ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη παιχνιδιού με μηχανή Unreal Engine 4

Μάριος Μπαμπούρης

Εισηγητής: Δρ. Κωσταντίνος Κουκουλέτσος, Καθηγητής

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη παιχνιδιού με μηχανή Unreal Engine 4

Μάριος Μ. Μπαμπούρης
Α.Μ. 44959

Εισηγητής:

Δρ. Κωσταντίνος Κουκουλέτσος, Καθηγητής

Εξεταστική Επιτροπή: Πάρις Μαστοροκώστας, Ιωάννης Έλληνας

Ημερομηνία εξέτασης:14/10/2010

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα πτυχιακή εργασία ολοκληρώθηκε σε ένα χρονικό διάστημα πολλών μεταβολών, σε ένα ενδιαφέρον αντικείμενο και δύσκολο, δηλαδή την ανάπτυξη και τον σχεδιασμό παιχνιδιών μέσω της μηχανής του Unreal Engine 4. Την προσπάθειά αυτή υποστήριξε ο επιβλέπων καθηγητής, τον Κύριο Κωσταντίνο Κουκουλέτσο τον οποίο θα ήθελα να ευχαριστήσω. Επιπλέον την Μαρία Πιστικού για την επιμέλεια του κειμένου και τον συμφοιτητή μου Σωτήρη Παπαδόπουλο για το Testing του παιχνιδιού που αναπτύχθηκε.

ΠΕΡΙΛΗΨΗ

Η παρούσα πτυχιακή εργασία ασχολείται με την σχεδίαση και ανάπτυξη ενός παιχνιδιού με τη βοήθεια της μηχανής Unreal Engine 4. Η ανάπτυξη παιχνιδιών με Game Engines έχει δώσει την δυνατότητα στους προγραμματιστές και στις βιομηχανίες να μπορούν να σχεδιάζουν και να αναπτύσσουν ένα παιχνίδι με μεγαλύτερη ευκολία και αποτελεσματικότητα. Η συγκεκριμένη πτυχιακή εργασία ξεκινάει με μια ιστορική αναδρομή, του αντικειμένου και συνεχίζει αναλύοντας τα Game Engines, επικεντρώνοντας στην λειτουργεία του Unreal Engine 4. Τελικός σκοπός της πτυχιακής είναι η δημιουργία ενός FPS(First Person Shooter) παιχνιδιού με χρήση του Unreal Engine 4.

ABSTRACT

The present thesis is a study on the design and the development of a video game using Game Engines. In particular we are using the Unreal Engine 4. Nowadays, it is common that the game industry is developing games using Game Engines. Game Engines are a more stable approach on designing and developing a 3D game, giving more reliability, effectiveness and offering more convenience on developing a video Game with easily available tools that anybody can acquire and learn. The thesis starts with a historical review of game development, explains how a Game Engine works, and gives details of Unreal Engine 4 and how can be implemented. Finally the development of a 3D FPS (First Person Shooter) Video Game based on Unreal Engine 4 from scratch is presented.

ΕΠΙΣΤΗΜΟΝΙΚΗ ΠΕΡΙΟΧΗ: Ανάπτυξη λογισμικού

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Unreal Engine 4, game engine, game development, FPS

Περιεχόμενα

Κεφάλαιο 1ο : Εισαγωγή

Ιστορική αναδρομή

Η ιστορία του Game Development(ανάπτυξης παιχνιδιών) ξεκίνησε από την ανάπτυξη του πρώτου Video Game. Βέβαια, όσον αφορά ποιο ήταν το πρώτο video game εξαρτάται με το τί θέλουμε να ορίσουμε σαν Video Game.Τα πρώτα παιχνίδια που δημιουργήθηκαν δεν είχαν τόσο ψυχαγωγικό χαρακτήρα. Οι προγραμματιστές της εποχής που τα ανέπτυξαν εστίαζαν περισσότερο στη δημιουργία ενός παιχνιδιού με σκοπό και όχι τόσο στο τι εμπειρία θα πάρει ο χρήστης αλλά στην δημιουργία ενός λογισμικού. Αυτά τα πρώτα παιχνίδια για να «τρέξουν» απαιτούσαν Mainframe Computers ή παλμογράφους (Moore, Michael E.; Novak, Jeannie, 2010).

Με τον όρο βιντεοπαιχνίδι εννοείται οποιοδήποτε παιχνίδι αναπαράγεται με τη χρήση κάποιας ηλεκτρονικής συσκευής. Αυτή μπορεί να είναι ένας ηλεκτρονικός υπολογιστής, μια κονσόλα βιντεοπαιχνιδιών, ένα κινητό τηλέφωνο και άλλα. Περιλαμβάνει αλληλεπίδραση με μια διεπαφή χρήστη για την παραγωγή οπτικής ανάδρασης σε μια συσκευή βίντεο. (Στεναμερίδης, 2016)

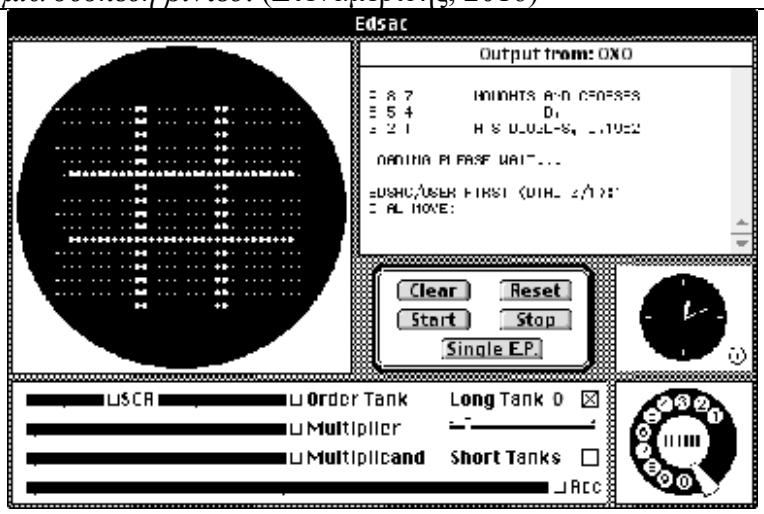


Figure 1 OXO το πρώτο Video game με ψηφιακή οθόνη.

Γιά παράδειγμα, τα πρώτα παιχνίδια που ξεκίνησαν να τρέχουν σε προσιτά συστήματα ήταν εκείνα που είχαν σαν οθόνη έναν παλμογράφο, όπως το “Tennis For Two”.

Figure 2 Game: Tennis for two.

Τα πρώτα παιχνίδια που αναπτυχθήκαν την δεκαετία του 1960 δεν ήταν για το κοινό. Χρειάζονταν Mainframe υπολογιστές. Η ανάπτυξη των παιχνιδιών που θα κυκλοφορούσε εμπορικά ξεκίνησε την δεκαετία του 70’ με τον ερχομό των πρώτων κονσόλων και των πρώτων οικιακών υπολογιστών όπως ο Apple I.

Computer Space

Η πραγματική, όμως, εμπορική και τεχνολογική ανάπτυξη και σχεδίαση παιχνιδιών ξεκίνησε από το 1970 και μετά, όταν τα Arcade παιχνίδια, στις πρώτες κονσόλες της εποχή, όπως το VCS,NES κλπ… αρχίζαν να κυκλοφορούν. Αυτή η εποχή έχει σημαδευτεί από ένα παιχνίδι με το όνομα Computer space, το οποίο ήταν το πρώτο παιχνίδι που υπήρξε στην αγορά και το οποίο έδινε την δυνατότητα ο χρήστης να παίξει βάζοντας κέρματα.

Figure 3 Computer Space

Ιδιαίτερα πρωτοποριακό για την τότε εποχή στο Computer space ήταν ότι ως οθόνη χρησιμοποιείτο μία ασπρόμαυρη τηλεόραση και ότι το υπολογιστικό του σύστημα ήταν φτιαγμένο από TTL Chips 74 σειράς (Yagoda, 2008).

Pong

Από το 1970 και ύστερα ο κλάδος του Video Games είχε αρχίσει να αναπτύσσεται ραγδαία, με εταιρίες, όπως η Atari, ανακοινώνοντας το θρυλικό παιχνίδι Pong, το οποίο έκανε τόση μεγάλη επιτυχία που οδήγησε πολλές ακόμα εταιρίες να δημιουργήσουν κλώνους του Pong,γεννώντας έτσι την βιομηχανία των Video game (Miller Michael , 2005).

Το Pong ήταν ένα από τα πρώτα Arcade Video Games.Στην ουσία ήταν ένα παιχνίδι πινγκ-πονγκ που παρουσιαζόταν σε γραφικά 2 διαστάσεων. Το Pong ήταν τόσο επιτυχημένο , το οποίο συνέβαλε ριζικά στην άνοδο της βιομηχανία των Video Games και των Κονσόλων. Το Pong έδωσε στην Atari την ώθηση να ενθαρρύνει το προσωπικό της να δημιουργήσει πιο πρωτοποριακά παιχνίδια (Wikipedia, Pong, 2019).

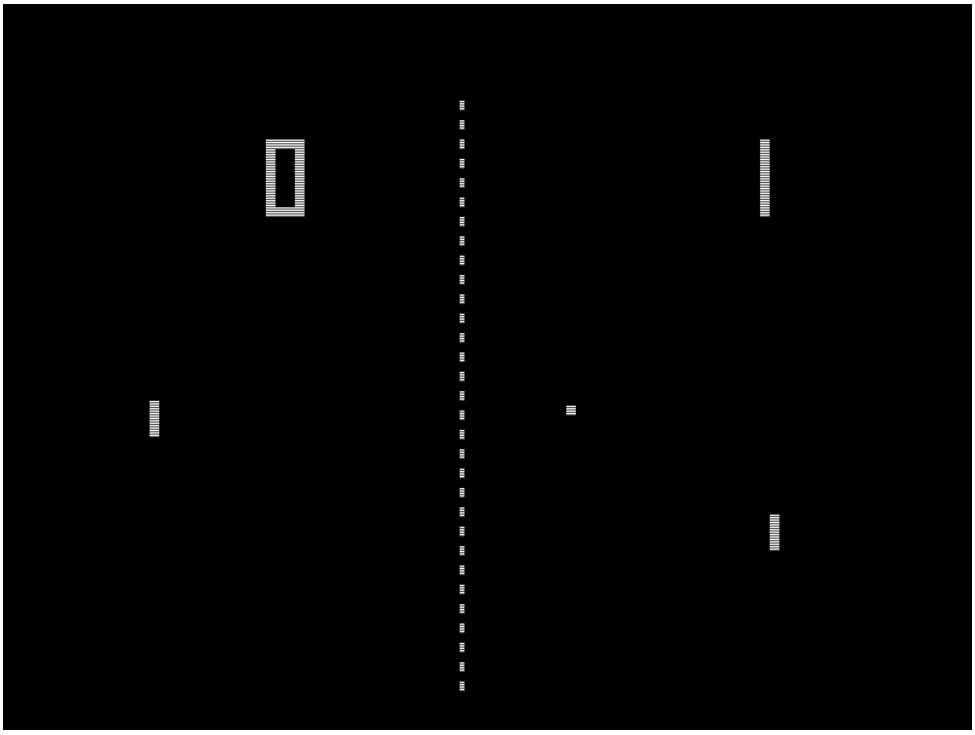


Figure 4 «Το θρυλικό *Pong*»

Κονσόλες

> *Μια κονσόλα παιχνιδιών είναι ένας υπολογιστής ειδικού σκοπού το οποίο είναι σχεδιασμένο για δραστική ψυχαγωγία. Μια κονσόλα λειτουργεί κατά πως είναι κατασκευασμένη, όπως ένας υπολογιστής με τα ίδια κύρια εξαρτήματα, όπως μια κεντρική μονάδα επεξεργασίας, μια κάρτα γραφικών και μια μνήμη. Μια κονσόλα για να εξισορροπήσει το κόστος σε σχέση με τους υπολογιστές πολύ συχνά χρησιμοποίει παλιά εξαρτήματα* (Techopedia, 2019).

Magnavox odyssey

Την ιδιά χρονιά η Magnavox ανακοίνωσε το πρώτο video game system για τους καταναλωτές. Το Odyssey εφευρέθηκε από τον Ralph H. Baer και μπορούσε να συνδεθεί με μια απλή τηλεόραση, το οποίο δεν είχε μικροεπεξεργαστή σαν υπολογιστική μονάδα. Ο πυρήνας του συστήματος ήταν μια πλακέτα με 40 τρανζίστορ και διόδους. Το Odyssey ήταν πολύ περιορισμένο, μπορούσε να πράξει πολύ απλά γραφικά.

Figure 5 Το Odyssey

Το Odyssey, όμως, δεν κατάφερε να ολοκληρώσει την εμπορική επιτυχία της πρωτοτυπίας του μέχρι το 1975 που το Arcade παιχνίδι της Atari "Pong" έγινε ένα από τα πιο δημοφιλή βιντεοπαιχνίδια που ενσωματώθηκαν στην κονσόλα (Tyson, 2019).

Atari 2600

Έχοντας κατακτήσει τον κόσμο και τα Arcade με το Pong την δεκαετία του 70΄, η Atari αποσκόπησε να εφεύρει ξανά το Gaming με μια κονσόλα που θα μπορούσε συνεχώς να αυξάνει την βιβλιοθήκη της στα παιχνίδια. Με αυτό τον τρόπο η Atari κυριάρχησε κάνοντας το Atari 2600 την πιο επιτυχημένη κονσόλα δεύτερης γενιάς. Μια κονσόλα, η οποία κυριάρχησε στην αγορά κι έσπασε ρεκόρ πωλήσεων μέσα σε περίοδο 13 χρονών που βρισκόταν στην αγορά. Πολλοί καταναλωτές αγόρασαν ένα Atari μόνο και μόνο για το Space Invaders. (Cohen, 2019)

Για την εποχή του ήταν αρκετά εξελιγμένο και πρωτοπόρο με χαρακτηριστικά όπως:

Figure 6 Το εσωτερικό του Atari 2600

- Τον MOS 6502 μικροεπεξεργαστή.
- Stella, ένα τροποποιημένο τσιπάκι γραφικών το οποίο ήλεγχε τον συγχρονισμό με την τηλεόραση με όλες τις άλλες διεργασίες βίντεο του επεξεργαστή.
- 128 bytes of Μνήμη Ram
- 4-kilobyte ROM-based κασέτες παιχνιδιών.

Τα Chips βρίσκονταν σε μια πλακέτα PCB, η οποία ένωνε, επίσης, και τις θύρες των Joystick και την υποδοχή των κασετών, καθώς και την τροφοδοσία και την έξοδο του βίντεο. Τα παιχνίδια κωδικοποιούνταν σε λογισμικό και γράφονταν σε ROM τσιπάκια που ενσωματωνόταν στις κασέτες. Αυτό το σύστημα οδήγησε στην μεγάλη επιτυχία στο Atari 2600, καθώς έδινε την δυνατότητα στην βιομηχανία να αναπτύσσει διαφορά παιχνίδια και να αναπαραχθούν στην κονσόλα δίνοντας έτσι την δυνατότητα σε χιλιάδες κόσμο να μπορεί να απολαύσει μια μεγάλη γκάμα από παιχνίδια, δίχως να χρειάζεται να αλλάξει σε κάποια άλλη κονσόλα ή να αγοράσει Arcade Μηχανές (Tyson, 2019).

NES

Στις αρχές της δεκαετίας του 1980, οι άνθρωποι της Nintendo ξεκίνησαν να δουλεύουν με σκοπό να παράγουν μία νέα κονσόλα, η οποία ναι μεν θα ήταν φθηνή, αλλά δε θα είχε κι επεξεργαστική ισχύ. Ο εγκέφαλος της κονσόλας ήταν ο επεξεργαστής 6502 μαζί με μία συμπληρωματική μονάδα επεξεργασίας εικόνας.

Το Nintendo Entertainment System (NES) ήταν η πρώτη κονσόλα που είχε χειριστήριο σε αντίθεση με joystick. Το αρχικό όνομα του NES ήταν το Famicom ή Family computer και στην ουσία είχε βγει ήδη στις αγορές τις Ιαπωνίας από το 1983, όπου εκεί πούλησε 2.5 εκατομμύρια μονάδες. Βλέποντας αυτήν την επιτυχία στην Ιαπωνία, η Nintendo αποφάσισε να το βγάλει και στις παγκόσμιες αγορές (sofasandsectionals, 2019).

Το NES κατέκτησε την πρώτη θέση στις πωλήσεις ανάμεσα στις κονσόλες της τότε εποχής και έδωσε μία ακόμα ευκαιρία στην αγορά των βιντεοπαιχνιδιών, που εκείνη την εποχή περνούσε τεράστια κρίση, μετά την κατάρρευση του 1983. Το παιχνίδι που πούλησε περισσότερα απ' όλα ήταν το Super Mario Bros., ενώ άλλα γνωστά παιχνίδια είναι το The Legend of Zelda, τα Super Mario Bros. 2 και 3, το Metroid και το Final Fantasy (Wikipedia, NES, 2019).

Figure 7 NES με το χειριστήριο του

Figure 8 Super Mario Bros

Η σύγχρονη εποχή του Gaming

Από το 2000 και μετά, μπαίνοντας στην εποχή του διαδικτύου οι δυνατότητες και η υπολογιστική ισχύ είχαν εξελιχτεί σε τόσο γρήγορο ρυθμό, που με κάθε νέα κυκλοφορία έβγαιναν καινούργια γραφικά, καινούργιοι τρόποι ανάπτυξης και όλο και περισσότερος κόσμος επένδυε πάνω στην βιομηχανία του Gaming.

Αυτό οφείλεται και στο γεγονός ότι το κόστος των εξαρτημάτων όπως οι servers, οι υπολογιστές και του διαδικτύου είχε γίνει πιο οικονομικό σε σχέση με τα παλιότερα χρόνια. Με αποτέλεσμα το διαδίκτυο να είναι διαδεδομένο σε κάθε γωνιά της γης καθώς 3.2 δις εκατομμύρια άνθρωποι έχουν πρόσβαση στο σε αυτό (ESA-Essential-Facts-2015, 2015).

Σύμφωνα με τους ESA Computer and Video Games Industry, μία ερευνά του 2015 δείχνει ότι, τουλάχιστον, 1.5 δις εκατομμύρια άνθρωποι που έχουν ιντερνέτ παίζουν κάποιο video game.

Multiplayer

Πλέον η τεχνολογία, μέσω του διαδίκτυού, δίνει την δυνατότητα σε εκατομμύρια ανθρώπους ανά τον κόσμο να απολαμβάνουν τα παιχνίδια σαν μια κοινωνική δραστηριότητα μέσω του Multiplayer. Μια πρόσφατη έρευνα της ESA (ESA-Essential-Facts-2015, 2015) έδειξε ότι το 54% των ανθρώπων που παίζουν παιχνίδια νιώθουν ότι αυτό το hobby τους βοηθάει κοινωνικά, δηλαδή στο να δημιουργήσουν φιλίες με άλλα άτομα, ενώ ένα 45% δήλωσε ότι είναι μια δραστηριότητα που περνάνε χρόνο με την οικογένεια τους.

Στην ουσία, το Multiplayer είναι ένας τρόπος με τον οποίο πολλοί άνθρωποι, που παίζουν ένα συγκεκριμένο παιχνίδι, συνδέονται μεταξύ τους προκείμενου να παίξουν μαζί και να πετύχουν διάφορους σκοπούς, απολαμβάνοντας μια ξεχωριστή εμπειρία μέσω αυτής της κοινωνικής δραστηριότητας.

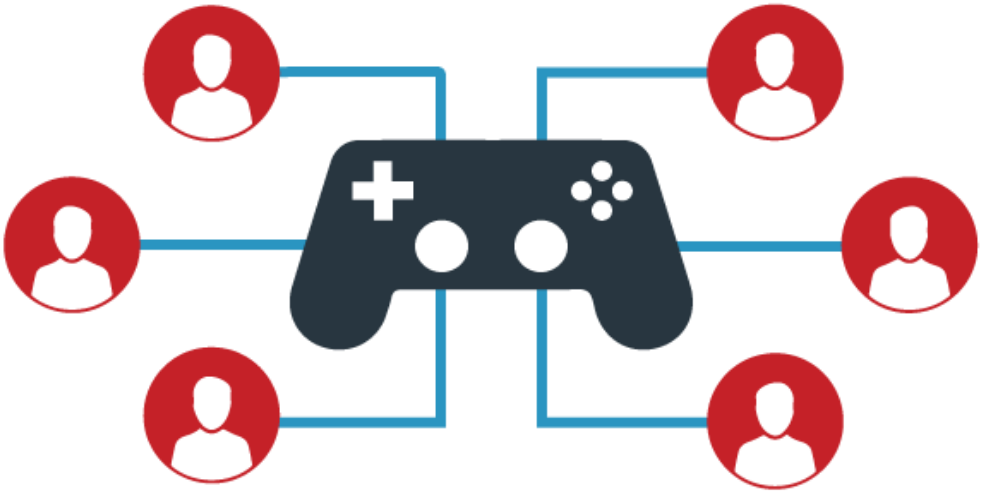


Figure 9 Multiplayer

Η Στροφή στο Mobile Gaming.

Από τότε που τα Smartphones και τα App Stores χτύπησαν την αγορά το 2007, η βιομηχανία των παιχνιδιών έχει δεχτεί μια ραγδαία εξέλιξη που έχει αλλάξει όχι μόνο τους ανθρώπους που παίζουν παιχνίδια αλλά επίσης έφερε την βιομηχανία στην που κουλτούρα. Οι γρήγορες καινοτομίες στην κινητή τεχνολογία την τελευταία δεκαετία έχουν δημιουργήσει μια έκρηξη των κινητών παιχνιδιών, η οποία έχει ξεπεράσει την αγορά των παιχνιδιών που είναι για κονσόλες. (Chikhani, 2015)

Το Μέλλον

Η μεταφορά στην κινητή τεχνολογία έχει ορίσει ένα νέο κεφάλαιο στην βιομηχανία των παιχνιδιών, αλλά, όσο καταλληλά κι αν είναι αυτού του είδους τα παιχνίδια στους ανθρώπους, αυτά τα παιχνίδια έχουν και τους περιορισμούς τους. Οι οθόνες των κινητών είναι μικρές και η επεξεργαστική ισχύ των κινητών τηλέφωνων καθώς και η μνήμη δεν μπορούν να συγκριθούν με έναν κανονικό υπολογιστή, περιορίζοντας τις δυνατότητες των παιχνιδιών.

Ήδη τα τελευταία χρονιά βλέπουμε ότι οι τάσεις στο Mobile Gaming αρχίζουν και πέφτουν, καθώς το κόστος των εταιριών και το αντίστοιχο κόστος της διαμοίρασης αυτών των παιχνιδιών έχουν εκτοξευθεί τα τελευταία χρονιά.

Παρόλο που το Mobile Gaming αναπτύσσεται, οι κονσόλες ακόμα ανθίζουν με κάθε καινούργια γενιά να παρουσιάζει μια νέα αποχή τεχνολογιών και δυνατοτήτων. Δυο νέες τεχνολογίες, οι οποίες θα μπορούσαν να παίξουν ένα κύριο ρολό στο μέλλον των παιχνιδιών είναι η «Εικονική πραγματικότητα» και η «Τεχνητή Νοημοσύνη».

Όταν οι αλλαγές αυτές γίνουν, υποστηρίζεται ότι το 2025 η βιομηχανία των παιχνιδιών θα είναι τελείως μη αναγνωρίσιμη από αυτό που είναι σήμερα. (Chikhani, 2015).

Κεφάλαιο 2ο : Game Engines

Ιστορικά

Πριν από τις μηχανές παιχνιδιών, τα παιχνίδια γράφονταν εξ' ολοκλήρου σαν οντότητες, όπως για παράδειγμα, ένα παιχνίδι για το Atari 2600, έπρεπε να σχεδιαστεί από το μηδέν για να κάνει βέλτιστη χρήση του υλικού εμφάνισης – αυτή η κεντρική ρουτίνα εμφάνισης σήμερα ονομάζεται πυρήνας από τους δημιουργούς retro.

Άλλες πλατφόρμες είχαν περισσότερες δυνατότητες, αλλά ακόμα και όταν η εμφάνιση δεν ήταν μέριμνα, οι περιορισμοί μνήμης συνήθως σαμποτάριζαν τις προσπάθειες για τη δημιουργία της βαριά, σε δεδομένα, σχεδίασης που μια μηχανή απαιτούσε. Ακόμα και σε πιο φιλικές πλατφόρμες, πολύ λίγα μπορούσαν να χρησιμοποιηθούν ανάμεσα σε παιχνίδια. Η γρήγορη ανάπτυξη του arcade υλικού σήμαινε ότι το μεγαλύτερο μέρος του κώδικα θα έπρεπε να πεταχτεί μετά έτσι και αλλιώς, καθώς οι μεταγενέστερες γενιές παιχνιδιών θα χρησιμοποιούσαν τελείως διαφορετικές σχεδιάσεις παιχνιδιών οι οποίες εκμεταλλεύονταν επιπλέον πόρους συστήματος.

Οι περισσότερες σχεδιάσεις παιχνιδιών τη δεκαετία του 1980 διεξάγονταν μέσου ενός σκληρά κωδικοποιημένου συνόλου κανόνων με μια μικρή ποσότητα δεδομένων επιπέδου και γραφικών.

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

Figure 10 Game Engines

Ο όρος «μηχανή παιχνιδιού» άρχισε να εμφανίζεται κατά τη δεκαετία του 1990, ειδικά σε σχέση με 3D παιχνίδια όπως FPS.

Μεταγενέστερα παιχνίδια, όπως το Quake III Arena και το Unreal του 1998 της Epic Games σχεδιάζονταν με βάσει αυτή την προσέγγιση, δηλαδή με τη μηχανή και το περιεχόμενο αναπτυγμένα ξεχωριστά.

Οι μοντέρνες μηχανές παιχνιδιών είναι μερικά από τα πιο περίπλοκα και χρονοβόρα προγράμματα για να αναπτυχθούν. Η συνεχής εξέλιξη των μηχανών παιχνιδιών έχει δημιουργήσει ένα ισχυρό διαχωρισμό ανάμεσα στο rendering, το scripting, την τέχνη, και το σχεδιασμό επιπέδων (Laird and Jamin, 2003).

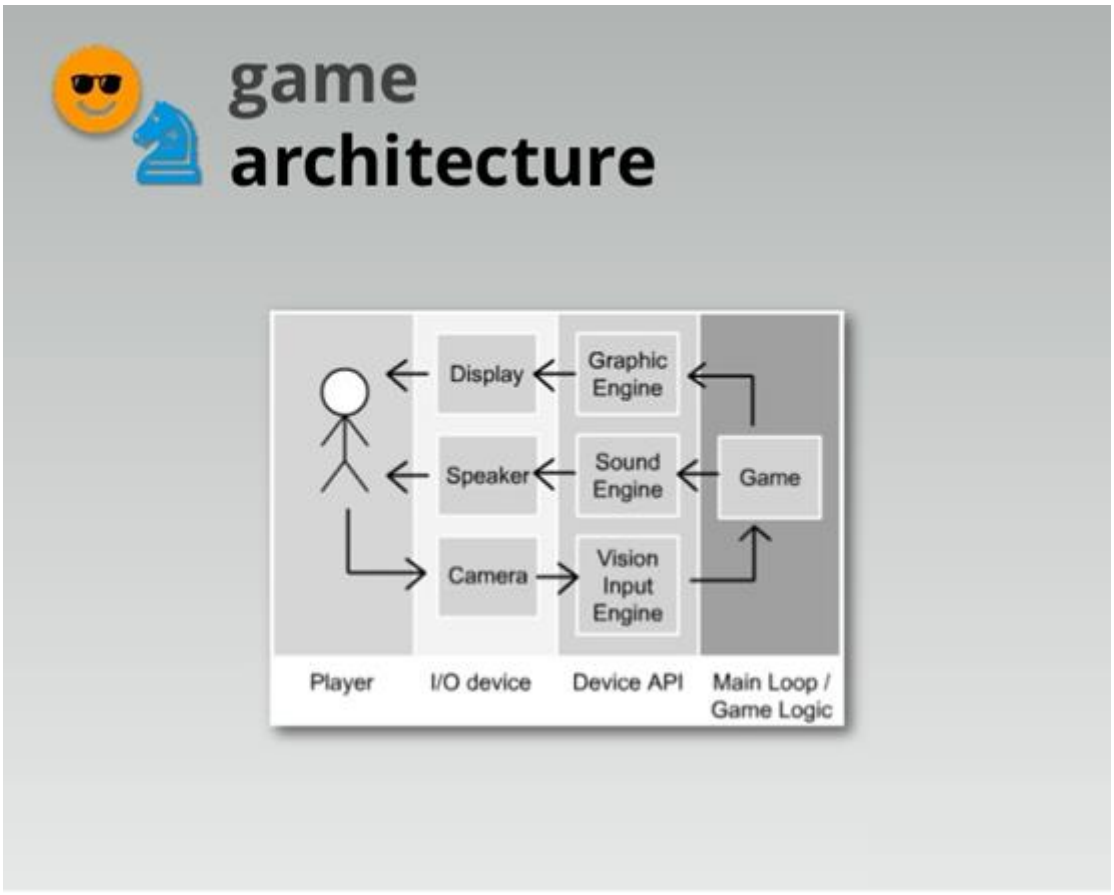


Figure 11 Πως λειτουργεί η αρχιτεκτονική των παιχνιδίων.

Στην παραπάνω εικόνα βλέπουμε πως λειτουργεί η αρχιτεκτονική μιας μηχανής παιχνιδιού. Όπως μπορούμε να δούμε η αρχιτεκτονική, είναι χωρισμένη σε 4 κομμάτια. Το πρώτο κομμάτι είναι η αλληλεπίδραση του παίχτη με την μηχανή. Πιο συγκεκριμένα ο παίχτης αλληλοεπιδρά με τις συσκευές εισόδου, για παράδειγμα την γωνία εστίασης που θέλει να έχει στο παιχνίδι. Με την σειρά τους, οι συσκευές εισόδου, αλληλοεπιδρούν με διάφορα API, όπως API ήχου ή γραφικών. Τέλος όλα αυτά επεξεργάζονται από την λογική του παιχνιδιού, που μας δίνει πίσω εξόδους.

