ΑΝΩΤΑΤΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΠΕΛΟΠΟΝΝΗΣΟΥ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ Τ.Ε.

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

"Χρήση του Arduino για το χειρισμό τρισδιάστατου αντικειμένου ή χαρακτήρα, στη μηχανή ανάπτυξης παιχνιδιών Unreal Engine 4 (UE4)"

Αλκαίος-Δημήτριος Α. Αναγνωστόπουλος
Α.Μ.: 2010201

Επιβλέπων Καθ.: Σταύρος Δεληγιαννίδης, Καθηγητής Εφαρμογών

ΣΠΑΡΤΗ
ΣΕΠΤΕΜΒΡΙΟΣ 2018

ΑΝΩΤΑΤΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΠΕΛΟΠΟΝΝΗΣΟΥ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ Τ.Ε.

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

"Χρήση του Arduino για το χειρισμό τρισδιάστατου αντικειμένου ή χαρακτήρα, στη μηχανή ανάπτυξης παιχνιδιών Unreal Engine 4 (UE4)"

Αλκαίος-Δημήτριος Α. Αναγνωστόπουλος
Α.Μ.: 2010201

Επιβλέπων Καθ.: Σταύρος Δεληγιαννίδης, Καθηγητής Εφαρμογών

ΣΠΑΡΤΗ
ΣΕΠΤΕΜΒΡΙΟΣ 2018

ΠΕΡΙΛΗΨΗ

Στην παρούσα πτυχιακή εργασία, μελετήθηκε η αλληλεπίδραση ενός μικροελεγκτή Arduino ως προς το περιβάλλον του Η/Υ και κατ' επέκταση, το περιβάλλον της μηχανής ανάπτυξης παιχνιδιών Unreal Engine 4. Για την διεκπεραίωση του έργου χρειάστηκε η εκμάθηση προγραμματισμού του μικροελεγκτή Arduino, ώστε να χρησιμοποιηθεί ως χειριστήριο στο τρισδιάστατο περιβάλλον της Unreal Engine 4.

Εν συνεχεία, πραγματοποιήθηκε η κατασκευή ενός τρισδιάστατου χαρακτήρα σε τρισδιάστατο ψηφιακό περιβάλλον, η κατασκευή ανεξάρτητων μοντέλων Τεχνητής Νοημοσύνης καθώς και η παραμετροποίηση του ίδιου του περιβάλλοντος στο οποίο θα αλληλεπιδρούν τόσο ο χαρακτήρας όσο και τα μοντέλα.

Καθοριστικός παράγοντας για την εκπόνηση του έργου, ήταν ο έλεγχος ορθής λειτουργίας του μικροελεγκτή Arduino ως προς τον ολοκληρωμένο τρισδιάστατο κόσμο της Unreal Engine 4, κατά τη διάρκεια όλων των φάσεων της υλοποίησης.

Λέξεις κλειδιά: Arduino, Leonardo, Unreal Engine 4, Sparkfun Joystick Shield, μικροελεγκτής, τρισδιάστατο μοντέλο, τεχνητή νοημοσύνη, χειριστήριο.

ABSTRACT

In the present diploma thesis, the interaction of an Arduino microcontroller with the PC environment and thus the environment of the game engine used, Unreal Engine 4, it was studied. In order to accomplish this project, it was needed to study and learn the Programming language behind Arduino microcontroller, so it could be used as a gamepad in Unreal Engine 4's virtual 3D environment.

In the second phase of this project, were required the construction of a 3D model in a 3D model environment, the construction of independent Artificial Intelligent models and the configuration of the 3D environment itself, where the character and the AI models will react.

A key factor in this thesis, was to test and inspect the functionality of Arduino microcontroller within the constructed 3D world of Unreal Engine 4, in all phases of implementation.

Keywords: Arduino, Leonardo, Unreal Engine 4, Sparkfun Joystick Shield, microcontroller, 3D model, Artificial Intelligence, gamepad.

ΔΗΛΩΣΗ ΜΗ ΛΟΓΟΚΛΟΠΗΣ ΚΑΙ ΑΝΑΛΗΨΗΣ ΠΡΟΣΩΠΙΚΗΣ ΕΥΘΥΝΗΣ

"Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ενυπογράφως ότι είμαι αποκλειστικός συγγραφέας της παρούσας Πτυχιακής Εργασίας, για την ολοκλήρωση της οποίας κάθε βοήθεια είναι πλήρως αναγνωρισμένη και αναφέρεται λεπτομερώς στην εργασία αυτή. Έχω αναφέρει πλήρως και με σαφείς αναφορές, όλες τις πηγές χρήσης δεδομένων, απόψεων, θέσεων και προτάσεων, ιδεών και λεκτικών αναφορών, είτε κατά κυριολεξία είτε βάση επιστημονικής παράφρασης. Αναλαμβάνω την προσωπική και ατομική ευθύνη ότι σε περίπτωση αποτυχίας στην υλοποίηση των ανωτέρω δηλωθέντων στοιχείων, είμαι υπόλογος έναντι λογοκλοπής, γεγονός που σημαίνει αποτυχία στην Πτυχιακή μου Εργασία και κατά συνέπεια αποτυχία απόκτησης του Τίτλου Σπουδών, πέραν των λοιπών συνεπειών του νόμου περί πνευματικών δικαιωμάτων. Δηλώνω, συνεπώς, ότι αυτή η Πτυχιακή Εργασία προετοιμάστηκε και ολοκληρώθηκε από εμένα προσωπικά και αποκλειστικά και ότι, αναλαμβάνω πλήρως όλες τις συνέπειες του νόμου στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δε μου ανήκει διότι είναι προϊόν λογοκλοπής άλλης πνευματικής ιδιοκτησίας."

Όνομα και Επώνυμο Συγγραφέα (Με Κεφαλαία):
ΑΛΚΑΙΟΣ-ΔΗΜΗΤΡΙΟΣ Α. ΑΝΑΓΝΩΣΤΟΠΟΥΛΟΣ

Υπογραφή (Ολογράφως, χωρίς μονογραφή): ……...…….....

Ημερομηνία (Ημέρα – Μήνας – Έτος): ……....

Η παρούσα πτυχιακή εργασία
είναι αφιερωμένη στην μητέρα μου

ΕΥΧΑΡΙΣΤΙΕΣ

Μέσα από την παρούσα πτυχιακή εργασία θα ήθελα να ευχαριστήσω ιδιαιτέρως, τον καθηγητή μου κ.Σταύρο Δεληγιαννίδη, καθώς υπήρξε καθοδηγητικός σε όλες τις φάσεις της δημιουργίας της. Πέρα από την καθοδήγησή του, σημαντικό ρόλο για την ανάληψη του παρόντος θέματος, ήταν η προτροπή του ιδίου να ασχοληθώ με την κατασκευή παιχνιδιών εφόσον γνώριζε τις προτιμήσεις του.

Ακόμα ευχαριστώ θερμά, τον καθηγητή μου κ.Ιωάννη Λιαπέρδο, για τη βοήθειά του κατά τη διάρκεια των σπουδών μου τόσο σε ακαδημαϊκό όσο και σε προσωπικό βαθμό.

Τέλος, πρέπει να αποδώσω το μεγαλύτερο “ευχαριστώ”, στην οικογένειά μου για την απέραντη υποστήριξή τους απέναντί μου, καθ’όλη τη διάρκεια των σπουδών μου.

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΡΟΛΟΓΟΣ

Η παρούσα εργασία, είναι αποτέλεσμα μελέτης και υλοποίησης δύο ξεχωριστών τεχνολογιών με σκοπό τη διασύνδεση και αλληλεπίδρασή τους στο περιβάλλον του Η/Υ.

Κατά το 1ο μέρος της εργασίας, παραθέτονται όλες οι πληροφορίες σχετικά με τη μελέτη του μικροελεγκτή Arduino, μιας πλακέτας που έφερε την επανάσταση στον κόσμο της ηλεκτρονικής και της ρομποτικής. Εξερευνώντας το τεχνολογικό υπόβαθρο του μικροελεγκτή, και συγκεκριμένα του μοντέλου Leonardo, μελετήθηκε ο τρόπος με τον οποίο ο μικροελεγκτής σε συνδυασμό με το εξάρτημα Joystick Shield, θα αλληλεπιδράσει με το λειτουργικό σύστημα του Η/Υ και με ένα παράγωγο λογισμικό της μηχανής ανάπτυξης παιχνιδιών Unreal Engine 4.

Στο ίδιο μέρος της εργασίας, αναλύθηκε εις βάθος ο τρόπος ανάπτυξης ενός παιχνιδιού μέσα από τη μηχανή Unreal Engine 4, όπως και η βασική μέθοδος προγραμματισμού της μέσα από μεταβλητές, συναρτήσεις, εργαλεία, κ.α.

Στο 2ο μέρος της εργασίας, παρουσιάζεται η διαδικασία κατασκευής ενός Gamepad, με τη χρήση του Arduino Leonardo, του Joystick Shield και του απαραίτητου προγραμματισμού και των δύο. Αφού δοκιμάστηκε κάτω από διαφορετικές συνθήκες και κατέστη δυνατόν προς λειτουργία, ξεκίνησε η δημιουργία παιχνιδιού μέσα από τη μηχανή Unreal Engine 4.

Κατά την κατασκευή του παιχνιδιού, δοκιμάστηκαν και εκτελέστηκαν πολλές μέθοδοι ανάπτυξης και σχεδιασμού του, ενώ ο πυρήνας του προγράμματος όπως και όλες οι λειτουργικότητες αυτού, βασίστηκαν στον αντικειμενοστραφή προγραμματισμό με Blueprints.

Η τελική φάση της εργασίας, υπήρξε και αυτή της αλληλεπίδρασης μεταξύ του Gamepad και του παιχνιδιού που παράχθηκε από τη μηχανή Unreal Engine 4. Μέσα από αυτή τη φάση, καθορίστηκαν όλες οι τελικές τροποποιήσεις που έγιναν σε αυτές τις δύο τεχνολογίες, ενώ καταγράφηκαν όλα τα σχετικά συμπεράσματα και οι βελτιώσεις που μπόρεσαν να γίνουν.

Είναι σημαντικό να αναφερθεί, πως ένα πολύ μεγάλο ποσοστό επιδιόρθωσης σφαλμάτων που προέκυψαν, έγινε χάρη στην πληθώρα ανθρώπων που απαρτίζουν τις κοινότητες του Arduino και της Unreal Engine 4, με αποτέλεσμα να μειωθεί αισθητά ο χρόνος υλοποίησης του συνολικού έργου. Αυτό συνέβαλε στην αξιοποίηση του κερδισμένου πλέον χρόνου, για τις δοκιμές και την επέκταση του συνολικού έργου ώστε να επιτευχθεί η καλύτερη δυνατή αίσθηση ρεαλισμού προς τον χρήστη.

ΚΕΦΑΛΑΙΟ 1ο : ΘΕΩΡΗΤΙΚΟ ΜΕΡΟΣ

1.1. Εισαγωγή:

Στον κόσμο των ηλεκτρονικών παιχνιδιών, παρουσιάζεται καθημερινά η ανάγκη για εξέλιξη στο χώρο των τεχνολογιών που τα δημιουργεί όπως και για ανάπτυξη του ρεαλισμού, που αυτά προσφέρουν στους χρήστες.

Με την υλοποίησή τους να απαιτεί μεγάλη αφοσίωση από το δημιουργό, ιδιαίτερα κατά τη συγγραφή του κώδικα, οι μηχανές ανάπτυξης παιχνιδιών ελαχιστοποιούν το χρόνο ανάπτυξης κατακόρυφα. Η δημοφιλέστερη από αυτές, είναι η μηχανή ανάπτυξης παιχνιδιών Unreal Engine 4, η οποία και χρησιμοποιήθηκε στην υλοποίηση του παρόντος παιχνιδιού.

Παράλληλα, με την πληθώρα νέων ηλεκτρονικών παιχνιδιών έχουν εξελιχθεί αισθητά οι συσκευές χειρισμού των χαρακτήρων των παιχνιδιών, αλλά και τα εξαρτήματα ρομποτικής που χρησιμοποιούνται στον πραγματικό κόσμο, αλλά και στην επαυξημένη ή εικονική πραγματικότητα.

Η δημιουργία χειριστηρίου από τον μικροελεγκτή Arduino, είναι κάτι δημοφιλές το οποίο εξυπηρετεί πλήθος χρηστών που επιθυμούν να ελέγξουν ένα ρομπότ ή ένα χαρακτήρα παιχνιδιού. Ο τρόπος με τον οποίο θα κατασκευαστεί φυσικά, είναι προϊόν μελέτης και προγραμματισμού του μικροελεγκτή αλλά και ελέγχου αλληλεπίδρασης με το εκάστοτε παιχνίδι.

Βέβαια, το σημαντικότερο ίσως κομμάτι ενός τέτοιου έργου, είναι η επικοινωνία του δημιουργού, με ομάδες ανοιχτού κώδικα λογισμικού, οι οποίες ενισχύουν το αίσθημα της ομαδικότητας αλλά και την αυτοπεποίθηση του ίδιου του δημιουργού μέσα από υπάρχοντα έργα, ερωτήσεις, προβληματισμούς και επεξηγήσεις.

1.2. Ιστορική Αναδρομή:

1.2.1. Arduino:

Το Arduino παρουσιάστηκε για πρώτη φορά στις αρχές του 2000, ως ερευνητικό έργο στο Ινστιτούτο Σχεδιασμού Αλληλεπίδρασης της Ivrea (Interaction Design Institute of Ivrea) με δημιουργούς του, τους Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino και David Mellis. Το ίδιο το έργο βασίζεται στο "Processing project", μια γλώσσα προγραμματισμού για την εκμάθηση της χρήσης κώδικα, στο πλαίσιο των εικαστικών τεχνών (visual arts), που ανέπτυξαν οι Casey Reas & Ben Fry. Επιπροσθέτως, το Arduino, καθώς και το θεωρητικό του υπόβαθρο μέσω του έργου των ανωτέρων, συντέλεσε στην ολοκλήρωση της διατριβής του Hernando Barragan για την "Καλωδίωση Πλακέτας".

Η πρώτη πλακέτα Arduino έκανε την εμφάνισή της το 2005, ώστε να βοηθήσει σπουδαστές σχεδιασμού ψηφιακών μοντέλων, οι οποίοι δεν είχαν εμπειρία στην Ηλεκτρονική ή στον προγραμματισμό μικροελεγκτων. Σκοπός του έργου ήταν μέσω της πλακέτας Arduino, να δημιουργήσουν πρότυπα εργασίας συνδέοντας τον πραγματικό με τον ψηφιακό κόσμο. Η επίσημη εμφάνιση του Arduino στην αγορά, έγινε το 2006 με το Arduino Mini, ενώ από τότε έχει γίνει το πιο δημοφιλές πρότυπο ηλεκτρονικό εργαλείο, το οποίο χρησιμοποιείται από ένα τεράστιο πλήθος μηχανικών, καθώς και εταιρειών ανά τον κόσμο.

Χάρις στη χρησιμότητά του σε ένα τόσο μεγάλο φάσμα εργασιών, το Arduino αποτέλεσε το πρώτο έργο Ανοιχτού Κώδικα Υλικού (Open Source Hardware project), το οποίο έγινε παγκοσμίως γνωστό. Αυτό φυσικά είχε ως αποτέλεσμα, τη δημιουργία μιας παγκόσμιας κοινότητας η οποία εργάζεται εθελοντικά για τη βελτίωση και τους τρόπους χρήσης του Arduino, έτσι ώστε να επωφεληθούν από αυτό ακόμη περισσότεροι άνθρωποι. Είναι επίσης σημαντικό να αναφερθεί πως για την περαιτέρω ανάπτυξή του, βοήθησαν εθελοντικά άνθρωποι από όλο τον κόσμο, εντοπίζοντας και επιλύοντας προβλήματα στον υπάρχοντα κώδικα, αναπτύσσοντας υλικό για την εκμάθηση του

Arduino σε νέους ανθρώπους καθώς και υποστηρίζοντας το σύνολο που ανήκε σε αυτήν την κοινότητα.

Σήμερα το Arduino, αριθμεί χιλιάδες χρήστες στην κοινότητά του, οποίοι συνεχίζουν το έργο της ιδρυτικής ομάδας και αναπτύσσουν συνεχώς νέο λογισμικό για αυτό, όπως και παραδείγματα, χρησιμοποιώντας πλέον όλους τους διαφορετικούς μικροελεγκτές που έχουν δημιουργηθεί μετά το Arduino Mini του 2006. Η ειλικρίνεια και η ευκολία στη χρήση του συνολικού έργου, οδήγησε στη υιοθέτησή του, από έργα βασισμένα στους μικροελεγκτές, ενώ υπήρξε ο καταλύτης για τη δημιουργία του κινήματος "Maker Movement".

Το Arduino πλέον, είναι η πρωταρχική επιλογή για τους δημιουργούς ηλεκτρονικών, με ιδιαίτερη χρήση και στόχευση στην αγορά του IoT (Internet of Things – Ιντερνετ των Πραγμάτων). Η ίδια η πλακέτα έχει εξελιχθεί και στην αγορά υπάρχει μια μεγάλη σειρά από απογόνους του πρώτου Arduino. Παράλληλα, έχουν δημιουργηθεί και περιφερειακά εργαλεία τα οποία μετατρέπουν τους μικροελεγκτές Arduino σε κάτι διαφορετικό όπως αισθητήρες κίνησης ή ήχου, πομπούς και δέκτες συχνοτήτων, μικροελεγκτές ασύρματων ή ενσύρματων συσκευών με τη χρήση σειριακού ή ψηφιακού σήματος, κ.α.

1.2.2. Unreal Engine 4:

Η "Unreal Engine" είναι μια μηχανή δημιουργίας παιχνιδιών η οποία αναπτύχθηκε από την Epic Games και έκανε για πρώτη φορά την εμφάνισή της το 1998, με το first-person shooting παιχνίδι "Unreal". Παρόλο που ο πρωταρχικός της σχεδιασμός ήταν εμπνευσμένος ώστε να εξυπηρετεί first-person shooting παιχνίδια, χρησιμοποιήθηκε επιτυχώς στο σχεδιασμό και άλλου τύπου παιχνιδιών, όπως παιχνίδια ρόλων, παιχνίδια για πολλούς παίκτες (ταυτόχρονα), κ.α.

Τα παιχνίδια που δημιουργούνται από την "Unreal Engine", καθώς και η ίδια η μηχανή, είναι γραμμένα στη γλώσσα προγραμματισμού C++, κάτι το οποίο εξυπηρετεί

τη φορητότητα και την συμβατότητα του παιχνιδιού σε πολλές πλατφόρμες ή και διαφορετικές εκδόσεις της ίδιας πλατφόρμας.

Η πρώτη έκδοση της "Unreal Engine", αναπτύχθηκε από τον ιδρυτή της Epic Games, Tim Sweeney, ο οποίος απέδωσε την έμπνευσή του στα παιχνίδια "Doom" & "Quake", και στον πρωτοποριακό τρόπο με τον οποίο τα προγραμμάτισε ο δημιουργός τους John Carmack. Πιο συγκεκριμένα, το 1995 ξεκίνησε η ανάπτυξη της ίδιας της μηχανής ώστε να δημιουργηθεί το παιχνίδι "Unreal".

Βασικό χαρακτηριστικό της "Unreal Engine", ήταν ο τρόπος με τον οποίο γινόταν η ανίχνευση συγκρούσεων (collision detection) μεταξύ όλων των στοιχείων του παιχνιδιού, η χρήση έγχρωμου φωτισμού (colored lighting) στο ψηφιακό περιβάλλον, καθώς και η στοιχειώδης έκδοση του φιλτραρίσματος υφής (texture filtering) των ψηφιακών επιφανειών του παιχνιδιού. Επιπροσθέτως, υπήρξε η δυνατότητα ταυτόχρονης δημιουργίας και επεξεργασίας επιπέδων, κατι το οποίο εξυπηρέτησε στην ανάπτυξη μιας πληθώρας διαφορετικών επιπέδων μέσα στο παιχνίδι, τα οποία είχαν κοινά χαρακτηριστικά, διαφορετικά δημιουργικά κομμάτια, αλλά συνδέονταν με τις βασικές συναρτήσεις του κορμού, εξαλείφοντας έτσι ολοένα και περισσότερα σφάλματα.

Η δεύτερη έκδοση της μηχανής, "Unreal Engine 2", έκανε το ντεμπούτο της το 2002, με το δωρεάν παιχνίδι "America's Army", το οποίο αναπτύχθηκε από τον Αμερικανικό Στρατό ώς μέσω στρατολόγησης. Αμέσως οι διαφορές στα νέα χαρακτηριστικά της μηχανής, ειδικά στο κομμάτι των γραφικών, κέρδισαν το κοινό με αποτέλεσμα μέχρι τα τέλη του 2004, η μηχανή να υποστηρίζει την ανάπτυξη παιχνιδιών για την παιχνιδομηχανή "Xbox".

Το 2004, η Epic Games, εξέδωσε φωτογραφίες της τρίτης έκδοσης της μηχανής, "Unreal Engine 3", έχοντας ήδη αναπτύξει το έργο για 18 μήνες. Αντίθετα από την "Unreal Engine 2", η τρίτη έκδοση έκανε την επανάσταση στον χώρο τον γραφικών, καθώς πλέον η δημιουργία φωτισμού όπως και οι υπολογισμοί για αυτούς, γινόντουσαν ανά pixel και όχι ανά vertex. Μέχρι τότε, η "Unreal Engine 3" υποστήριζε ήδη αρκετές πλατφόρμες όπως το PlayStation 3 και το Xbox 360, ενώ τα iOS και Android εντάχθηκαν

μέχρι και το 2010. Έως το 2013, η "Unreal Engine 3" κατάφερε να σχεδιάζει παιχνίδια τα οποία υποστήριζαν το Adobe Flash Player 11, και να τα αναπτύσσει για το Wii U, τα Windows 8 ή την κοινότητα του Steam. Συμπληρωματικά, μέσω της γλώσσας προγραμματισμού ιντερνετ "HTML5", εκδόθηκαν παιχνίδια τα οποία αφορούσαν τον φυλλομετρητή Mozilla.

Μέχρι εκείνη τη χρονική στιγμή, ήταν αδύνατον για το κοινό να χρησιμοποιήσει την μηχανή για την ανάπτυξη ενός παιχνιδιού, καθώς αντίβαινε στα πνευματικά δικαιώματα που διατηρούσε πάνω της η Epic Games. Παρόλα αυτά, το 2009, δημιουργήθηκε και δημοσιοποιήθηκε το δωρεάν SDK της "Unreal Engine 3", το οποίο ονομάστηκε "Unreal Development Kit" και έδινε τη δυνατότητα στον κόσμο, να παράγει δικά του versions για υπάρχοντα παιχνίδια που έχουν σχεδιαστεί μέσω της Unreal Engine 3, ή να δημιουργήσει εξ ολοκλήρου ένα παιχνίδι είτε για ακαδημαϊκούς, είτε για εμπορικούς σκοπούς. Το Δεκέμβρη του 2010, το Unreal Development Kit, υποστήριζε τη δημιουργία παιχνιδιών και εφαρμογών τόσο για όλες τις παραπάνω κονσόλες και περιβάλλοντα, όσο και για το λειτουργικό iOS.

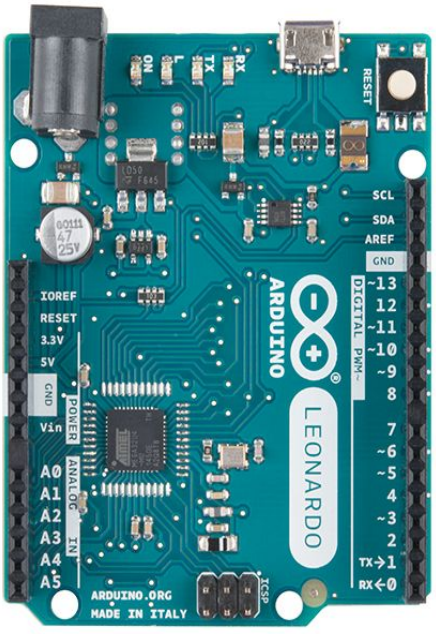
Τα επόμενα χρόνια υπήρξαν άκρως παραγωγικά για την Epic Games, η οποία ανακοίνωσε ότι η ανάπτυξη της τέταρτης έκδοσης της μηχανής, ετοιμαζόταν από το 2003, με σκοπό να παρουσιαστεί σε περιορισμένο κοινό το 2012, όπως και έγινε. Στα μέσα του 2012, η "Unreal Engine 4" , παρουσιάστηκε στο ετήσιο συνέδριο "Game Developers Conference", όπου και εξέπληξε τους πάντες με τις νέες της δυνατότητες αλλά και με τους έξυπνους πλέον αλγορίθμους της που αντικατέστησαν τους χρόνους εκτέλεσης υπολογισμών των μέχρι τότε συναρτήσεων, κατά την κατασκευή ενός έργου. Τον Μάρτιο του 2014, η Unreal Engine 4 παραδόθηκε σε σχολεία και πανεπιστήμια χωρίς αντίτιμο, ώστε μαθητές και σπουδαστές να μπορέσουν να εργαστούν πάνω στην ανάπτυξη ηλεκτρονικών παιχνιδιών, να εντρυφήσουν στην επιστήμη της πληροφορικής, στις τέχνες, στην αρχιτεκτονική μέσω ενός ψηφιακού μέσου καθώς και στην προσομοίωση του πραγματικού κόσμου μέσω προγραμμάτων. Τέλος, ένα χρόνο μετά, η Unreal Engine 4 διατέθηκε στο κοινό δωρεάν, μαζί με όλες τις απαραίτητες ενημερώσεις, ενώ παράλληλα τα δικαιώματα του εκάστοτε παιχνιδιού που ανέπτυσσε κάποιος, ανήκουν στον ίδιο.

