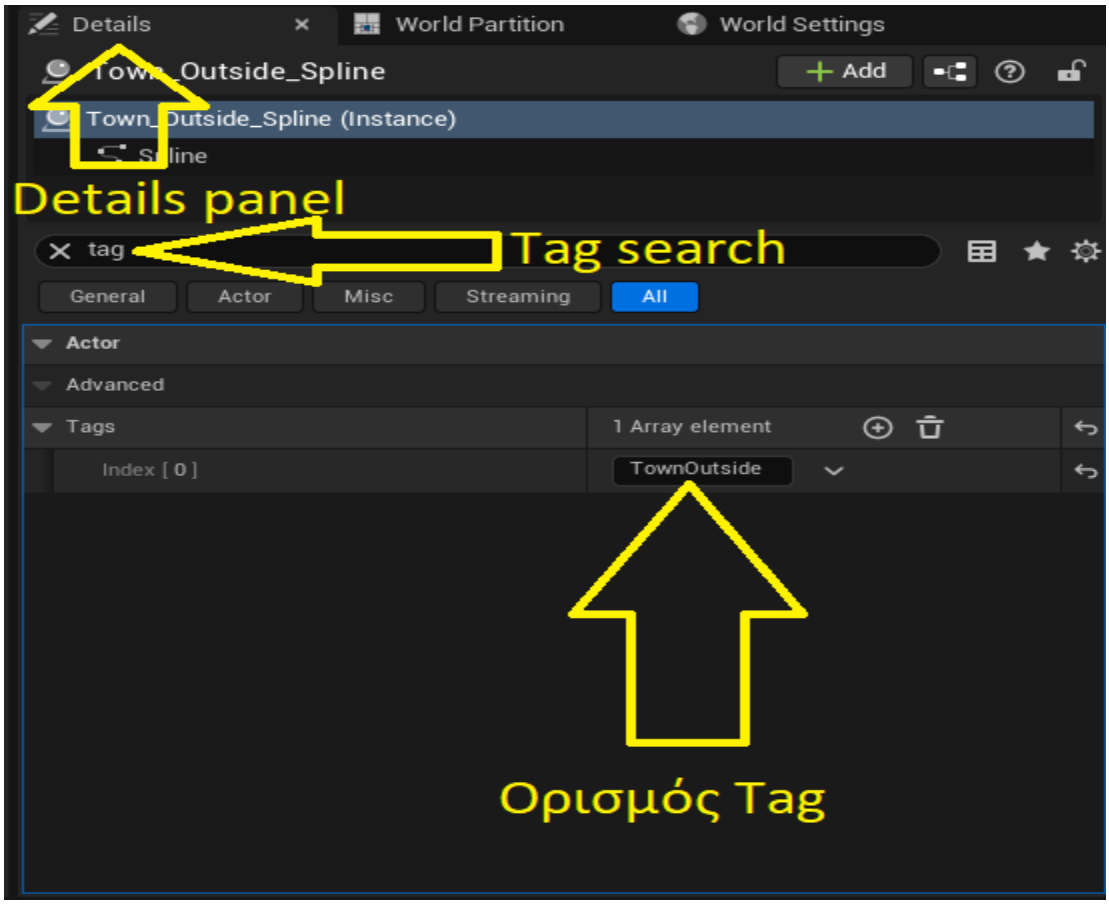
Όταν ήμουν ευχαριστημένος με το αποτέλεσμα, το σύνδεσα με το <Transform> node, μέσα από το οποίο μπορούσα ανά πάσα στιγμή να αλλάξω την θέση, την περιστροφή και την κλίμακα των σωματιδίων. Ορίζοντας ένα min και ένα max, το κάθε σωματίδιο θα έπαιρνε μια τυχαία τιμή μέσα σε αυτό το εύρος, δίνοντας μια ποικιλία.

Είχα έτοιμα έτσι τα σωματίδια στα οποία αργότερα θα έδινα γραφικά.

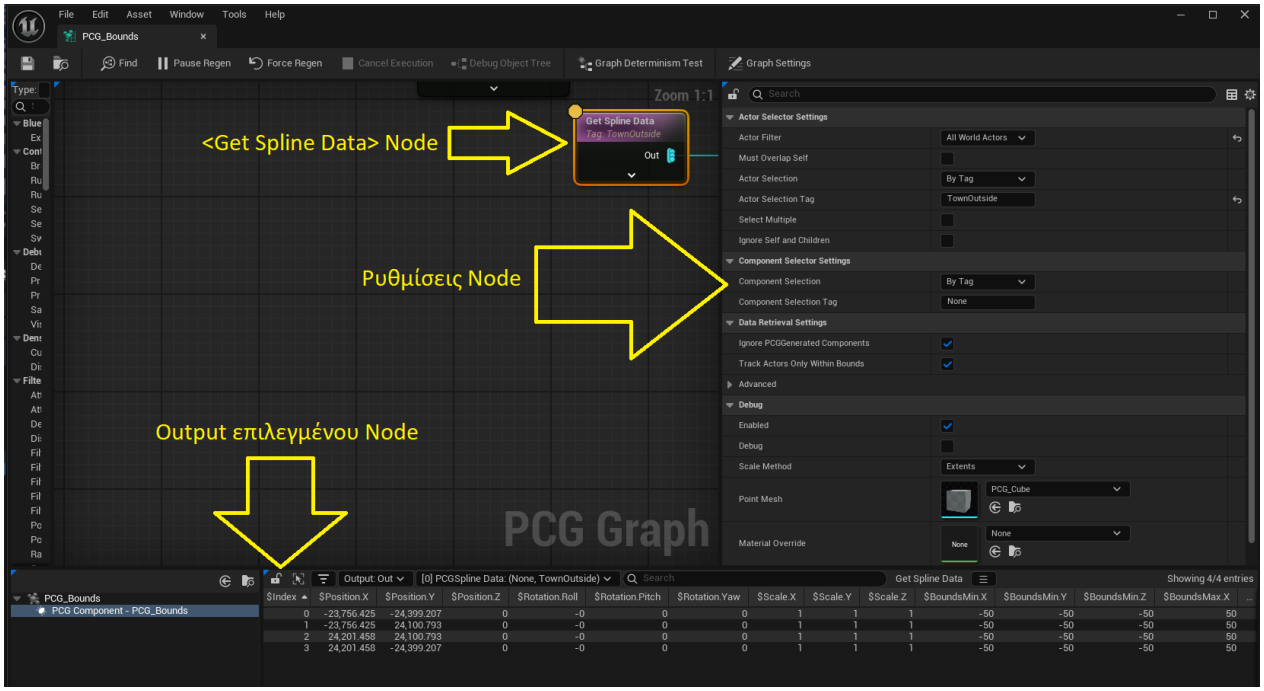
Επόμενο βήμα είναι να «κόψω» τα σημεία του landscape στα οποία δεν ήθελα να δημιουργηθούν σωματίδια, άρα και να μην υπάρχουν δέντρα. Για να το πετύχω αυτό, δημιούργησα ένα Spline (Town_Outside_Spline).

Επέκτεινα το spline έτσι ώστε να έχει 4 κορυφές (points) και ενεργοποίησα την ρύθμιση Closed Loop στο τελευταίο point κλείνοντας το spline και δημιουργώντας ένα κλειστό σχήμα. Στην συνέχεια μετέφερα της κορυφές του έτσι ώστε να σχηματιστεί ένα τετράγωνο κοντά στην άκρη του χάρτη. Και αφού είχα σχηματίσει και τον χώρο που δεν ήθελα να δημιουργηθούν δέντρα, έπρεπε να προσθέσω την λογική στο PCG_Bounds.

Για να «κόψω» τα σημεία μέσα από το spline, στο Blueprint PCG_Bounds (εικόνα 5.3), πρόσθεσα το <Get Spline Data>. Με τις προκαθορισμένες ρυθμίσεις όμως, δεν έπαιρνα τα δεδομένα του spline (τα 4 points). Γι' αυτό χρειάστηκε να προσθέσω κάποιο tag στο ίδιο το spline αντικείμενο στον editor (εικόνα 5.4). Έπειτα, στις ρυθμίσεις του <Get Spline Data> στο παράθυρο του blueprint, άλλαξα το πεδίο Actor Filter στην επιλογή "All World Actors" και το πεδίο Actor Selection στην "By Tag". Έτσι όταν στο αμέσως επόμενο πεδίο Actor Selection Tag, έγραψα το tag που είχα προηγουμένως δώσει στο spline μου, αυτό ξεκίνησε να διαβάζει τα 4 points (εικόνα 5.5).



5.4) Ορισμός Tag στον Editor



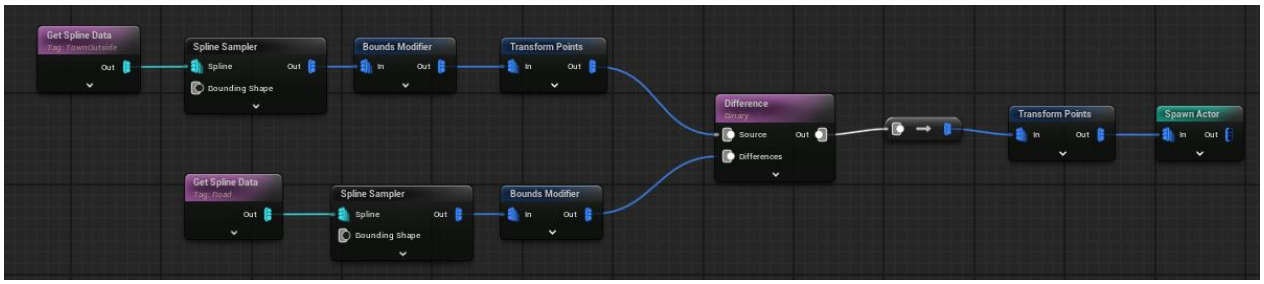
5.5) Ρυθμίσεις Spline Data

Αφού το Blueprint μου έπαιρνε δεδομένα από το spline, σύνδεσα στα Node του spline μου το <Spline Sampler>. Η μόνη ρύθμιση που χρειάστηκε να κάνω σε αυτό το node, ήταν να ορίσω το πεδίο Dimension στην επιλογή “On Interior”, επαληθεύοντας και από το Output ότι πλέον διαβαζόντουσαν όλα τα σημεία που βρίσκονταν εσωτερικά του spline.

Τέλος έμεινε να ενώσω τις 2 συνδεσμολογίες που είχα <Get Landscape Data> και <Get Spline Data>, με τέτοιο τρόπο ώστε από όλα τα σημεία που υπάρχουν στο landscape, να αφαιρούνται τα σημεία στο εσωτερικό του spline. Αυτήν την δουλειά μου την έκανε το node <Difference>. Συνδέοντας το <Transform> node στο pin “Source” του <Difference> και το <Spline Sampler> στο “Differences” pin, κατάφερα να δημιουργήσω το τελικό αποτέλεσμα που ήθελα, και πλέον είχα τα όρια του κόσμου μου.

Πρόσθεσα ακόμα ένα <Transform> node για δική μου ασφάλεια ότι θα υπήρχε μια τυχαιότητα στον τρόπο που θα εμφανίζονταν τα γραφικά μου, και τελείωσα το Blueprint με το node <Spawn Actor>, δίνοντας σαν μεταβλητή στο πεδίο “Template Actor Class”, το Blueprint για το δέντρο που ήθελα να εφαρμοστεί. Έτσι είχα έτοιμα τα όρια που μπορούσε να μετακινηθεί ο παίκτης.

Επόμενο βήμα μου, ήταν να δημιουργήσω τα κτίρια που θα αποτελούσαν την πόλη. Ακολούθησα την ίδια διαδικασία με την δημιουργία των ορίων. Τα κτίρια τοποθετήθηκαν με ένα δεύτερο PCG Graph (PCG_Town) στο εσωτερικό του Spline που είχα ήδη δημιουργήσει.



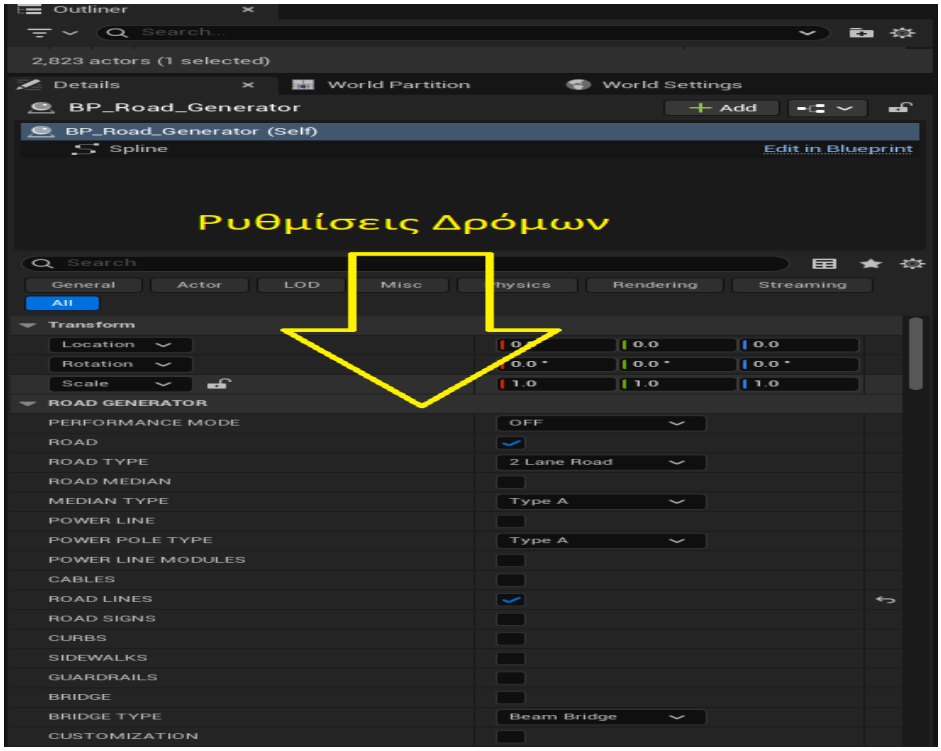
5.6) Nodes PCG_Town

Για να δημιουργήσω τα particles, πρώτα πήρα τα σημεία εσωτερικά του Spline που είχα ήδη. Χρησιμοποίησα πάλι τα 2 nodes <Get Spline Data> και <Spline Sampler> με την μόνη διαφορά στις ρυθμίσεις να είναι ότι στο <Spline Sampler>, στο πεδίο "Interior Sample Spacing" να αλλάξω την τιμή για να μην δημιουργούνται τόσα πολλά particles. Έπειτα τα σύνδεσα πάλι με τα 2 nodes <Bounds Modifier> <Transform Points> και στο "Source" pin ενός <Difference> node, αφού πάλι θα έπρεπε να αφαιρέσω τα particles επάνω από τα τους δρόμους μου.

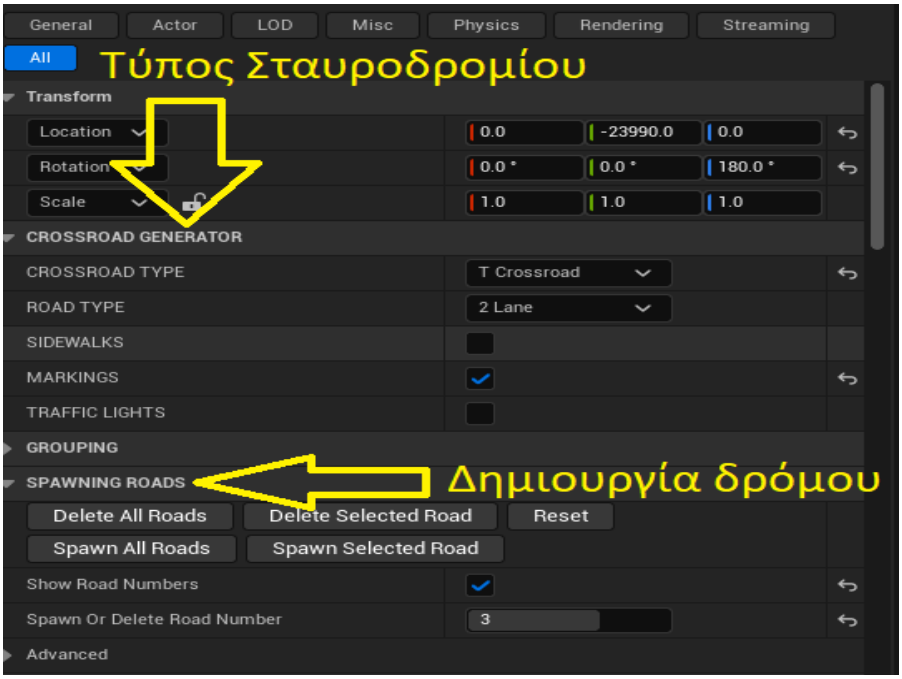
Για την δημιουργία των δρόμων χρησιμοποίησα έτοιμο περιεχόμενο από το marketplace. Το συγκεκριμένο περιεχόμενο, Crossroad Generator (εμφανίζεται σαν Ghost_Town_VOL8 μέσα στο project), περιέχει blueprints για την αυτόματη δημιουργία δρόμων. Περιέχει 2 blueprints, ένα είναι για τον δρόμο και το άλλο για σταυροδρόμια. (BP_Road_Generator και BP_Crossroad_Generator). Για την χρήση τους πρέπει απλά να γίνουν Drag and Drop μέσα στο Viewport. Με την ένταξη τους στο χάρτη, δημιουργούνται πολύ απλά meshes χωρίς πολλά γραφικά. Μέσω του Details Panel όμως μπορούν να προστεθούν και να γίνουν ακόμα πιο ρεαλιστικά. Υπάρχουν διάφορες ρυθμίσεις να ενεργοποιήσεις όπως πόσες λωρίδες θες να υπάρχουν στον δρόμο, αν θα υπάρχουν πεζοδρόμια, φανάρια, γραμμές στον δρόμο, τι είδους σταυροδρόμι θα είναι (X Crossroad , T Crossroad, Y Crossroad, Κυκλικό ή απλή διάβαση πεζών).

Τοποθέτησα πρώτα τα 5 σταυροδρόμια που χρειαζόμουν, 1 στην μέση (X Crossroad) και 4 στην μέση κάθε πλευράς (T Crossroad). Έπειτα από τις ρυθμίσεις κάθε σταυροδρομιού, επέλεξα να δημιουργηθούν αυτόματα δρόμοι. Έτσι τα blueprints των δρόμων στο Outliner δημιουργήθηκαν σαν παιδιά σε αυτά των σταυροδρομιών, οπότε ήταν πιο οργανωμένα. Τέλος, έπρεπε να επεκτείνω τους δρόμους ώστε να ενωθούν κατάλληλα για να διαμορφωθούν όπως ήθελα.

Τα blueprints των δρόμων, δημιουργούν splines. Αυτά τα splines έχουν την λογική ώστε σε κάθε segment (ένωση point με point), να δημιουργείται ένα mesh. Έτσι πρόσθετα points και τα επεξεργαζόμουν όπου χρειαζόταν, μέχρι να γίνουν snap στο τελευταίο point ενός άλλου spline δρόμου. Για να φαίνεται η στροφή πιο ρεαλιστική, χρειάστηκα αρκετή ώρα να επεξεργαστώ τα points που δημιουργούσαν την στροφή στις 4 άκρες του εξωτερικού δρόμου.



5.7) Details Panel για τις ρυθμίσεις των δρόμων

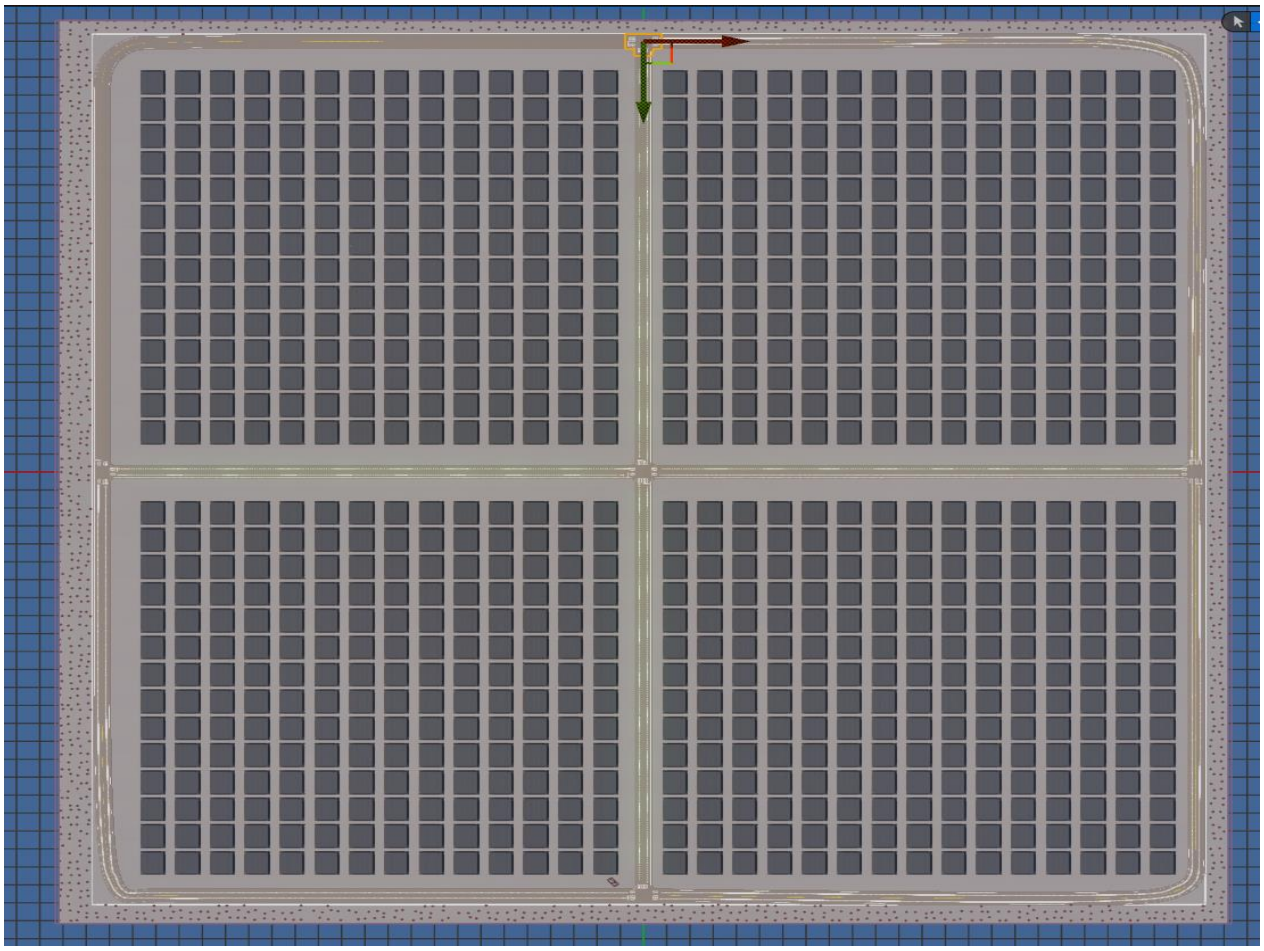


5.8) Ρύμιση για αλλαγή τύπου σταυροδρομίου

Έτσι είχα έτοιμα τους δρόμους και τα κτίρια, και το μόνο που μου έμενε είναι να αφαιρούσα τα σημεία των κτιρίων που βρίσκονταν στο χώρο που υπήρχαν δρόμοι. Αυτό το έκανα μέσω του PCG_Town. Στο blueprint πρόσθεσα τα nodes <Get Spline Data>, <Spline Sampler> και αργότερα πρόσθεσα και το <Bounds Modifier>. Έβαλα το ίδιο tag σε όλα τα blueprints δρόμων και εφάρμοσα την ίδια λογική για διάβασμα σημείων, όπως σε όλα τα προηγούμενα ίδια node.

Στο <Spline Sampler> αυτή την φορά χρησιμοποίησα το “On Spline” ενώ στο πεδίο Mode, “Subdivision”. Όπως και στο PCG_Bounds, σύνδεσα αυτήν την συνδεσμολογία στο Differences pin του <Difference> και αφαίρεσα τα κτίρια που δεν ήθελα. Ξανά, σύνδεσα το <Difference> με ένα <Transform> και ένα <Spawn Actor>, και διαλέγοντας το blueprint του κτιρίου που ήθελα, εμφανίστηκαν μόνο τα κτίρια που χρειαζόμουν.

Δυστυχώς, τα γραφικά των κτιρίων ήταν αρκετά μεγάλα και σε κάποια σημεία έπεφταν πάνω στον δρόμο. Προσθέτοντας το <Bounds Modifier> στο <Get Spline Data> των δρόμων, και αλλάζοντας τα Bounds Min και Max, πρόσθετα όρια για το πόσο μακριά από το βασικό spline θα διαβάζει σημεία. Έτσι κόβοντας περισσότερα particles από αυτά που θα εμφανίζονταν, είχα και το επιθυμητό αποτέλεσμα. Βέβαια αυτή η μέθοδος είχε σαν αποτέλεσμα να υπάρχει μεγάλο κενό μεταξύ του δρόμου και τα πρώτα κτίρια.