Ορισμός

Μια μηχανή παιχνιδιού είναι ένα σύστημα λογισμικού σχεδιασμένο για τη δημιουργία και την ανάπτυξη βιντεοπαιχνιδιών. Υπάρχουν πολλές μηχανές παιχνιδιών, οι

οποίες είναι σχεδιασμένες να δουλεύουν σε κονσόλες, σε λάπτοπ, καθώς και σε υπολογιστές με λειτουργικά συστήματα Microsoft Windows, Linux, και Mac OS. Στην ουσία, αυτό που κάνει μια μηχανή παιχνιδιού είναι να περιλαμβάνει μια μηχανή που επεξεργάζεται γραφικά ("renderer") για 2D ή 3D γραφικά, μια μηχανή φυσικής ή εντοπισμού συγκρούσεων (collision detection, καθώς και collision response), ήχο, scripting, animation, τεχνητή νοημοσύνη (A.I), δικτύωση, streaming, διαχείριση μνήμης, νήματα (threading), υποστήριξη τοπικοποίησης, και ένα γράφο σκηνής (scene graph).

Η διαδικασία της ανάπτυξης ενός παιχνιδιού συχνά ελαχιστοποιείται με το γεγονός ότι σε μεγάλο μέρος η ίδια μηχανή παιχνιδιού επαναχρησιμοποιείται για να δημιουργηθούν διαφορετικά παιχνίδια. (GameCarrierGuide, 2008)

Ή αλλιώς θα μπορούσαμε να δώσουμε ένα εναλλακτικό ορισμό :

Μια μηχανή παιχνιδιού είναι ένα λογισμικό, το οποίο παρέχει στους προγραμματιστές τα απαραίτητα εργαλεία για να φτιάξουν ένα παιχνίδι γρηγορά και αποτελεσματικά. (Unity, 2019)

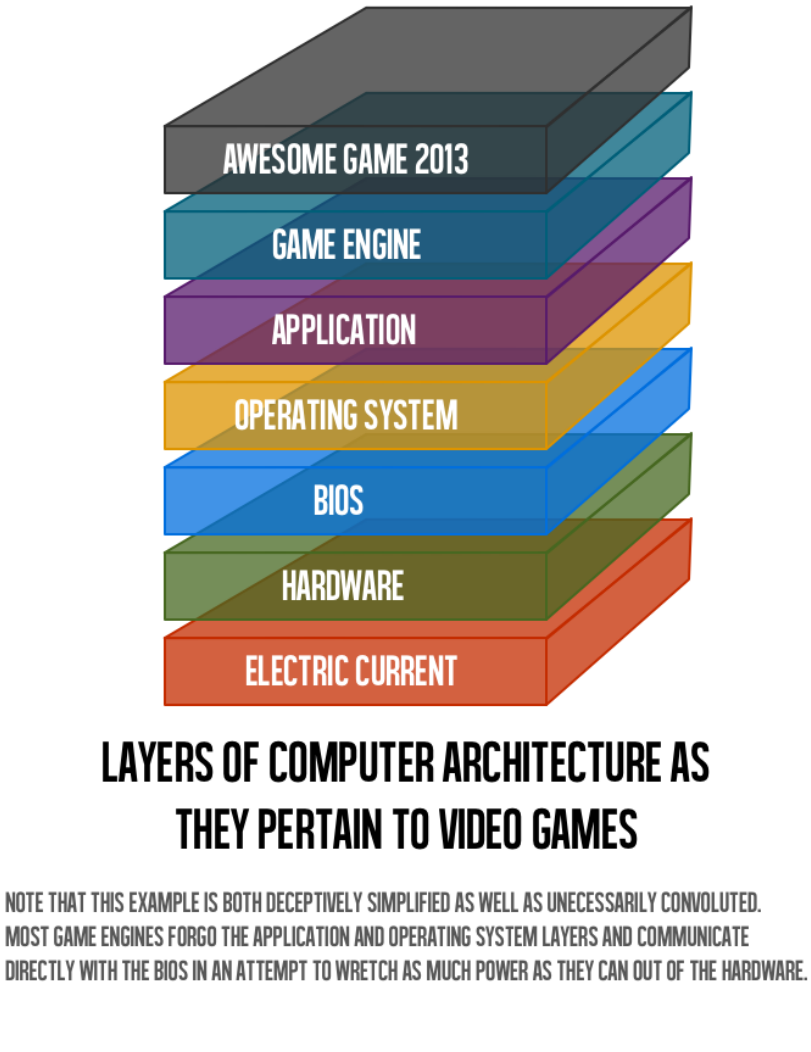


Figure 12 Στρώματα αρχιτεκτονικής

Επισκόπηση

Οι μηχανές παιχνιδιών παρέχουν μια πλειάδα εργαλείων ανάπτυξης εκτός από επαναχρησιμοποιούμενα στοιχεία λογισμικού. Αυτά τα εργαλεία γενικά παρέχονται σε ένα ολοκληρωμένο περιβάλλον ανάπτυξης (IDE), ώστε να βοηθήσουν στην απλή και γρήγορη ανάπτυξη παιχνιδιών. Αυτές οι μηχανές παιχνιδιών ονομάζονται "game middleware" επειδή, όπως με την εμπορική έννοια του όρου, παρέχουν μια ευέλικτη και επαναχρησιμοποιήσιμη πλατφόρμα λογισμικού, η οποία παρέχει όλη την κεντρική λειτουργικότητα που απαιτείται για την ανάπτυξη μια εφαρμογής παιχνιδιού, ενώ ταυτόχρονα μειώνει το κόστος, την πολυπλοκότητα του προγράμματος, καθώς επίσης και τον χρόνο ανάπτυξης σε μια ανταγωνιστική αγορά (The Real cost of Middleware, 2008).

Οι μηχανές παιχνιδιών, συνήθως, επιτρέπουν στο ίδιο το παιχνίδι να τρέχει σε διάφορες πλατφόρμες συμπεριλαμβανομένων των κονσόλων παιχνιδιών και των προσωπικών υπολογιστών με λίγες, αν όχι καθόλου, αλλαγές στον πηγαίο κώδικα του παιχνιδιού.

Παρά την συγκεκριμένη έννοια του ονόματος, οι μηχανές παιχνιδιών συχνά χρησιμοποιούνται για άλλα είδη διαδραστικών εφαρμογών με πραγματικού χρόνου γραφικές απαιτήσεις, όπως επιδείξεις μάρκετινγκ, αρχιτεκτονικές οπτικοποιήσεις, εκπαιδευτικές εξομοιώσεις και περιβάλλοντα μοντελοποίησης (Madison).

Πιο συχνά, οι 3D μηχανές ή τα συστήματα rendering στις μηχανές παιχνιδιών κατασκευάζονται πάνω σε ένα API γραφικών, όπως τα Direct3D ή OpenGL. Βιβλιοθήκες για τα DirectX, SDL και OpenAL επίσης χρησιμοποιούνται συχνά σε παιχνίδια καθώς παρέχουν πρόσβαση ανεξάρτητη από την κάρτα γραφικών σε άλλο υλικό του υπολογιστή, όπως συσκευές εισόδου (ποντίκι, υπολογιστές), πληκτρολόγιο και joystick), κάρτες δικτύου και κάρτες ήχου. Το rendering λογισμικού ακόμα χρησιμοποιείται σε κάποια εργαλεία μοντελοποίησης ή για εικόνες ακίνητα rendered, όταν η οπτική ακρίβεια αποτιμάται πάνω από την επίδοση πραγματικού χρόνου (καρέ ανά δευτερόλεπτο) ή, όταν το υλικό του υπολογιστή δεν συναντά απαιτήσεις, όπως υποστήριξη shader.

Middleware

Με την ευρύτερη έννοια, οι μηχανές παιχνιδιών μπορούν να χαρακτηριστούν και ως middleware.Αλλά στο περιεχόμενο των παιχνιδιών ο όρος Middleware, συνήθως, χρησιμοποιείται σε ξεχωριστές λειτουργίες μέσα στην μηχανή παιχνιδιών.

Μερικές εταιρείες τώρα ειδικεύονται στην ανάπτυξη σουιτών λογισμικού γνωστές ως "middleware". Οι δημιουργοί middleware προσπαθούν να βοηθήσουν τον προγραμματιστή με την ανάπτυξη σουιτών λογισμικού, οι οποίες περιλαμβάνουν πολλά στοιχεία που ένας δημιουργός παιχνιδιού μπορεί να χρειαστεί για να κατασκευάσει ένα παιχνίδι. Τα περισσότερα προγράμματα middleware είναι δομημένα, έτσι που κάνουν εύκολη την ανάπτυξη, όπως λειτουργίες γραφικών, ήχου, Φυσικής και AI (Develop-online.net, 2007).

Μερικά middleware κάνουν μόνο ένα πράγμα, αλλά το κάνουν πιο πειστικά ή πιο αποδοτικά απ' ότι οι γενικού σκοπού μηχανές. Για παράδειγμα,

η SpeedTree χρησιμοποιήθηκε για να κάνει render τα ρεαλιστικά δέντρα και βλάστηση στο παιχνίδι ρόλων The Elder Scrolls IV: Oblivion (Dalmau, 2003).

Figure 12 Το SpeedTree μπορεί να δημιουργήσει δέντρα όλων των τύπων

Μερικά middleware περιέχουν πλήρη πηγαίο κώδικα, άλλα απλά παρέχουν API.

Εξαρτήματα

Μια τέτοια σουίτα για την ανάπτυξη παιχνιδιών αποτελείται από μια ποικιλία διαφορετικών εξαρτημάτων που συμβάλουν στην ανάπτυξη (Engineering, 2016).

Κυρίο πρόγραμμα του παιχνιδιού

Η λογική του παιχνιδιού θα πρέπει να υλοποιηθεί με τον ίδιο τρόπο που κατασκευάζουμε κανονικά προγράμματα, δηλαδή με μερικούς αλγορίθμους. Το κύριο πρόγραμμα είναι διαφορετικό κομμάτι από το rendering,τον ήχο και το input (Engineering, 2016)

Μηχανή Rendering (Rendering Engine)

Η μηχανή που κάνει το rendering,στην ουσία δημιουργεί τα 3D γραφικά και τα animation του παιχνιδιού και αυτό το κάνει με διάφορους μεθόδους που μπορεί να έχει μια Game Engine(rasterization, ray-tracing.) (Engineering, 2016).

Αντί να προγραμματίζεται, να μεταγλωττίζεται και να εκτελείται κατευθείαν στην κεντρική μονάδα επεξεργασίας ή στην κάρτα γραφικών, πιο συχνά οι περισσότερες Μηχανές για Rendering έχουν χτιστεί πάνω σε ένα ή πολλά APIs όπως είναι το Direct3D,OpenGL ή πιο πρόσφατο το Vulkan.Τέτοιες βιβλιοθήκες χρησιμοποιούνται συχνά, γιατί μπορούν να δώσουν πρόσβαση στο παιχνίδι σε άλλους πόρους συστήματος, όπως είναι το ποντίκι πληκτρολόγιο και όχι μόνο.

Οι Μηχανές παιχνιδιών μπορούνε να είναι γραμμένες σε οποιαδήποτε γλώσσα προγραμματισμού θέλουμε, αλλά συνήθως είναι σε C++,C ή JAVA καθώς κάθε γλώσσα

είναι δομικά διαφορετική από άλλη. Κάθε μηχανή παιχνιδιών έχει και πιο ξεχωριστές λειτουργίες ή χαρακτηρίστηκα (Engineering, 2016)

Μηχανή ήχου (Audio engine)

Η Μηχανή του Ήχου είναι ένα πολύ σημαντικό εξάρτημα, το οποίο περιλαμβάνεται από αλγορίθμους που σχετίζονται με το ποσό φορτίζει, μπορεί να τροποποιηθεί και να ακούγεται ο ήχος στον δεκτή. Στην χειρότερη θα πρέπει ο ήχος να φορτώνει, να αποσυμπιέζεται και να αναπαράγει ηχητικά αρχεία. Πιο εξελιγμένες μηχανές ήχου μπορούν να υπολογίσουν και να παράγουν πράγματα, όπως εφέ Doppler,echoes,ενίσχυση ήχου, ταλάντωση και αλλά πολλά.

Οι Μηχανές ήχου μπορούν να κάνουν τους υπολογισμούς τους στην Κεντρική μονάδα Επεξεργασίας (CPU) ή σε ένα συγκεκριμένο ASIC όπως κάρτες ήχου (Wikipedia, Wikipedia, 2016).

Physics engine

Η Physics Engine είναι υπεύθυνη για την ρεαλιστική εξομοίωση των νόμων της φυσικής μέσα στην εφαρμογή, ώστε να φαίνεται πιο αληθινή και πιο κοντά στην πραγματικότητα. Παρέχει ένα σετ από συναρτήσεις και λειτουργίες για την σωστή εξομοίωση των φυσικών δυνάμεων, συγκρούσεων και το πώς συμπεριφέρονται διαφορά αντικείμενα μέσα στην εφαρμογή μας (Bourg, 2002).

Τεχνητή Νοημοσύνη(A.I)

Η Τεχνητή Νοημοσύνη ενός παιχνιδιού συνήθως αναθέτεται από το κύριο πρόγραμμα του παιχνιδιού σε ένα ξεχωριστό κομμάτι, για να γραφτεί και να σχεδιαστεί από προγραμματιστές που έχουν συγκεκριμένη γνώση από τεχνητή νοημοσύνη. Κάθε παιχνίδι εισάγει μια δικιά του Τεχνητή Νοημοσύνη, λόγω της διαφορετικότητας των τίτλων. Έτσι κάθε τεχνητή νοημοσύνη σε κάθε παιχνίδι θεωρείται πως είναι συγκεκριμένη για κάθε ξεχωριστό τίτλο. Πολλές σύγχρονες Μηχανές παιχνιδιών έρχονται έτοιμες με βιβλιοθήκες που έχουν αλγορίθμους τεχνητής νοημοσύνης, όπως ο A-star,Dijkstra κλπ. για να βοηθήσουν την συμπεριφορά της τεχνητής νοημοσύνης μέσα στο παιχνίδι (Wikipedia, Wikipedia, 2016)

Μοντέρνες τάσεις

Καθώς η τεχνολογία των μηχανών παιχνιδιών ωριμάζει στο σύγχρονο κόσμο, γίνεται πιο φιλική προς τον χρήστη. Οι εφαρμογές των μηχανών παιχνιδιών είναι ποίκιλες εκτός από παιχνίδια και μπορούν να βρεθούν σε οπτικοποιήσεις, στην εκπαίδευση, στην ιατρική αλλά και σε στρατιωτικές εξομοίωσης εφαρμογές. (Gazette.net, 2007)

Για να διευκολύνουν αυτή την προσβασιμότητα οι μηχανές παιχνιδιών στοχεύουν τώρα νέες πλατφόρμες, όπως κινητά τηλεφωνά (π.χ. το iPhone) και εφαρμογών φυλλομετρηθώ(π.χ. Shockwave, Flash, Silverlight, Unity Web Player, O3D). (M-trends.org, 2011)

Επιπρόσθετα, περισσότερες μηχανές παιχνιδιών αναπτύσσονται πάνω σε γλώσσες υψηλού επιπέδου όπως η Java και η C#/.NET (π.χ. TorqueX, Blade3D, και

Visual3D.NET) ή Python (Panda3D). Καθώς τα περισσότερα 3D παιχνίδια είναι περισσότερο εξαρτώμενα από την GPU (δηλαδή περιορισμένα από την ισχύ της κάρτας γραφικών). (Stefan Zerbst, Oliver Düvel, 2004)

Σαν γλώσσες υψηλού επιπέδου, φυσικά υπάρχει και το πρόβλημα της επίδοσης, καθώς οι υψηλές γλώσσες προγραμματισμού όπως η JAVA έχουν διάφορα εξαρτήματα, όπως ο Garbage Collector και άλλους αυτοματισμούς οι οποίοι ναι μεν διευκολύνουν τον προγραμματιστή, αλλά δε μειώνουν την απόδοση του προγράμματος. Βέβαια, οι ενδεχόμενες καθυστερήσεις των υψηλού επιπέδου γλωσσών προγραμματισμού γίνονται αμελητέες, καθώς τα οφέλη παραγωγικότητας που προσφέρονται από αυτές τις γλώσσες στους δημιουργούς μηχανών παιχνιδιών γίνονται κέρδος. (Krause, 2007)

Προγραμματισμός σε Νήματα

Ο προγραμματισμός με νήματα γίνεται όλο και πιο αναγκαίος και σημαντικός λόγω των μοντέρνων πολυπύρηνων συστημάτων και των αυξημένων απαιτήσεων σε ρεαλισμό στα γραφικά. Τα νήματα χωρίζουν τον φόρτο εργασίας ανά νήμα, κάποια νήματα αναλαμβάνουν το rendering, το streaming, τον ήχο, και τη Φυσική.

Τα αγωνιστικά παιχνίδια ήταν τα πρώτα παιχνίδια του προγραμματισμού με νήματα με τη μηχανή Φυσικής να τρέχει ένα ξεχωριστό νήμα, πριν άλλα υποσυστήματα του πυρήνα μετακινηθούν στα νήματα του επεξεργαστή, και αυτό γινόταν, γιατί το rendering και άλλες εργασίες απαιτούν ενημέρωση στα 30-60 Hz.

Μηχανές παιχνιδιών FPS (First Person Shooters)

Ένα πολύ γνωστό υποσύνολο των μηχανών παιχνιδιών είναι οι μηχανές των παιχνιδιών πρώτου προσώπου (FPS). Πρωτοποριακή ανάπτυξη έχει σημειωθεί τα τελευταία χρονιά για την οπτική ποιότητα για να φτάσουν τα FPS στα σημερινά δεδομένα.

Ενώ σε παιχνίδια με εξομοιωτές πτήσης και οδήγησης, καθώς και τα παιχνίδια στρατηγικής πραγματικού χρόνου(RTS) παρέχουν αυξανόμενο ρεαλισμό σε μια μεγάλη κλίμακα, τα παιχνίδια πρώτου προσώπου(FPS) έχουν ακόμα μεγαλύτερη ακρίβεια τόσο σε επίπεδο ρεαλισμού όσο και στην ποιότητα της εικόνας.

Η ανάπτυξη των μηχανών γραφικών FPS μπορεί να χαρακτηριστεί από μια σταθερή αύξηση σε τεχνολογίες έχοντας κάποιες καινοτομίες. Πολλές φορές από μια παλιά γενιά μηχανών παιχνιδιών, μέσα από πολλές τροποποιήσεις, οδηγούμαστε σε μια καινούργια μηχανή παιχνιδιών. (Hauteville, 2011)

Η κατάταξη είναι πολύπλοκη καθώς οι μηχανές παιχνιδιών συνδυάζουν παλιές και νέες τεχνολογίες. Χαρακτηριστικά που θεωρούνται προχωρημένα σε ένα νέο παιχνίδι τον ένα χρόνο, γίνονται αναμενόμενα πρότυπα τον επόμενο χρόνο. Παιχνίδια με ένα μείγμα παλιάς γενιάς και νεότερων χαρακτηριστικών είναι η νόρμα. Για παράδειγμα το Jurassic Park: Trespasser (1998) εισήγαγε την Φυσική στα παιχνίδια FPS, αλλά αυτό δεν έγινε κοινό μέχρι το 2002.

Figure 13 First Person Shooter Game

Οι μηχανές παιχνιδιών στην βιομηχανία

Οι παραγωγοί των μηχανών παιχνιδιών αποφασίζουν πως θα αφήσουν τους χρήστες να χρησιμοποιήσουν τα προϊόντα τους. Όπως η βιομηχανία των παιχνιδιών έχει χτιστεί, έτσι και η βιομηχανία των παιχνιδιών είναι χτισμένη έτσι ώστε πιο καλές μηχανές έρχονται σε ανάλογες τιμές, είτε είναι σε κάποια μορφή συνδρομής ή σε μορφή να αγοράς αδειών. (The 10 Best Video Game Engines | 2018 Edition, 2017)

Μία από τις πιο επιτυχημένες μηχανές παιχνιδιών που έχουν χρησιμοποιηθεί για να φτιάξουν πολύ γνωστά και επιτυχημένα παιχνίδια όπως είναι το Fortnight, Player Unknown's Battlegrounds, and Life is Strange 2, είναι το Unreal Engine 4.Αυτη η μηχανή είναι χρησιμοποιείται χωρίς χρέωση, αλλά όταν λανσάρεις το παιχνίδι σου θα πρέπει να πληρώσεις ένα ποσοστό από το μεριδίων των πωλήσεων στην μηχανή. (Savage, Phil, 2015)

Παρόλο που οι διαφορές μεταξύ των διαφορετικών μηχανών παιχνιδιών ξεχνιούνται καθώς αναπτύσσεται ένα παιχνίδι, περισσότεροι προγραμματιστές παιχνιδιών είναι αρκετά εξοικειωμένοι με μια μηχανή παιχνιδιών και έτσι δυσκολεύονται να αλλάξουν.

Μια άλλη γνωστή μηχανή παιχνιδιών είναι η Unity Engine.Το Unity χρησιμοποιεί τον ίδιο τρόπο όπως η Unreal Engine για να βγάλει χρήματα, δηλαδή είναι ελεύθερη για χρήση και κερδίζει χρήματα από τις πωλήσεις ενός παιχνιδιού που είναι χτισμένο πάνω σε αυτό. Γνωστά παιχνίδια πάνω στο Unity είναι: Rust, Sub Nautica και Life is Strange Before the Storm. Μεταξύ άλλων μηχανών παιχνιδιών, θεωρείται πως το Unity Και το Unreal Engine είναι οι καλύτερες μηχανές και πολύ συχνά συγκρίνονται και θεωρούνται ανταγωνιστές στην βιομηχανία ανάπτυξης παιχνιδιών (Insights, 2018).

Figure 14 Unity and Unreal Engine 4

Κεφάλαιο 3ο:Unreal Engine

Τι είναι το Unreal Engine

Το Unreal Engine είναι μια μηχανή παιχνιδιών που αναπτύχθηκε από την Epic Games και πρωτοεμφανίστηκε το 1998 στο FPS παιχνίδι Unreal.Παρόλο που η συγκεκριμένη μηχανή αναπτύχθηκε για First-Person-Shooters(FPS) παιχνίδια, έχει χρησιμοποιηθεί και αναπτύξει με επιτυχία διαφορά παιχνίδια διαφορετικού είδους όπως Fighting Games,MMORPGs κλπ.

Το Unreal είναι μια μηχανή που έχει γραφτεί και έχει βασιστεί σε C++. Το Unreal παρέχει έναν μεγάλο βαθμό φορητότητας και εργαλείων σαν IDE Περιβάλλον. Επίσης, επιτρέπει την ανάπτυξη παιχνιδιών σε διαφορετικές πλατφόρμες από PC, κονσόλες μέχρι και κινητά. Για αυτούς τους λογούς το Unreal είναι τόσο διαδεδομένο αναμεσά στους προγραμματιστές παιχνιδιών.

Μια δυνατότητα που δίνει το Unreal Engine είναι ότι αφήνει στον προγραμματιστή τον τρόπο που θα αναπτύξει ένα παιχνίδι. Πιο έμπειροι προγραμματιστές μπορούν να χρησιμοποιήσουν κατευθείαν C++ γράφοντας τα δικά τους script τροποποιώντας επιπλέον, εάν θέλουν, και μέρη της μηχανής, ενώ πιο ερασιτέχνες μπορούν να χρησιμοποιήσουν scripting για να αναπτύξουν το παιχνίδι.

Η πιο πρόσφατη έκδοση της μηχανής είναι το Unreal Engine 4.0 που κυκλοφόρησε το 2014 (Denham, 2016).

Διάσημα και πετυχημένα παιχνίδια αναπτυγμένα με Unreal Engine είναι:

1. Fortnite
2. Rocket League
3. Darksiders 3
4. Kingdom Hearts 3
5. Dragon Ball Fighter-z

Figure 15 Unreal Engine C++ Using Pragma for pararrel programming

Σχετικά με την ανάπτυξη του Unreal Engine

Η πρώτη γενιά του Unreal Engine είχε αναπτυχθεί από τον Tim Sweeney τον ιδρυτή της Epic Games την εταιρία που έχει το Unreal (Tim, 2005).

Έχοντας μεγάλη εμπειρία σαν προγραμματιστής από μικρή ηλικία, ο Sweeney ξεκίνησε να γράφει την μηχανή περί το 1995 για την ανάπτυξη ενός παιχνιδιού πρώτου προσώπου (FPS) με το όνομα Unreal που τελικά ανακοινώθηκε το 1998 έχοντας τον Sweeney να έχει γράψει το 90% του προγράμματος της μηχανής (Keighley, Geoffrey., 2001).

Ανάμεσα στα χαρακτηριστικά του ήταν και το Collision detection, Colored Lighting και μια περιορισμένη μορφή Texture Filtering (Lily, Paul, 2009). Η μηχανή, επίσης, ενσωμάτωνε ένα εργαλείο για σχεδιασμό των επίπεδων, το UnrealEd, το οποίο παρείχε υποστήριξη σε πραγματικού χρόνου επεξεργασία γεωμετρίας και, επομένως, έδινε την δυνατότατά στους σχεδιαστές να σχεδιάζουν τα επίπεδα και να κάνουν αλλαγές πιο αποτελεσματικά και άμεσα (Thomsen, 2012).

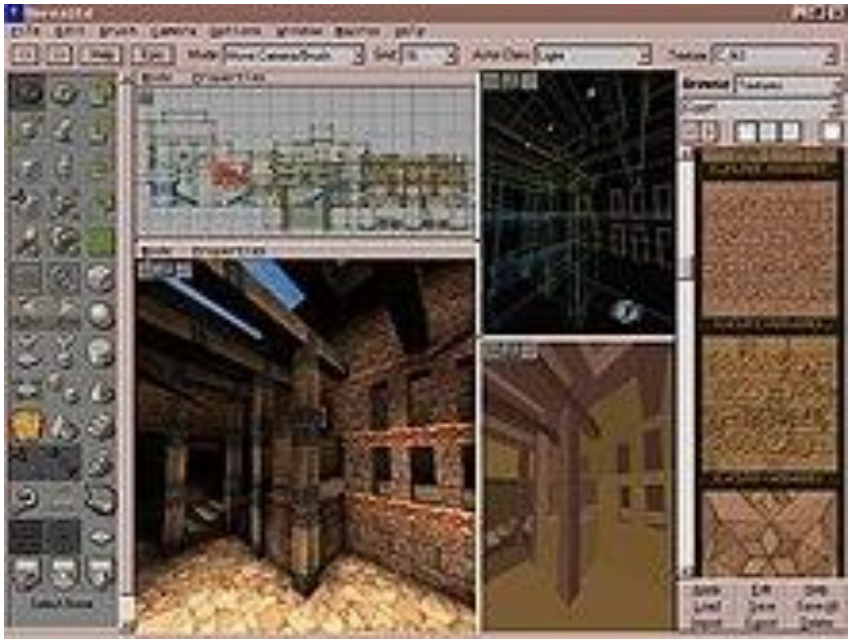


Figure 16 UnrealEd User Interface γραμμένο σε Visual Basic

Στην αρχή η μηχανή βασιζόταν σε software rendering, που αυτό σημαίνει ότι οι υπολογισμοί που πρέπει να γίνουν για τα γραφικά γινόντουσαν από την κεντρική μονάδα επεξεργασίας(CPU) αντί της κάρτα γραφικών (GPU). Ωστόσο, μετά από λίγο καιρό ο Sweeney κατάφερε να ξαναγράψει αρκετούς αλγορίθμους για rendering, ώστε η μηχανή να εκμεταλλευτεί τα πλεονεκτήματα των καρτών. Ως αποτέλεσμα υπήρχε και Software και Hardware rendering στην μηχανή, καθώς μπορούσε να υποστηρίξει και το OpenGL και το Direct3D (Tim, 2005).

Όσο αναφορά τον ήχο, η Epic προσέλαβε την Galaxy Sound System, η οποία ανάπτυξε ένα πρόγραμμα γραμμένο σε Assembly, το οποίο υποστήριζε και 8-bit και 16-bit ήχο μαζί με EAX και Aurea δυνατότητες (Brandon, 2004).

Επιπλέον η μηχανή ήταν διαθέσιμη για τα Windows, τα Linux, τα Mac και τα Unix. Επίσης, η μηχανή μεταφέρθηκε και στις κονσόλες χάρη στο παιχνίδι Unreal Tournament στο PlayStation 2 και στο Dream Cast. (Herz, 1999).

Όπως αναφέρει ο κύριος Sweeney, και το παραθέτω χαρακτηριστικά χωρίς να γίνει μετάφραση :

“ The big goal with the Unreal technology all long was to build up a base of code that could be extended and improved through many generations of games. Meeting that goal required keeping the technology quite general-purpose, writing clean code, and designing the engine to be very extensible. The early plans to design an extensible multi-generational engine happened to give us a great advantage in licensing the technology as it reached completion. After we did a couple of licensing deals, we realized it was a legitimate business. Since then, it has become a major component of our strategy.

(Sweeney, 1998)

Unreal Engine 2

Τον Οκτώβριο του 1998, η IGN ανάφερε ότι ο Sweeney έκανε έρευνα για την επόμενη μηχανή καινούργιας γενιάς που θα ανέπτυσσε (Staff, Tim Sweeney Looks Ahead, 2012). Με την ανάπτυξη να ξεκινάει αργότερα εκείνο τον χρόνο, η δεύτερη έκδοση της μηχανής έκανε την εμφάνιση της το 2002 με το παιχνίδι American’s Army ένα Multiplayer παιχνίδι που είχε αναπτυχθεί από το σώμα στρατού της Αμερικής σαν στρατηγική στρατολόγησης (Kennedy, 2002).

Figure 7 Killing Floor in Unreal Engine 2.

Παρόλο που η δεύτερη έκδοση της μηχανής είναι βασισμένη στην προηγουμένη, η δεύτερη γενιά είδε πολλές προόδους στο Rendering, καθώς και αρκετές βελτιώσεις και στα εργαλεία της (McLean-Foreman, 2001). Έκτος από ένα ολοκαίνουργιο γραμμένο Renderer που μπορούσε να πράξει επίπεδα περισσότερο λεπτομερή, περίπου 100 φορές από το προηγούμενο, η μηχανή είχε επίσης νέα χαρακτηριστικά, όπως το Skeletal Animation σύστημα το οποίο χρησιμοποιήθηκε στο PlayStation 2. Σε αυτήν την έκδοση στο UnrealEd το Framework του ξαναγράφτηκε χρησιμοποιώντας την C++ όντας μια πιο αποδοτική γλώσσα σε σχέση με την Visual Basic (Lightbown, 2018).

