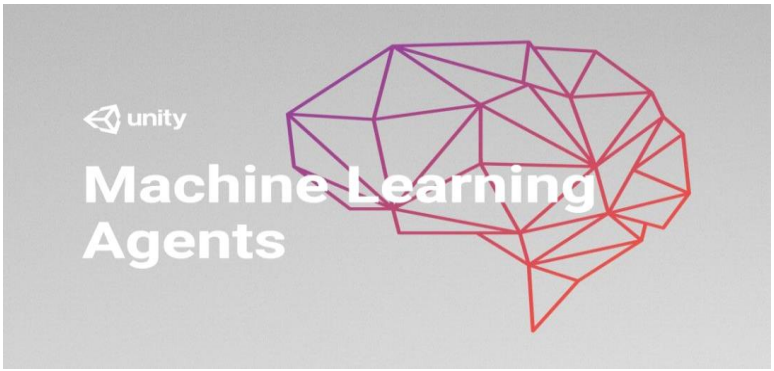
Πανεπιστήμιο Πελοποννήσου
Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΤΙΤΛΟΣ

GAME DESIGN WITH UNITY USING MACHINE LEARNING AGENTS



Ονοματεπώνυμο: Πανανός Μιχαήλ
Αριθμός Μητρώου: 2892

Επιβλέπων Καθηγητής: Σωτήριος Χριστοδούλου

Πάτρα, Οκτώβριος 2022

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΤΙΤΛΟΣ

ΑΝΑΠΤΥΞΗ ΠΑΙΧΝΙΔΙΟΥ ΜΕ UNITY ΧΡΗΣΙΜΟΠΟΙΩΝΤΑΣ ΕΥΦΥΕΙΣ ΠΡΑΚΤΟΡΕΣ

Ονοματεπώνυμο: Πανανός Μιχαήλ
Αριθμός Μητρώου: 2892

Επιβλέπων Καθηγητής: Σωτήριος Χριστοδούλου

Πάτρα, Οκτώβριος 2022

Πρόλογος

Η παρούσα πτυχιακή εργασία ασχολείται με τη ανάπτυξη ενός παιχνιδιού χρησιμοποιώντας την μηχανή ανάπτυξης παιχνιδιών Unity3D και τους ML-Agents (Machine Learning Agents) που ενσωματώνει.

Η Unity3D είναι μία από τις δημοφιλέστερες μηχανές ανάπτυξης παιχνιδιών καθώς είναι πολύ φιλική προς τον χρήστη όσον αφορά το περιβάλλον ανάπτυξης που διαθέτει. Είναι αρκετά εύκολη για άτομα που μπαίνουν για πρώτη φορά στο χώρο του Game Developing και πολύ ισχυρή για τους ειδικούς. Έχει πάρα πολλές δυνατότητες και είναι ικανή για να πραγματοποιήσει οποιαδήποτε ιδέα.

Οι ML-Agents είναι ένα ανοιχτού κώδικα έργο που επιτρέπει στα παιχνίδια να λειτουργούν σαν περιβάλλοντα για την εκπαίδευση ευφυών πρακτόρων. Τέτοιοι πράκτορες μπορούν να εκπαιδευτούν με διάφορες τεχνικές υπολογιστικής νοημοσύνης καθώς δεν υπάρχει κανένας περιορισμός. Μπορούν να χρησιμοποιηθούν για πολλαπλούς σκοπούς σε ένα παιχνίδι. Στην παρούσα πτυχιακή εργασία θα αναλύσουμε το τρόπο με τον οποίο χρησιμοποιήσαμε τους ML-Agents για να αναπτύξουμε το παιχνίδι μας καθώς τον κώδικα που έχουμε γράψει προκειμένου ο Agent μας να μάθει αυτό που του έχουμε θέσει σαν στόχο.

Περίληψη

Σκοπός της παρούσας πτυχιακής είναι η ανάπτυξη ενός παιχνιδιού με τη βοήθεια της μηχανής Unity. Η Unity είναι ικανή να αναπτύξει δισδιάστατα και τρισδιάστατα παιχνίδια. Το παιχνίδι που αναπτύχθηκε στα πλαίσια της πτυχιακής εργασίας αφορά ένα τρισδιάστατο παιχνίδι αγώνων αυτοκινήτων όπου διάφοροι ευφυείς πράκτορες θα βρεθούν αντίπαλοι στην ίδια πίστα, ώστε να δούμε πως αλληλεπιδρούν και τον χρόνο που θα κάνει ο κάθε ένας για να τερματίσει.

Επίσης θα κάνουμε μια αναφορά στην ιστορία των βιντεοπαιχνιδιών, των μηχανών παιχνιδιών και ειδικότερα στο Unity3D. Θα δούμε αναλυτικά το περιβάλλον ανάπτυξης της Unity καθώς και τις λειτουργίες του παιχνιδιού. Στην παρούσα πτυχιακή θα δούμε αναλυτικά πως ένας εκπαιδευμένος πράκτορας έχει μάθει να οδηγεί ένα αυτοκίνητο σε μια αρκετά περίπλοκη πίστα με εμπόδια καθώς και με άλλους πράκτορες ταυτόχρονα, όσο το δυνατόν με καλύτερο χρόνο. Επίσης θα αναλύσουμε λεπτομερώς τον κώδικα που έχουμε αναπτύξει καθ' όλες τις φάσεις της εκπαίδευσης και άλλων παραμέτρων όπως η καταμέτρηση του χρόνου για κάθε ευφυή πράκτορα. Πολύ σημαντικό είναι και το περιβάλλον της εκπαίδευσης, οπότε θα γίνει και αναλυτική περιγραφή αυτού.

Abstract

The purpose of this thesis is to develop a game with the help of the Unity engine. Unity is capable of developing 2D or 3D games. The game developed for this thesis is a 3D car game where several intelligent agents will be opponents on the same track, so that we can see how they interact with each other as well as with the environment and the time each one will take to finish.

We will also make a reference to the history of video games, game engines and Unity3D in particular. We will look in detail at the Unity development environment as well as the game functions. In this thesis we will see in detail how a trained agent has learned to drive a car on a rather complicated obstacle course as well as with other agents at the same time, with the best possible time. We will also analyze in detail the code we have developed throughout the training phase and other parameters such as the time count for each intelligent agent. The educational environment is also very important, so a detailed description of this will be made.

Contents

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

Κεφάλαιο 1: Εισαγωγή στις Βασικές Έννοιες

1.1 Τί είναι Βιντεοπαιχνίδι

Βιντεοπαιχνίδι είναι ένα ηλεκτρονικό παιχνίδι το οποίο περιλαμβάνει αλληλεπίδραση με τον χρήστη, δηλαδή δίνει την δυνατότητα μέσω μιας διεπαφής χρήστη, να δημιουργήσει μια αλληλεπίδραση μεταξύ του ανθρώπου και του μέσου προβολής του παιχνιδιού όπως για παράδειγμα μια οθόνη . Σκοπό έχει την παραγωγή μιας οπτικής ανάδρασης σε μια συσκευή βίντεο. Για να κατανοήσουμε απόλυτα την έννοια βιντεοπαιχνίδι πρέπει να αναλύσουμε τις λέξεις από τις οποίες αποτελείται. Το βίντεο και το παιχνίδι. Η λέξη βίντεο στο βιντεοπαιχνίδι αναφέρεται σε μια συσκευή εμφάνισης γραφικών τύπου raster. Στα γραφικά των υπολογιστών καθώς και στην ψηφιακή φωτογραφία, ένα γραφικό raster είναι μια δισδιάστατη εικόνα από τετράγωνα εικονοστοιχεία, τα pixels, ορατή από κάποιο μέσο προβολής. Η πιο δημοφιλής χρήση του όρου βιντεοπαιχνίδι είναι ένα ηλεκτρονικό παιχνίδι το οποίο μπορεί να παιχτεί σε κάθε τύπου συσκευή που μπορεί να απεικονίσει δισδιάστατα ή τρισδιάστατα γραφικά. Τα ηλεκτρονικά συστήματα που χρησιμοποιούνται για την αναπαραγωγή ενός βιντεοπαιχνιδιού ονομάζονται πλατφόρμες. Τέτοιες πλατφόρμες μπορεί να είναι προσωπικοί υπολογιστές, παιχνιδομηχανές arcade ή και κονσόλες χειρός.

1.2 Ιστορία των βιντεοπαιχνιδιών

Η ιστορία των βιντεοπαιχνιδιών κάνει την εμφάνιση της στα τέλη της δεκαετίας του ’40. Συγκεκριμένα το 1947 οι Τόμας Γκόλντσμιθ και Έστλε Ρέι Μαν εφηύραν τις θερμικές βαλβίδες εμπνευσμένοι από την τεχνολογία του ραντάρ, που ήταν ένας σωλήνας καθοδικών ακτινών και χρησιμοποιήθηκαν για την κατασκευή του πρώτου βιντεοπαιχνιδιού και ονομάστηκε cathode ray tube amusement device. Ήταν μια αναλογική συσκευή προσομοίωσης πυραύλων. Επέτρεπε στον χρήστη να ελέγχει μια κουκίδα για την προσομοίωση ενός βλήματος σε κάποια σταθερά εικονοστοιχεία σε μία οθόνη CRT.

Μερικά χρόνια μετά, το 1951 ο Κρίστοφερ Στρέιτσι προσπάθησε να δημιουργήσει ένα επιτραπέζιο παιχνίδι ταβλιού στους υπολογιστές Pilot ACE, αλλά το πρόγραμμα που είχε αναπτύξει υπερέβαινε την χωρητικότητα της μνήμης του υπολογιστή, όμως αυτό δεν τον σταμάτησε αφού τον Οκτώβριο της ίδιας χρονιάς κωδικοποίησε το πρόγραμμα για ένα μηχάνημα μεγαλύτερης χωρητικότητας. Παραμένουμε στο 1951, όπου ο εφευρέτης Ράλφ Μπάερ που είναι και γνωστός ως ο πατέρας των βιντεοπαιχνιδιών σκέφτηκε ότι αν δινόταν στο χρήστη η δυνατότητα να ελέγχει αυτό που προβάλετε στην οθόνη, ο ρόλος του θα ήταν διαδραστικός.

Ένα χρόνο μετά ο Αλεξάντερ Σάφτο Ντάγκλας δημιούργησε το OXO, μια ψηφιακή εκδοχή της τρίλιζας. Ήταν το πρώτο ψηφιακό παιχνίδι στο οποίο ο άνθρωπος έπαιζε εναντίον του υπολογιστή. Η ιστορία συνεχίζεται το 1958, όπου ο Γουίλιαμ Χιγκίνμποθαμ, δημιούργησε το Tennis For Two. Θεωρείται το πρώτο βιντεοπαιχνίδι στην ιστορία το οποίο δημιουργήθηκε με την χρήση ενός παλμογράφου και ενός αναλογικού υπολογιστή της εποχής. Το παιχνίδι αυτό απεικονίζει ένα γήπεδο τένις και μια μπάλα που ελεγχόταν από την βαρύτητα. Ο χειρισμός την μπάλας γινόταν από δύο κουμπιά σε σχήμα κουτιού, ένα για την

τροχιά της μπάλας και ένα για το χτύπημα της. Τρία χρόνια αργότερα, το 1961 μια ομάδα φοιτητών του MIT, προγραμμάτισαν το Spacewar. Το παιχνίδι αυτό δημιουργήθηκε στον PDP-1, έναν καινούργιο υπολογιστή της τότε εποχής. Ήταν παιχνίδι που παιζόταν από δύο άτομα αντίπαλους όπου ο κάθε ένας ελέγχει ένα διαστημόπλοιο που ρίχνει πυραύλους καθώς και ένα αστέρι στο κέντρο της οθόνης που αποτελούσε απειλή για τα διαστημόπλοια. Παράλληλα την ίδια εποχή δημιουργήθηκε το πρώτο παιχνίδι εξομοίωσης για υπολογιστή. Ήταν ένα παιχνίδι μπέιζμπολ και για την ανάπτυξή του χρησιμοποιήθηκε ο υπολογιστής IBM 1620.

Το 1966 δημιουργήθηκε η πρώτη παιχνιδομηχανή που έκανε χρήση ενός ψεύτικου όπλου και λειτουργούσε ως αντικείμενο για να παίξει κάποιος ως είσοδο στο παιχνίδι. Το 1969, φτιάχτηκε το πρώτο παιχνίδι βασισμένο στο λειτουργικό σύστημα Unix. Ήταν η πρώτη εμφάνιση του λειτουργικού και το παιχνίδι που αναπτύχθηκε ονομάστηκε Space Travel και ήταν γραμμένο στην γλώσσα προγραμματισμού Fortran. Δύο χρόνια αργότερα, στο πανεπιστήμιο του Στάνφορντ, κάνει την εμφάνισή του ένα νέο παιχνίδι βασισμένο στο Spacewar. Ονομάστηκε Galaxy Game και ήταν το πρώτο παιχνίδι που λειτουργούσε με νομίσματα για να παίξει κάποιος. Αρχικά είχε δημιουργηθεί μόνο ένα μοντέλο αλλά ένα χρόνο μετά το project ανανεώθηκε.