5.9) Top Down View της πόλης

5.4 Paths

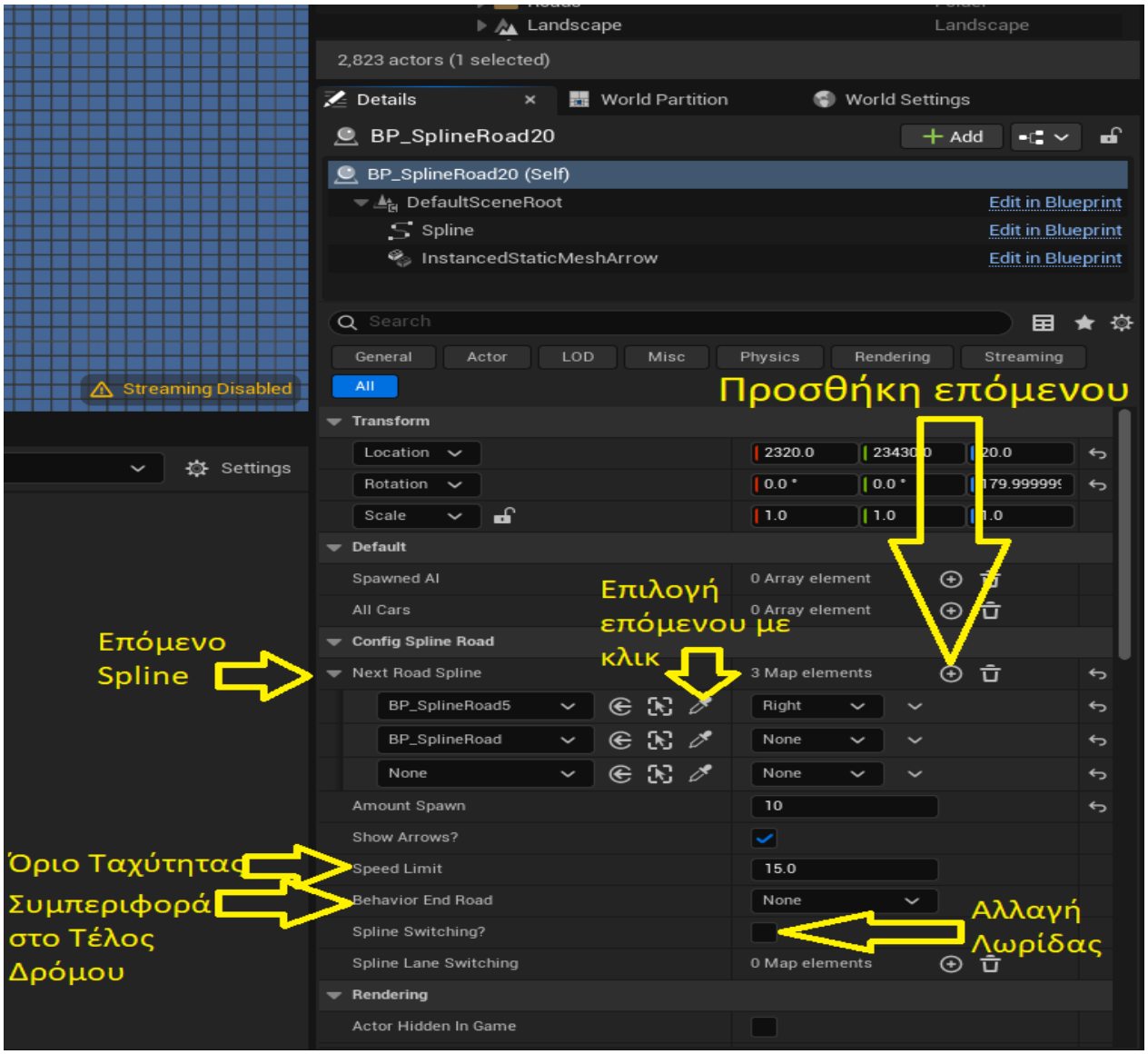
Τα Paths των υπόλοιπων αυτοκινήτων στον κόσμο, έγινε μέσω του αποκτημένου περιεχομένου Traffic AI System. Το Traffic AI System είναι το μόνο από τα περιεχόμενα που απέκτησα με πληρωμή, αλλά προχώρησε την δημιουργία του παιχνιδιού πολύ. Το συγκεκριμένο περιεχόμενο περιέχει διάφορα assets, αυτοκίνητα, χαρακτήρα, φανάρια, δρόμους και splines για την κίνηση και το παρκάρισμα των αυτοκινήτων.

Για όλα τα παραπάνω έχει blueprint με πολύπλοκη λογική, που χρειάστηκε αρκετή προσοχή και διάβασμα για την κατανόηση της. Το blueprint που χρειάστηκα εγώ ήταν το BP_Spline_Road το οποίο ήταν υπεύθυνο για την δημιουργία και κίνηση των αυτοκινήτων AI. Έπρεπε να τοποθετήσω αρκετά τέτοια blueprints στον χάρτη μου και να τα προσαρμόσω, καθώς κάθε ένα ήταν αρμόδιο για μια λωρίδα. Για παράδειγμα, για κάθε δρόμο στην δικιά μου περίπτωση, χρειάστηκα 2 τέτοια blueprints, αφού οι δρόμοι αποτελούνταν από μια λωρίδα για κάθε κατεύθυνση. Σύνολο χρησιμοποίησα 24, ενώ για πιο μεγάλο χάρτη και για μεγαλύτερους δρόμους, αυτός ο αριθμός μπορεί πολύ εύκολα να γίνει πολύ μεγαλύτερος.

Στα θετικά, μέσα από αυτά τα blueprints μπορείς να ρυθμίσεις την ταχύτητα τον αυτοκινήτων που κινούνται πάνω σε κάθε spline. Στο Details Panel, υπάρχει η ρύθμιση “Speed Limit”. Επίσης, μπορείς να ενεργοποιήσεις ρυθμίσεις για αλλαγή λωρίδας, και συμπεριφορά αυτοκινήτου στο τέλους του spline (διαγραφή αυτοκινήτου ή επανάληψη). Η πιο σημαντική ρύθμιση όμως είναι η “Next Road Spline”, μέσα από την οποία πρόσθετα τα επόμενα spline που μπορεί να μετακινηθεί το αυτοκίνητο καθώς και την μεριά στην οποία θα στρίψει, για να ανάψει το κατάλληλο φλας.

Μπορείς να προσθέσεις πολλαπλά spline και είτε να διαλέξεις από το dropdown το επόμενο spline είτε να διαλέξεις το στυλό στα δεξιά του dropdown και να κάνεις κλικ στο Viewport στο spline που θέλεις. Σε περίπτωση που ενεργοποιηθεί η αλλαγή λωρίδας, η διαδικασία για την επιλογή λωρίδας στην οποία μπορεί να αλλάξει, είναι η ίδια με την επιλογή επόμενου spline.

Αφού μελέτησα αρκετά το blueprint και το περιεχόμενο, κατάλαβα ότι με κάποιες διαμορφώσεις στα blueprint, μπορούν να χρησιμοποιηθούν για την κίνηση όχι μόνο αυτοκινήτων αλλά και οτιδήποτε άλλο θέλει κανείς, όπως πεζούς για παράδειγμα.

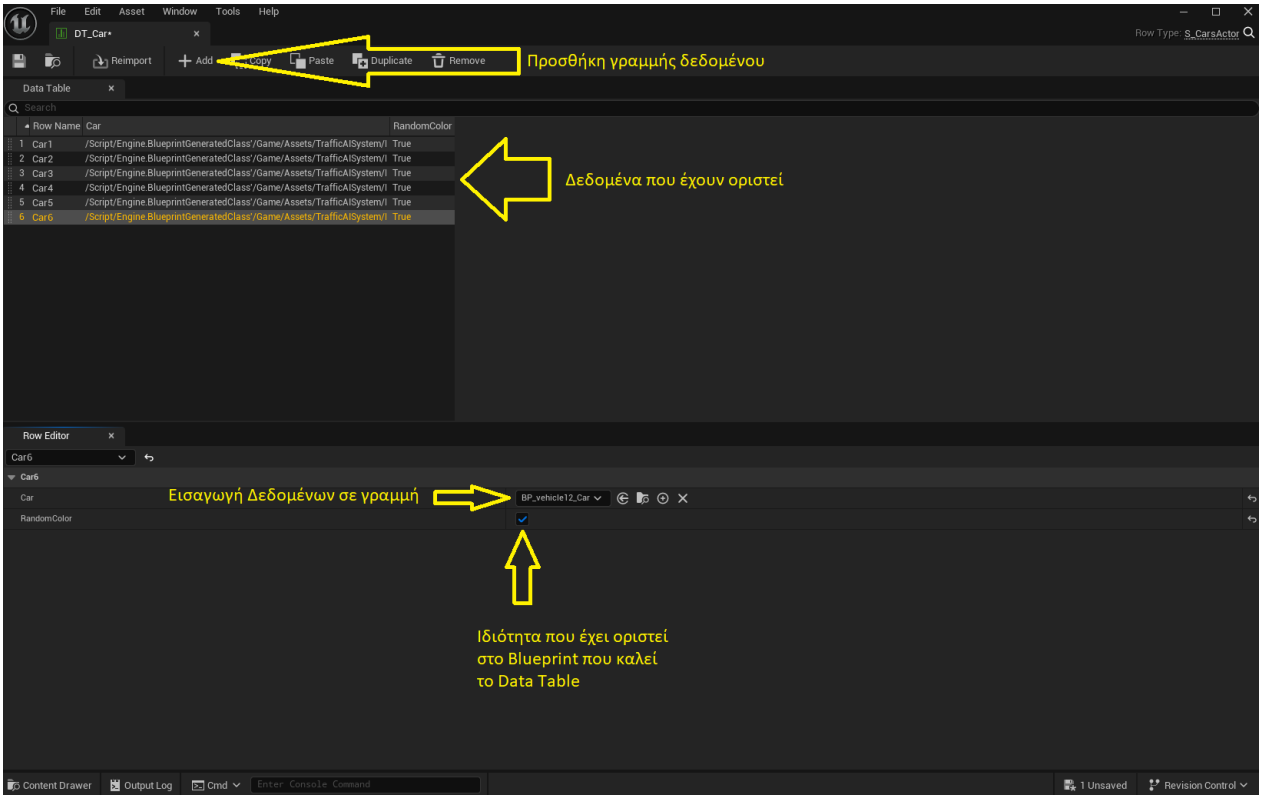


5.10) Ρυθμίσεις BP_SplineRoad

5.5 Vehicle

Τα αυτοκίνητα επίσης δημιουργήθηκαν μέσα από το Traffic AI System. Ο φάκελος Cars περιέχει 6 έτοιμα blueprints για διαφορετικά αυτοκίνητα. Όλα τα blueprint έχουν λογική για φώτα φρένων και φλάς, τα οποία ενεργοποιούνται με βάση την απόσταση από κάποιο αμάξι μπροστά για την αποφυγή τρακαρίσματος και τις ρυθμίσεις για κάθε είδος αλλαγής spline πάνω στο οποίο κινούνται.

Όλα τα blueprints έχουν οριστεί σαν δεδομένα στο DT_Car Data Table. Το Data Table λειτουργεί σαν πίνακας με δεδομένα έτσι ώστε άλλα περιεχόμενα να μπορούν να τα διαβάσουν όλα και να επιλέξουν κάποιο από αυτά. Data Table μπορούμε να δημιουργήσουμε με δεξί κλικ στο content browser και να το επιλέξουμε από το μενού Miscellaneous.



5.11) Παράθυρο DT_Car

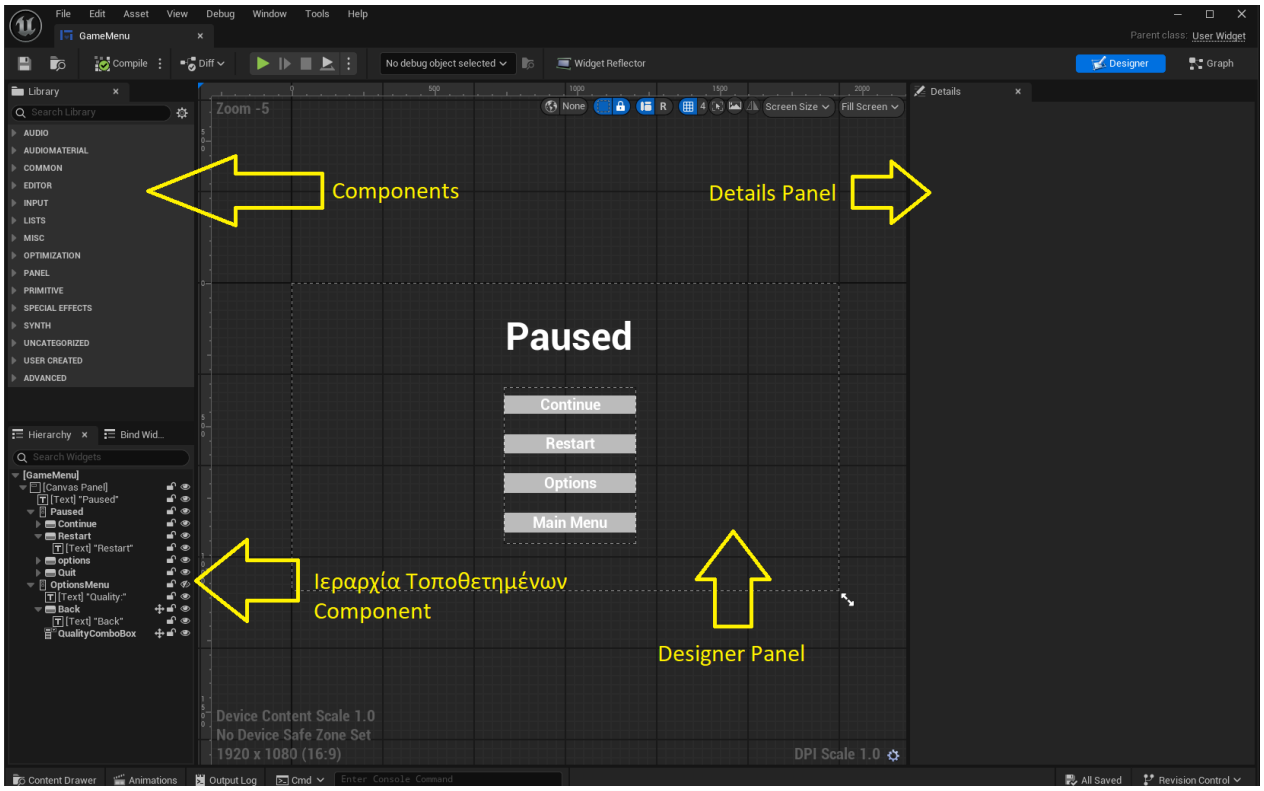
Αυτό το καταλαβαίνουμε μέσα από την χρήση του στο BP_SplineRoad, το οποίο χρησιμοποιεί αυτό το Data Table ώστε να δημιουργήσει τυχαία αυτοκίνητα. Μέσα από τις ρυθμίσεις του spline, ο χρήστης μπορεί να επιλέξει τον αριθμό τον αυτοκίνητων που θέλει να δημιουργηθούν (πεδίο "Amount Spawn" κάτω από το πεδίο επόμενου Spline). Έχοντας τον επιθυμητό αριθμό και το Data Table, το blueprint δημιουργεί τυχαία όσα αυτοκίνητα θέλουμε από αυτά που έχουμε θέσει σαν δεδομένα. Τα ίδια αυτοκίνητα μπορούν να δημιουργηθούν παραπάνω από 1 φορά. Μεγάλη προσοχή θέλει ο επιθυμητός αριθμός. Ο αριθμός που ορίζουμε στο πεδίο είναι ο αριθμός των αυτοκινήτων που θα εμφανιστούν σε κάθε spline. Για παράδειγμα, αν στον κόσμο βάλουμε 10 splines και βάλουμε την τιμή 10 στο "Amount Spawn" σε όλα, τότε κατά την έναρξη του παιχνιδιού θα εμφανιστούν 10*10=100 αυτοκίνητα, το οποίο μπορεί να μην το θέλουμε. Επίσης όσα περισσότερα αυτοκίνητα θέσουμε να δημιουργηθούν κατά την έναρξη, τόσο μεγαλύτερος θα είναι και ο φόρτος που θα έχει ο υπολογιστής για να κάνει render το παιχνίδι, άρα μεγαλύτερη δαπάνη πόρων.

5.6 Διεπαφή Χρήστη

Για την καλύτερη αλληλεπίδραση με τον χρήστη, δημιούργησα μερικά μενού (Start Menu, Pause Menu και Collision Menu). Για την δημιουργία τους, έφτιαξα 3 Widget Blueprint, το καθένα με την δίκη του λογική, για το πότε θα εμφανίζονται και τις λειτουργίες που θα έχουν. Στην αρχή δημιούργησα έναν ξεχωριστό

φάκελο Widgets που θα τα αποθήκευα όλα, και έπειτα δεξί κλικ στο Content Browser και από το μενού User Interface, την επιλογή Widget Blueprint.

Πρώτα έφτιαξα το Pause Menu. Άνοιξα το widget blueprint, και εμφανίστηκε το παράθυρο το οποίο χωρίζεται σε 4 μέρη. Στο αριστερά μέρος, υπάρχουν το Library, που περιέχει όλα τα Components που μπορούν να ενταχθούν για να δημιουργηθεί ένα Widget όπως κουμπιά, Texts και Dropdown Lists, και το Hierarchy, που περιέχει όλα τα Components που χρησιμοποιούνται. Στο κέντρο υπάρχει ένα Viewport (Designer panel) στο οποίο εμφανίζονται τα components όπως θα εμφανίζονται στον παίκτη. Στα δεξιά, υπάρχει το Details, μέσω του οποίου μπορείς να επεξεργαστείς τα components του Widget.



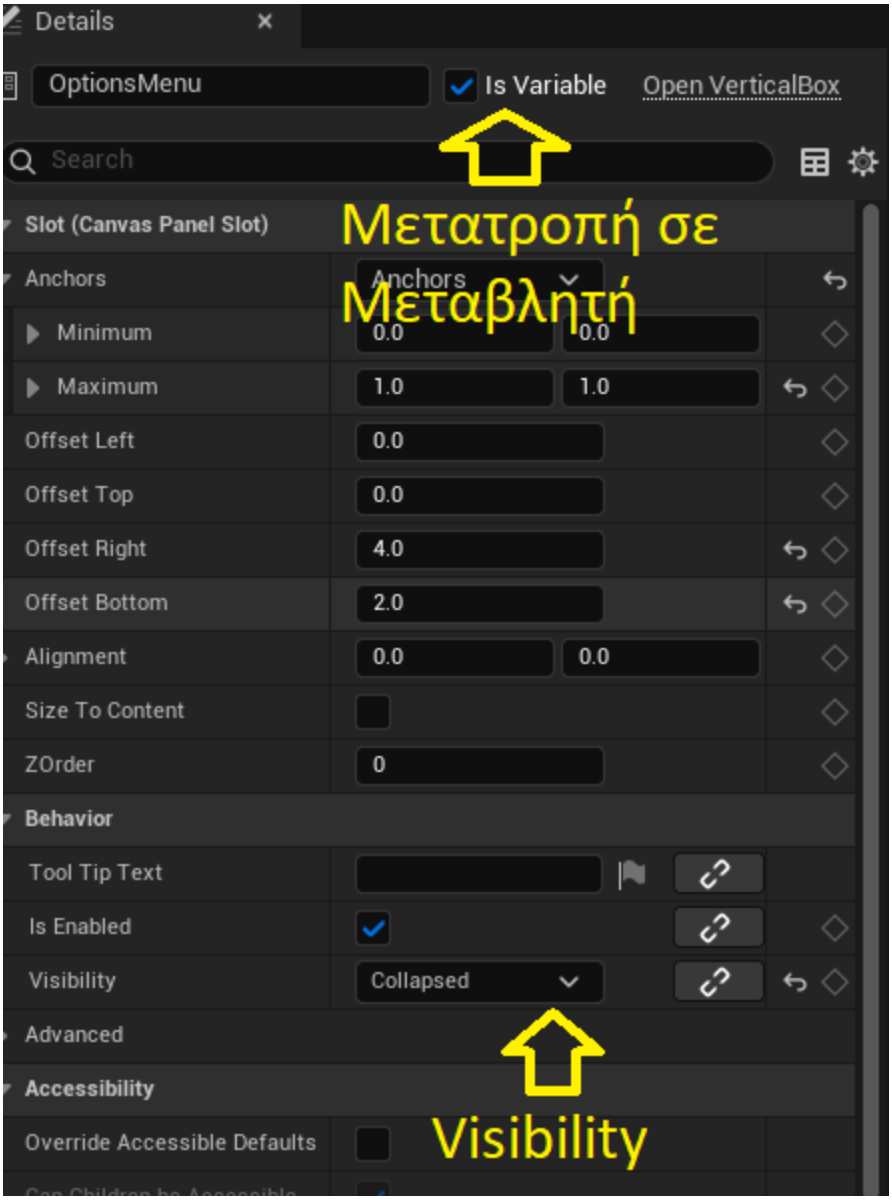
5.12) Παράθυρο Widget Blueprint

Ξεκίνησα με το να τοποθετώ ένα Canvas το οποίο θα περιέχει όλα τα υπόλοιπα components. Μέσα σε αυτό τοποθέτησα ένα Text στο πάνω μέρος του canvas με την λέξη “Paused” για να ξέρει ο παίκτης ότι το παιχνίδι έχει σταματήσει. Επίσης στο canvas πρόσθεσα 2 vertical boxes, ένα με τα αρχικά κουμπιά που θα φαίνονται όταν ανοίγει το μενού (Paused) και ένα με τις ρυθμίσεις του παιχνιδιού (Options Menu).

Στο Paused χρειάστηκα 4 κουμπιά και 4 Text για κάθε κουμπί. Το πρώτο ήταν το κουμπί συνέχειας παιχνιδιού (Continue), το επόμενο ήταν το κουμπί επανεκκίνησης παιχνιδιού (Restart), το τρίτο ήταν το κουμπί ρυθμίσεων (Options) και τελευταίο το κουμπί αρχικού μενού (Main Menu) για επιστροφή στο Start Menu. Στο Options χρειάστηκα ένα κουμπί επιστροφής (Back), για επιστροφή στο Paused, ένα Combo Box για τις επιλογές που θα είχε ο παίκτης για αλλαγή ποιότητας γραφικών και ένα Text με την περιγραφή του Combo Box.

Η επεξεργασία των ρυθμίσεων για να εμφανίζεται το μενού όπως το σκεφτόμουν ήταν αρκετά απλή. Δεν χρειάστηκαν πολλές αλλαγές πέρα από την μετακίνηση των component στο σημείου που ήθελα να

εμφανίζονται, αλλαγή όλων των component που θα είχαν κάποια λειτουργία σε μεταβλητές (ενεργοποίηση τις επιλογής “Is Variable” στην κορυφή των Details) και την αλλαγή του “Visibility” πεδίου του Options Vertical Box σε Collapsed, ώστε να μην εμφανίζεται όταν ανοίγει το ο χρήστης το μενού.



5.13) Ρυθμίσεις Component

Αφού το μενού ήταν ικανοποιητικό, μετά έπρεπε να προσθέσω τις λειτουργίες. Από το πάνω δεξιά μέρος, πλοηγήθηκα στο Graph για την προσθήκη nodes. Για κάθε κουμπί χρησιμοποίησα το node <On Clicked>, ενώ για το Combo Box χρησιμοποίησα το <On Selection Changed>.

Για το κουμπί Continue πρόσθεσα τα <Remove All Widgets> και <Set Game Paused> (False τιμή) nodes ώστε να εξαφανίζονται όλα τα Widget στην οθόνη και να συνεχίζει το παιχνίδι από εκεί που σταμάτησε.

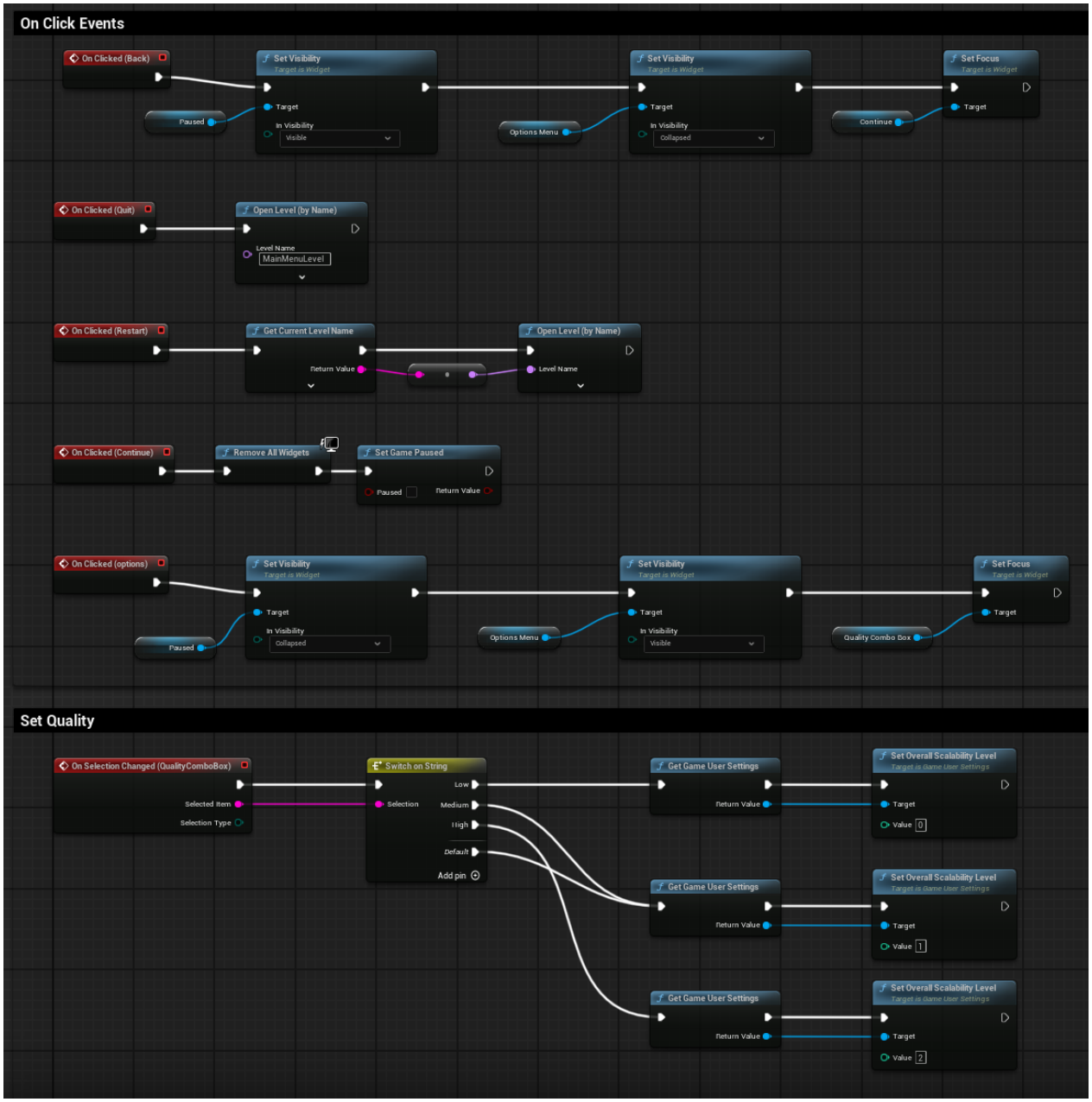
Για το κουμπί Restart πρόσθεσα τα nodes <Get Current Level Name> και <Open Level (by Name)> ώστε να διαβάζεται το Level που είναι ανοικτό και να ξανανοίγει από την αρχή, κάνοντας reset.