Από το 1996 μέχρι και σήμερα, η Unreal Engine χρησιμοποιείται για την κατασκευή, ανάπτυξη και παράδοση παιχνιδιών σε σχεδόν κάθε τύπου λειτουργικό πρόγραμμα ή πλατφόρμα όπως τα Microsoft Windows, τα MacOS, το SteamOS, το FreeBSD, η HTML5, τα iOS & Android, το Nintendo Switch, το PlayStation 4, το Xbox One, το Magic Leap One και τα περιβάλλοντα εικονικής πραγματικότητας SteamVR/HTC Vice, Oculus Rift, PlayStation VR, Google Daydream, OSVR και Samsung Gear VR.

1.3. Επί μέρους ανάλυση του Arduino και του Joystick Shield:

1.3.1. Arduino Leonardo:

Το Arduino Leonardo, είναι ένας μικροελεγκτής βασισμένος στον επεξεργαστή ATmega32u4. Έχει 20 ψηφιακούς ακροδέκτες εισόδου/εξόδου, εκ των οποίων οι 7 μπορούν να χρησιμοποιηθούν ως είσοδοι τύπου PWM και οι 12 ως αναλογικοί είσοδοι. Διαθέτει κρυσταλλικό ταλαντωτή 16 MHz, σύνδεση τύπου micro USB, βύσμα παροχής ηλεκτρικού ρεύματος, κεφαλίδα ICSP και πλήκτρο reset. Εμπεριέχει οτιδήποτε χρειάζεται για την υποστήριξη του μικροελεγκτή ενώ απλώς απαιτεί τη σύνδεσή του με τον Η/Υ ή την παροχή ρεύματος από σταθερή ή φορητή πηγή.

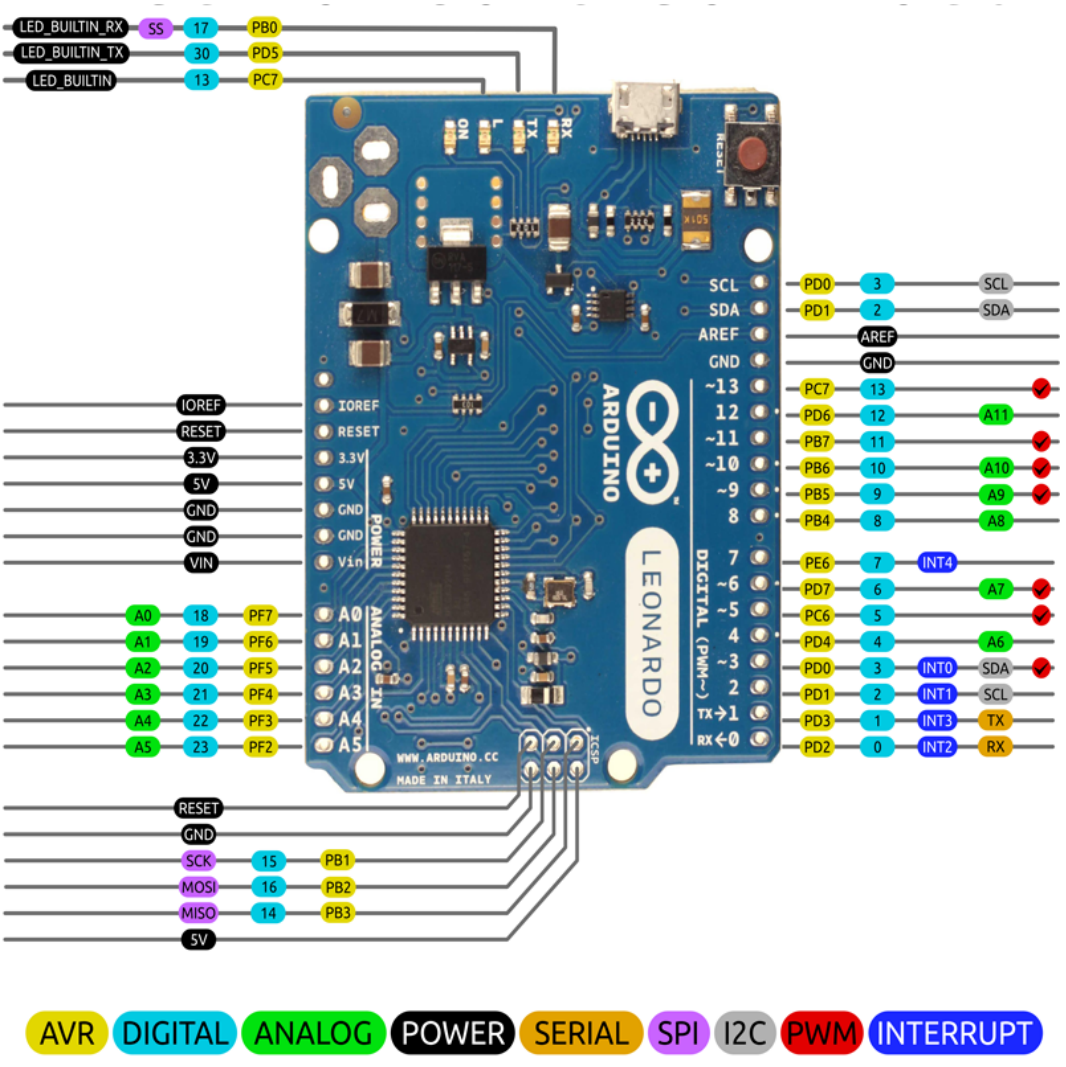


Εικ. 1.1: Ο μικροελεγκτής Arduino Leonardo

Το Leonardo, διαφέρει από όλες τις προηγούμενες πλακέτες, καθώς ο ATmega32u4 επεξεργαστής έχει ενσωματωμένη την επικοινωνία μέσω USB, εξαλείφοντας έτσι, την ανάγκη για δεύτερο επεξεργαστή. Αυτή η κατάσταση επιτρέπει στο Leonardo, να εμφανίζεται στον συνδεδεμένο Η/Υ, ως ποντίκι ή πληκτρολόγιο, μέσω μιας εικονικής (CDC) σειριακής / COM θύρας.

1.3.2. Δομή και ανάλυση:

Είναι σημαντικό, κατά τη χρήση ενός μικροελεγκτή Arduino, να λαμβάνονται υπόψιν όλα τα τεχνικά χαρακτηριστικά του για την αποφυγή σφαλμάτων αλλά και για τη μελέτη σχετικά με το έργο, για το ποιος μικροελεγκτής είναι κατάλληλος προς χρήση. Συνοπτικά οι τομείς που πρέπει να εξετάζονται ανά μικροελεγκτή είναι η ενέργεια ή η ισχύς του, η μνήμη του, η είσοδος και η έξοδος της πλακέτας, η επικοινωνία του, ο προγραμματισμός του, τα φυσικά χαρακτηριστικά του, κ.α.



Εικ. 1.2: Σχηματική απεικόνιση όλων των ακροδεκτών του μικροελεγκτή Arduino Leonardo

Πιο συγκεκριμένα, τα βασικά τεχνικά χαρακτηριστικά του Arduino Leonardo είναι:

Microcontroller	**ATmega32u4**
Operating Voltage	5V
Input Voltage (Προτεινόμενη)	7-12V
Input Voltage (όρια)	6-20V
Ψηφιακοί Ακροδέκτες I/O	20
Κανάλια PWM	7
Κανάλια Analog Input	12
DC Current για I/O ακροδέκτη	40 mA
DC Current για 3.3V ακροδέκτη	50 mA
Flash Memory	32 KB (ATmega32u4) εκ των οποίων τα 4 KB χρησιμοποιούνται από το bootloader
SRAM	2.5 KB (ATmega32u4)
EEPROM	1 KB (ATmega32u4)
Clock Speed	16 MHz
Μήκος	68.6 mm
Πλάτος	53.3 mm
Βάρος	20 g

Πίνακας 1.1: Τεχνικά Χαρακτηριστικά του μικροελεγκτή Arduino Leonardo

Αναλυτικότερα, οι τομείς που πρέπει να εξεταστούν για το Arduino Leonardo είναι οι εξής:

Ενέργεια:

Η παροχή ρεύματος στο Arduino Leonardo γίνεται μέσω της micro USB σύνδεσης και της πηγής ρεύματος ή κάποιας εξωτερικής πηγής. Επίσης, υπάρχει η δυνατότητα παροχής ρεύματος (χωρίς τη χρήση του micro USB) είτε από αντάπτορα τύπου AC σε DC είτε από μπαταρία. Ο αντάπτορας μπορεί να συνδεθεί απευθείας στην πλακέτα χρησιμοποιώντας ένα αρσενικού τύπου καλώδιο 2.1mm. Εάν συνδεθεί το Arduino με μπαταρία, τότε δύο καλώδια συνδέονται από τους πόλους της μπαταρίας με τη γείωση (Gnd) και τον Vin ακροδέκτη του συνδέσμου της παροχής ρεύματος της πλακέτας. Οι ακροδέκτες της παροχής ρεύματος είναι οι εξής:

Vin: Αντιστοιχεί στην είσοδο του ηλεκτρικού ρεύματος στην πλακέτα Arduino όταν χρησιμοποιείται εξωτερική πηγή (σε αντίθεση με τα 5V που παρέχονται από τη σύνδεση με USB ή με άλλες ρυθμιζόμενες πηγές ίδιου τύπου).

5V: Αντιστοιχεί στη ρυθμιζόμενη πηγή ρεύματος που χρησιμοποιείται για τη χρήση του μικροελεγκτή και άλλων εξαρτημάτων επάνω στην πλακέτα. Αυτή η πηγή ρεύματος είτε είναι σε ακολουθία με το Vin μέσω ενός ενσωματωμένου ρυθμιστή στην πλακέτα, είτε παρέχεται μέσω USB ή άλλης ρυθμιζόμενης παροχής τύπου 5V.

3V3: Η παροχή τύπου 3.3V, παράγεται από τον ρυθμιστή που είναι ενσωματωμένος στην πλακέτα.

GND: Ακροδέκτες γείωσης.

IOREF: Η τάση κατά την οποία λειτουργούν οι ακροδέκτες εισόδου/εξόδου της πλακέτας. Για το Leonardo η τάση αυτή είναι 5V.

Μνήμη:

Η μνήμη του επεξεργαστή ATmega32u4 είναι 32 KB (με τα 4 KB να χρησιμοποιούνται κατά την ενεργοποίηση του μικροελεγκτή). Επίσης διαθέτει 2.5 KB SRAM και 1 KB EEPROM (η οποία χρησιμοποιείται με την ομώνυμη βιβλιοθήκη).

Είσοδος/Εξοδος:

Καθένας από τους 20 ψηφιακούς ακροδέκτες εισόδου/εξόδου του Leonardo μπορεί να χρησιμοποιηθεί ως είσοδος ή έξοδος μέσω των συναρτήσεων pinMode(), digitalWrite() και digitalRead(). Η τάση κατά την οποία λειτουργούν είναι 5V. Κάθε ακροδέκτης μπορεί να λάβει ή να δώσει το μέγιστο 40mA, ενώ περιέχει αντίσταση άνω έλξης με τιμές 20 έως 50 kOhms (η οποία είναι αποσυνδεδεμένη εξ αρχής). Επιπλέον, κάποιοι ακροδέκτες έχουν ξεχωριστές λειτουργικότητες, όπως:

Σειριακοί ακροδέκτες: 0(RX) και 1(TX). Χρησιμοποιούνται ως δέκτες (RX) και ως πομποί (TX) μεταξύ TTL (transistor-transistor logix) σειριακών δεδομένων, μέσω των δυνατοτήτων του ATmega32U4 επεξεργαστή. Στο Leonardo η κλάση Serial αναφέρεται σε αυτή την επικοινωνία μέσω USB, ενώ για το μοντέλο TTL χρησιμοποιείται η κλάση Serial1.

Ακροδέκτες TWI: 2 τύπου SDA και 3 τύπου SCL, υποστηρίζουν την επικοινωνία TWI με τη χρήση της βιβλιοθήκης Wire.

Ακροδέκτες εξωτερικών διακοπών: 3 (διακοπή 0), 2 (διακοπή 1), 0 (διακοπή 2), 1 (διακοπή 3) και 7 (διακοπή 4). Αυτοί οι ακροδέκτες μπορούν να ρυθμιστούν ώστε να ενεργοποιούν τη διακοπή σε χαμηλές τιμές τάσης, την άνοδο όταν εντοπίζεται πτώση της τάσης ή να αλλάζουν γενικά τιμή, με τη χρήση της συνάρτησης attachInterrupt(). Ακροδέκτες PWM: Είναι οι 3, 5, 6, 9, 10, 11 και 13. Διαθέτουν έξοδο 8-bit, τύπου PWM, με τη χρήση της συνάρτησης analogWrite().

Ακροδέκτες SPI: Βρίσκονται πάνω στην ICSP κεφαλή. Αυτοί οι ακροδέκτες υποστηρίζουν την επικοινωνία τύπου SPI με τη χρήση της βιβλιοθήκης SPI. Δεν είναι

συνδεδεμένοι σε κανένα ψηφιακό ακροδέκτη, σε αντίθεση με το Arduino Uno και είναι διαθέσιμοι μόνο στον σύνδεσμο ICSP. Αυτό σημαίνει ότι αν εφαρμόζεται στο Leonardo κάποιο Shield το οποίο χρησιμοποιεί το SPI, αλλά δεν διαθέτει σύνδεσμο ICSP 6-ακροδεκτών, τότε το Shield δεν θα λειτουργήσει.

Ακροδέκτης LED: Επάνω στην πλακέτα Leonardo, βρίσκεται ένα LED, συνδεδεμένο στον ψηφιακό ακροδέκτη 13. Όταν η τιμή του ακροδέκτη είναι HIGH το LED είναι ενεργοποιημένο, ενώ όταν η τιμή του είναι LOW είναι απενεργοποιημένο.

Αναλογικές Είσοδοι: A0-A5, A6 - A11 (επάνω στους ψηφιακούς ακροδέκτες 4, 6, 8, 9, 10 και 12). Το Leonardo έχει 12 αναλογικές εισόδους, από το A0 έως το A11, οι οποίες μπορούν να χρησιμοποιηθούν και ως ψηφιακές είσοδοι/έξοδοι. Οι ακροδέκτες A0 έως A5 εμφανίζονται στην ίδια θέση με αυτούς στην πλακέτα Uno. Οι ακροδέκτες A6 έως A11 είναι στις θέσεις των ακροδεκτών των ψηφιακών εισόδων/εξόδων 4, 6, 8, 9, 10 και 12 αντιστοίχως. Κάθε αναλογική είσοδος προσδίδει ανάλυση των 10 bits (π.χ. 1024 διαφορετικές τιμές). Εξ ορισμού, οι τιμές σε Volts, που μετράνε οι αναλογικές είσοδοι είναι από τη γείωση έως και τα 5V. Παρόλα αυτά είναι δυνατόν να αλλάξει το μέγιστο όριο τους, με τη χρήση του ακροδέκτη AREF και της συνάρτησης analogReference().

Ακροδέκτης AREF: Αναφέρεται στην τάση των αναλογικών εισόδων. Χρησιμοποιείται με τη συνάρτηση analogReference().

Reset (επαναφορά): Επαναφέρει τον μικροελεγκτή στις αρχικές του ρυθμίσεις. Συνήθως χρησιμοποιείται στις περιπτώσεις που συνδυαστικά με το Leonardo υπάρχει και κάποιο Shield, όπως στην περίπτωση της παρούσας εργασίας.

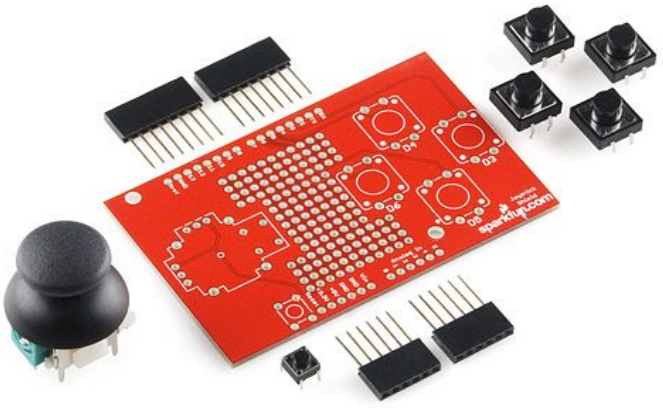
Επικοινωνία:

Το Leonardo έχει έναν αριθμό προεγκατεστημένων υπηρεσιών για την επικοινωνία με τον Η/Υ, με κάποιο άλλο Arduino και γενικά με άλλους μικροελεγκτές. Ο επεξεργαστής ATmega32U4, παρέχει σειριακή επικοινωνία τύπου UART TTL (5V) η οποία είναι διαθέσιμη στους ψηφιακούς ακροδέκτες 0 (RX) και 1 (TX). Ακόμη, ο επεξεργαστής επιτρέπει την σειριακή επικοινωνία μέσω USB και εμφανίζεται μέσω εικονικής θύρας,

στα προγράμματα του υπολογιστή. Το chip επίσης, συμπεριφέρεται ώς συσκευή USB 2.0 τόσο σε ταχύτητα όσο και σε επιδόσεις, χρησιμοποιώντας τους κοινότυπους USB COM οδηγούς προγράμματος. Το λογισμικό του Arduino εμπεριέχει ένα σειριακό μόνιτορ το οποίο επιτρέπει την αποστολή απλού κειμένου από και προς αυτό. Τα LED τύπου RX & TX της πλακέτας, ανταποκρίνονται κατά τη μετάδοση δεδομένων μέσω σύνδεσης USB (αυτό δεν ισχύει για σειριακή επικοινωνία στους ακροδέκτες 0 και 1). Η βιβλιοθήκη SoftwareSerial, είναι αυτή που επιτρέπει τη σειριακή επικοινωνία μεταξύ όλων των ψηφιακών ακροδεκτών του Leonardo. Επίσης ο ATmega32U4, υποστηρίζει τις επικοινωνίες I2C (TWI) και SPI. Το λογισμικό του Arduino εμπεριέχει τη βιβλιοθήκη Wire η οποία απλουστεύει τη χρήση του διαύλου I2C ενώ για την SPI επικοινωνία, χρησιμοποιείται η βιβλιοθήκη SPI. Το Leonardo ανταποκρίνεται ως πληκτρολόγιο ή ποντίκι ενώ μπορεί να προγραμματιστεί κατάλληλα ώστε να χειρίζεται τέτοιες συσκευές εισόδου χρησιμοποιώντας τις κλάσεις Keyboard και Mouse.

1.3.3. Joystick Shield:

Το Arduino Joystick Shield, είναι μια πλακέτα τεσσάρων ψηφιακών πλήκτρων και ενός αναλογικού Joystick, τα οποία μπορούν τα μετατρέψουν το εκάστοτε Arduino σε χειριστήριο. Τα έργα στα οποία μπορεί να χρησιμοποιηθεί το Joystick Shield αφορούν τόσο τον ψηφιακό, όσο και τον αναλογικό κόσμο. Στην παρούσα εργασία, το Joystick Shield χρησιμοποιήθηκε για τον χειρισμό ενός ψηφιακού χαρακτήρα της Unreal Engine 4 και για την επίτευξη του σκοπού χρειάστηκε να γίνει η συνδεσμολογία του από την αρχή, ο έλεγχος ορθής λειτουργίας του, σε συνδυασμό με το Arduino Leonardo, καθώς και ο προγραμματισμός του για το χειρισμό πληθώρας παιχνιδιών, έως ότου χρησιμοποιηθεί αποκλειστικά σε παιχνίδι ανεπτυγμένο από την Unreal Engine 4.



Εικ. 1.3: Η πλακέτα Arduino Joystick Shield Kit

Συγκεκριμένα, η Sparkfun, η εταιρεία που κατασκευάζει το Joystick Shield, σε συνεργασία με το Arduino, δίνει όλες τις απαραίτητες πληροφορίες για τη σωστή συνδεσμολογία του Joystick με τα πλήκτρα και με την εκάστοτε πλακέτα Arduino.

Το πακέτο του Arduino Joystick Shield περιλαμβάνει τα παρακάτω:

- ➔ 1 Joystick Shield πλακέτα, σχεδιασμένη με το πρότυπο λογισμικό δημιουργίας κυκλωμάτων PCB.
- ➔ 4 πλήκτρα 12mm.
- ➔ 1 Joystick.
- ➔ 1 μικρό πλήκτρο στην πλακέτα για τη λειτουργία του Reset.
- ➔ 2 κεφαλές για τους 6 ακροδέκτες του Arduino.
- ➔ 2 κεφαλές για τους 8 ακροδέκτες του Arduino.

1.3.4. Serial & Digital Read, Write, Print:

Ένα βασικό κομμάτι του προγραμματισμού ενός μικροελεγκτή Arduino είναι η χρήση σειριακών ή ψηφιακών εντολών που αφορούν το κομμάτι της ανάγνωσης, της εγγραφής και της εκτύπωσης. Φυσικά υπάρχουν περισσότερες από τρεις συναρτήσεις χρήσης της πλακέτας στο σειριακό ή στο ψηφιακό επίπεδο, αλλά χρησιμοποιούνται σε πιο εξειδικευμένες περιπτώσεις προγραμματισμού.

Σειριακή ανάγνωση - Serial.read():

Η σειριακή ανάγνωση ή serial.read(), είναι η συνάρτηση που διαβάζει τα εισερχόμενα σειριακά δεδομένα, τα οποία λαμβάνει από την κλάση ροής του προγράμματος. Σε αντίθεση με πολλές συναρτήσεις, η serial.read() δεν χρησιμοποιεί κάποια παράμετρο για την ανάγνωση των δεδομένων, ενώ επιστρέφει το πρώτο byte της τιμής που διαβάζει. Στην περίπτωση που δεν υπάρχουν δεδομένα προς ανάγνωση, τότε επιστρέφει την τιμή -1.

Παράδειγμα χρήσης της serial.read():

```
// for incoming serial data
int incomingByte = 0;

void setup() {
// opens serial port, sets data rate to 9600 bps
  Serial.begin(9600);
}

void loop() {
// send data only when you receive data:
  if (Serial.available() > 0) {
  // read the incoming byte:
    incomingByte = Serial.read();

  // say what you got:
    Serial.print("I received: ");
    Serial.println(incomingByte, DEC);
```

```
  }
}
```

Σειριακή εγγραφή - Serial.write():

Η σειριακή εγγραφή ή serial.write(), είναι η συνάρτηση, η οποία, γράφει δεδομένα σε δυαδική μορφή μέσω της σειριακής θύρας. Αυτά τα δεδομένα, στέλνονται ως ένα byte ή μια σειρά από bytes. Στην περίπτωση που πρέπει να σταλούν οι χαρακτήρες που αντιπροσωπεύουν τα ψηφία ενός αριθμού, είναι προτιμότερη η χρήση της συνάρτησης print(). Οι παράμετροι οι οποίοι μπορεί να λάβει η serial.write() είναι οι εξής:

val: μια τιμή που στέλνεται ως ένα byte
str: ένα αλφαριθμητικό το οποίο στέλνεται ως μια σειρά από bytes
buf: ένας πίνακας ο οποίος στέλνεται ως μια σειρά από bytes
len: το μήκος του buffer (αλφαριθμητικού)

Η τιμή που θα επιστρέψει η serial.write(), είναι ο αριθμός των bytes που εγγράφηκαν. Η ανάγνωση αυτού του αριθμού όμως, είναι προεραιτική.

Παράδειγμα χρήσης της serial.write():

```
void setup(){
  Serial.begin(9600);
}

void loop(){
// send a byte with the value 45
  Serial.write(45);

//send the string "hello" and return the length of the string
   int bytesSent = Serial.write("hello");
}
```

Σειριακή εκτύπωση - serial.print():

Η σειριακή εκτύπωση ή serial.print(), είναι η συνάρτηση η οποία εκτυπώνει δεδομένα μέσω της σειριακής θύρας σε μορφή ASCII (αναγνώσιμη από τον άνθρωπο). Αυτή η συνάρτηση, μπορεί να πάρει πολλές μορφές. Οι ακέραιοι αριθμοί εκτυπώνονται χρησιμοποιώντας χαρακτήρες ASCII για κάθε τους ψηφίο. Οι δεκαδικοί εκτυπώνονται με παρόμοιο τρόπο ως ψηφία ASCII, με την εκτύπωση να τους εμφανίζει με δύο δεκαδικά ψηφία, εξ ορισμού. Τα bytes στέλνονται ως αυτόνομοι χαρακτήρες. Οι χαρακτήρες και τα αλφαριθμητικά στέλνονται με τη μορφή που έχουν.

Οι παρακάτω χρήσεις της serial.print() εξηγούν τον τρόπο με τον οποίο η συνάρτηση αυτή εκτυπώνει τα δεδομένα στην οθόνη του χρήστη:

Παράμετρος	**Έξοδος**
serial.print(78)	78
serial.print(1.23456)	1.23
serial.print('N')	"N"
serial.print("Hello world.")	"Hello world."