Στα τέλη της δεκαετίας του ’70, μία παιχνιδομηχανή διαφημίζεται για πρώτη φορά στην τηλεόραση. Ήταν το Computer Space με δημιουργούς τους Νόλαν Μπάσνιλ και Τεντ Ντάμπνι. Το Computer Space είχε εμφανιστεί στις ΗΠΑ, Ευρώπη, Ιαπωνία, λατινική Αμερική και είχε κάνει πάρα πολλές πωλήσεις παγκοσμίως. Ύστερα από αυτήν την επιτυχία οι Νόλαν Μπάσνιλ και Τεντ Ντάμπνι δημιουργούν την Atari, Inc. λίγο καιρό πριν ανακοινώσουν το νέο του παιχνίδι, Pong. Ήταν το πρώτο Arcade βιντεοπαιχνίδι με πολύ μεγάλη επιτυχία. Ήταν βασισμένο στο κλασικό παιχνίδι πινγκ-πονγκ. Σαν εταιρία πούλησαν πάνω από 19000 αντίτυπα του παιχνιδιού και αυτό είχε ως αποτέλεσμα πολλοί να προσπαθήσουν να την αντιγράψουν.

Την ίδια χρονική περίοδο το πρώτο παιχνίδι που βγήκε στην αγορά κάνοντας χρήση μικροεπεξεργαστή ήταν το Gun Fight. Αυτό έμπνευσε τον δημιουργό του το 1978 να χρησιμοποιήσει μικροεπεξεργαστή στο νέο παιχνίδι Space Invaders, το οποίο είχε τεράστια επιτυχία και έτσι η βιομηχανία των Arcade βιντεοπαιχνιδιών μπήκε στη χρυσή εποχή των βιντεοπαιχνιδιών. Λόγο αυτού τα arcade βρίσκονταν παντού, σε εμπορικά κέντρα, βιτρίνες και καταστήματα ψιλικών. Το γεγονός έγινε θέμα σε όλα τα μέσα της εποχής και κατάφερε πάνω από 360.000 πωλήσεις μηχανημάτων παγκοσμίως και μέχρι το 1982 τα έσοδα ήταν στο ύψος των δυο δισεκατομμυρίων δολαρίων. Το 1979 η Namco είχε πουλήσει πάνω από 40.000 arcade μηχανές και η Atari κυκλοφόρησε το βιντεοπαιχνίδι Asteroids και κατάφερε να πουλήσει πάνω από 70.000.

Κατά της χρυσή εποχή, οι πωλήσεις των arcade μηχανών στην Αμερική είχαν αυξηθεί από 50 εκατομμύρια το 1978 σε 900 εκατομμύρια δολάρια μέχρι το 1981. Την εμφάνισή τους κάνουν τα έγχρωμα βιντεοπαιχνίδια στα τέλη το ’79 και το 1980 κυκλοφόρησε το pacman το οποίο πούλησε πάνω από 350.000 arcade Μηχανές. Ήταν το πρώτο παιχνίδι που κατά την διάρκεια του 20^{ου} αιώνα είχε πάνω από 10 δισεκατομμύριο δολάρια έσοδα.

Μέχρι και το 1982, οι βιομηχανία των arcade παιχνιδομηχανών είχε πουλήσει τόσες μηχανές συνολικής αξίας 8 δισεκατομμυρίων δολαρίων της τότε εποχής. Την ίδια εποχή η Nintendo κυκλοφόρησε το Donkey Kong ένα παιχνίδι που θα αφήσει ιστορία. Από αυτό το σημείο η βιομηχανία των βιντεοπαιχνιδιών έχει αναπτυχθεί ραγδαία επιχειρεί να φτιάξει κονσόλες με τις οποίες οι άνθρωποι θα μπορούν να παίζουν στα σπίτια τους.

Το 1985 κυκλοφόρησαν δυο παιχνίδια που μέχρι και σήμερα παίζονται από πολλούς ανθρώπους παγκοσμίως. Το πρώτο είναι το Tetris. Αναπτύχθηκε από τον Ρώσο προγραμματιστή Άλεξ Παγιτνοφ και έγινε το πρώτο ανταγωνιστικό παιχνίδι που θα απασχολούσε εκατομμύρια ανθρώπους για ώρες. Ύστερα η Nintendo δημιούργησε το Super Mario. Κυκλοφόρησε στην Ιαπωνία στα τέλη του 1985. Θεωρείται το καλύτερο παιχνίδι που κυκλοφόρησε ποτέ και παρέμεινε έτσι για αρκετά χρόνια. Ήταν το πρώτο παιχνίδι με τεχνολογία Side-scrolling που σημαίνει ότι καθώς ο χαρακτήρας προχωρούσε προς τα δεξιά, το περιβάλλον πήγαινε προς την αντίθετη κατεύθυνση.

Το 1989, η Nintendo κυκλοφόρησε την κονσόλα GameBoy. Δυο χρόνια μετά η Sega δημιουργεί το Sonic The Hedgehog, το οποίο ανταγωνίζεται αρκετά με το supermario. Μερικούς μήνες μετά η Capcom λανσάρει το Street Fighter και έτσι ξεκίνησαν να βγαίνουν παιχνίδια τύπου beat-em-ups.Το 1993 η βιομηχανία ασχολείται με παιχνίδια βίας και έτσι δημιουργήθηκαν οι τίτλοι mortal combat και night trap.

Το 1995 η Sony ανακοινώνει την πρώτη της παιχνιδομηχανή, το PlayStation. Με αυτή της κίνηση η Sony άλλαξε ολοκληρωτικά την βιομηχανία των παιχνιδιών καθώς ήταν η πρώτη κονσόλα με ιδανικά τεχνικά χαρακτηριστικά ώστε να μπορεί να τρέξει τρισδιάστατα και περίπλοκα παιχνίδια. Ένα χρόνο αργότερα η Nintendo ανακοινώνει το Nintendo 64. Ήταν μια αντίστοιχη παιχνιδομηχανή με το PlayStation, με την μόνη διαφορά ότι τα παιχνίδια που έπαιζαν σε αυτήν ήταν αποκλειστικά της ίδιας εταιρίας.

Το 2001, η Microsoft ανακοινώνει το Xbox. Μαζί με την Sony μέχρι και σήμερα οι δύο ανταγωνιστικές αυτές εταιρίες αναβαθμίζουν τις κονσόλες τους έχοντας αλλάξει τελείως την αντίληψη του κόσμου για τα βιντεοπαιχνίδια τα οποία έχουν γίνει σημαντικό μέρος της κουλτούρας του σύγχρονου κόσμου.

1.3 Τι είναι μια μηχανή παιχνιδιών

Για την δημιουργία οποιουδήποτε παιχνιδιού σήμερα είναι απαραίτητη η χρήση μίας μηχανής παιχνιδιού. Μία μηχανή παιχνιδιού είναι ένα σύστημα λογισμικού που είναι σχεδιασμένο να δουλεύει σε κονσόλες παιχνιδιών και σε λειτουργικά συστήματα ηλεκτρονικών υπολογιστών όπως τα Microsoft Windows, το Linux τα Mac OS X και πολλά ακόμα.

Οι μηχανές παιχνιδιών περιλαμβάνουν το λογισμικό του παιχνιδιού, καθώς παρέχουν πολλές λειτουργίες απαραίτητες για την δημιουργία ενός παιχνιδιού όπως είναι η μηχανή φωτοαπόδοσης δηλαδή ο “renderer” για τα 2D ή 3D γραφικά, τη μηχανή φυσικής η οποία είναι υπεύθυνη για τον εντοπισμό συγκρούσεων (collision detection), τον ήχο, την

ανάπτυξη κώδικα, τα animations, την διαχείριση της μνήμης, την τεχνητή νοημοσύνη και πολλά ακόμα. Το μεγαλύτερο πλήθος των μηχανών παιχνιδιών μοιράζονται τα ίδια χαρακτηριστικά. Τέτοια εργαλεία είναι απαραίτητα για την ανάπτυξη ενός παιχνιδιού

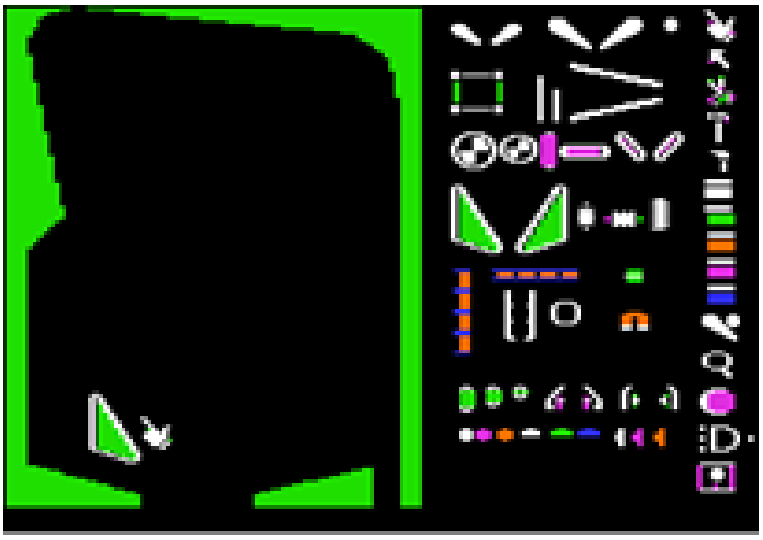
Τέτοιες μηχανές παιχνιδιών έχουν βοηθήσει πολύ τους προγραμματιστές και τους σχεδιαστές παιχνιδιών να μειώσουν τον χρόνο και τον κόπο υλοποίησης ενός παιχνιδιού διότι προσφέρουν πολλές ανέσεις στον τρόπο δημιουργίας ενός παιχνιδιού. Συχνά οι εταιρείες επαναχρησιμοποιούν την ίδια η μηχανή για την δημιουργία διαφορετικών παιχνιδιών, πράγμα που εξοικονομεί χρήματα διότι δεν χρειάζεται να κάνει αγορά κάποιας άλλης μηχανής παιχνιδιού.

Οι 3D μηχανές είναι κατασκευασμένες πάνω σε ένα API (Application Programming Interface) όπως είναι το Direct3D ή το OpenGL.

1.3.1 Ιστορία των μηχανών παιχνιδιών

Πριν από τη δημιουργία των μηχανών παιχνιδιών, τα παιχνίδια έπρεπε να γραφτούν από την αρχή ως μοναδικές οντότητες. Για να δημιουργηθεί ένα παιχνίδι σε κονσόλες όπως το Atari, έπρεπε το παιχνίδι να σχεδιαστεί εξολοκλήρου έτσι ώστε να κάνει βέλτιστη χρήση του υλικού εμφάνισης της οθόνης και του περιορισμού που υπήρχε στην μνήμη. Οι προγραμματιστές έκαναν χρήση απλών εργαλείων κειμένου για να γράφουν κώδικα, ο οποίος δεν μπορούσε να χρησιμοποιηθεί ξανά διότι με τον χρόνο οι διαθέσιμοι πόροι που υπήρχαν άλλαζαν. Για αυτό και κάθε παιχνίδι που έβγαινε στη αγορά γραφόταν εξ-ολοκλήρου από την αρχή.

Κατά την διάρκεια της δεκαετίας του ’80, υπήρχαν διάφορα δισδιάστατα συστήματα δημιουργίας παιχνιδιών όπως ήταν το Pinball construction set (1983), στο οποίο ο χρήστης μπορούσε να φτιάξει το δικό του επίπεδο.



Εικόνα 1 - Pinball construction set (1983)

Ένα χρόνο αργότερα η εταιρία EA (Electronic Arts) έβγαλε στην αγορά το Advanced Construction Set αρχικά για την πλατφόρμα Commodore 64 και αργότερα για το Apple II , την Amiga και το MS-DOS. Χρησιμοποιήθηκε για την κατασκευή παιχνιδιών περιπέτειας βασισμένο σε πλακίδια. Αυτό έδωσε στους χρήστες την δυνατότητα να δημιουργούν πίστες, πλάσματα, αντικείμενα και να κάνουν χρήση ενός βασικού μενού για να αλλάζουν την λογική του παιχνιδιού.

Εικόνα 2 - Δωμάτιο φτιαγμένο από πλακίδια στην Amiga

Από αυτή τη χρονική περίοδο είχε αρχίσει η παραγωγή των πρώτων μηχανών παραγωγής παιχνιδιών για παιχνίδια δυο διαστάσεων ακόμα διότι τα μηχανήματα της εποχής δεν είχαν την υπολογιστική δύναμη για κάτι παραπάνω.