Unreal Engine 3

Το Unreal Engine 3 ξεκίνησε να διαφημίζεται από το 2004, που από τότε ήταν ήδη υπό ανάπτυξη για 18 μήνες (Reed, 2004).Λόγο του τρόπου που ήταν γραμμένη η μηχανή (Modular), το Unreal Engine 3 περιείχε κώδικα και από το πρώτο Unreal Engine. Οι βασικές αρχιτεκτονικές αποφάσεις ήταν ορατές στους προγραμματιστές σε ένα Object-oriented σχεδιασμό, καθώς ο προγραμματισμός με βάση τα δεδομένα και τη προσέγγιση που γίνεται στα υποσυστήματα, παρέμενε το ίδιο. Αλλά στα κομμάτια των παιχνιδιών που είναι ορατά στους καταναλωτές, όπως το Renderer, το Physics Engine, το σύστημα του ήχου και τα εργαλεία, είναι όλα καινούργια και ραγδαία πιο δυνατά. Στο Unreal Engine 3 όλοι οι υπολογισμοί για τον φωτισμό και τις σκιές γίνονταν για κάθε ένα Pixel ξεχωριστά. Στο Rendering το Unreal Engine 3 πρόσφερε υποστήριξη για ένα Gamma-correct-High-Dynamic Range Renderer. (Maximum, 2004)

Τα πρώτα παιχνίδια που ανακοινωθήκαν με το Unreal Engine 3 ήταν το Gears Of War για την πλατφόρμα του Xbox 360 και το RoboBlitz για τα Windows (Caron, 2008).

Figure 18 Tera Developed using Unreal Engine 3.

Αρχικά το Unreal Engine 3 θα υποστήριζε τα Windows, PlayStation 3, Xbox 360.Αλλα στην συνέχεια το Android και το iOS προσδέθηκαν το 2010. Με το παιχνίδι Infinity Blade να είναι το πρώτο παιχνίδι στο iOS αναπτυγμένο με Unreal Engine 3, το 2011 η Epic ανακοίνωσε πως η μηχανή θα παρείχε υποστήριξη για Adobe Flash Player 11 καθώς και το 2013 η Epic μαζί με την Mozilla συνεργάστηκαν για να αναπτύσσουν μια έκδοση του Unreal Engine 3 να δουλεύει και στα Browsers με την χρήση της HTML5 (Ligman, 2013).

Unreal Development Kit

Καθώς το Unreal Engine 3 ήταν διαθέσιμο στους modders (άτομα που αλλάζουν την νοοτροπία ενός παιχνιδιού είτε γραφικά ειτε στην προγραμματιστική λογική) για να μπορούν να δουλέψουν πάνω σε αυτό, η δυνατότητα να μπορείς να εκδόσεις και να πουλάς παιχνίδια χρησιμοποιώντας την συγκεκριμένη μηχανή ήταν πολύ περιορισμένη, χάρις στις άδειες που είχε η μηχανή. Παρόλο αυτά, το Νοέμβριο του 2009 η Epic κυκλοφόρησε μια ελεύθερη έκδοση του Unreal Engine 3 SDK με το όνομα Unreal

Development Kit (UDK), το οποίο ήταν διαθέσιμο για το κοινό (Staff, Epic Games Announces Unreal Development Kit, Powered by Unreal Engine 3, 2012).

Το Δεκέμβριο του 2010, το συγκεκριμένο kit ενημερώθηκε από την Epic για να υποστηρίζει και να δημιουργεί παιχνίδια στο iOS (Staff, Epic Games Releases Unreal Development Kit with iOS Support, 2012).

Unreal Engine 4

Τον Αύγουστο του 2005 ο Mark Rein, ο αντιπρόεδρος της Epic Games, ανακοίνωσε ότι το Unreal Engine 4 ήταν ήδη υπό ανάπτυξη από το 2003 (Rein,Houlihan, John, 2005).Μέχρι το 2008 η ανάπτυξη της μηχανής είχε γίνει «βασικά» από τον Sweeney. Το 2012 η Epic αποκάλυψε στο Game Developers conference το Unreal Engine 4 σε λίγους καλεσμένους πρώτη επίδειξη της μηχανής έγινε από τον τεχνικό καλλίτεχνι Alan "Talisman" τον Ιούνιο του 2012 στο Game Trailers TV σε μορφή βίντεο (Parrish, 2005).

Figure 18 Logo του Unreal Engine 4

Ένα από τα πιο βασικά χαρακτηριστικά σχεδιασμένα για το Unreal Engine 4 ήταν αληθινού χρόνου Global Illumination χρησιμοποιώντας Voxel Cone Tracing.Το Global illumination είναι ένα σετ από αλγορίθμους που χρησιμοποιούνται στα 3D γραφικά για τον φωτισμό στις σκηνές για να φαίνονται πιο ρεαλιστικές (Thomas, 2012). Το Unreal Engine 4 επίσης προσέφερε καλύτερη απόδοση και στο τρέξιμο της C++ και της μεταγλώττισης στον χρόνο. Το Unreal Engine 4 εφηύρε το concept των "Blueprints" τα οποία επέτρεπαν την ανάπτυξη ενός παιχνιδιού χρησιμοποιώντας ένα σύστημα scripting το οποίο ήταν πολύ πιο γρήγορο από τον παραδοσιακό τρόπο, καθώς μείωνε την δυσκολία και τον χρόνο που θα χρειάζονταν οι προγραμματιστές, οι σχεδιαστές και οι καλλιτέχνες για να αναπτύξουν ένα παιχνίδι μόνο χρησιμοποιώντας την C++ στην μηχανή. Τα Blueprint στην ουσία είναι βασισμένα στην C++, απλά διευκολύνουν τους προγραμματιστές (Thier, 2012) (Totilo, 2012).

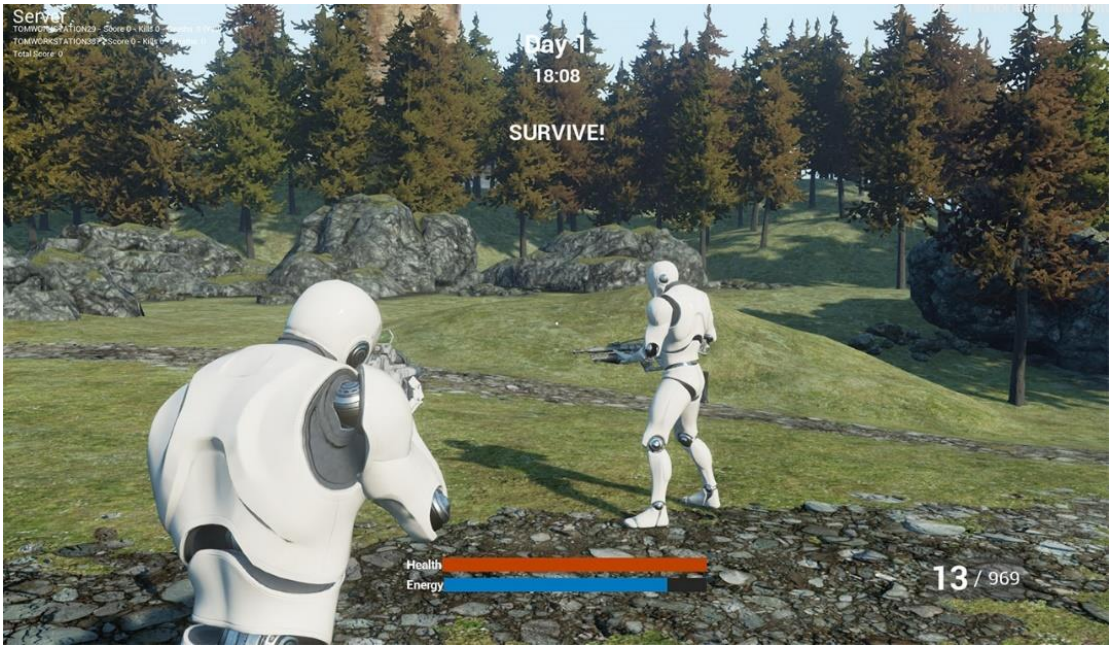


Figure 19 Ένα απλό παιχνιδι από έναν ερασιτέχνη προγραμματιστή βασισμένο στο Unreal Engine 4.

Στις 19 Μάρτιου το 2014, πάλι στο Game Developers Conference, η Epic Games επιτέλους ανακοίνωσε την άφιξη του Unreal Engine 4 με ένα νέο σύστημα αδειών, πιο ευέλικτο. Με 19 δολάρια το μηνά οι προγραμματιστές είχαν πλήρη πρόσβαση στην μηχανή συμπεριλαμβανομένου και τον πηγαίο κώδικά της, που ήταν σε C++ το οποίο μπορούσαν να κατεβάσουν από το GitHub. Επιπλέον, κάθε καινούργιο παιχνίδι που είχε αναπτυχθεί με Unreal Engine 4 και έβγαινε στη αγορά η Epic έπαιρνε μόνο το 5% από τις πωλήσεις. Το πρώτο Παιχνίδι που ανακοινώθηκε χρησιμοποιώντας το Unreal Engine 4 ήταν το Daylight (unrealengine, Zombie Studios Licenses Unreal Engine 4, 2012) (PCGamesN, 2014).

Τον Σεπτέμβριο του 2014, η Epic κυκλοφόρησε το Unreal Engine 4 σε σχολεία και σε πανεπιστήμια χωρίς χρέωση, συμπεριλαμβάνοντας προσωπικές άδειες σε κάθε φοιτητή που ήταν εγγεγραμμένος σε μαθήματα ανάπτυξης παιχνιδιών, Computer Science, Animation και Simulation (Sykes, 2014).

Το 2015 η Epic πάλι στο Game Developers Conference ανακοίνωσε ότι θα κυκλοφορήσει το Unreal Engine 4 μαζί με όλες τις επόμενες ενημερώσεις του, χωρίς καμία χρέωση για τους χρήστες. Ακόμα, ανακοίνωσε ότι θα ζητάει 5% για τα παιχνίδια που θα καταφέρνουν να βγάζουν πάνω από 3000 δολάρια το 4μηνο, ενώ παράλληλα ανακοίνωσε και την δημιουργία της ψηφιακής αγοράς του Unreal Engine, το οποίο θα βοηθούσε τους χρήστες να αγοράζουν κομμάτια ή μοντέλα και όχι μόνο για τα παιχνίδια τους από άλλους χρήστες. Παρατηρήθηκε ότι με αυτήν την αλλαγή η χρήση του Unreal σαν μηχανή παιχνιδιών δεκαπλασιάστηκε (Gaudiosi, 2015) (Devin, Connors, 2014).

Με την κυκλοφορία του Παιχνιδιού Fortnite τον Σεπτέμβριο του 2017,η Epic έκανε κάποιες ουσιώδες αλλαγές στο Unreal Engine στο οποίο επέτρεπε η μηχανή να σηκώνει πάνω από 100 συνδέσεις σε ένα Server διατηρώντας το ίδιο υψηλό Bandwidth (McAloon, 2017).

Το 2018 η Epic το πήγε ένα βήμα παραπάνω και άνοιξε μια δικιά της πλατφόρμα για αγορά παιχνιδιών, το οποίο δεν θα χρέωνε το 5% που ζητούσε πιο πριν από τις πωλήσεις των παιχνιδιών που ήταν αναπτυγμένα σε unreal engine, αρκεί να είχαν κυκλοφορήσει σε αυτήν την νέα και δικιά τους πλατφόρμα (Frank, 2018).

Marketplace του Unreal Engine 4

Μαζί με το Unreal Engine 4,η Epic άνοιξε και ένα Marketplace μέσα στο Unreal Engine 4. Στην ουσία, ήταν ένα ψηφιακό μαγαζί το οποίο επέτρεπε στους δημιουργούς περιεχομένων, όπως Animations,Arts, μοντέλων, ήχων, περιβλεπόντων ακόμα και κομματιών κώδικα να μπορούν να πουλάνε το περιεχόμενο τους και άλλοι χρήστες να το αγοράζουν, ώστε να το ενσωματώσουν στα παιχνίδια τους. Σε αυτήν την ψηφιακή αγορά υπάρχουν και αντικείμενα, τα οποία είναι ελεύθερα για να τα κατεβάσει ο χρήστης και να τα ενσωματώσει στα δικά του παιχνίδια. Επίσης, ακόμα και η Epic πρόσφερε περιεχόμενο χωρίς καμία χρέωση, όπως Assets και Tutorials. Επιπλέον, σε αυτή την αγορά υπάρχουν και διάφορα Tutorials και οδηγίες σχετικά με την ανάπτυξη παιχνιδιών ή διάφορων τεχνικών (Brown, 2014).

Εργαλεία του Unreal Engine 4

1. Rendering και γραφικά: Το σύστημα του Render, οι φωτισμοί, τα γραφικά σκιές και τα εφέ.
2. UMG UI Designer: Χρησιμοποιώντας υλικά του Unreal για να φτιάξεις User Interface.
3. Skeletal Mesh Animation System: Μοντέλα «ηθοποιών» και Animations
4. Niagara Editor: Visual Effects
5. Sequence Editor: Σύστημα για να φαίνονται οι σκηνές πιο Σινεματικές.
6. Audio and Sound.
7. Physics Simulation: Ρεαλιστική εξομοίωση φυσικής στο παιχνίδι μας
8. Open World Tools: Εργαλεία για Παιχνίδια με μεγάλα περιβάλλοντα.
9. Landscape Outdoor Terrain: Σύστημα για την δημιουργία μεγάλων περιβαλλόντων, όπως βουνών.
10. Foliage Tool: Σύστημα για την ομοιόμορφη γεωμετρία στις επιφάνειες.
11. Level Streaming: Ασύγχρονα το πώς φορτώνει και εκφορτώνει διάφορα επίπεδα του παιχνιδιού με αποδοτικό τρόπο στους πόρους.
12. Hieratical Level Design.
13. «Blueprints».
14. Matinee and Cinematics: Σύστημα που επιτρέπει πιο ρεαλιστικά Animation στους χαρακτήρες.
15. Media Framework.
16. Replay System: Recording και αναπαραγωγή του Gameplay.
17. Performance and profiling: Σύστημα που αντιμετωπίζει προβλήματα με την απόδοση.
18. Packaging and Cooking Games: Σύστημα που πακετάρει το παιχνίδι και το ετοιμάζει για την συγκεκριμένη πλατφόρμα που αναπτύχθηκε.
19. Networking and multiplayer.
20. Paper 2D: Σύστημα ανάπτυξης για 2D.

21. Assets and Packages: Από τα πιο σημαντικά εργαλεία, καθώς μπορούν να βρεθούν διάφορα Assets και πακέτα που να έχουν να κάνουν με γραφικά, με μοντέλα κλπ. και βοηθούν στην ανάπτυξη του παιχνιδιού.
22. Coordinate Space Technology: Εργαλείο για το χώρο μέσα στο παιχνίδι.
23. Screenshots.
24. Distributions of Values.
25. Redirectors: Σύστημα το οποίο απεικονίζει ποια αντικείμενα είναι χρησιμοποιημένα σε ποια πακέτα.
26. Collaboration of Unreal Engine 4: Να μπορείς να μοιράζεσαι δικά σου Assets με άλλους χρήστες.
27. Artificial intelligence.
28. Build-in-Plugins
29. Professional Video I/O: Περιέχει πληροφορίες για την ποιότητα των βίντεο.
30. Proxy Geometry Tool: Εργαλείο το οποίο βοηθάει τα αντικείμενα μέσα στο παιχνίδι να κρατάνε την ποιότητα τους, χωρίς να επηρεάζουν την απόδοση της μηχανής.
    (unrealengine, Engine Features, 2019)

Unreal Engine Vs Unity

Unreal Engine vs Unity, είναι το μεγαλύτερο debate μεταξύ των προγραμματιστών παιχνιδιών και τον σχεδιαστών παιχνιδιών. Χωρίς καμία αμφιβολία δύο από τα πιο δημοφιλέστερα Game Engines, τα οποία είναι διαθέσιμα. Το Unity και το Unreal χρησιμοποιούνται από μεγάλα studio, αλλά και από ερασιτέχνες προγραμματιστές, αλλά τί είναι αυτό τα κάνει να διαφέρουν; Και πού το ένα είναι καλύτερο από το άλλο;

Υπάρχουν 4 βασικά σημεία που θα πρέπει να σκεφτεί αυτός που έχει να επιλέξει μεταξύ των δυο.

1. Σε τι επίπεδο θα είναι τα γραφικά που θέλουμε να πέτυχουμε;

Ένα από τα πιο σημαντικά στοιχεία που τα διαφοροποιούν είναι το επίπεδο των γραφικών. Το Unreal προσφέρει υψηλής ποιότητας γραφικά βοηθώντας τον προγραμματιστή να μην παιδευτεί πολύ, ενώ, βέβαια, και το Unity μπορεί να προσφέρει τις ίδιας ποιότητα γραφικά, χρειάζεται πολύ παραπάνω δουλειά για να τροποποιήσεις τα μοντέλα σου και τα πακέτα σου να φαίνονται το ίδιο, όπως το Unreal και ακόμα και τότε δεν θα έχει την ίδια ποιότητα.

Για αυτό το λόγο τα «Μεγάλα» παιχνίδια από τα μεγάλα στούντιο αποφασίζουν να πάνε με το Unreal. Για αυτό, εάν κάποιος θέλει ένα φωτορεαλιστικό παιχνίδι όσο πιο ευκολά και γρηγορά γίνεται, μπορεί να γίνει πιο εύκολα με το Unreal.

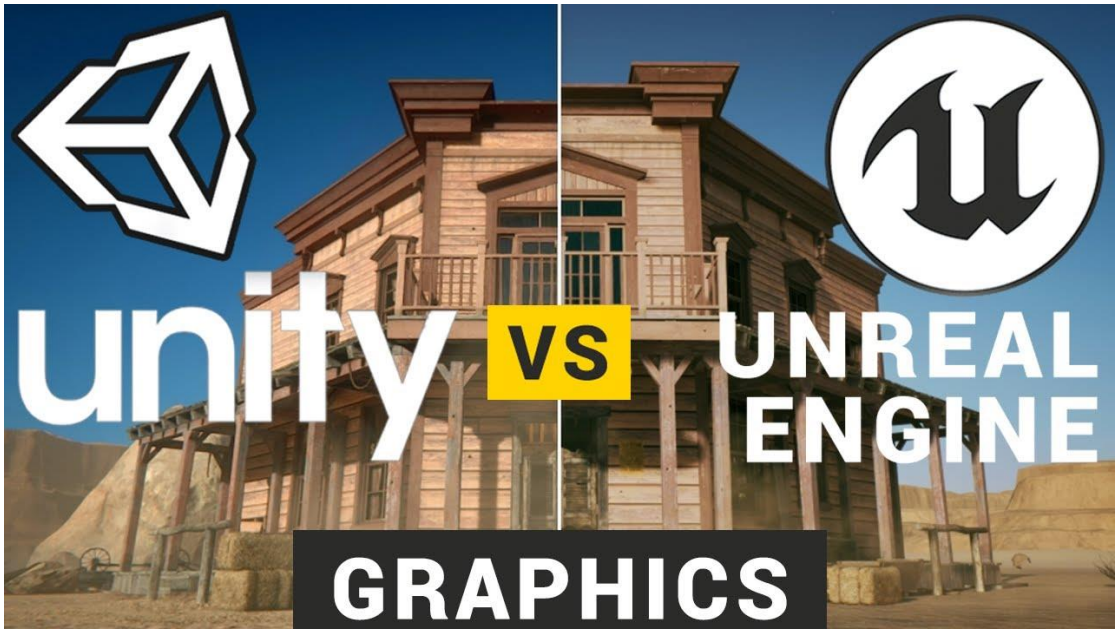


Figure 20 Unity Vs Unreal Graphics

2. *Σε τι συσκευές απευθύνεται το παιχνίδι;*

Εξαρτάται πού σκεφτόμαστε να δημιουργήσουμε ένα Project· ένα Project, το οποίο θα θέλουμε να τρέχει σε χαμηλότερης απόδοσης μηχανές, όπως τα Smartphone, τότε η υψηλή επεξεργαστική ισχύ που απαιτεί το Unreal δεν θα μας βοηθήσει. Εδώ είναι που έρχεται το Unity. Αρχικά σχεδιασμένο για να τρέχει σε Smartphone και κονσόλες, το Unity μας επιτρέπει να δημιουργούμε περίπλοκα παιχνίδια και να τρέχουν χωρίς κανένα πρόβλημα σε χαμηλής επεξεργαστικής δύναμης μηχανές.

3. *Εάν υπάρχει μια ομάδα από πίσω ή όχι;*

Η επικρατούσα άποψη είναι ότι για να πάρεις το καλύτερο αποτέλεσμα χρειάζεσαι Unreal Engine, αλλά για να το καταφέρει κάποιος αυτό, χρειάζεται μια μεγάλη ομάδα που ο καθένας μέσα στην ομάδα να έχει ειδική γνώση για το σημείο το οποίο θα αναπτύξει. Πχ ένας Προγραμματιστής για τον Κώδικα και ένας Σχεδιαστής για τα Animation.

Το Unity από την άλλη μεριά, είναι μια πλατφόρμα πιο εύκολη να κατασκευάσεις παιχνίδια χωρίς να χρειάζεται απαραιτήτως κάποια ομάδα. Είναι μια πολύ καλή επιλογή για κάποιον που θέλει να αναπτύξει ένα παιχνίδι μονός του ή ακόμα και για μικρές ομάδες .Το ψηφιακό κατάστημα του Unity παρέχει παραπάνω Assets από του Unreal και αυτό κάνει πιο εύκολο στους προγραμματιστές να αναπτύξουν ένα παιχνίδι, χωρίς να χρειάζεται μεγάλη ομάδα.

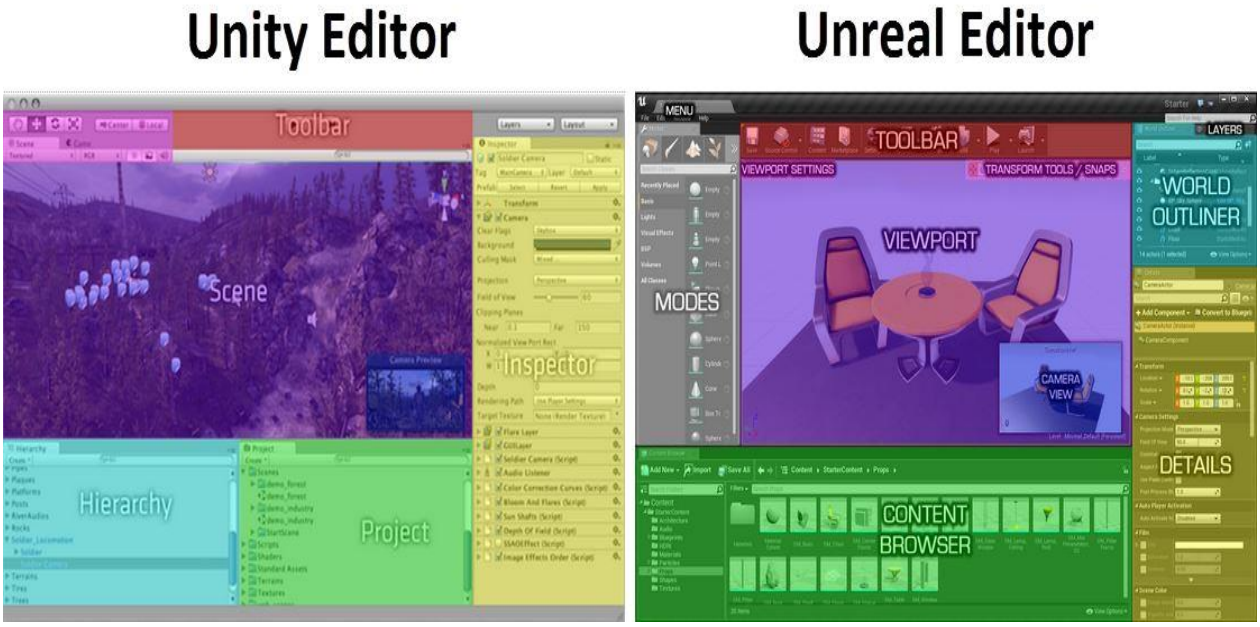


Figure 21 Unity Editor Vs Unreal Editor

4. *Απευθύνεται σε προγραμματιστές ή σε Σχεδιαστές* ;

Χωρίς καμία αμφιβολία, είναι ένα από τα πιο σημαντικά πράγματα που ρυθμίζουν αυτήν την προτίμηση μεταξύ των δυο. Οι προγραμματιστές προτιμούν το Unity για τον τρόπο προγραμματισμού ενώ οι σχεδιαστές προτιμούν το Unreal, επειδή προσφέρει καλύτερα γραφικά. Γενικά, όμως, και οι δυο μηχανές πάνω κάτω προσφέρουν τις ιδίες δυνατότητες και λειτουργίες.

Τον τελευταίο καιρό οι γραμμές που τα διαφοροποιούν αρχίζουν και ξεθωριάζουν καθώς και οι δύο μηχανές έχουν αναπτυχθεί αρκετά και έχουν προσφέρει πολλά ΑΑΑ Παιχνίδια στο Κοινό.

Το Unreal στην ουσία είχε ξεκινήσει για μεγάλα παιχνίδια και με τον καιρό έχει καταφέρει να γίνει και επιλογή για μικρότερες ομάδες προγραμματιστών για άλλου τύπου παιχνίδια, ενώ το Unity από εκεί που ήταν για μικρότερα παιχνίδια έχουν αρχίσει διάφορα Studio να αναπτύσσουν και μεγάλα παιχνίδια βασισμένα σε αυτό.

Οι μεγαλύτερες διάφορες που θα πρέπει να σκεφτεί κάποιος θα πρέπει να είναι τί ποιότητας γραφικά θέλουμε; Και σε τί πλατφόρμα στοχεύουμε να αναπτυχθεί το παιχνίδι; Αλλά, προσωπικά, πιστεύω ότι σε λίγο καιρό αυτές οι δυο μηχανές θα προσφέρουν τις ίδιες δυνατότητες και η επιλογή θα γίνεται ανάλογα με την προσωπική προτίμηση που θα έχει ο καθένας (World, 2019).

Κεφάλαιο 4ο

Δημιουργία παιχνιδιού με το Unreal Engine 4

Σε αυτό το κεφάλαιο θα δείξουμε κάποια χαρακτηριστικά του Engine, καθώς και ορισμένες τεχνικές που αναπτύχθηκε το παιχνίδι της πτυχιακής.

Αρχικά, ανοίγουμε το Epic Game Launcher, πηγαίνουμε στο Unreal Engine Tab, πάμε στο Library και κατεβάζουμε την έκδοση του Unreal που θέλουμε να ξεκινήσουμε να δημιουργούμε το παιχνίδι.

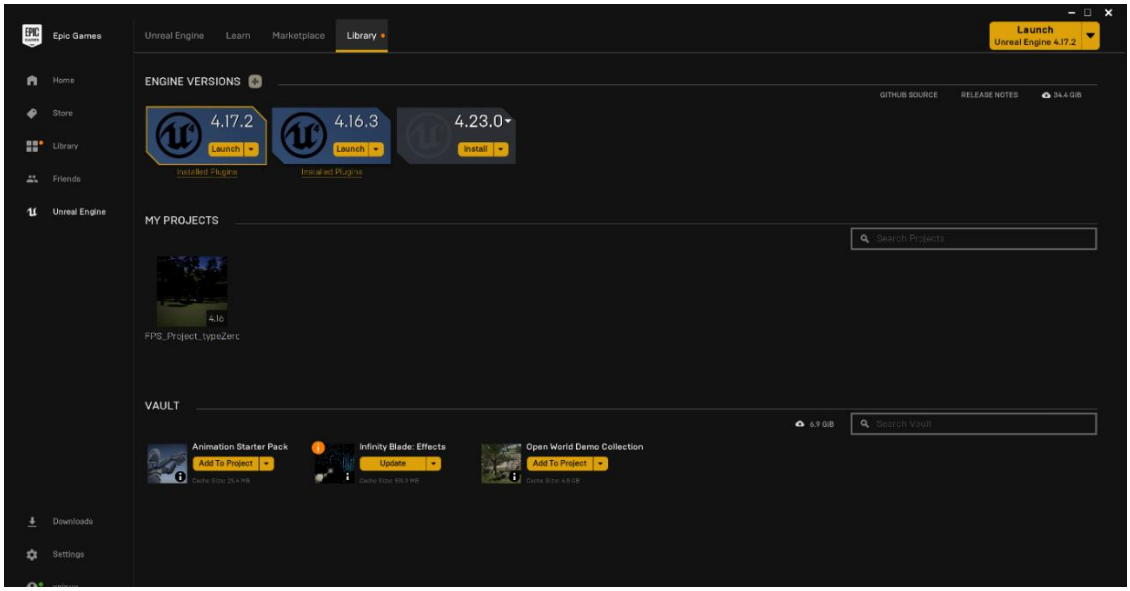


Figure 22 Library της Unreal Engine

Συνήθως τα παιχνίδια δημιουργούνται από μια έκδοση, οπότε ένα παιχνίδι μπορεί να έχει αναπτυχθεί στην έκδοση, για παράδειγμα 4.16.3, ενώ μπορεί να κυκλοφορήσει μια καινούργια έκδοση, όπως η 4.23.0.Το Project δεν μπορεί να μετακινηθεί από μια έκδοση σε νεότερη.

Στην συνέχεια, αφού πηγαίνουμε στην έκδοση που έχουμε κατεβάσει και κλικάρουμε «Launch».Θα μας ανοίξει ένα παράθυρο, το οποίο θα μας ζητάει, είτε να ξεκινήσουμε ένα νέο Project, είτε να ανοίξουμε ένα ήδη υπάρχων.

Επιλέγουμε New Project, Blueprint,Blank και δηλώνουμε στην μηχανή ότι πρόκειται για Desktop/Console, ότι θέλουμε το Project που θα ξεκινήσουμε να είναι με Maximum Quality και ότι θέλουμε, επίσης, το Starter πακέτο που μας προσφέρει η Epic για να αναπτύξουμε το παιχνίδι μας. Στην περίπτωση μας θα επιλέξουμε την επιλογή Third Person και στην συνέχεια θα κάνουμε κάποιες αλλαγές, έτσι ώστε να φέρουμε την κάμερα στο First Person View.

Το Starter Content είναι ένα πακέτο, το οποίο το προσφέρει η Epic δωρεάν, το οποίο περιέχει διάφορα μικρά Assets, μοντέλα, εφέ ήχου, το οποίο σε βοηθάει να ξεκινήσεις ένα Project. Βέβαια, δεν είναι αρκετό για να αναπτυχθεί ένα παιχνίδι, για αυτό και πολλές φορές χρειάζονται και αλλά Assets.