Πίνακας 1.2.α: Η έξοδος της συνάρτησης Serial.print()

Μια δευτερεύουσα παράμετρος, ορίζει τη μορφοποίηση που θα χρησιμοποιηθεί. Οι επιτρεπόμενες σε αυτή τιμές είναι οι BIN (δυαδική μορφοποίηση), OCT (οκταδική μορφοποίηση), DEC (δεκαδική μορφοποίηση), HEX (δεκαεξαδική μορφοποίηση). Για τα δεκαδικά μέρη των δεκαδικών αριθμών, η παράμετρος ορίζει τον αριθμό των δεκαδικών ψηφίων που θα εκτυπωθούν. Για παράδειγμα:

Παράμετρος	**Έξοδος**
serial.print(78, BIN)	1001110

serial.print(78, OCT)	116
serial.print(78, DEC)	78
serial.print(78, HEX)	4E
serial.println(1.23456, 0)	1
serial.println(1.23456, 2)	1.23
serial.println(1.23456, 4)	1.2346

Πίνακας 1.2.β: Η έξοδος της συνάρτησης Serial.print()

Οι παράμετροι που λαμβάνει η συνάρτηση serial.print() είναι, η val (η τιμή που θα τυπωθεί ανεξαρτήτως τύπου) και η format (ορίζει τον αριθμό της βάσης για ακέραιους ή δεκαδικούς αριθμούς).

Παράδειγμα χρήσης της serial.print():

```
/*
Uses a FOR loop for data and prints a number in various formats.
*/
// variable
int x = 0;

void setup() {
// open the serial port at 9600 bps
  Serial.begin(9600);
}

void loop() {
  // print labels
  Serial.print("NO FORMAT");
  Serial.print("\t");

  Serial.print("DEC");
  Serial.print("\t");
```

```
  Serial.print("HEX");
  Serial.print("\t");

  Serial.print("OCT");
  Serial.print("\t");

  Serial.print("BIN");
  Serial.print("\t");

  // only part of the ASCII chart, change to suit
  for(x=0; x< 64; x++){

    // print it out in many formats:

    // print as an ASCII-encoded decimal - same as "DEC"
    Serial.print(x);
    // prints a tab
    Serial.print("\t");

    // print as an ASCII-encoded decimal
    Serial.print(x, DEC);
    // prints a tab
    Serial.print("\t");

    // print as an ASCII-encoded hexadecimal
    Serial.print(x, HEX);
    // prints a tab
    Serial.print("\t");

    // print as an ASCII-encoded octal
    Serial.print(x, OCT);
    // prints a tab
    Serial.print("\t");

    // print as an ASCII-encoded binary
    Serial.println(x, BIN);
    // then adds the carriage return with "println"
    delay(200);             // delay 200 milliseconds
  }
  // prints another carriage return
  Serial.println("");
}
```

Ψηφιακή ανάγνωση - digitalRead():

Η συνάρτηση ψηφιακής ανάγνωσης ή digitalRead(), χρησιμοποιείται για να διαβάσει την τιμή από ένα συγκεκριμένο ψηφιακό ακροδέκτη του μικροελεγκτή. Οι τιμές αυτές είναι είτε “HIGH” είτε “LOW”. Επίσης, η digitalRead(), μπορεί να δώσει τυχαία μία από τις τιμές “HIGH” ή “LOW”, στην περίπτωση που ο ψηφιακός ακροδέκτης δεν συνδέεται κάπου. Ακόμη, μπορούν και οι αναλογικοί ακροδέκτες να χρησιμοποιηθούν ως ψηφιακοί, με τη συνάρτηση αυτή.

Παράδειγμα χρήσης της digitalRead():

```
// LED connected to digital pin 13
int ledPin = 13;
// pushbutton connected to digital pin 7
int inPin = 7;
// variable to store the read value
int val = 0;


void setup()
{
  // sets the digital pin 13 as output
  pinMode(ledPin, OUTPUT);
  // sets the digital pin 7 as input
  pinMode(inPin, INPUT);
}

void loop()
{
  // read the input pin
  val = digitalRead(inPin);
  // sets the LED to the button's value
  digitalWrite(ledPin, val);
}
```

Ψηφιακή εγγραφή - digitalWrite():

Η συνάρτηση ψηφιακής εγγραφής μπορεί να εγγράψει την τιμή "HIGH" ή "LOW" σε ένα ψηφιακό ακροδέκτη. Εφόσον ο ακροδέκτης έχει οριστικοποιηθεί ώς έξοδος με την συνάρτηση pinMode(), η τάση του θα είναι της τάξης των 5V (ή 3.3V για 3.3V πλακέτες) για την τιμή "HIGH" και 0V (γείωση) για την τιμή "LOW".

Από τη στιγμή που ο ακροδέκτης έχει οριστικοποιηθεί ώς είσοδος, η συνάρτηση digitalWrite(), θα ενεργοποιήσει (τιμή "HIGH") ή θα απενεργοποιήσει (τιμή "LOW") την εσωτερική έλξη του ακροδέκτη.

Οι παράμετροι τους οποίους λαμβάνει η συνάρτηση digitalWrite() είναι ο αριθμός του ακροδέκτη και οι τιμές "HIGH" και "LOW", ενώ δεν επιστρέφει κάποια τιμή.

Παράδειγμα χρήσης της digitalWrite():

```
void setup(){

// sets the digital pin 13 as output
  pinMode(13, OUTPUT);
}

void loop()
{
  // sets the digital pin 13 on
  digitalWrite(13, HIGH);
  // waits for a second
  delay(1000);
  // sets the digital pin 13 off
  digitalWrite(13, LOW);
  // waits for a second
  delay(1000);
}
```

1.3.5. Sketch & Libraries:

Το Sketch, είναι για τη γλώσσα του Arduino, το πρόγραμμα το οποίο θα εισαχθεί στον μικροελεγκτή και θα εκτελέσει μια σειρά από συναρτήσεις ώστε να εξυπηρετήσει το σκοπό χρήσης του. Με λίγα λόγια είναι το "script", όπως λέγεται για μια άλλη γλώσσα προγραμματισμού, στο οποίο βρίσκεται το λειτουργικό κομμάτι του κώδικα.

Πιο αναλυτικά, κάθε Sketch εμπεριέχει μια σειρά από μεταβλητές, συναρτήσεις και σχόλια, ενώ από αυτό δεν λείπουν οι δύο ειδικές συναρτήσεις που είναι απαραίτητες για την ορθή εκτέλεσή του. Οι συναρτήσεις setup() & loop(), είναι το κέλυφος του sketch, κάτω από το οποίο θα χτιστεί όλο το προγραμματιστικό του περιβάλλον.

Η συνάρτηση setup(), καλείται μία φορά κατά την εκκίνηση του προγράμματος. Μέσα σε αυτή ορίζονται οι ακροδέκτες οι οποίοι θα χρησιμοποιηθούν στο πρόγραμμα και καλούνται όλες οι απαραίτητες βιβλιοθήκες.

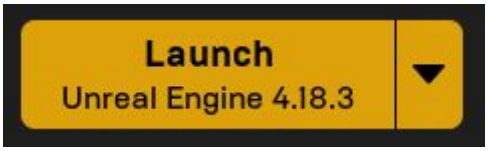
Η συνάρτηση loop() είναι αυτή που καλείται επαναλαμβανόμενα και δίνει ζωή στο πρόγραμμα του μικροελεγκτή. Μέσα σε αυτή, επαναλαμβάνονται όλες οι διαδικασίες που πρέπει να εκτελέσει το Arduino, με στόχο την ορθή λειτουργία του.

Ακόμα, είναι σημαντική η χρήση και των δύο παραπάνω συναρτήσεων ώστε να λειτουργήσει σωστά το sketch, ακόμα και αν δεν χρησιμοποιείται κάποια από τις δύο..

1.4. Επί μέρους ανάλυση της Unreal Engine 4:

1.4.1. Version 4.18:

Στην παρούσα πτυχιακή εργασία, χρησιμοποιήθηκε η έκδοση 4.18.3 της μηχανής ανάπτυξης παιχνιδιών Unreal Engine 4. Στη συγκεκριμένη έκδοση, μπορεί να πραγματοποιηθεί η δημιουργία παιχνιδιών και εφαρμογών στο πλαίσιο της εικονικής ή επαυξημένης πραγματικότητας, με μεγάλη ακρίβεια στην προσθήκη ρεαλιστικού περιεχομένου. Η υποστήριξη της πλατφόρμας της μηχανής, έχει επεκταθεί και βελτιωθεί με αποτέλεσμα να κατασκευάζονται πλέον παιχνίδια πιο ευφυή, σε ένα σταθερό ψηφιακό περιβάλλον χωρίς προβλήματα.



Εικ. 1.4: Εκτέλεση της Unreal Engine 4.18.3

Η Unreal Engine 4.18 διαφέρει από τις προηγούμενες εκδόσεις της μηχανής καθώς εντάσσει στην πλατφόρμα της, ογκομετρικούς χάρτες φωτός και πολλαπλές αναπηδήσεις φωτός από εξωγενείς πομπούς, εντός του ψηφιακού κόσμου. Αυτό έχει ως αποτέλεσμα, την ακριβέστερη προσέγγιση του πραγματικού κόσμου μέσα από τον ψηφιακό, ενώ δίνει στον χρήστη, την αίσθηση του ρεαλισμού και της πιστότητας.

Ακόμη, η έκδοση 4.18, συγχωνεύει μέσω της επαυξημένης πραγματικότητας, το αποτέλεσμα της μηχανής με τον πραγματικό κόσμο, μέσα από το φακό μιας κάμερας.

Ο τρόπος με τον οποίο γίνεται η σύνταξη της εφαρμογής προς την κατασκευή της, ανανεώνεται συνεχώς, ώστε να είναι η μηχανή πιο φιλική σε κάθε είδους λειτουργικό σύστημα και πλατφόρμα.

1.4.2. “ Blueprints ή C++ ; ” :

Η Unreal Engine 4, παρέχει δύο μεθόδους προγραμματισμού ενός παιχνιδιού. Αυτές είναι η γλώσσα προγραμματισμού C++ και το Blueprints Visual Scripting.

Η χρήση της C++ σε ένα παιχνίδι, απαιτεί μεγάλη εξοικείωση με τη γλώσσα, άριστη κατανόησή της, ενώ μπορεί να δημιουργήσει large-scale παιχνίδια, δηλαδή παιχνίδια των οποίων ο ψηφιακός κόσμος και οι δυνατότητές του επεκτείνονται χωρίς όρια.

Η χρήση του Blueprints Visual Scripting, δεν απαιτεί την ίδια εξοικείωση με τον προγραμματισμό, παρόλα αυτά απαιτεί άριστη προγραμματιστική λογική καθώς τα blueprints χρησιμοποιούν εξίσου παρόμοιες μεταβλητές, συναρτήσεις, λογικούς τελεστές κλπ. Ακόμη, χρησιμοποιούνται περισσότερο για την γρήγορη εκμάθηση της μηχανής ή για την κατασκευή ενός low-scale παιχνιδιου. Συνηθίζεται τα low-scale παιχνίδια να μην επεκτείνουν το ψηφιακό τους περιβάλλον και να μην έχουν τόσο μεγάλες απαιτήσεις συστήματος όπως τα large-scale.

Είναι σημαντικό να αναφερθεί, πως για την δημιουργία των Blueprints, χρησιμοποιείται η C++, και πως αυτά δεν αποτελούν μια γλώσσα προγραμματισμού, αλλά κάποια έτοιμα προσχέδια για την απευθείας χρήση της γλώσσας.

Στην παρούσα εργασία, χρησιμοποιήθηκε η μέθοδος Blueprints Visual Scripting καθώς το παιχνίδι που κατασκευάστηκε δεν είναι large-scale, ενώ παράλληλα, έπρεπε να ισομοιραστεί ο χρόνος μεταξύ κατασκευής του τρισδιάστατου χαρακτήρα/περιβάλλοντος και του μικροελεγκτή Arduino ως Gamepad.

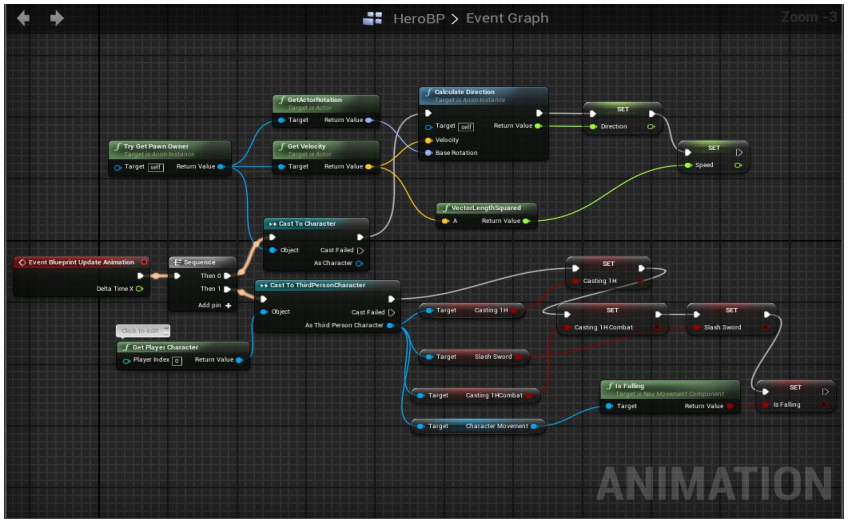
1.4.3. Επεξήγηση Blueprints:

Η μέθοδος Blueprints Visual Scripting αποτελεί ένα ολοκληρωμένο σύστημα κατασκευής παιχνιδιών, βασισμένο στη λογική της διασύνδεσης των στοιχείων του παιχνιδιού και του επεξεργαστή της Unreal Engine 4, μέσω κόμβων. Όπως πολλές γλώσσες προγραμματισμού, έτσι και τα Blueprints χρησιμοποιούνται για να ορίσουν κλάσεις ή αντικείμενα, παράγωγα του αντικειμενοστραφή προγραμματισμού.

Αυτή η μέθοδος προγραμματισμού είναι άκρως ευέλικτη και ισχυρή καθώς προσδίδει τη δυνατότητα στους σχεδιαστές παιχνιδιών, να χρησιμοποιούν με ψηφιακά μέσα, όλα τα εργαλεία που θα χρησιμοποιούσε ένας προγραμματιστής.

Στην αρχική τους μορφή, τα Blueprints προγραμματίζονται εικονικά ανάλογα με τις ανάγκες του παιχνιδιού. Συνδέοντας κόμβους, συναρτήσεις και μεταβλητές μέσω της Wire σύνδεσης είναι εφικτό να δημιουργηθεί ένα αρκετά πολύπλοκο παιχνίδι.

Τα Blueprints χρησιμοποιούν γράφους κόμβων για πληθώρα περιπτώσεων μέσα στον εικονικό κόσμο με σκοπό να εντάξουν συγκεκριμένη συμπεριφορά ή άλλη λειτουργικότητα σε ανεξάρτητα σημεία του παιχνιδιού. Μπορούν επίσης, να επιτεύξουν τη δημιουργία ενός ανεξάρτητου αντικειμένου, ανεξάρτητων συναρτήσεων καθώς και ανεξάρτητων συμβάντων μέσα στο ίδιο το παιχνίδι για την επίτευξη αυτού του σκοπού.



Εικ. 1.5: Blueprint βασικού χαρακτήρα

Τα πιο γνωστά Blueprints που χρησιμοποιούνται είναι τα “Level Blueprints” και τα “Blueprint Classes”. Βέβαια, αυτοί είναι μόνο δύο από τους πολλούς τύπους Blueprints που υπάρχουν.

Level Blueprint:

Κάθε παιχνίδι στην Unreal Engine 4, αποτελείται από τα εκάστοτε επίπεδα (Levels) που έχει ορίσει ο προγραμματιστής. Τα “Level Blueprint” λειτουργούν αυτόνομα σε κάθε επίπεδο επηρεάζοντας πάντα την ίδια οντότητα, όπως το βασικό χαρακτήρα, άλλα αντικείμενα ή συναρτήσεις.

Blueprint Class:

Τα “Blueprint Class”, είναι ιδανικά για τη δημιουργία δυναμικού περιεχομένου όπως πόρτες, διακόπτες, αντικείμενα προς συλλογή ή αντικείμενα προς καταστροφή, τα οποία αλληλεπιδρούν με το χαρακτήρα και το συνολικό περιβάλλον καθ’ όλη διάρκεια ζωής του παιχνιδιού.

Λόγω της αυτο-ολοκληρωτικής συμπεριφοράς που τα διέπει, τα Blueprints μπορούν να κατασκευαστούν με τέτοιο τρόπο ώστε απλώς να προστεθούν στο εκάστοτε επίπεδο και να λειτουργήσουν ορθά χωρίς κάποια περαιτέρω ενέργεια ή ρύθμιση. Κατ’ επέκταση, η παραμετροποίηση ενός Blueprint μπορεί να γίνει μία φορά και να επηρεάσει όλο το έργο.

1.4.4. Ανάλυση Παιχνιδιού:

Με σκοπό την υλοποίηση ενός νέου παιχνιδιού, είναι απαραίτητο να γίνει ανάλυση εις βάθος, στο σύνολό του. Αυτό επιτυγχάνεται αναλύοντας βηματικά τις ανάγκες του και τους λόγους για τους οποίους θα δημιουργηθεί.

Το παρόν παιχνίδι, είχε ως κύρια απαίτηση, το χειρισμό ενός χαρακτήρα σε τρισδιάστατο περιβάλλον με τη χρήση Joystick. Συνεπώς, κατά τη δημιουργία του έργου χρησιμοποιήθηκε η μακέτα ενός τρισδιάστατου εικονικού κόσμου.

Στη δεύτερη φάση της ανάλυσης, σχεδιάστηκε το πλάνο κάτω από το οποίο θα υλοποιηθούν οι κινήσεις, το δημιουργικό κομμάτι και οι λειτουργικότητες του μοντέλου, σύμφωνα πάντα με τους άξονες X, Y & Z.

Έπειτα, σχεδιάστηκαν όλες οι λειτουργικότητες που έχει στη διάθεσή του ο χρήστης, αλλά και που είναι δυνατές να πραγματοποιήσει το μοντέλο. Οι λειτουργικότητες αυτές διαφοροποιούνται σε αναγκαίες και προαιρετικές. Στις αναγκαίες λειτουργικότητες, συμπεριλαμβάνονται το επίπεδο και ο χώρος στον οποίο κινείται το μοντέλο, η ίδια η κίνηση του μοντέλου ως προς όλους τους άξονες αλλά και οι δυνατότητές του με βάση την επίθεση, την άμυνα ή την κίνηση σε συγκεκριμένα τμήματα του επιπέδου. Στις προαιρετικές λειτουργικότητες, κατατάσσονται όλες αυτές που ενισχύουν τη λογική συνέχεια που πρέπει να έχει ένα παιχνίδι, όπως τα οπτικά εφέ κατα την επίθεση ή την άμυνα, οι ήχοι που θα χρησιμοποιηθούν κατά το βηματισμό ή κατά τη συλλογή ενός αντικειμένου, η ίδια η συλλογή ενός αντικειμένου, τα εικονικά μοντέλα τεχνητής νοημοσύνης, ο φωτισμός του χώρου και του χαρακτήρα, κ.α.

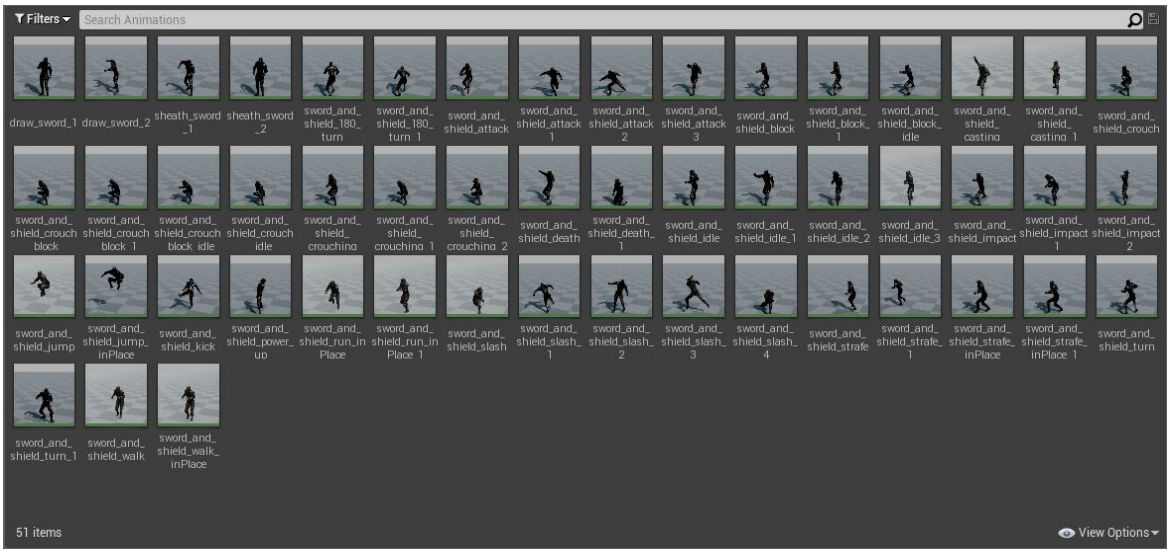
Εφόσον, όλα τα παραπάνω υλοποιούνται κατά προτίμηση, πρέπει να σχεδιαστεί το μέρος του παιχνιδιού που παραμένει στατικό προς το χρήστη ή/και ενεργοποιείται μόνο κατ' εντολή του. Για παράδειγμα, οι ζωτικές ενδείξεις του χαρακτήρα, οι ζωτικές ενδείξεις των AI μοντέλων, το μενού κατά την εκκίνηση ή κατά την παύση του παιχνιδιού, τα εφέ στο στατικό τμήμα του επιπέδου όπως ο ουρανός ή η θάλασσα και το συνολικό σταθερό περιβάλλον που προβάλλει επιπλέον πληροφορίες για το παιχνίδι όπως το όνομα του χαρακτήρα, ή τα εικονίδια των επιθέσεων (HUD), ολοκληρώνουν αυτό το στατικό μέρος.

Με την ολοκλήρωση του έργου, έχει σειρά η διασύνδεση του Arduino Leonardo, με την Unreal Engine 4, αλλά και η αυτόνομη διασύνδεση του μικροελεγκτή με το παράγωγο της Unreal Engine 4, δηλαδή το .exe του παιχνιδιού και την φακελική δομή του.

1.4.5. Assets Animations και Skeletal Meshes:

Animation Assets:

Τα Animation Assets, είναι ξεχωριστά δημιουργικά τμήματα του χαρακτήρα, τα οποία βασισμένα στον ίδιο σκελετό, δημιουργούν μια οντότητα η οποία τροποποιείται ανάλογα με τις ρυθμίσεις που θα δώσει ο προγραμματιστής. Πιο συγκεκριμένα, ο σκελετός αποτελείται από λίγα έως πολλά σημεία κώδικα, όπου το καθένα αφορά και ένα διαφορετικό κομμάτι του εικονικού σώματος. Τα assets είναι ολοκληρωμένα σώματα με το ίδιο δημιουργικό ανά χαρακτήρα, βασισμένα στο ίδιο πρότυπο σκελετού και σχεδιασμένα να καλύπτουν κάθε πιθανή κίνηση που μπορεί να κάνει ο χαρακτήρας. Λειτουργούν είτε στατικά(ο χαρακτήρας πεθαίνει και βρίσκεται ξαπλωμένος έως ότου ο χρήστης προβεί σε μια νέα ενέργεια) είτε επαναλαμβανόμενα (ο χαρακτήρας κινείται, πηδάει ή αλληλεπιδρά με τον εικονικό κόσμο) . Θα μπορούσε κάποιος να παρομοιάσει τα Assets, με το δέρμα, τις κινήσεις και τις αντιδράσεις του εικονικού σκελετού.

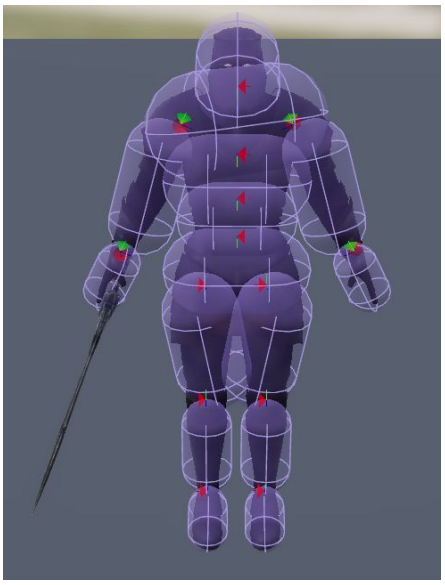


Εικ. 1.6: Όλα τα Animation Assets του βασικού χαρακτήρα

Skeletal Mesh Animations:

Τα Skeletal Mesh, είναι ο πυρήνας ενός χαρακτήρα, αντικειμένου ή AI μοντέλου κατά τη φάση δημιουργίας του παιχνιδιού. Χάρη σε αυτά, ο χαρακτήρας έχει σταθερότητα μέσα στον τρισδιάστατο κόσμο και εικονική υπόσταση. Χωρίς αυτά, όποιες κι αν είναι οι

εντολές που θα λάβει ο χαρακτήρας, δε θα μπορεί να τις εκτελέσει, εφόσον έχουν να αλληλεπιδράσουν με την κίνησή του.