Μέχρις ότου το 1990 όπου βγήκε η Ultima Engine όπου εκεί ξεκίνησαν τα πρώτα ψευδό-τρισδιάστατα παιχνίδια. Ήταν τα πρώτα παιχνίδια που χαρακτηρίστηκαν ως τρισδιάστατα διότι τα αντικείμενα είχαν ύψος και κεκλιμένες επιφάνειες που μπορούσαν να έχουν και τρισδιάστατα εφέ. Επίσης είχε και αλγόριθμο τέτοιο ώστε να μπορεί να τοποθετεί μία δισδιάστατη εικόνα πάνω σε ένα τρισδιάστατο μοντέλο. Παρόλο που οι μη ελεγχόμενοι χαρακτήρες ήταν ακόμα δισδιάστατα sprites, όλα τα αντικείμενα στον κόσμο ήταν τριών διαστάσεων κάτι που δεν υπήρχε μέχρι τότε.

Η id Software το 1992 κυκλοφόρησε το πρώτο εξ-ολοκλήρου τρισδιάστατο παιχνίδι πρώτου προσώπου, Castle Wolfenstein 3D το οποίο εκτόξευσε το τρόπο που βλέπουμε τα παιχνίδια μέχρι και σήμερα. Ένα χρόνο αργότερα βγήκε το Doom Engine το οποίο χρησιμοποιήθηκε για να παραχθούν τα Doom όπως τα ξέρουμε μέχρι και σήμερα. Από τότε προγραμματιστές άλλων εταιριών αγόρασαν τον πυρήνα του λογισμικού των μηχανών και έκαναν τις δικές τους μετατροπές και δικές τους εκδόσεις.

Εικόνα 3 - Castle Wolfenstein 3D 1992 MS-DOS

Φτάσαμε στην χρυσή εποχή, την εποχή που βγήκε η Unreal Engine από την Epic Games το 1998. Έγινε η μεγαλύτερη ανταγωνιστική εταιρία για την Id Software. Έχουν γίνει πολλές τροποποιήσεις στην μηχανή όπου σήμερα έχουν την 5η έκδοση της. Το πιο γνωστό παιχνίδι της Unreal είναι το Unreal Tournament, ένα πρώτου προσώπου ανταγωνιστικό Multiplayer το οποίο ανακοινώθηκε παιχνίδι της χρονιάς για το 1999.

Εικόνα 4 - Unreal Engine Logo

Αργότερα έρχεται η Valve το 2004 να μας παρουσιάσει την Source Engine με την οποία κατασκευάστηκαν οι πολύ επιτυχημένοι τίτλοι: Half-life, Team Fortress, Portal και Counter-Strike. Την ίδια χρονιά κάνει την εμφάνισή της η CryEngine της εταιρίας CryTek η οποία έφτιαξε του τίτλους: FarCry, Crysis και Warface.

Εικόνα 5 - Source Engine Logo

Εικόνα 6 - CryEngine Logo

Δυο χρόνια μετά εμφανίζεται η Frostbite της εταιρίας Dice, υπεύθυνη αρχικά για την δημιουργία των τίτλων Battlefield και αργότερα για τίτλους όπως το Need For Speed, Fifa, Star Wars. Την ίδια χρονιά η Rockstar χρησιμοποίησε την Rage Engine (Rockstar Advanced Game Engine) για την κατασκευή του τίτλου Grand Theft Auto IV.

Εικόνα 7 - FrostBite Engine Logo

Εικόνα 8 - Rage Engine Logo

Τέλος το 2005 κάνει την εμφάνισή της η πρώτη έκδοση της Unity με την οποία ακόμα και σήμερα κατασκευάζονται και έχουν κατασκευαστεί πάρα πολλοί τίτλοι. Η Unity κάνει συνεχώς αναβαθμίσεις, διορθώνει bugs και παρέχει ένα μεγάλο πλήθος δωρεάν αντικειμένων.

Εικόνα 9 - Unity Game Engine Logo

Υπάρχουν πολλές ακόμα παρόμοιες μηχανές παιχνιδιών πράγμα που κάνει κάθε προγραμματιστή, ερασιτέχνη και εταιρεία να διαλέξει μεταξύ αυτών όποια επιθυμεί ανάλογα με τις ανάγκες του.

Οι δημοφιλέστερες μηχανές παιχνιδιών έως και σήμερα είναι οι εξής:

- Unity
- Unreal Engine
- Game Maker Studio
- Cry Engine
- Godot
- Source Engine

1.4 Ιστορική αναδρομή της Τεχνητή Νοημοσύνης

Για να κατανοήσουμε την έννοια της μηχανικής μάθησης θα πρέπει να ρίξουμε μια ματιά στα πρώτα βήματα της τεχνητής νοημοσύνης (ΤΝ). Η τεχνητή νοημοσύνη (Artificial Intelligence, AI) καθιστά τις μηχανές ικανές να μαθαίνουν από την εμπειρία, να προσαρμόζονται σε νέα εισαγόμενα δεδομένα και να εκτελούν ανθρωπομορφικά έργα. Το 1943 παρουσιάστηκε η πρώτη εργασία που αναγνωρίζεται ως ΤΝ από τους Warren McCulloch και Walter Pitts. Βασίστηκαν στην γνώση της βασικής φυσιολογίας και λειτουργίας των νευρώνων του εγκεφάλου και στην θεωρία υπολογισμού του Turing. Ο Alan Turing στο άρθρο του ‘’Computer Machinery and Intelligence’’ το 1950 ήταν αυτός που διατύπωσε ένα πλήρες όραμα για τη ΤΝ. Στο άρθρο αυτό παρουσίασε την δοκιμασία Turing, την μηχανική μάθηση (Machine Learning), τους γενετικούς αλγόριθμους και την ενισχυτική μάθηση.

Η τεχνητή νοημοσύνη πήρε το όνομά της το 1956 στο Dartmouth College από τον John McCarthy. Ο ίδιος συγκέντρωσε επιστήμονες από όλη την Αμερική με ενδιαφέρον για την

τεχνητή νοημοσύνη και έτσι διοργανώθηκε δίμηνη συνάντηση εργασίας. Από τότε ξεκίνησε η ΤΝ να θεωρείται ξεχωριστό πεδίο στην επιστήμη της πληροφορικής. Στη δίμηνη αυτή συνάντηση οι Allen Newell, Herbert Simon και Cliff Shaw παρουσίασαν ένα πρόγραμμα συλλογιστικής το Logic Theorist, το οποίο μπορούσε να αποδείξει πολλά μαθηματικά θεωρήματα. Η πορεία την ΤΝ από το 1956, όταν πήρε το όνομά της και όταν θεωρήθηκε ξεχωριστός κάδος της επιστήμης των υπολογιστών, μέχρι σήμερα μπορεί να χωριστεί σε 5 περιόδους.

Η εποχή μεταξύ του 1957 και 1974 χαρακτηρίζεται η χρυσή εποχή της ΤΝ. Οι υπολογιστές της εποχής αυτής αναπτύχθηκαν ραγδαία έχοντας την δυνατότητα να αποθηκεύουν περισσότερες πληροφορίες να επεξεργάζονται γρηγορότερα δεδομένα και να είναι πιο προσβάσιμοι σε περισσότερα πανεπιστήμια. Επίσης βελτιώθηκαν οι αλγόριθμοι μηχανικής μάθησης άλλα και οι άνθρωποι που χειρίζονταν του υπολογιστές αυτούς, έχοντας αποκτήσει πλέον εμπειρία για το ποιος είναι ο κατάλληλος αλγόριθμος για τη λύση κάποιου προβλήματος. Το 1958 ο McCarthy δημιούργησε την γλώσσα υψηλού επιπέδου "LISP" στο M.I.T. η οποία είναι ακόμα και σήμερα η αγαπημένη γλώσσα προγραμματισμού για την έρευνα της τεχνητής νοημοσύνης. Το 1959 ο Arthur Samuel επινόησε τον όρο Machine Learning (ML) όταν έδινε λόγο για να προγραμματιστεί ένας υπολογιστής που θα μπορεί να παίζει σκάκι καλύτερα και από τον δημιουργό του ίδιου του προγράμματος. Το 1961 ο George Devol σχεδίασε το πρώτο βιομηχανικό ρομπότ, που χρησιμοποιήθηκε στο εργοστάσιο της Generals Motors για εργασίες που θεωρούνταν επικίνδυνες για τον άνθρωπο. Ένα άλλο ρομπότ που κατασκευάστηκε το 1966 ήταν το Shakey the Robot, το οποίο κατασκευάστηκε από τον Charles Rosen και ήταν το πρώτο ρομπότ γενικού σκοπού. Ονομάστηκε "the first electonic person". Μεγάλη πρόοδος έγινε αυτή την περίοδο και στην ικανότητα της ΤΝ να επικοινωνεί. Σπουδαιότερο επίτευγμα είναι η ELIZA, ένα πρόγραμμα γραμμένο από τον Joseph Weizenbaum, ικανό να συνομιλεί στα αγγλικά με κάποιον άνθρωπο.

Η περίοδος 1974-1980 αποτελεί τον πρώτο χειμώνα της ΤΝ. Στις αρχές της δεκαετίας αυτής η ΤΝ κατακρίθηκε από πολλούς και υπήρξαν πολλά οικονομικά προβλήματα που παρουσιάστηκαν στις εφαρμογές της, με αποτέλεσμα την μεγάλη ελάττωση των χρηματοδοτήσεων. Η έρευνα πάνω στα νευρωνικά δίκτυα σχεδόν εξαφανίστηκε εξαιτίας τις κριτικής του Marvin Minsky για τους νευρώνες. Παρόλες τις δυσκολίες όμως, υπάρχουν και κάποιες εξαιρετικές ιδέες που εξερευνήθηκαν σε τομείς της ΤΝ, όπως τον λογικό προγραμματισμό, την κοινή λογική και άλλους.

Η επιστροφή των χρηματοδοτήσεων αλλά και η περαιτέρω ανάπτυξη των αλγορίθμων της τεχνητής νοημοσύνης συνετέλεσαν στον ερχομό μιας δεύτερης χρυσής εποχής στην

έρευνα της ΤΝ από το 1980 έως το 1987. Σημαντικό ρόλο σε αυτό έπαιξε η εισαγωγή των expert systems από τον Edward Feigenbaum. Τα συστήματα αυτά μιμούνταν την διαδικασία λήψης αποφάσεων ειδικών. Το πρόγραμμα ρωτούσε έναν ειδικό πως θα συμπεριφερόταν σε μια κατάσταση και μετά, αφού είχε μάθει για όλες τις πιθανές καταστάσεις, συμβούλευε μη ειδικούς. Η Ιαπωνική κυβέρνηση χρηματοδότηση πολύ μεγάλα ποσά στην κατεύθυνση αυτή. Σημαντική εξέλιξη υπήρξε και στις τεχνικές του deep learning, και στα νευρωνικά δίκτυα γενικότερα τα οποία ξεκίνησαν πάλι να απασχολούν ερευνητές μετά από την δεκάχρονη στάση ανάπτυξής τους.

Η έλλειψή πρακτικών εφαρμογών στην βιομηχανία οδήγησε στον δεύτερο χειμώνα της ΤΝ την περίοδο 1987 έως 1993. Κύριος λόγος ήταν ότι οι επιχειρήσεις πίστεψαν ότι η τεχνολογία αύτη δεν ήταν βιώσιμη οικονομικά. Η τεχνολογία των expert systems θεωρήθηκε πολύ δαπανηρή για να διατηρηθεί και περισσότερες από 300 εταιρείες ειδικευμένες στην ΤΝ αναγκάστηκαν να κλείσουν. Μια θετική εξέλιξη ήταν ότι πολλοί ερευνητές πρότειναν καινούριες οπτικές πάνω στην ιδέα της ΤΝ, ερευνητές όπως ο Rodney Brooks και ο Hans Moravec στη ρομποτική. Αξίζει να αναφερθεί η δημιουργία του πρώτου αυτοοδηγούμενου αυτοκινήτου το 1986 από την Mercedes-Benz. Ήταν εξοπλισμένο με κάμερες και αισθητήρες και μπορούσε να αναπτύξει ταχύτητα έως 55mph και να οδηγήσει σε δρόμο χωρίς άλλα αυτοκίνητα, εμπόδια ή ανθρώπους.

Από το 1993 και ύστερα η ΤΝ κατάφερε να πετύχει πολλούς από τους αρχικούς της στόχους. Ένας από τους λόγους είναι η μεγάλη αύξηση της υπολογιστικής ισχύος των σύγχρονων υπολογιστών. Αυτό οδήγησε στην πιο αποτελεσματική χρησιμοποίηση των αλγόριθμων της ΤΝ από τη βιομηχανία, η οποία ήταν ακόμα επιφυλακτική απέναντι στην επανάσταση που έφερναν οι ιδέες της τεχνητής νοημοσύνης. Ένας άλλος λόγος της ραγδαίας ανάπτυξης της ΤΝ αυτήν την εποχή είναι η διαχώριση των τομέων της από τους ερευνητές, και η πιο ειδικευμένη ενασχόληση με τους διάφορους τομείς από εξειδικευμένες ομάδες έρευνας. Το 1995 δημιουργήθηκε η A.L.I.C.E.,ένα chatbot εμπνευσμένο από την ELIZA. Το 1997 δημιουργήθηκε ο Deep Blue, ένα πρόγραμμα για να παίζει σκάκι, το οποίο κατάφερε να κερδίσει έναν αγώνα ενάντια στον παγκόσμιο πρωταθλητή. Στα χρόνια που ακολούθησαν, μικρό μέρος της κοινότητα της ΤΝ υποστήριξε πως δεν κατάφερε να αποδώσει τα όνειρα των πρώτων ερευνητών για δημιουργία μιας αληθινής νοημοσύνης, αλλά είναι γεγονός ότι η ΤΝ εξελίχθηκε με τεράστια βήματα, πιο προσεκτική και πιο χρήσιμη από ποτέ πριν.