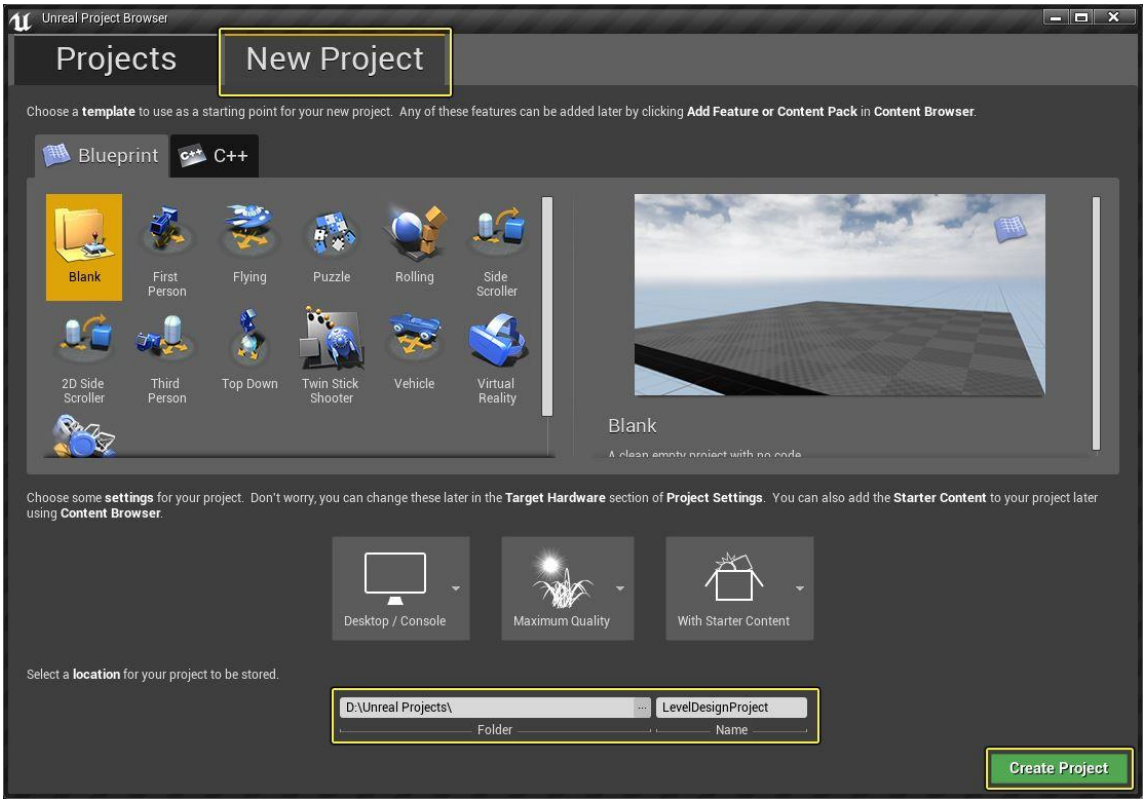


Figure 23 Δημιουργώντας New project

Πρώτα ονομάζουμε το Project μας και ορίζουμε τον χώρο που θα αποθηκευτεί. Η πρώτη εικόνα που μας καλωσορίζει είναι ένας άδειος χώρος με έναν άδειο «ηθοποιό», το οποίο εμείς μπορούμε να αρχίζουμε να προγραμματίζουμε, να σχεδιάσουμε να τιμές και να του δώσουμε ζωή με Animations. Έπειτα, συνεχίζουμε με το περιβάλλον, διάφορα Materials και το AI.

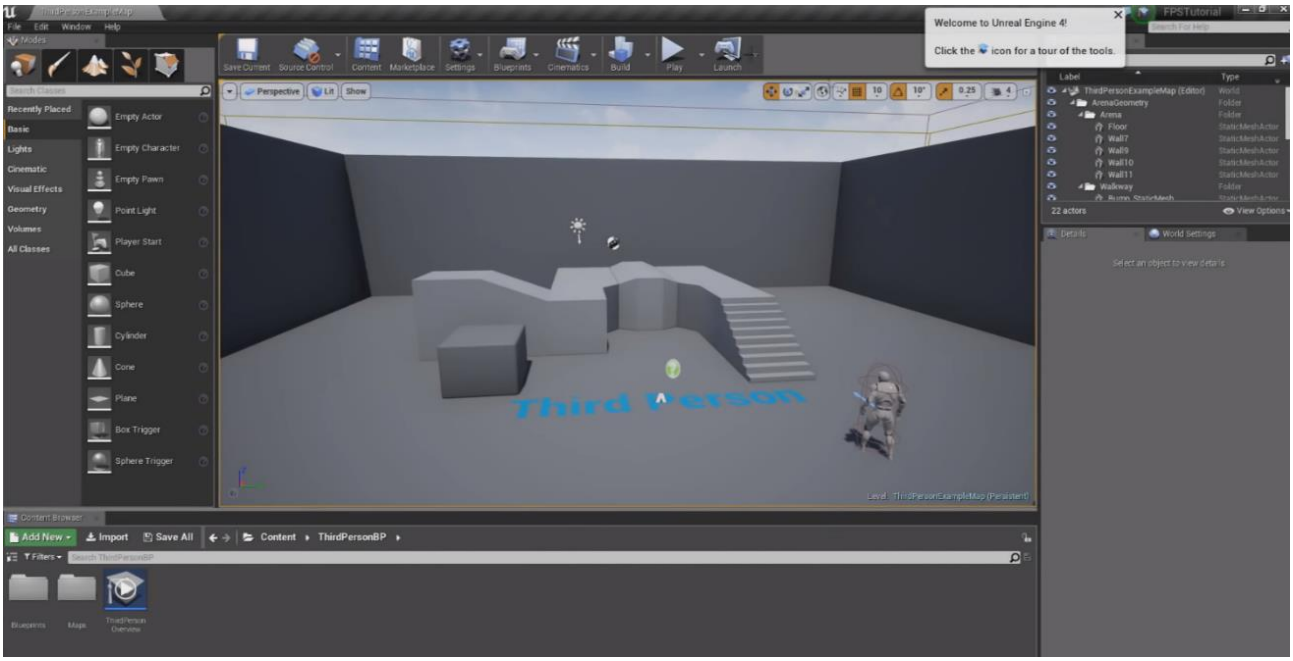


Figure 24 Sample Level

Starting Level

Το πρώτο πράγμα που θα κάνουμε είναι να φτιάξουμε το Level μας, δηλαδή το επίπεδο μας. Αρχικά, πρέπει να το κάνουμε πιο μεγάλο, ώστε να χωράνε περισσότερα πράγματα.

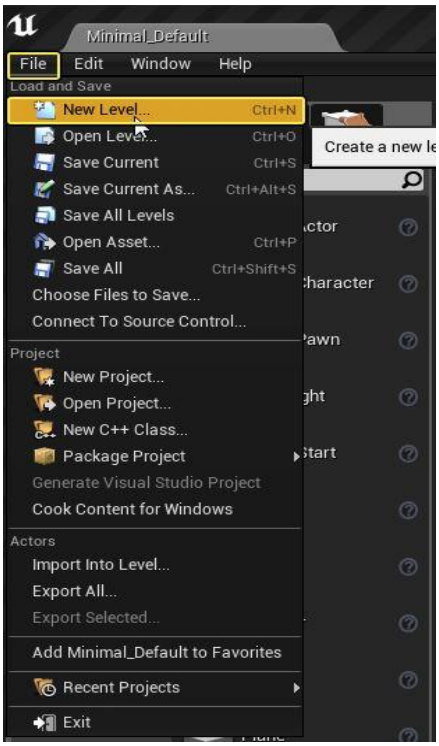


Figure 24 New Level

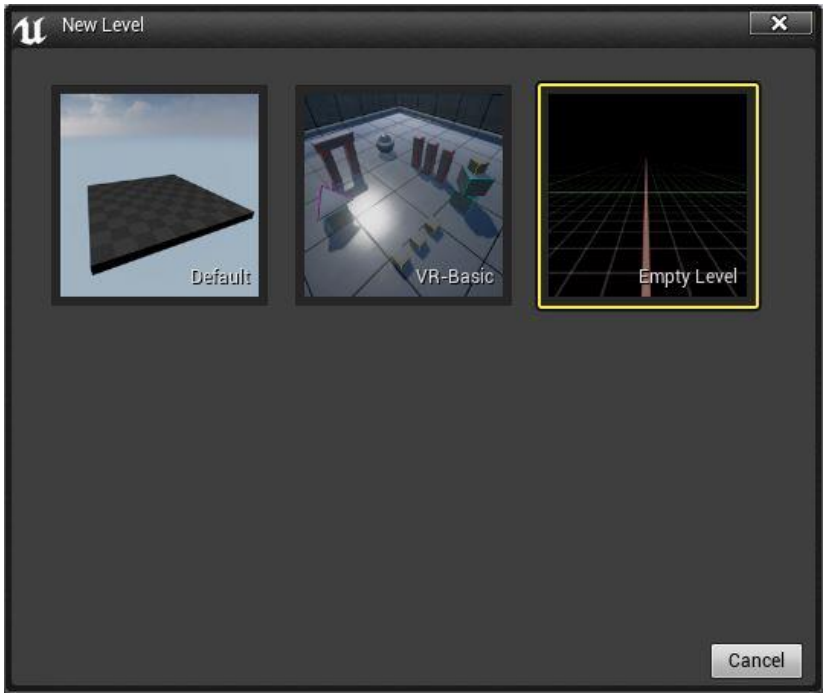


Figure 25 Δημιουργώντας Empty Level

Έπειτα, η μηχανή μας δίνει ένα χώρο κενού στο οποίο εμείς θα δημιουργήσουμε ένα δάπεδο για το επίπεδο μας. Αυτό γίνεται με αυτόν τον τρόπο:

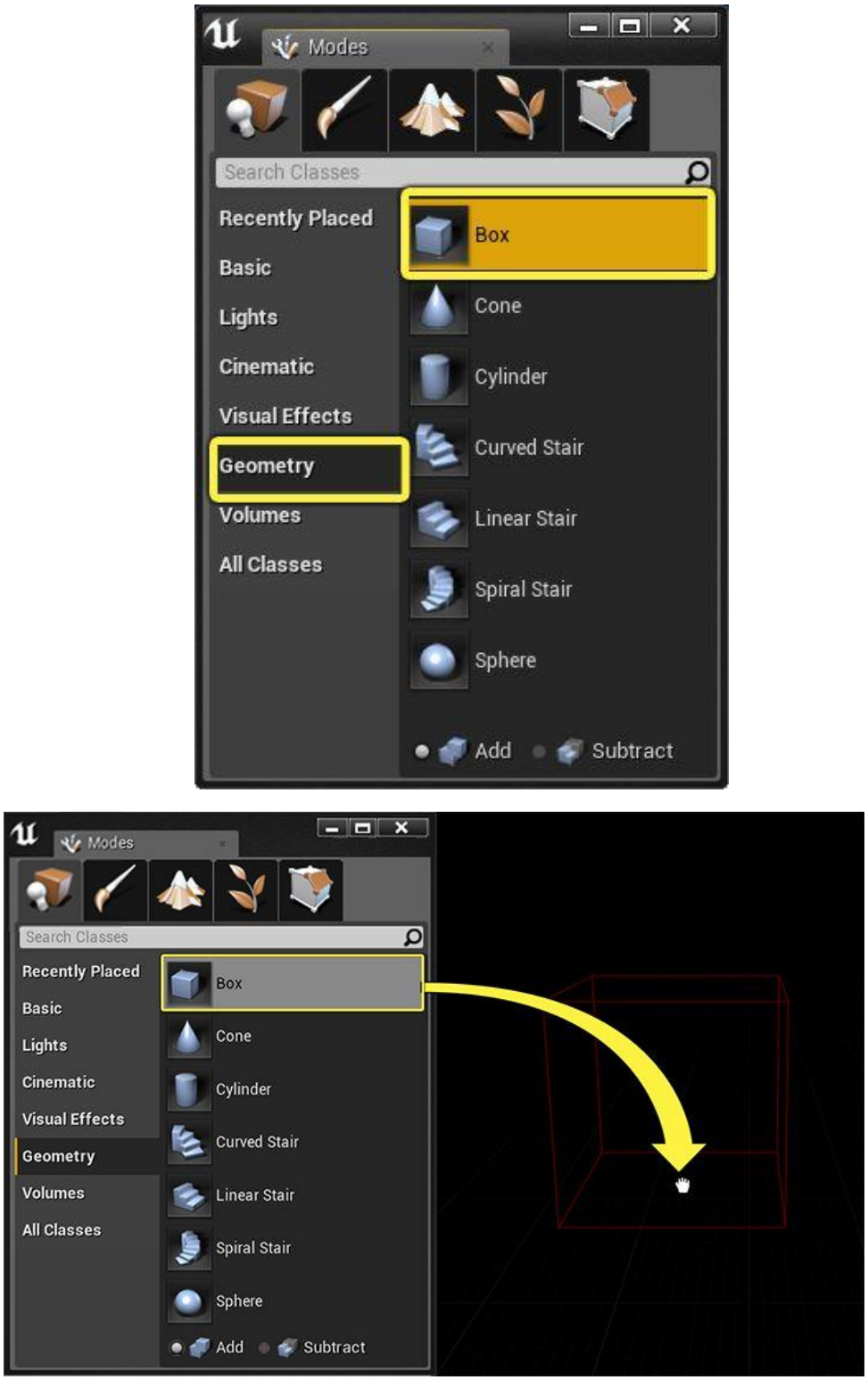


Figure 26-27 Box για δημιουργία Arenas

Στην συνέχεια ορίζουμε το μέγεθος του «Κουτιού» ώστε να μοιάζει σαν δάπεδο, με αυτόν τον τρόπο έχουμε μια βάση του επίπεδου που θέλουμε να φτιάξουμε, ώστε να μπορούμε να δουλέψουμε άλλα πράγματα, προτού το ολοκληρώσουμε.

Το Unreal Engine έχει τρία Transform, τα οποία μπορούμε να τροποποιήσουμε για κάθε αντικείμενο/εξάρτημα που βάζουμε. Το πρώτο είναι το Location:Στο Location αλλάζουμε το σημείο που βρίσκεται ένα αντικείμενο. Το δεύτερο είναι το Rotation:Αλλάζουμε την Κλίση του αντικειμένου και το τρίτο είναι το Scale:Αλλάζουμε την Κλίμακα που θέλουμε στο αντικείμενο

Στην συγκεκριμένη περίπτωση πειράζουμε το Scale και δημιουργούμε ένα μεγάλο επίπεδο Δάπεδο.

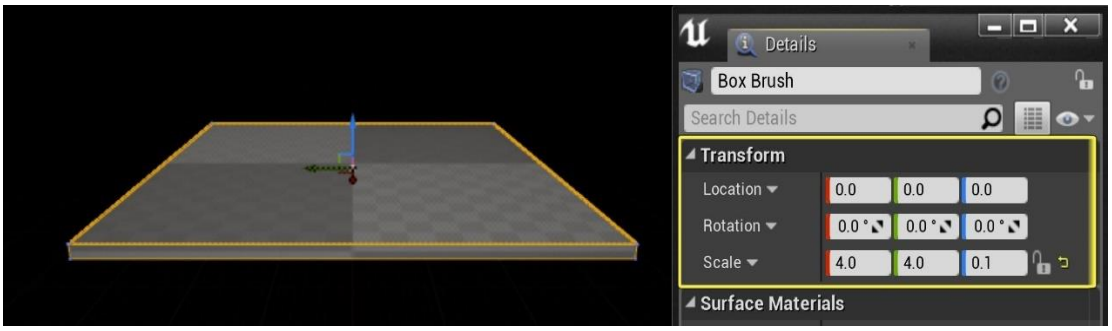


Figure 28 Φτιάχνοντας τον χώρο που θέλουμε

Στη συνέχεια, θα βάλουμε φωτισμό και κάποια εφέ, όπως Post-Processing για σκιές κλπ., που τα επιλέγουμε από τη Μηχανή.

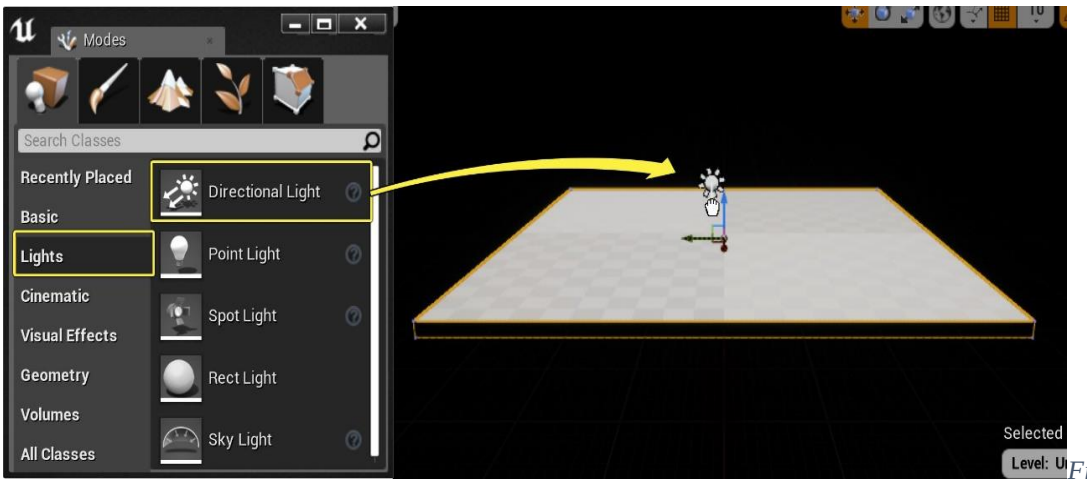


Figure 29 Βάζουμε τον φωτισμό που θέλουμε

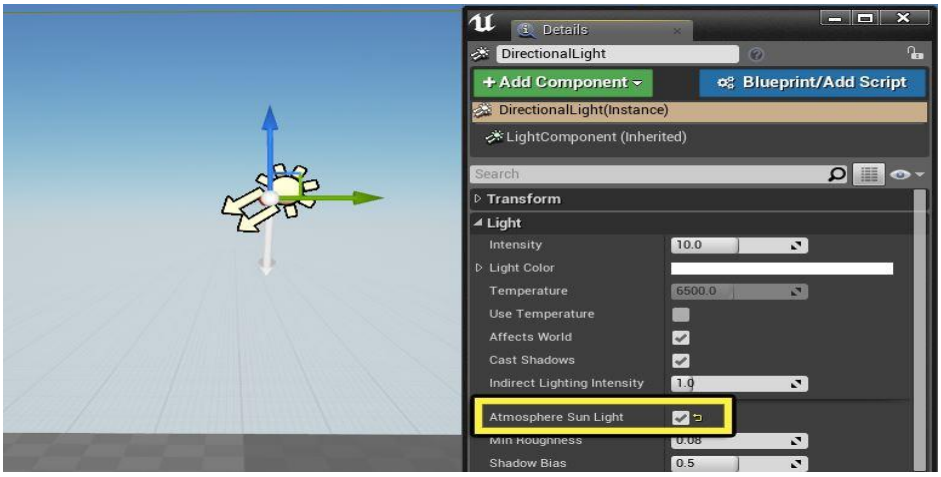


Figure 30 Φωτισμός ηλιου

Προς το παρόν έχουμε δημιουργήσει μια επιφάνεια, στην οποία μπορούμε να δουλέψουμε. Θα αφήσουμε το Level Design για αργότερα, ώστε να ασχοληθούμε πιο πολύ στο παιχνίδι, τον προγραμματισμό του και τον σχεδιασμό του.

Κάμερα

Θα ξεκινήσουμε αλλάζοντας την κάμερα. Στην αρχή είχαμε επιλέξει ότι το Παιχνίδι στην μηχανή θα έδινε σαν οπτική ένα Third Person View. Δηλαδή, θα είχαμε μια οπτική γωνία από «πάνω» από τον χαρακτήρα και όχι στο πρώτο πρόσωπο. Έτσι και εμείς θα πρέπει να πάμε να κάνουμε την αλλαγή αυτή.

Δηλαδή, θα βάλουμε την Κάμερα Λίγο πίσω και κοντά στο πρόσωπο του «ηθοποιού» μας, ώστε όταν αρχίζει το παιχνίδι, η οπτική γωνία που να έχουμε να είναι πρώτου προσώπου.

Αυτό για να το κάνουμε θα πειράξουμε το Transform του «Follow Camera» που το βρίσκουμε στο Viewport του βασικού μας Blueprint,στα Components.

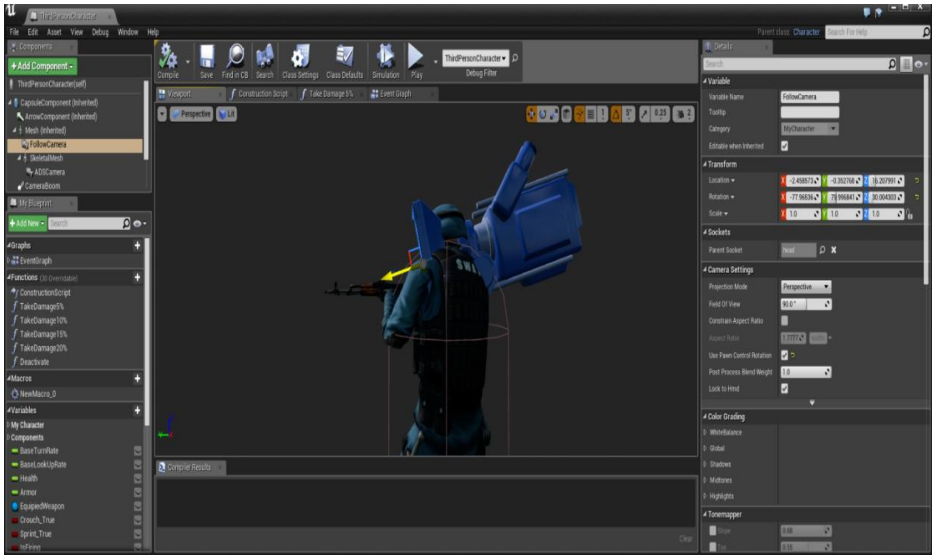


Figure 3 Βάζοντας την κάμερα μπροστά στο κεφάλι καταφέρνουμε το First Person View

Inputs

Αφού έχουμε τελειώσει με την κάμερα και τώρα έχουμε πλέον ένα First Person View πρέπει να προσθέσουμε τις πολύ βασικές λειτουργίες, δηλαδή τις κινήσεις του ηθοποιού μας.

Αρχικά αυτό που κάνουμε είναι να πάμε να ορίσουμε σε ποια πλήκτρα θέλουμε να τα προγραμματίσουμε. Σε αυτήν την φάση θα βάλουμε όλα τα πλήκτρα που μας ενδιαφέρουν, θα τους δώσουμε ένα όνομα σαν «Μεταβλητή», ώστε να μπορούμε να τα χρησιμοποιήσουμε στον προγραμματισμό και στην συνέχεια θα προγραμματίζουμε.

Αναλυτικά, πάμε Edit και Project Preferences και επιλέγουμε το Input.

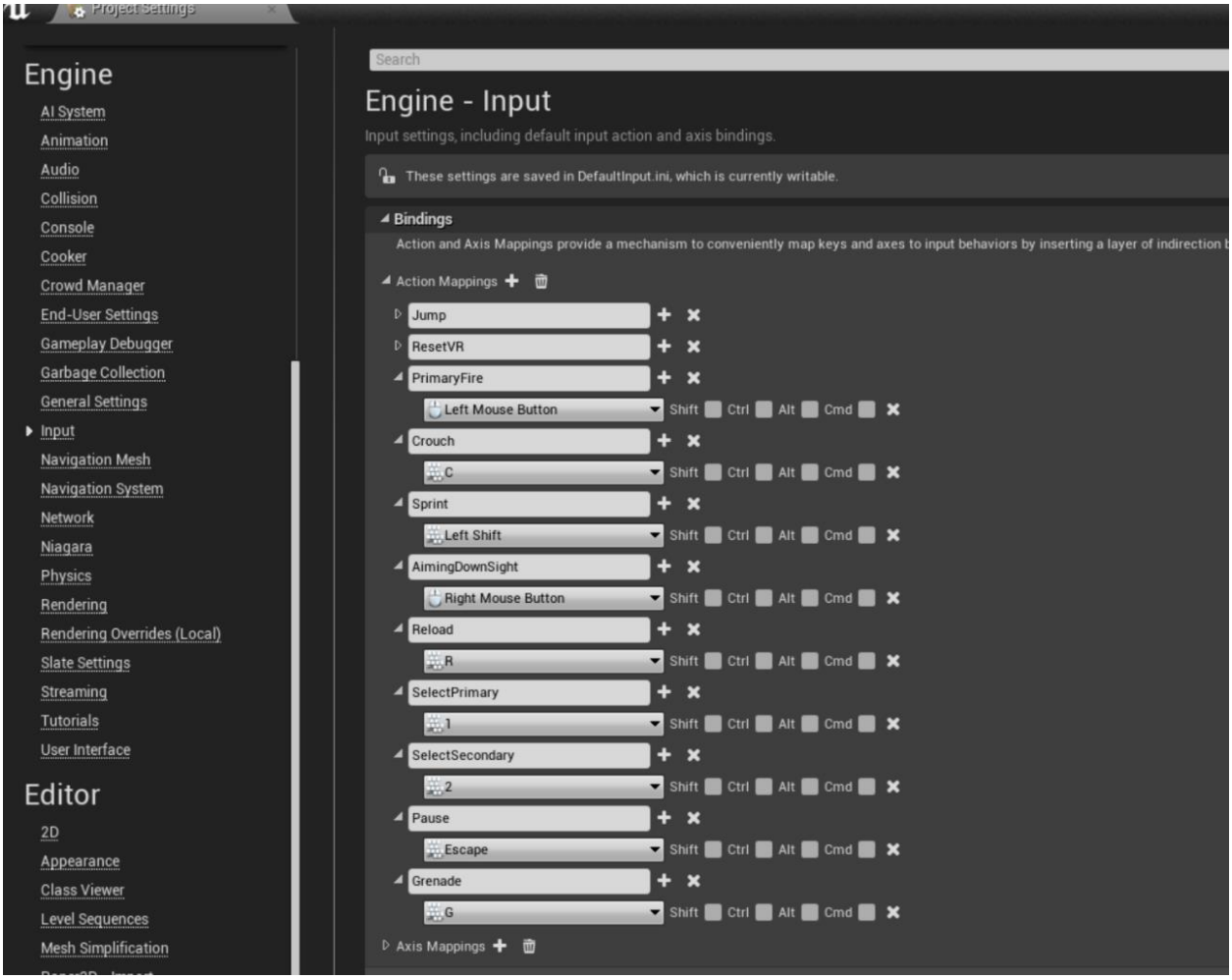


Figure 31 Φτιάχνοντας τα κουμπία που θέλουμε για ενέργειες

Ορίζουμε όλα τα πλήκτρα που θέλουμε για το Project μας, καθώς και τους δίνουμε την ανάλογη ονομασία.

Για παράδειγμα με το ‘C’ ο «ηθοποιός» θα σκύβει, ενώ με το ‘Left Shift’ θα τρέχει πιο γρηγορά, με το δεξί κλικ θα πυροβολεί το όπλο και το με αριστερό κλικ θα κάνει Zoom κλπ.

Το Unreal Engine μας προσφέρει διάφορα αντικείμενα και ο τρόπος προγραμματισμού αυτών των αντικειμένων είναι ποικίλος.

Blueprints, Mouse και άλμα

Το Unreal Engine προσφέρει το Concept των Blueprints.Τα Blueprints είναι η βασική μονάδα προγραμματισμού της λογικής που θέλουμε να προγραμματίσουμε, καθώς μέσα από αυτά τα Blueprints που δημιουργούμε μπορούμε να προγραμματίσουμε από κάμερα και AI μέχρι ακόμα και το πώς θα συμπεριφέρεται ένα δέντρο. Ο Προγραμματισμός των Blueprints θυμίζει γλώσσες προγραμματισμού, καθώς γίνεται με Scripting βασισμένο στην C++.

Το κύριο μας Blueprint είναι το ThirdPersonBP που είναι ο χαρακτήρας μας. Ο κύριος χαρακτήρας μας που μπορεί να ελέγχει ο χρήστης, στο Unreal Engine 4

ονομάζεται «Ηθοποιός». Εκεί θα κάνουμε τον περισσότερο προγραμματισμό για να προσθέσουμε διάφορες λειτουργίες στον χαρακτήρα μας, από απλά βήματα μέχρι το πώς γεμίζει η ζωή του.

Για αρχή, θα προσθέσουμε τη λειτουργία του ποντικιού για να βλέπει τριγύρω καθώς και του «Jump», δηλαδή του άλματος.