Για το κουμπί Options πρόσθεσα 2 <Set Visibility> nodes, με target τα vertical boxes που είχα. Στο node με target το Paused έθεσα το Visibility σε Collapsed για να εξαφανιστεί από την οθόνη, και το node με target το Options Menu σε Visible για να εμφανιστεί. Ύστερα πρόσθεσα και ένα <Set Focus> με target το Combo Box, ώστε να μπορεί να γίνει πλοήγηση με το χειριστήριο γιατί σε διαφορετική περίπτωση δεν θα γινόταν.

Για το κουμπί Main Menu πρόσθεσα το node <Open Level (by Name)> με μεταβλητή το Level του Start Menu, αφού το Start Menu ήταν δημιουργημένο σε διαφορετικό Map.

Για το κουμπί Back ακολούθησα την ίδια ακριβώς συνδεσμολογία με το Options, με την διαφορά ότι Visibility ήταν Visible για το Paused και Collapsed για το Options Menu.

Στο Combo Box ήταν πιο πολύπλοκη διαδικασία. Στο <On Selection Changed> node σύνδεσα ένα <Switch on String> στο οποίο μπορούσα να θέσω τιμές και να τις συγκρίνει με τις τιμές που θα λάμβανε από το Combo Box. Πρόσθεσα τις Low, Medium και High, σε κάθε σύνδεσα ένα <Get Game User Settings> το οποίο συνδεόταν στο <Set Overall Scalability Level>. Το Scalability όμως μετριέται σε integer τιμές από το 0-4 για Low, Medium, High, Epic, Cinematic. Αφού εγώ ήθελα τα πρώτα 3 επίπεδα έπρεπε να θέσω τις τιμές στα αντίστοιχα <Set Overall Scalability Level> ως 0-2. Με αυτό τον τρόπο ο παίκτης θα μπορούσε να θέσει τα γραφικά του παιχνιδιού για καλύτερη απόδοση ή καλύτερη οπτική ποιότητα.

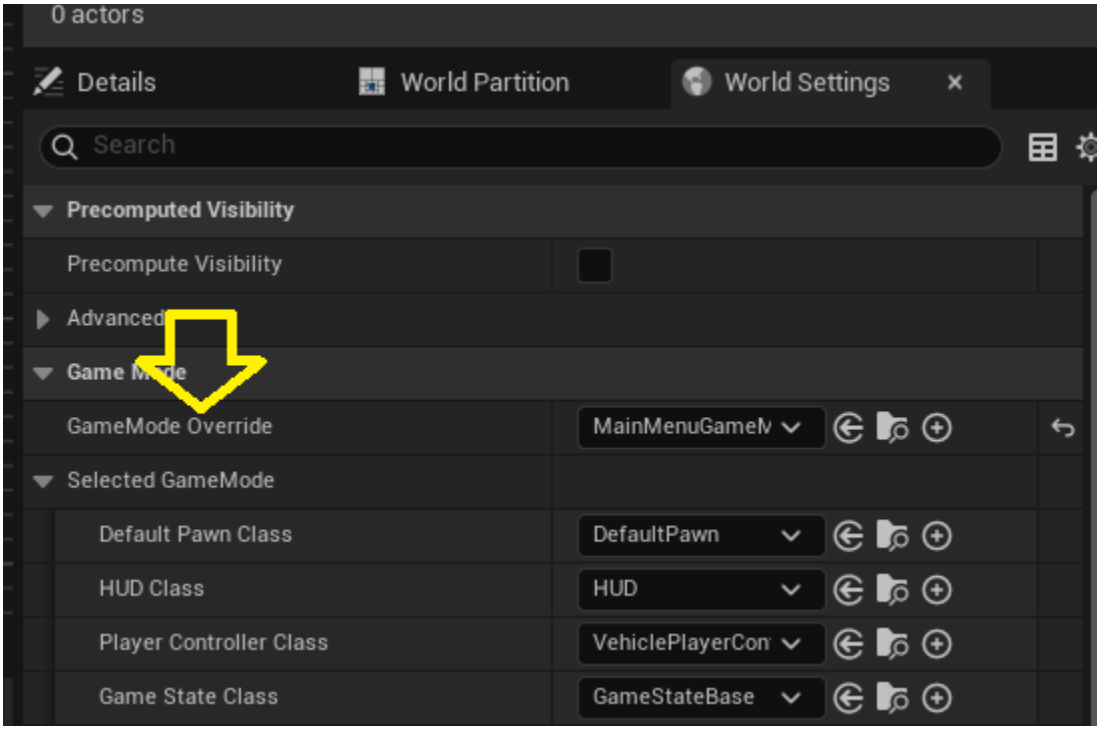


5.14) Pause Menu Graph

Ίδια ακριβώς λογική ακολουθήθηκε και για το Collision Menu. Ξεκίνησα με ένα Canvas και στο εσωτερικό ένα Text για το μήνυμα τρακαρίσματος και 2 κουμπιά Restart και Main Menu για επανεκκίνηση του παιχνιδιού και για επιστροφή στο αρχικό μενού αντίστοιχα.

Τελευταίο έφτιαξα το Start Menu. Για να το κάνω αυτό πρώτα δημιούργησα ένα καινούργιο Level (Δεξί κλικ στο Content Browser και Level), ένα Game Mode Base Blueprint (δεξί κλικ στο Content Browser, Blueprint και Game Mode Base τύπο blueprint) και ένα Widget Blueprint.

Το Level αυτό είναι άδειο, χωρίς Landscape και χωρίς κανένα άλλο αντικείμενο. Η εμφάνιση του Widget στο Level, γίνεται μέσω του Game Mode που δημιουργήσαμε. Στο World Settings Panel του Level υπάρχει το πεδίο "GameMode Override". Σε αυτό πρέπει να δώσουμε την επιλογή του Game Mode που δημιουργήσαμε.

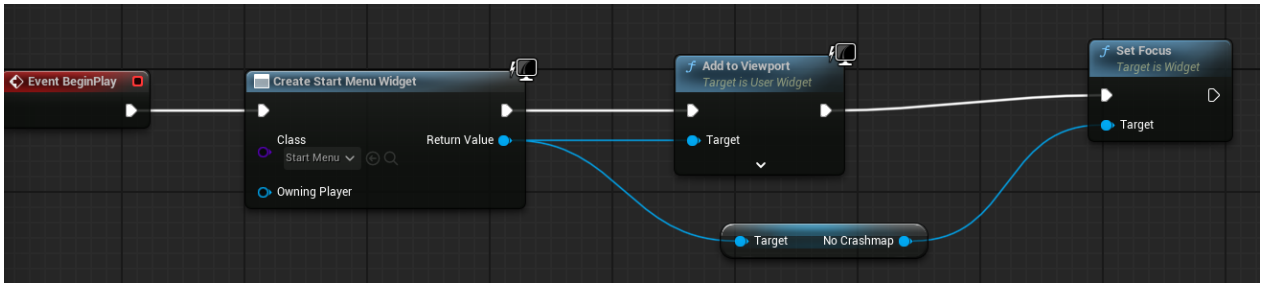


5.15) Ρύθμιση GameMode

Στην συνέχεια, θα δημιουργήσουμε το Widget. Για το χτίσιμο του, ακολούθησα την ίδια λογική με τα προηγούμενα μενού με την επιπλέον προσθήκη του Image Component, για την εικόνα στο πίσω μέρος του μενού. Για την εισαγωγή εικόνας στο project μας, δεν αρκεί ένα απλό Drag and Drop στον φάκελο που θέλουμε. Πρέπει από το toolbar του Content Browser να γίνει Import.

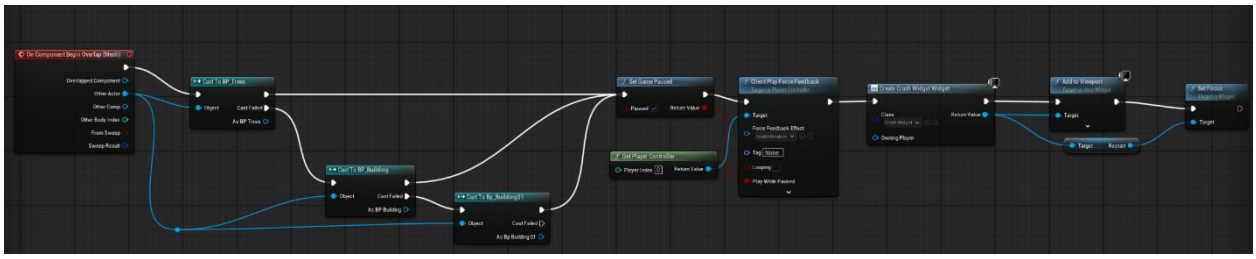
Αφού η εμφάνιση του αρχικού μενού μου ήταν έτοιμο, έμενε να του δώσω τις λειτουργίες που έπρεπε. Για τα κουμπιά ακολούθησα ξανά την ίδια λογική με τα προηγούμενα μενού, αφού οι λειτουργίες των κουμπιών ήταν ίδιες (NoCrashMap και SurvivalMap τα 2 Levels, Options το μενού ρυθμίσεων και Exit το κλείσιμο της εφαρμογής).

Έχοντας το Widget έτοιμο, έπρεπε να το εμφανίζω με το άνοιγμα του Level. Αυτό έγινε μέσα από το Game Mode. Στο node <Event BeginPlay> σύνδεσα πρώτα το <Create Widget> με επιλεγμένη κλάση το Start Menu widget. Έπειτα αυτό το σύνδεσα με το <Add to Viewport> για να εμφανίζεται στην οθόνη του παίκτη. Τέλος, πρόσθεσα και ένα <Set Focus> με target το πρώτο κατά σειρά κουμπί που είχα στο αρχικό μενού μου (No Crash Map).



5.16) Game Mode Graph

5.7 Vehicle Παίκτη



5.17)Blueprint Αυτοκινήτου

Η παραπάνω εικόνα [3.3], αντιστοιχεί στην λογική που έδωσα στο αμάξι της εφαρμογής μου για τις περιπτώσεις τρακαρίσματος με άλλα Blueprints που υπάρχουν στον χάρτη. Όταν το όχημα ξεκινήσει να "πέφτει" (On Component Begin Overlap) πάνω σε κάποιο άλλο αντικείμενο (Cast to "Blueprint" του αντικειμένου), τότε κάνει παύση το παιχνίδι (Set Game Paused), δημιουργεί και εμφανίζει στο χρήστη ένα γραφικό στοιχείο (Widget) με το κατάλληλο μήνυμα (<Create " Όνομα Widget"> και <Add to Viewport> αντίστοιχα), ενώ παράλληλα στην περίπτωση που ο χρήστης χρησιμοποιεί χειριστήριο (Gamepad), εφαρμόζει μια μικρή δόνηση (Client Play Force Feedback).

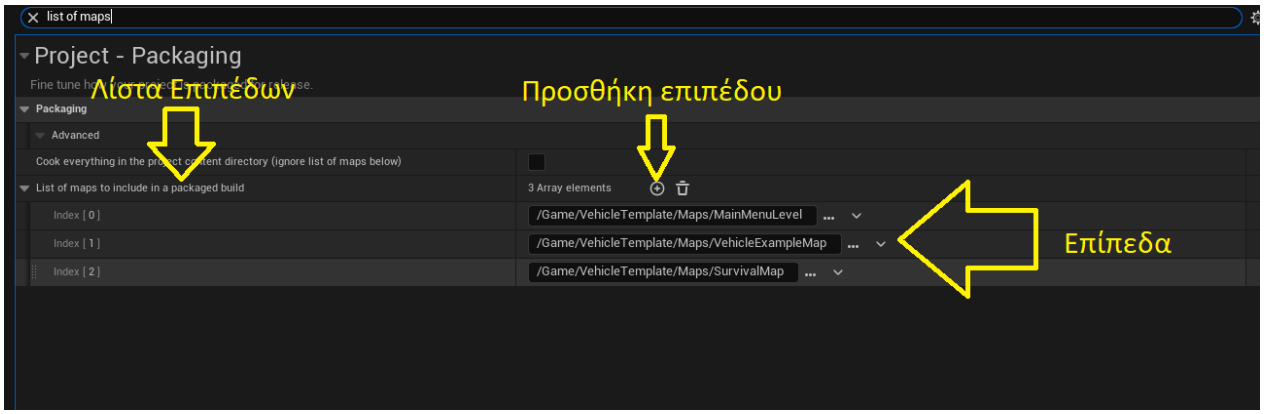
5.8 Build και Εκτέλεση

Έχοντας υλοποιήσει όλα τα επιμέρους στοιχεία που ήθελα, έμενε πλέον να ρυθμίσω την τελική μορφή του παιχνιδιού και να το δοκιμάσω πλήρως. Για να γίνει αυτό σωστά, έπρεπε να καθοριστούν οι σκηνές (levels) που θα περιλαμβάνονται στο παιχνίδι και να οριστεί το αρχικό επίπεδο (level) εκκίνησης.

Αυτές οι ρυθμίσεις γίνονται μέσα από το Project Settings. Συγκεκριμένα, από το κεντρικό μενού επιλέγουμε Edit – Project Settings και στο αριστερό μενού του παραθύρου που ανοίγει, πλοηγούμαστε στην ενότητα Maps & Modes. Εκεί μπορούμε να ορίσουμε το Edit Startup Map, το επίπεδο που φορτώνεται όταν ανοίγουμε τον Editor, και το Game Default Map, το επίπεδο από το οποίο θέλουμε να ξεκινάει το παιχνίδι μας.

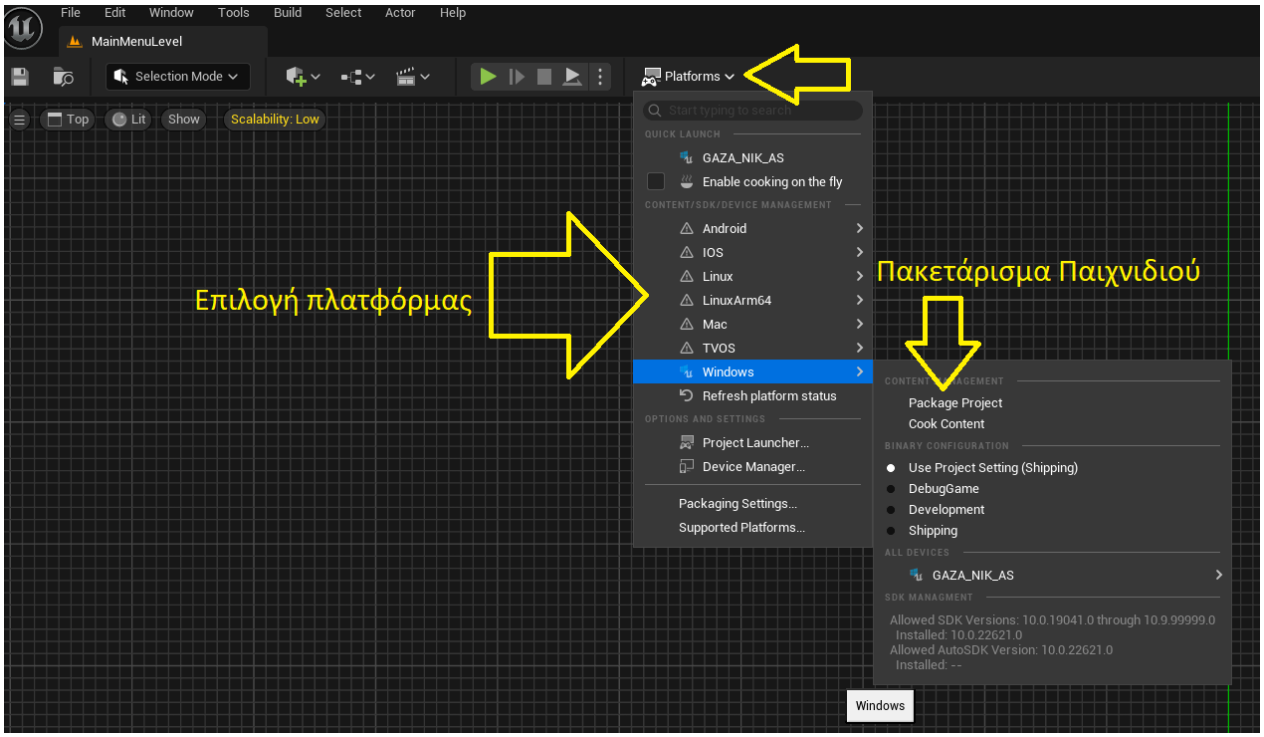
Επέλεξα λοιπόν στο Game Default Map το επίπεδο του αρχικού μενού (StartMenu), έτσι ώστε αυτό να είναι το πρώτο που φορτώνεται όταν τρέχει η εφαρμογή.

Για να διασφαλίσω ότι όλα τα απαραίτητα levels θα συμπεριληφθούν στο τελικό build, πήγα στο File - Package Project - Packaging Settings, και βεβαιώθηκα ότι στο τμήμα List of Maps to Include in a Packaged Build, περιλαμβάνονται όλα τα βασικά επίπεδα. Αν κάποιο λείπει, μπορώ να το προσθέσω χειροκίνητα μέσω της επιλογής +.



5.18) Προσθήκη Επιπέδων

Μετά την ολοκλήρωση αυτών των ρυθμίσεων, μπορούσα να κάνω build και να τρέξω το παιχνίδι. Η Unreal προσφέρει την επιλογή να κάνω Package το project μου από το toolbar (Platforms) , όπου μπορώ να επιλέξω την πλατφόρμα στην οποία θέλω να κάνω export (π.χ. Windows, Linux, Android κ.λπ.).



5.19) Package Project

Το πρώτο build μπορεί να πάρει περισσότερο χρόνο, καθώς δημιουργούνται όλα τα απαραίτητα αρχεία και γίνεται αρχική μεταγλώττιση περιεχομένου. Μόλις ολοκληρωθεί η διαδικασία, ο φάκελος εξαγωγής θα περιέχει το εκτελέσιμο αρχείο του παιχνιδιού.

Μια καλή τακτική είναι ο χρήστης να κάνει τακτικά Build και να δοκιμάζει την εφαρμογή, για να είναι σίγουρος ότι η αλλαγές που έχουν γίνει είναι σωστές και λειτουργούν όπως θέλει.

Κεφάλαιο 6ο Υλοποίηση Έργου σε Unity

Και εδώ πριν την έναρξη της δημιουργίας του κάθε έργου, έγινε μα έρευνα για τα asset που θα χρησιμοποιηθούν, πάλι προσπαθώντας να είναι δωρεάν. Τα asset που βρέθηκαν στην Unity ενώ σαν όγκος ήταν ικανοποιητικός και τα δωρεάν ήταν αρκετά λειτουργικά, η ποιότητα τους ήταν χαμηλή (Low-Poly), με συνέπεια το έργο να μοιάζει αρκετά «παιδικό». Για όλα τα αντικείμενα της εφαρμογής έχουν δημιουργηθεί prefabs και βρίσκονται στον αντίστοιχο φάκελο.

Το project στην Unity ξεκίνησε με την έκδοση 2022.3 και άλλαξε στην έκδοση 6.0.5 που ήταν και πιο σταθερή. Υπήρχε και η έκδοση 6.1, καθώς όμως μόλις είχε κυκλοφορήσει, αποφάσισα να μην την χρησιμοποιήσω, αφού δεν ήξερα πόσο σταθερή είναι.

6.1 Assets

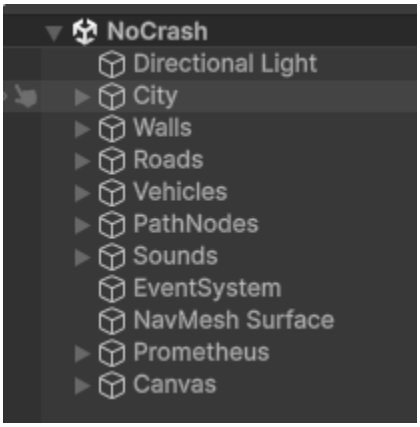
Ο παρακάτω πίνακας παρουσιάζει τα assets που χρησιμοποιήθηκαν στην εφαρμογή. Όλα τα assets αποκτήθηκαν δωρεάν από το Unity Asset Store ή από διαδικτυακά αποθετήρια.

Όνομα	Πηγή	Περιγραφή
SimplePoly City	Unity Asset Store	Κτίρια, Οχήματα και Δρόμοι.
Prometeo	Unity Asset Store	Όχημα του παίκτη και controller
Skybox Cubemap Extended	Unity Asset Store	Skybox (animated)

2) Τα Assets που χρησιμοποιήθηκαν στην εφαρμογή.

6.2 Οργάνωση Project

Όπως φαίνεται και στην εικόνα 5.2 , η οργάνωση των αντικειμένων στην κεντρική σκηνή της εφαρμογής έγινε με τέτοιο τρόπο ώστε να είναι αρκετά ξεκάθαρο. Τα γονικά αντικείμενα City, Walls και Roads περιέχουν τα επιμέρους αντικείμενα για τα κτίρια, την εξωτερική περίφραξη και τους δρόμους που απαρτίζουν την πόλη. Το γονικό αντικείμενο Vehicles περιλαμβάνει όλα τα οχήματα που κινούνται στους δρόμους, ενώ το PathNodes περιέχει όλους τους κόμβους που αποτελούν το δίκτυο των διαθέσιμων διαδρομών που μπορούν να ακολουθήσουν τα οχήματα.



6.1) Οργάνωση Αντικειμένων Σκηνής Unity

Όλα τα αντικείμενα της σκηνής που παραμένουν σταθερά έχουν δηλωθεί ως static, ώστε να ενισχυθεί η απόδοση της εφαρμογής. Επιπλέον, χρησιμοποιείται NavMesh, το οποίο έχει γίνει bake ώστε να περιλαμβάνει αποκλειστικά μόνο τους δρόμους της πόλης.

6.3 Δημιουργία Κόσμου

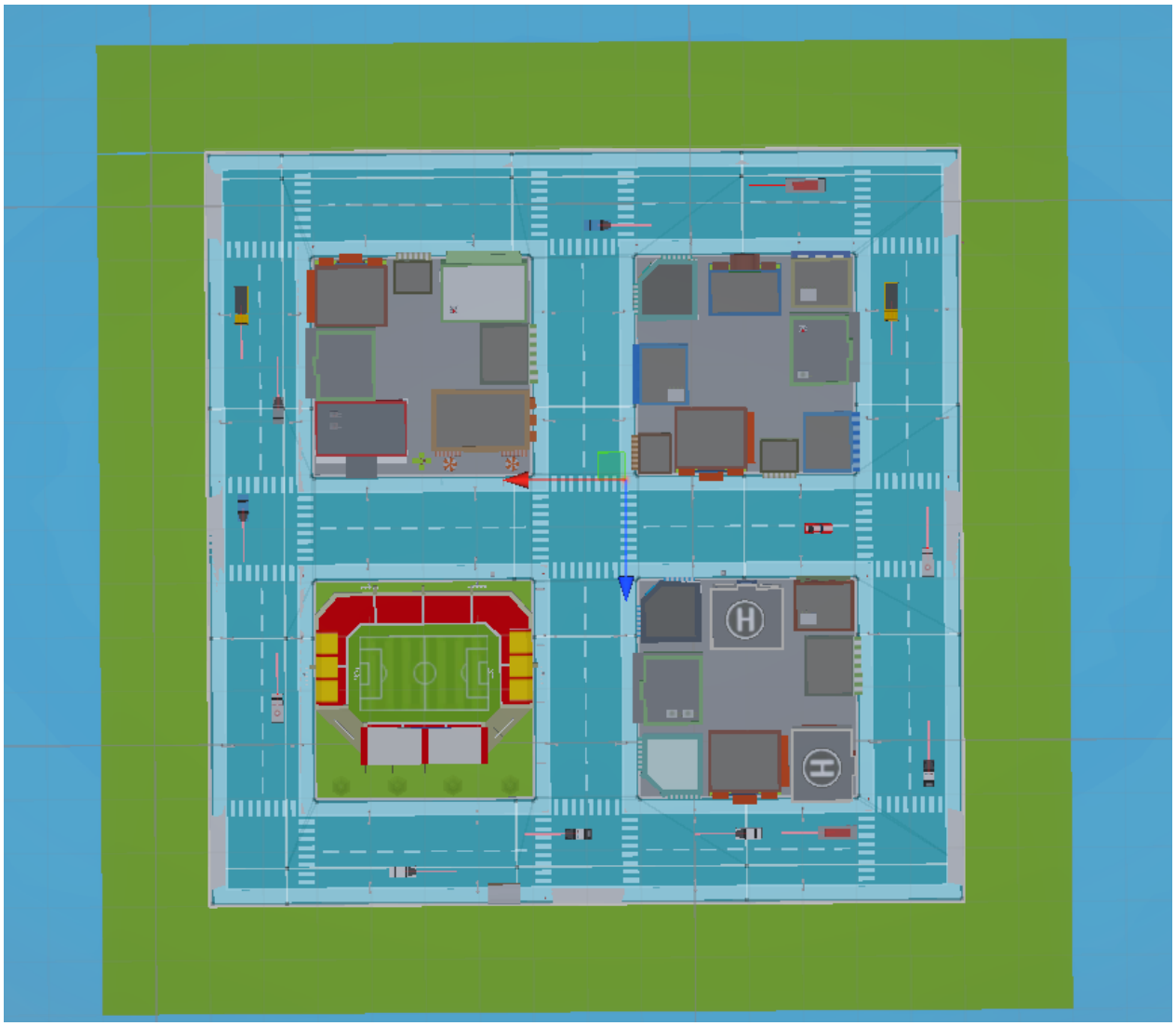
Η δημιουργία του κόσμου ήταν και στην Unity το δυσκολότερο και πιο απαιτητικό κομμάτι της υλοποίησης. Χωρίς κάποιον παρόμοιο τρόπο με τα PCG της Unreal, και μετά από αρκετό ψάξιμο, κατέληξα σε 2 τρόπους που μπορούσα να εφαρμόσω.