Κεφάλαιο 2: Unity

2.1 Γιατί Unity;

Η Unity είναι ένα ολοκληρωμένο σύστημα ανάπτυξης cross-platform παιχνιδιών που αναπτύχθηκε από την εταιρία 'Unity Technologies' και χρησιμοποιείται για την ανάπτυξη παιχνιδιών για υπολογιστές, κονσόλες, κινητά και ιστοσελίδες. Η 'Unity Technologies' είναι μία εταιρεία ανάπτυξης λογισμικού παιχνιδιών η οποία βρίσκεται στο Σαν Φρανσίσκο της Καλιφόρνιας. Ιδρύθηκε το 2004 από τους David Helgason και Nicholas Francis και αρχικά είχε το όνομα OTEE (Over The Edge Entertainment), όπου το 2007 πήρε το όνομα που ξέρουμε μέχρι και σήμερα. Η εταιρία τον Οκτώβριο του 2009 παραχώρησε δωρεάν την Unity με την έκδοση 2.6 όπου μέχρι και σήμερα διατίθεται δωρεάν. Αυτό είχε ως αποτέλεσμα να αυξηθεί πολύ ο αριθμός των developers και συνεπώς των παιχνιδιών που αναπτύχθηκαν εκείνη την εποχή. Όπως αναφέρθηκε προηγούμενως η Unity είναι ένα σύστημα ανάπτυξης cross-platform παιχνιδιών, το οποίο σημαίνει ότι δίνει την δυνατότητα στον developer να αναπτύξει το παιχνίδι του σε οποιαδήποτε πλατφόρμα επιθυμεί.

Οι πλατφόρμες αυτές είναι οι εξής:

- PC
- Android
- Universal Windows Platform
- tvOS
- PS4
- IOS
- Xbox One
- WebGL (Για Ιστοσελίδες)

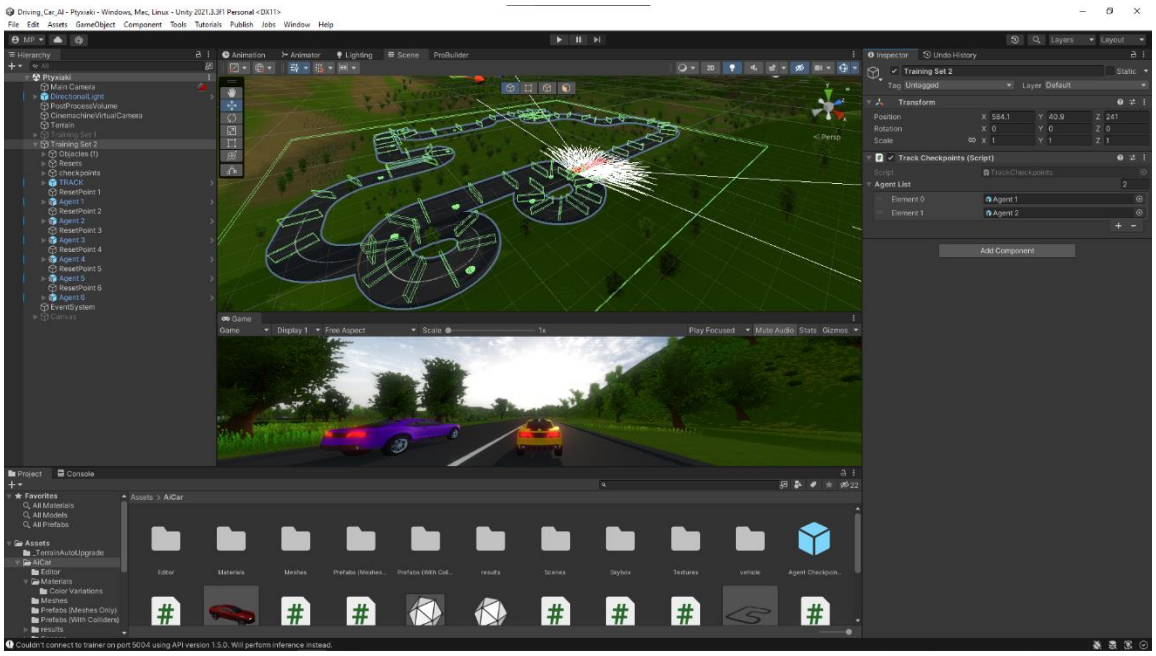
Η επιλογή της Unity για την ανάπτυξη του παιχνιδιού δεν ήταν τυχαία και οι λόγοι ποικίλουν. Το περιβάλλον ανάπτυξης της είναι εύχρηστο και εύκολο στην εκμάθηση του. Χρησιμοποιεί τις γλώσσες προγραμματισμού C# και JavaScript. Στην παρούσα πτυχιακή θα ασχοληθούμε με την C# η οποία είναι πολύ κοντά στην λογική της C πράγμα που την κάνει εύκολη στην κατανόηση της. Είναι πολύ παραγωγική που αυτό σημαίνει ότι ο χρήστης έχει την δυνατότητα να δημιουργήσει ένα παιχνίδι σε ένα μικρό χρονικό διάστημα. Επίσης πιο σημαντικό είναι ότι έχει πολύ μεγάλη κοινότητα στην οποία ο κάθε ένας μπορεί να βρει πληροφορίες για κάποιο πρόβλημα που αντιμετωπίζει ή να προτείνει ο ίδιος μια λύση σε κάποιο πρόβλημα. Υπάρχουν πάρα πολλά tutorials για την κάθε

λεπτομέρεια της μηχανής και για την ανάπτυξη κώδικα καθώς και για πολλές τεχνικές υλοποίησης. Όπως αναφέραμε και παραπάνω η Unity έχει και δικό της ηλεκτρονικό κατάστημα, το Asset Store το οποίο είναι πολύ χρήσιμο για τους developers. Αυτοί ήταν οι βασικότεροι παράγοντες για την επιλογή της Unity. Η έκδοση που θα χρησιμοποιήσουμε στην παρούσα πτυχιακή είναι η 2021.3.3f1.

2.2 Unity Editor

Ο Editor στη Unity αποτελείται από διάφορα panel τα οποία μπορούν να προσαρμοστούν στις προτιμήσεις του developer και έτσι να δημιουργήσει το δικό του Interface. Τα κυριότερα Panel είναι τα ακόλουθα:

- Ιεραρχία (Hierarchy) / Inspector
- Project Panel
- Scene View / Game View
- Animator/Animation
- Console
- Asset Store
- Lighting

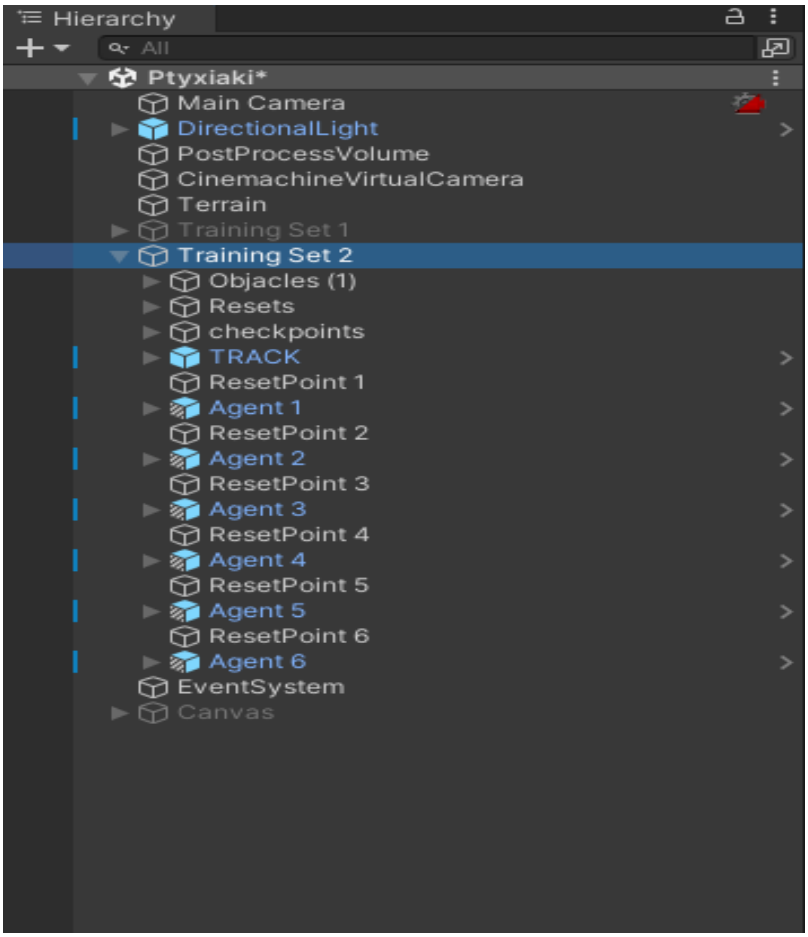


Εικόνα 10 - Unity Editor

Παρακάτω θα αναλύσουμε κάθε ένα ξεχωριστά για να καταλάβουμε την χρησιμότητα τους.

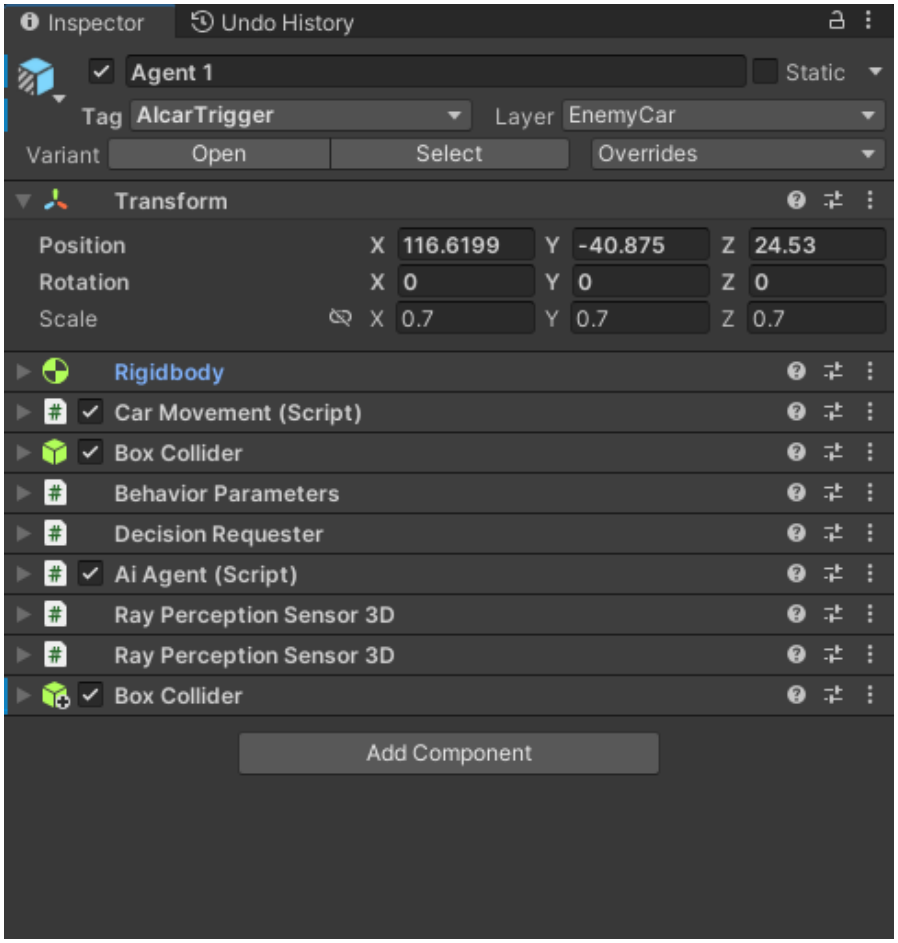
2.2.1 Ιεραρχία (Hierarchy) / Inspector

- Το Panel της Ιεραρχίας είναι ένα από τα βασικότερα παράθυρα του Editor διότι είναι εκείνο που περιέχει όλα τα αντικείμενα που έχουμε δημιουργήσει στο παιχνίδι μας στην συγκεκριμένη σκηνή. Τα αντικείμενα αυτά μπορούν να δημιουργηθούν είτε από το παράθυρο Project, δηλαδή αν έχουμε φτιάξει οποιοδήποτε αρχείο είναι συμβατό με την Unity και το έχουμε αποθηκεύσει σε ένα φάκελο στον υπολογιστή μας, είτε από ένα απλό drag and drop από το windows explorer. Επιπλέον ο developer μπορεί να δημιουργήσει κάποια βασικά έτοιμα αντικείμενα που προσφέρει η Unity όπως είναι διάφορα 3d και 2d μοντέλα, διάφορα εφέ, φωτισμό που θα αναλύσουμε παρακάτω, πηγές ήχου και βίντεο, 3d cameras καθώς και επιλογή για να δημιουργήσει το δικό του User Interface. Όλα τα παραπάνω μπορούν να επεξεργαστούν όταν επιλεχθούν από το παράθυρο Inspector.