Εικ. 1.7: Το Skeletal Mesh του βασικού χαρακτήρα, με τις δυνάμεις φυσικής

1.4.6. Blueprint Variables και Blueprint Functions:

Blueprint Variables:

Οι μεταβλητές που χρησιμοποιούνται στα Blueprints δεν διαφέρουν σε καμία περίπτωση από τις συνήθεις μεταβλητές που χρησιμοποιούνται σε άλλες γλώσσες προγραμματισμού. Παράλληλα, διαχωρίζονται χρωματικά στον επεξεργαστή της Unreal Engine 4, για τη διευκόλυνση του προγραμματιστή. Πιο συγκεκριμένα, οι μεταβλητές αυτές είναι:

Τύπος Μεταβλητής	Χρωματισμός	Περιγραφή
boolean	Κόκκινο ↔	Δεδομένα με τιμή True/False
integer	Κυανό ↔	Ακέραιοι αριθμοί όπως 0, 152 και -226
float	Πράσινο ↔	Δεκαδικοί αριθμοί όπως 0.0553, 101.2887 και -78.322

string	Ματζέντα ↔	Αλφαριθμητικοί χαρακτήρες όπως “Hello World” και “2nd Part”
text	Ροζ ↔	Απλό κείμενο
vector	Χρυσό ↔	Δεκαδικοί αριθμοί για τη χρήση στοιχείων όπως οι άξονες X,Y & Z και το RGB πρότυπο χρώματος
rotator	Μωβ ↔	Σύνολο αριθμών που αντιπροσωπεύουν την περιστροφή στον τρισδιάστατο χώρο
transform	Πορτοκαλί ↔	Συνδυασμός δεδομένων για την τρισδιάστατη θέση, την περιστροφή και την κλίμακα
object	Μπλε ↔	Όλα τα αντικείμενα που κατασκευάζονται στο περιβάλλον της Unreal Engine 4

Πίνακας 1.3: Χαρακτηριστικά μεταβλητών της Unreal Engine 4

Blueprint Functions:

Οι συναρτήσεις που χρησιμοποιούν τα Blueprints είναι κόμβοι γράφων που ανήκουν αποκλειστικά σε ένα Blueprint που μπορεί να εκτελεστεί ή κληθεί από έναν άλλο γράφο μέσα στο ίδιο Blueprint. Οι συναρτήσεις έχουν μόνο ένα πεδίο εισόδου, σχεδιασμένο από ένα κόμβο με το ίδιο όνομα όπως αυτό της συνάρτησης, ο οποίος έχει μόνο ένα εκτελέσιμο σημείο εξόδου. Όταν μια συνάρτηση καλείται από έναν άλλο γράφο, το εκτελέσιμο σημείο εξόδου ενεργοποιείται, δημιουργώντας την επιτυχή σύνδεση αυτού του εσωτερικού δικτύου και κατόπιν την εκτέλεση της συνάρτησης.

Οι κλήσεις των συναρτήσεων είναι ενέργειες οι οποίες μπορούν να σχηματιστούν, μέσα στα Blueprints που επικοινωνούν με άλλες συναρτήσεις που ανήκουν σε διαφορετικές οντότητες, όπως ο χαρακτήρας ή τα αντικείμενα του παιχνιδιού.

Self Function Calls:

Οι συναρτήσεις τύπου “Self Function Calls” είναι αυτές που ανήκουν στο ίδιο το Blueprint και είτε έχουν δηλωθεί μέσα σε αυτό είτε τις έχει κληρονομήσει από κάποιο άλλο Blueprint.

Other:

Οι συναρτήσεις τύπου “Other Function Calls”, είναι αυτές που ανήκουν σε άλλα αντικείμενα ή μοντέλα, εκτός από το Blueprint.

Pure:

Οι συναρτήσεις τύπου “Pure Function Calls”, είναι εξειδικευμένες ενέργειες οι οποίες εκτελούνται και δεν επηρεάζουν άμεσα τον εικονικό κόσμο, τους χαρακτήρες ή τα αντικείμενα σε αυτόν. Χρησιμοποιούνται συνήθως για την επιστροφή μαθηματικών συνόλων ή την τιμή κάποιας οντότητας.

ΚΕΦΑΛΑΙΟ 2ο : ΠΡΑΚΤΙΚΟ ΜΕΡΟΣ

2.1. Έλεγχος ορθής λειτουργίας Arduino Leonardo

2.1.1. Προγραμματισμός Arduino με το Blink Test:

Κατά την πρώτη σύνδεση του Arduino Leonardo με τον Η/Υ, είναι απαραίτητος ο έλεγχος ορθής του λειτουργίας, έτσι ώστε ο μικροελεγκτής να είναι διαθέσιμος για οποιοδήποτε πρόγραμμα εισάγει ο προγραμματιστής. Δεδομένου αυτού, η Arduino δίνει στους προγραμματιστές κάποια προκαθορισμένα Sketches, τα οποία είτε εξυπηρετούν απλές περιπτώσεις χρήσης του μικροελεγκτή, είτε επιβεβαιώνουν τον έλεγχο ορθής λειτουργίας του.

Το προκαθορισμένο πρόγραμμα που χρησιμοποιήθηκε και βεβαίωσε πως το Arduino Leonardo αλληλεπιδρά με τον Η/Υ, και πως όλες οι ενδείξεις του λειτουργούν σωστά, είναι το “Blink”.

Το “Blink”, είναι ένα απλό πρόγραμμα το οποίο μπορεί να δείξει στο χρήστη ότι η φυσική έξοδος του Arduino, είναι σε πλήρη λειτουργία. Αυτό επιτυγχάνεται καθώς το ίδιο το πρόγραμμα, ενεργοποιεί το on-board LED του μικροελεγκτή και με τη χρήση της επανάληψης του κώδικα, αυτο ενεργοποιείται και απενεργοποιείται σε χρόνο ανάλογο με αυτόν που έχει ορίσει ο προγραμματιστής. Στο παρακάτω παράδειγμα, ο χρόνος ενεργοποίησης - απενεργοποίησης ανέρχεται στα 1000 ms ή αλλιώς 1 sec:

```
/*   Blink   */
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
  delay(1000);
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000);
}
```

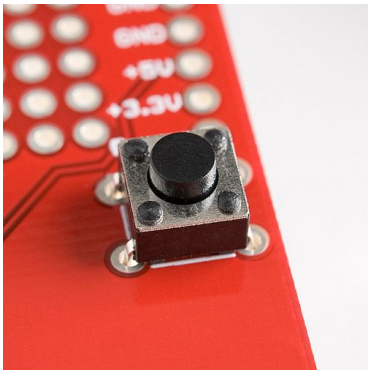
2.2. Προετοιμασία Joystick Shield

2.2.1. Συναρμολόγηση Joystick Shield:

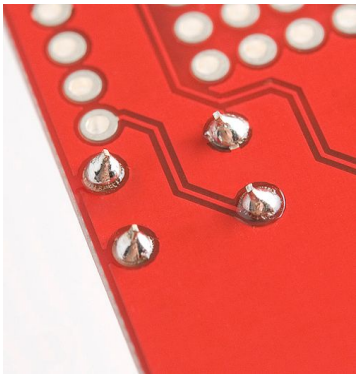
Για να καταστεί δυνατή η χρήση του Joystick Shield σε συνδυασμό με το Arduino Leonardo, θα πρέπει πρώτα, να γίνει η συναρμολόγησή του. Για τη συναρμολόγηση, είναι απαραίτητη η χρήση σίδερου συγκόλλησης και συγκολλητικού κράματος μαλακής συγκόλλησης (καλάι). Πιο συγκεκριμένα, το συγκολλητικό κράμα που χρησιμοποιήθηκε είναι ένας μείγμα περιεκτικότητας 96.35% σε κασσίτερο, 3.0% σε χαλκό, 0.5% σε άργυρο και 0.15% σε αντιμόνιο και απαιτεί συγκόλληση σε χαμηλές θερμοκρασίες.

Για να επιτευχθεί η συγκόλληση του Joystick Shield, είναι αναγκαίο να τοποθετηθούν στην βασική πλακέτα του, όλα τα εξαρτήματα που συντελούν στην ολοκλήρωσή του. Εφόσον τοποθετηθούν και ελεγχθεί η σταθερότητά τους, τότε αρχίζει βηματικά η συγκόλληση τους. Η σειρά με την οποία θα εφαρμοστούν και συγκολληθούν τα εξαρτήματα του Joystick Shield φαίνεται παρακάτω:

Πρώτα θα πρέπει να συγκολληθεί το reset πλήκτρο:

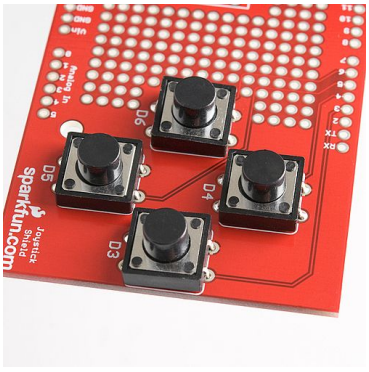


Εικ. 2.1: Πλήκτρο "Reset" άνω όψη

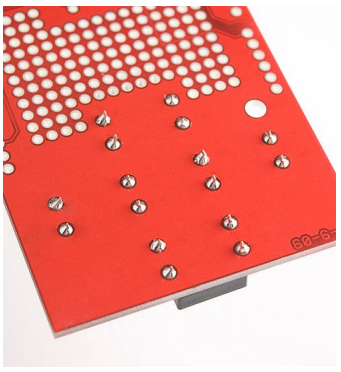


Εικ. 2.2: Πλήκτρο "Reset" κάτω όψη

Σειρά έχουν τα 4 βασικά πλήκτρα:

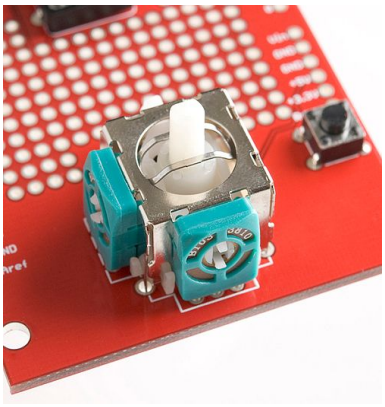


Εικ. 2.3: Πλήκτρα ψηφιακών ακροδεκτών D3 - D6 (άνω όψη)

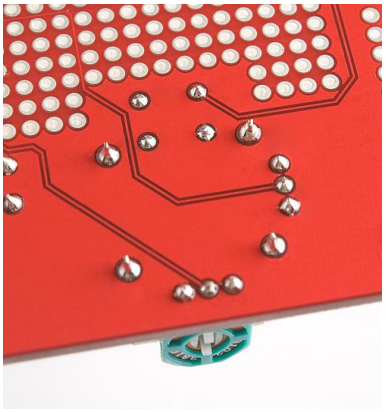


Εικ. 2.4: Πλήκτρα ψηφιακών ακροδεκτών D3 - D6 (κάτω όψη)

Το επόμενο βήμα απαιτεί τη συγκόλληση του αναλογικού μοχλού ή αλλιώς Joystick:

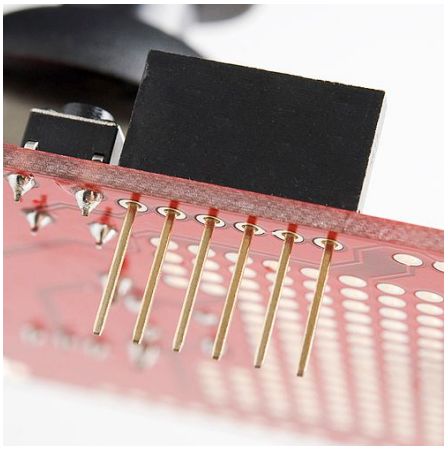


Εικ. 2.5:Αναλογικός μοχλός (άνω όψη)

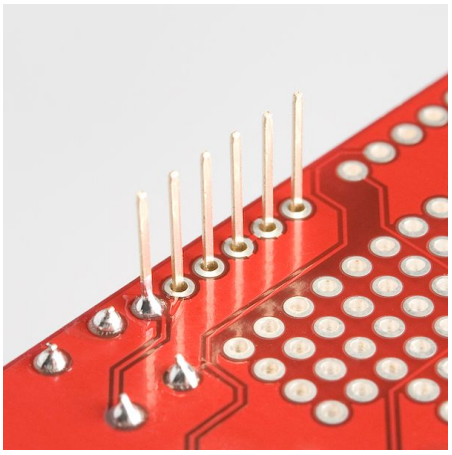


Εικ. 2.6:Αναλογικός μοχλός (κάτω όψη)

Τέλος, σειρά έχουν οι κεφαλές του Joystick Shield:



Εικ. 2.7:Κεφαλές πλακέτας (άνω όψη)



Εικ. 2.8: Κεφαλές πλακέτας (κάτω όψη)

Το τελευταίο βήμα είναι η τοποθέτηση του Joystick Shield στο Arduino Leonardo ώστε να βεβαιωθεί η σωστή εφαρμογή του σε αυτό. Εφόσον εφαρμόζεται σωστά, τότε είναι έτοιμο, για την πρώτη του προγραμματιστική δοκιμή.

2.2.2. Σύνδεση με Arduino Leonardo:

Η σύνδεση με το Arduino Leonardo είναι άκρως εφικτή. Το Joystick Shield, απλώς τοποθετείται πάνω από το Arduino Leonardo, και το δεύτερο με τη σειρά του, συνδέεται στον Η/Υ ώστε να ξεκινήσει η αναγνώριση τόσο αυτού, όσο και του Joystick Shield.

Σαν πρώτο έλεγχο διασύνδεσης του Joystick με το περιβάλλον του Η/Υ, χρησιμοποιείται η συνάρτηση Serial.print(), για τους ακροδέκτες με τους οποίους επικοινωνούν τα 5 πλήκτρα και το Joystick. Έτσι, πιέζοντας κάποιο από τα πλήκτρα, ή μετακινώντας το Joystick, επιστρέφεται στην οθόνη του Η/Υ μέσα από το πρόγραμμα επεξεργασίας του Arduino, η τιμή που έχουν ως όρισμα.

Από τη στιγμή που η επιστρεφόμενη τιμή υπάρχει, η σύνδεση και η επικοινωνία μεταξύ Arduino Leonardo και Joystick Shield είναι επιτυχημένη. Στην περίπτωση που δεν επιστρέφεται κάποια τιμή κατά τη χρήση του Joystick Shield, τότε υπάρχει η πιθανότητα να μην έχει γίνει σωστά η συγκόλληση. Εν συνεχεία, εάν η επιστρεφόμενη τιμή είναι λάθος, τότε απλώς πρέπει να ελεγχθεί η παράμετρος της Serial.print() για τυχόν σφάλματα, ή για χρήση χαρακτήρα που δεν ανήκει στην κωδικοποίηση του κειμένου που εκτυπώνεται.

Εάν όλα τα βήματα ελέγχου, εκτελεστούν επιτυχώς και το αποτέλεσμα είναι θετικό, μπορεί πλέον να ξεκινήσει ο βασικός προγραμματισμός του Arduino Leonardo. Στην περίπτωση που είναι αρνητικό, τότε θα πρέπει να ελεγχθούν βηματικά για τυχόν σφάλματα ο μικροελεγκτής και το Joystick. Φυσικά, ο έλεγχος αφορά τόσο το φυσικό όσο και το ψηφιακό επίπεδο.

Εικ. 2.9:Διασύνδεση του Joystick Shield με τον μικροελεγκτή Arduino Leonardo

2.3. Προγραμματισμός Gamepad

2.3.1. Προβλήματα στην επικοινωνία:

Αρχικοποιώντας, τη δομή του κώδικα που πρέπει να χρησιμοποιηθεί για τον έλεγχο και τη λειτουργία του Arduino Leonardo, παρατηρήθηκε πως η συνάρτηση serial.print() που εκτυπώνει την είσοδο που έχουμε ορίσει σε κάθε πλήκτρο, εκτύπωνε αποτελέσματα μόνο για το Joystick και όχι για τα πλήκτρα.

Αυτό οφειλόταν στην επικοινωνία του Joystick Shield με τους ψηφιακούς ακροδέκτες του Arduino. Συνεπώς η σειριακή επικοινωνία μεταξύ χειριστηρίου και λογισμικού δεν ήταν εφικτή.

Για να λυθεί αυτό το πρόβλημα, έγινε μετά από μελέτη, χρήση του "UE4Duino" (λογισμικό ανοιχτού κώδικα), το οποίο επιτρέπει τη σειριακή επικοινωνία μεταξύ χειριστηρίου και Unreal Engine 4. Δυστυχώς το λογισμικό αυτό, δεν είναι ακόμη συμβατό με την έκδοση 4.18 της μηχανής, συνεπώς το πρόβλημα παρέμεινε άλυτο.

2.3.2. Αντιμετώπιση προβλημάτων:

Το Arduino Leonardo, λόγω της διασύνδεσης μέσω USB που χρησιμοποιεί, επιτρέπει στον επεξεργαστή του να επεξεργαστεί κάποιες Arduino βιβλιοθήκες, που εκτελούνται μόνο σε μικροελεγκτές με USB υποστήριξη όπως το Arduino UNO R3, το Arduino Micro, το Arduino Nano και το Arduino Leonardo.

Μία από αυτές τις βιβλιοθήκες είναι η "Keyboard.h", η οποία προσπερνάει τη σειριακή διασύνδεση μέσω του USB και χρησιμοποιεί την κωδικοποίηση ASCII για να στείλει εντολές στο λογισμικό, όπως ένα πληκτρολόγιο. Συνεπώς τα ορίσματα στον κώδικα του Gamepad, αφορούν τα πλήκτρα τα οποία ορίζει η Unreal Engine 4 για την κίνηση του χαρακτήρα, την αλληλεπίδρασή του με το τρισδιάστατο περιβάλλον και την εκκίνηση του παιχνιδιού.

Ένα απλό παράδειγμα που μας επιτρέπει να χρησιμοποιήσουμε την εντολή "Ctrl + N", στο περιβάλλον του H/Y και μόνο για τα λειτουργικά συστήματα Windows και Linux είναι το παρακάτω:

```
#include <Keyboard.h>

char ctrlKey = KEY_LEFT_CTRL;
void setup() {
```

```
  // make pin 2 an input and turn on the
  // pullup resistor so it goes high unless
  // connected to ground:
  pinMode(2, INPUT_PULLUP);
  // initialize control over the keyboard:
  Keyboard.begin();
}

void loop() {
  while (digitalRead(2) == HIGH) {
    // do nothing until pin 2 goes low
    delay(500);
  }
  delay(1000);
  // new document:
  Keyboard.press(ctrlKey);
  Keyboard.press('n');
  delay(100);
  Keyboard.releaseAll();
  // wait for new window to open:
  delay(1000);
```

Η χρήση αυτού του παραδείγματος επιτρέπει το άνοιγμα μιας νέας καρτέλας στο φυλλομετρητή που χρησιμοποιεί ο χρήστης.

2.4. Βιβλιοθήκη *Keyboard.h*

2.4.1. Χρήση της βιβλιοθήκης και δημιουργία του προγράμματος:

Οι συναρτήσεις της βιβλιοθήκης Keyboard.h ενεργοποιούν μικροελεγκτές τύπου 32u4 ή SAMD ώστε να γίνει δυνατή η αποστολή εντολών στο περιβάλλον του Η/Υ, με συμπεριφορά ίδια όπως αυτή του πληκτρολογίου ή του ποντικιού, μέσω της micro USB θύρας τους.

Μπορεί να εκτυπώσει σχεδόν όλους τους ASCII χαρακτήρες, εκτός από αυτούς που δεν εκτυπώνονται σε φυσικό επίπεδο. Είναι σημαντικό να ειπωθεί πως η βιβλιοθήκη Keyboard.h επιτρέπει στους μικροελεγκτές τύπου 32u4 και SAMD, όπως οι Leonardo, Esplora, Zero, Due και η οικογένεια των MKR, να εμφανίζονται στο περιβάλλον του Η/Υ ως συνδεδεμένο πληκτρολόγιο ή ποντίκι. Το βασικό, βέβαια, μειονέκτημα της βιβλιοθήκης είναι πως αν το πρόγραμμα του μικροελεγκτή διαβάζεται από τον Η/Υ και ταυτόχρονα γίνεται επεξεργασία του κώδικα, τότε θα έρχεται σε σύγκρουση η εκτέλεση με την παραμετροποίηση καθώς κάποια πλήκτρα ή ο κέρσορας θα εκτελούνται αυτόματα. Για τον λόγο αυτό, είναι προτιμότερο, να εκτελείται το πρόγραμμα εφόσον έχει ολοκληρωθεί η σύνταξη του κώδικά του, ενώ όταν πρόκειται για παραμετροποίηση, τότε το πρόγραμμα να σταματά προσωρινά.

Οι συναρτήσεις που χρησιμοποιεί η βιβλιοθήκη Keyboard.h περιγράφονται αναλυτικά παρακάτω:

Keyboard.begin():

Η συνάρτηση αυτή, δίνει την εντολή για την εκκίνηση της μίμησης από τον μικροελεγκτή, ως πληκτρολόγιο.

Keyboard.end():

Σε αντίθετη περίπτωση με την Keyboard.begin(), η συνάρτηση Keyboard.end() τερματίζει την κατάσταση της μίμησης του μικροελεγκτή.

Keyboard.press():

Όταν καλείται η συνάρτηση αυτή, λειτουργεί όπως το χτύπημα ενός πλήκτρου στο πληκτρολόγιο, δηλαδή μεταφέρει στον υπολογιστή το πλήκτρο που ο χρήστης επιλέγει.

Keyboard.print():

Η συνάρτηση Keyboard.print(), εκτυπώνει στην οθόνη του Η/Υ, την τιμή που έχει δωθεί, στη μεταβλητή που ορίζεται ως πλήκτρο.

Keyboard.println():

Σε αναλογία με την Keyboard.print(), η συνάρτηση Keyboard.println() εκτυπώνει την τιμή που έχει οριστεί, μαζί με μια νέα γραμμή.

Keyboard.release():

Εφόσον η Keyboard.press() είναι σε χρήση, η συνάρτηση Keyboard.release() αποδεσμεύει από την πρώτη, την παράμετρο που έχει λάβει ως πλήκτρο.

Keyboard.releaseAll():

Στην περίπτωση που μέσω της Keyboard.press() πληκτρολογούνται πολλά πλήκτρα, η συνάρτηση Keyboard.release() τα αποδεσμεύει όλα.

Keyboard.write():

Η συνάρτηση Keyboard.write(), λειτουργεί όπως ένα πληκτρολόγιο. Δηλαδή, εκτυπώνει όπως ένα πλήκτρο, την τιμή που έχει λάβει και το αποδεσμεύει άμεσα.