Ο πρώτος και ο πιο χρονοβόρος ήταν να τον φτιάξω χειροκίνητα. Έχοντας τα prefab και έχοντας το προσχέδιο στο μυαλό μου, έκανα Drag and Drop ότι χρειαζόμουν στην σκηνή και το τοποθετούσα στην σωστή θέση.

Ξεκίνησα πρώτα από τους δρόμους και το σταυροδρόμι στην μέση του χάρτη, και συνέχισα με τις ευθείες που φεύγαν από αυτό. Όταν έγιναν ικανοποιητικά μεγάλες για εμένα, πρόσθεσα μια τριπλή διασταύρωση και ξανά ευθείες ξεκινώντας από την κάθε πλευρά της διασταύρωσης. Τέλος, ένωσα τους 2 δρόμους με ένα prefab στροφής, φτιάχνοντας έτσι τους δρόμους, με σχήμα ένα τετράγωνο εξωτερικά και ένας σταυρός στο κέντρο του, που επεκτεινόταν από την μέση κάθε πλευράς του τετραγώνου έως την απέναντι.

Στα 4 κενά σημεία ανάμεσα από τους δρόμους, πρόσθεσα από ένα 3D Object “Plane” και επάνω τοποθέτησα πάλι χειροκίνητα τα κτίρια, ενώ στα πεζοδρόμια τοποθέτησα διάφορα αντικείμενα δρόμου, όπως φώτα και πυροσβεστικούς κρουνούς. Έτσι είχα έτοιμο το βασικό layout της πόλης μου.

Γύρω από τον εξωτερικό δρόμο, τοποθέτησα ακόμα 4 plane, στα οποία θα τοποθετούσα δέντρα σαν τα όρια του χάρτη.

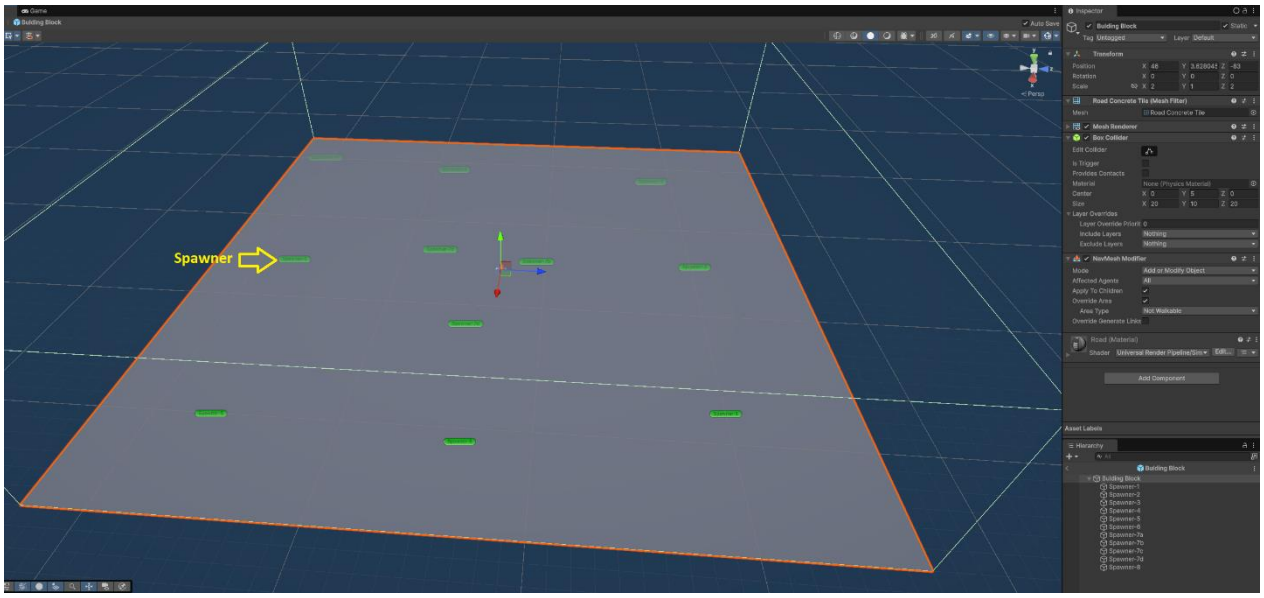


6.2) Top-Down View της πόλης

Σε αυτό το σημείο όμως η δημιουργία της πόλης κόλλησε, αφού δεν είχα τρόπο να τοποθετήσω τα δέντρα στο εξωτερικό σημείο του χάρτη. Χειροκίνητα θα έπαιρνε πάρα πολύ χρόνο, και σίγουρα θα υπήρχε καλύτερος τρόπος. Έτσι στράφηκα στην δημιουργία script. Μετά από πολλές δοκιμές, κατάφερα να δημιουργήσω το BoundsScript.cs, που μου επέτρεπε να δίνω από τον Inspector 4 GameObjects και 1 Prefab που ήθελα. Το script διάβαζε το μέγεθος και την τοποθεσία των 4^{ων} GameObject και στην επιφάνεια του τοποθετούσε το prefab όσες φορές ήθελα. Αναβάθμισα το script ώστε στον Inspector να υπάρχει ένα slider, το οποίο έθετε την τιμή για το πόσα prefabs ήθελα να τοποθετηθούν. Δεν χρειαζόταν έτσι να αλλάξω τιμή κατευθείαν στο script.

Ο δεύτερος τρόπος απλούστευσε λίγο την διαδικασία. Πάλι τοποθέτησα τους δρόμους χειροκίνητα και τα planes ανάμεσα στους δρόμους. Αυτή την φορά δημιούργησα ένα prefab για το εσωτερικό plane με διάφορα σημεία (Spawners). Αφού τα τοποθέτησα στα σωστά σημεία, ετοίμασα ένα script (Spawner.cs) με το οποίο διάβαζα την τοποθεσία κάθε σημείου spawner, και τοποθετώντας 3 prefabs σαν δεδομένα για το script στον Inspector, το script διάλεγε τυχαία ένα από αυτά με την μέθοδο SpawnRandom(), και το πρόσθετε στην σκηνή στο εκάστοτε σημείο.

```
public void SpawnRandom()
    {
        if (prefabs == null || prefabs.Count == 0)
            return;

        int index = Random.Range(0, prefabs.Count);
        GameObject prefabToSpawn = Instantiate(prefabs[index], transform.position, transform.rotation);
        //Set the parent of the spawned object

        prefabToSpawn.transform.SetParent(transform);
    }
```



6.3)Prefab Εσωτερικού plane με Spawners

Με την τοποθέτηση των σωστών spawner στην σκηνή, το Spawner.cs μπορεί να επαναχρησιμοποιηθεί για άλλα αντικείμενα, όπως αυτοκίνητα και props.

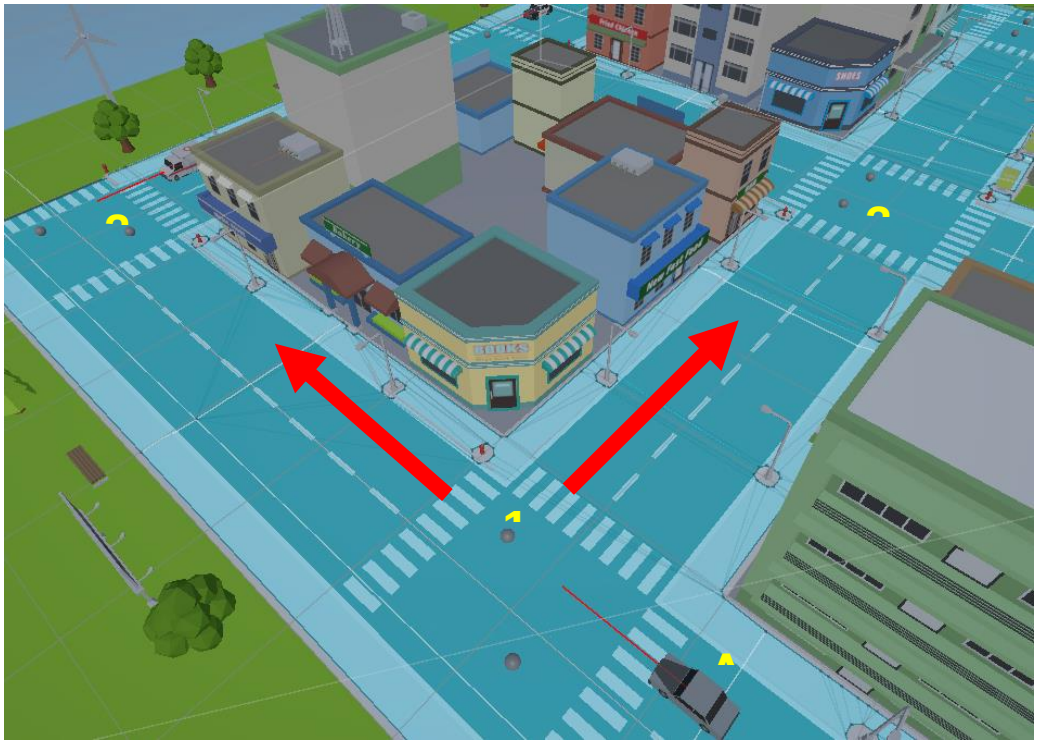
Για τα δέντρα γύρω από την πόλη χρησιμοποίησα το ίδιο script BoundsScript.cs.

6.4 Paths

Το αντικείμενο PathNode αντιπροσωπεύει έναν κόμβο, ο οποίος αποτελεί τμήμα ενός δικτύου κόμβων που περιγράφουν τις διαθέσιμες διαδρομές στους δρόμους της πόλης. Ο mesh renderer του αντικειμένου έχει απενεργοποιηθεί κατά την εκτέλεση της εφαρμογής και ενεργοποιείται μόνο στον editor, ώστε να είναι πιο εύκολη η μεταβολή της θέσης του. Το αντικείμενο περιέχει το script PathNode.cs, το οποίο διατηρεί μια δυναμική λίστα με nodes που αντιπροσωπεύουν τους διαθέσιμους πιθανούς επόμενους κόμβους.

[SerializeField] private List<PathNode> pathNodes = new List<PathNode>();

Η Εικόνα 5.3 παρουσιάζει ένα σενάριο κατά το οποίο το όχημα Α κινείται προς τον κόμβο 1. Η λίστα pathNodes του κόμβου 1 περιέχει τους κόμβους 2 και 3. Όταν το όχημα φτάσει στον προορισμό του, τότε επιλέγεται τυχαία ο επόμενος προορισμός με βάση το περιεχόμενο της λίστας (δηλαδή, κόμβοι 2 ή 3). Με αυτόν τον τρόπο μπορούν να δημιουργηθούν σύνθετες ή απλές διαδρομές, ανάλογα με το αν ο δρόμος αποτελεί σταυροδρόμι ή όχι.



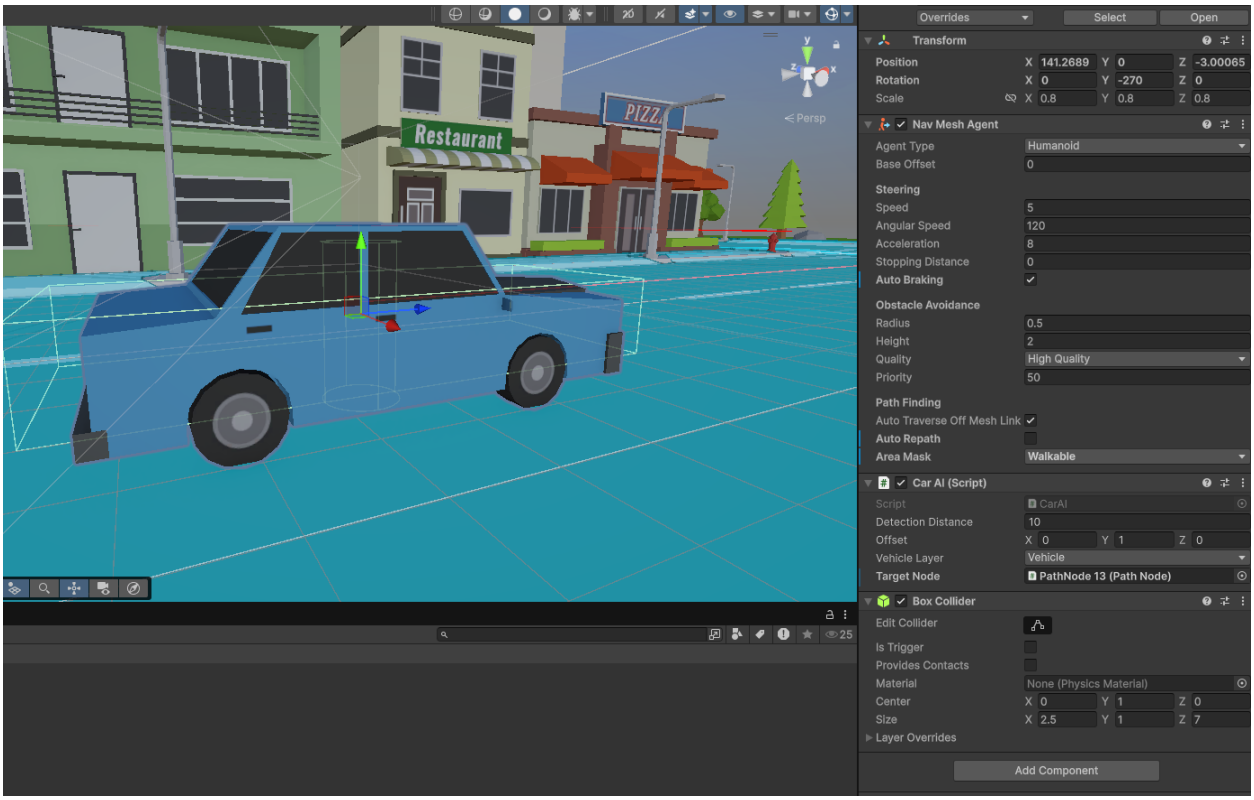
6.4)Σενάριο Λειτουργίας Κόμβων (PathNodes)

Η επιλογή του επόμενου κόμβου πραγματοποιείται καλώντας τη μέθοδο GetNextPathNode:

```
// Returns a random PathNode from the pathNodes list
public PathNode GetNextPathNode() {
    if (pathNodes == null || pathNodes.Count == 0)
        return null;

    // Ensure the index is within the bounds of the pathNodes list
    int index = Random.Range(0, pathNodes.Count);

    // Return the position of the randomly selected PathNode
    return pathNodes[index];
}
```

6.5 Vehicle

Το αντικείμενο vehicle αντιπροσωπεύει ένα όχημα στην εφαρμογή. Εμπεριέχει το components NavMeshAgent, BoxCollider καθώς και το script carAI.cs (Εικόνα 5.5).



6.5) Το Όχημα με τα Components από τα οποία απαρτίζεται

Το component NavMesh επιτρέπει στο όχημα να κινείται στις περιοχές που έχουν χαρακτηριστεί ως Walkable κατά τη διαδικασία του bake. Κάθε όχημα κατά την εκκίνηση της εφαρμογής ορίζει τον αρχικό του προορισμό, που αποτελεί ένα PathNode, μέσω της κλήσης της μεθόδου SetDestination του component NavMeshAgent.

```
void Start() {
    // check if navmesh agent is attached and then assign it
    if (GetComponent<NavMeshAgent>() == null) {
        Debug.LogError("NavMeshAgent component is missing on this GameObject.");
        return;
    } else {
        navMeshAgent = GetComponent<NavMeshAgent>();
        navMeshAgent.isStopped = false;
```

```
        if (targetNode == null) {
            Debug.LogError("Missing starting PathNode!.");
            return;
        } else {
            // Set the initial destination to the target node's position
            navMeshAgent.SetDestination(targetNode.transform.position);
        }
    }
}
```

Κατά την αρχικοποίηση του αντικειμένου πραγματοποιούνται έλεγχοι για το αν είναι διαθέσιμος ο navMeshAgent, καθώς και για το αν έχει οριστεί αρχικός κόμβος (PathNode). Σε αντίθετη περίπτωση εμφανίζεται το αντίστοιχο μήνυμα λάθους. Στον editor, μέσω των χαρακτηριστικών Speed, Steering, Acceleration κλπ. του navMeshAgent, μπορούν να οριστούν τα ιδιαίτερα χαρακτηριστικά κάθε οχήματος, καθιστώντας για παράδειγμα το λεωφορείο πιο αργό από ένα επιβατικό όχημα. Υπάρχουν συνολικά 12 οχήματα που κινούνται στην πόλη, τα οποία διαφοροποιούνται ως προς το μοντέλο τους (αστυνομία, φορτηγό, αγωνιστικό, λεωφορείο, ασθενοφόρο κλπ.) και τα προαναφερθέντα επιμέρους χαρακτηριστικά.

Όλα τα οχήματα της εφαρμογής περιλαμβάνουν επίσης box collider, οι διαστάσεις του οποίου έχουν οριστεί έτσι ώστε να καλύπτουν τα όρια του κάθε ξεχωριστού μοντέλου από τους διαθέσιμους τύπους οχημάτων. Όλα τα οχήματα ανήκουν επίσης στο Layer Vehicle, ώστε να μπορεί να γίνει η αναγνώρισή τους κατά τη διαδικασία της πρόσκρουσης και αποφυγής. Ο έλεγχος πραγματοποιείται με κλήση της μεθόδου CheckForVehicleInFront(), κατά την οποία μέσω raycast ελέγχεται εάν υπάρχει μπροστά από το όχημα άλλο αντικείμενο που ανήκει στο Layer Vehicle. Ο έλεγχος πραγματοποιείται τόσο για τα οχήματα που κινούνται μέσω navMeshAgent όσο και για το όχημα του παίκτη.

```
bool CheckForVehicleInFront() {
    Ray ray = new Ray(transform.position + offset, transform.forward);
    RaycastHit hit;

    // Perform the raycast using the vehicleLayer mask
    if (Physics.Raycast(ray, out hit, detectionDistance, vehicleLayer)) {
        return true; // Vehicle detected
    }
```

```
        return false; // No vehicle detected
    }
```

Ο κεντρικός βρόχος επανάληψης του κάθε οχήματος πραγματοποιεί έλεγχο για το αν υπάρχει εμπόδιο μπροστά από το όχημα και το ακινητοποιεί εάν αυτό ισχύει. Εάν ο δρόμος δεν έχει εμπόδιο, πραγματοποιεί έλεγχο για το αν το όχημα έχει φτάσει στον προορισμό του και, αν αυτό ισχύει μέσω του ελέγχου του υπολοίπου της απόστασης του οχήματος από τον προορισμό, καλεί τη μέθοδο GetNextPathNode του τρέχοντος κόμβου για να αναζητήσει τον επόμενο στόχο-προορισμό.

```
    void Update() {

        if(CheckForVehicleInFront()) {
            // if a vehicle is detected in front, stop the nav mesh
            navMeshAgent.isStopped = true;
            //Debug.Log("Vehicle detected in front, stopping the NavMeshAgent.");
        }
        else {
            // move nav agent towards the target node position
            if (targetNode != null) {
                if (navMeshAgent.isStopped == true) {
                    navMeshAgent.isStopped = false;
                }
                navMeshAgent.SetDestination(targetNode.transform.position);
            }

            // check if the nav mesh agent has reached the target node
            if (navMeshAgent.remainingDistance <= navMeshAgent.stoppingDistance) {
                //Debug.Log("Reached target node: " + targetNode.name);
                // Get a new random position from the target node's path nodes
```

```
            PathNode newTargetNode = targetNode.GetNextPathNode();

            if (newTargetNode != null) {
                targetNode = newTargetNode; // Update the target node
                navMeshAgent.SetDestination(targetNode.transform.position);
                //Debug.Log("Moving to PathNode: " + targetNode.name);
            } else {
                Debug.LogWarning("No valid PathNode found.");
            }
        }
    }
}
```

Τέλος, για να γίνει πιο εύκολη η διαδικασία του debugging, μέσω της OnDrawGizmos σχεδιάζεται το ray που χρησιμοποιείται κατά το raycast για εύρεση εμποδίων μπροστά από το όχημα.

```
void OnDrawGizmos() {
    Gizmos.color = Color.red;
    Gizmos.DrawRay(transform.position + offset, transform.forward * detectionDistance);
}
```

6.6 Διεπαφή Χρήστη

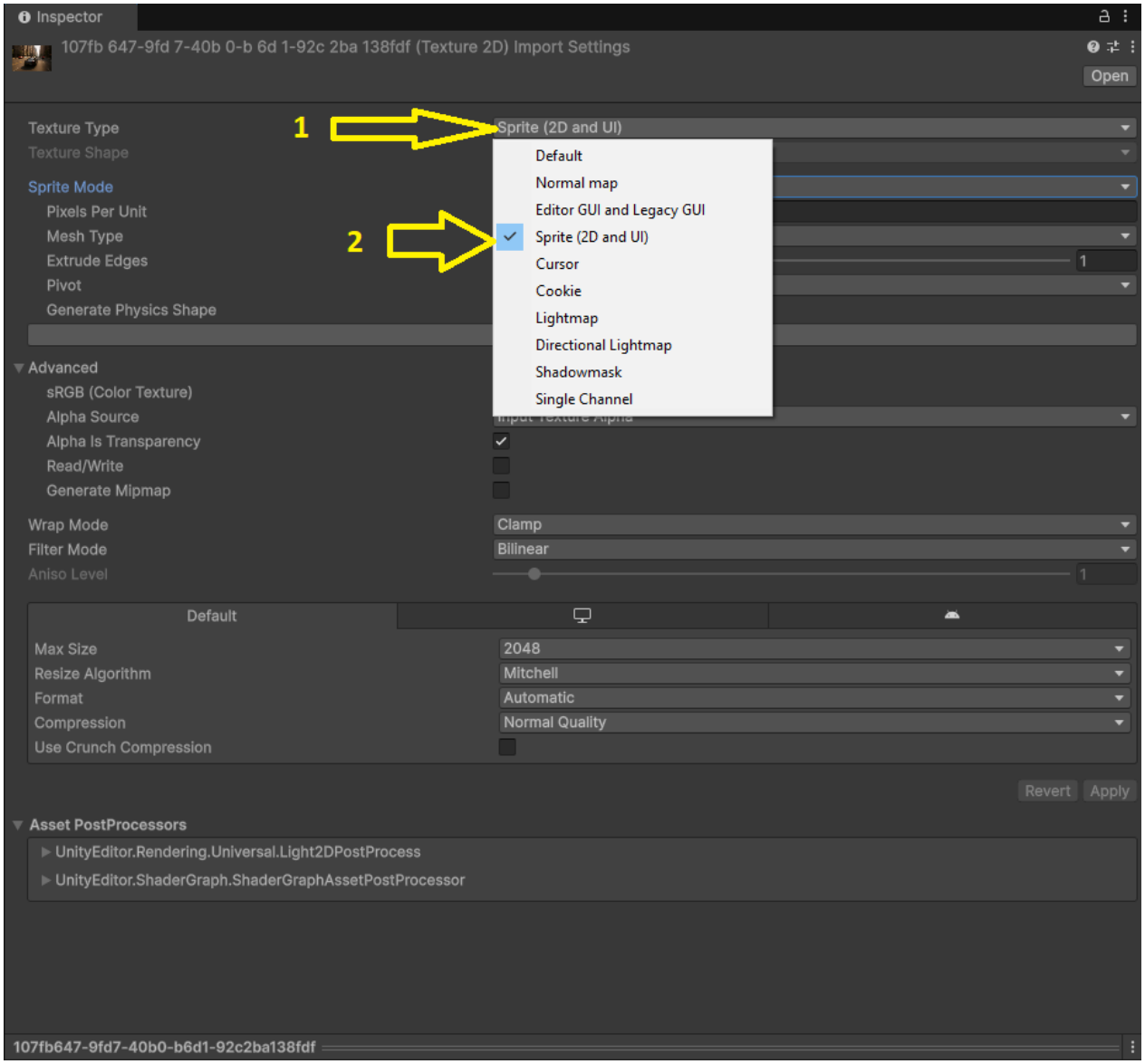
Για την εφαρμογή υλοποιήθηκαν διάφορα μενού και μια οθόνη ενδείξεων (Heads-Up-Display – HUD) για την καλύτερη αλληλεπίδραση του χρήστη με αυτήν. Το παιχνίδι ξεκινάει με το αρχικό μενού της εικόνας 5.4.

6.6) Αρχικό Μενού

Η δημιουργία του έγινε σε ένα ξεχωριστό Scene το "Start Menu", ενώ η διαδικασία ήταν αρκετά απλή. Ανοίγοντας το άδειο Scene, το πρώτο πράγμα ήταν να φτιάξουμε το Layout και την εμφάνιση του μενού. Έτσι προστέθηκε ένα UI αντικείμενο Canvas το οποίο θα περιείχε όλα τα επιμέρους αντικείμενα που θα χρειαζόμασταν.