Εικόνα 11 - Hierarchy

- Το παράθυρο Inspector είναι ένα από τα σημαντικότερα παράθυρα που έχει η Unity. Σε αυτό εμφανίζονται όλα τα χαρακτηριστικά των αντικειμένων που έχουμε δημιουργήσει για το παιχνίδι μας. Όπως και πριν ο χρήστης επιλέγει το αντικείμενο που θέλει από την ιεραρχία και απευθείας εμφανίζονται όλα τα χαρακτηριστικά του στο παράθυρο αυτό. Βασικά χαρακτηριστικά είναι το transform το οποίο επιστρέφει την θέση, την περιστροφή και το μέγεθος του αντικειμένου μέσα στον χώρο. Ύστερα μπορούμε μέσα από το παράθυρο του Inspector να δώσουμε στο επιλεγμένο αντικείμενο οποιοδήποτε από τα συστατικά μας παρέχει η Unity. Στα κεφάλαιο 4 θα αναλύσουμε τα συστατικά που έχουμε χρησιμοποιήσει για να δημιουργήσουμε τον πράκτορα μας και όχι μόνο. Οι πληροφορίες που μπορεί να πάρει ο χρήστης για κάθε αντικείμενο ποικίλουν ανάλογα με το είδος του αντικειμένου.

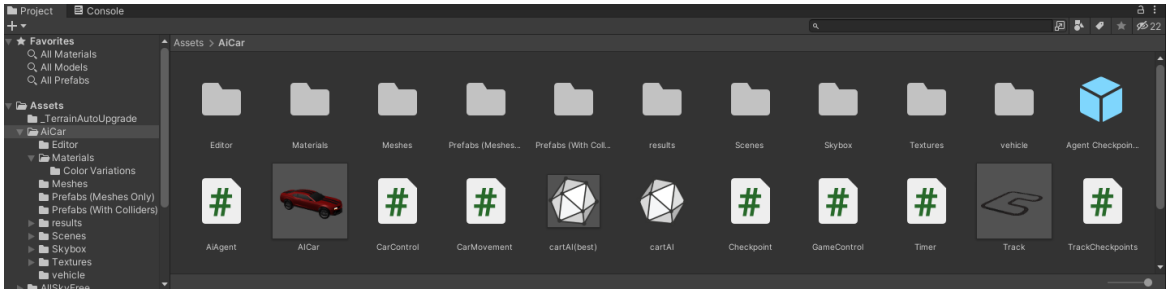


Εικόνα 12 - Inspector

2.2.2 Project Panel

Το παράθυρο Project είναι εκείνο που περιέχει όλα τα αρχεία που είναι υπεύθυνα για την δημιουργία ενός παιχνιδιού. Εκεί βρίσκονται όλοι οι φάκελοι που μπορεί να φτιάξει ο developer για την καλή οργάνωση των αρχείων του καθώς και όλες τις εικόνες, τα βίντεο, τους ήχους, τα τρισδιάστατα και δισδιάστατα μοντέλα, τα scripts και πολλά ακόμα που επιθυμεί να χρησιμοποιήσει. Στο παράθυρο αυτό ο developer μπορεί να δημιουργήσει το μεγαλύτερο πλήθος από αντικείμενα απαραίτητα για την ανάπτυξη ενός παιχνιδιού, όπως για παράδειγμα: η δημιουργία αρχείων κώδικα, materials τα οποία είναι υπεύθυνα για τα textures του κάθε αντικειμένου, τα animations, διάφορα εφέ και πολλά ακόμα απαραίτητα

εργαλεία.



Εικόνα 13 - Project Panel

2.2.3 Scene View / Game View

Το Scene View είναι το panel εκείνο στο οποίο μπορεί να κατασκευαστεί η σκηνή του παιχνιδιού. Είναι ο χώρος που μπορεί ο developer να επεξεργαστεί όλα του τα αντικείμενα χρησιμοποιώντας τα εξής εργαλεία που προσφέρει ο editor:

- **Hand Tool**

  Με το εργαλείο αυτό ο χρήστης μπορεί να περιηγηθεί μέσα στον χώρο της σκηνής που είναι ενεργής μέσω της 3D κάμερας.

- **Move Tool**

  Με το εργαλείο αυτό ο χρήστης μπορεί να επιλέξει ένα αντικείμενο από το παράθυρο της ιεραρχίας ή από το παράθυρο της σκηνής και να το μετακινήσει στο χώρο προς όλες τις κατευθύνσεις.

- **Rotate Tool**

  Με το εργαλείο αυτό ο χρήστης μπορεί να επιλέξει ένα αντικείμενο όπως αναφέραμε προηγουμένως και να το περιστρέψει στο επιλεγμένο άξονα περιστροφής του αντικειμένου.

- **Scale Tool**

  Με το εργαλείο αυτό ο χρήστης μπορεί να αλλάζει το μέγεθος και τις διαστάσεις του επιλεγμένου αντικειμένου.

Εικόνα 14 - Scene View

Το Game View είναι το παράθυρο εκείνο που δείχνει κάθε φορά την τελική μορφή του παιχνιδιού. Εκεί ο χρήστης μπορεί να χειριστεί και να πειραματιστεί με τον χαρακτήρα που έχει φτιάξει, να εντοπίσει λάθη και να αποφασίσει τις επόμενες του ενέργειες.

Εικόνα 15 - Game View

Οι ενέργειες αυτές πραγματοποιούνται με τη χρήση των παρακάτω πλήκτρων:

- **Play**

  Με το πλήκτρο play ο χρήστης μπορεί να τρέξει και να δοκιμάσει την τελική μορφή του παιχνιδιού για την αντίστοιχη σκηνή.

- **Pause**

  Το πλήκτρο pause χρησιμοποιείται για να γίνεται παύση την σκηνής σε ένα συγκεκριμένο καρέ, ώστε να γίνουν οποιεσδήποτε αλλαγές.

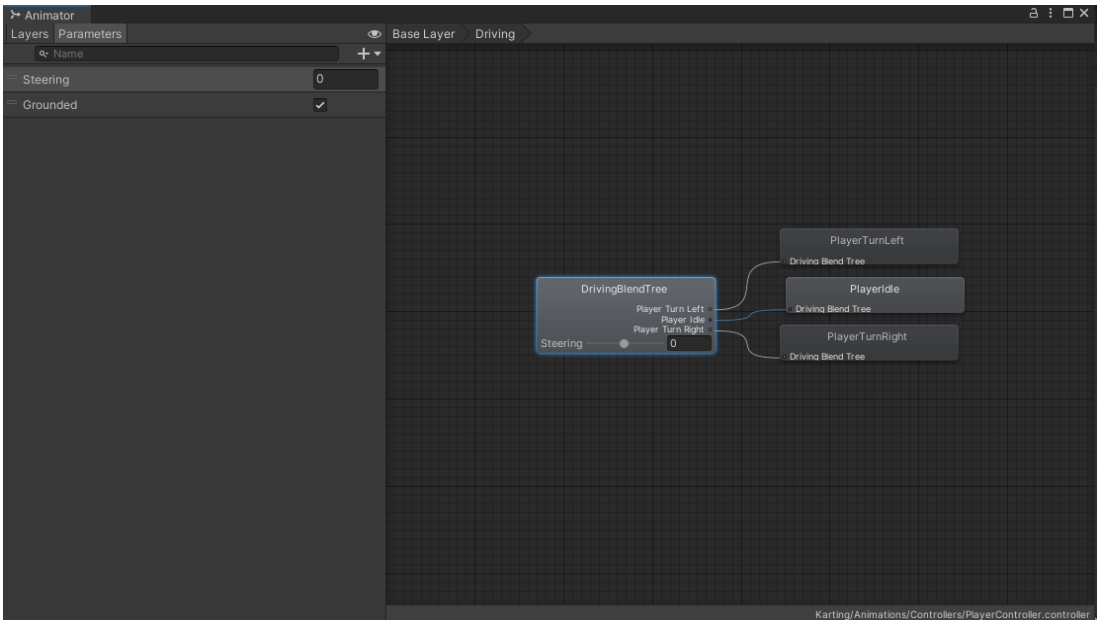
- **Step**

  Το πλήκτρο Step χρησιμοποιείται ώστε να μπορεί ο χρήστης να προχωράει καρέ καρέ δηλαδή βήμα βήμα στο παιχνίδι για να ελέγχει καλύτερα τις αλλαγές που γίνονται είτε σε τιμές είτε σε άλλες μεταβλητές του παιχνιδιού.

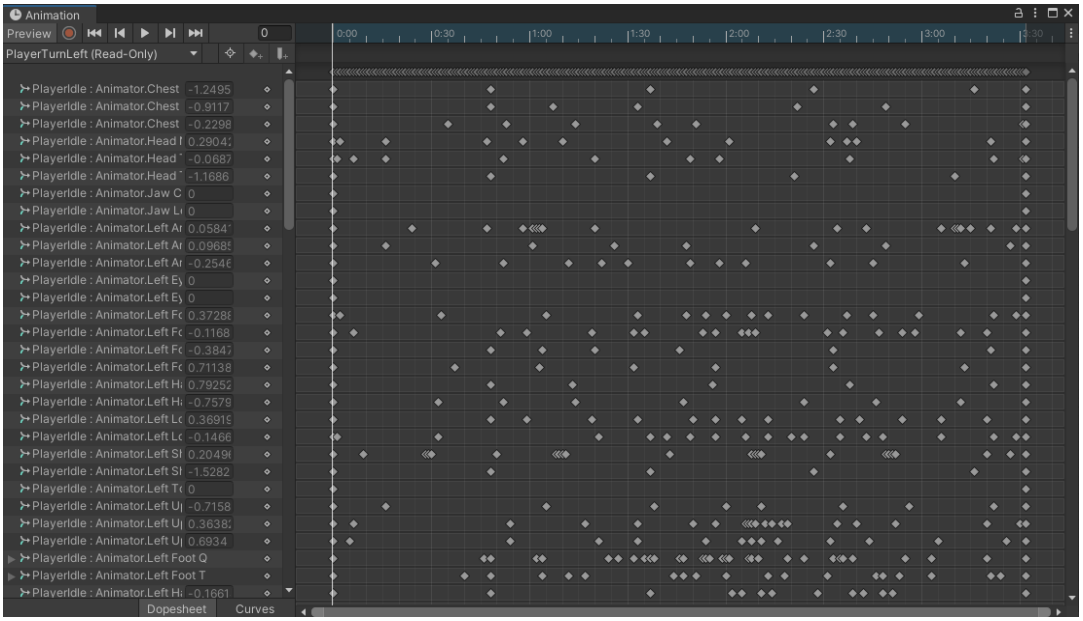
Εικόνα 16 - Control Buttons

2.2.4 Animator/Animation

Υπάρχουν δυο παράθυρα για τα animations. Το παράθυρο Animator και το Animation. Το παράθυρο Animation είναι υπεύθυνο για την δημιουργία των animation κάθε αντικειμένου. Η Unity έχει κάνει πολύ εύκολη την χρήση και την δημιουργία των animation καθώς έχει ένα πολύ εύκολο στη χρήση παράθυρο. Από την άλλη το παράθυρο Animator είναι υπεύθυνο για την επεξεργασία των animation που δημιουργήσαμε πριν και για τον έλεγχο και για τις μεταβάσεις από ένα animation σε ένα άλλο με την χρήση διαφόρων τύπων μεταβλητών όπως int, float, bool κλπ. Όλα τα Animation που μπορεί να φτιάξει ο developer μπορούν να προγραμματιστούν μέσα από τον κώδικα με διάφορες εντολές ελέγχου που παρέχονται από την βιβλιοθήκη της Unity.



Εικόνα 17 - Animator



Εικόνα 18 - Animation

2.2.5 ProBuilder

Το ProBuilder είναι ένα χρήσιμο εργαλείο που χρησιμοποιήθηκε στα πλαίσια δημιουργίας του παιχνιδιού. Συγκεκριμένα δίνει την δυνατότητα κατασκευής και σχεδιασμού τρισδιάστατων αντικειμένων, όπως για παράδειγμα ένα βράχο μέχρι και ένα ολόκληρο κτήριο, χωρίς τη χρήση κάποιου άλλου προγράμματος. Για να χρησιμοποιήσουμε αυτό το εργαλείο πρέπει πρώτα να το εγκαταστήσουμε από το Unity Package Manager που θα αναφέρουμε παρακάτω.