Με τη χρήση των παραπάνω συναρτήσεων αλλά και με τις αρχές προγραμματισμού του Arduino, δημιουργήθηκε το παρακάτω πρόγραμμα, στο οποίο ορίζονται ως σταθερές οι χαρακτήρες ‘X’, ‘C’, ‘V’, ‘SPACEBAR’ για τα πλήκτρα του Joystick Shield και ‘ ← ’, ‘ → ’, ‘ ↑ ’, ‘ ↓ ’ για τις θέσεις των αξόνων στο Joystick. Έπειτα εισάγονται σε ένα περιβάλλον if/else έτσι ώστε, όταν ένα πλήκτρο του Gamepad, ή όταν ο άξονας του Joystick χρησιμοποιηθούν, να γίνει η αντιστοίχιση με αυτά. Να σημειωθεί φυσικά πως η επιλογή των συγκεκριμένων πλήκτρων οριστικοποιήθηκε από τα πλήκτρα που χρησιμοποιήθηκαν στο περιβάλλον της Unreal Engine 4:

```
#include <Keyboard.h>

//Ορίζουμε τα πλήκτρα του Joystick Shield
const char btn_joy = ' ';
const char btn_right = 'c';
const char btn_up = 'x';
```

```
const char btn_left = (char) 0x20; //Το πλήκτρο SPACEBAR σε HEX
const char btn_down = 'v';

//Ορίζουμε τα pins που χρησιμοποιούνται από κάθε πλήκτρο
const int btn_joy_pin = 2;
const int btn_right_pin = 3;
const int btn_up_pin = 4;
const int btn_left_pin = 5;
const int btn_down_pin = 6;

//Ορίζουμε την κατάσταση του κάθε πλήκτρου
int btn_joy_status = HIGH;
int btn_right_status = HIGH;
int btn_up_status = HIGH;
int btn_left_status = HIGH;
int btn_down_status = HIGH;

//Ορίζουμε τα pins που χρησιμοποιούνται από το Joystick
const int joystick_x_pin = A0;
const int joystick_y_pin = A1;

//Ορίζουμε την κατάσταση του κάθε άξονα (x,y)
int joystick_x_status = 0;
int joystick_y_status = 0;

void setup() {
  pinMode(btn_joy_pin, INPUT_PULLUP);
  pinMode(btn_right_pin, INPUT_PULLUP);
  pinMode(btn_up_pin, INPUT_PULLUP);
  pinMode(btn_left_pin, INPUT_PULLUP);
  pinMode(btn_down_pin, INPUT_PULLUP);

  Keyboard.begin();
}

void loop() {
  //Έλεγχος κάθε 50 millisec για την παρουσία πλήκτρων
  static unsigned long timestamp = 0;
  if(millis() - timestamp > 50){
    int status;
    //Έλεγχος για πάτημα του btn_joy
    status = digitalRead(btn_joy_pin);
    if(status != btn_joy_status){
      if(status == LOW){//αν πατηθεί
        //να γίνει το παρακάτω keystroke και να σταματήσει
        Keyboard.press(btn_joy);
```

```
      Keyboard.release(btn_joy);
    }
    btn_joy_status = status;
  }

  //Έλεγχος για πάτημα του btn_right
  status = digitalRead(btn_right_pin);
  if(status != btn_right_status){
    if(status == LOW){//αν πατηθεί
      //να γίνει το παρακάτω keystroke και να σταματήσει
      Keyboard.press(btn_right);
      Keyboard.release(btn_right);
    }
    btn_right_status = status;
  }

  //Έλεγχος για πάτημα του btn_up
  status = digitalRead(btn_up_pin);
  if(status != btn_up_status){
    if(status == LOW){//αν πατηθεί
      //να γίνει το παρακάτω keystroke και να σταματήσει
      Keyboard.press(btn_up);
      Keyboard.release(btn_up);
    }
    btn_up_status = status;
  }

  //Έλεγχος για πάτημα του btn_left
status = digitalRead(btn_left_pin);
if(status != btn_left_status){
  if(status == LOW){//αν πατηθεί
    ///να γίνει το παρακάτω keystroke και να σταματήσει
    Keyboard.press((char) 0x20);
    delay(50);
    Keyboard.release((char) 0x20);
    delay(50);
  }
  btn_left_status = status;
}

//Έλεγχος για πάτημα του btn_down
  status = digitalRead(btn_down_pin);
  if(status != btn_down_status){
    if(status == LOW){//αν πατηθεί
      //να γίνει το παρακάτω keystroke και να σταματήσει
      Keyboard.press(btn_down);
```

```
        Keyboard.release(btn_down);
      }
      btn_down_status = status;
    }

    int analog_status;

    //Παίρνουμε την θέση του άξονα x
    analog_status = get_joystick_status(joystick_x_pin);
    if(analog_status != joystick_x_status)
    {
      //Αν βρίσκεται στη μέση
      if(joystick_x_status == 0)
      {
        //Αν βρίσκεται δεξιά, τότε χρησιμοποιούμε το Keystroke
για το δεξί βέλος
        if(analog_status == 1)
        {
          Keyboard.release(KEY_LEFT_ARROW);
          Keyboard.press(KEY_RIGHT_ARROW);
        }
        else
        {
          //Αν βρίσκεται αριστερά, τότε χρησιμοποιούμε το
Keystroke για το αριστερό βέλος
          Keyboard.press(KEY_LEFT_ARROW);
          Keyboard.release(KEY_RIGHT_ARROW);
        }
      }
      else
      {
        //Αν δεν βρίσκεται στη μέση τότε ελευθερώνουμε τον άξονα
        Keyboard.release(KEY_LEFT_ARROW);
        Keyboard.release(KEY_RIGHT_ARROW);
      }
      //Αποθηκεύουμε τη νέα θέση
      joystick_x_status = analog_status;
    }

    //Επαναλαμβάνουμε για τον άξονα y
    analog_status = get_joystick_status(joystick_y_pin);
    if(analog_status != joystick_y_status)
    {
      if(joystick_y_status == 0)
      {
        if(analog_status == 1)
```

```
        {
          Keyboard.release(KEY_DOWN_ARROW);
          Keyboard.press(KEY_UP_ARROW);
        }
        else
        {
          Keyboard.press(KEY_DOWN_ARROW);
          Keyboard.release(KEY_UP_ARROW);
        }
      }
      else
      {
        Keyboard.release(KEY_UP_ARROW);
        Keyboard.release(KEY_DOWN_ARROW);
      }
      joystick_y_status = analog_status;
    }

  }
}

//Η παρακάτω συνάρτηση επιστρέφει την τιμή 0 εάν η θέση του
joystick είναι στη μέση, την τιμή 1 εάν η θέση είναι δεξιά ή πάνω
//και -1 εάν η θέση είναι αριστερά ή κάτω
int get_joystick_status(int pin)
{
  int joystick_status = analogRead(pin);
  if(joystick_status > 1000)
    joystick_status = 1;
  else if(joystick_status < 50)
    joystick_status = -1;
  else
    joystick_status = 0;

  return joystick_status;

}
```

2.4.2. Έλεγχος ορθής λειτουργίας μετά τον προγραμματισμό:

Κατά τη λήξη του προγραμματιστικού μέρους και την ολοκλήρωση της δημιουργίας του προγράμματος, μπορούν να πραγματοποιηθούν συγκεκριμένες δοκιμές ώστε να επιβεβαιωθεί το γεγονός πως πατώντας ένα πλήκτρο του Joystick Shield ή μετακινώντας

τον άξονα του Joystick, το Arduino αλληλεπιδρά με τον Η/Υ και εκτυπώνει τις αντίστοιχες εντολές.

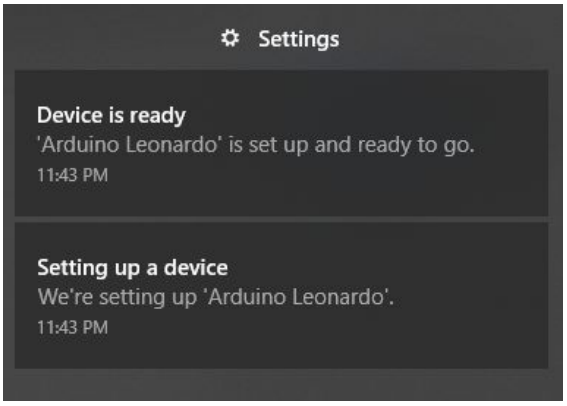
Μέσω ενός προγράμματος δημιουργίας εγγράφου, μπορεί να επιβεβαιωθεί πως το Arduino εκτυπώνει στο φύλλο του εγγράφου, όλους τους παραπάνω χαρακτήρες.

Με τον έλεγχο ορθής λειτουργίας σε ένα τέτοιο περιβάλλον μπορούν να ελεγχθούν παράγοντες όπως η σωστή εγγραφή και αντιστοίχιση των πλήκτρων, οι χρόνοι απόκρισης από τη στιγμή που δόθηκε η εντολή έως ότου εκτυπώθηκε, αλλά και περαιτέρω παραμετροποίησεις σχετικα με δοκιμές μεταξύ ευφυέστερων μεθόδων σύνταξης του προγράμματος.

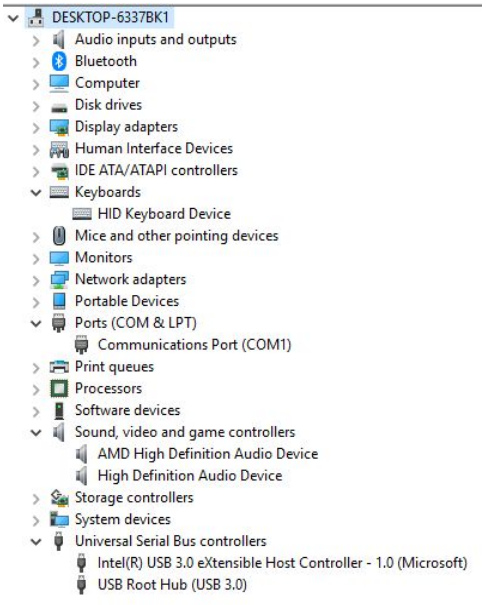
2.5. Αναγνώριση ως Gamepad

2.5.1. Έλεγχος λειτουργίας σε άλλα παιχνίδια:

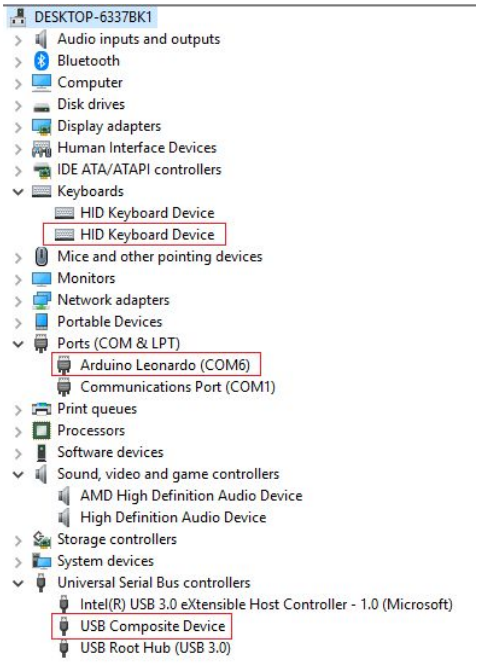
Κατά τη σύνδεση του μικροελεγκτή με τον Η/Υ, χρειάζεται να γίνουν κάποιες ενέργειες, ώστε το λειτουργικό σύστημα να αναγνωρίσει το Arduino ως Gamepad, και όχι απλώς ως μια συσκευή USB. Για να επιτευχθεί αυτό, χρησιμοποιήθηκε η βιβλιοθήκη ανοιχτού κώδικα λογισμικού "ArduinoJoystickLibrary". Χάρη σε αυτή, το λειτουργικό σύστημα αναγνωρίζει πλέον το μικροελεγκτή ώς συσκευή HID, συνεπώς και ως Gamepad. Το επόμενο βήμα είναι να ρυθμιστεί στο εκάστοτε περιβάλλον το Gamepad και να ελεγχθεί η σωστή λειτουργία όλων των πλήκτρων και όλων των θέσεων του Joystick στους άξονες X, Y, Z.



Εικ. 2.10: Ο μικροελεγκτής Arduino Leonardo, αναγνωρίζεται από το λειτουργικό σύστημα



Εικ. 2.11: Πριν τη σύνδεση του μικροελεγκτή



Εικ. 2.12: Μετά τη σύνδεση του μικροελεγκτή

Βασική προϋπόθεση για την επιτυχημένη μετατροπή του Arduino Leonardo σε Gamepad, είναι η δοκιμή του σε πληθώρα παιχνιδιών τα οποία έχουν δημιουργηθεί για διαφορετικές πλατφόρμες. Συγκεκριμένα, υπάρχουν ιστότοποι οι οποίοι παρέχουν HTML5 παιχνίδια, τα οποία χρησιμοποιούν το πληκτρολόγιο για την κίνηση ενός χαρακτήρα, ή ενός αντικειμένου. Εφόσον το Gamepad λειτουργήσει φυσιολογικά στο HTML5 περιβάλλον, η δοκιμή του μπορεί να συνεχιστεί στο Software based παιχνίδι "SuperTuxKart". Το συγκεκριμένο παιχνίδι είναι πρότυπο όσον αφορά την αλληλεπίδρασή του με άλλα λογισμικά ή συσκευές εισόδου, σε ερευνητικό επίπεδο.

Εικ. 2.13: Το Linux based ηλεκτρονικό παιχνίδι "SuperTuxKart"

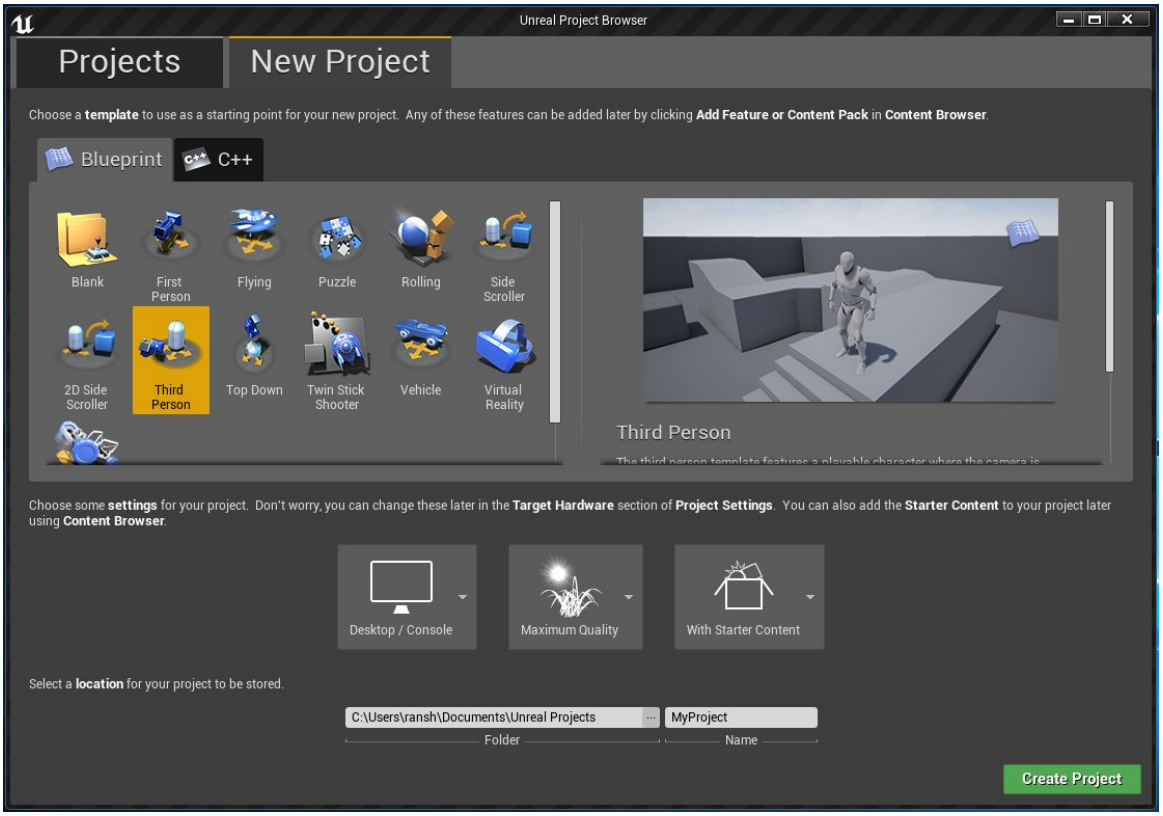
Με το πέρας της ορθής λειτουργίας του Gamepad, μπορεί να πραγματοποιηθεί η τελική ανασκόπηση του κώδικα και να γίνουν, εφόσον υπάρχουν, περαιτέρω βελτιώσεις σε αυτόν.

2.6. Δημιουργία Project στην Unreal Engine 4 (v4.18)

2.6.1. Δημιουργία Χαρακτήρα:

Κατά τη δημιουργία ενός νέου παιχνιδιού στην Unreal Engine 4, δίνεται η δυνατότητα στον προγραμματιστή να επιλέξει ανάμεσα σε πολλά έτοιμα προσχέδια, τα οποία αρχικοποιούν το παιχνίδι και τη φακελική του δομή και παράλληλα είναι αναγκαία για τη χρήση δισδιάστατου ή τρισδιάστατου χώρου, για τη δημιουργία του πρωταγωνιστικού μοντέλου (βασικός χαρακτήρας), για την εμφάνιση αντικειμένων, κ.α.

Μία από τις απαιτήσεις του συγκεκριμένου έργου είναι η δημιουργία ενός τρισδιάστατου χαρακτήρα, ο οποίος είναι εξαναγκασμένος να κινείται σε τρισδιάστατο περιβάλλον ώστε να υφίσταται η ύπαρξή του στον ψηφιακό κόσμο. Για το λόγο αυτό χρησιμοποιήθηκε το προσχέδιο “Third Person”, πάνω στο οποίο έγινε η υλοποίηση του παιχνιδιού.



Εικ. 2.14: Δημιουργία νέου “Third Person” project στην UE4

Επιλέγοντας αυτό το προσχέδιο, γίνεται αυτόματα και η δημιουργία του βασικού χαρακτήρα, με τον ψηφιακό σκελετό και τις βασικές συναρτήσεις κίνησης στους άξονες X, Y, Z σε λειτουργία. Το πρότυπο μοντέλο λέγεται Mannequin και είναι ένας έτοιμος

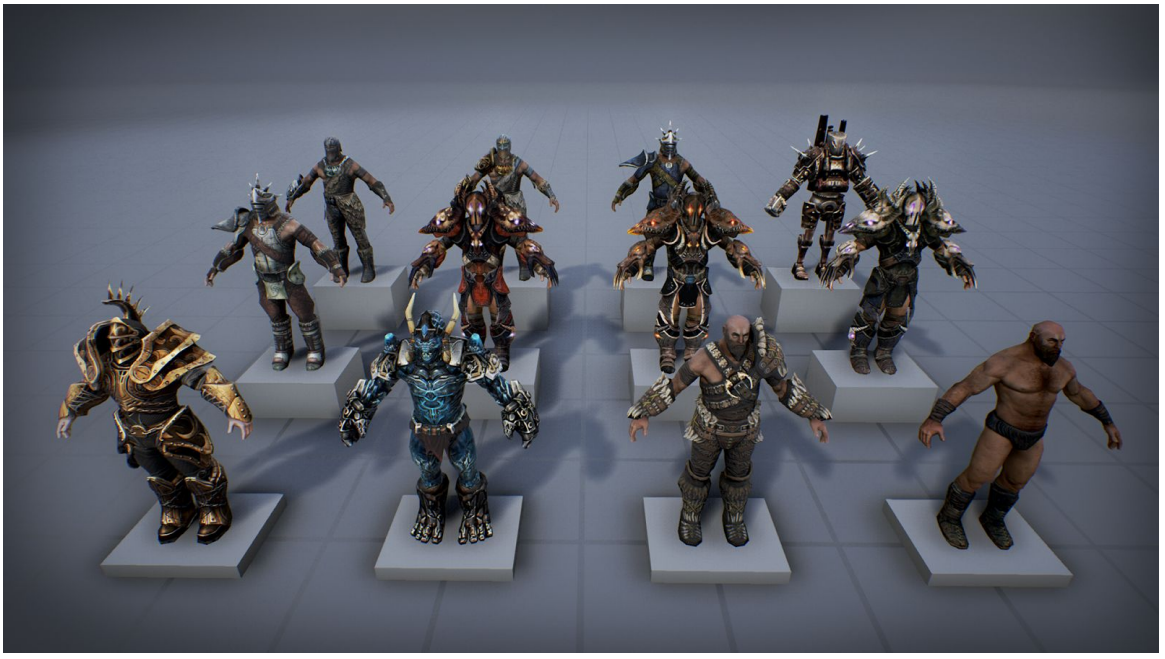
χαρακτήρας, χωρίς assets αλλά με πλήρη λειτουργικότητα. Με τη σωστή τροποποίησή του το τελικό αποτέλεσμα είναι αντάξιο του χαρακτήρα που έχει επιλέξει ο προγραμματιστής, όλη όμως η τεχνική επεξεργασία του χαρακτήρα, γίνεται πάνω στο Mannequin.

Εικ. 2.15: Ο χαρακτήρας "Mannequin"

Το επόμενο βήμα, είναι η τοποθέτηση των Assets και των Animation που θα ολοκληρώσουν την εμφάνιση του χαρακτήρα και θα δώσουν την πρώτη αίσθηση διαδραστικότητας στον προγραμματιστή.

Εικ. 2.16: Ο βασικός χαρακτήρας, μετά την τοποθέτηση των Assets

Για το συγκεκριμένο χαρακτήρα, χρησιμοποιήθηκαν κάποια ανοιχτού κώδικα Assets, δημιουργημένα από την ίδια την Unreal Engine 4 και τη Mixamo της Adobe. Η σειρά "Infinity Blade", δίνει τη δυνατότητα στον προγραμματιστή, να χρησιμοποιήσει ένα τεράστιο φάσμα από assets όπως τοπία, χαρακτήρες, αντικείμενα, κ.α., με σκοπό την εμβάθυνσή του στο λειτουργικό κομμάτι του παιχνιδιού και όχι στο γραφιστικό.



Εικ. 2.17: Τα assets που χρησιμοποιήθηκαν για τα μοντέλα Τεχνητής Νοημοσύνης

Εικ. 2.18: Τα assets που χρησιμοποιήθηκαν για όλα τα αντικείμενα και το περιβάλλον του βασικού επιπέδου

Στη συνέχεια, πρέπει να δημιουργηθούν λειτουργικότητες για τα ζωτικά σημεία του χαρακτήρα. Συνεπώς, είναι απαραίτητη η δημιουργία μεταβλητών που αναλόγως με τις ενέργειες του χαρακτήρα επηρεάζεται άμεσα η ένδειξη ζωής του, η απαιτούμενη

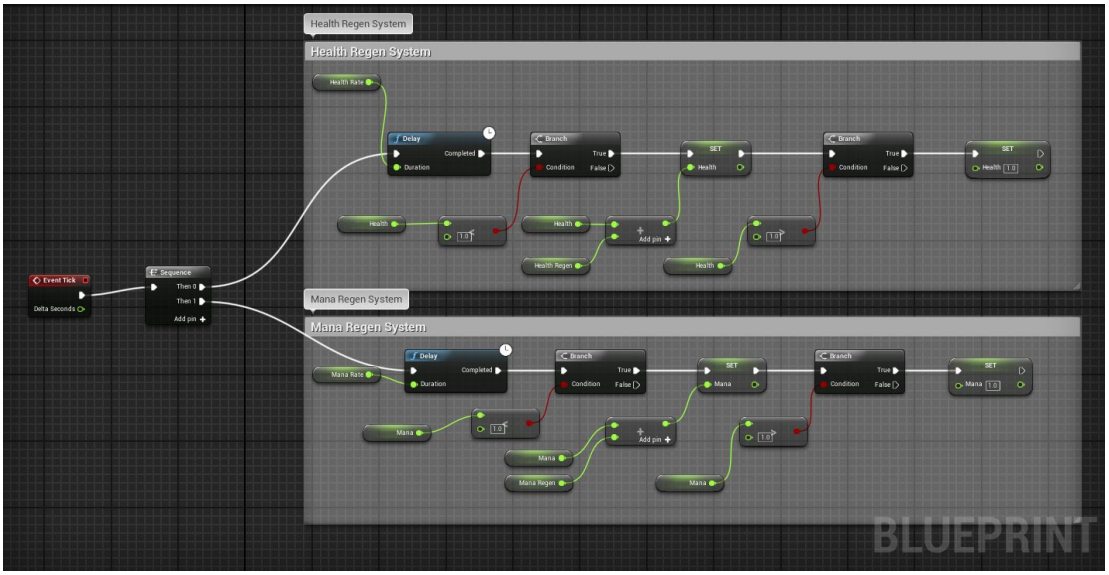
ενέργεια που πρέπει να χρησιμοποιήσει για μια επίθεση και το πόσα αντικείμενα πρέπει να συλλέξει, κατά την εξερεύνηση του επιπέδου.

Οι ζωτικές μεταβλητές είναι η Health και η Energy. Η πρώτη αφορά την εικονική ζωή του χαρακτήρα και η δεύτερη το ποσοστό ενέργειας που διαθέτει ώστε να πραγματοποιήσει την επόμενη επίθεση ή ενδυνάμωση.

Η μεταβλητή Health, είναι ένας δεκαδικός αριθμός με εύρος 0.00 έως 1.00. Η τιμή που λαμβάνει κάθε φορά έχει να κάνει με το αν ο χαρακτήρας έχει δεχτεί κάποια επίθεση ή αν έχει χρησιμοποιήσει την εικονική ενδυνάμωση. Όταν η μεταβλητή αυτή είναι ίση με το 0.00 τότε το παιχνίδι σταματά και ο χαρακτήρας δεν αλληλεπιδρά πλέον με το επίπεδο. Όταν η τιμή είναι ίση με 1.00 τότε ο χαρακτήρας δεν μπορεί να χρησιμοποιήσει την πλέον την ενδυνάμωση ζωής. Συμπληρωματικά, όλες οι τιμές που είναι μεγαλύτερες του 0.00, επιτρέπουν στο χαρακτήρα να συνεχίσει την εξερεύνηση του επιπέδου.

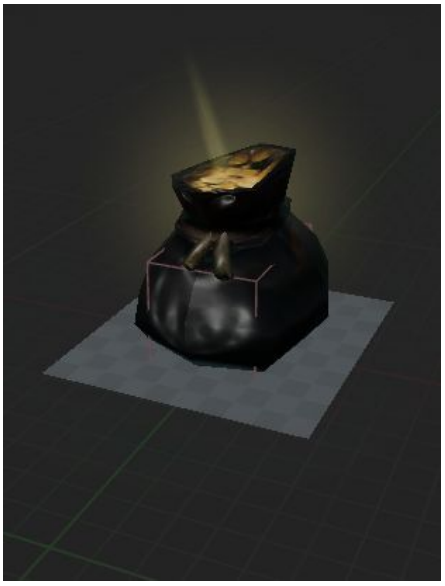
Η μεταβλητή Energy, λειτουργεί όπως και η τιμή Health, μόνο που επηρεάζει τον αριθμό σύνθετων επιθέσεων ή ενδυναμώσεων που μπορεί να εκτελέσει ο χαρακτήρας. Όταν η τιμή είναι κάτω από 0.10 ο χαρακτήρας δεν μπορεί να χρησιμοποιήσει καμία ενέργεια. Επίσης, κάθε σύνθετη επίθεση ή ενδυνάμωση, αφαιρεί ένα ποσοστό της ενέργειας οπότε ο αριθμός αυτός μικραίνει.

Και για τις δύο αυτές ζωτικές μεταβλητές, ενεργοποιείται μια επαναληπτική κατάσταση, η οποία τους προσθέτει την τιμή 0.05 ανά τακτά χρονικά διαστήματα, με αποτέλεσμα αν ο χαρακτήρας παραμείνει σε περιβάλλον μη αλληλεπίδρασης με άλλα αντικείμενα ή μοντέλα τεχνητής νοημοσύνης, να επαναφέρει τα ζωτικά του επίπεδα στο μέγιστο δυνατό.