Μέσα στο Canvas, προστέθηκε και ένα Image, το οποίο λειτουργεί σαν background στο μενού μας, κάνοντας το πιο όμορφο για τον χρήστη. Η εικόνα πάρθηκε από το Google Images και για την χρήση της έπρεπε να γίνει import. Για να γίνει import στο project, μπορεί να γίνει με δεξί κλικ στο Content Browser (όντας στον φάκελο που θέλουμε να αποθηκευτεί), πατώντας την επιλογή "Import New Asset" και επιλέγοντας τις εικόνες που έχουμε αποθηκεύσει στον υπολογιστή μας και θέλουμε να εισάγουμε.

Αμέσως θα εμφανιστούν στο project μας, αλλά δεν είναι έτοιμα για χρήση ακόμη. Οι εικόνες χρειάζεται να μετατραπούν σε αρχείο τύπο Sprite, ένα ειδικού τύπο asset της Unity που περιέχει πληροφορίες για την χρήση του σε rendering, animation και άλλα. Για να γίνει αυτή η μετατροπή πρέπει να ανοίξουμε της πληροφορίες της εικόνας στον Inspector κάνοντας κλικ στην εικόνα από το Content Browser. Έπειτα αλλάζουμε το Texture Type από Default σε Sprite (2d and UI), όπως φαίνεται στην εικόνα 5.5, και η εικόνα είναι έτοιμη για χρήση.



6.7) Μετατροπή εικόνας σε Sprite.

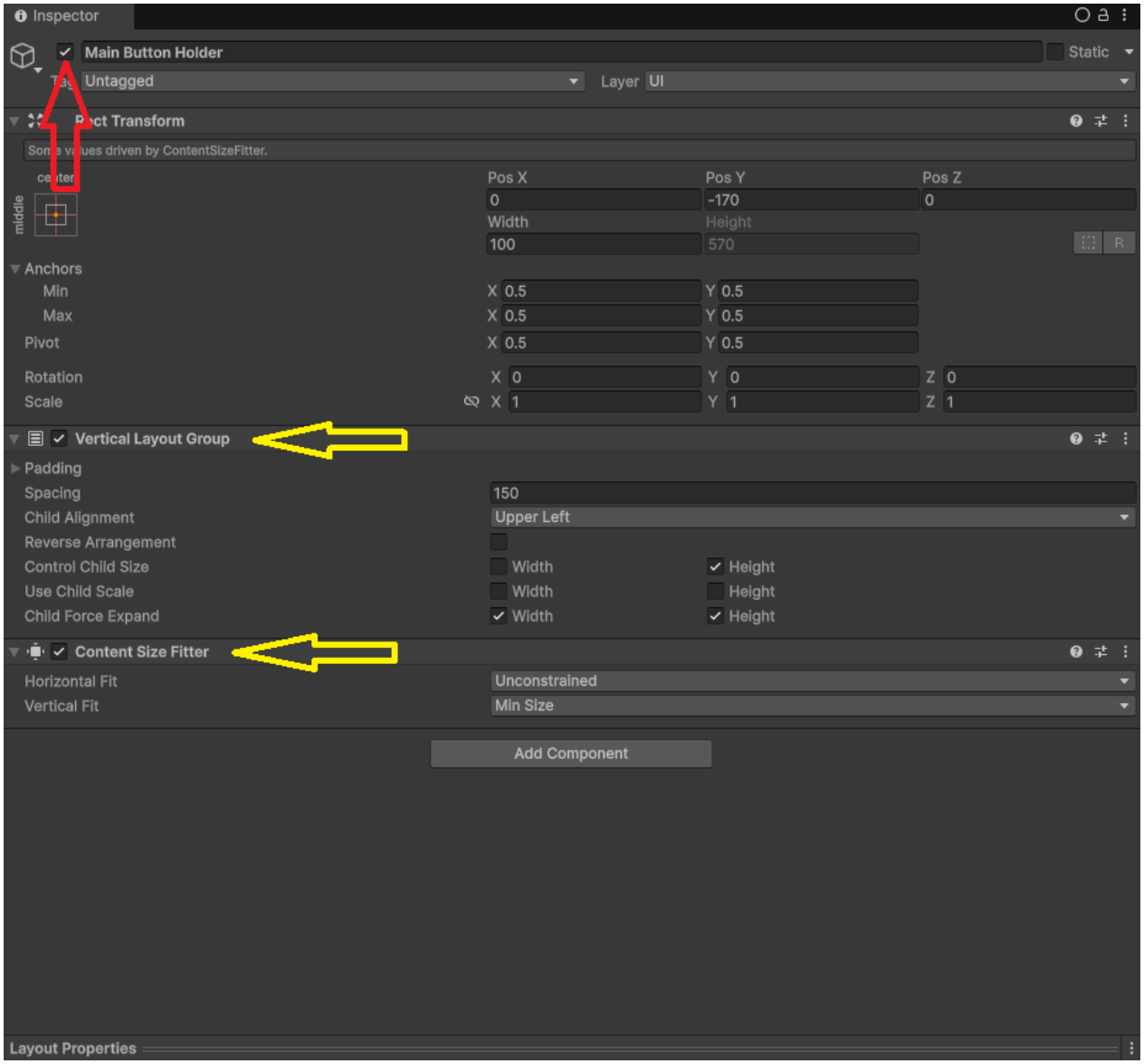
Αφού λοιπόν έγινε αυτή η μετατροπή, η εικόνα έγινε Drag and Drop στο Source Image πεδίο του Image αντικειμένου που προστέθηκε στην σκηνή και επεξεργάστηκε κατάλληλα για να καλύπτει όλο το Canvas.

Στο αρχικό μενού υπάρχουν 3 βασικά αντικείμενα, το κείμενο στην κορυφή, το βασικό μενού και το μενού ρυθμίσεων που ανοίγει με το πάτημα του κουμπιού Options. Το κείμενο είναι ένα απλό Text Object παραμετροποιημένο με το κατάλληλο Font, Size και τοποθετημένο στην κατάλληλη θέση.

Για τα 2 μενού θεώρησα σωστό να ήταν 2 ξεχωριστά GameObjects. Για αυτό τον λόγο δημιουργήθηκαν 2 κενά GameObjects, το Main Button Holder και το Options Button Holder.

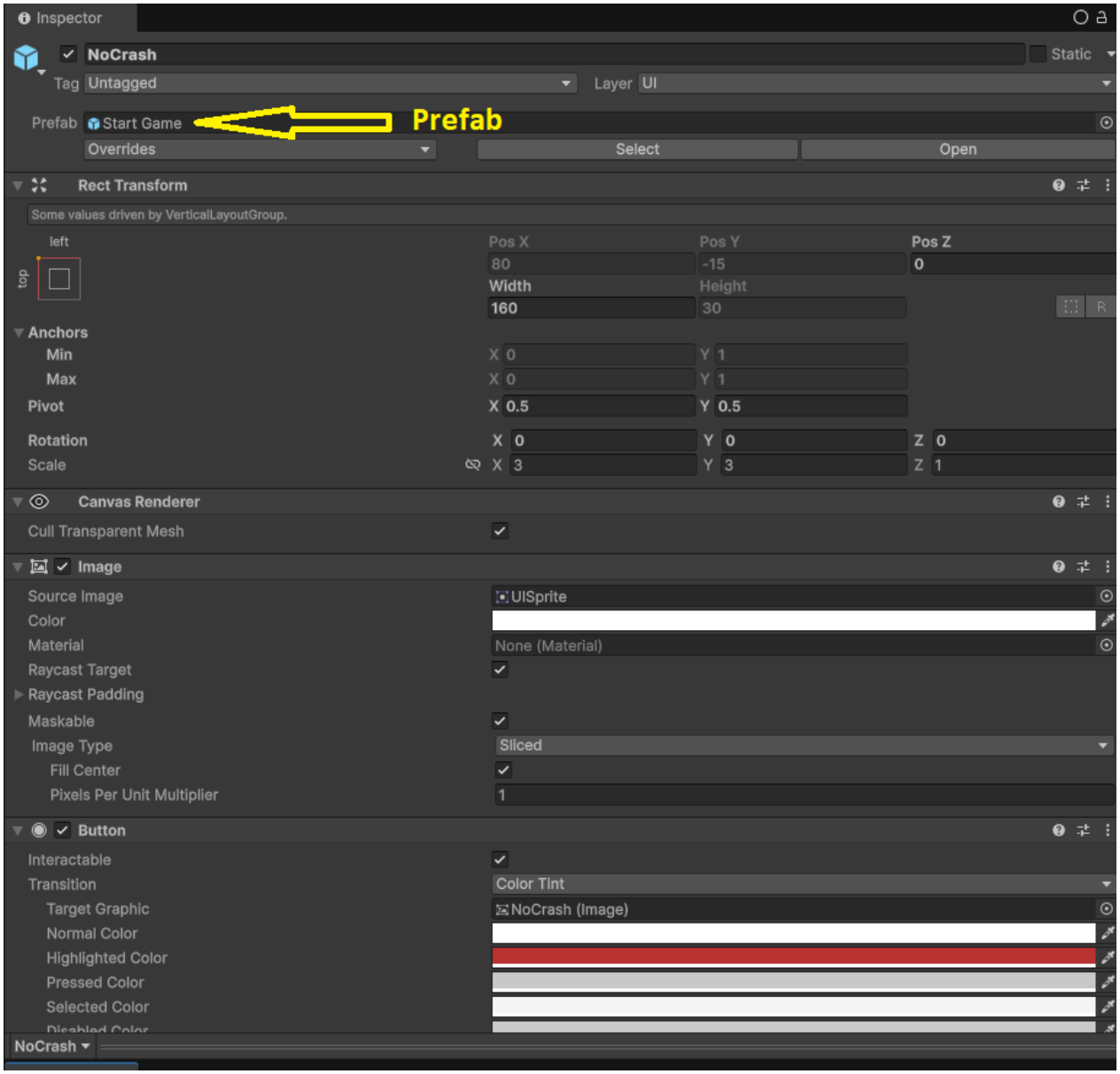
Το βασικό μενού περιέχει 4 κουμπιά, το No Crash Game, το Survival, το Options και το Exit Game. Προστέθηκαν σαν children του βασικού μενού. Πρώτα δημιούργησα ένα prefab κουμπιού με τις ρυθμίσεις που θέλησα, έχοντας έτσι ένα έτοιμο κουμπί απλά κάνοντας το duplicate όσες φορές και σε όποιο σημείο

του project χρειάστηκα. Για να έχουν τα κουμπιά ίδιες ιδιότητες, όπως την τοποθεσία που βρίσκονται, το μέγεθος τους και την απόσταση μεταξύ τους, στο Main Button Holder, πρόσθεσα το Vertical Layout Group και το Content Size Filter. Βάζοντας το πρώτο, τοποθέτησα όλα τα αντικείμενα μέσα στο Holder σε ένα κατακόρυφο γκρουπ από το οποίο μπορούσα να διαχειριστώ ρυθμίσεις όπως το Spacing μεταξύ των αντικειμένων μέσα σε αυτό. Με το δεύτερο, μπορούσα να διαχειριστώ τον χώρο που θα πιάνουν τα κουμπιά μου μέσα σε αυτό το Holder.



6.8) Vertical Layout Group και Content Size Filter

Τέλος, προστέθηκαν το προηγουμένως δημιουργημένο prefab κουμπιού 4 φορές, σιγουρεύοντας το από το Inspector σε κάθε ένα, όπως φαίνεται στην εικόνα 5.7.



6.9) Όνομα Prefab που έπρεπε να έχουν όλα τα κουμπιά μου.

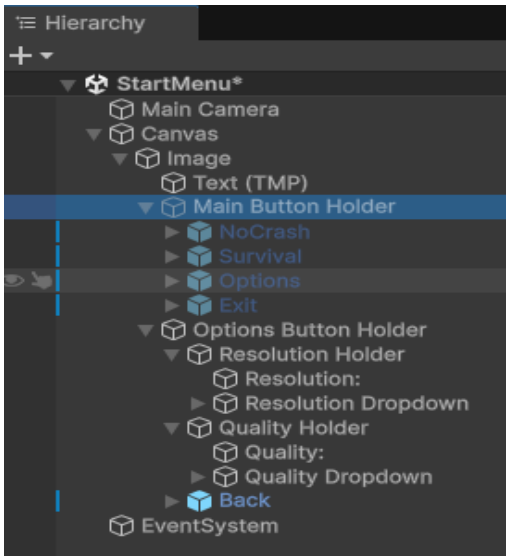
Με το να έχουν όλα τα κουμπιά μου την συγκεκριμένη ρύθμιση, μπορούσα κάνοντας αλλαγές σε ένα, να εφαρμόσω τις ίδιες ακριβώς αλλαγές και σε όλα τα υπόλοιπα. Αφού έκανα τις αλλαγές που ήθελα σε ένα, απλά πατώντας στο κουμπί Overrides κάτω από το όνομα του Prefab και Apply All στο αναδυόμενο, οι αλλαγές αμέσως θα εφαρμόζονταν σε όλα τα αντικείμενα του συγκεκριμένου prefab.

Όταν ήμουν ικανοποιημένος με το βασικό μενού, προχώρησα στην δημιουργία του μενού ρυθμίσεων. Ακολούθησα την ίδια λογική με το Main Button Holder. Δημιούργησα ένα κενό GameObject με Vertical Layout Group και Content Size Filter. Αυτή την φορά όμως δεν χρειαζόμουν απλά κουμπιά. Έτσι πρόσθεσα ένα κουμπί για την επιστροφή στο βασικό μενού και 2 κενά GameObject που θα χρησιμοποιούσα σαν χώρο ομαδοποίησης για τα αντικείμενα της κάθε διαφορετικής ρύθμισης (Quality Holder, Resolution Holder).

Μέσα στο κάθε Holder υπάρχει ένα απλό Text για την επεξήγηση της ρύθμισης που αλλάζει το Dropdown που ακολουθεί, και το αντίστοιχο Dropdown με τις επιλογές του.

6.10) Μενού Ρυθμίσεων

Σε κάθε dropdown πρόσθεσα 3 επιλογές. 1280x720, 1920x1080 και 2560x1440 για το Resolution, Low, Medium και High για το Quality. Μια καλή πρακτική που εφάρμοσα ήταν να κρύβω το UI που δεν χρησιμοποιούσα. Για παράδειγμα, όταν έφτιαχνα το μενού ρυθμίσεων έκρυψα το βασικό μενού για να μην με μπερδεύει. Αυτό γίνεται πατώντας το κουτάκι δίπλα στο όνομα από το αντικείμενο που θες στον Inspector (κόκκινο βελάκι στην εικόνα 5.6). Έτσι είχα έτοιμο το UI του αρχικού μενού και έμενε η λογική.



6.11) Οργάνωση Αντικειμένων Αρχικού Μενού

Για την λογική δημιούργησα το Script “MainMenu” το οποίο πρόσθεσα στο EventSystem για να μπορούν τα υπόλοιπα GameObject να το διαβάσουν χωρίς να χρειάζεται να το προσθέσω σε όλα ή να δημιουργήσω πολλαπλά script.

Ξεκίνησα δημιουργώντας όλες τις μεθόδους για κάθε λειτουργία κάθε αντικειμένου. Έτσι δημιουργήθηκαν οι μέθοδοι:

- StartGame(): διαβάζει το όνομα του κουμπιού που την κάλεσε και κάνει μια σύγκριση. Αν το κουμπί ήταν το NoCrash, τότε ανοίγει το αντίστοιχο Scene. Διαφορετικά ανοίγει το Survival.

```
public void StartGame()
{
    GameObject selectedObject = EventSystem.current.currentSelectedGameObject;
    if(selectedObject.name == "NoCrash")
    {
        SceneManager.LoadScene("NoCrash");
    }
    else if(selectedObject.name == "Survival")
    {
        SceneManager.LoadScene("Survival");
    }
}
```

- LoadSettings(): Ανοίγει τις ρυθμίσεις κρύβοντας το Main Button Holder και εμφανίζοντας το Options Button Holder.

```
public void LoadSettings()
  {
    settingsPanel.SetActive(true);
    BackButton.Select();
    MainMenuPanel.SetActive(false);
  }
```

- QuitGame(): Κλείνει την εφαρμογή.

```
public void QuitGame()
```

```
{
    // Quit the application
    Application.Quit();
}
```

- ChangeQuality(): Διαβάζει την τιμή του Dropdown και ανάλογα με την integer τιμή που έχει αλλάζει την ποιότητα των γραφικών.

```
public void ChangeQuality(int value)
{
    // 0: Low, 1: Medium, 2: High
    switch (value)
    {
        case 0:
            QualitySettings.SetQualityLevel(0, true); // Low
            break;



        case 1:
            QualitySettings.SetQualityLevel(2, true); // Medium
            break;
        case 2:
            QualitySettings.SetQualityLevel(5, true); // High
            break;
        default:
            QualitySettings.SetQualityLevel(2, true); // Default to Medium
            break;
    }
    PlayerPrefs.SetInt("QualitySetting", value);
    PlayerPrefs.Save();
```

```
}
```

- ChangeResolution(): Διαβάζει την τιμή του Dropdown και ανάλογα με την integer τιμή που έχει αλλάζει το Resolution.

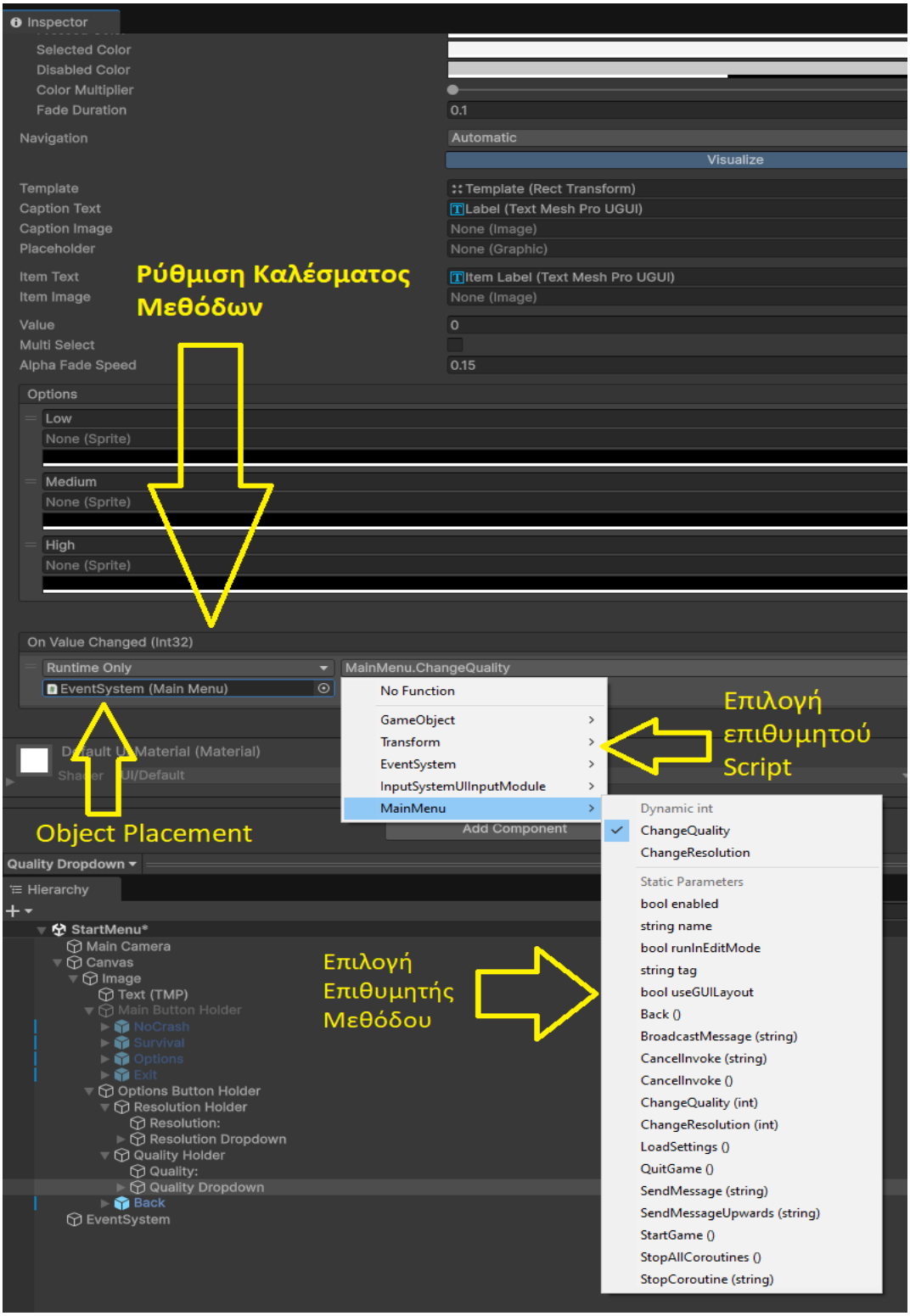
```
public void ChangeResolution(int value)
{
    int width, height;
    switch (value)
    {
        case 0:
            width = 1280;
            height = 720;
            break; // 720p
        case 1:
            width = 1920;
            height = 1080;
            break; // 1080p

        case 2:
            width = 2560;
            height = 1440;
            break; // 1440p
        default:
            width = 1920;
            height = 1080;
            break; // Default to 1080p
    }
    Screen.SetResolution(width, height, Screen.fullScreen);
    PlayerPrefs.SetInt("ResolutionSettings", value);
```

```
        PlayerPrefs.Save();
    }
```

- Back(): Εμφανίζει το Main Button Holder και κρύβει το Options Button Holder.

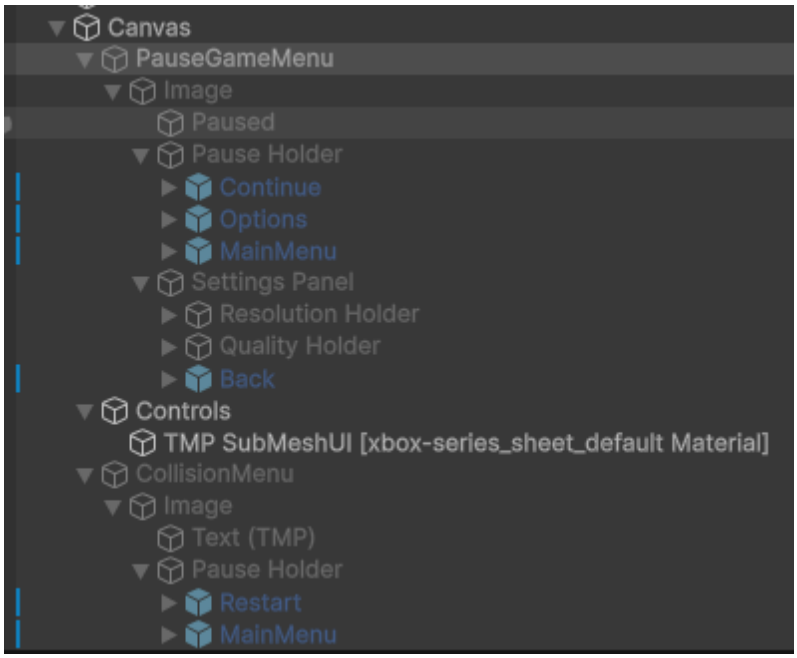
```
    public void Back()
    {
        settingsPanel.SetActive(false);
        MainMenuPanel.SetActive(true);
        NoCrashButton.Select(); // Set focus back to the Start button
    }
```

Όταν το script ήταν έτοιμο και προστέθηκε και στο EventSystem, έπρεπε να ρυθμίσω το κάθε κουμπί και dropdown για την μέθοδο που θα καλούσε. Για να γίνει αυτό έπρεπε να πάω στο Inspector για κάθε ένα αντικείμενο και να βρω την ρύθμιση On Click στα κουμπιά και On Value Changed() στα dropdown. Εκεί διάλεξα το EventSystem σαν Object και την αντίστοιχη μέθοδο.



6.12) Προσθήκη Μεθόδων στα Αντικείμενα

Στην εφαρμογή υπάρχουν ακόμα 2 μενού, το μενού παύσης (Pause Game Menu) και το μενού τρακαρίσματος (Collision Menu). Τα 2 αυτά μενού είναι παιδιά του ίδιου Canvas, ενώ μαζί με αυτά υπάρχει και ένα αντικείμενο που χρησιμοποιείται σαν HUD για την εμφάνιση των κουμπιών παιχνιδιού (Control Mapping) στην οθόνη του παίκτη.