2.2.6 Lighting

Το Lighting στην Unity λειτουργεί προσεγγίζοντας το πώς συμπεριφέρεται το φως στον πραγματικό κόσμο. Η Unity χρησιμοποιεί λεπτομερή μοντέλα για το πως φως για ένα πιο ρεαλιστικό αποτέλεσμα ή απλοποιημένα μοντέλα για ένα πιο στυλιζαρισμένο αποτέλεσμα.

Η Unity διαθέτει τέσσερις τύπους για την δημιουργία μίας πηγής φωτός

1. **Point Light:** Πηγή που διαχέει το φως προς όλες τις κατευθύνσεις σφαιρικά όπως μια λάμπα. Ανάλογα με την ένταση που έχει δώσει ο χρήστης, αυτή μειώνεται αντιστρόφως ανάλογα με την απόσταση από την πηγή.

2. **Spot Light:** Πηγή που διαχέει το φως κωνικά σαν ένας φακός. Με κατάλληλη ρύθμιση μπορούμε να αλλάξουμε την γωνία και κατά συνέπεια το πλάτος του κώνου, την ένταση του φωτός και το εύρος του.

3. **Directional Light:** Πηγή που η ένταση δεν εξαρτάται από την απόσταση και συνεπώς μπορεί να τοποθετηθεί οπουδήποτε μέσα στην σκήνη.

4. **Area Light**: Η περιοχή που διαχέει το φως ορίζεται από ένα ορθογώνιο. Το φως εκπέμπεται σε όλες τις κατευθύνσεις ομοιόμορφα σε όλη την επιφάνεια του, αλλά μόνο από την μία πλευρά του ορθογωνίου. Δεν υπάρχει χειροκίνητος έλεγχος για το εύρος της περιοχής φωτισμού, ωστόσο η ένταση θα μειωθεί στο αντίστροφο τετράγωνο της απόστασης καθώς απομακρύνεται από την πηγή. Σε αυτήν την περίπτωση επειδή ο υπολογισμός του φωτισμού είναι αρκετά απαιτητικός για το σύστημα, η χρήση των Area Lights δεν μπορεί να γίνει Real

Time αλλά μόνο Baked.

Η Unity διαθέτει τρείς τρόπους υπολογισμού για τον φωτισμό

1. **RealTime Lighting :** Είναι η ποιο απαιτητική επιλογή για το σύστημα καθώς χρησιμοποιείται για πηγές φωτός που πρέπει να αλλάξουν τις ιδιότητες τους ή που δημιουργούνται μέσω σεναρίων κατά την διάρκεια το παιχνιδιού. Η Unity υπολογίζει και ενημερώνει τον φωτισμό αυτών των φωτών κάθε καρέ κατά τον χρόνο εκτέλεσης.

2. **Baked Lighting:** Είναι η λιγότερο απαιτητική επιλογή διότι η Unity υπολογίζει εκ των προτέρων τον φωτισμό από αυτά τα φώτα πριν τον χρόνο εκτέλεσης.

3. **Mixed Lighting:** Είναι ένας συνδυασμός των δύο προηγούμενων και χρησιμοποιείται κυρίως για σκηνές που δεν είναι απαραίτητο να κάνει υπολογισμούς Real Time σε όλες τις φωτεινές πηγές, έτσι ώστε να μην καταναλώνονται πολλοί πόροι του υπολογιστή άσκοπα.

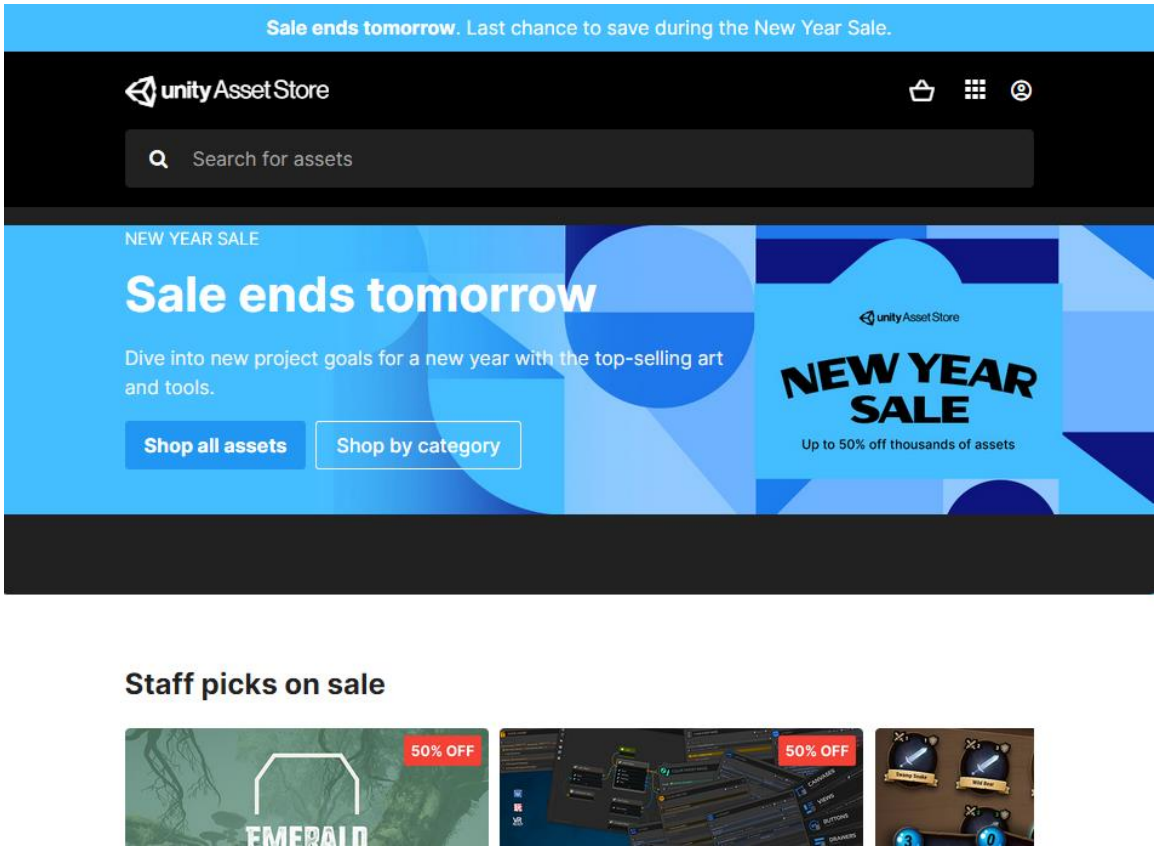
2.2.7 Console

Το παράθυρο Console είναι εκείνο που εμφανίζει σφάλματα, προειδοποιητικά μνήματα και άλλα μηνύματα που δημιουργούνται από την Unity. Αυτό βοηθάει τον προγραμματιστή να αντιληφθεί οτιδήποτε αφορά τον κώδικα που συνδέεται με τα αντικείμενα του παιχνιδιού και όχι μόνο. Είναι από τα σημαντικότερα παράθυρα στην Unity διότι κάνει αρκετά απλή την διαδικασία της απασφαλμάτωσης και προειδοποιεί για προβλήματα που μπορεί να προκύψουν κατά την ανάπτυξη του παιχνιδιού. Επιπρόσθετα δίνει την δυνατότητα στον χρήστη να εμφανίσει τα δικά του μηνύματα για να ανιχνεύσει λάθη κάνοντας χρήση της κλάση Debug.

2.2.8 Asset Store

Η Unity έχει ένα δικό της ηλεκτρονικό κατάστημα που ονομάζεται Asset Store το οποίο περιέχει χιλιάδες δωρεάν ή σε προσιτές τιμές στοιχεία, πράγμα που κάνει τον developer να εξοικονομεί πολύτιμο χρόνο και προσπάθεια από την δουλειά του. Τέτοια στοιχεία μπορεί να είναι τρισδιάστατα και δισδιάστατα μοντέλα, έτοιμα αρχεία κώδικα που υλοποιούν

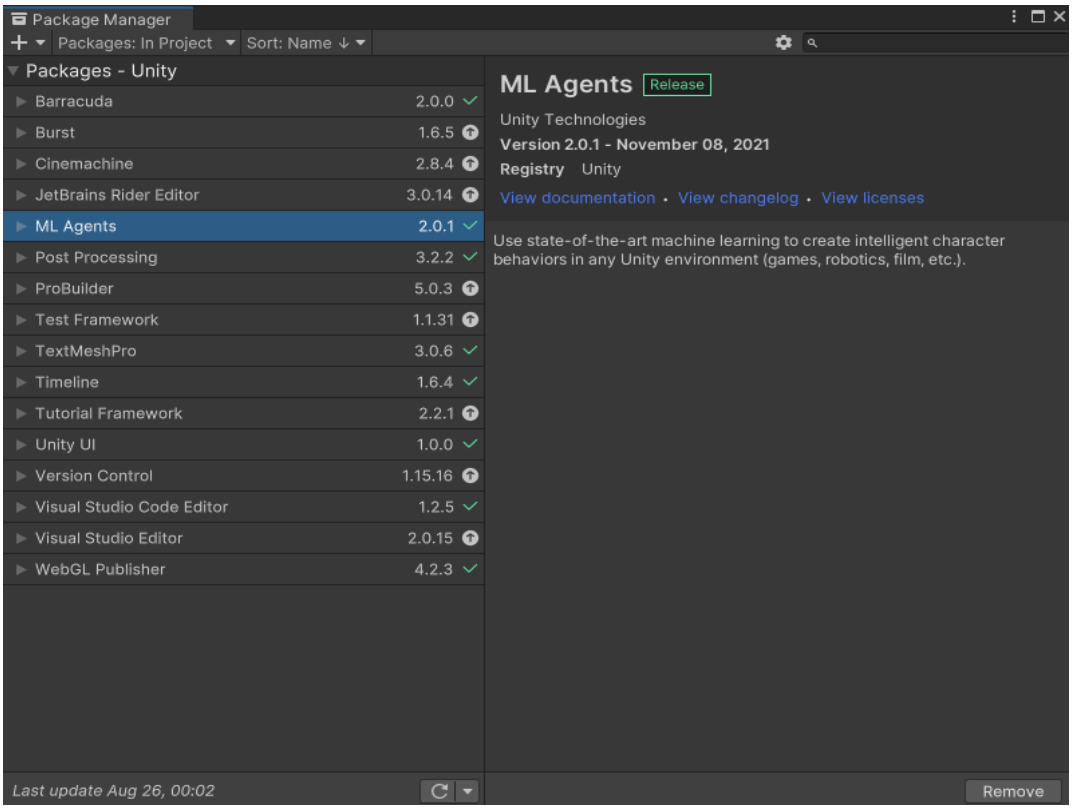
διάφορες καταστάσεις, αρχεία ήχου, εικόνας και αμέτρητα ακόμα.



Εικόνα 19 - Assets Store

2.2.9 Package Manager

Επιπλέον η Unity διαθέτει ένα εσωτερικό Package Manager με τον οποίο ο χρήστης μπορεί να εγκαταστήσει γρήγορα και εύκολα πολλά πακέτα για το project του. Ένα πακέτο είναι ένας συνδυασμός από Assets, Shaders, Textures, Plug-ins, Icons και Scripts με τα οποία ο χρήστης μπορεί να βελτιστοποιήσει πολλά κομμάτια του project του. Ένα από αυτά τα πακέτα είναι και οι **ML-Agents** το οποίο και θα δείξουμε παρακάτω τον τρόπο εγκατάστασής τους.



Εικόνα 20 - Package Manager / ML Agents

2.2.10 Android Build Support

Το μεγαλύτερο πλήθος των βίντεο παιχνιδιών που έχουν αναπτυχθεί για κινητές συσκευές είναι με την Unity. Αν το παιχνίδι μας προοριζόταν για android συσκευή θα πρέπει να λάβουμε υπόψη μας την επιλογή που μας δίνεται για Android Build Support. Αυτόματα η Unity κατεβάζει το SDK (Software Development Kit) και το JDK (Java Development Kit) που είναι απαραίτητα για να δημιουργηθεί το εκτελέσιμο apk αρχείο. Επιπρόσθετα για τους developers που προορίζουν το παιχνίδι τους για Android συσκευές, υπάρχει διαθέσιμη εφαρμογή για android και IOS συσκευές από το Google Play Store και για το App Store αντίστοιχα που ονομάζεται Unity Remote. Αυτή η εφαρμογή συνδέεται με την Unity ενώ εκτελείται το Project στην λειτουργία Play από τον Editor. Η οπτική έξοδος από τον Editor στέλνεται στην οθόνη της συσκευής που είναι συνδεδεμένη μέσω καλωδίου και οι είσοδοι που δίνει ο χρήστης μέσω της συσκευής του, αποστέλλονται πίσω στο τρέχον έργο στη Unity. Αυτό επιτρέπει στο να υπάρχει μια καλή εντύπωση

για το πως θα φαίνεται και χειρίζεται το παιχνίδι στην συσκευή προορισμού, χωρίς την ταλαιπωρία μιας πλήρους δημιουργίας εκτελέσιμου αρχείου κάθε φορά για δοκιμή.