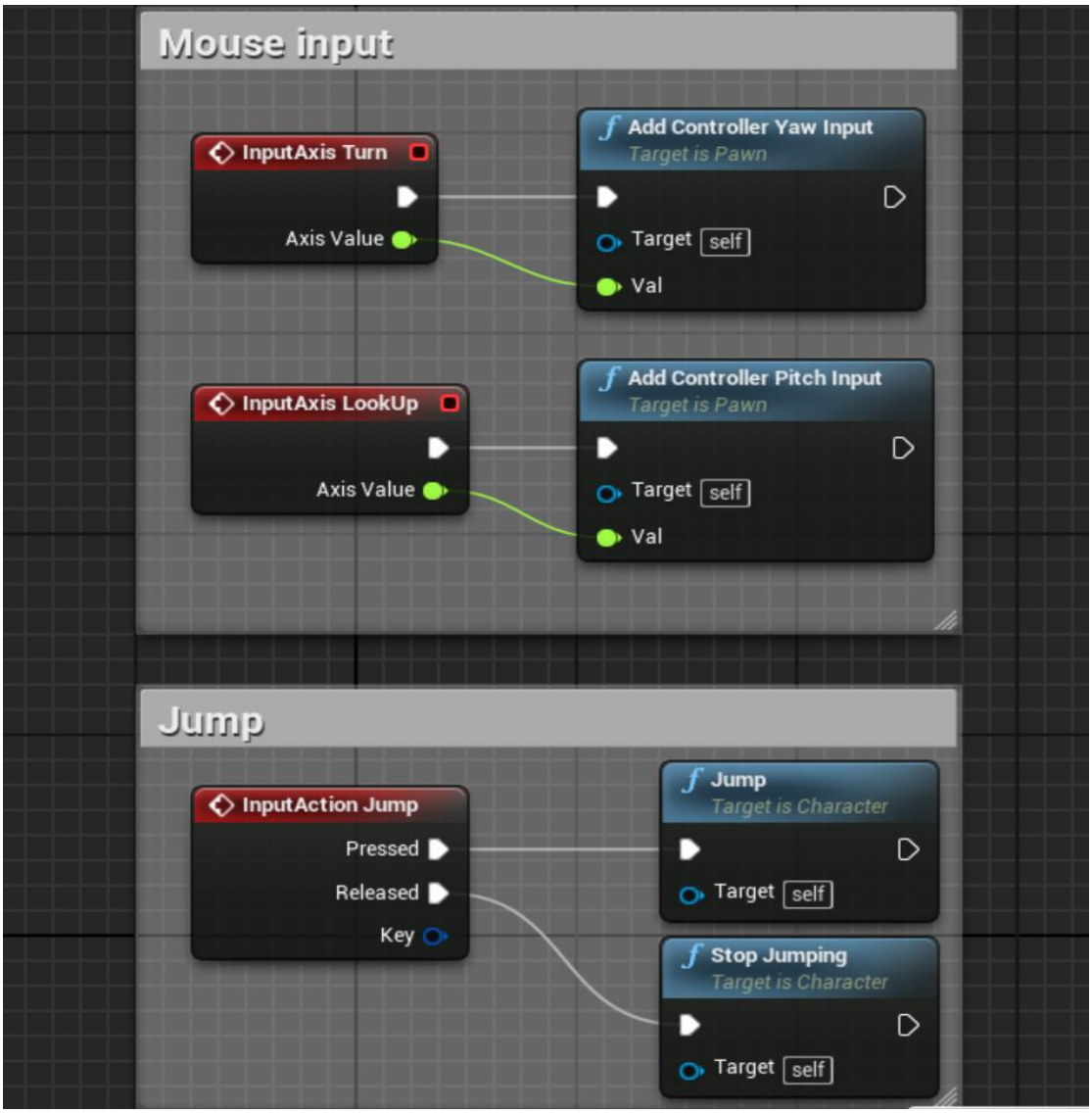


Figure 32 Προγραμματίζοντας την κίνηση του πηδήματος και της στροφής.

Όπως είπαμε και πριν, για κάθε πλήκτρο έχουμε βάλει ένα συγκεκριμένο όνομα. Ουσιαστικά, για το «Jump»,όταν ο χρήστης πατήσει το R, τότε θα κάνει άλμα ο «ηθοποιός» και, όταν το αφήσει, θα σταματάει.

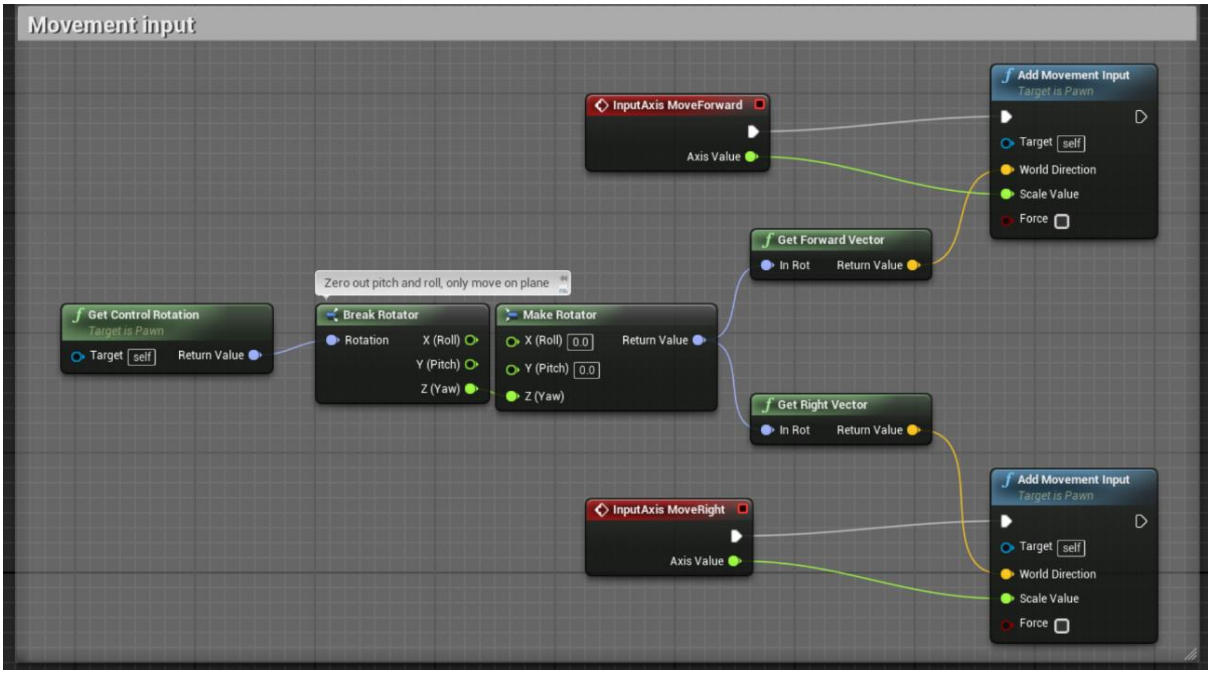


Figure 33 Κινήσεις.

Animations

Το επόμενο δύσκολο κομμάτι που έχουμε να δείξουμε είναι να πως προσθέτουμε Animations και μια μηχανή καταστάσεων και πιο συγκεκριμένα, μεταξύ του περπατήματος και του τρεξίματος. Με αυτόν τον τρόπο θα πέτυχουμε ένα ρεαλιστικό σχεδιασμό στο πως ο «ηθοποιός» τρέχει ή περπατάει καθώς και στην λογική που γίνονται οι εναλλαγές και πως φαίνονται.

Για να προσθέσουμε Animations, καθώς και άλλες λειτουργίες πάνω στο παιχνίδι μας θα πρέπει να κατεβάσουμε και άλλα ελεύθερα Assets, εκτός από το Starter pack, ή μπορούμε να αγοράσουμε ή να δημιουργήσουμε μέσω Blender.

Πιο συγκεκριμένα πάμε στο Marketplace της Epic και κατεβάζουμε αυτά εδώ:

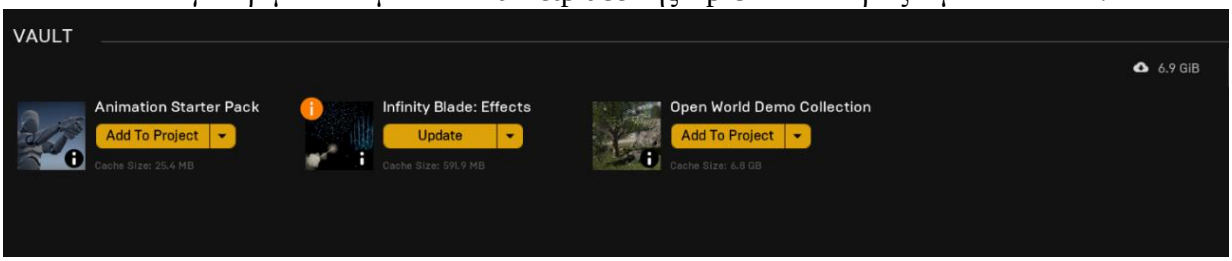


Figure 34 Προσθέτοντας τα Assets μας

Ξεκινάμε πρώτα φτιάχνοντας ένα «Skeletal Mesh». Μέσα από το Skeletal Mesh μπορούμε να δώσουμε μορφή στο απλό πιόνι μας, δηλαδή τον «Ηθοποιό» μας και να πάρει μια πιο κανονική μορφή χαρακτήρα, συγκεκριμένα στην δική μας υλοποίηση, έναν στρατιώτη.

Μέσα από το Skeletal Mesh μπορούμε να δώσουμε μορφή, να ορίσουμε χρώματα, να του ορίσουμε, επίσης, τί Φυσική θα έχει αυτός ο χαρακτήρας και πολλά αλλά.

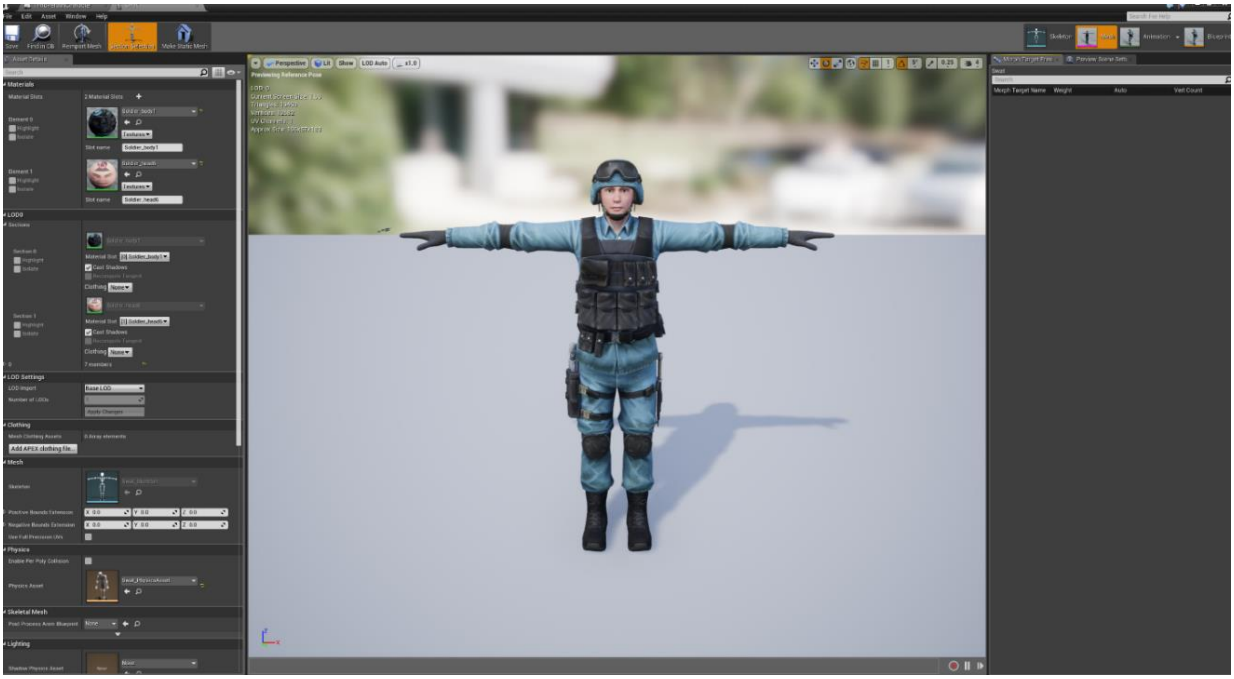


Figure 35 Skeletal Mesh

Αφού τελειώσουμε με την εμφάνιση, τώρα μπορούμε να ξεκινήσουμε με τα Animations του χαρακτήρα. Έτσι, λοιπόν, φτιάχνουμε ένα αρχείο «Animation» που μας προσφέρει το Unreal Engine και ξεκινάμε τον προγραμματισμό του.

Η αρχική κατάσταση του Animation Blueprint Είναι αυτή εδώ:

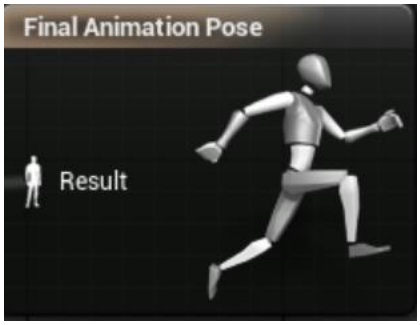


Figure 36 Animatiion BP

Στην συνέχεια, πρέπει να βάλουμε κάποιες καταστάσεις για το πώς θα φαίνεται αυτό το τελικό Animation, ξεκινώντας με ένα State «Walk_Run».

Στην Ουσία, το State Walk_Run θα ορίζει πως γίνονται οι μεταβολές των Animations από το απλό περπάτημα μέχρι το γέμισμα του όπλου και το τρέξιμο, καθώς, βέβαια, θα ορίζει και σε κάθε κατάσταση ποια Animation θα αναπαράγονται από τον «ηθοποιό».

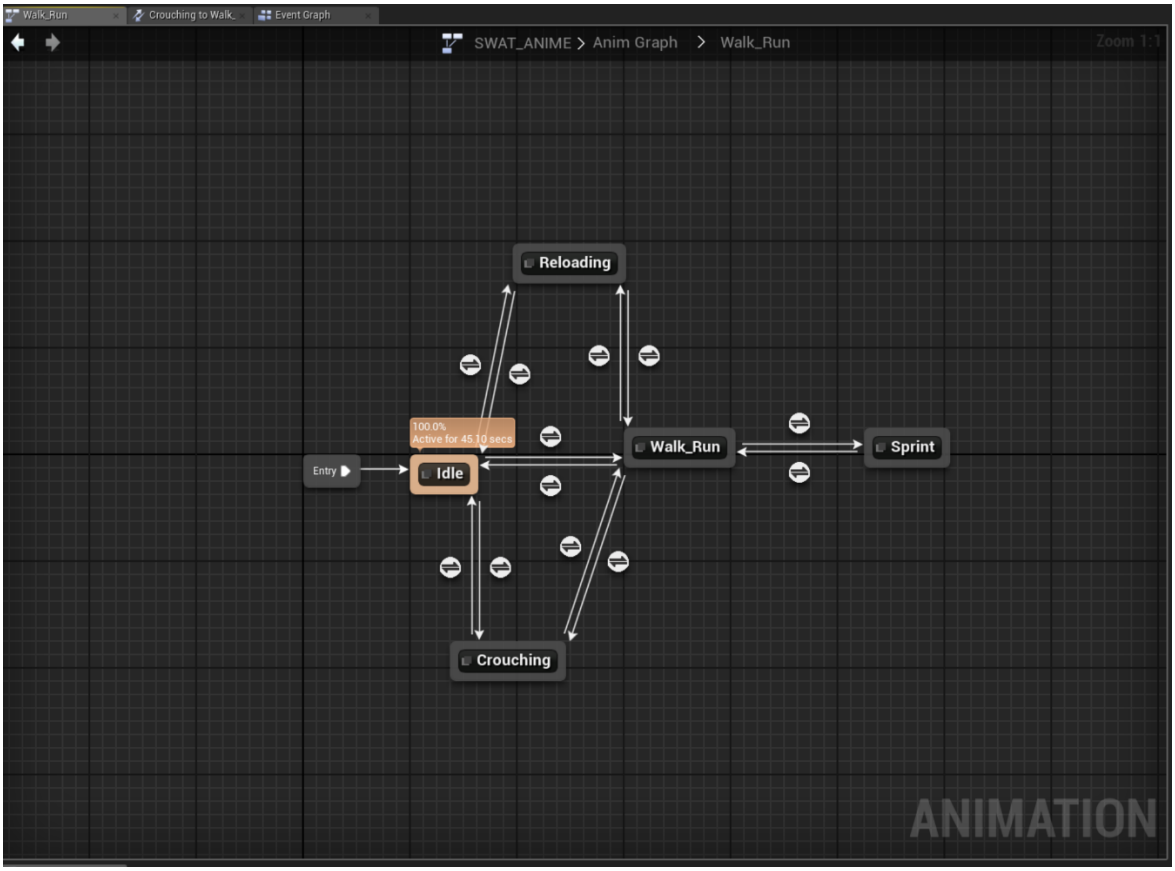


Figure 37 Walk_Run State

Όπως βλέπουμε, ξεκινάμε από μια χαλαρή «Idle» κατάσταση και γίνονται εναλλαγές μεταξύ Walk_Run, δηλαδή μια κατάσταση που τρέχει λίγο πιο γρηγορά, μια κατάσταση Sprint που τρέχει πολύ πιο γρηγορά, πώς σκύβει ο χαρακτήρας. καθώς και πως γεμίζει το όπλο.

Ορίζοντας την IDLE που έχουμε δημιουργήσει, μπαίνουμε μέσα και απλά βάζουμε το Animation που ήδη έχουμε, με τον ίδιο τρόπο γίνονται και οι άλλες:

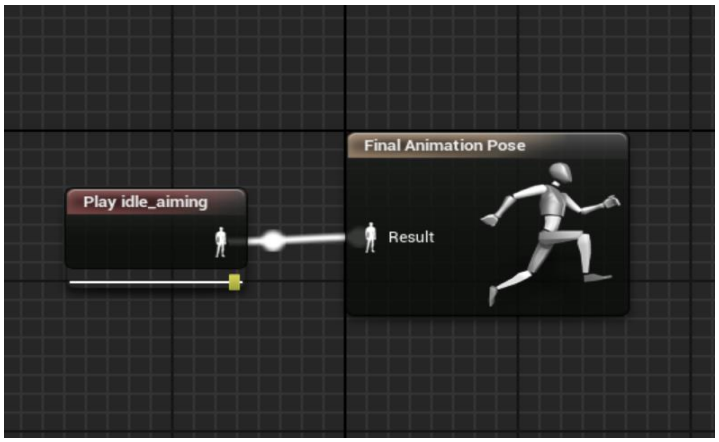


Figure 38 Κάνοντας Bind animation σε κίνηση.

Κατόπιν, θα πρέπει να ορίσουμε μια λογική για το πώς θα γίνονται οι εναλλαγές αυτές. Μπαίνουμε,λοιπόν, σε ένα από τα Βελάκια που είναι στο Βασικό Σχηματικό

Animation μας, για παράδειγμα από Idle κατάσταση σε Walk_Run και ορίζουμε την λογική.

- Από Idle σε Walk Run:

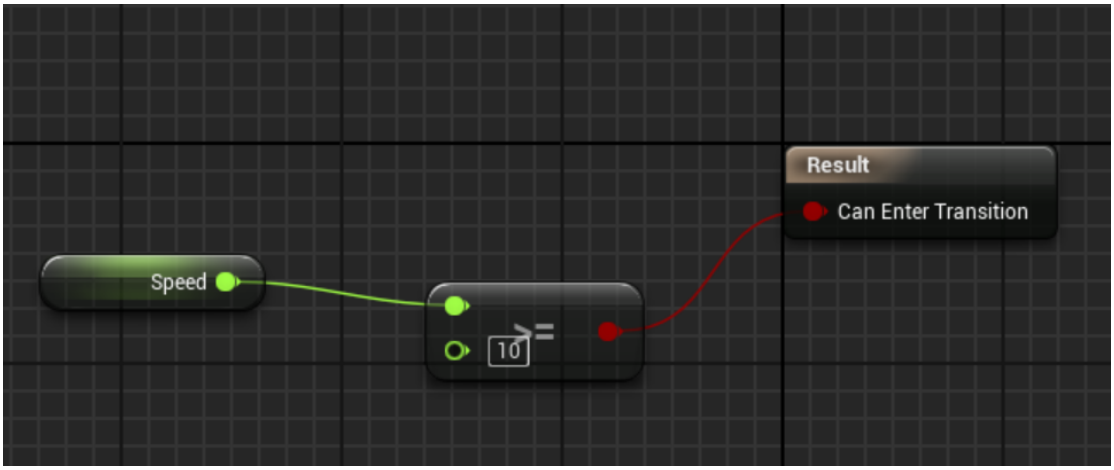


Figure 39 ορίζοντας τα Animation ανάλογα με την ταχύτητα εάν είναι πάνω από 10 τρέχα.

Στην ουσία λέμε, εάν η ταχύτητα είναι πάνω από 10, τότε άλλαξε κατάσταση. Δηλαδή, εάν μετακινηθεί ο παίχτης.

Και το αντίθετο:

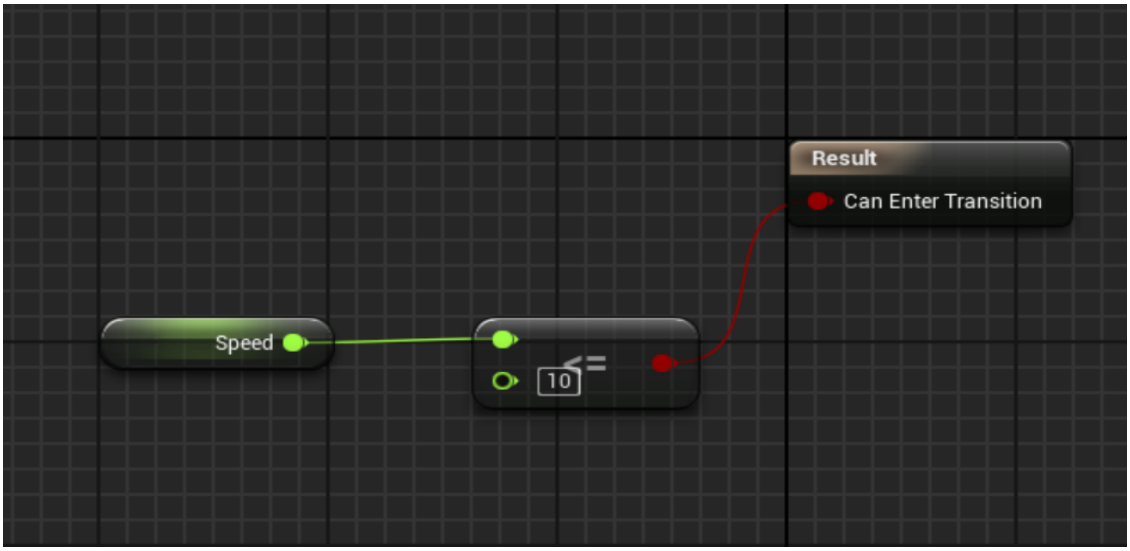


Figure 40 ορίζοντας τα Animation ανάλογα με την ταχύτητα εάν είναι κάτω από 10 περπάτα.

Για να ορίσουμε το Sprinting, δηλαδή το πιο γρήγορο τρέξιμο, θα πρέπει να πάμε πρώτα στο βασικό μας Blueprint, δηλαδή το ThirdPersonBP και να προγραμματίσουμε λιγάκι την λογική. Πιο συγκεκριμένα να προσθέσουμε μια μεταβλητή τύπου Boolean που θα παίρνει τιμές true ή false, εάν ο χρήστης πατάει το κουμπί που ορίσαμε για να τρέχει γρηγορά.

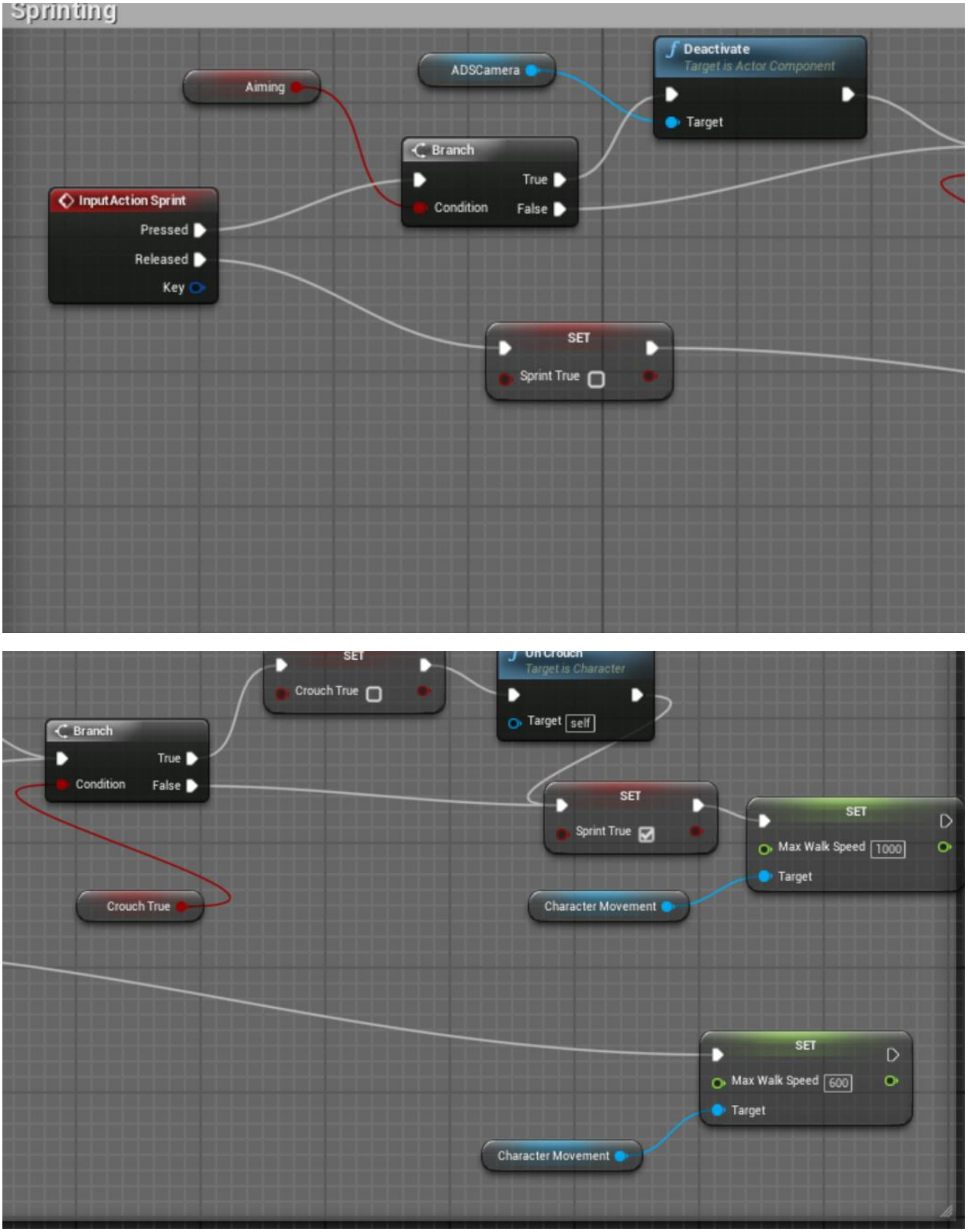


Figure 41 Aiming Logic

Όταν ο χρήστης πατάει το Sprint, γίνεται πρώτα ένα If (Branch), εάν κάνει Aiming το απενεργοποιεί, συνεχίζει κάνοντας ένα Branch, εάν σκύβει το απενεργοποιεί, ορίζει την μεταβλητή Sprint true και μας ορίζει την ταχύτητα με την συνάρτηση Set στα 1000.

Όταν ο Χρήστης αφήσει το Κουμπί, τότε η μεταβλητή Sprint γίνεται False και αλλάζουμε πάλι την τιμή της ταχύτητας στα 600.

- Η εναλλαγή της Spent στο State Machine.

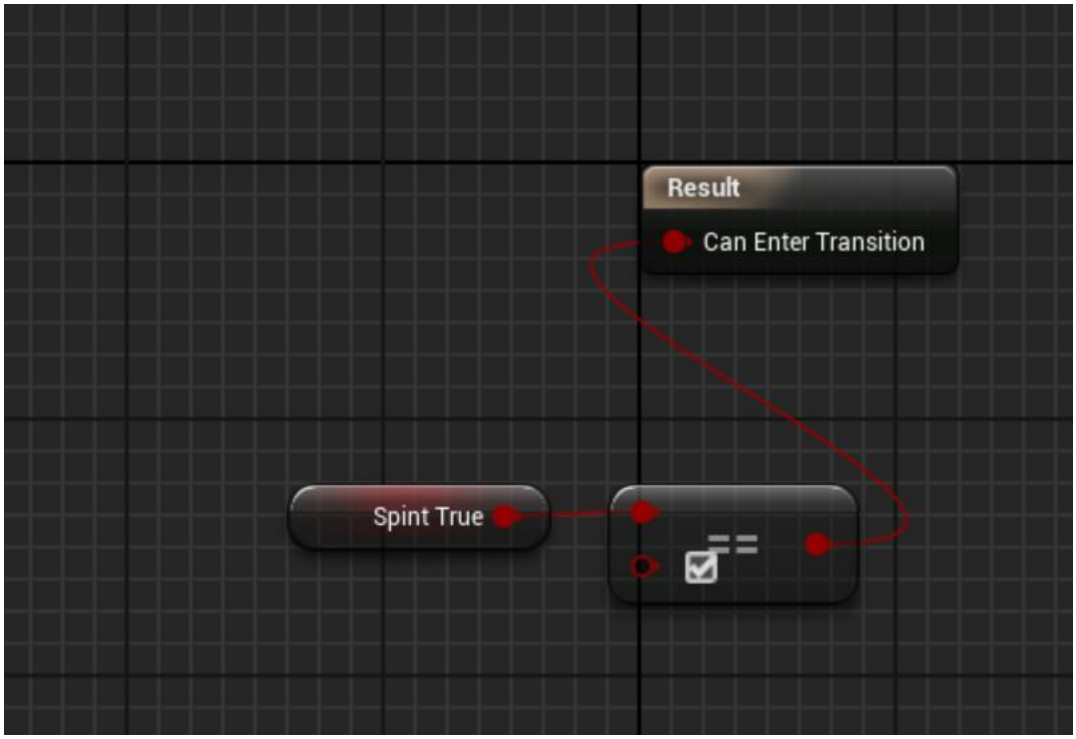


Figure 42 Sprint Is true

- Και η Sprint σαν Animation Logic στο AnimationBP:

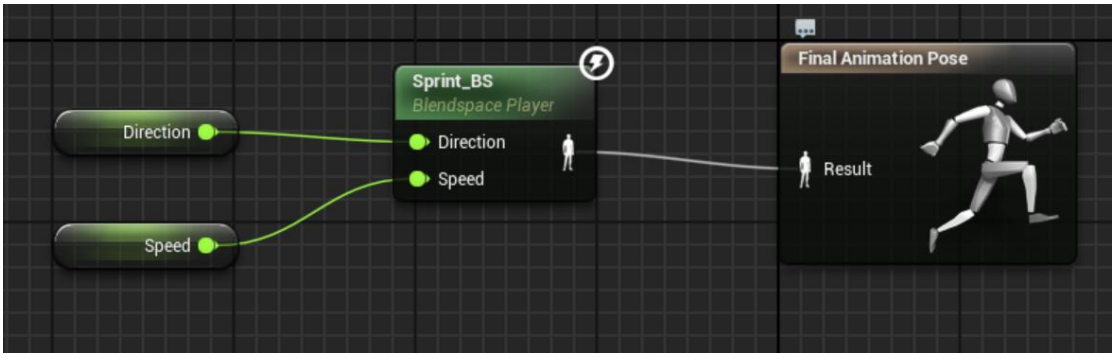


Figure 43 Τελικό Animation από Direction και Speed

Τέλος για τα Animation, το Unreal Engine μας προσφέρει ένα ακόμα χρήσιμο εργαλείο για να ορίσουμε το κάθε Animation που θέλουμε σε κάθε φορά που θα έχει ο «ηθοποιός» μας και σε διαφορετική ταχύτητα που θα έχει. Αυτό το εργαλείο, ουσιαστικά, είναι σαν ένα X-Y γράφημα, στο οποίο ορίζουμε τις 2 μεταβλητές που θέλουμε και ανάλογα σε ποιες καταστάσεις βρίσκεται βάζουμε τα κατάλληλα Animation να αναπαράγονται. Σε εμάς οι δυο μεταβλητές που θα μπουν στο γράφημα X-Y είναι το Direction και το Speed. Δίπλα από το γράφημα έχουμε όλα τα αρχεία των Animation σε μορφή Video, ώστε να μπορούμε να τα βάλουμε πάνω στο γράφημα, ανάλογα την λογική που έχουμε.