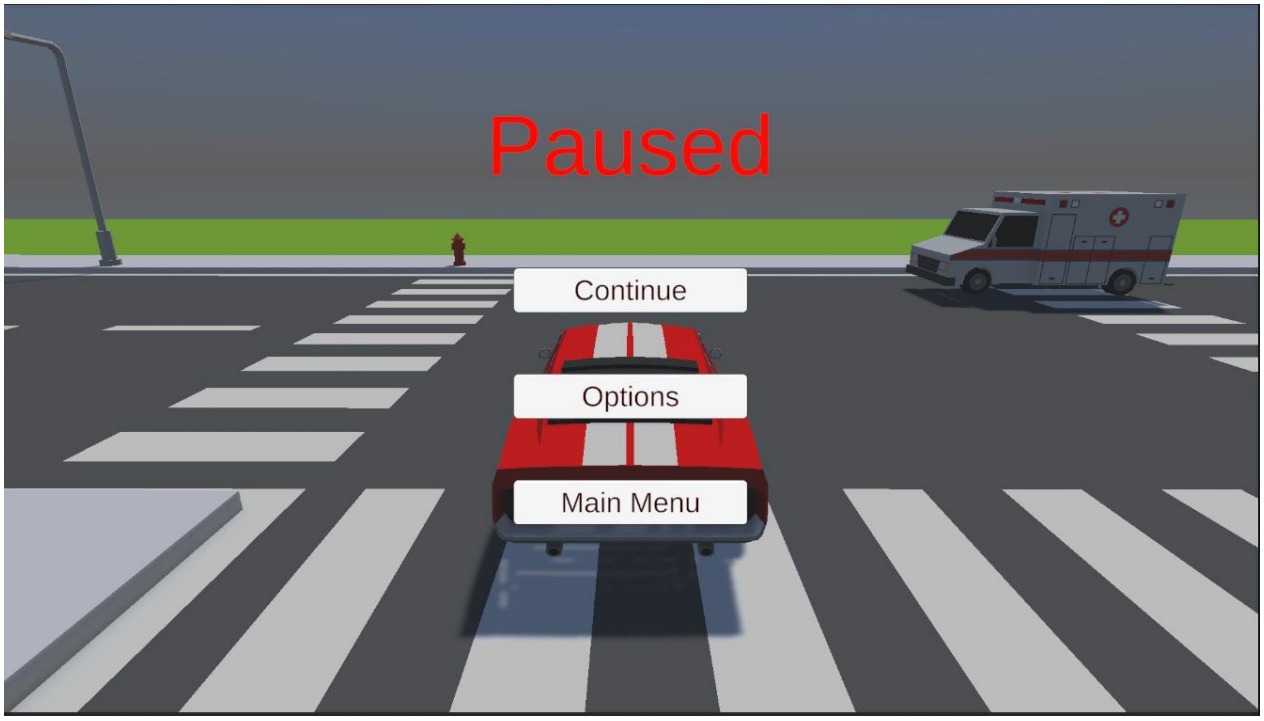


6.13) Οργάνωση Μενού Παιχνιδιού

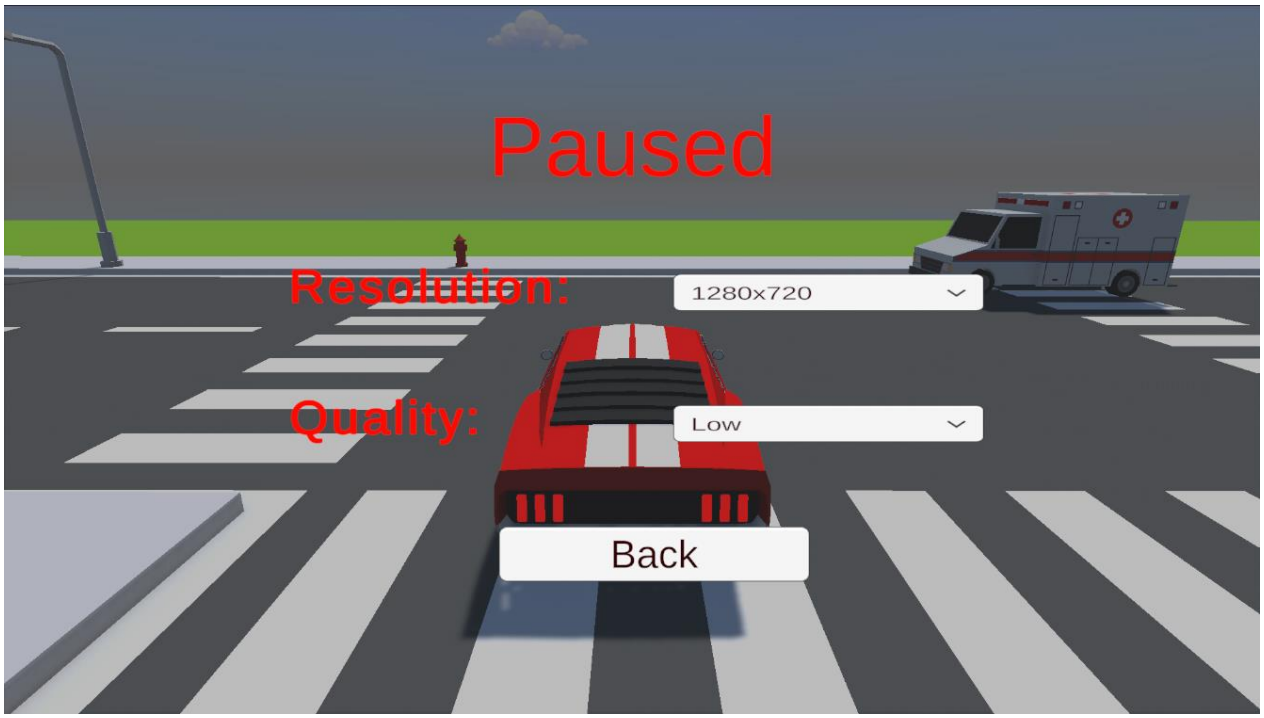
Τα 2 μενού Pause Game Menu και Collision Menu είναι χτισμένα με τον ίδιο τρόπο με το αρχικό μενού. Με την μόνη διαφορά ότι στο Image δεν πρόσθεσα εικόνα αλλά έκανα το χρώμα μαύρο και έριξα το Transparency του ώστε όταν είναι ενεργά, να μαυρίζει λίγο η εικόνα αλλά ακόμα να φαίνεται το παιχνίδι πίσω.

Το Pause Game Menu έχει τρείς επιλογές, Continue, Options και Main Menu, τα οποία συνεχίζουν το παιχνίδι από το σημείου έγινε το Pause, ανοίγει το μενού ρυθμίσεων (ίδιο με αυτό του αρχικού μενού) και επιστρέφει τον παίκτη στον αρχικό μενού αντίστοιχα.

Το Collision Menu έχει 2 επιλογές, Restart και Main Menu, τα οποία επανεκκινούν την σκηνή από την αρχή και γυρνάνε τον παίκτη στο αρχικό μενού αντίστοιχα. Ο τρόπος και ο κώδικας εμφάνισης αυτού του μενού θα αναλυθεί μαζί με το αντικείμενο του παίκτη.



6.14)Pause Game Menu



6.15)Pause Options Menu

6.16) Collision Menu

Ξανά αφού δημιούργησα οπτικά τα μενού μου, έπρεπε να γράψω και να προσθέσω την λογική σε κάθε κουμπί. Έτσι έφτιαξα το script PauseMenu με τις εξής μεθόδους:

- ResumeGame(): Κρύβει το Pause Game Menu και συνεχίζει το παιχνίδι.

```
public void ResumeGame()
    {
        isPaused = false;
        pauseMenuUI.SetActive(false); // Hide the pause menu UI
        buttonHUD.SetActive(true); // Show the button HUD
        Time.timeScale = 1f; // Resume the game
    }
```

- PauseGame(): Σταματάει το παιχνίδι και εμφανίζει το Pause Game Menu.

```
public void PauseGame()
    {
        isPaused = true;
        pauseMenuUI.SetActive(true); // Show the pause menu UI
        buttonHUD.SetActive(false); // Hide the button HUD
```

```
        Time.timeScale = 0f; // Pause the game
    }
```

- ReturnToMainMenu(): Επέστρεφε τον παίκτη στο αρχικό μενού.

```
    public void ReturnToMainMenu()
    {
        Time.timeScale = 1f; // Ensure the game is running at normal speed
        // Load the main menu scene
        SceneManager.LoadScene("StartMenu");
    }
```

- LoadSettings(): Έκρυβε το Pause Holder και εμφάνιζε το Settings Panel με τα κουμπιά ρυθμίσεων.

```
    public void LoadSettings()
    {
        settingsPanel.SetActive(true);
        BackButton.Select();
        pausePanel.SetActive(false);
    }
```

- Back(): Έκρυβε το Settings Panel και εμφάνιζε το Pause Holder με τα βασικά κουμπιά του Pause Game Menu.

```
    public void Back()
    {
        settingsPanel.SetActive(false);
        pausePanel.SetActive(true);
        ContinueButton.Select(); // Set focus back to the Start button
    }
```

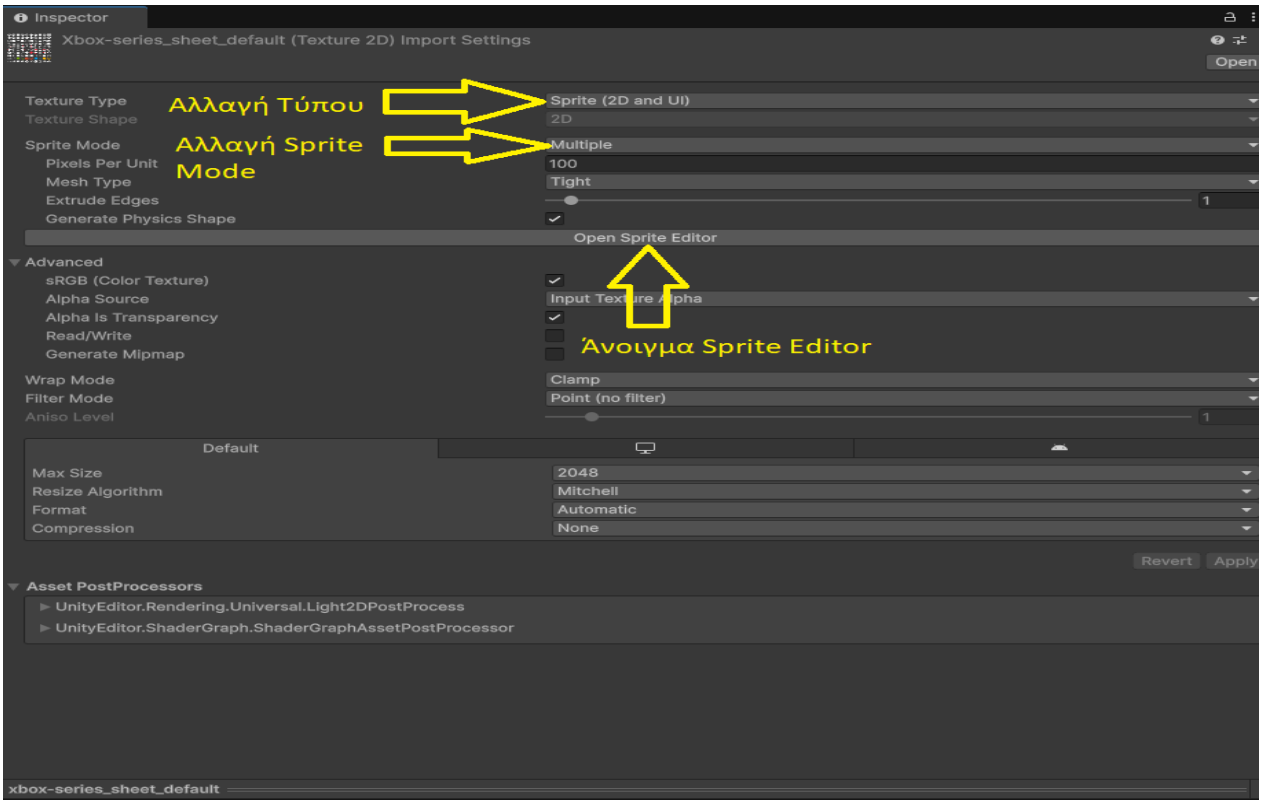
- Restart(): Επανεκκίνηση της σκηνής και του κόσμου.

```
    public void Restart()
    {
```

```
        Time.timeScale = 1f; // Ensure the game is running at normal speed

        SceneManager.LoadScene(SceneManager.GetActiveScene().name); // Reload the current scene

    }
```

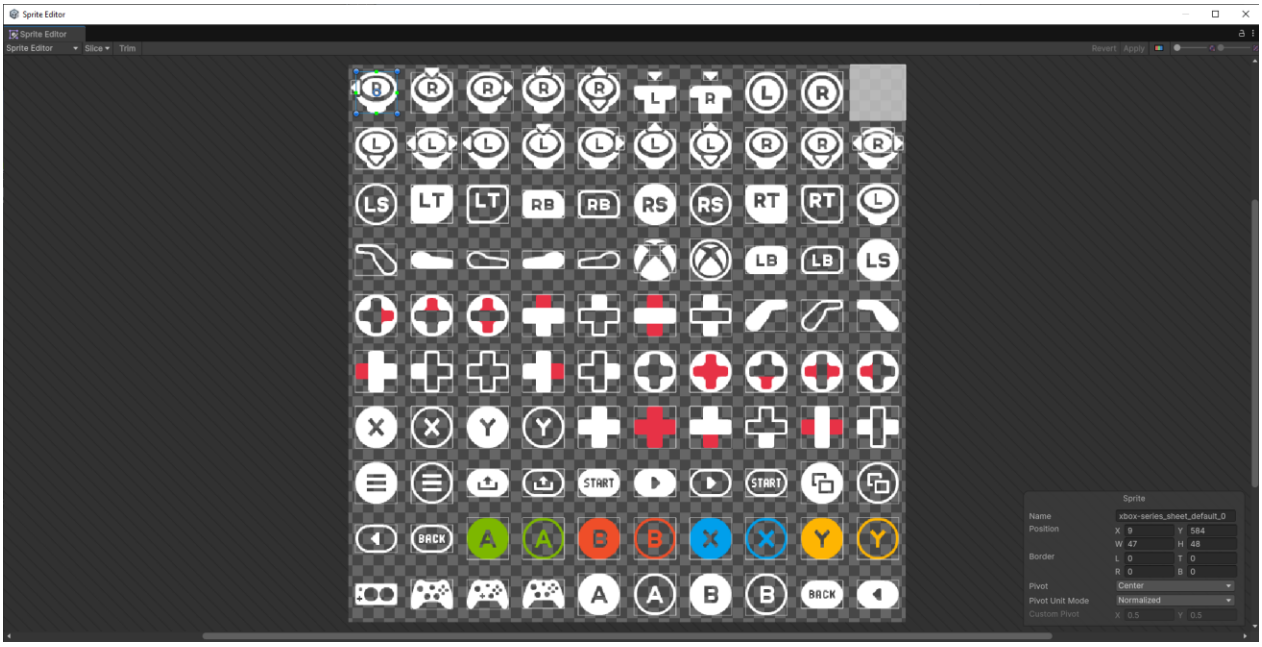
Εννοείται πως μετά έπρεπε να γίνει και η σωστή ανάθεση της κάθε μεθόδου στο σωστό κουμπί, όπως έγινε και με το αρχικό μενού.

Μέσα στο Canvas, πρόσθεσα και ένα ακόμα UI αντικείμενο, το Text – TextMeshPro (το Controls). Εκεί έγραψα όλα τα κουμπιά που θα χρειαστεί ο παίκτης. Στο HUD πρόσθεσα και Sprites, εικονοποιόντας τα κουμπιά του Gamepad. Πρώτα κατέβασα από το διαδίκτυο μια εικόνα. Υπάρχουν πολλές σελίδες με έχουν έτοιμες τέτοιες εικόνες που περιέχουν πολλά εικονίδια σε μια. Αφού βρήκα αυτή που μου άρεσε και την κατέβασα, την έκανα Import και την μετέτρεψα σε Sprite (2D and UI) όπως έκανα και με την εικόνα του αρχικού μενού, ενώ επίσης επέλεξα την επιλογή Multiple στο Sprite Mode.



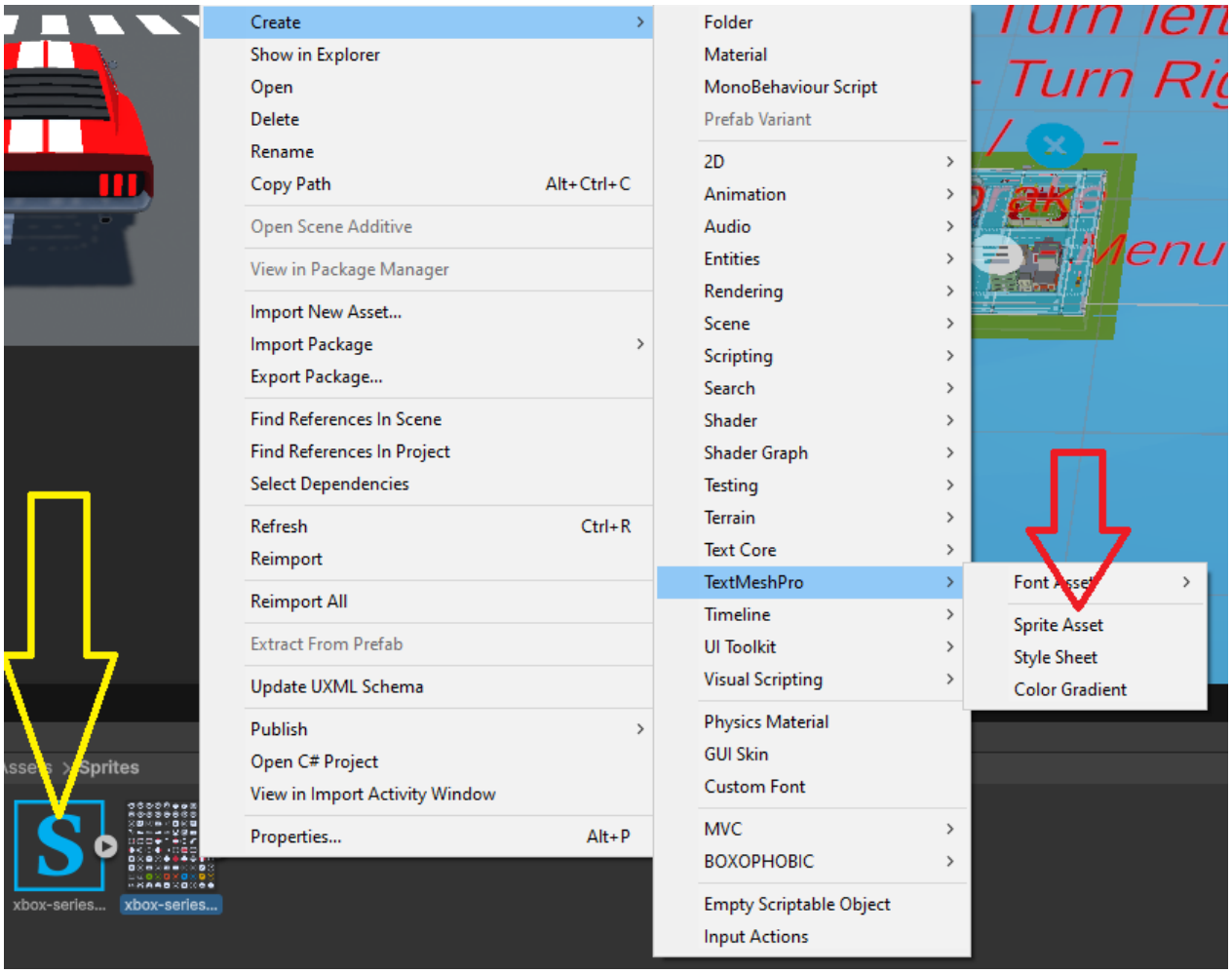
6.17)Ρυθμίσεις Εικόνας με Sprite

Μετά πατώντας το κουμπί Open Sprite Editor, άνοιξε ένα παράθυρο με την εικόνα, στην οποία έπρεπε να «κόψω» όλα τα διαφορετικά εικονίδια.



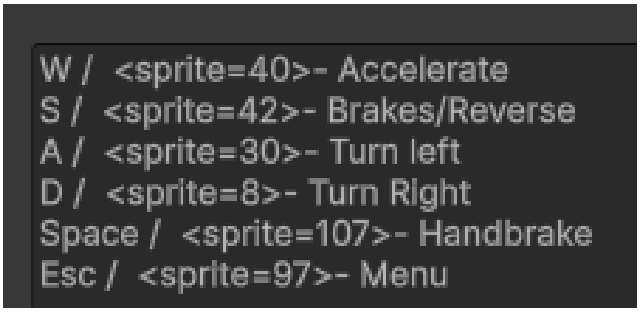
6.18) Sprite Editor

Με το τέλος αυτής της διαδικασίας, έπρεπε να φτιάξω με βάση το παραπάνω ένα Sprite Asset. Κάνοντας δεξί κλικ στην εικόνα που μόλις επεξεργάστηκα και πηγαίνοντας στο Create – TextMeshPro - Sprite Asset (κόκκινο βελάκι στην εικόνα 5.17) δημιουργήθηκε το αρχείο (κίτρινο βελάκι στην εικόνα 5.17).

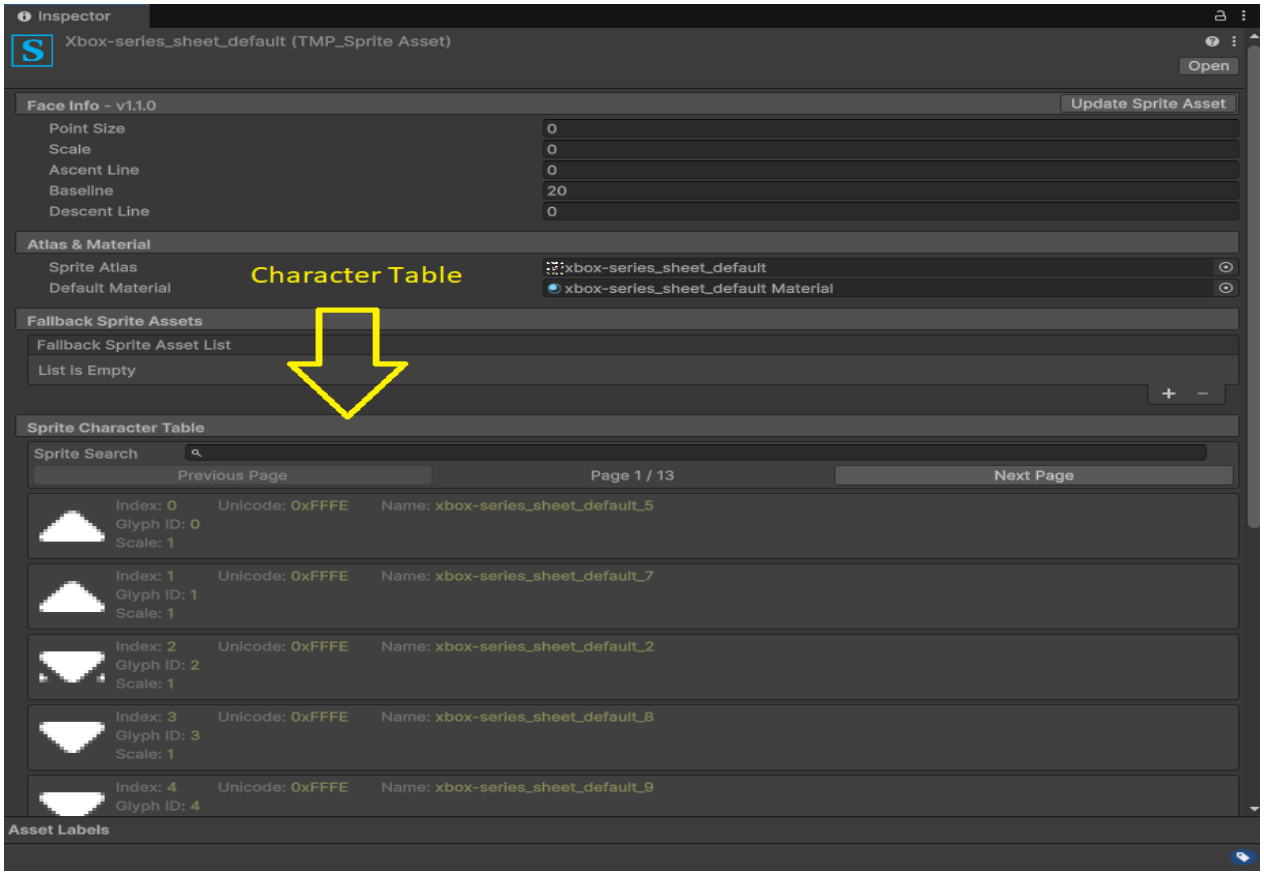


6.19) Δημιουργία Sprite Asset

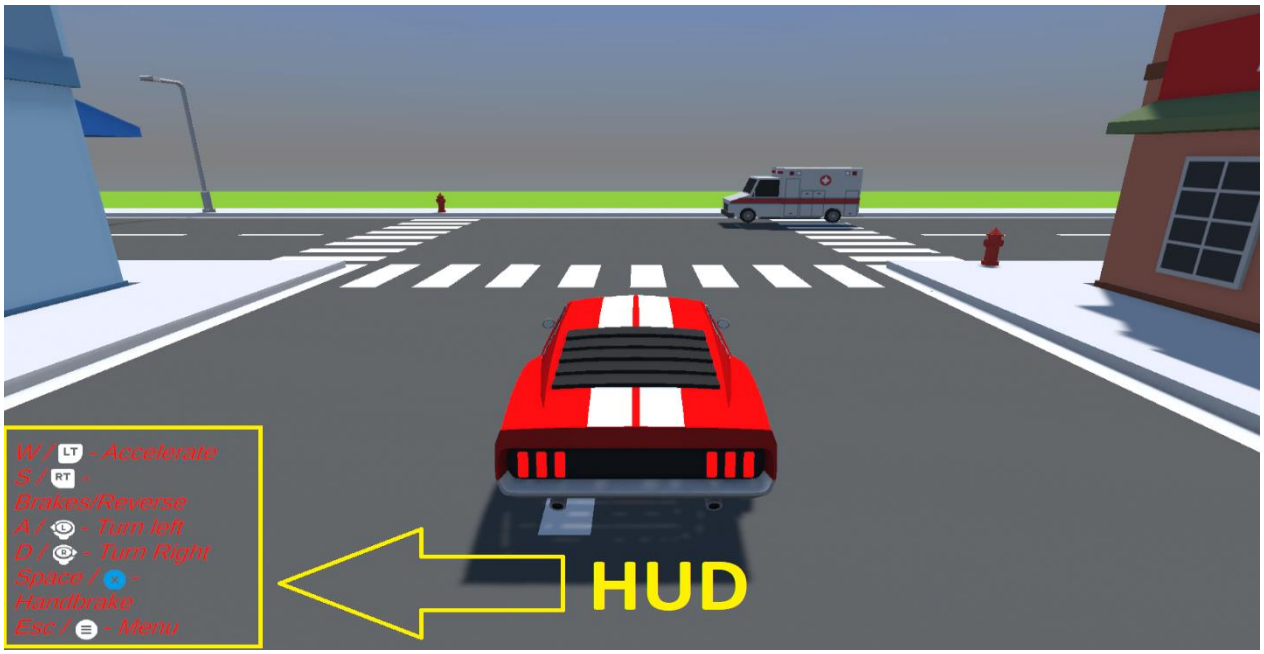
Πλέον μπορούσα να χρησιμοποιήσω αυτά τα Sprites σε ένα text field, δημιουργώντας έτσι και το HUD. Όταν ήθελα να υπάρχει ένα απλά έγραφα <sprite=?>, όπου «?» ένας μοναδικός αριθμός που αντιστοιχεί στο κάθε sprite και μπορείς να δεις στο Sprite Character Table στον Inspector του Sprite Asset.