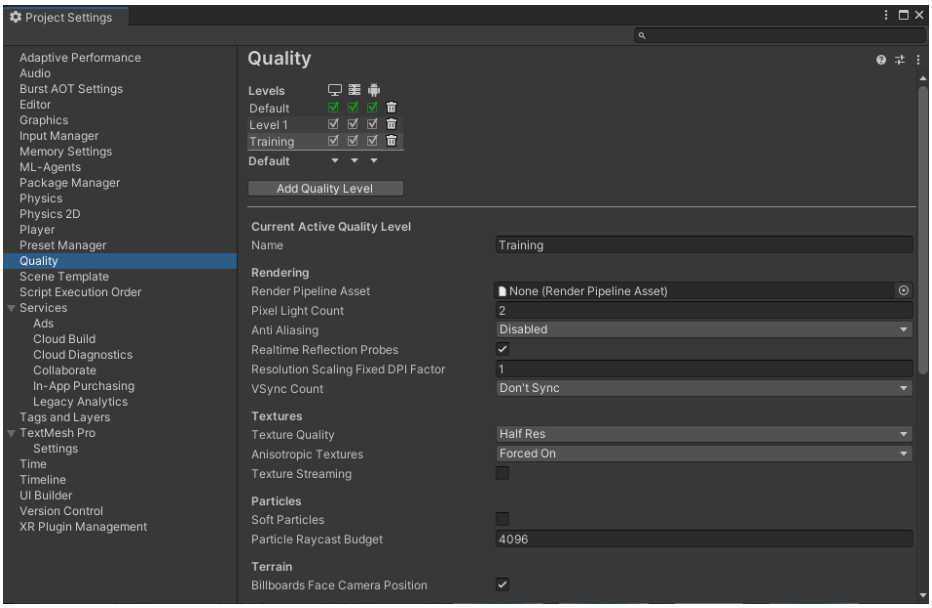
Τα δεδομένα εισόδου από την συσκευή που μεταδίδονται πίσω στον Editor είναι τα εξής:

- Είσοδοι από touch και γραφίδα.
- Επιταχυνσιόμετρο
- Γυροσκόπιο
- Κάμερα συσκευής
- Πυξίδα
- GPS
- Joystick

2.2.11 Project Settings

Επιπλέον η Unity έχει ένα παράθυρο που ονομάζεται Project Setting, μέσα στο οποίο υπάρχει ένας πλήθος από καρτέλες για τις γενικές ρυθμίσεις που σχετίζονται με την Unity. Μία από αυτές είναι για την δημιουργία του εκτελέσιμου αρχείο όπως είναι το όνομα της εφαρμογής, η έκδοση, το όνομα της εταιρία που το δημιουργεί, το εικονίδιο και πολλές ακόμα επιλογές για τους προγραμματιστές της εφαρμογής. Επιπλέον, υπάρχει μια καρτέλα που ονομάζεται Quality η οποία μας επιτρέπει να ορίσουμε το επίπεδο ποιότητας γραφικών που θέλουμε η Unity να αποδώσει. Για κινητές συσκευές ή για παλαιότερο υλικό προτείνεται να στοχεύουμε σε μικρότερη ποιότητα εάν θέλουμε να πετύχουμε υψηλότερα καρέ ανά δευτερόλεπτο. Για να πάμε στο παράθυρο αυτό κάνουμε αριστερό κλίκ στο Edit > Project Settings. Στην παρακάτω φωτογραφία μπορούμε να δούμε τον πίνακα ποιότητας όπου κάτω από αυτόν υπάρχουν οι ρυθμίσεις για το επιλεγμένο επίπεδο

ποιότητας.



Εικόνα 21 - Project Settings

2.2.12 Tags και Layers

Η unity μας παρέχει τρόπους αναγνώρισης των gameobject που χρησιμοποιούμε. Αυτό γίνεται με την χρήση των tag και των Layers. Τα tags είναι ένα όνομα που μπορούμε να δώσουμε στα gameobjects και με αυτόν τον τρόπο να τα χρησιμοποιήσουμε μέσα στον κώδικα για την επίτευξη διάφορων λειτουργιών. Τα Layers είναι η ομαδοποίηση των gameobjects που μοιράζονται συγκεκριμένα χαρακτηριστικά και μπορούμε να τα χρησιμοποιήσουμε μεταξύ άλλων για να περιορίσουμε διάφορες λειτουργίες της unity.

2.3 Τεχνολογίες Unity

Η Unity υποστηρίζει δύο γλώσσες προγραμματισμού για την δημιουργία των Scripts. Την C# και την Javascript όπου και οι δυο είναι αντικειμενοστραφείς. Οι γλώσσες αυτές ενσωματώθηκαν στον Editor μετά από την 5η έκδοση της Unity καθώς μέχρι τότε υποστήριζε την γλώσσα Boo. Για την υλοποίηση του παιχνιδιού που θα δείξουμε παρακάτω έχει γίνει η χρήση της γλώσσας C#. Επιπροσθέτως η μηχανή αυτή μπορεί να υποστηρίξει το μεγαλύτερο πλήθος των τρισδιάστατων προγραμμάτων για την

μοντελοποίηση τρισδιάστατων αντικειμένων όπως είναι 3DSMax, blender, Lightware, XSI, Cinema4D και πολλά ακόμα. Όλα τα παραπάνω εξάγουν τα μοντέλα τους σε αρχεία τέτοια που η Unity μπορεί να υποστηρίξει και να επεξεργαστεί.

2.3.1 Η γλώσσα Προγραμματισμού JavaScript

Η γλώσσα προγραμματισμού JavaScript είναι διερμηνευτική γλώσσα και αρχικά αποτέλεσε μέρος υλοποίησης των φυλλομετρητών ιστού, ώστε πελάτες και χρήστες να μπορούν να ανταλλάσσουν δεδομένα ασύγχρονα και να αλλάζουν δυναμικά το περιεχόμενο του εγγράφου που τους εμφανιζόταν. Είναι μια γλώσσα σεναρίων και η σύνταξη της είναι επηρεασμένη από την γλώσσα C. Αρχικά δημιουργήθηκε από τον Brendan Eich της εταιρίας Netscape με την Επωνυμία Mocha, όπου αργότερα μετονομάστηκε σε JavaScript. Στην Unity τα αρχεία κώδικα έχουν κατάληξη .js.

2.3.2 Η γλώσσα Προγραμματισμού C#

Η γλώσσα προγραμματισμού C# δημιουργήθηκε από την Microsoft μαζί με την πλατφόρμα Visual Studio.NET το 2000 η οποία ήταν παρόμοια με την Java. Η C# είναι μία απλή, μοντέρνα, γενικού σκοπού αντικειμενοστραφής γλώσσα προγραμματισμού. Είναι σχετικά απλή γλώσσα, χρησιμοποιεί κλάσεις και δημιουργεί αντικείμενα τα οποία έχουν μεταβλητές που περιγράφουν τον τύπο του αντικειμένου και τις μεθόδους που περιγράφουν τι κάνει το αντικείμενο. Επίσης παρέχει πολλές ενσωματωμένες λειτουργίες και βιβλιοθήκες που κάνουν την ανάπτυξη πιο γρήγορη. Ο χρόνος σύνταξης και εκτέλεσης της είναι πολύ γρήγορος και είναι ασφαλής που αυτό σημαίνει ότι έχει πρόσβαση μόνο στη θέση μνήμης στην οποία έχει άδεια εκτέλεσης. Η διαδικασία δημιουργίας ενός C# αρχείου στην Unity είναι πολύ απλή. Πλοηγούμαστε στο παράθυρο Projects, κάνουμε δεξί κλικ / create / C# script. Όλα τα C# αρχεία έχουν την κατάληξη .cs.

2.3.3 Monodevelop και Visual Studio

Το Monodevelop είναι ένα πρόγραμμα – ένα περιβάλλον ανάπτυξης το οποίο δίνει την δυνατότητα στον προγραμματιστή να δημιουργήσει κώδικα για την εκτέλεση του παιχνιδιού. Όταν γίνεται εγκατάσταση της Unity υπάρχει η επιλογή της εγκατάστασης και του Monodevelop που μαζί με την Unity υποστηρίζουν τις γλώσσες προγραμματισμού που αναφερθήκαμε προηγουμένως. Επιπλέων η Unity δίνει την δυνατότητα να γίνει η χρήση και του Visual Studio σαν επιλογή για το περιβάλλον ανάπτυξης του κώδικα το οποίο και

θα χρησιμοποιήσουμε στα πλαίσια της πτυχιακής.

2.3.4 Prefabs

Τα Prefabs είναι ένας ειδικός τύπος στοιχείου που επιτρέπει την πλήρη αποθήκευση των GameObjects στο project για επαναχρησιμοποίηση. Αυτά τα στοιχεία στην συνέχεια μπορούν να μοιραστούν μεταξύ σκηνών ή ακόμα και άλλων έργων χωρίς να χρειάζεται να διαμορφωθούν ξανά. Ένα prefab που χρησιμοποιείται στην ιεραρχία είναι ένα αντίγραφο του prefab που βρίσκεται στο παράθυρο του Project. Αυτό σημαίνει ότι οποιαδήποτε αλλαγή και αν εφαρμοστεί στο αντικείμενο, μεταδίδεται και στο αρχικό prefab. Επιπροσθέτως μπορούμε να δημιουργήσουμε ένα αντικείμενο μέσα από στο άλλο, κάνοντας το απόγονο. Ένα αντικείμενο απόγονος παίρνει τα χαρακτηριστικά του πρόγονου του όπως είναι οι συντεταγμένες και άλλα που μπορεί να έχουμε ορίσει εμείς από πριν.

Τα Prefabs μοιάζουν αρκετά με άλλα αντικείμενα που εμφανίζονται στο παράθυρο Project, ωστόσο ο τύπος αρχείου τους είναι .prefab. Στο παράθυρο της ιεραρχίας τα αντικείμενα αυτά αντιπροσωπεύονται με μπλε κείμενο και μπλε κύβο.

Κάθε Prefab που τοποθετούμε στον κόσμο μας έχει ένα component υπεύθυνο για την θέση του και το μέγεθος του. Αυτό είναι το Transform. Όπως βλέπουμε και στην παρακάτω εικόνα ένα αντικείμενο αυτόματα παίρνει τιμές από το περιβάλλον όπως είναι η θέση του, η περιστροφή και το μέγεθος του. Αυτές οι τιμές βέβαια αλλάζουν με βάση την δική μας αρέσκεια.

Εικόνα 22 - Transform

Στο παιχνίδι που έχουμε αναπτύξει, έχει γίνει χρήση διαφόρων αντικειμένων των οποίων και θα αναλύσουμε, καθώς και το πως ενσωματώνουμε κομμάτια κώδικα που ελέγχουν όλες τις λειτουργίες τους.

Κεφάλαιο 3: Ανάπτυξη παιχνιδιού

3.1 Στόχος

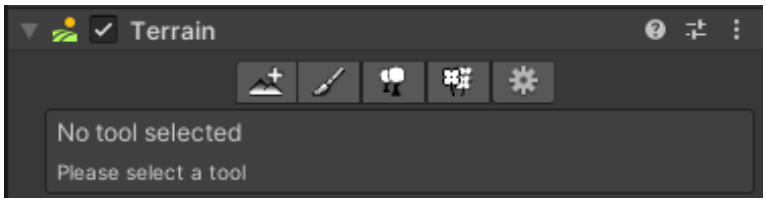
Ο σκοπός της παρούσας πτυχιακής είναι η κατανόηση των βασικών λειτουργιών της μηχανικής μάθησης μέσα από τους ML-Agents. Σαν βασικός στόχος είναι ο πράκτορας μας, που στην συγκεκριμένη περίπτωση είναι το αυτοκίνητο μας, να καταφέρει να ολοκληρώσει την πίστα που έχουμε δημιουργήσει σε όσο το δυνατόν γρηγορότερο χρόνο και αποφεύγοντας ταυτόχρονα τα εμπόδια που υπάρχουν στο δρόμο του. Είτε αυτά είναι άλλα αυτοκίνητα, είτε άλλα εμπόδια. Τον τρόπο εκπαίδευσης του θα τον αναλύσουμε με λεπτομέρειες στο τέταρτο κεφάλαιο .