Εικ. 2.19: Το Blueprint ροής των μεταβλητών Health & Energy

Σχεδιασμένη με την ίδια αρχιτεκτονική όπως οι Health και Energy μεταβλητές, είναι και η μεταβλητή Loot. Η Loot είναι ένας ακέραιος αριθμός που αντιπροσωπεύει το πλήθος των αντικειμένων αξίας, που έχει συλλέξει ο χαρακτήρας. Η συλλογή τους επιτυγχάνεται είτε βρίσκοντας ένα αντικείμενο στο επίπεδο, είτε πολεμώντας με τα τεχνητής νοημοσύνης μοντέλα. Όταν ο χαρακτήρας συλλέξει 10 αντικείμενα αξίας, τότε έχει ολοκληρώσει το σκοπό του παιχνιδιού.



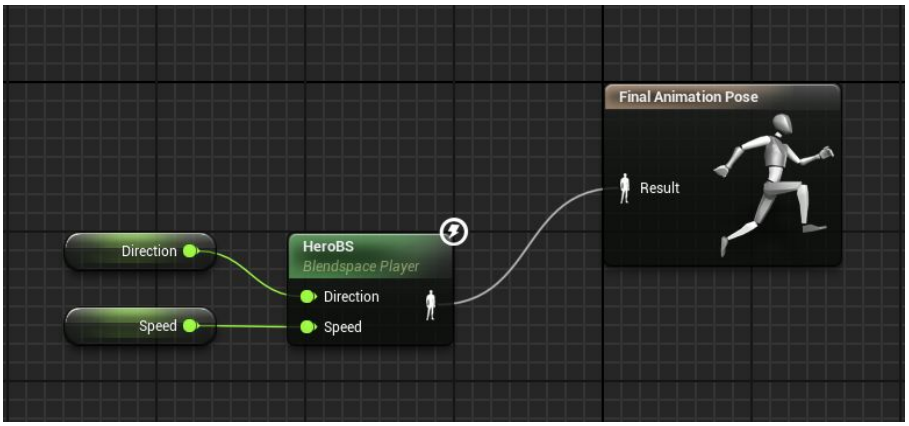
Εικ. 2.20: Το asset που χρησιμοποιήθηκε για το αντικείμενο "Loot"

2.6.2 Δημιουργία Κινήσεων:

Η κίνηση στον τρισδιάστατο χώρο, είναι μια επαναληπτική διαδικασία που δίνει την αίσθηση του πραγματικού στο χρήστη. Κατά την εισαγωγή όλων των assets του χαρακτήρα, υπάρχουν και τα Motion Assets, τα οποία προσομοιώνουν την κίνηση του μοντέλου με αυτή ενός ανθρώπου στο φυσικό επίπεδο. Για να λειτουργήσει όμως σωστά το σύνολο της κίνησης, δηλαδή όχι μόνο το δημιουργικό μέρος (η κίνηση ως στατικό αποτέλεσμα) αλλά και το πρακτικό (η κίνηση μέσα στο χώρο), πρέπει να δημιουργηθεί ένα Animation Blueprint που συνδέει τα assets μεταξύ τους και ενεργοποιεί την επανάληψη στο χώρο.

Δημιουργώντας το Animation Blueprint, επιλέγεται ο σκελετός για τον οποίο θα δημιουργηθεί η κίνηση, και σε αυτή την περίπτωση είναι ο Warrior Skeleton. Κατά την επεξεργασία του Animation Blueprint θα πρέπει να εισαχθούν με λογική σειρά τα assets κίνησης. Όμως, σε κάθε κατάσταση στην οποία βρίσκεται ο χαρακτήρας, μπορεί να εμπεριέχεται μόνο ένα asset κίνησης. Για τον λόγο αυτό, είναι απαραίτητη η κατασκευή διαφόρων καταστάσεων (States), τα οποία συνδέονται μεταξύ τους, το ένα ενεργοποιεί ή

τερματίζει το άλλο και μέσα σε αυτά εμπεριέχονται τα assets στατικής κατάστασης, κίνησης δεξιά, κίνησης αριστερά, κίνησης πίσω δεξιά, κίνησης πίσω αριστερά, κ.α.



Εικ. 2.21: Η συνάρτηση του λόγου "κατεύθυνση προς ταχύτητα" για οποιαδήποτε κατεύθυνση

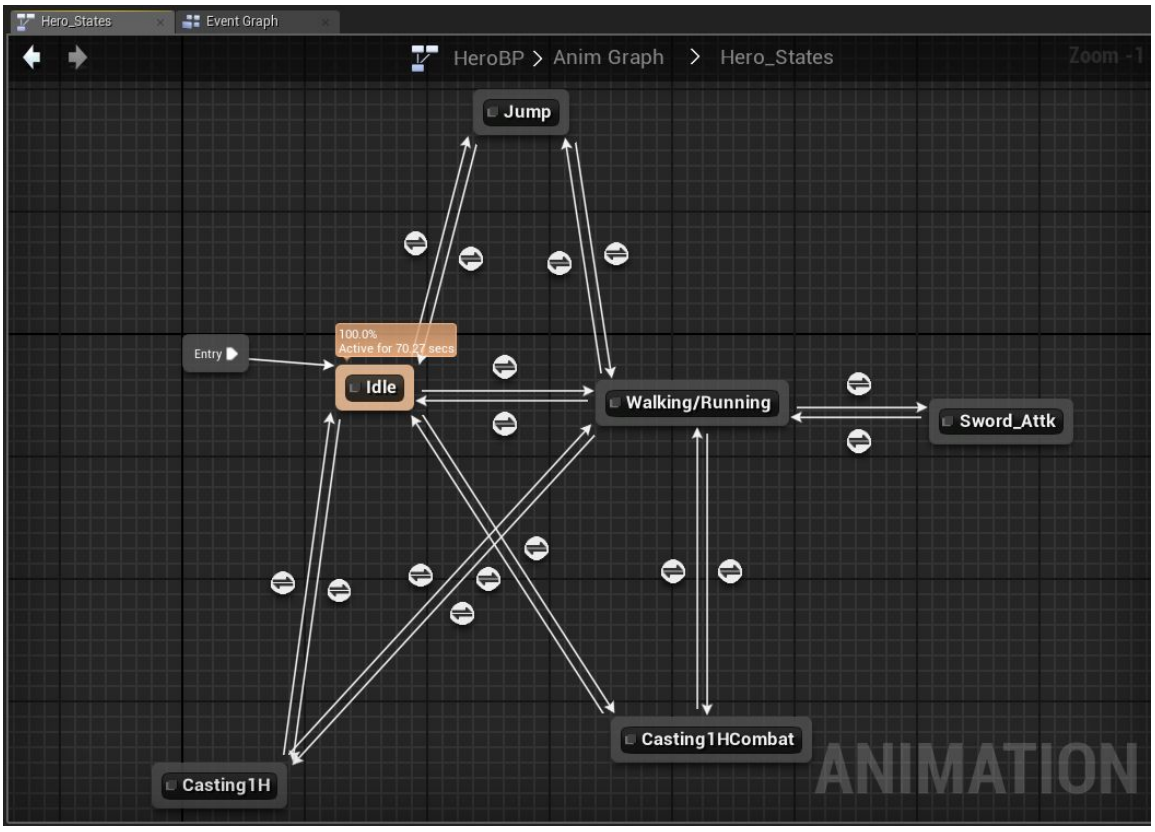
Η πρώτη φάση δημιουργίας της κίνησης, είναι η δημιουργία ενός State Machine, μέσα στο οποίο θα συμπεριληφθούν όλες οι παραπάνω καταστάσεις. Κάνοντας Compile το παιχνίδι, παρατηρείται ότι το State Machine είναι πλέον σε λειτουργία και ο χαρακτήρας βρίσκεται στην προκαθορισμένη κατάσταση (T-pose).

Εφόσον επιλεχθεί το asset που αφορά την στατική κατάσταση του χαρακτήρα, εισάγεται στο Idle state και ο χαρακτήρας πλέον αντί για το T-pose αλληλεπιδρά με το δημιουργικό του Idle-pose που επιλέχθηκε.

Με την ίδια λογική, συνεχίζεται η κατασκευή του Walking/Running State κατά το οποίο θα πρέπει να εισαχθούν τα assets κίνησης που ο χαρακτήρας περπατάει ή τρέχει και θα είναι η λογική συνέχεια του Idle State. Από τα δεδομένα του Walking/Running state επιλέγουμε όλα τα assets που αντιπροσωπεύουν την κίνηση με κατεύθυνση δεξιά, αριστερά, πίσω και μπροστά ώστε ο χαρακτήρας να έχει πλήρη έλεγχο στους άξονες X, Y & Z.

Όπως δημιουργήθηκε το Walking/Running state, θα δημιουργηθεί στη συνέχεια και το Fighting state, όπου ο χαρακτήρας θα ενσωματώσει το Fighting asset, την κίνηση δηλαδή της απλής επίθεσης.

Ολοκληρώνοντας τη δημιουργία του State Machine, τα States από τα οποία απαρτίζεται είναι το Idle State (ο χαρακτήρας μένει ακίνητος), το Walking/Running State (ο χαρακτήρας περπατάει ή τρέχει), το Jump State (ο χαρακτήρας αναπηδά στο χώρο) και τα Sword_Atk, Casting 1H και Casting 1HCombat States (αφορούν την απλή επίθεση, τη σύνθετη επίθεση και την ενέργεια ενδυναμωσης).



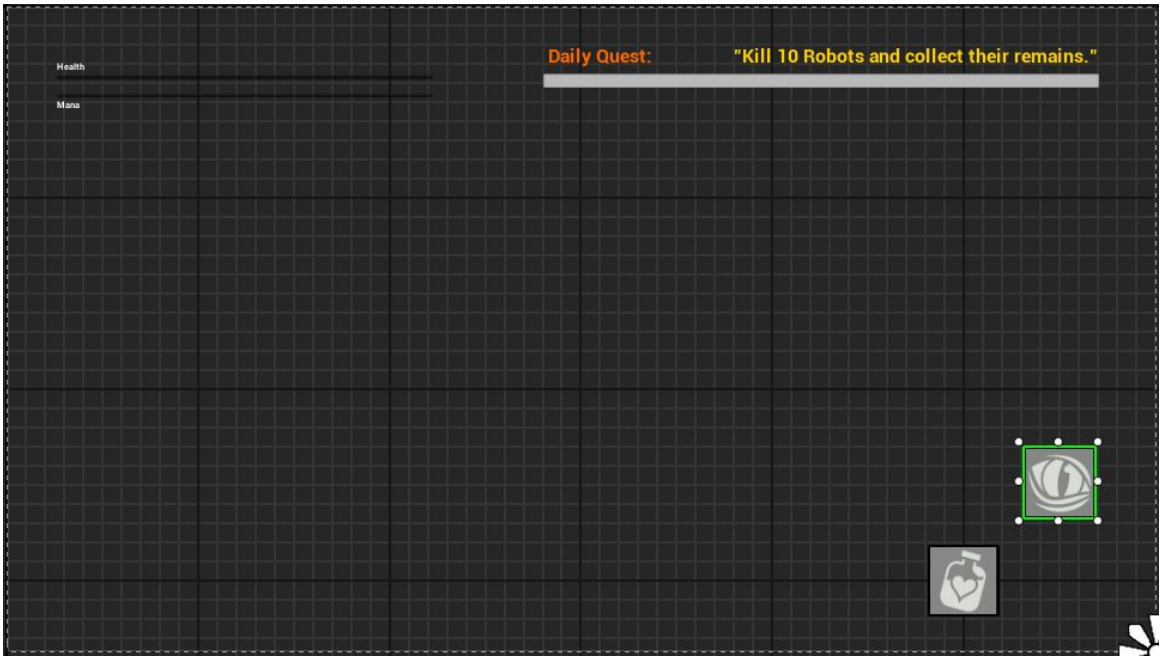
Εικ. 2.22: Το State Machine και η απεικόνιση της αλληλεπίδρασης όλων των καταστάσεων του χαρακτήρα, σε συνάρτηση με τα Animation Assets που αποτελούν την κάθε κατάσταση

2.6.3. Δημιουργία βασικού HUD:

Μια βασική λειτουργία του παιχνιδιού, είναι η μεταφορά σημαντικών πληροφοριών στο χρήστη, με τη μορφή κειμένου, εικόνας ή animation. Πληροφορίες σαν και αυτές είναι η ζωή και η ενέργεια του χαρακτήρα, το όνομα και το επίπεδο ενδυνάμωσης στο οποίο βρίσκεται, και άλλες σημαντικές πληροφορίες που βοηθούν στην κατανόηση του τρισδιάστατου κόσμου αλλά και στην εξέλιξη του παιχνιδιού. Όλα αυτά επιτυγχάνονται, με τη δημιουργία του HUD (Heads Up Display).

Το HUD του παιχνιδιού αποτελείται από δημιουργικά πεδία, τα οποία δίνουν όλη την απαραίτητη πληροφορία στο χρήστη για να χρησιμοποιήσει τον χαρακτήρα και να αλληλεπιδράσει με κάθε οντότητα του παιχνιδιού. Εφόσον ο χαρακτήρας διαθέτει τις μεταβλητές ζωτικών ενδείξεων, θα πρέπει να τις παρουσιάζει στον χρήστη καθ'όλη τη διάρκεια του παιχνιδιού. Έτσι, δημιουργείται σε ένα διάφανο επίπεδο, όλος ο σχεδιασμός του HUD. Σε αυτό το επίπεδο θα κατασκευαστούν μπάρες, τετράγωνα και περιοχές κειμένου, που θα μεταφράσουν κάθε συνδεδεμένη μεταβλητή με τον χαρακτήρα.

Εφόσον επιλεχθεί το σημείο της οθόνης όπου θα πρέπει να μπουν οι πληροφορίες, σειρά έχει η δημιουργία μιας μπαρας η οποία έχει 2 χρωματικούς συνδυασμούς (π.χ. έναν κόκκινο και ένα διάφανο) και λαμβάνει ως παράμετρο τη μεταβλητή του χαρακτήρα (π.χ. Health). Ο διάφανος χρωματικός συνδυασμός απευθύνεται στην απώλεια των ζωτικών ενδείξεων ενώ ο κόκκινος στην πληρότητα. Αυτό έχει ως αποτέλεσμα, ο χρήστης να έχει πλήρη εικόνα κάθε φορά που η ζωτική μεταβλητή Health αυξομειώνεται, και μπορεί να γνωρίζει κατα προσέγγιση ποια είναι τα επίπεδά της.



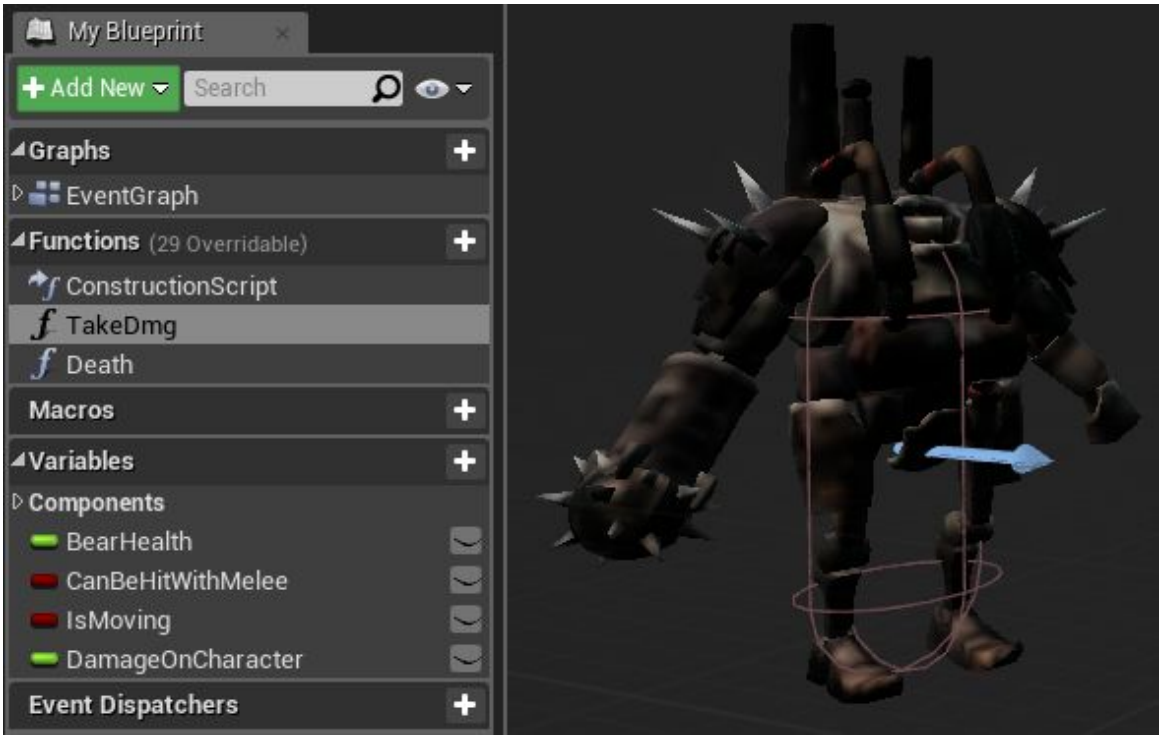
Εικ. 2.23: Το HUD του παιχνιδιού. Συμπεριλαμβάνει τις μπάρες Health, Energy & Loot, το μήνυμα "Daily Quest" που αφορά την αποστολή που πρέπει να εκπληρώσει ο χρήστης και τα εικονίδια των δύο ενεργειών του χαρακτήρα Heal και Beam.

Με τον ίδιο τρόπο, δημιουργούνται 2 τετράγωνα, τα οποία θα συνδεθούν μελλοντικά με την σύνθετη επίθεση και με την ενδυνάμωση του χαρακτήρα, μία μπάρα για τη συλλογή αντικειμένων, και το επιθυμητό μήνυμα που πρέπει να βλέπει ο χρήστης και είθισται να ορίζει κάποια αποστολή που πρέπει να φέρει εις πέρας κατά τη διάρκεια του παιχνιδιού.

2.6.4. Δημιουργία μοντέλων Τεχνητής Νοημοσύνης:

Τα μοντέλα Τεχνητής Νοημοσύνης, είναι αντικείμενα τύπου Actors, όπως και ο βασικός χαρακτήρας, και είναι σχεδιασμένα όπως και ο ίδιος. Κατά μία έννοια το μόνο που τους διαφοροποιεί είναι ότι ο χαρακτήρας υπακούει εξ ορισμού στον χρήστη του παιχνιδιού, ενώ τα μοντέλα όχι.

Η δημιουργία τους είναι απλή, ενώ η παραμετροποίησή τους είναι παρόμοια με αυτή του χαρακτήρα. Όπως για τον χαρακτήρα, έτσι και για τα μοντέλα Τεχνητής Νοημοσύνης, θα πρέπει να δημιουργηθούν μεταβλητές ζωτικών ενδείξεων, να προστεθούν όλα τα απαραίτητα Assets κίνησης, να δημιουργηθεί δικό τους State Machine και να αλληλεπιδράσουν με κάποιο τρόπο με τον χαρακτήρα.



Εικ. 2.24: Το asset του μοντέλου AI και οι μεταβλητές των ζωτικών του σημείων και κινησεων

Η τοποθέτησή τους στον τρισδιάστατο κόσμο είναι αρκετά απλή, ενώ είναι εφικτό να κινούνται όπως εκείνα θέλουν, μέσα σε ορισμένα διαστηματα του επιπέδου. Σημαντικό κομμάτι της αλληλεπίδρασης των μοντέλων με τον χαρακτήρα είναι η εντολή που πρέπει να λάβουν ώστε να τον ακολουθούν και να του επιτίθενται, μόλις ο χαρακτήρας εισέλθει στα εικονικά όρια που έχουν τεθεί.

Η τελική φάση της δημιουργίας των μοντέλων τεχνητής νοημοσύνης, είναι ο σκοπος ύπαρξής τους. Τα ερωτήματα “Γιατί υπάρχουν στον τρισδιάστατο κόσμο;” και “Τι προσφέρουν στο χαρακτήρα και κατ’ επέκταση στον χρήστη;” είναι πολύ σημαντικά να απαντηθούν τόσο πριν όσο και μετά τη δημιουργία τους. Τα μοντέλα αυτά, υπάρχουν στον τρισδιάστατο κόσμο, με σκοπό την ύπαρξη αλληλεπίδρασης χαρακτήρα με μια ουδέτερη οντότητα.

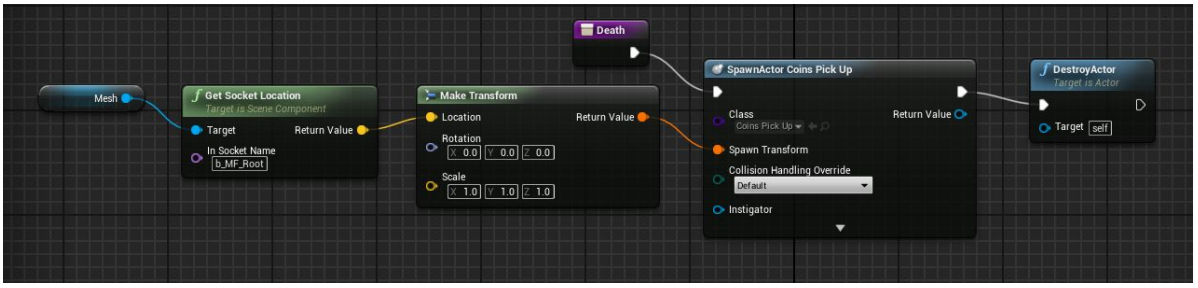
Εικ. 2.25: Το μοντέλο AI ακολουθεί τον χαρακτήρα, όταν αυτός εισέλθει στο οπτικό του πεδίο

Έτσι το παιχνίδι αποκτά ενδιαφέρον για τον χρήστη και τον θέτει σε κατάσταση σκέψης και αντιμετώπισης εικονικών προβληματισμών όπως το να ολοκληρώσει επιτυχώς το επίπεδο. Σε συνέχεια του πρώτου ερωτήματος, η προσφορά των μοντέλων είναι μέγιστη, καθώς ενισχύουν την ποιότητα της αλληλεπίδρασης και δίνουν κάποιο εικονικό αντίτιμο στο χρήστη όταν εκείνος αλληλεπιδρά.

Κατά το τέλος της μάχης του χαρακτήρα απέναντι στα μοντέλα τεχνητής νοημοσύνης και εφόσον εκείνος νικήσει το μοντέλο, εμφανίζεται το αντίτιμο της νίκης. Το αντίτιμο αυτό ή αλλιώς loot, είναι ένα αντικείμενο τύπου actor, το οποίο έχει οριστεί να εμφανίζεται όταν η Health μεταβλητή του μοντέλου φτάσει στο 0.00.

Εικ. 2.26: Το οπτικό πεδίο του μοντέλου AI, στο οποίο αν εισέλθει ο χαρακτήρας, το AI αλληλεπιδρά με αυτόν

Ακόμη, έχει επιλεχθεί για αυτό το αντικείμενο, ένα από τα Assets της Infinity Blade, ωστε να πάρει την επιθυμητή μορφή.



Εικ. 2.27: Το Blueprint όλων των ενεργειών του μοντέλου AI

Στις συναρτήσεις του Blueprint του αντικειμένου, έχει οριστεί η αλληλεπίδρασή του με τον χαρακτήρα, όπου όταν ο χαρακτήρας έρθει σε επαφή με τον ψηφιακό χώρο που βρίσκεται το αντικείμενο, τότε αυτό εξαφανίζεται και στη μεταβλητή Loot του χαρακτήρα προστίθεται η τιμή 1.

2.7. Τρισδιάστατο Επίπεδο - Ενίσχυση Διαδραστικότητας

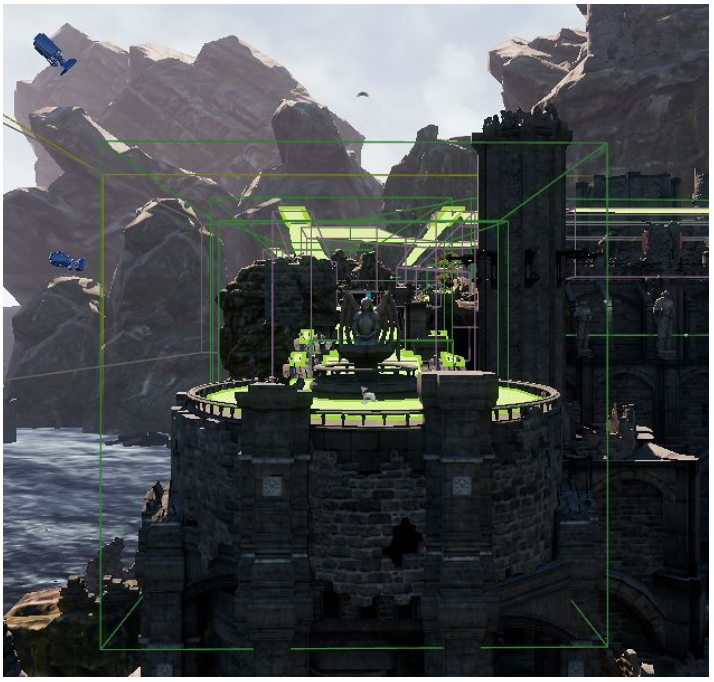
2.7.1. Σχεδιασμός και λειτουργικότητα επιπέδου:

Από τη στιγμή που έχει επιλεχθεί το "Third Person" προσχέδιο για τη δημιουργία του παιχνιδιού, έχει συμπεριληφθεί σε αυτό, το πρότυπο επίπεδο του τρισδιάστατου κόσμου, στο οποίο ο προγραμματιστής επιλέγει κατά την εκκίνηση του έργου, αν θα συμπεριλάβει τα πρότυπα αντικείμενα, μαζί με τα Assets του δαπέδου.