6.20) Text Field με Sprites



6.21) Sprite Character Table

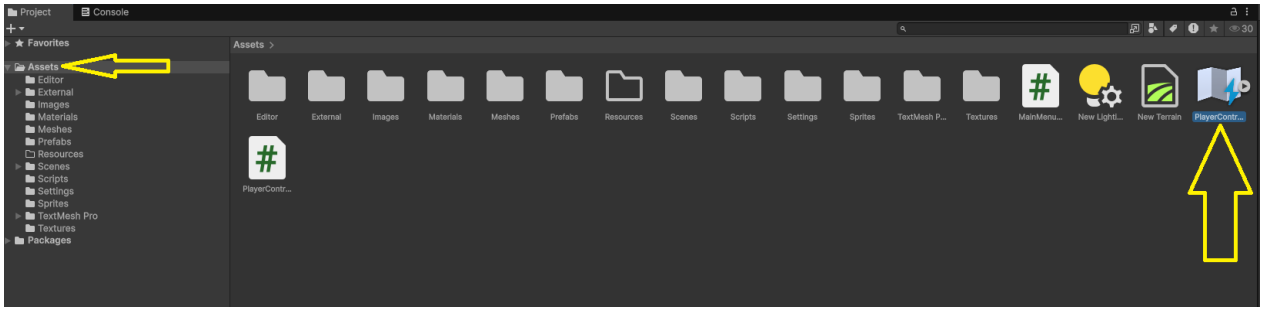


6.22)Τελική εμφάνιση HUD

6.7 Vehicle Παίκτη

Για τον παίκτη έψαξα και βρήκα ένα asset από το marketplace δωρεάν, το οποίο είχε ένα έτοιμο prefab αυτοκινήτου με τα κατάλληλα script για physics και οδήγηση του, το Prometeo: Car Controller. Κάνοντας το Import και επεξεργάζοντας αναλυτικά τα αντικείμενα και τα αρχεία που περιείχε, παρατήρησα ότι είχε λειτουργίες που δεν χρειαζόμουν και ο κώδικας για την ανάγνωση των κουμπιών (inputs) δεν χρησιμοποιούσε το Input Action Asset που προσφέρει η Unity. Γι' αυτό αποφάσισα να ξαναγράψω το συγκεκριμένο κομμάτι κώδικα.

Πρώτα όμως επεξεργάστηκα το PlayerControls Input Action Asset, διαμορφώνοντας τις μεθόδους που ήθελα και κάνοντας bind τα κουμπιά που ήθελα να καλούν την κάθε μέθοδο. Από default, κάθε project έχει ένα PlayerControls και το βρίσκεις μέσα στον κεντρικό φάκελο Assets στο Content Browser.



6.23) PlayerControls

Ανοίγοντας το έχει προκαθορισμένα inputs, τα οποία δεν μου χρησίμευαν. Έτσι τα διέγραψα όλα και άφησα το asset κενό. Πρώτη μου δουλεία ήταν να σκεφτώ πόσα διαφορετικά Action Maps ήθελα να έχει η εφαρμογή μου, δηλαδή αν ήθελα διαφορετικές σκηνές οθόνες ή ακόμα και απλά αντικείμενα να έχουν διαφορετικά inputs. Οπότε δημιούργησα 2, το CarMovement και το UI, για την κίνηση του αμαξιού και για μετακίνηση μεταξύ των αντικειμένων του UI.

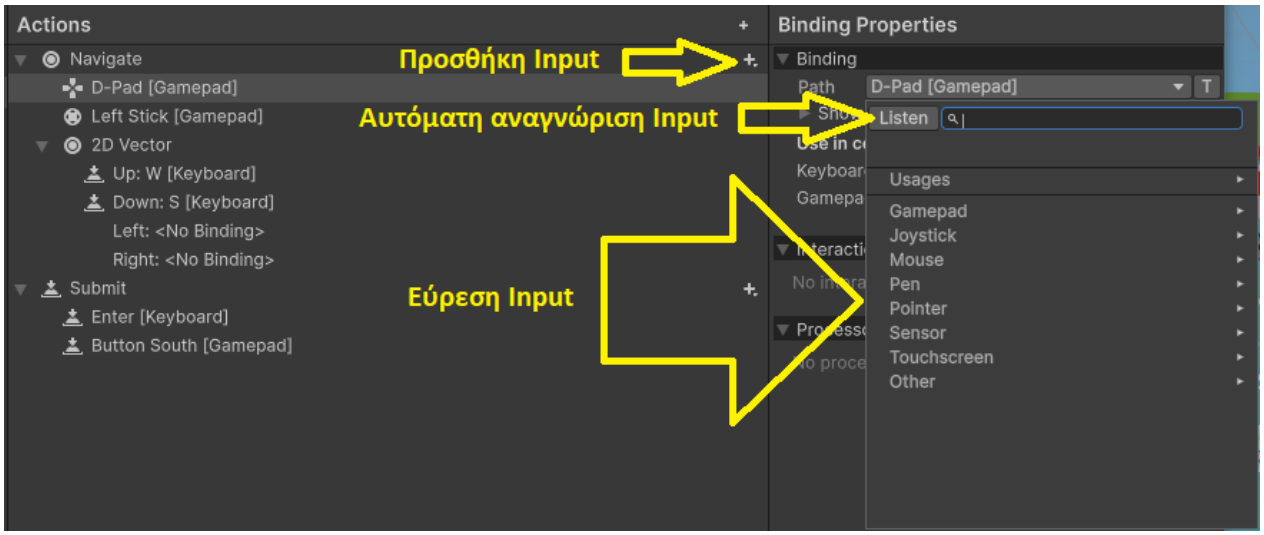
Αφού είχα τα Action Maps μου, ξεκίνησα να προσθέτω τα κατάλληλα actions στο καθένα. Για το CarMovement, τα Throttle, Brake, Handbrake, Move, OpenMenu και Camera, ενώ για το UI, τα Navigate και Submit.

Τέλος, έθεσα για κάθε μέθοδο ένα input. Για τις μεθόδους που καλούνταν από κάποιο κουμπί έπρεπε το Action Type να ήταν τύπου "Button", ενώ όταν καλούνταν και από κάποιο μοχλό του Gamepad, για παράδειγμα η κίνηση του αυτοκινήτου που γίνεται και με τα κουμπιά W/A/S/D του πληκτρολογίου και με τον αριστερό μοχλό του χειριστηρίου, το Action Type έπρεπε να είναι τύπου "Value" και Control Type τύπου "Vector 2".

6.24) Τύπου Button μέθοδος

6.25) Τύπου Value μέθοδος

Για την εισαγωγή input στην κάθε μέθοδο, πρέπει να πατηθεί το "+" στο τέλος της μεθόδους, και να επιλεχθεί ο τύπος του input. Οι τύποι αλλάζουν ανάλογα με τον τύπο της μεθόδου. Για προσθήκη input από το πληκτρολόγιο στην value μέθοδο Navigate, για παράδειγμα, χρειάστηκε να εισάγω ένα "2D Vector" επιλέγοντας το "Add Up/Down/Left/Right Composite", και να θέσω τα W και S κουμπιά του πληκτρολογίου στα Up και Down αντίστοιχα. Οτιδήποτε είδος και να είναι η μέθοδος, για input που προέρχονται από οτιδήποτε πέρα από πληκτρολόγιο, πρέπει να γίνει μια απλή εισαγωγή "Add Binding".



6.26) Προσθήκη Input

6.27) Τύποι Add Input

Δημιούργησα έτσι το Input Asset μου με όλες τις κινήσεις που θα χρειαζόταν ο παίκτης. Για να λειτουργεί όμως με κάθε script, έπρεπε κατά την εκκίνηση του script να τα διαβάζει και να τα ενεργοποιεί. Για να το πετύχω, σε κάθε script που είχα ήδη και δημιούργησα αργότερα, έφτιαξα μια μεταβλητή τύπου PlayerControls και στην μέθοδο void Start() της έθετα τιμή και την έκανα Enable.

```
public PlayerControls playerControls;
```

```
// Start is called before the first frame update
void Start()
{
  playerControls = new PlayerControls();
  playerControls.Enable();
                .
                .
                .
      }
```

Έχοντας έτοιμα τα player controls τα οποία διαβάζονται και από το έτοιμο script του αμαξιού, το μόνο που έμενε ήταν να αλλάξω τις συνθήκες των if εντολών με βάση τα controls που μόλις είχα θέσει και να ελέγχει τον τρόπο που καλείτε η κάθε μια. Ο γενικός τύπος για μια τέτοια συνθήκη είναι :

- ΌνομαΜεταβλητής.ActionMap.Action.τρόποςΕνεργοποίησης

Για παράδειγμα, η συνθήκη για το Throttle είναι: playerControls.CarMovement.Throttle.isPressed().

Αφού το αμάξι ήταν πλήρως λειτουργικό, προχώρησα στην ανάπτυξη του script για τις συγκρούσεις, CollisionScript.cs, που τοποθετήκε σαν component του prefab του αυτοκινήτου.

Η δουλειά του script συγκρούσεων, είναι να ελέγχει συνέχεια αν το αυτοκίνητο τρακάρει με κάποιο άλλο αντικείμενο στην σκηνή. Πιο συγκεκριμένα, ελέγχει αν το MeshCollider, που έχει σαν component ο κορμός (Body) του αμαξιού έρθει σε επαφή με κάποιο BoxCollider component κάποιου άλλου αντικειμένου με την μέθοδο OnCollisionEnter(Collision collision).

```
void OnCollisionEnter(Collision collision)
  {
    Scene scene = SceneManager.GetActiveScene();
    if (scene.name == "Survival")
    {
      // Increment collision count and show panel after 4 collisions
      collisionCount++;
      if (collisionCount >= 4)
      {
```

```
            Time.timeScale = 0f;
            StartCoroutine(VibrateGamepadForSeconds(0.123f, 0.234f, 1f));
            CollisionPanel.SetActive(true);
            RestartButton.Select(); // Automatically select the restart button when the panel is shown
        }
    }
    else
    {
        if (collision.collider is BoxCollider)
        {
            Time.timeScale = 0f;
        }
        StartCoroutine(VibrateGamepadForSeconds(0.123f, 0.234f, 1f));
        CollisionPanel.SetActive(true); // Show the collision panel
        RestartButton.Select(); // Automatically select the restart button when the panel is shown
    }
}
```

Αφού εντοπιστεί κάποια σύγκρουση, ο κώδικας κάνει έλεγχο για την σκηνή στην οποία είναι ο παίκτης, και ανάλογα είτε του δίνει 4 ευκαιρίες πριν εμφανίσει το Μενού σύγκρουσης είτε το εμφανίζει στην πρώτη φορά. Για το λόγο αυτό υπάρχουν 2 διαφορετικά Scenes στο Project, το Survival και το No Crash. Μέσα στο μπλοκ εντολών υπάρχει μια συνεργαζόμενη ρουτίνα (coroutine). Μέσα από αυτή, κάθε φορά που ενεργοποιείται κάποιον collision, αν ο χρήστης χρησιμοποιεί gamepad, τότε θα εκτελεστεί μια δόνηση ενός δευτερολέπτου.

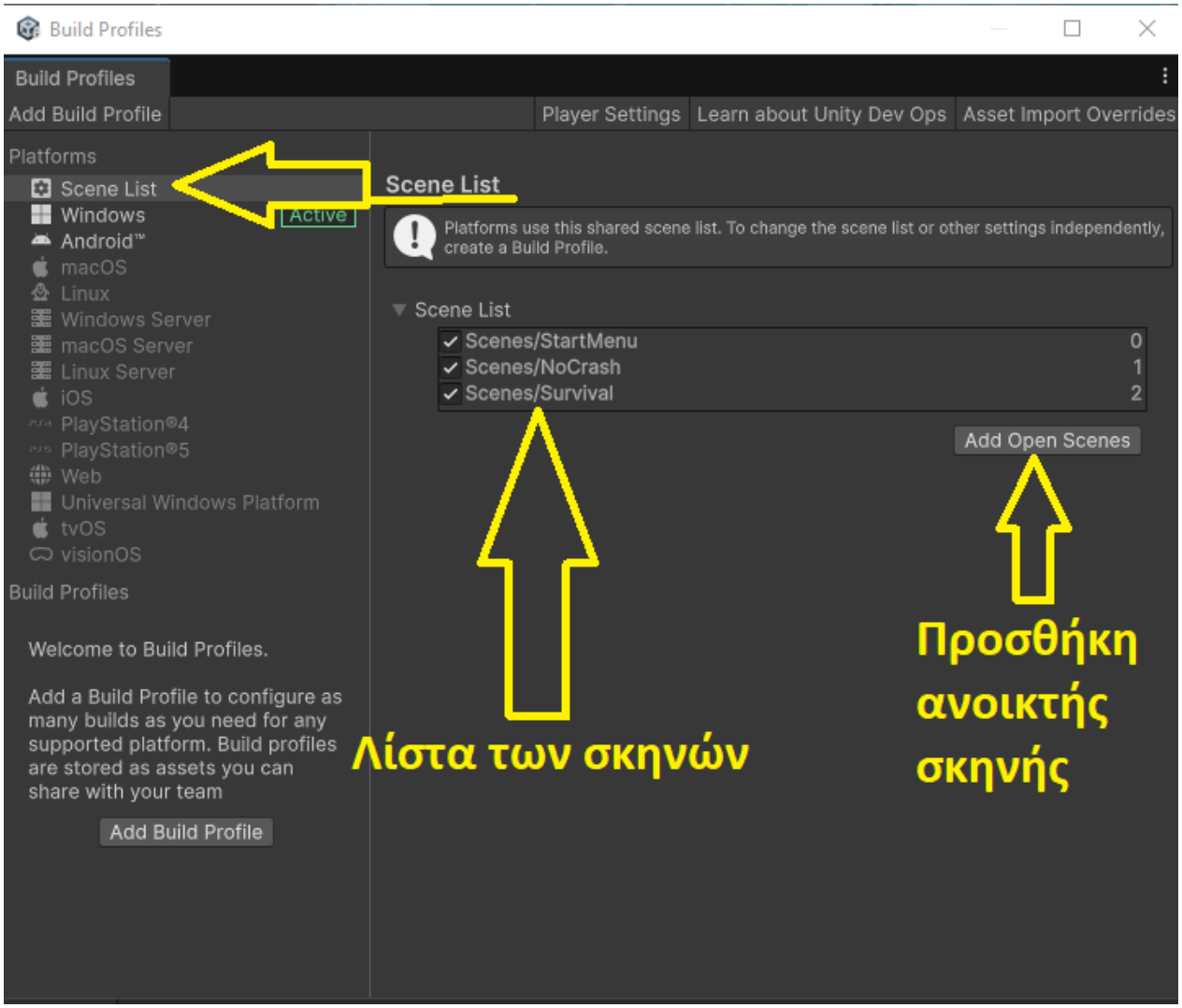
```
private IEnumerator VibrateGamepadForSeconds(float lowFreq, float highFreq, float duration)
{
```

```
        if (Gamepad.current != null)
        {
            Gamepad.current.SetMotorSpeeds(lowFreq, highFreq);
            yield return new WaitForSecondsRealtime(duration);
            Gamepad.current.SetMotorSpeeds(0f, 0f);
        }
    }
```

6.8 Build και Εκτέλεση

Έχοντας χτίσει τα πάντα που ήθελα, έμενε να χτίσω το παιχνίδι και να δοκιμάσω το τελικό προϊόν. Για να γίνει όμως σωστά, έπρεπε να γίνουν κάποιες απαραίτητες ρυθμίσεις. Συγκεκριμένα, έπρεπε να προστεθούν όλα τα scenes που ήθελα να υπάρχουν στο τελικό παιχνίδι και να οριστεί το scene με το οποίο ήθελα να ξεκινάει το παιχνίδι. Αυτό γίνεται μέσω του "Build Profiles" στο κεντρικό μενού (File – Build Profiles). Στο παράθυρο που θα ανοίξει, επέλεξα στα αριστερά την μενού Scene List. Στο κεντρικό σημείο του παραθύρου πρόσθεσα με drag and drop από το Content Browser τα Scenes που ήθελα και τα ανακατάταξα έτσι ώστε το scene του αρχικού μενού (StartMenu) να ήταν στην θέση 0.

Αν απλά θέλουμε να προσθέσουμε την σκηνή που δουλεύουμε εκείνη την ώρα, υπάρχει το κουμπί "Add Open Scene", που την προσθέτει κατευθείαν στο τέλος της λίστας.



6.28) Build Settings

Με το τέλος αυτής της διαδικασίας, μπορούσα να κάνω Build και να τρέξω το παιχνίδι. Πατώντας το “Build and Run” στο File του κεντρικού μενού ή το Ctrl + B στο πληκτρολόγιο, το παιχνίδι θα κάνει build. Το πρώτο Build θα κάνει περισσότερη ώρα από τα επόμενα. Με το που τελειώσει το Build το παιχνίδι θα ανοίξει κατευθείαν.

Μια καλή πρακτική είναι ο προγραμματιστής να κάνει και εδώ τακτικά “Build and Run”, ειδικά μετά από σημαντικές τροποποιήσεις, για έλεγχο ότι δουλεύει σωστά η εφαρμογή.

Κεφάλαιο 7ο Τελική Σύγκριση

Η Unity και η Unreal Engine αποτελούν τις δύο κυρίαρχες μηχανές ανάπτυξης διαδραστικού περιεχομένου, που χρησιμοποιούνται τόσο στον κλάδο της βιομηχανίας παιχνιδιών όσο και σε πεδία όπως η προσομοίωση, η εκπαίδευση, η αρχιτεκτονική απεικόνιση και η εικονική πραγματικότητα (VR). Παρότι αμφότερες παρέχουν πληθώρα εργαλείων και τεχνολογιών, διαφέρουν ουσιαστικά στη σχεδιαστική φιλοσοφία, την απόδοση, τη δυνατότητα επέκτασης, καθώς και στο τεχνολογικό οικοσύστημα που υποστηρίζουν.

Η Unity είναι περισσότερο προσανατολισμένη στην ευκολία πρόσβασης και στην ταχεία ανάπτυξη εφαρμογών. Η χρήση της γλώσσας προγραμματισμού C# καθιστά τη μηχανή προσβάσιμη σε ένα ευρύ φάσμα προγραμματιστών, ιδίως σε όσους προέρχονται από περιβάλλοντα .NET. Ο επεξεργαστής της Unity (Unity Editor) προσφέρει μια οπτικά κατανοητή διεπαφή, υποστηρίζοντας τη δημιουργία περιεχομένου μέσω drag-and-drop, χωρίς απαραίτητα να απαιτείται σύνθετος προγραμματισμός για τη διαχείριση βασικών λειτουργιών. Επιπλέον, η Unity παρέχει ενσωματωμένα εργαλεία για 2D και 3D ανάπτυξη, καθώς και ευρεία υποστήριξη για φορητές συσκευές, WebGL, κονσόλες και AR/VR πλατφόρμες. Αντιθέτως, η Unreal Engine, αν και πιο σύνθετη σε πρώτο επίπεδο, προσφέρει εγγενή υποστήριξη για γραφικά υψηλής πιστότητας (photorealism) και ένα εξαιρετικά εξελιγμένο rendering pipeline, το οποίο εξυπηρετεί καλύτερα ανάγκες AAA ανάπτυξης ή έργα που απαιτούν κινηματογραφική ποιότητα εικόνας.

Στο επίπεδο της απόδοσης και του rendering, η Unreal Engine διαθέτει ισχυρό πλεονέκτημα. Το σύστημα φωτισμού της, μέσω τεχνολογιών όπως το Lumen (real-time global illumination και reflections), παρέχει αποτελέσματα πολύ υψηλής ποιότητας χωρίς ανάγκη για baking, ενώ το Nanite επιτρέπει την απεικόνιση γεωμετρίας εξαιρετικά υψηλής λεπτομέρειας χωρίς significant performance drops. Αυτά τα χαρακτηριστικά καθιστούν την Unreal κατάλληλη για φωτορεαλιστικές εφαρμογές, virtual production, κινηματογραφικά έργα και αρχιτεκτονική απεικόνιση. Η Unity έχει κάνει σημαντικά βήματα προς αυτή την κατεύθυνση, εισάγοντας το Universal Render Pipeline (URP) και το High Definition Render Pipeline (HDRP), καθώς και τεχνολογίες όπως τα Adaptive Probe Volumes και η GPU-accelerated baking. Ωστόσο, αυτά τα χαρακτηριστικά βρίσκονται συχνά σε πειραματική φάση, ενώ η ολοκληρωμένη υλοποίησή τους είναι πιο κατακερματισμένη σε σχέση με την ομοιογένεια που προσφέρει η Unreal.

Όσον αφορά τη λογική ανάπτυξης και επέκτασης, η Unity διακρίνεται για την ευελιξία της. Το σύστημα component-based architecture επιτρέπει τη γρήγορη ανάπτυξη λειτουργιών μέσω scripts που εφαρμόζονται σε GameObjects, διευκολύνοντας την ανάπτυξη ανεξάρτητων modules. Επιπλέον, η κοινότητα της Unity και το Asset Store παρέχουν χιλιάδες επεκτάσεις, συστήματα και εργαλεία τρίτων που ενισχύουν την παραγωγικότητα. Από την άλλη πλευρά, η Unreal βασίζεται περισσότερο στη χρήση Blueprints, ενός οπτικού scripting συστήματος που επιτρέπει τον ορισμό λογικής χωρίς τη χρήση κώδικα. Αν και το σύστημα αυτό είναι ιδιαίτερα ισχυρό, συχνά θεωρείται πιο περίπλοκο στην εμβάθυνση από ό,τι οι αντίστοιχες μέθοδοι της Unity, ιδιαίτερα όταν απαιτείται μετάβαση σε C++ για την ανάπτυξη πιο εξειδικευμένων λειτουργιών. Η Unreal προσφέρει πιο ισχυρό εργαλείο σύνολο εκτός κουτιού (out-of-the-box), ενώ η Unity στηρίζεται περισσότερο στην προσθήκη και παραμετροποίηση εργαλείων από το οικοσύστημα της.

Η εκμάθηση των δύο μηχανών παρουσιάζει αξιοσημείωτες διαφορές. Η Unity, με την προσβασιμότητά της και το εκτενές online documentation, tutorials, και ενεργή κοινότητα, θεωρείται πιο κατάλληλη για αρχάριους ή μικρές ομάδες ανάπτυξης. Η Unreal, ενώ παρέχει ποιοτικά εργαλεία και εκτενή τεκμηρίωση, έχει μεγαλύτερη μαθησιακή καμπύλη, ιδιαίτερα για όσους δεν έχουν εξοικείωση με τη γλώσσα C++ ή την έννοια της memory management. Επίσης, το workflow της Unreal για mobile ή 2D ανάπτυξη θεωρείται πιο περιοριστικό και λιγότερο αποδοτικό σε σχέση με την Unity, που έχει σχεδιαστεί εξαρχής με γνώμονα την υποστήριξη πολλών πλατφορμών.

Ένας άλλος σημαντικός άξονας σύγκρισης είναι η διαχείριση πόρων και απαιτήσεων. Η Unreal, λόγω του πολύπλοκου rendering και των δυνατοτήτων της, έχει υψηλότερες απαιτήσεις σε hardware και είναι πιο βαριά για τον μέσο υπολογιστή, τόσο σε runtime όσο και σε επίπεδο editor. Αντιθέτως, η Unity είναι αισθητά πιο ελαφριά και πιο φιλική σε μικρότερες ομάδες, φορητές συσκευές και συστήματα χαμηλής κατανάλωσης. Η ελαφρότητα της Unity διευκολύνει, επίσης, τη συνεχή ενσωμάτωση και δοκιμή αλλαγών, κάτι που υποστηρίζεται από το modular σύστημα scripting της.

Στον οικονομικό και νομικό τομέα, η Unity είχε παραδοσιακά πιο ευέλικτο μοντέλο αδειοδότησης, αν και οι πρόσφατες αλλαγές πολιτικής (π.χ. Runtime Fee) έχουν δημιουργήσει αβεβαιότητα στην κοινότητα. Η Unreal προσφέρει δωρεάν πρόσβαση στον πλήρη πηγαίο κώδικα και χρεώνει royalty μόνο μετά από συγκεκριμένο όριο εσόδων, το οποίο είναι ελκυστικό για μικρές ομάδες και studios με υψηλές προσδοκίες απόδοσης. Η δυνατότητα άμεσης τροποποίησης του source code στην Unreal προσφέρει επιπλέον τεχνική ευελιξία, ιδιαίτερα σε εφαρμογές που απαιτούν προσαρμογές σε χαμηλό επίπεδο.

Το Unity Asset Store αποτελεί ένα από τα πιο εκτεταμένα marketplaces στην ανάπτυξη διαδραστικού περιεχομένου, με χιλιάδες διαθέσιμα assets, plugins, εργαλεία και συστήματα, που καλύπτουν ανάγκες από γραφικά και animation έως AI, networking και UI. Η πληθώρα διαθέσιμου υλικού, σε συνδυασμό με τις συχνές προσφορές και τη χαμηλή τιμολόγηση, καθιστούν το Asset Store ιδανικό για ανεξάρτητους developers και μικρές ομάδες. Το Unreal Marketplace, αν και μικρότερο σε πλήθος, προσφέρει assets υψηλής ποιότητας, εστιασμένα σε ρεαλιστικά περιβάλλοντα και AAA παραγωγές. Επιπλέον, η Epic Games προσφέρει επιλεγμένα premium assets δωρεάν κάθε μήνα, ενισχύοντας την πρόσβαση σε επαγγελματικό περιεχόμενο.