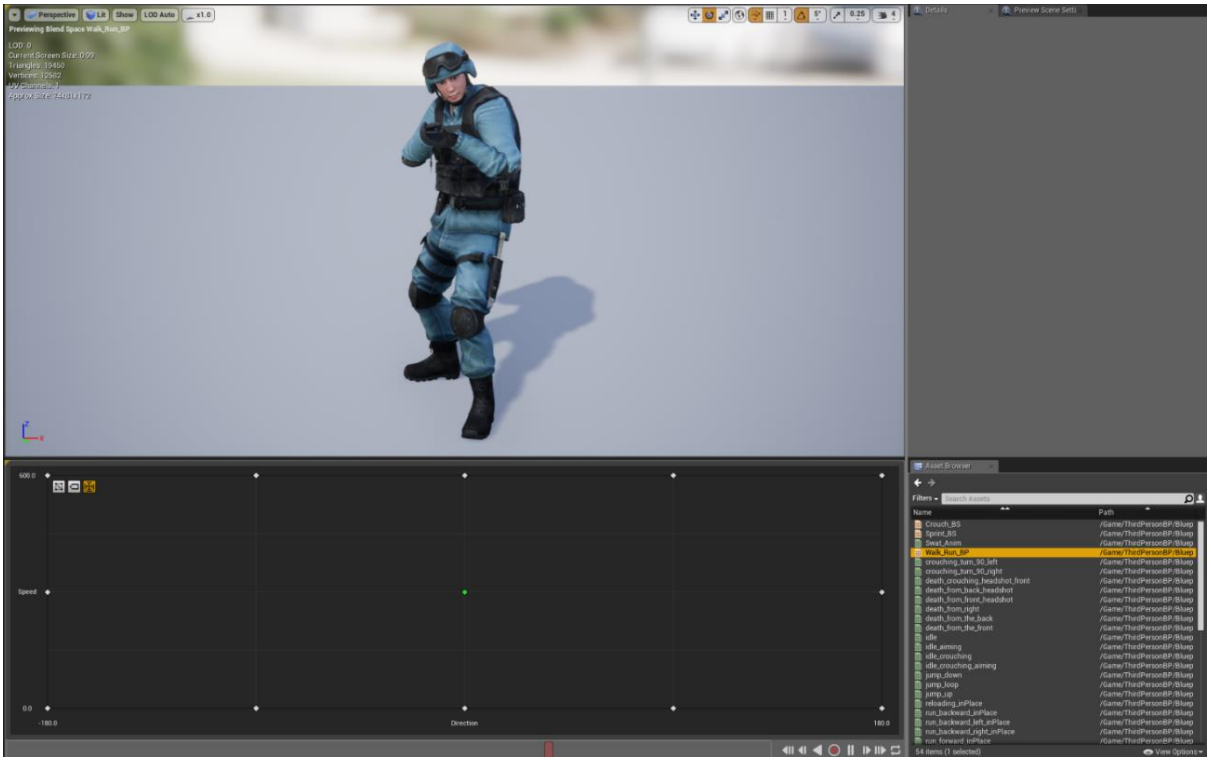


Figure 44 Setting Animations On Graph

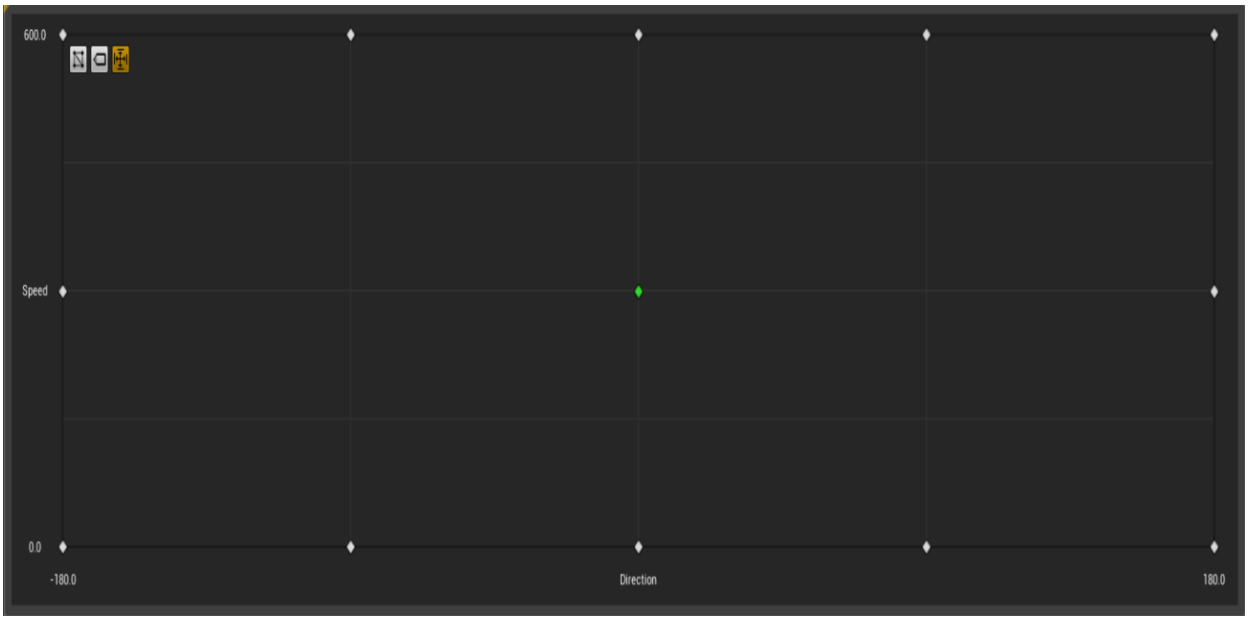


Figure 45 Γράφημα Speed-Direction για τα Animations

Στο Y άξονα έχουμε την ταχύτητα και στο X άξονα έχουμε την Κατεύθυνση. Οι κουκίδες είναι τα Animation που θα παίζουν σε κάθε κατάσταση. Για παράδειγμα, στην κατάσταση 0 speed, 0 γωνίες Direction Το Animation που θα παίζει θα είναι το IDLE_Aiming, ενώ στην κατάσταση 600 speed -90 Direction θα έχουμε ένα άλλο Aanimation το run_forward_left_inplace.

Με τον ίδιο τρόπο που έχουμε ορίσει την κατάσταση Walk_run, καθώς και τα Animations και το γράφημα, έτσι ορίζουμε άλλες 2 καταστάσεις. Το Sprint και το Crouching προγραμματίζοντας ανάλογα.

Με αυτόν τον τρόπο δείξαμε πως κάποιος δημιουργεί Skeletal Mesh, πως το ντύνει, δημιουργεί καταστάσεις, δημιουργεί εναλλαγές, ορίζει Animations, καθώς και τη λογική και τέλος το γράφημα των καταστάσεων, ανάλογα με την ταχύτητα και τη φορά.

Βέβαια, δεν τελειώνει εκεί, πρέπει να τα ενώσουμε όλα αυτά. Ο Κώδικας που ενώνει όλες τις καταστάσεις στο συγκεκριμένο παιχνίδι είναι αρκετά μεγάλος και παρατίθεται ενδεικτικός Κώδικας που βρίσκεται Στο AnimationBP στο Event Graph.

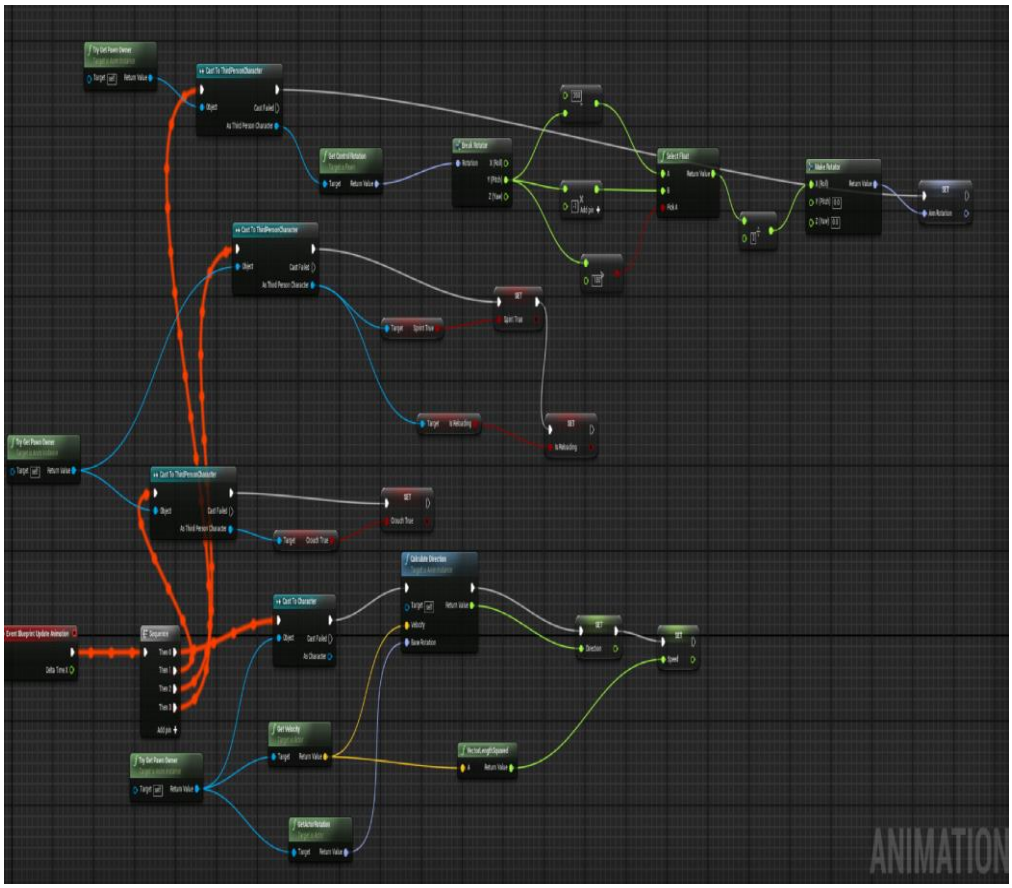


Figure 46 Animation_BP.

Τα αρχεία του Animation καθώς και τα schematics:

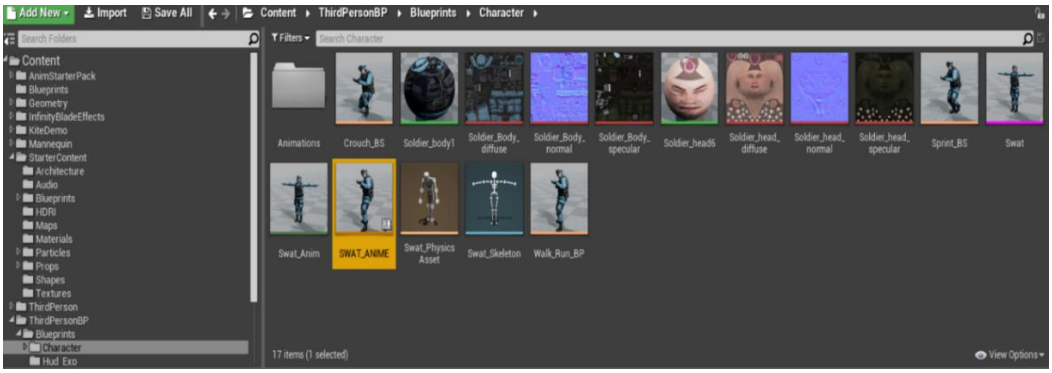


Figure 47 SWAT_ANIME bp & Sprint_BS, Walk_Run, Crouch_BS

Δημιουργία του πρώτου Όπλου:

Για την Δημιουργία του Πρώτου μας όπλου θα χρησιμοποιήσουμε κάποια έτοιμα Assets, δηλαδή κάποια έτοιμα μοντέλα, αφού δεν έχουμε χρησιμοποιήσει Blender για να φτιάξουμε εμείς.

Συγκεκριμένα, το πρώτο όπλο που θα χρησιμοποιήσουμε θα είναι το Ak-47. Θα δείξουμε έναν ενδεικτικό τρόπο για το πώς προγραμματίζουμε και σχεδιάζουμε, καθώς και πώς προσθέτουμε ένα όπλο στο παιχνίδι μας.

Αρχικά, θα φτιάξουμε ένα Material, το οποίο θα έχει την βάση του χρώματος για την εμφάνιση του όπλου μας. Το Material θα πάει και θα μπει πάνω στο μοντέλο του όπλου, ώστε να πάρει την εμφάνιση του.

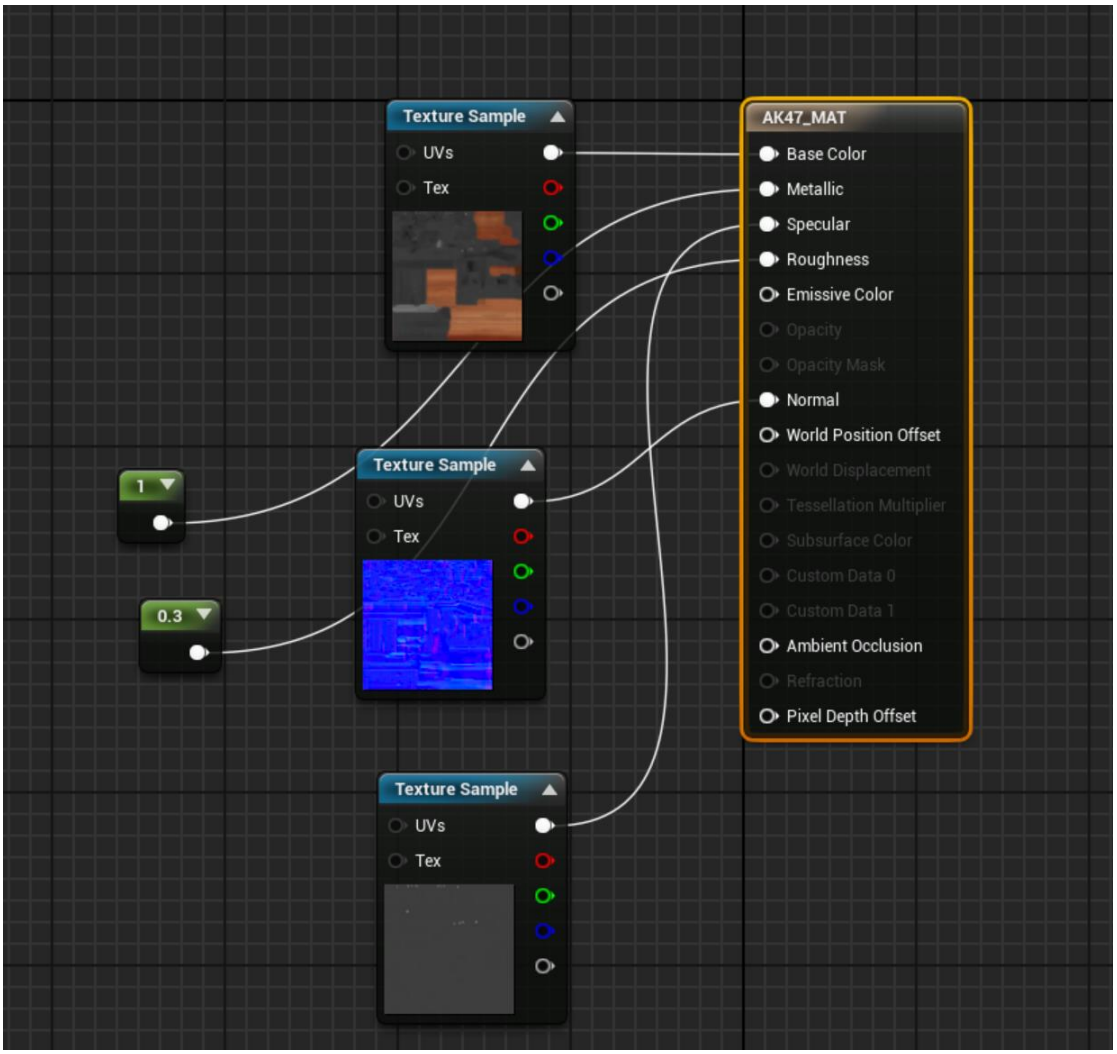


Figure 48 "Βάφοντας" το όπλο

Στην συνέχεια κάνουμε Import τα Assets μας, στην συγκεκριμένη περίπτωση το Ak-47 και το στολίζουμε με το Material που έχουμε δημιουργήσει.

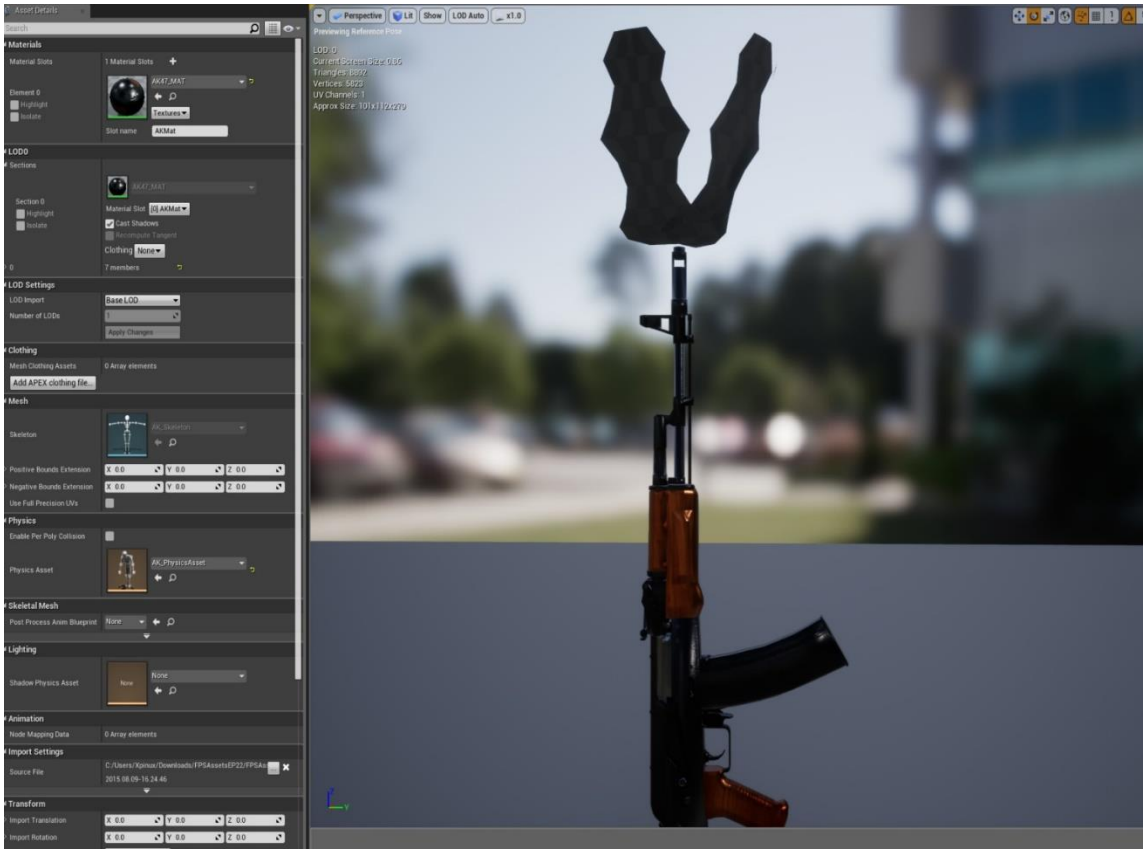


Figure 49 Mesh Όπλου

Το επόμενο βήμα μας είναι να το αναθέσουμε πάνω στον «ηθοποιό» μας. Έτσι πάμε πάνω στο Skeletal mesh του ηθοποιού και κάνουμε το Όπλο Attach πάνω στο χέρι του, ώστε να εμφανίζεται πάνω του και στο σωστό σημείο.

Figure 50 Κανωντας Bind το όπλο

Τέλος πρέπει να αναθέσουμε το όπλο να εμφανίζεται από την αρχή και να εμφανίζεται στον «ηθοποιό» μας. Όποτε πηγαίνουμε στο βασικό μας Blueprint, το blueprint του ηθοποιού, το οποίο λέγεται ThirdPersonBP και γράφουμε τον κώδικα για το όπλο.

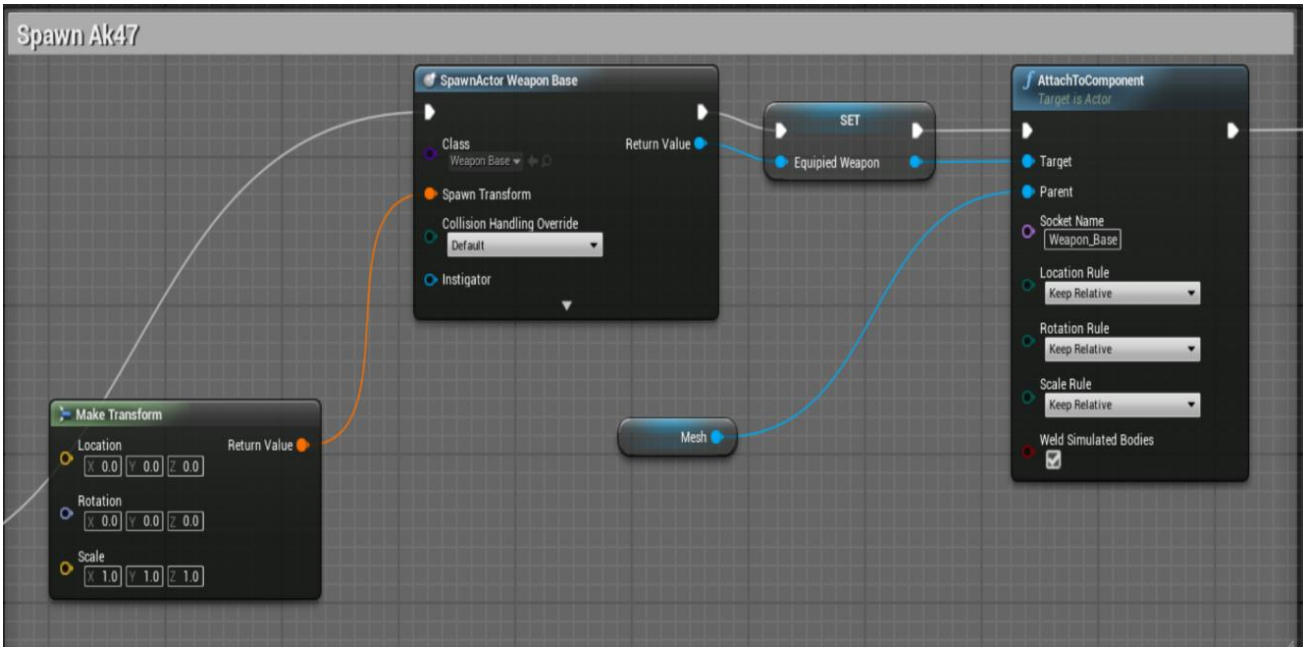
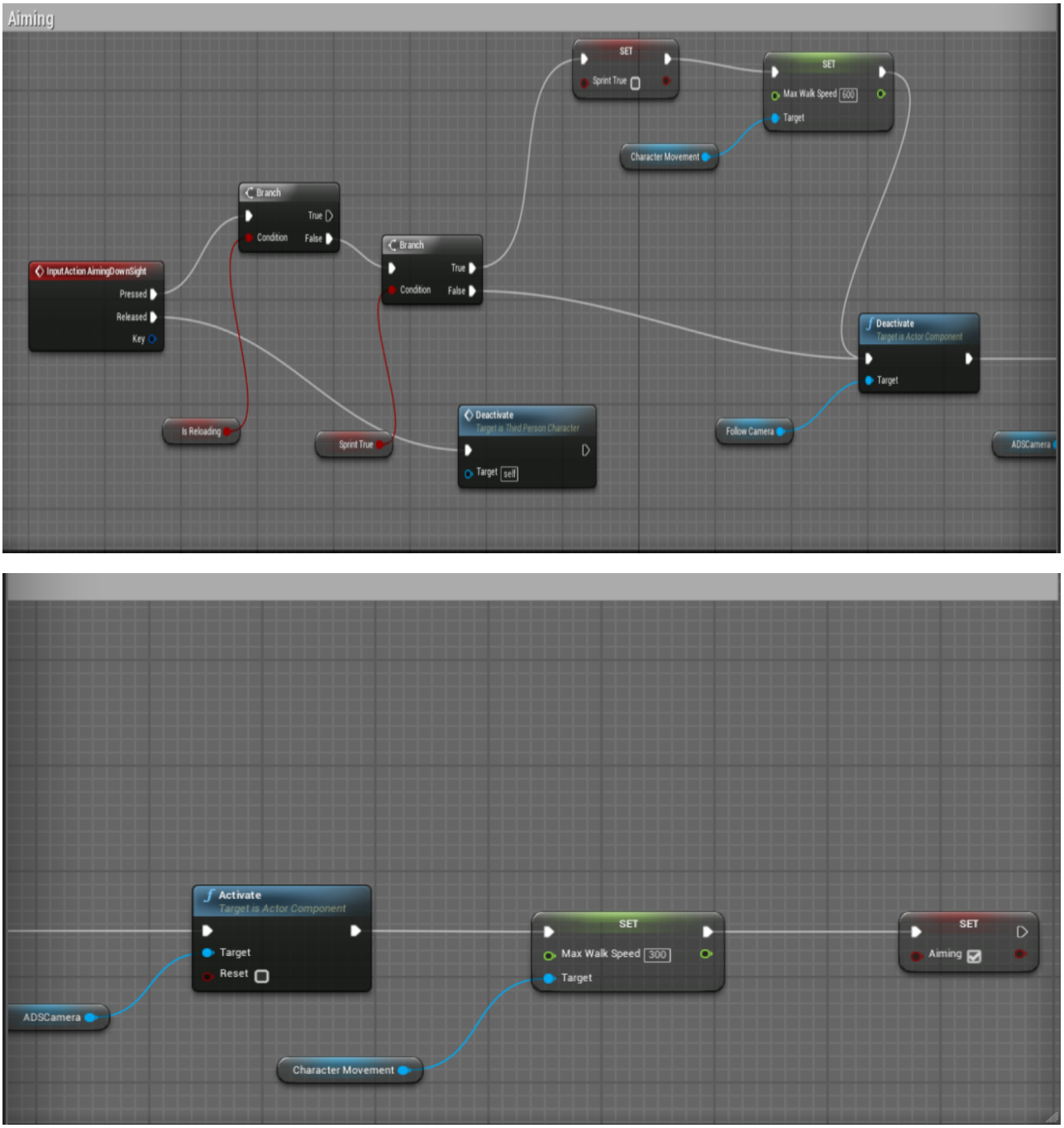


Figure 51 εμφανίζοντας το όπλο.

Aiming Down the Sights

Για να κάνουμε το Aiming Down the Sights, δηλαδή να πατάμε αριστερό κλικ και στο όπλο να μας κάνει zoom in για να βλέπουμε τους αντίπαλους, πρώτα πρέπει να προσθέσουμε στο Viewport άλλη μια κάμερα, μια ADS Camera και να την βάλουμε στο Zoom του όπλου και στην συνέχεια να γράψουμε την λογική.

Η λογική θα λέει, όποτε θα πατήσει ο χρήστης το αριστερό κλικ, θα κάνει δύο ελέγχους πρώτος να δει, εάν γεμίζει το όπλο, κι ο δεύτερος, εάν ο χρήστης τρέχει ο «ηθοποιός». Εάν τρέχει, θα του κατεβάσει την ταχύτητα στην κανονική, στα 600.Θα αλλάξει την Boolean τιμή του Sprint και στην συνέχεια θα κάνει activate την ADS Camera και θα κάνει μια τιμή Boolean Aiming True. Εάν κάνει Reloading, τότε δεν θα κάνει τίποτα.



Figures 52-53 Λογική για Zoom όπλου

Κεφάλαιο 5ο

Reload Mechanism

Στο μέρος 2ο, θα δείξουμε διάφορες λειτουργίες του παιχνιδιού και πως έχουν κωδικοποιηθεί.

Η πρώτη λειτουργία που θα δείξουμε είναι το Reload.Το Reload είναι, όταν το όπλο μας τελειώσει από σφαίρες ή εμείς θέλουμε να το γεμίσουμε. Όταν ο χρήστης πατάει το 'R', τότε θα κάνει Reload ανάλογα με το όπλο που έχει διαλέξει, καθώς και θα κοιτάει μια Boolean μεταβλητή και θα την θέσει, εάν γεμίζει το όπλο ή όχι.