Εφόσον πρόκειται για ένα παιχνίδι που είναι σημαντικό να λειτουργεί σωστά ο χαρακτήρας και τα μοντέλα τεχνητής νοημοσύνης, το πρότυπο επίπεδο που δίνεται αποκτά ρόλο δοκιμαστικού περιβάλλοντος, οπότε είναι άκρως σημαντικό να συμπεριληφθεί στη δημιουργία του παιχνιδιού.

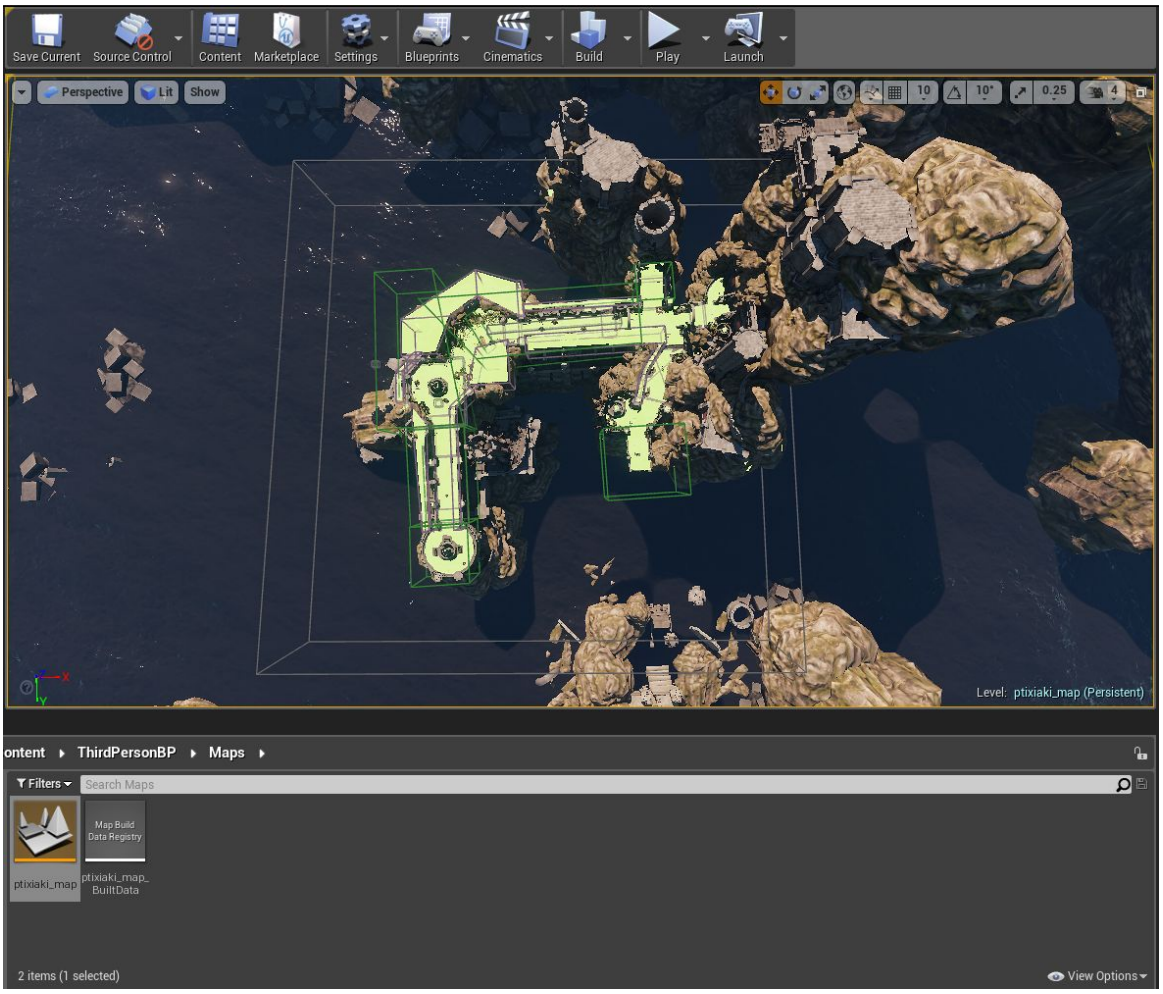
Όταν όλες οι δοκιμές μεταξύ χαρακτήρα και τρίτων οντοτήτων ολοκληρωθούν, έχει σειρά η δημιουργία ενός επιπέδου, που αντιπροσωπεύει τον χώρο που περικλείει οτιδήποτε έχει πρότινος δημιουργηθεί και σχεδιαστεί.

Δημιουργώντας το νέο επίπεδο, υπάρχει η δυνατότητα σχεδιασμού του περιβάλλοντος με τη χρήση της παλέτας των επιπέδων που διαθέτει το Unreal Engine 4. Ο προγραμματιστής, μπορεί να σχεδιάσει ένα πλήρως λειτουργικό περιβάλλον όπως βουνά, λίμνες, δρόμους, κτίρια, κ.α.



Εικ. 2.28: Η δημιουργία του επιπέδου με τη χρήση της παλέτας της Unreal Engine 4 και των "Elven Ruins" assets της Infinity Blade

Μετά τη δημιουργία του χώρου έχει σειρά ο εικαστικός σχεδιασμός του. Κατά τη φάση αυτή της δημιουργίας, υπάρχει η επιλογή χρωματισμού όλων των σημείων, σύμφωνα με τις προτιμήσεις του προγραμματιστή, αλλά υπάρχει και η επιλογή της χρήσης κάποιων Assets περιβάλλοντος, τα οποία είναι μικρά εικονίδια με έτοιμο σχέδιο και μπορούν να χρησιμοποιηθούν σε περιπτώσεις που το δάπεδο θα πρέπει να έχει την όψη του γρασιδιού, του χώματος, του νερού κ.α. Η χρήση αυτών των assets, είναι σημαντική από τη στιγμή που η αίσθηση του ρεαλισμού παίζει σημαντικό παράγοντα στην εμπειρία χρήσης του παιχνιδιού. Η Infinity Blades, διαθέτει εκατοντάδες από αυτά τα Assets, αλλά και έτοιμα level για σκοπούς επίδειξης.



Εικ. 2.29: Η κάτοψη του ολοκληρωμένου βασικού επιπέδου

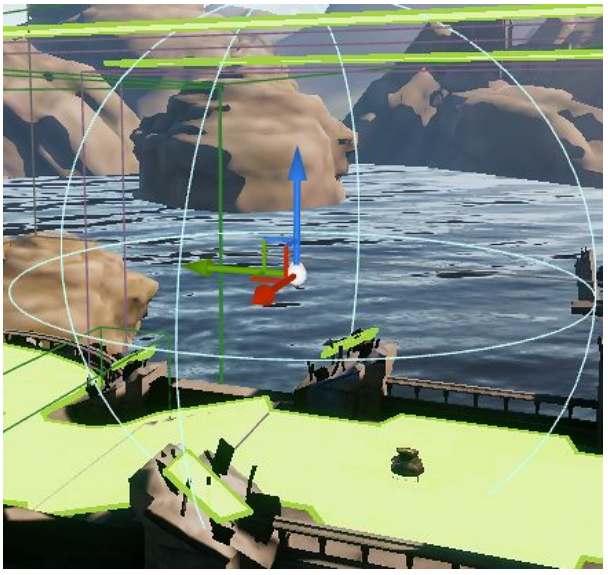
Η τελική φάση της δημιουργίας του επιπέδου είναι ο ορισμός του ως προκαθορισμένο επίπεδο για το παιχνίδι. Από τη στιγμή που ο σχεδιασμός, η δημιουργία και ο χρωματισμός έχουν τελειώσει, όλα τα Blueprints που αφορούν τη λειτουργικότητα του παιχνιδιού, των αντικειμένων, των μοντέλων και του χαρακτήρα πρέπει να αντιγραφούν κάτω από τη φακελική δομή του νέου επιπέδου. Μόλις αυτή η κατάσταση ολοκληρωθεί, τότε το πρωταρχικό επίπεδο μπορεί να διαγραφεί. Σε περιπτώσεις που θα δημιουργηθεί αρχικό μενού ή μενού παύσης, επειδή η δημιουργία του απαιτεί να γίνει σε διαφορετικό

επίπεδο από αυτό του παιχνιδιού, μπορεί το πρωταρχικό επίπεδο να παραμείνει στη φακελική δομή.

2.7.2. Φωτισμός & ήχος επιπέδου:

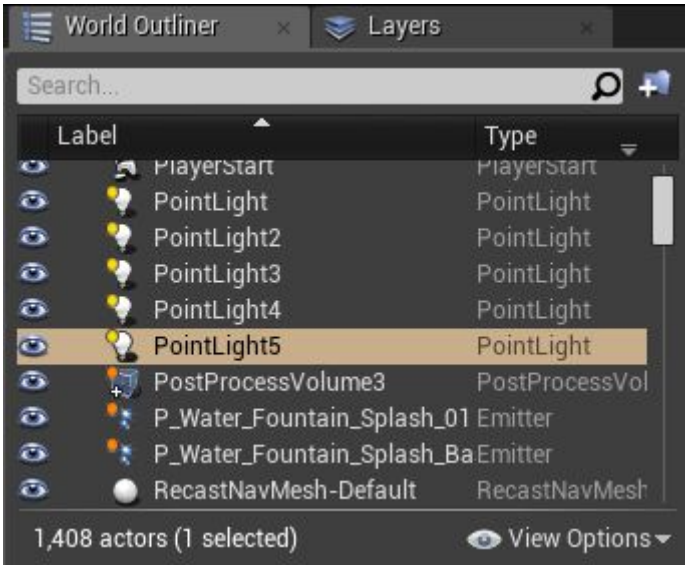
Με το πέρας της δημιουργίας του νέου επιπέδου, ο χαρακτήρας και όλα τα περιεχόμενα του παιχνιδιού, είναι έτοιμα προς χρήση. Παρ'όλα αυτά, επειδή όπως έχει ήδη προαναφερθεί, ο ρεαλισμός που δίνεται μέσω του παιχνιδιού είναι το βασικότερο κομμάτι στη δημιουργία του, η προσθήκη φωτισμού και τα ηχητικά εφέ που μπορούν να εισαχθούν, συμβάλλουν στην επίτευξη αυτού του στόχου.

Ο φωτισμός στο βασικό επίπεδο, υπάρχει ήδη μέσω προκαθορισμένων ρυθμίσεων. Πρόκειται για φυσικό φωτισμό, ο οποίος προέρχεται από στοιχεία της εικονικής φύσης, όπως ο ήλιος, το φεγγάρι ή μια φωτιά, που βρίσκονται μέσα σε αυτό. Βέβαια, η προσπίπτουσες γωνίες του φωτός είναι κάθετες, με αποτέλεσμα να χάνεται η αίσθηση της σκιάς ή η διαβάθμιση του ίδιου του φωτός. Αυτό δίνει τη δυνατότητα στον προγραμματιστή, να συμπεριλάβει τμήματα φωτός μέσα στο επίπεδο, τα οποία παραμένουν στατικά, ενώ το χρώμα τους μπορεί να είναι οποιοδήποτε ανάμεσα στο φάσμα του φωτός. Κατά συνέπεια, προστέθηκαν συγκεκριμένες δέσμες φωτός σε μερικά σημεία του επιπέδου, ώστε να δώσουν έμφαση στον τρόπο που φαίνεται το ίδιο το επίπεδο, αλλά και ο χαρακτήρας ή τα μοντέλα μέσα σε αυτό και υπάρχουν με σκοπό τη δημιουργία αντίθεσης όταν επί παραδείγματι, ο χαρακτήρας περνάει δίπλα από ένα μνημείο, που έχει στην αντίθετη πλευρά το φως του ήλιου, άρα δημιουργείται η σκιά του χαρακτήρα.



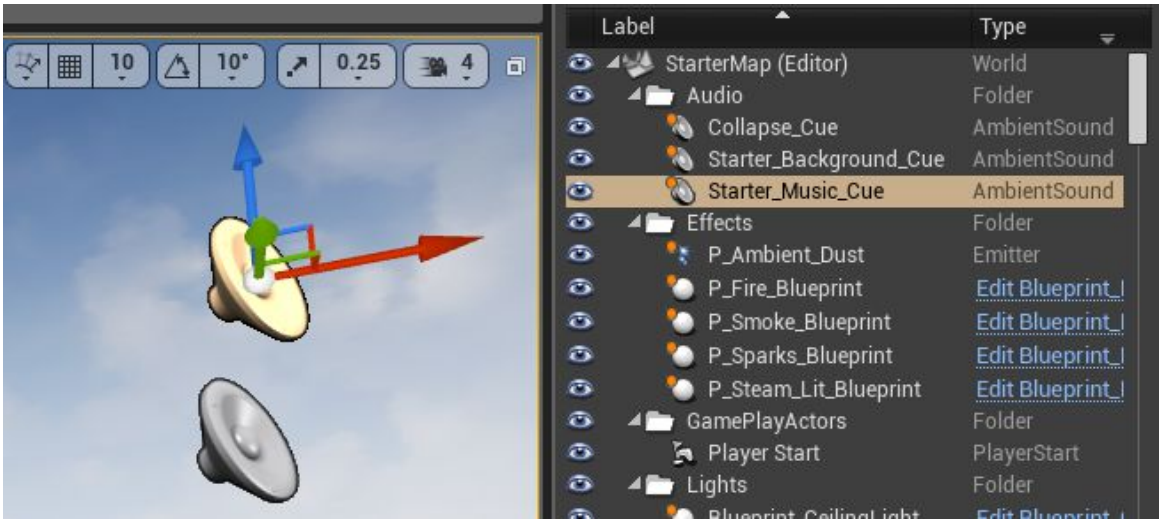
Εικ. 2.30: Σημείο στατικού φωτισμού στο βασικό επίπεδο. Το εύρος που καλύπτει δίνεται από τον προγραμματιστή

Επίσης, ο φωτισμός μπορεί να ενσωματωθεί σε στατικά ή κινούμενα αντικείμενα που υπάρχουν μέσα στο επίπεδο, όπως δάδες φωτιάς, εξωτερικά φώτα ή στα μοντέλα τεχνητής νοημοσύνης. Αυτά εξ αρχής διαθέτουν μόνο τις χρωματικές αντιθέσεις, που δίνουν την αίσθηση του πραγματικού κατά το ήμισυ. Προσθέτοντας λοιπόν, τεχνητό φως στο χώρο συμπληρώνεται και το άλλο τους μισό.



Εικ. 2.31: Όλα τα τεχνητά φώτα τύπου PointLight(φως σημείου) μέσα στο βασικό επίπεδο

Όπως ο φωτισμός, έτσι και ο ήχος, είναι ένα φυσικό μέσο που μπορεί να συμβάλλει στο ρεαλισμό του παιχνιδιού. Η διαφορά τους είναι, πως ο ήχος δεν προϋπάρχει στα πρότυπα μέσα, όπως το φως. Φυσικά, μπορεί να εισαχθεί με τρόπο εύκολο και μεθοδικό, σε όλο το φάσμα του επιπέδου και των οντοτήτων του.



Εικ. 2.32: Σημείο ήχου που αντιπροσωπεύει το ηχητικό κλιπ κατά την έναρξη του παιχνιδιού

Ο ήχος επικρατεί κυρίως κατά το βάδισμα του χαρακτήρα, κατά την επίθεσή του, κατά την κίνηση των φυσικών στοιχείων του επιπέδου όπως η θάλασσα ή η βροχή, και

φυσικά μπορεί να εισαχθεί σε ενορχηστρωμένη μορφή, σε συνδυασμό με τους φυσικούς ήχους. Στη συγκεκριμένη εργασία, κρίθηκε απαραίτητο να προστεθούν μόνο φυσικοί ήχοι, μιας και η χρήση ενορχηστρωμένης μορφής ήχου δεν θα εξυπηρετούσε στην επίδειξή της.

2.8. Δημιουργία Περιήγησης

2.8.1. Αρχική Περιήγηση:

Η αρχική περιήγηση, αφορά το μενού με βάση το οποίο ο χρήστης μπορεί να ξεκινήσει το βασικό επίπεδο του παιχνιδιού, να διαμορφώσει τις εκάστοτε ρυθμίσεις που δίνονται, να δει μερικές πληροφορίες για τους δημιουργούς του παιχνιδιού ή να το τερματίσει. Φυσικά σε αυτό το μενού, μπορεί να μπει οποιαδήποτε επιλογή επιθυμεί ο προγραμματιστής, ενώ η επιλογή αυτή πρέπει να επηρεάζει τον τρόπο παιχνιδιού του χρήστη. Για την κατασκευή του αρχικού μενού, θα χρειαστεί να δημιουργηθεί ένα νέο Blueprint και μια κλάση ίδια με αυτή του υπάρχοντος HUD. Το μενού στην πραγματικότητα, είναι ένα HUD, το οποίο δημιουργείται σε ένα δευτερεύων επίπεδο και το επίπεδο αυτό ενεργοποιείται με την εκκίνηση του παιχνιδιού.

Η βασική λειτουργικότητα του Menu HUD είναι η σύνδεση του δημιουργικού “Start Game” με τη συνάρτηση που μεταφέρει την έξοδο του προγράμματος, από το δευτερεύων επίπεδο και την είσοδο στο βασικό. Αυτός είναι και ο μόνος τρόπος να ξεκινήσει το βασικό επίπεδο. Με παρόμοιο τρόπο, κάθε δημιουργικό όπως το “Options” ή το “Credits” συνδέονται με μια συνάρτηση που καλεί κάτι διαφορετικό.

Εικ. 2.33: Δομή Blueprint Αρχικής Περιήγησης

Η δομή του αρχικού μενού, πρέπει να περιλαμβάνει τουλάχιστον δύο λειτουργικότητες. Αυτή που ξεκινάει το παιχνίδι και αυτή που το τερματίζει. Το συγκεκριμένο αρχικό μενού περιλαμβάνει επιπλέον τη λειτουργικότητα "Options", μέσα από την οποία, ο χρήστης μεταφέρεται σε ένα εσωτερικό μενού για να αλλάξει την ανάλυση του παιχνιδιού. Οι εντολές ανάλυσης της οθόνης παρέχονται μέσω έτοιμων μακροεντολών που δίνει η Unreal Engine 4. Επίσης περιλαμβάνει και τη λειτουργικότητα "Credits", η οποία εμφανίζει ένα κείμενο σχετικά με τον συγγραφέα της εργασίας, τις ευχαριστίες, κ.α.

Εικ. 2.34: Το HUD του αρχικού μενού, μαζί με όλες τις συναρτήσεις, δημιουργικά και πληροφορίες που περιέχει. Κάθε ένα από αυτά εμφανίζεται επιλεκτικά αναλόγως με το σημείο στο οποίο βρίσκεται ο χρήστης

2.8.2. Παύση του παιχνιδιού:

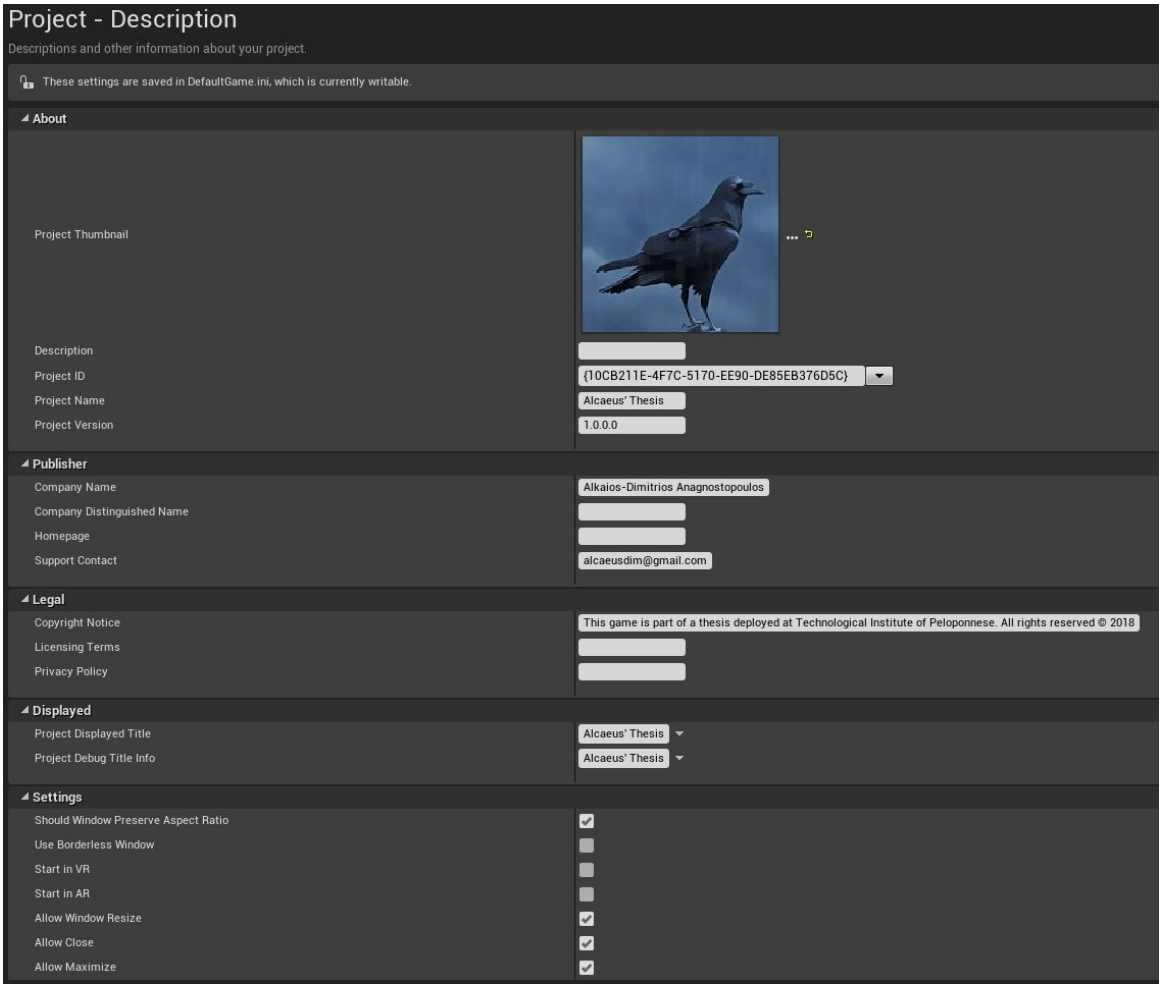
Η παύση του παιχνιδιού, επιτυγχάνεται με τρόπο ίδιο όπως και αυτός της αρχικής περιήγησης. Κατά την αλληλεπίδραση του χρήστη με τον χαρακτήρα, δίνεται η επιλογή πληκτρολογώντας ένα συγκεκριμένο πλήκτρο που έχει οριστεί, να σταματήσει η περιήγησή του στο βασικό επίπεδο και να μεταφερθεί σε ένα ξεχωριστό επίπεδο, το οποίο θα περιλαμβάνει επιλογές όπως αυτές του αρχικού μενού. Η διαφορά τους είναι πως στο μενού παύσης, συνηθίζεται να υπάρχει η επιλογή "Resume" ώστε να συνεχιστεί το παιχνίδι, αντί του "Start Game", που ξεκινά το παιχνίδι από την αρχή. Επειδή στην παρούσα εργασία, η χρήση του παιχνιδιού έγινε με το Arduino ως Gamepad, δεν δημιουργήθηκε μενού παύσης καθώς δεν υπήρξε κάποιο ελεύθερο πλήκτρο στο Joystick Shield.