Η Unity διακρίνεται για την ευρεία υποστήριξή της από μια ενεργή κοινότητα, με πλήθος διαθέσιμων tutorials, forums, επίσημη τεκμηρίωση, καθώς και την εκπαιδευτική πλατφόρμα Unity Learn, που διευκολύνει σημαντικά την είσοδο νέων χρηστών. Η Unreal προσφέρει επίσης πλούσια τεκμηρίωση, hands-on παραδείγματα και πλήρη πρόσβαση στον source code μέσω GitHub, καθιστώντας την ιδιαίτερα ισχυρή για προχωρημένους χρήστες και ερευνητές. Και οι δύο μηχανές υποστηρίζονται από μεγάλα communities που συνεισφέρουν ενεργά σε συζητήσεις, tutorials και ανοιχτού κώδικα εργαλεία, ενισχύοντας το οικοσύστημα τους.

Κατηγορία	*Unreal Engine*	*Unity*
Γλώσσα Προγραμματισμού	C++ / Blueprints	C#
Γραφικά	Πολύ υψηλή ποιότητα – photorealism (Nanite, Lumen)	Καλά (HDRP/URP) – λιγότερο photorealism
Απόδοση σε Mobile/2D	Υψηλή	Περιορισμένη
Απόδοση σε AAA/VR	Μέτρια	Εξαιρετική
Ευκολία Εκμάθησης	Υψηλή	Απότομη Καμπύλη
Επεκτασιμότητα	Asset Store	Πλήρη Πρόσβαση στον Source Code
Hardware Απαιτήσεις	Χαμηλές	Πολύ Υψηλές
Μοντέλο Αδειοδότησης	Δωρεάν με περιορισμούς + Runtime Fee	Δωρεάν μέχρι έσοδα $1M + royalties

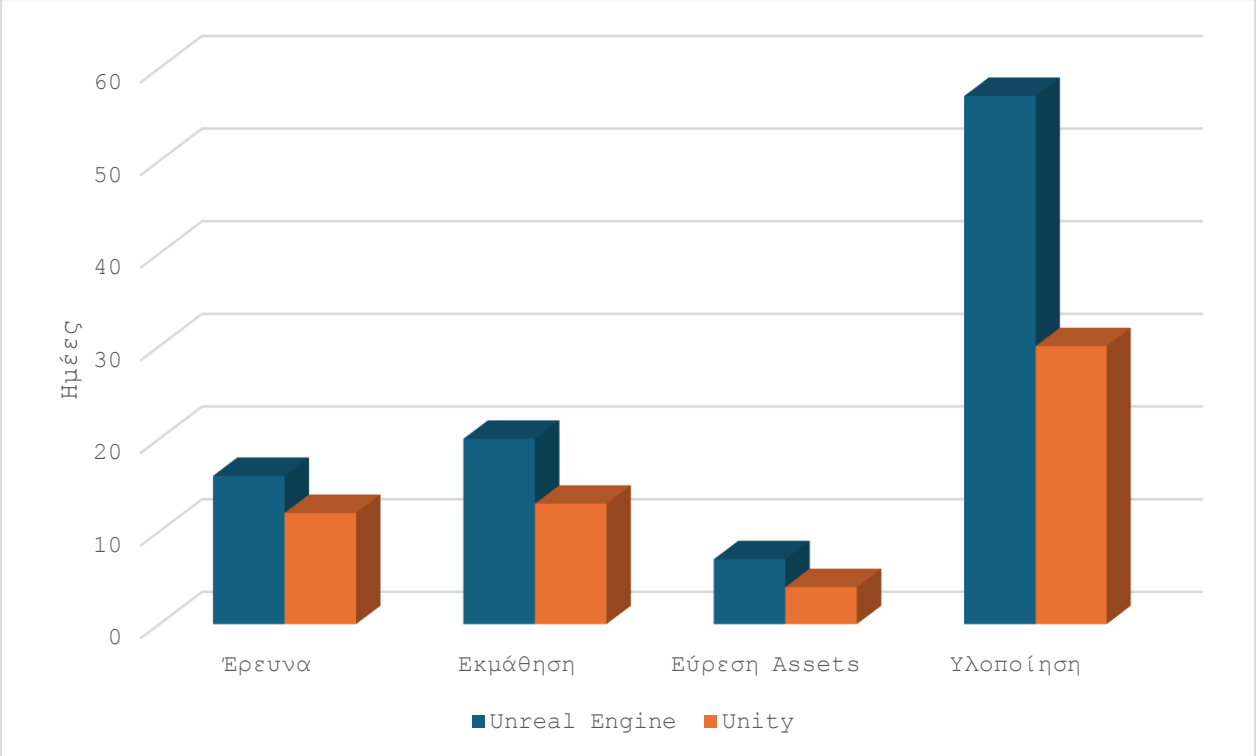
3. Συνοπτική Σύγκριση των 2 μηχανών

Μέσα από την δικιά μου εμπειρία, συνειδητοποίησα πως προσωπικά βρήκα και τις 2 μηχανές μέτριες στην εκμάθηση τους. Κάθε μια παρουσίασε τις δικές της δυσκολίες και ευκολίες. Συγκεκριμένα, ερχόμενος από ένα καλό υπόβαθρο γνώσεων κώδικα, το project της Unity, ολοκληρώθηκε σε εξαιρετικά μικρότερο χρόνο από αυτό της Unreal. Αυτό που με δυσαρέστησε λίγο ίσως ήταν ότι ενώ για το δικό μου project που ήταν μικρό, οι τεχνολογίες και ο κώδικας ήταν αρκετά άμεσος, κατάλαβα ότι σε μεγαλύτερα, ο χρήστης ίσως χρειαστεί να βρει έμμεσους τρόπους δημιουργίας πραγμάτων μέσω του κώδικα.

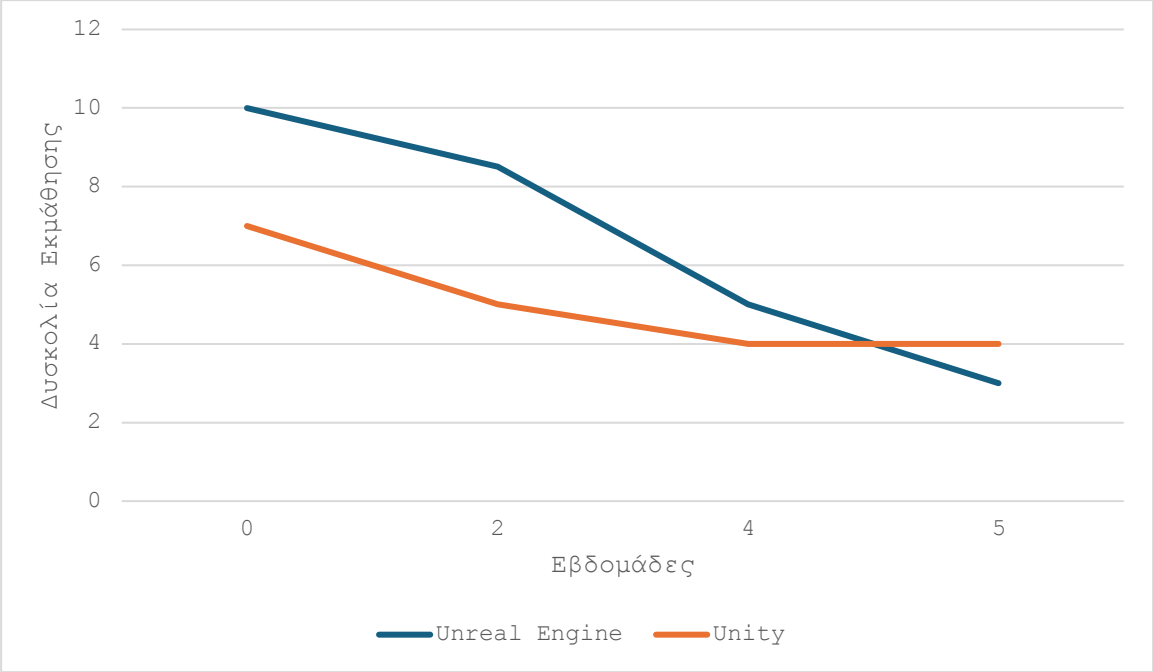
Σε αντίθεση, η Unreal ένιωσα ότι μου έδινε μια τεράστια ελευθερία κινήσεων. Στην αρχή αυτό ήταν κακό. Επένδυσα πάρα πολύ χρόνο να ψάχνω τι τεχνολογίες και πώς θα χρησιμοποιήσω αυτές τις τεχνολογίες που μου προσφέρονταν. Έπειτα, έπρεπε να επενδύσω επιπλέον χρόνο στο μάθω τα Blueprints. Αφού όμως ξεκίνησα να καταλαβαίνω και να αποκτώ μια μεγαλύτερη ευχέρεια με την μηχανή, η δημιουργία του project απλουστεύτηκε πολύ. Τα εργαλεία που μου δίνονταν, μου επέτρεπαν να φτιάξω πολύπλοκους και μεγάλους κόσμους και λογικές, σε λίγες ώρες.

Η επιλογή μεταξύ Unity και Unreal δεν έχει μια απόλυτα “σωστή” απάντηση. Κάθε έργο και κάθε ομάδα έχουν διαφορετικές απαιτήσεις, τεχνολογικούς στόχους και διαθέσιμους πόρους. Ενώ η Unity ευνοεί την ευελιξία, την ταχύτητα και την πολυπλατφορμικότητα, η Unreal προσφέρει απαράμιλλη ποιότητα γραφικών, ισχυρά ενσωματωμένα εργαλεία και πλήρη τεχνική ελευθερία.

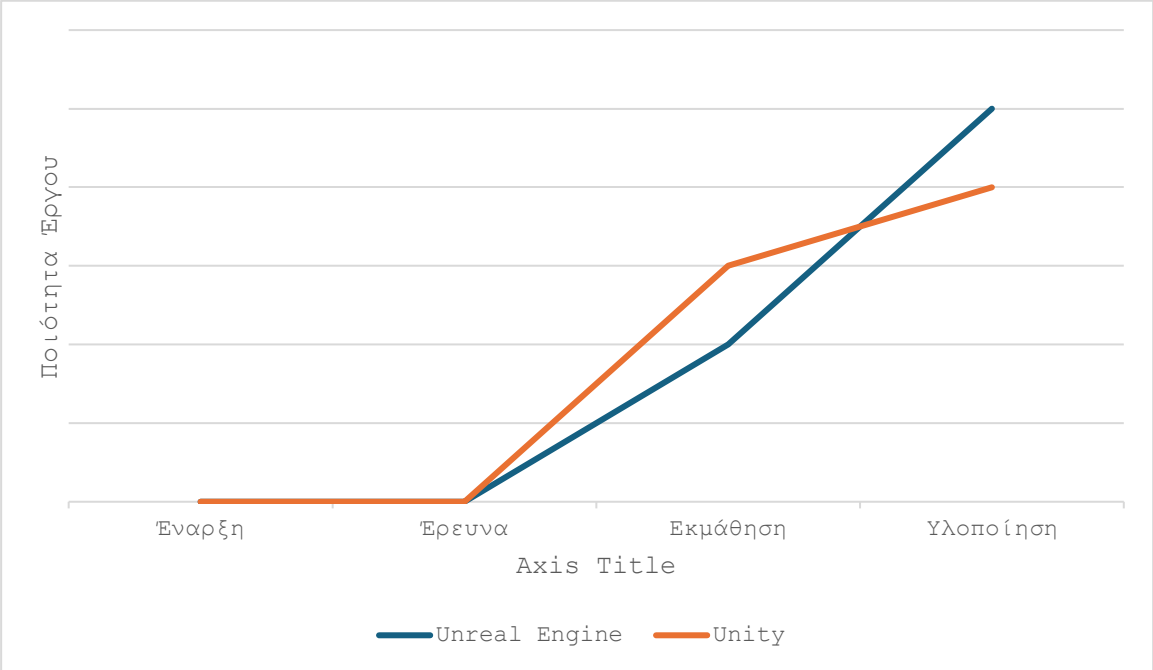
Στα παρακάτω διαγράμματα φαίνονται οι χρόνοι (σε ημέρες) που μου πήρε το κάθε στάδιο για την ολοκλήρωση του, η καμπύλη δυσκολίας με το πέρασμα των εβδομάδων κατά το στάδια Έρευνας και Εκμάθησης και τέλος η καμπύλη στην οποία φαίνεται η ποιότητα του έργου κατά την διάρκεια κάθε σταδίου.



7.1) Ημέρες Ολοκλήρωσης κάθε Σταδίου



7.2) Δυσκολία στα στάδια Έρευνας και Εκμάθησης



7.3) Ποιότητα Έργου σε Κάθε Στάδιο

Κεφάλαιο 8ο Επίλογος

8.1 Συνοπτική Αποτίμηση και Μελλοντικές Κατευθύνσεις

Η παρούσα πτυχιακή εργασία επικεντρώθηκε στη συγκριτική μελέτη δύο εκ των σημαντικότερων μηχανών ανάπτυξης παιχνιδιών, της **Unreal Engine** και της **Unity**, αναλύοντας τις τεχνολογίες, τα εργαλεία και τις δυνατότητες που προσφέρουν. Ως μεθοδολογική προσέγγιση επιλέχθηκε η υλοποίηση της ίδιας εφαρμογής και στις δύο μηχανές, με στόχο να αξιοποιηθούν ορισμένες από τις διαθέσιμες τεχνολογίες και να αναδειχθούν τα πλεονεκτήματα και οι περιορισμοί κάθε περιβάλλοντος.

Η μελέτη αυτή ανέδειξε τον σημαντικό ρόλο που διαδραματίζουν οι μηχανές παιχνιδιών όχι μόνο στη βιομηχανία του gaming, αλλά και σε ευρύτερους τομείς όπως η εκπαίδευση, η αρχιτεκτονική απεικόνιση, η εικονική και επαυξημένη πραγματικότητα. Παράλληλα, ανέδειξε τη σημασία της σωστής επιλογής εργαλείων από τους νέους developers, καθώς αυτή μπορεί να επηρεάσει καθοριστικά τη διαδικασία ανάπτυξης, τη βιωσιμότητα ενός έργου αλλά και τις προοπτικές εξέλιξης μιας ομάδας ή ενός ατόμου στον κλάδο.

Μελλοντικά, η περαιτέρω εμβάθυνση στις σύγχρονες δυνατότητες που προσφέρουν οι δύο μηχανές – όπως τα συστήματα τεχνητής νοημοσύνης, τα εργαλεία φυσικής προσομοίωσης, οι μηχανισμοί βελτιστοποίησης για διαφορετικές πλατφόρμες και η ενσωμάτωση εργαλείων παραγωγής περιεχομένου με χρήση AI – θα μπορούσε να οδηγήσει σε πιο ολοκληρωμένα και καινοτόμα αποτελέσματα.

8.2 Συμπεράσματα

Η υλοποίηση της ίδιας εφαρμογής και στις δύο πλατφόρμες παρείχε πολύτιμα συμπεράσματα σχετικά με την εμπειρία χρήσης, την αποδοτικότητα των εργαλείων και την καμπύλη εκμάθησης που αντιμετωπίζει ο νέος developer.

Η Unity αναδείχθηκε ως μια πιο φιλική και ευέλικτη μηχανή, κατάλληλη για μικρές ομάδες και γρήγορη ανάπτυξη εφαρμογών, με έμφαση στη φορητότητα σε πολλές πλατφόρμες. Από την άλλη, η Unreal Engine ξεχώρισε για τα κορυφαία γραφικά, τα προηγμένα εργαλεία real-time rendering και την ενσωμάτωση σύγχρονων τεχνολογιών, καθιστώντας την ιδανική επιλογή για απαιτητικά έργα υψηλής ποιότητας.

Συνολικά, η εργασία ανέδειξε ότι δεν υπάρχει «καλύτερη» μηχανή με απόλυτους όρους· η επιλογή εξαρτάται από το πλαίσιο ανάπτυξης, τους στόχους του εκάστοτε έργου και τις ανάγκες της ομάδας. Αυτό ακριβώς το στοιχείο της προσαρμοστικότητας και της ποικιλίας καθιστά τις δύο μηχανές αλληλοσυμπληρούμενες και πολύτιμες για το οικοσύστημα ανάπτυξης παιχνιδιών και διαδραστικών εφαρμογών.

8.3 Μελλοντικές Επεκτάσεις και Προοπτικές

Η παρούσα εργασία μπορεί να αποτελέσει αφετηρία για περαιτέρω ερευνητική και πρακτική ενασχόληση με τις δύο μηχανές ανάπτυξης παιχνιδιών. Μια φυσική συνέχεια θα μπορούσε να είναι η περαιτέρω αξιοποίηση και επέκταση των δυνατοτήτων που προσφέρουν οι τεχνολογίες **procedural generation** και **AI**, οι οποίες χρησιμοποιήθηκαν στην παρούσα εφαρμογή. Η ενσωμάτωση πιο σύνθετων αλγορίθμων

τεχνητής νοημοσύνης θα μπορούσε να βελτιώσει τη συμπεριφορά των χαρακτήρων και των αντικειμένων μέσα στο παιχνίδι, ενώ η χρήση procedural generation θα μπορούσε να επεκταθεί ώστε να δημιουργεί δυναμικά μεγαλύτερους και πιο πολύπλοκους κόσμους, παρέχοντας συνεχή ανανέωση περιεχομένου χωρίς επιπλέον χειροκίνητη εργασία.

Παράλληλα, η ανάπτυξη εφαρμογών **VR και AR** ανοίγει νέους ορίζοντες για χρήση πέρα από τον χώρο του gaming, όπως στην εκπαίδευση, την αρχιτεκτονική και την ιατρική προσομοίωση, επιτρέποντας την αξιοποίηση των ήδη εφαρμοσμένων τεχνολογιών AI και procedural generation σε ρεαλιστικά και διαδραστικά περιβάλλοντα. Επίσης, η δημιουργία εργαλείων που επιτρέπουν τη μεταφορά έργων και την αξιοποίηση των πλεονεκτημάτων κάθε μηχανής θα ενίσχυε τη συνεργασία και την ευελιξία των developers.

Τέλος, η εμπειρία που προέκυψε από τη σύγκριση της ανάπτυξης της ίδιας εφαρμογής σε Unreal Engine και Unity προσφέρει σημαντικά διδάγματα για νέους developers. Μέσα από αυτήν, γίνεται σαφής η πρακτική αξία προηγμένων τεχνολογιών όπως η τεχνητή νοημοσύνη και το procedural generation, ενώ παράλληλα αναδεικνύονται τα πλεονεκτήματα και οι περιορισμοί κάθε μηχανής. Η γνώση αυτή μπορεί να καθοδηγήσει τους νέους δημιουργούς στην επιλογή της κατάλληλης πλατφόρμας και στην αξιοποίηση των εργαλείων της με τρόπο που να υποστηρίζει αποτελεσματικά καινοτόμα και αποδοτικά έργα.

Βιβλιογραφία

[1] R. Edwards, "The Game Production Pipeline: Concept to Completion. What goes into making a game?," 16 Μαρτίου 2006. [Online]. Available: https://www.ign.com/articles/2006/03/16/the-game-production-pipeline-concept-to-completion.

[2] "What is a Gaming or Game Engine?," Arm, [Online]. Available: https://www.arm.com/glossary/gaming-engines.

[3] P. A. Bizjak, "Unity vs Unreal Engine: Features, performance, and usability," 14 Μαΐου 2024. [Online]. Available: https://proxify.io/articles/unity-vs-unreal-engine.

[4] K. T. Jensen, "25 Years Later: The History of Unreal and an Epic Dynasty," PCMag, 22 Μαΐου 2023. [Online]. Available: https://www.pcmag.com/news/25-years-later-the-history-of-unreal-and-an-epic-dynasty.

[5] T. Sweeney, "If You Love Something, Set It Free," 2 Μαρτίου 2015. [Online]. Available: https://www.unrealengine.com/en-US/blog/ue4-is-free.

[6] U. Engine, "Frequently Asked Questions," Epic Games, [Online]. Available: https://www.unrealengine.com/en-US/faq.

[7] U. Engine, "Working with Content," Unreal Engine Documentation version 5.0, [Online]. Available: https://docs.unrealengine.com/5.0/en-US/working-with-content-in-unreal-engine/.

[8] "Programming with C++," Epic Games, [Online]. Available: https://dev.epicgames.com/documentation/en-us/unreal-engine/programming-with-cplusplus-in-unreal-engine.

[9] A. Agarwal, "Unreal Engine and its Evolution," External Labs, 12 Απριλίου 2023. [Online]. Available: https://externlabs.com/blogs/unreal-engine-and-its-evolution/.

[10] "What's new in Unreal Engine 5.1," Epic Games, [Online]. Available: https://www.unrealengine.com/en-US/updates/unreal-engine-5-1.

[11] "Hardware and Software Specifications," Epic Games, [Online]. Available: https://dev.epicgames.com/documentation/en-us/unreal-engine/hardware-and-software-specifications-for-unreal-engine.

[12] "Blueprints Visual Scripting," Epic Games, [Online]. Available: https://dev.epicgames.com/documentation/en-us/unreal-engine/blueprints-visual-scripting-in-unreal-engine.

[13] "Lumen Global Illumination and Reflections," Epic Games, [Online]. Available: https://dev.epicgames.com/documentation/en-us/unreal-engine/lumen-global-illumination-and-reflections-in-unreal-engine.

[14] "Landscape Splines," Epic Games, [Online]. Available: https://dev.epicgames.com/documentation/en-us/unreal-engine/landscape-splines-in-unreal-engine.

[15] "Procedural Content Generation Framework," Epic Games, [Online]. Available: https://dev.epicgames.com/documentation/en-us/unreal-engine/procedural-content-generation--framework-in-unreal-engine.

[16] "Nanite Virtualized Geometry," Epic Games, [Online]. Available: https://dev.epicgames.com/documentation/en-us/unreal-engine/nanite-virtualized-geometry-in-unreal-engine.

[17] "ZoneGraph Quick Start Guide," Epic Games, [Online]. Available: https://dev.epicgames.com/community/learning/tutorials/qz6r/unreal-engine-zonegraph-quick-start-guide.

[18] "Basic Navigation," Epic Games, [Online]. Available: https://dev.epicgames.com/documentation/en-us/unreal-engine/basic-navigation-in-unreal-engine.

[19] "Sequencer Overview," Epic Games, [Online]. Available: https://dev.epicgames.com/documentation/en-us/unreal-engine/unreal-engine-sequencer-movie-tool-overview.

[20] "World Partition," Epic Games, [Online]. Available: https://dev.epicgames.com/documentation/en-us/unreal-engine/world-partition-in-unreal-engine.

[21] "History of Unity Game Engine," Agate, 1 Ιουνίου 2023. [Online]. Available: https://agate.id/history-of-unity-game-engine/.

[22] M. Best, "Unity 6 is here: See what's new," Unity, 17 Οκτωβρίου 2024. [Online]. Available: https://unity.com/blog/unity-6-features-announcement.

[23] "System requirements for Unity 6.1," Unity, [Online]. Available: https://docs.unity3d.com/6000.1/Documentation/Manual/system-requirements.html.

[24] "Unity Package Manager UI," Unity Technologies, [Online]. Available: https://docs.unity3d.com/Manual/upm-ui.html.

[25] "Plugins," Unity Technologies, [Online]. Available: https://docs.unity3d.com/Manual/Plugins.html.

[26] "Prefabs," Unity Technologies, [Online]. Available: https://docs.unity3d.com/550/Documentation/Manual/Prefabs.html.

[27] "Forward and Deferred rendering," Unity Technologies, [Online]. Available: https://docs.unity3d.com/Packages/com.unity.render-pipelines.high-definition%4014.0/manual/Forward-And-Deferred-Rendering.html.

[28] "Render pipelines," Unity Technologies, [Online]. Available: https://docs.unity3d.com/6000.1/Documentation/Manual/render-pipelines.html.

[29] "Unity's Data-Oriented Technology Stack (DOTS," Unity Technologies, [Online]. Available: https://unity.com/dots.

[30] E. f. Unity, Unity Technologies, [Online]. Available: https://unity.com/ecs.

[31] "Job system overview," Unity Technologies, [Online]. Available: https://docs.unity3d.com/Manual/job-system-overview.html.

[32] "Burst compiler," Unity Technologies, [Online]. Available: https://docs.unity3d.com/Packages/com.unity.burst@1.8/manual/index.html.

[33] "ML-Agents Overview," Unity Technologies, [Online]. Available: https://docs.unity3d.com/Packages/com.unity.ml-agents@3.0/manual/index.html.

[34] "ML-Agents Toolkit (GitHub Repository)," Unity Technologies, [Online]. Available: https://github.com/Unity-Technologies/ml-agents.

[35] "Powering cameras for real-time productions," Unity Technologies, [Online]. Available: https://unity.com/features/cinemachine.

[36] "About Timeline," Unity Technologies, [Online]. Available: https://docs.unity3d.com/Packages/com.unity.timeline@1.2/manual/index.html.

[37] "Incremental garbage collection," Unity Technologies, [Online]. Available: https://docs.unity3d.com/2023.1/Documentation/Manual/performance-incremental-garbage-collection.html.

[38] "Garbage collection modes," Unity Technologies, [Online]. Available: https://docs.unity3d.com/6000.1/Documentation/Manual/performance-incremental-garbage-collection.html.

[39] "Garbage collector overview," Unity Technologies, [Online]. Available: https://docs.unity3d.com/2023.1/Documentation/Manual/performance-garbage-collector.html.

[40] "Introduction to Adaptive Probe Volumes," Unity Technologies, [Online]. Available: https://docs.unity3d.com/6000.1/Documentation/Manual/urp/probevolumes-concept.html.

[41] "Adaptive Probe Volumes Release HDRP/URP," Unity Technologies, [Online]. Available: https://unity.com/558-adaptive-probe-volumes-release-hdrp-urp.

[42] "Understanding Adaptive Probe Volumes," Unity Technologies, [Online]. Available: https://docs.unity3d.com/Packages/com.unity.render-pipelines.high-definition%4017.0/manual/probevolumes-concept.html.

[43] "Adaptive Probe Volumes (APV)," Unity Technologies, [Online]. Available: https://unity.com/de/2048-adaptive-probe-volumes-apv.