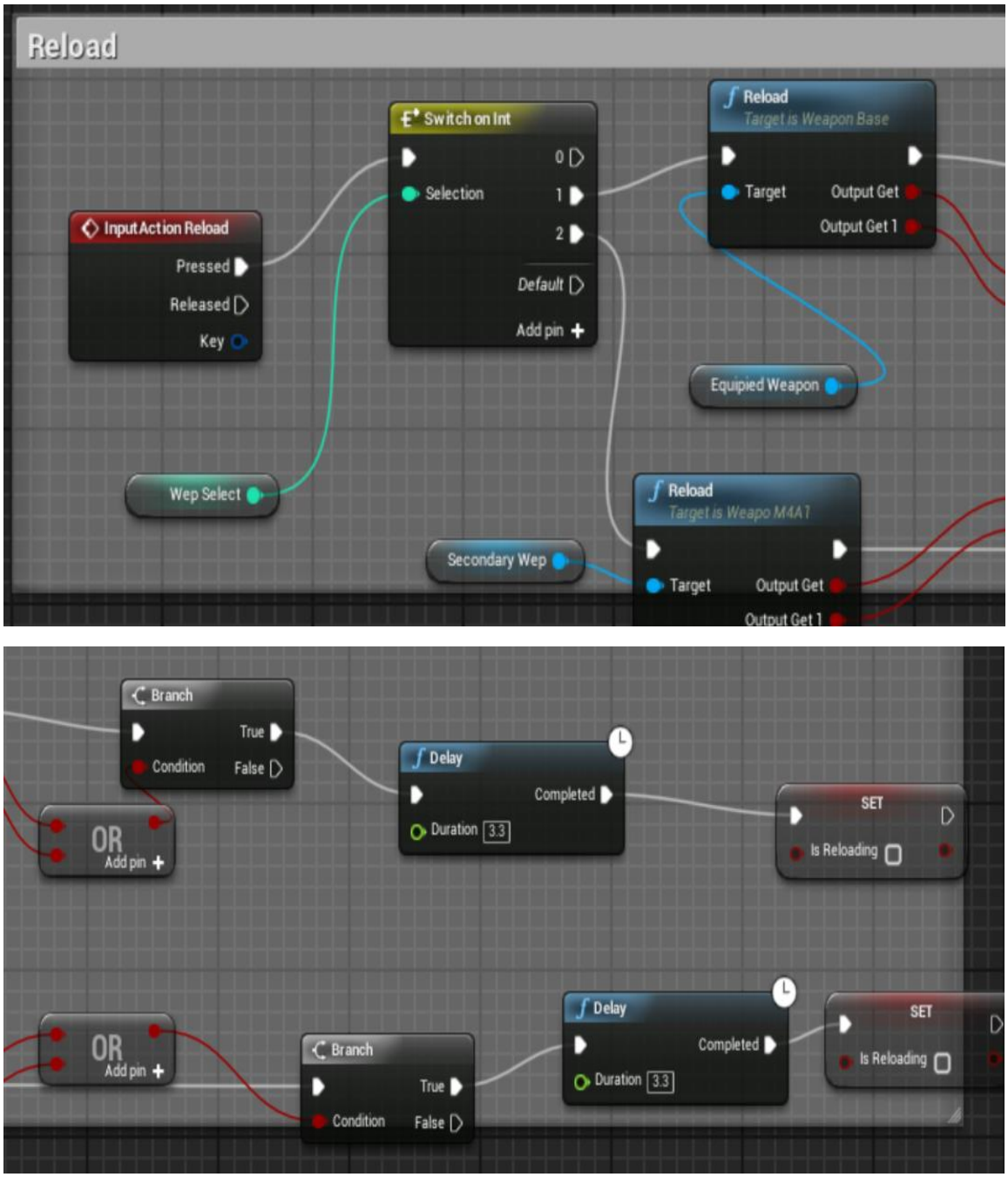


Figure 54 Λογικη ξαναγεμίσματος

FPS Widget

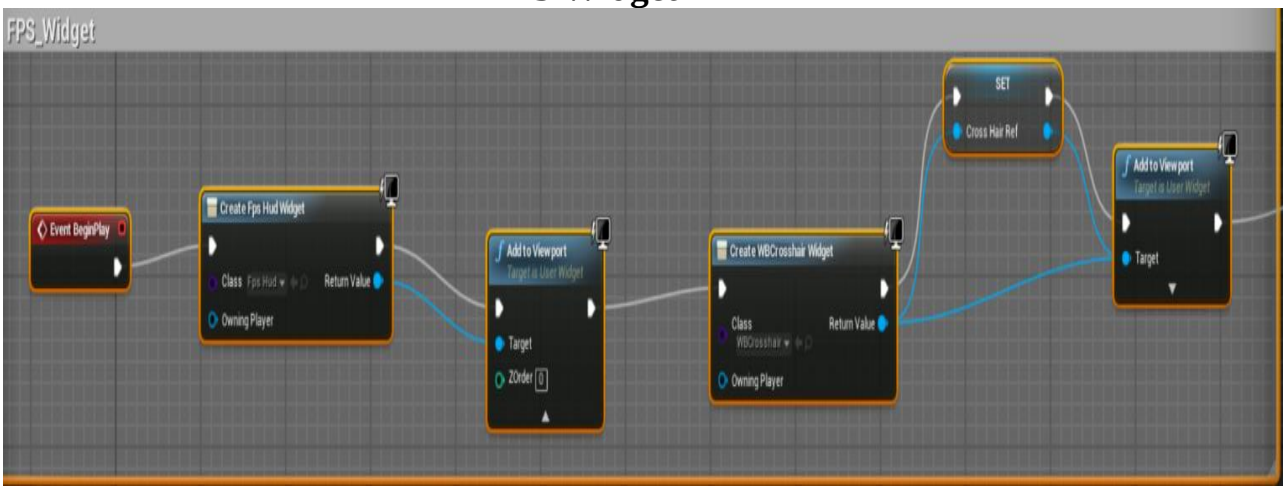


Figure 55 λογική Widget

Το FPS Widget, στην ουσία, είναι το UI που έχουμε σχεδιάσει για να βλέπει ο χρήστης, όταν παίζει το παιχνίδι. Αυτό το UI αποτελείται από τη ζωή και την πανοπλία του χρήστη, έναν μικρό χάρτη για να βλέπει το περιβάλλον και τους εχθρούς, το όπλο, πόσες σφαίρες έχει κλπ. Το Unreal Engine μας επιτρέπει να προγραμματίσουμε ξεχωριστά κάθε Text η το ποσοστό % για την ζωή καθώς και να βάλουμε Objectives, score και μνήματα. Όλα αυτά εμφανίζονται στο UI.

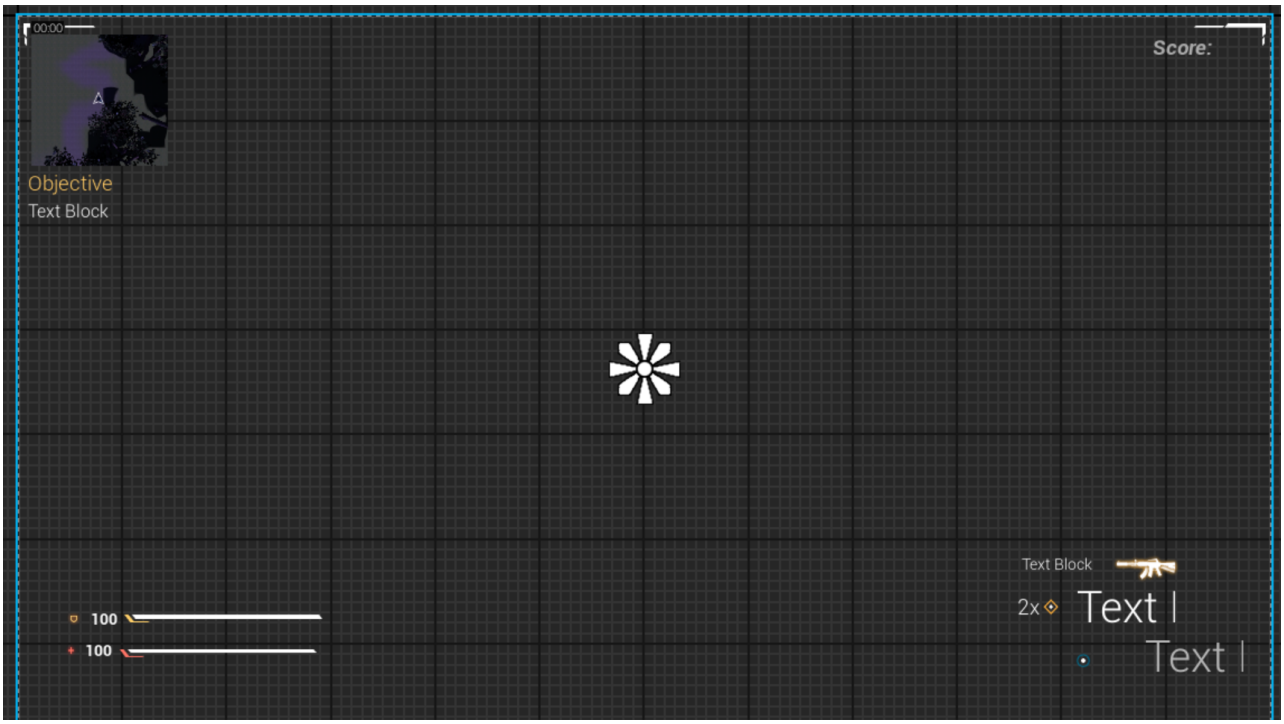


Figure 56 UI

Ενδεικτικά, για να προγραμματίσουμε το Text κάτω από το εικονίδιο του όπλου και να το βάλουμε να παίρνει όσες σφαίρες έχει, πρέπει να δημιουργήσουμε την λογική κάτω στο Text.

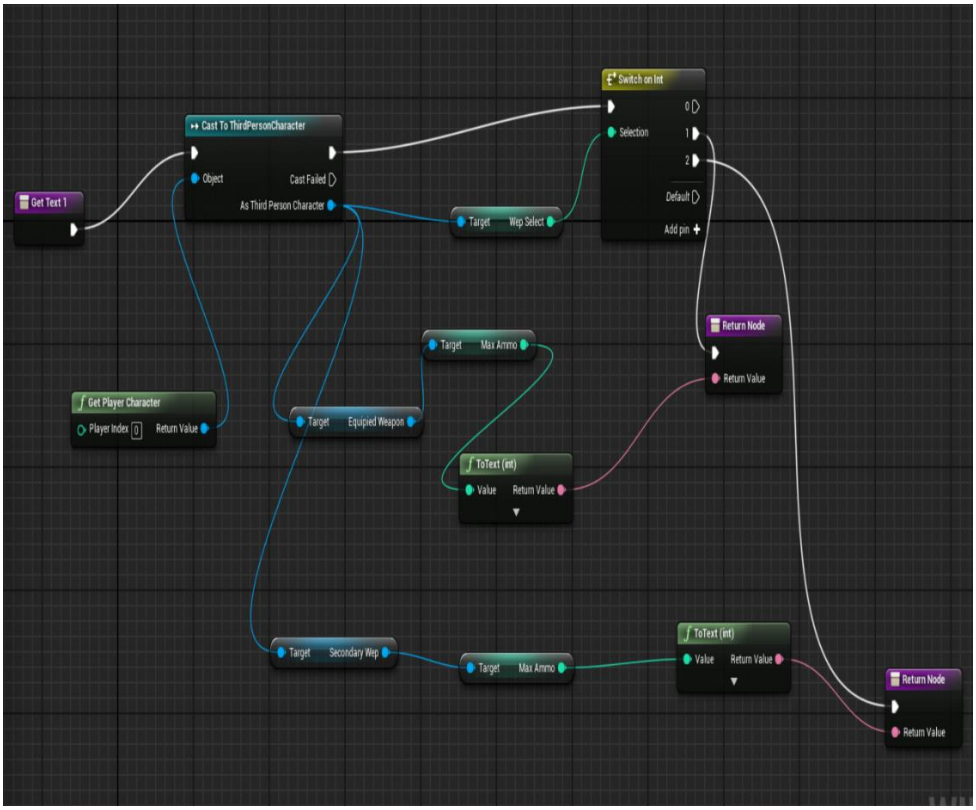


Figure 56 Προγραμματισμός Text Box για να παίρνει πόσες σφαίρες έχει το όπλο

Για να δώσουμε τιμή στο συγκεκριμένο Text, κάνουμε μια αναφορά στο βασικό μας BP, ώστε να μπορούμε να κάνουμε μια αναφορά και να χρησιμοποιήσουμε τις μεταβλητές Ammo, Max Ammo,wep_Select, με σκοπό να εμφανίσουμε το Max_Ammo που έχει το όπλο μας.

Έτσι, με τον ίδιο τρόπο, αλλά με διαφορετική λογική προγραμματίζονται και τα αλλά στοιχεία του UI, λόγω οικονομίας δεν παραθέτω τα κομμάτια του κώδικα, καθώς είναι μεγάλα. Ενδεικτικά, το πώς προσθέτουμε UI, καθώς και πώς του δίνουμε λογική και σχεδιασμό στο παραπάνω παράδειγμα.

Λειτουργία Ζωής και Regen Ζωής

Για να ορίσουμε την Ζωή και την Πανοπλία του χαρακτήρα μας, πηγαίνουμε στο ThirdPersonBP και δημιουργούμε δυο νέες μεταβλητές με το όνομα Health και Armor και τους δίνουμε Default τιμές 1.0, δηλαδή 100%.

Figure 57 δημιουργώντας μεταβλητή Integer ζωής.

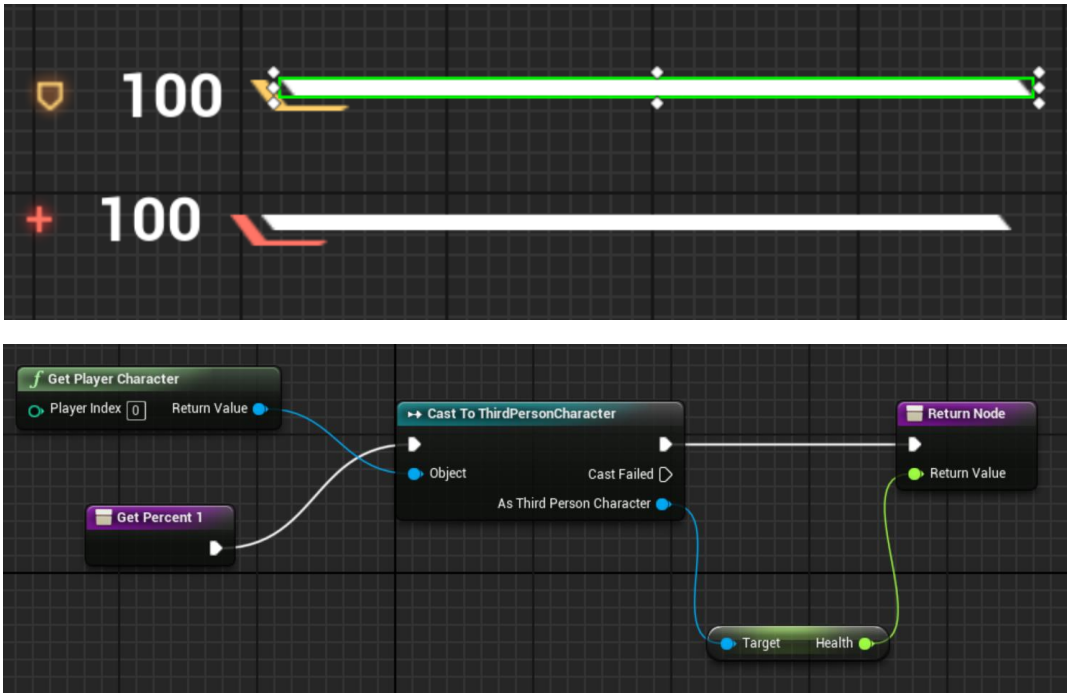


Figure 58 προγραμματίζοντας την μπάρα με τα ποσοστά

Η λογική είναι πολύ απλή, κάνουμε μια αναφορά στο ThirdPersonCharacter δηλαδή στον «ηθοποιό» μας και «παίρνουμε» την μεταβλητή Health και με την εντολή Return Node, καθώς και με την χρήση του Get Percent, η τιμή της Ζωής γυρνάει σαν ποσοστό στην μπάρα και αυξομειώνεται. Ακολουθούμε την ίδια διαδικασία για τα νούμερα δίπλα, καθώς και για την πανοπλία αλλάζοντας την μεταβλητή σε Armor που θα «τραβάμε».

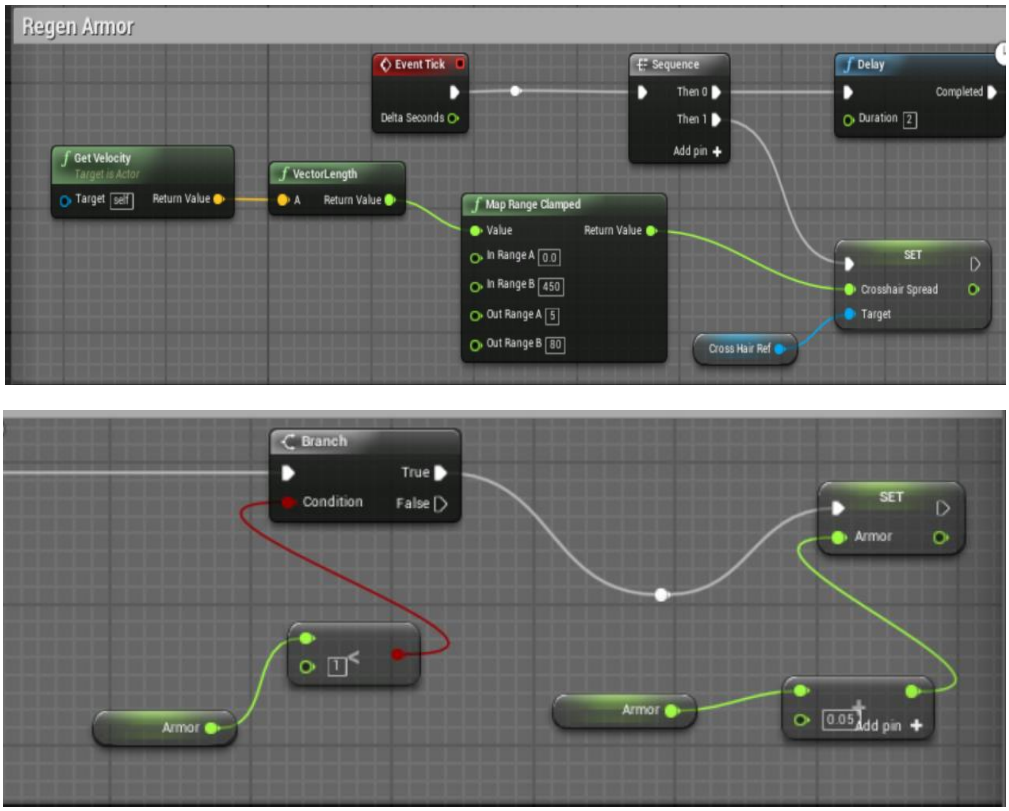


Figure 60-61 Λογική πανοπλίας

Η λογική της Regen είναι, όταν ο «ηθοποιός» δεν τον χτυπάει κάποιο Zombie, τότε να έχει την ικανότητα, άμα του έχει σπάσει η πανοπλία, να μπορεί να γεμίζει ζωή του.

Zombie AI

Η Δημιουργία ενός Zombie γίνεται, όπως δημιουργήσαμε τον βασικό χαρακτήρας μας, απλά αυτήν την φορά, επειδή είναι AI κάνουμε την επιλογή «Pawn». Ακολουθούμε ακριβώς την ίδια διαδικασία για τον προγραμματισμό και τα Animations του περπατήματος και του τρεξίματος, καθώς και ορίζουμε την ζωή του με τον ίδιο ακριβώς τρόπο που κάναμε και στον «ηθοποιό».

Αυτό που το διαφοροποιεί και δίνει την έννοια στο παιχνίδι είναι, ότι είναι το βασικό Ai, το οποίο προσπαθεί να βρει τον «ηθοποιό» και να τον σκοτώσει. Έτσι δίνει έναν σκοπό στο παιχνίδι, ότι για να κερδίσει ο χρήστης πρέπει να σκοτώσει τα Zombie και να ανεβάσει το Score του δίχως να πεθάνει. Έχει 2 βασικές λογικές το Zombie AI :

- Λογική Sensing:Η Λογική Sensing είναι πως, όταν τα Zombie δουν τον παίχτη, τί λογική θα ακολουθούσουν και ποια βήματα θα κάνουν, ώστε να πέτυχουν το σκοπό που τους έχουμε ορίσει, δηλαδή να πάνε πάνω στον παίχτη, ώστε να τον χτυπήσουν.
- Λογική Locate Player:Όταν τα Zombie δεν μπορούν να δουν τον παίχτη, ακολουθούν μια λογική, ώστε να μπορέσουν να τον εντοπίσουν.

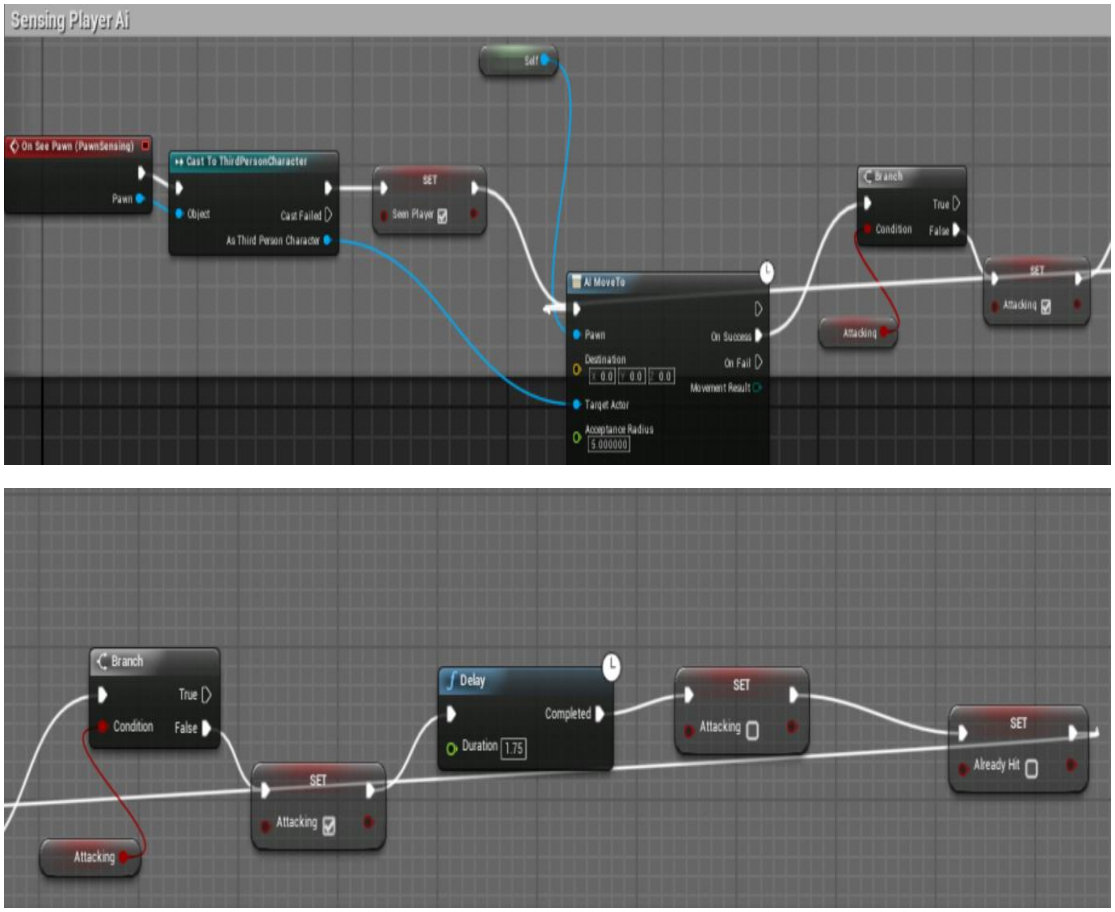


Figure 62 Ανίχνευση παίχτη

Όταν τα Zombie «Δουν» τον παίχτη, τότε ορίζουμε μια μεταβλητή Boolean με το όνομα Seen «True» και με την εντολή AI_move το μετακινούμε το Ai στην θέση που είναι ο ηθοποιός μας. Στην συνέχεια, γίνονται κάποιοι έλεγχοι, εάν το Zombie AI βαράει ήδη τον παίχτη. Εάν δεν τον βαράει, τότε ορίζουμε να τον χτυπήσει θέτοντας μια Boolean μεταβλητή Attacking «True» και κάνοντας ένα Delay 1.75 sec για να δώσουμε χρόνο να προλάβει το Zombie AI να αναπαράγει το Animation της επίθεσης.

Στα Animation που έχουμε βάλει στο Zombie Ai το έχουμε βάλει να χτυπάει με το δεξί χέρι. Οπότε, η λογική που πρέπει να ακολουθήσουμε, ώστε να χτυπήσει τον «ηθοποιό», θα πρέπει να βρίσκεται από το δεξί χέρι.

Πιο συγκεκριμένα, στο Δεξί χέρι πάνω στο Zombie Ai έχουμε αναθέσει ένα Component. Αυτό το Component, όταν θα έρχεται σε επαφή με τον «ηθοποιό», τότε θα καλεί την Συνάρτηση GetHurt, η οποία θα προκαλεί το Damage.

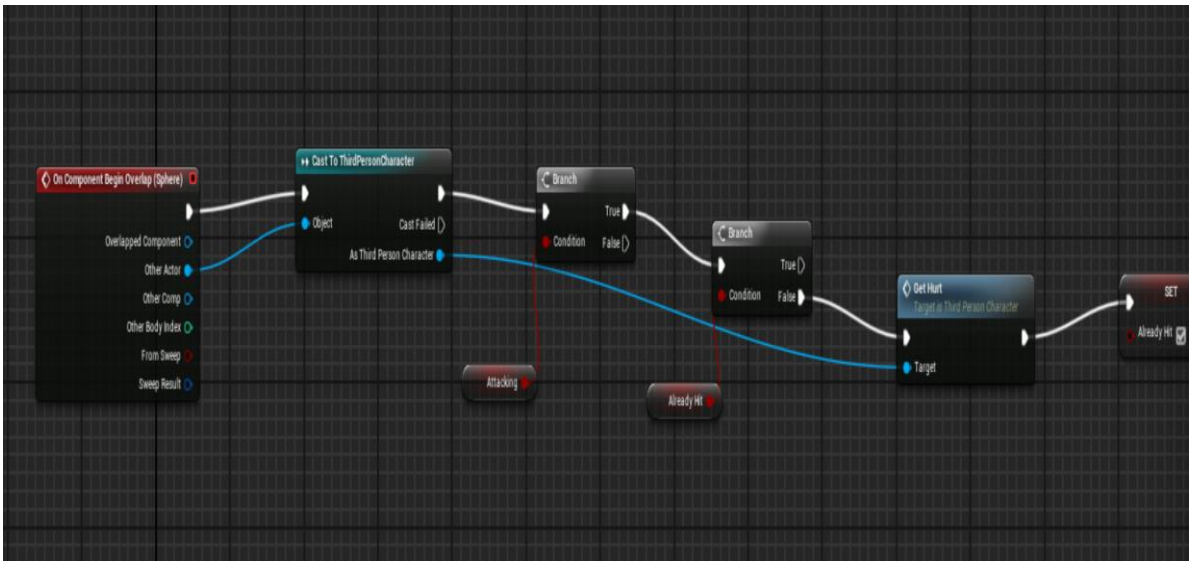


Figure 62 Επίθεση

Στην Λογικη Locate Player, όταν το Ai δν μπορει να δει τον παίχτη, τοτε είναι προγραμματισμενο να κανει ένα Ai move, δηλαδη να κινηθεί σε ένα RandomReachableRadius(σε μία ακτίνα που μπορεί να κινηθεί) με ακτίνα 10000. Στη συνέχεια έχουμε βάλει μία Branch που ελέγχει, εάν από την κίνηση που έκανε, έχει δει τον «ηθοποιό» ή όχι. Εάν τον έχει δει, τότε πάει στην Sensing συνάρτηση, αλλιώς τότε ξανακάνει την κίνηση Ai move, μέχρι να δει τον «ηθοποιό» .

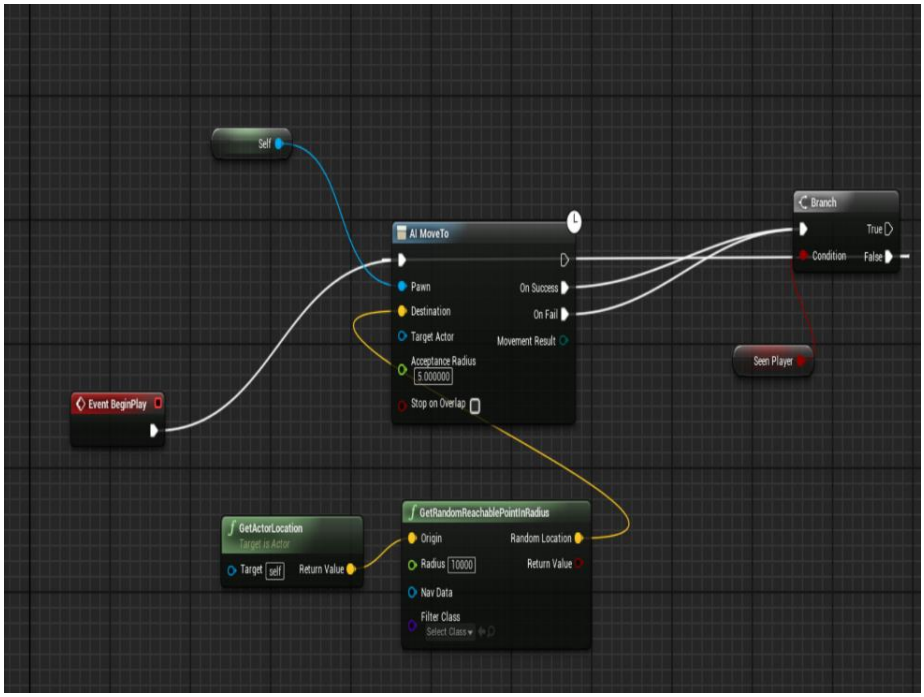


Figure 63 *Λογική της Locate*

Blood Splash Effect

Μέσω του Unreal Engine, μπορούμε κιόλας να δημιουργήσουμε διάφορα εφέ, τα οποία θα εμφανίζονται στην οθόνη, μετά από κάποιο Trigger που τα έχει προκαλέσει. Ένα από αυτά που αναπτυχθήκαν για το παιχνίδι είναι το Blood Splash Effect. Πρόκειται για ένα Εφέ, το οποίο θα κοκκινίζει λίγο την οθόνη του χρήστη, όταν ο χαρακτήρας του αρχίζει και χάνει ζωή.