2.9. Packaging

2.9.1. Η διαδικασία του Packaging:

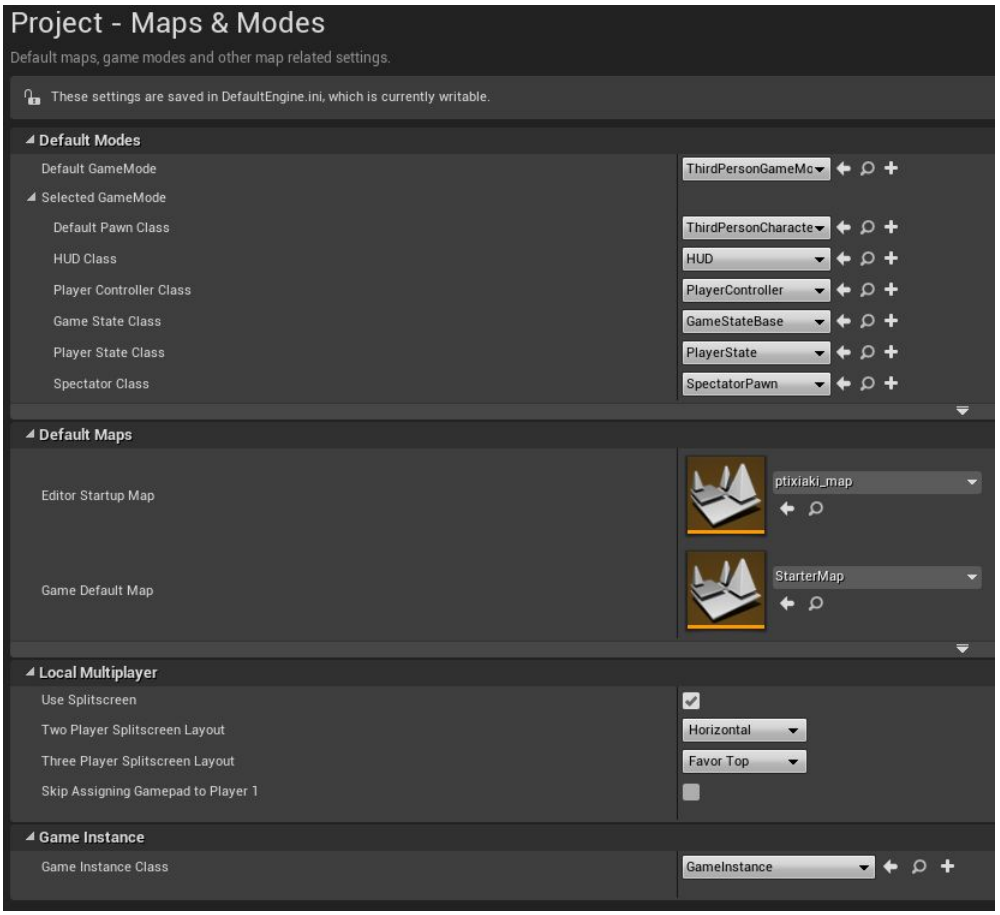
Για να προβεί ο προγραμματιστής στη φάση του Packaging, συνεπάγεται πως όλο το παιχνίδι έχει δημιουργηθεί και δοκιμαστεί, ενώ παράλληλα καλύπτονται όλες οι ανάγκες του χρήστη. Βέβαια, είναι απαραίτητο να γίνουν κάποιες τελευταίες, αλλά κρίσιμες ρυθμίσεις πριν το Packaging.

Στις ρυθμίσεις του υπάρχοντος Project, ο προγραμματιστής θα βρει τον τομέα της περιγραφής του παιχνιδιού. Εκεί θα πρέπει να συμπληρωθούν κάποιες βασικές πληροφορίες σχετικές με το παιχνίδι όπως η περιγραφή του, ο δημιουργός του, η έκδοση στην οποία βρίσκεται, η ηλεκτρονική διεύθυνση στην οποία ο χρήστης μπορεί να επικοινωνήσει για θέματα υποστήριξης, ένα σημείωμα σχετικό με τα πνευματικά δικαιώματα του παιχνιδιού, το εικονίδιο του και η επιλογή αλλαγής της ανάλυσης οθόνης του παιχνιδιού, από τον χρήστη.



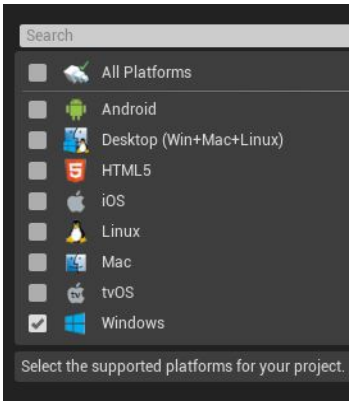
Εικ. 2.35: Ρυθμίσεις της περιγραφής του παιχνιδιού πριν το Packaging

Στη συνέχεια, θα πρέπει να οριστεί το βασικό επίπεδο το οποίο θα ενεργοποιείται κατά την εκκίνηση του παιχνιδιού (επίπεδο αρχικού μενού), το επίπεδο από το οποίο η μηχανή αντλεί όλες τις λειτουργικές πληροφορίες για το παιχνίδι (βασικό μενού), την κλάση του βασικού HUD και το Blueprint του βασικού χαρακτήρα.



Εικ. 2.36: Ρυθμίσεις σχετικά με το αρχικό επίπεδο, τον χαρακτήρα και τις αρχές λειτουργίας του παιχνιδιού

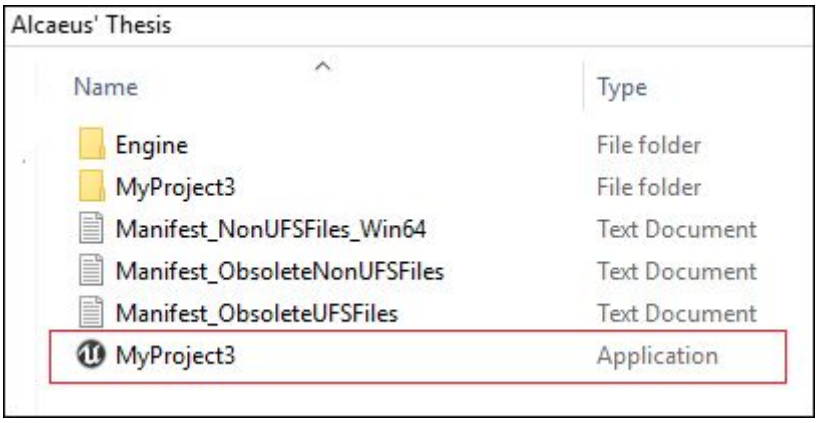
Τέλος είναι σημαντικό να οριστούν οι πλατφόρμες τις οποίες υποστηρίζει το παιχνίδι. Στη συγκεκριμένη περίπτωση υποστηρίζεται μόνο το λειτουργικό σύστημα των Windows.



Εικ. 2.37: Ρυθμίσεις σχετικά με τις πλατφόρμες που υποστηρίζει το παιχνίδι

2.9.2. Εκτελέσιμο αρχείο .exe:

Μόλις τελειώσει η διαδικασία του Packaging, ο προγραμματιστής μπορεί να ελέγξει από την οθόνη της Unreal Engine 4, για μηνύματα σχετικά με σφάλματα του παιχνιδιού ή της διαδικασίας του Packaging. Εφόσον δεν υπάρχουν σφάλματα, τότε στη διαδρομή που έχει ορίσει ο προγραμματιστής, θα εμφανιστεί η δομή του παιχνιδιού που περιλαμβάνει όλα τα assets και τις ρυθμίσεις που χρησιμοποιήθηκαν, όλα τα source files που εμπεριέχονται στα Blueprints του παιχνιδιού(σε μορφή C++) καθώς και το εκτελέσιμο αρχείο του παιχνιδιού (.exe).



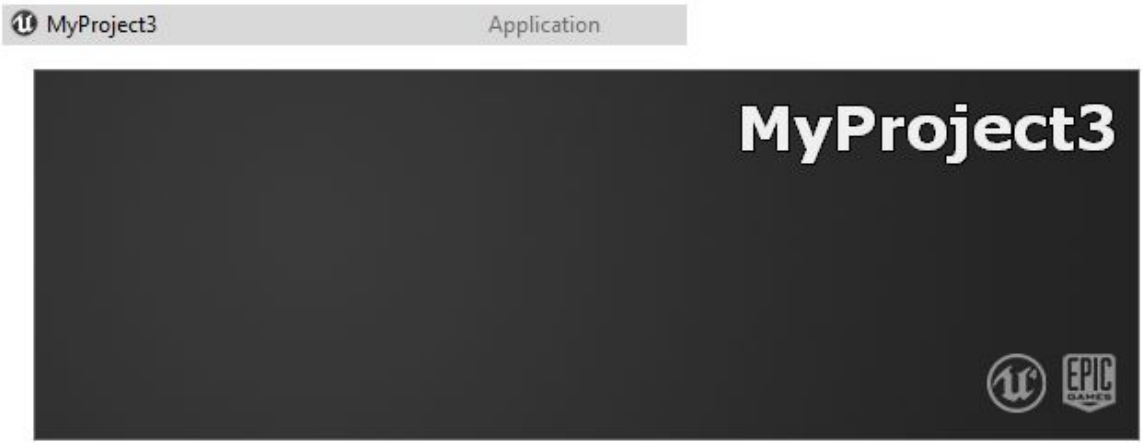
Εικ. 2.38: Η παράγωγη φακελική δομή της διαδικασίας του Packaging

Πλέον το παιχνίδι είναι έτοιμο προς χρήση μέσω του ***MyProject3.exe***, χωρίς την ανάγκη της Unreal Engine 4 και σε οποιοδήποτε Η/Υ με λειτουργικό σύστημα Windows.

2.10. Εκτέλεση εφαρμογής και χειρισμός του μοντέλου με τη χρήση του Arduino Leonardo

2.10.1. Εκτέλεση παιχνιδιού:

Το τελευταίο στάδιο της υλοποίησης του παιχνιδιού είναι η εκτέλεση του και η περιήγηση στον κόσμο που δημιουργήθηκε.



Εικ. 2.39: Εκτέλεση παιχνιδιού

Ίσως είναι και το πιο σημαντικό στάδιο του συνολικού έργου, καθώς ο προγραμματιστής παίρνει τη θέση του χρήστη και αλληλεπιδρά με τον τρισδιάστατο κόσμο και το βασικό χαρακτήρα που δημιούργησε.

Αυτή η αλληλεπίδραση, θα επισημάνει αλλαγές ή διορθώσεις που θα χρειαστεί να γίνουν άμεσα ή σε μια μεταγενέστερη έκδοση του παιχνιδιού.

2.10.2. Χρήση του Gamepad:

Ταυτόχρονα με την εκτέλεση του παιχνιδιού, μπορεί να γίνει και η τελική δοκιμή του Arduino Leonardo ως Gamepad. Θεωρείται δεδομένο, πως εφόσον το Arduino, αναγνωρίζεται από τον Η/Υ ως Gamepad και παράλληλα έχει ρυθμιστεί κατάλληλα, με βάση τις παραμετροποιήσεις που χρησιμοποιήθηκαν στην Unreal Engine 4, θα λειτουργήσει απευθείας μετά την έναρξη του παιχνιδιού.

Εικ. 2.40: Εκτέλεση παιχνιδιού και χειρισμός χαρακτήρα με τη χρήση του Arduino Leonardo και του Arduino Joystick Shield

Η ορθή λειτουργία των αξόνων X και Y για την μετακίνηση του χαρακτήρα, όπως και οι επιθέσεις ή ενέργειες μέσω των πλήκτρων, εφόσον λειτουργούν σωστά, επισημαίνουν και το τέλος του έργου. Στην περίπτωση βέβαια που κάποιο πλήκτρο δεν εξυπηρετεί τον σκοπό του, ο προγραμματιστής μπορεί πάντα να επιστρέψει στον προγραμματισμό του Arduino, και στην θέτηση των κατάλληλων πλήκτρων στο περιβάλλον της Unreal Engine 4.

ΚΕΦΑΛΑΙΟ 3ο : ΣΥΜΠΕΡΑΣΜΑΤΑ

Κατά την ανάπτυξη του παιχνιδιού, έγινε κατανοητό πως οι δυνατότητες της Unreal Engine 4, μπορούν να εκπληρώσουν το σχεδιασμό του παιχνιδιού στο μέγιστο. Το μεγάλο πλεονέκτημά της είναι πως το εκάστοτε παιχνίδι μπορεί να υλοποιηθεί ταυτόχρονα για διαφορετικές πλατφόρμες, ενώ μέσα από την κοινότητα της μηχανής, ο προγραμματιστής μπορεί να αντλήσει πληροφορίες που θα τον βοηθήσουν με προβλήματα που θα προκύψουν κατά τη διάρκεια της υλοποίησης.

Είναι επίσης σημαντικό να αναφερθεί, πως οι απαιτήσεις του συστήματος από την πλευρά του χρήστη, καθορίζονται από τις ρυθμίσεις που θα παραμετροποιήσει ο προγραμματιστής πριν το Packaging, αλλά και από την ποιότητα των Assets που θα χρησιμοποιήσει. Φυσικά, δεν μπορεί να παραληφθεί, η άμεση ενσωμάτωση του Arduino ως Gamepad, σε όλες τις φάσεις υλοποίησης του παιχνιδιού.

Η επιλογή του Arduino Leonardo ήταν καθοριστική, αφού τόσο ο προγραμματισμός του, όσο και η ευελιξία του στο κομμάτι της επίλυσης προβλημάτων που προέκυψαν, ήταν κερδοφόρα ως προς το χρόνο υλοποίησης του συνολικού έργου.

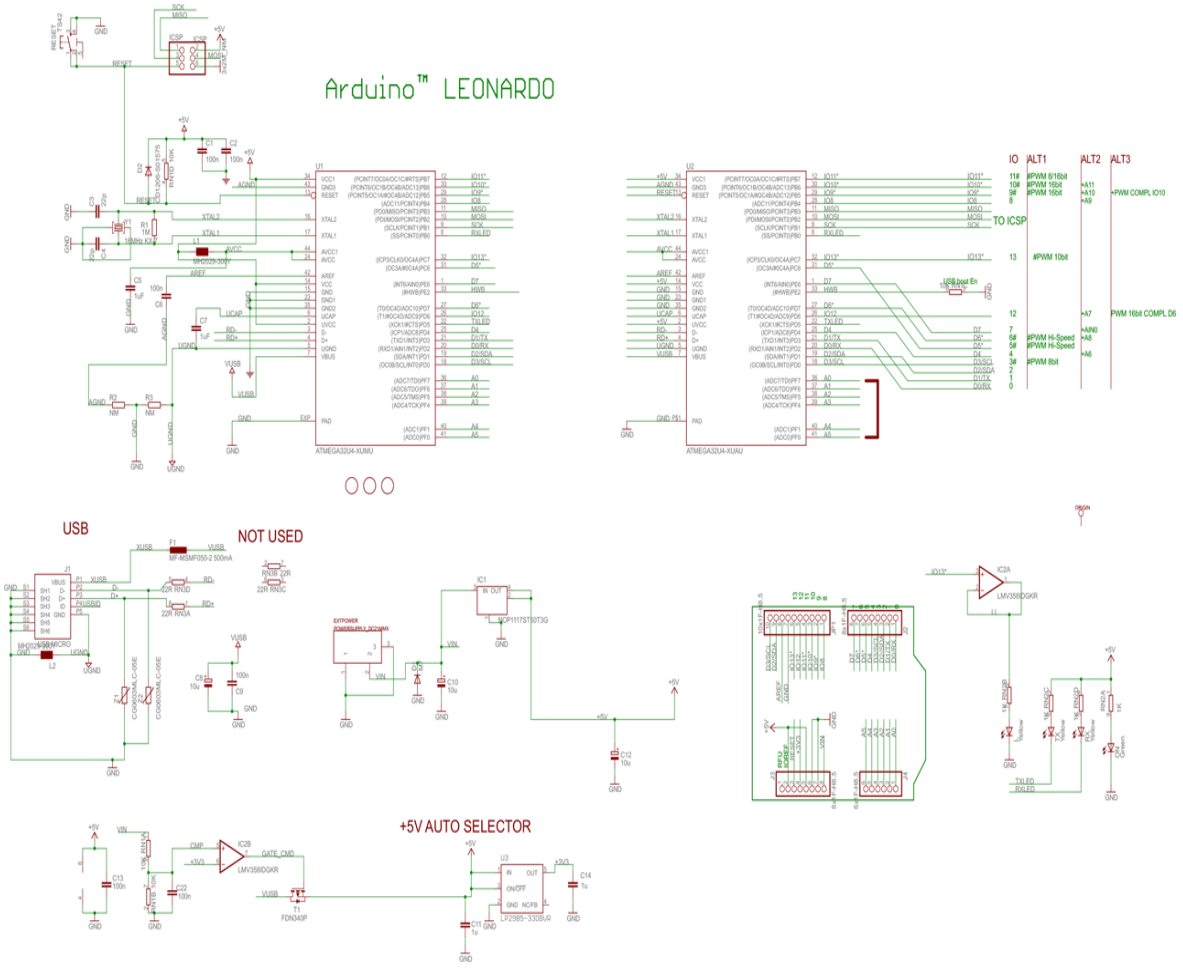
Όπως με την Unreal Engine 4, έτσι και με το Arduino, η ανοιχτού λογισμικού κοινότητα που το υποστηρίζει, αποτελεί πηγή γνώσης γύρω από τον προγραμματισμό και τις δυνατότητες που έχει ο εκάστοτε μικροελεγκτής, ενώ παράλληλα, προσφέρει επεξηγήσεις γύρω από τα έτοιμα παραδείγματά του.

ΚΕΦΑΛΑΙΟ 4ο : ΤΕΧΝΙΚΑ ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ

4.1. Arduino

4.1.1. Arduino Leonardo Schematic:

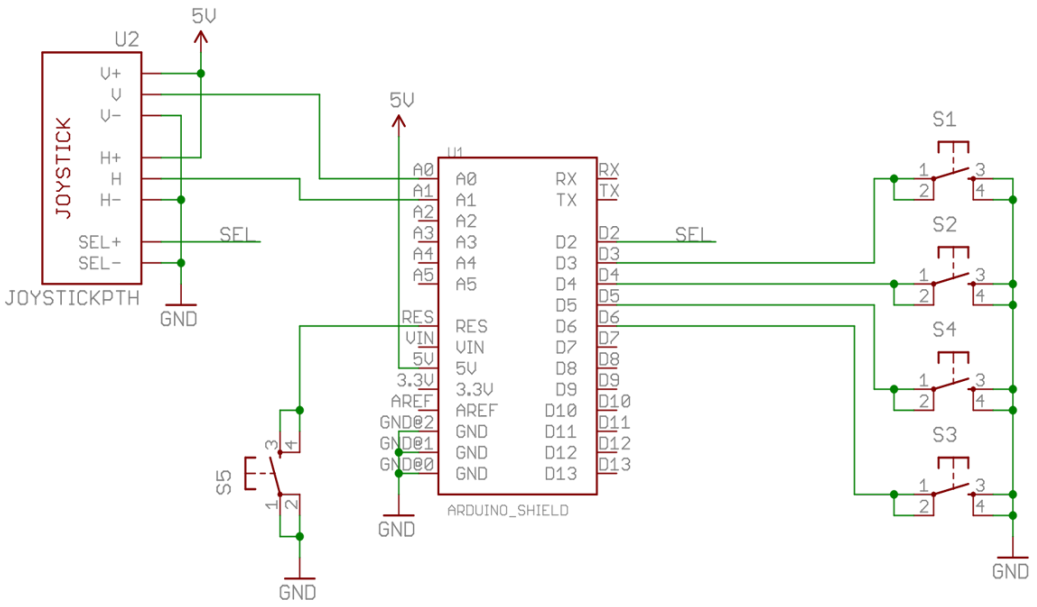
Η σχηματική απεικόνιση του μικροελεγκτή Arduino Leonardo:



Εικ. 4.1: Το Schematic του Arduino Leonardo

4.1.2. Joystick Shield Kit Schematic:

Η σχηματική απεικόνιση του Arduino Joystick Shield:



Εικ. 4.2: Το Schematic του Sparkfun Joystick Shield

4.2. Unreal Engine 4

4.2.1. Απαιτήσεις συστήματος:

Οι απαιτήσεις υλικού και λογισμικού του συστήματος, ανά λειτουργικό σύστημα:

Windows	
Προτεινόμενες Απαιτήσεις Υλικού	
Έκδοση Λειτουργικού Συστήματος	Windows 10 64-bit
Επεξεργαστής	Quad-core Intel ή AMD, 2.5 GHz ή ταχύτερος
Μνήμη	8 GB RAM
Κάρτα Γραφικών/Εκδοση DirectX	Οποιαδήποτε κάρτα συμβατή με την έκδοση DirectX 11

Ελάχιστες Απαιτήσεις Λογισμικού	
Έκδοση Λειτουργικού Συστήματος	Windows 10 64-bit
DirectX Runtime	DirectX End-User Runtimes (June 2010)
Visual Studio Version	● Visual Studio 2015 Update 3 ● Visual Studio 2017 v15.6 ή νεότερη (προτεινόμενη)

Πίνακας 4.1: Ελάχιστες απαιτήσεις συστήματος λειτουργικού Windows

MacOS	
Προτεινόμενες Απαιτήσεις Υλικού	
Έκδοση Λειτουργικού Συστήματος	macOS 10.13.5 High Sierra
Επεξεργαστής	Quad-core Intel, 2.5 GHz ή ταχύτερος
Μνήμη	8 GB RAM

Κάρτα Γραφικών	Metal 1.2 Συμβατή κάρτα γραφικών

Ελάχιστες Απαιτήσεις Λογισμικού	
Έκδοση Λειτουργικού Συστήματος	macOS 10.13.5 High Sierra
Xcode Version	9.4

Πίνακας 4.2: Ελάχιστες απαιτήσεις συστήματος λειτουργικού MacOS

Linux	
Προτεινόμενες Απαιτήσεις Υλικού	
Έκδοση Λειτουργικού Συστήματος	Ubuntu 18.04
Επεξεργαστής	Quad-core Intel ή AMD, 2.5 GHz ή ταχύτερος
Μνήμη	32 GB RAM
Κάρτα Γραφικών	NVIDIA GeForce 960 GTX ή μεγαλύτερη

Ελάχιστες Απαιτήσεις Λογισμικού	
Έκδοση Λειτουργικού Συστήματος	Οποιαδήποτε έκδοση Linux από την CentOS 7.x και πάνω
Linux Kernel Version	kernel 3.x ή νεότερη
Πρόσθετα	glibc 2.17 ή νεότερο
Μεταγλωττιστής	clang 5.0
Επεξεργαστής κώδικα	Visual Studio Code, CLion, QtCreator

Πίνακας 4.3: Ελάχιστες απαιτήσεις συστήματος λειτουργικού Linux

4.2.2. Διαδικασίες Update και Downgrade της Unreal Engine 4:

Η διαδικασία Update αφορά την αναβάθμιση της τρέχουσας έκδοσης της Unreal Engine 4. Κάθε νέα έκδοση που κυκλοφορεί, περιλαμβάνει διορθώσεις σφαλμάτων από προηγούμενες εκδόσεις ή κάποια νέα τεχνική βελτίωση. Αυτή η διαδικασία, είναι άκρως σημαντική για την εκσυγχρόνιση της μηχανής, ιδιαιτέρως από προγραμματιστές που χρησιμοποιούν συνεχώς την Unreal Engine 4 για τη δημιουργία παιχνιδιών και εφαρμογών.

Παρόλα αυτά, η αναβάθμιση της μηχανής στη νεότερη έκδοση, μπορεί να δημιουργήσει προβλήματα σε κάποιο υπάρχον έργο. Τα προβλήματα που μπορεί να εμφανιστούν είναι συνήθως σε πρόσθετες επεκτάσεις της μηχανής, όπως το UE4Duino, τα οποία δεν υποστηρίζουν ακόμη τη νέα αυτή έκδοση. Επίσης, παρουσιάζονται συχνά προβλήματα στην εκκίνηση του project μέσω της νέας έκδοσης, καθώς τα ο κώδικας στα πηγαία αρχεία, έχει κατασκευαστεί με βάση μια παλαιότερη έκδοση. Στην περίπτωση που προκύψει πρόβλημα με την εκκίνηση, συνίσταται να ανοίξει το παιχνίδι με την έκδοση την οποία κατασκευάστηκε. Να γίνει δηλαδή η διαδικασία Downgrade στην έκδοση της Unreal Engine 4.

Η ίδια η Epic Games, αλλά και η κοινότητα της Unreal Engine 4, συμβουλεύουν τους προγραμματιστές να μην επηρεάζουν τις εκδόσεις ενός παιχνιδιού, όσο αυτό βρίσκεται υπό κατασκευή. Έτσι μπορούν να αποφευχθούν κρίσιμα προβλήματα που επηρεάζουν τόσο το συνολικό αποτέλεσμα, όσο και τον χρόνο υλοποίησης, λόγω επίλυσής τους.

ΕΥΡΕΤΗΡΙΟ ΕΙΚΟΝΩΝ & ΠΙΝΑΚΩΝ

Μέρος Α΄ - Εικόνες:

Μέρος Β΄ - Πίνακες:

ΒΙΒΛΙΟΓΡΑΦΙΑ

Ιστότοποι:

- ➔ https://www.arduino.cc
- ➔ https://www.arduino.cc/en/Main/Software
- ➔ https://www.arduino.cc/en/Main/Education
- ➔ https://www.arduino.cc/en/Tutorial/HomePage
- ➔ https://forum.arduino.cc/
- ➔ https://www.sparkfun.com/tutorials/161
- ➔ https://www.sparkfun.com/tutorials/171
- ➔ https://supertuxkart.net/Main_Page
- ➔ https://docs.unrealengine.com/en-us/
- ➔ https://forums.unrealengine.com/
- ➔ https://wiki.unrealengine.com/Main_Page
- ➔ https://www.unrealengine.com/marketplace/infinity-blade-characters
- ➔ https://www.unrealengine.com/marketplace/infinity-blade-enemies
- ➔ https://www.unrealengine.com/marketplace/infinity-blade-effects
- ➔ https://www.unrealengine.com/marketplace/infinity-blade-weapons
- ➔ https://www.unrealengine.com/marketplace/infinity-blade-plain-lands
- ➔ https://www.mixamo.com/
- ➔ https://en.wikipedia.org/wiki/Unreal_Engine
- ➔ https://en.wikipedia.org/wiki/Arduino