Figure 64 Blood Splash Effect

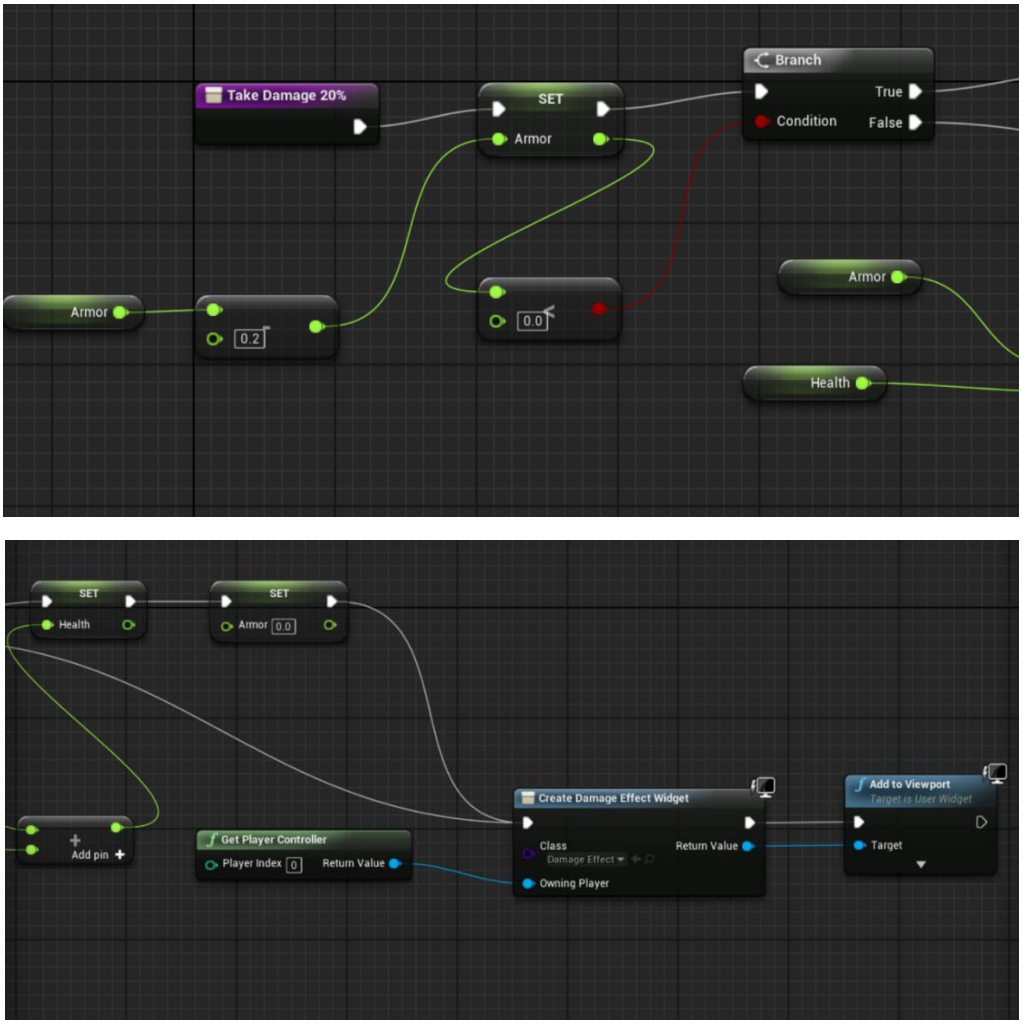


Figure 65 Συνάρτηση Take Damage 20%

Ορίζουμε από τις συναρτήσεις που κάνουμε Damage στον παίχτη, όταν ξεκινήσει το Damage, να πηγαίνει στο Health και να μας εμφανίζει στην οθόνη το Damage Effect με το αίμα που μόλις δημιουργήσαμε. Το μόνο που έχουμε να κάνουμε είναι να αλλάξουμε το Visibility, όταν αυτό το εφέ ενεργοποιείται.

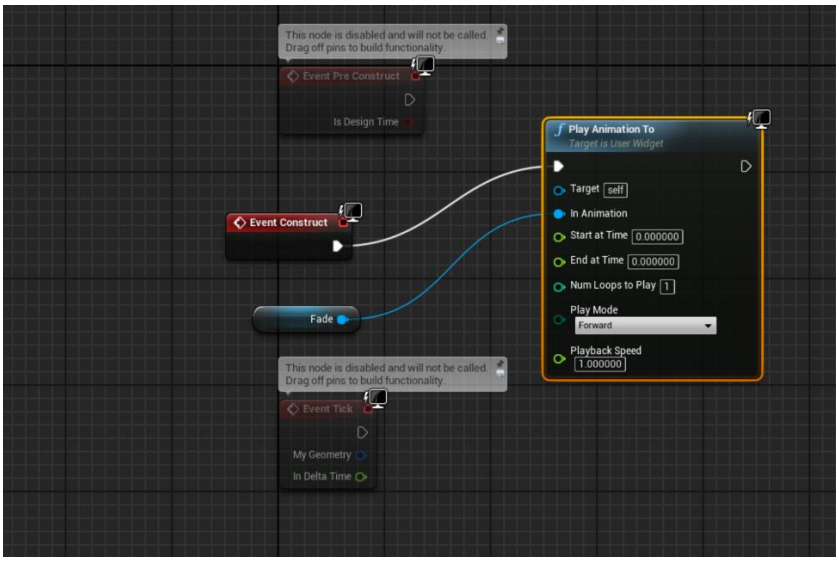


Figure 66 Fade Effect

Κατασκευή Επιπέδου

Το Unreal Engine μας προσφέρει μια μεγάλη ποικιλία από εργαλεία για να κατασκευάσει ο καθένας το επίπεδο που έχει φανταστεί. Τα εργαλεία που προσφέρει είναι από απλά σχήματα, τα οποία μπορείς να τα τροποποιήσεις από δημιουργία μεγάλων και μικρών επιφανειών μέχρι τροποποίηση γεωμετρίας επιπέδου και διάφορα post-processing effect στο επίπεδο για τον κατάλληλο φωτισμό, καθώς και εργαλεία διακόσμησης.

Ενδεικτικά με το Sculpt, μπορούμε να δημιουργήσουμε διάφορες επιφάνειες με βάση την φαντασίας μας, από βουνά και λόφους μέχρι και λιμνούλες ανάλογα τον τρόπο που θα χρησιμοποιήσουμε το συγκεκριμένο εργαλείο. Αφού τελειώσουμε το Sculpting, μπορούμε να δώσουμε με το Paint το χρώμα που έχουμε φανταστεί για το συγκεκριμένο περιβάλλον που δημιουργήσαμε.

Figure 67 Φτιάχνωτας το Level

Figure 68 *Level Designing, sculpting Lake and Mountain.*

Πακετάρισμα παιχνιδιού σε διάφορες πλατφόρμες

Αφότου ένα παιχνίδι έχει τελειώσει την ανάπτυξή του, προτού τελειώσει το Project, θα πρέπει να πακεταριστεί με τον κατάλληλο τρόπο. Το Packaging διασφαλίζει πως όλος ο κώδικας και το περιεχόμενο του Project είναι ενημερωμένο, καθώς και στην κατάλληλη μορφή για να τρέξει στην συγκεκριμένη πλατφόρμα που στοχεύουμε να αναπαραχθεί.

Χρειάζονται μερικά βήματα προτού ολοκληρωθεί η διαδικασία πακεταρίσματος. Πρώτον, θα πρέπει να μεταγλωττιστεί ο πηγαίος κώδικας του Project, ώστε να μην δείξει κανένα σφάλμα. Εφόσον έχει μεταγλωττιστεί ο κώδικας, όλο το περιεχόμενο απαιτείται να βρίσκεται στην κατάλληλη μορφή, ώστε να πακεταριστεί. Αφότου έχουνε γίνει αυτά τα δύο, τότε συγχωνεύονται σε ένα σετ από αρχεία και δημιουργείται ένα Installer.

Για να πακετάρουμε το project μας ακολουθούμε την διαδικασία:

Figure 4 "Cooking" Project για Windows

Κεφάλαιο 6ο
Συμπέρασμα

Συμπερασματικά, δουλεύοντας πάνω στο Unreal Engine, καταλαβαίνουμε ότι η ανάπτυξη παιχνιδιών είναι ένα χρονοβόρο και περίπλοκο ζήτημα, καθώς χρειάζονται διάφοροι άνθρωποι με συγκεκριμένες γνώσεις για να αναπτυχθεί ένα πετυχημένο παιχνίδι. Δηλαδή, χρειάζονται Designers, Coders, Animators, Story Makers, Testers, Asset Creators κλπ. Έτσι, καταλαβαίνουμε πως η ανάπτυξη ενός απλού παιχνιδιού, όπως το δικό μας που έχει έναν συγκεκριμένο σκοπό, είναι μία αρκετά χρονοβόρα και πολύπλοκη διαδικασία για έναν που πρέπει να αναλαμβάνει όλους τους ρόλους που απαιτούνται για την ανάπτυξη ενός παιχνιδιού. Ενδεικτικά, ο κώδικας μόνο του «ηθοποιού» είναι αρκετά μεγάλος και αρκετά περίπλοκος.

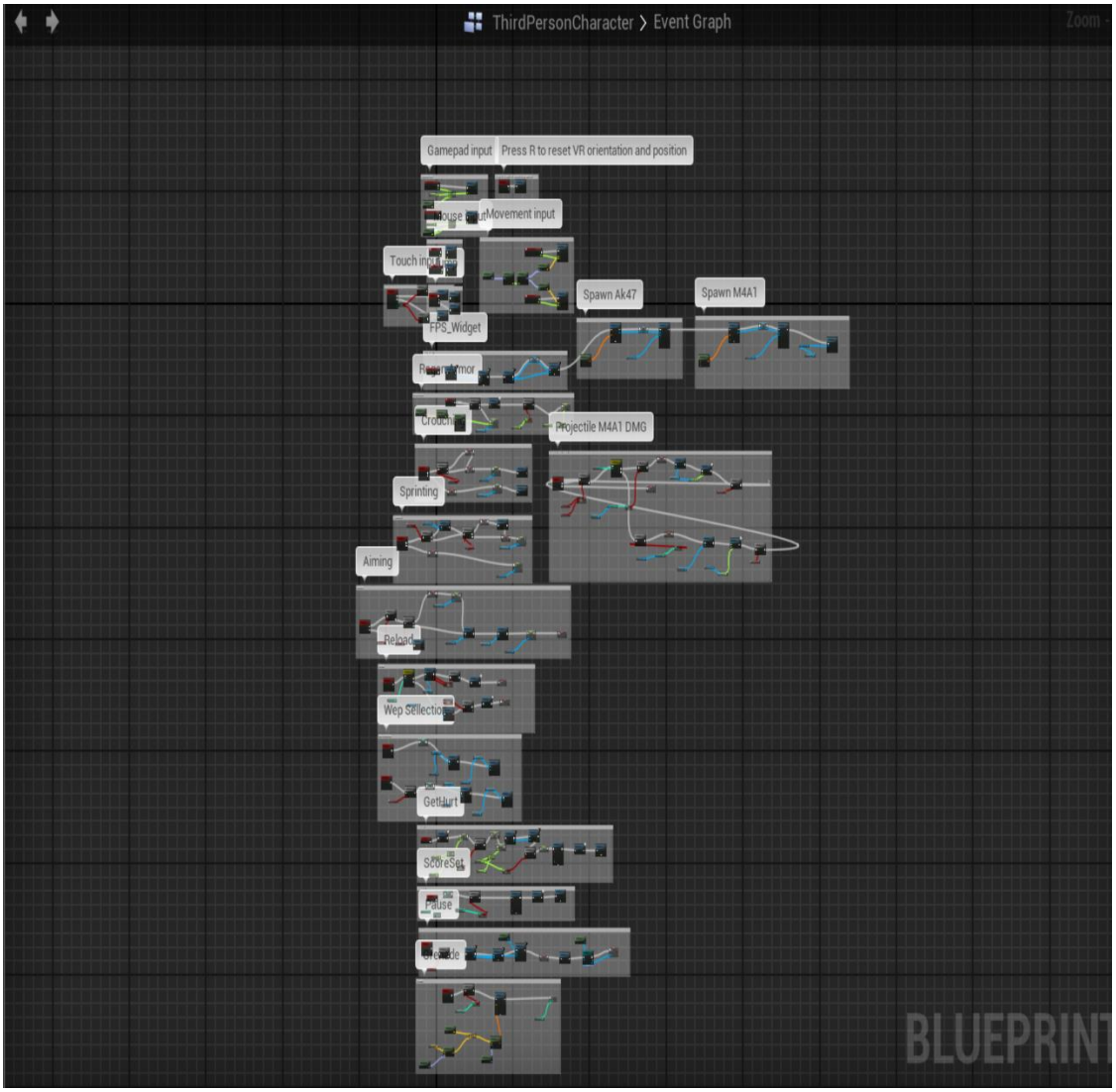


Figure 5 Όλος ο Κώδικας μόνο για το ThirdPersonBP

Παρόλα αυτά, το Unreal Engine είναι μια μηχανή, η οποία προσφέρει μια μεγάλη ποικιλία από εργαλεία, τα οποία δίνουν την δύναμη στον Developer να δημιουργήσει το έργο που έχει στο μυαλό του με πιο αποτελεσματικό τρόπο σε σχέση με τις τεχνικές που υπήρχαν παλιότερα για την ανάπτυξη παιχνιδιών. Είναι μία μηχανή, η οποία προσφέρει

από Level Designing εργαλεία μέχρι ακόμα και εργαλεία για Animation, μια μηχανή, η οποία σε διευκολύνει κατά πολύ κι έχει ωθήσει τις εταιρίες να αναπτύσσουν παιχνίδια και να τα λανσάρουν στην αγορά με πολύ γρήγορο τρόπο δημιουργώντας κέρδη, καθώς και την συμβολή της στο ξεκίνημα της βιομηχανίας των παιχνιδιών.

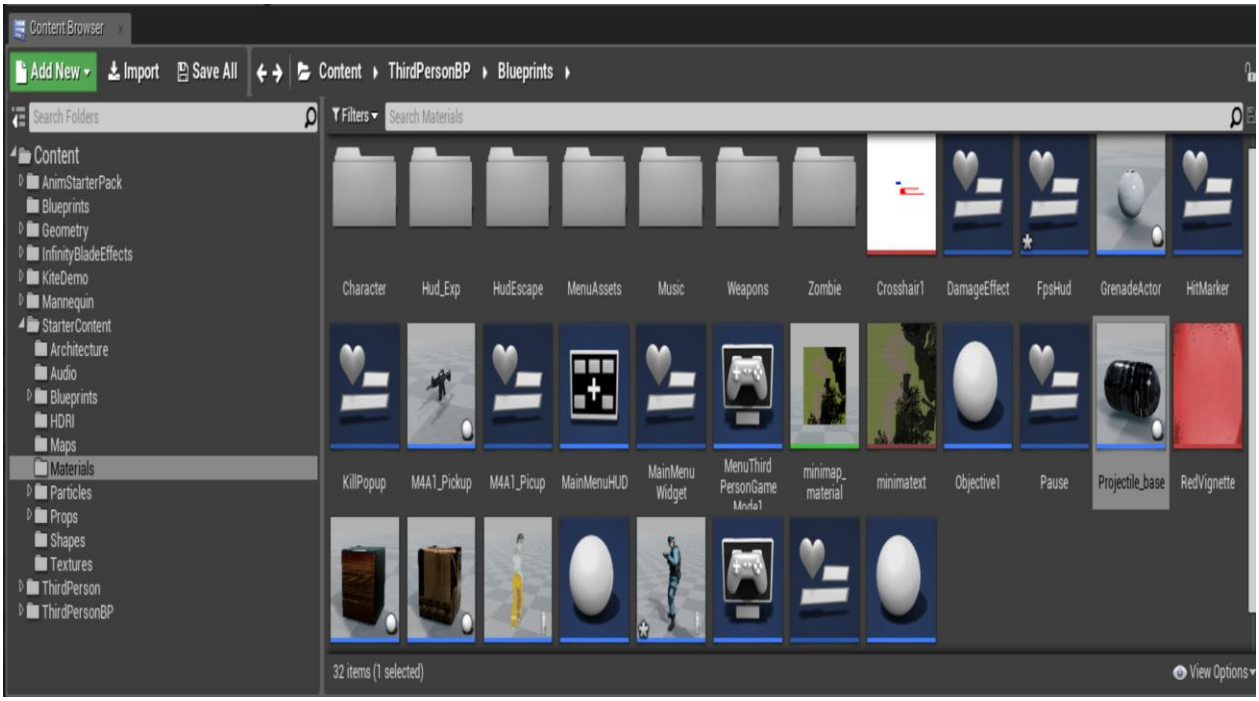


Figure 6 Διαφορά εξαρτήματα και Blueprints που Δημιουργήθηκαν και προγραμματίστηκαν για την δημιουργία του Παιχνιδιού.

Με την χρήση του Unreal Engine 4 καταφέραμε να δημιουργήσουμε ένα ολοκληρωμένο παιχνίδι, εξερευνώντας και εφαρμόζοντας τις περισσότερες πτυχές των εργαλείων του στην δημιουργία διαφόρων λειτουργιών από προγραμματισμό λογικής μέχρι πώς εφαρμόζουν και προγραμματίζονται τα Animation, καθώς και Level Designing και δημιουργία ενός AI.

Τέλος με την παρούσα πτυχιακή, μας δόθηκε η ευκαιρία να μάθουμε πώς αναπτύσσεται ένα παιχνίδι με Unreal Engine 4, καθώς και να μάθουμε διαδικασίες και τεχνικές για την ανάπτυξη ενός παιχνιδιού από τον προγραμματισμό μέχρι το σχεδιασμό. Στην πορεία της πτυχιακής εμφανίστηκαν πολλές δυσκολίες, οι οποίες σχετίζονταν με την ανάπτυξή του παιχνιδιού, καθώς πάρθηκε η πρωτοβουλία ανάπτυξής του από ένα άτομο δίχως αρχική εμπειρία στην ανάπτυξη παιχνιδιών, καθώς έπρεπε να αναλάβω από τον ρολό του Designer μέχρι του Coder. Αλλά, τελικώς, οι όποιες δυσκολίες προέκυψαν, αντιμετωπίστηκαν και, εν κατακλείδι, το παιχνίδι αναπτύχθηκε επιτυχώς, προσφέροντας πια εμπειρία πάνω στο εργαλείο Unreal Engine 4.

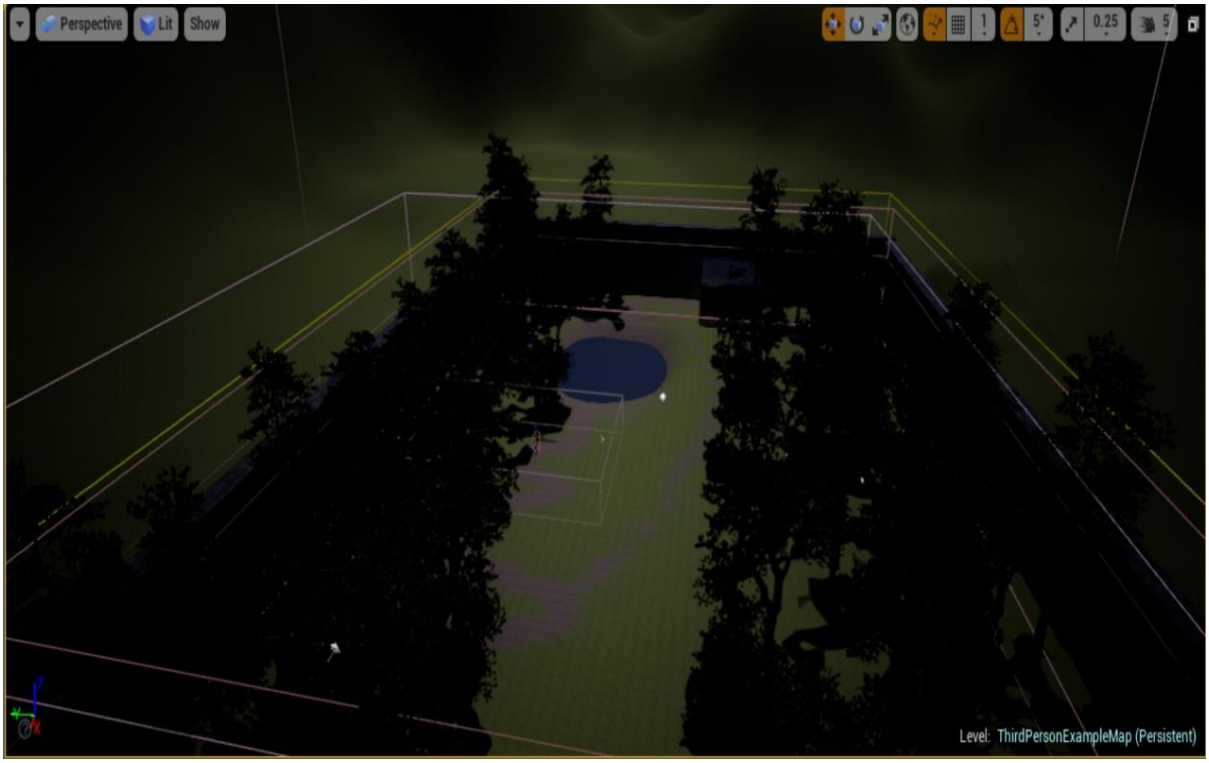


Figure 7 Το παιχνίδι της πτυχιακής.

Αναφορές

Keighley, Geoffrey. (2001, May 19). *GameSpot*. Retrieved from https://web.archive.org/web/20010519154729/http://www.gamespot.com/features/makeunreal/

(2008, April 28). Retrieved from GameCarrierGuide: https://www.gamecareerguide.com/features/529/what_is_a_game_.php?page=2

Bourg, D. M. (2002). *Physics for Game Developers.* O'Reilly & Associates.

Brandon, A. (2004). Audio for Games: Planning, Process, and Production. *New Riders*.

Brown, F. (2014). The Unreal Engine Marketplace is open for business. *PCGamesN*.

Caron, F. (2008, March 31). *Unreal Engine 4 to "exclusively target" next-gen consoles*. Retrieved from arstechnica: https://arstechnica.com/gaming/2008/03/unreal-engine-4-to-exclusively-target-next-gen-consoles/

Chikhani, R. (2015, October 31). *techcrunch*. Retrieved from https://techcrunch.com/2015/10/31/the-history-of-gaming-an-evolving-community/

Cohen, D. (2019, June 1). Retrieved from https://www.lifewire.com/atari-2600-console-729665

Dalmau, D. S.-C. (2003, October 1). *GamaSutra*. Retrieved from https://www.gamasutra.com/view/feature/2797/product_review_speedtree_rt__.php

Denham, T. (2016). *conceptartempire*. Retrieved from https://conceptartempire.com/what-is-unreal-engine/

Develop-online.net. (2007, July 6). Rise of Middleware.

Devin, Connors. (2014). Epic Games Opens Unreal Engine Marketplace to Developers. *The Escapist*.

Engineering, I. (2016, November 2). *Interesting Engineering*. Retrieved from https://interestingengineering.com/how-game-engines-work

(2015). *ESA-Essential-Facts-2015.* ESA.

Frank, A. (2018). *Epic Games is launching its own store, and taking a smaller cut than Steam*. Retrieved from polygon.com: https://www.polygon.com/2018/12/4/18125498/epic-games-store-details-revenue-split-launch-date

Games, E. (2019). Retrieved from https://docs.unrealengine.com/en-US/Engine/Basics/Projects/Packaging/index.html

Gaudiosi, J. (2015). *Why Epic Games is giving away its game technology*. Retrieved from fortune.com: https://fortune.com/2015/03/03/epic-games-unreal-tech-free/

Gazette.net. (2007, August 31). *Gazette.net.* Retrieved from http://www.gazette.net/stories/083107/businew11739_32356.shtml

Hauteville, C. (2011, April 2). *Aim systems in First Person Shooters.* Retrieved from Technical Game Design: http://technicalgamedesign.blogspot.com/2011/04/aim-systems-in-first-person-shooters.html

Herz, J. (1999). GAME THEORY; For Game Maker, There's Gold in the Code. *The New York Times*.

HUB, V. G. (2019). *FPS Tutorial*. Retrieved from YouTube: https://www.youtube.com/user/VirtusEdu/

Insights, C. (2018). *The Two Engines Driving the $120B Gaming Industry Forward.* cbinsights.com.

Kennedy, B. (2002, July 2). Uncle Sam Wants You (To Play This Game). *The New York Times*. Retrieved from GamaSutra.

Krause, S. (2007, October 2). *Java vs. C benchmark.* Retrieved from https://www.stefankrause.net/wp/?p=4

Laird and Jamin. (2003). *The Game Development Process.* Michigan: worcester polytechnic institute.

Lightbown, D. (2018, January 9). Classic Tools Retrospective: Tim Sweeney on the first version of the Unreal Editor. *GamaSutra*.

Ligman, K. (2013). *See Epic's Unreal Engine 3 running in HTML5*. Retrieved from Gamasutra: https://www.gamasutra.com/view/news/191642/See_Epics_Unreal_Engine_3_running_in_HTML5.php

Lily, Paul. (2009, July 21). Doom to Dunia: A Visual History of 3D Game Engines. *Maximum PC*.

Madison. (n.d.). *Report on Use of Middleware in Games.*

Maximum, P. (2004). "Game Engines – Exposed! *Maximum PC*.

McAloon, A. (2017). *Fortnite: Battle Royale prompted engine-wide Unreal improvements*. Retrieved from gamasutra.com: https://www.gamasutra.com/view/news/307051/Developing_Fortnite_Battle_Royale_prompted_enginewide_Unreal_improvements.php

McLean-Foreman, J. (2001, April 6). n Interview with Epic Games' Tim Sweeney. *GamaSutra*.

Miller Michael . (2005, April 1). *A History of Video Game Consoles*. Retrieved from https://web.archive.org/web/20071012152258/http://www.informit.com/articles/article.aspx?p=378141

Moore, Michael E.; Novak, Jeannie. (2010). Game Industry Career Guide. Cengage Learning.

M-trends.org. (2011, December 8). *M-trends.org.* Retrieved from http://www.m-trends.org/2008/01/mobile-and-wireless-trends-for-2008.html

Parrish, K. (2005). *Epic Revealing Unreal Engine 4 Later This Year*. Retrieved from tomshardware: https://www.tomshardware.com/news/Epic-Games-Unreal-Engine-4-UE4-Mark-Rein-DICE-2012,14682.html

PCGamesN. (2014). Making it in Unreal: How Daylight survived public pressure and became the very first UE4 game". *PCGamesN*.

Reed, K. (2004, July 4). *Unreal Engine 3*. Retrieved from Eurogamer.net: https://www.eurogamer.net/articles/i_epicgames_june04

Rein,Houlihan, John. (2005). Rein: "We've been working on Unreal Engine 4 for two years. *Computer and Video Games.*

Savage, Phil. (2015, March 2). Retrieved from pcgamer.com: https://www.pcgamer.com/unreal-engine-4-is-now-free/

sofasandsectionals. (2019). *sofasandsectionals*. Retrieved from https://www.sofasandsectionals.com/history-of-nintendo-entertainment-system

Staff, I. (2012). *Epic Games Announces Unreal Development Kit, Powered by Unreal Engine 3*. Retrieved from IGN: https://www.ign.com/articles/2009/11/05/epic-games-announces-unreal-development-kit-powered-by-unreal-engine-3

Staff, I. (2012). *Epic Games Releases Unreal Development Kit with iOS Support*. Retrieved from IGN: Epic Games Releases Unreal Development Kit with iOS Support

Staff, I. (2012). *Tim Sweeney Looks Ahead*. Retrieved from IGN: https://www.ign.com/articles/1998/10/24/tim-sweeney-looks-ahead

Stefan Zerbst, Oliver Düvel. (2004). *3D Game Engine Programming.* Premier Press.

Sweeney. (1998). Game Theory. *Maximum PC*, p. https://books.google.gr/books?id=1AEAAAAAMBAJ&lpg=PT8&dq=Tim+Sweeney+epic&pg=PT8&redir_esc=y#v=onepage&q=Tim%20Sweeney%20epic&f=false.

Sykes, T. (2014). Unreal Engine 4 now free for academic use. *PC Gamer*.

Techopedia. (2019). *Techopedia*. Retrieved from https://www.techopedia.com/definition/17141/video-game-console

The 10 Best Video Game Engines | 2018 Edition. (2017, March 11). Retrieved from gamesdesigning.org: https://www.gamedesigning.org/career/video-game-engines/

The Real cost of Middleware. (2008). *GameDaily*.

Thier, D. (2012). Epic's Tim Sweeney on How Unreal Engine 4 Will Change The Way Games Are Made, and Why You Care. *Forbes*.

Thomas, M. (2012). *Realtime Global Illumination techniques collection*. Retrieved from extremeistan: https://extremeistan.wordpress.com/2014/05/11/realtime-global-illumination-techniques-collection/

Thomsen, M. (2012, June 15). *History of the Unreal Engine*. Retrieved from IGN: https://www.ign.com/articles/2010/02/23/history-of-the-unreal-engine

Tim, S. (2005). Retrieved from https://developer.nvidia.com/gpugems/GPUGems2/gpugems2_frontmatter.html

Totilo, S. (2012). *kotaku*. Retrieved from How Unreal Engine 4 Will Change The Next Games You Play: https://kotaku.com/how-unreal-engine-4-will-change-the-next-games-you-play-5916859

Tyson, J. (2019). Retrieved from https://electronics.howstuffworks.com/video-game2.htm

Unity. (2019). Retrieved from Unity3d: https://unity3d.com/what-is-a-game-engine

unrealengine. (2012). *Zombie Studios Licenses Unreal Engine 4*. Retrieved from https://www.unrealengine.com/en-US/blog/zombie-studios-licenses-unreal-engine-4

UnrealEngine. (2019). Retrieved from https://www.unrealengine.com/en-US/

unrealengine. (2019). *Engine Features*. Retrieved from unrealengine: https://docs.unrealengine.com/en-US/Engine/index.html

Wikipedia. (2016). *Wikipedia*. Retrieved from https://en.wikipedia.org/wiki/Game_engine#cite_ref-11

Wikipedia. (2019). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Video_game_development

Wikipedia. (2019). *NES*. Retrieved from https://en.wikipedia.org/wiki/Nintendo_Entertainment_System

Wikipedia. (2019). *Pong*. Retrieved from https://en.wikipedia.org/wiki/Pong

World, 3. (2019, January). Unreal Vs Unity. *3D World 242*.

Yagoda, M. (2008). *1972 Nutting Associates Computer Space*. Retrieved from https://web.archive.org/web/20081228061939/http://www.marvin3m.com/arcade/cspace.htm

Στεναμερίδης, Π. (2016, April 16). Ορισμός Βιντεοπαιχνδιού.