3.2 Ανάπτυξη περιβάλλοντος

Για την ανάπτυξη και τη δημιουργία του περιβάλλοντος θα κάνουμε χρήση σχεδόν όλων των παραθύρων που αναφέραμε παραπάνω και θα δούμε αναλυτικά την πορεία δημιουργίας του. Επίσης θα δούμε τι αντικείμενα έχουμε χρησιμοποιήσει καθώς και πως θα τα δημιουργούμε σαν οντότητες για το παιχνίδι μας. Στο panel της ιεραρχίας έχουμε δημιουργήσει ένα empty object το οποίο το έχουμε ονομάσει Training Set. Στο εσωτερικό του υπάρχουν όλα όσα είναι απαραίτητα για την εκπαίδευση του Agent μας. Αυτή την τεχνική την εφαρμόζουμε ώστε όταν έρθει η στιγμή της εκπαίδευσης να μπορέσουμε, κάνοντας copy-paste πολλαπλά Training Set, να επιταχύνουμε το χρόνο εκμάθησης. Το συγκεκριμένο περιβάλλον που έχει φτιαχτεί στα πλαίσια της πτυχιακής εργασίας αποτελείται από ένα prefab που θα είναι το αυτοκίνητό δηλαδή ο ευφυής πράκτορας, την πίστα-δρόμο στην οποία θα κάνουμε την εκπαίδευση και το εξωτερικό περιβάλλον. Θα κρατήσουμε απλό και ελαφρύ το περιβάλλον για να μας βοηθήσει να κάνουμε την εκπαίδευση ποιο γρήγορα. Αυτό εξαρτάται από την δύναμη του υπολογιστή, γιατί όπως θα αναλύσουμε και στο επόμενο κεφάλαιο δεν θα είναι μόνο ένα περιβάλλον στο οποίο θα γίνει η εκπαίδευση.

3.2.1 Terrain

Το Terrain αποτελεί την βάση του περιβάλλοντος μας καθώς είναι εκείνο πάνω στο οποίο θα χτίσουμε όλα τα υπόλοιπα. Όταν ο χρήστης πατήσει την επιλογή δημιουργίας ενός εδάφους αυτόματα δημιουργείται ένα τετράγωνο έδαφος με συγκεκριμένες διαστάσεις ορισμένες από την Unity. Ο χρήστης μπορεί να επεξεργαστεί όπως θέλει το έδαφος αυτό κάνοντας χρήση των εργαλείων που του παρέχονται. Τα εργαλεία αυτά είναι τα εξής:



Εικόνα 23 - Terrain Settings

- Το πρώτο εργαλείο ονομάζεται **CREATE NEIHBOR TERRAIN** και δίνει την δυνατότητα στον χρήστη να προσθέσει γειτονικά εδάφη επεκτείνοντας έτσι το έδαφος που έχει φτιάξει.

- Το δεύτερο εργαλείο ονομάζεται **PAINT TERRAIN** και είναι το ποιο σημαντικό εργαλείο διότι παρέχει επιλογές επεξεργασίας του εδάφους όπως είναι η ανύψωση του, το ζωγράφισμα του με textures της αρεσκείας του, την διαμόρφωση του όπως για παράδειγμα την δημιουργία ενός βουνού, ενός ποταμού, να ορίσει ύψη και άλλα πολλά.

- Τρίτο εργαλείο ονομάζεται **PAINT TREES** και είναι για την προσθήκη δέντρων. Ο Χρήστης μπορεί να χρησιμοποιήσει τα έτοιμα τρισδιάστατα δέντρα που έχει η Unity, αλλά μπορεί επίσης να δημιουργήσει δικά του αν ο ίδιος το επιθυμεί.

- Το τέταρτο εργαλείο ονομάζεται **PAINT DETAILS** και χρησιμοποιείται για την προσθήκη όλων των υπόλοιπων λεπτομερειών όπως είναι το γρασίδι και για οποιοδήποτε άλλο τρισδιάστατο ή και δισδιάστατο μοντέλο ο χρήστης επιθυμεί να χρησιμοποιήσει.

- Το τελευταίο εργαλείο ονομάζεται **TERRAIN SETTINGS** και είναι για όλες τις γενικές ρυθμίσεις του εδάφους.

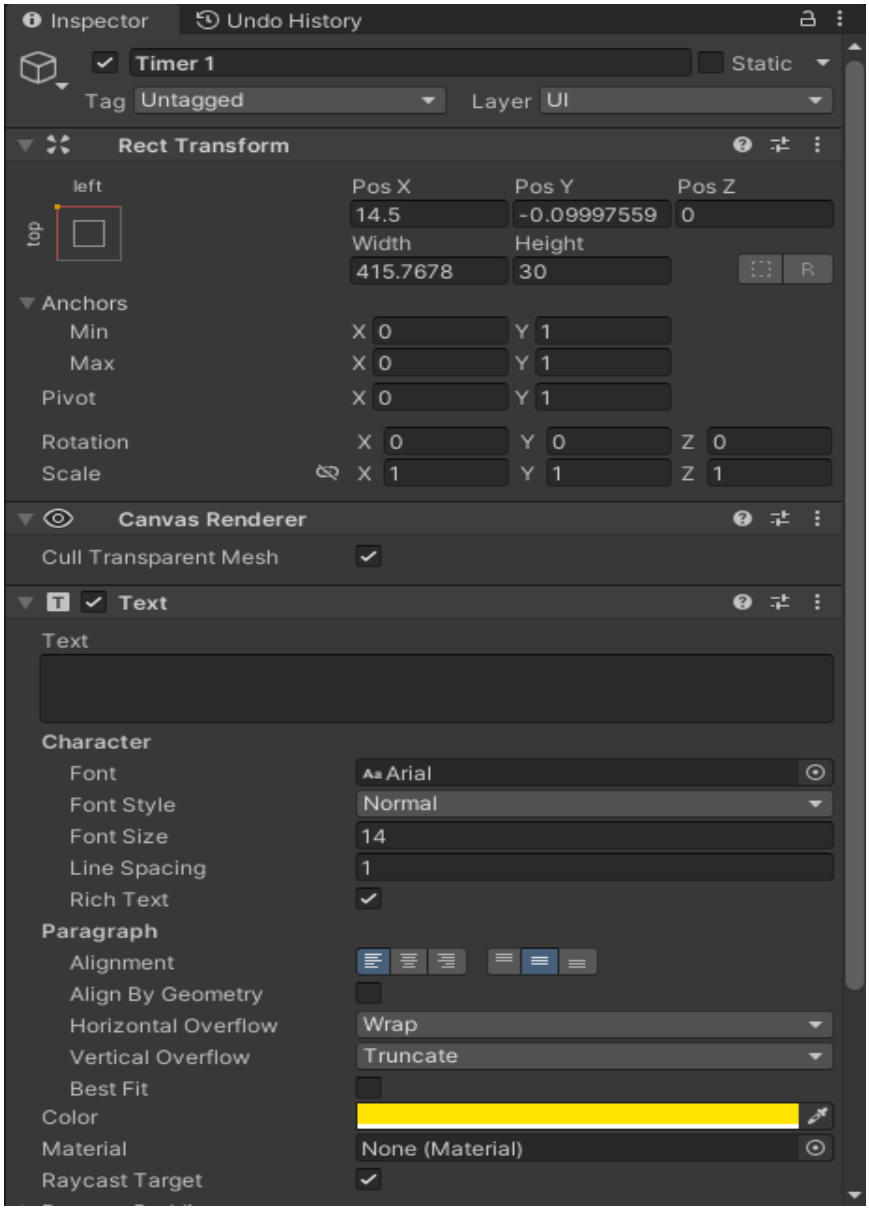
3.2.2 Canvas

Το canvas είναι ένα από τα βασικότερα στοιχεία στην Unity εάν θέλουμε να φτιάξουμε οποιοδήποτε αντικείμενο που θέλουμε να εμφανίσουμε στην οθόνη. Περιέχει όλα τα texts, εικόνες, βίντεο και άλλα γραφικά στοιχεία. Κάθε UI στοιχείο που θα δημιουργηθεί πρέπει να είναι παιδί του canvas για να πάρει όλες τις ρυθμίσεις και να εμφανιστεί. Για να δημιουργήσουμε ένα canvas αρκεί να πάμε στο παράθυρο της ιεραρχίας, και με δεξί κλικ επιλέγουμε UI -> Canvas. Τα UI στοιχεία εμφανίζονται με την σειρά που υπάρχουν στο παράθυρο της ιεραρχίας.

Για το παιχνίδι μας έχουμε φτιάξει δώδεκα texts, δύο για κάθε Agent. Όπως θα δούμε και

στην συνέχεια, θέλουμε να μετράμε το χρόνο που κάνει κάθε Agent στην πίστα και να τον εμφανίζουμε στην οθόνη. Τα πρώτα έξι text θα εμφανίζουν τον χρόνο που κάνουν από το ξεκίνημα του γύρου μέχρι να τον ολοκληρώσει και τα υπόλοιπα έξι θα κρατάνε τον καλύτερο χρόνο για τον κάθε ένα Agent. Ο έλεγχος του χρόνου καθώς και το όνομα που παίρνει το κάθε text για τον Agent θα δούμε πώς υλοποιήθηκε στην συνέχεια.

Κάθε text έχει τις δικιές του παραμέτρους όπως το μέγεθος της γραμματοσειράς, το χρώμα, την θέση και πολλές ακόμα επιλογές. Το χρώμα που έχουμε ορίσει αντιστοιχεί στο χρώμα του αυτοκινήτου του Agent για να είναι ξεκάθαρο για ποιον Agent αντιστοιχεί το κάθε ένα.



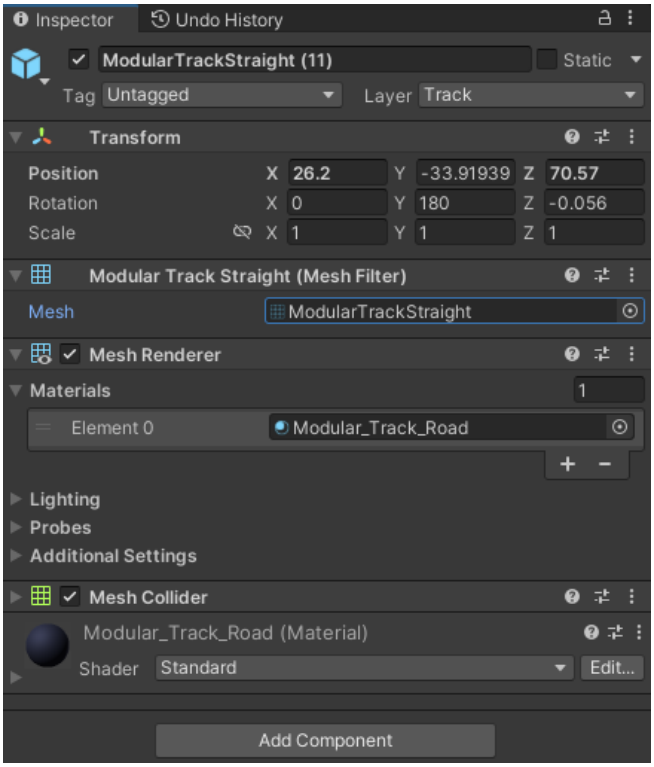
Εικόνα 24 - Canvas,Text

3.2.3 Track

Ο δρόμος που δημιουργήσαμε για την αποτελείται από διάφορα ξεχωριστά κομμάτια όπως είναι ευθείες, στροφές ανηφόρες και κατηφόρες. Να σημειωθεί ότι τα prefab του δρόμου δεν τα έχουμε δημιουργήσει εμείς καθώς τα παρέχει η Unity από ένα άλλο sample project που έχουν αναπτύξει οι ίδιοι και είναι δωρεάν προς λήψη. Στο track έχουμε ορίσει ως tag το όνομα road. Αφότου έχουμε δημιουργήσει το επιθυμητό μας επίπεδο, όλα τα κομμάτια από τα οποία αποτελείται, τα έχουμε ομαδοποιήσει σε ένα Empty object το οποίο το έχουμε ονομάσει TRACK να είναι ποιο οργανωμένα. Το κάθε ένα κομμάτι στην πίστας έχει κάποια components για να μπορεί να είναι λειτουργικό. Πέρα από το βασικό component που

είναι το transform που έχουμε αναφέρει, έχει άλλα τρία όπως φαίνεται στην εικόνα 24.

- Το Mesh Filter είναι οπτικός σχεδιασμός του δρόμου. Ένα τρισδιάστατο αντικείμενο όταν δημιουργείται έχει κάποια χαρακτηριστικά. Το βασικότερο είναι από πόσα πολύγωνα αποτελείται και το πως είναι σχεδιασμένα. Αυτό δίνει την τελική εμφάνιση στο αντικείμενο.
- Το επόμενο component που έχει είναι το Mesh Renderer. Σε αυτό το component μπορούμε να βάλουμε ένα η συνδυασμό από materials. Ένα material είναι ο χρωματικός σχεδιασμός του αντικειμένου. Η Unity έχει εργαλεία δημιουργίας ενός material, αλλά εμείς θα κρατήσουμε το υπάρχον που έχει το αντικείμενο γιατί καλύπτει τις ανάγκες μας.
- Τελευταίο component είναι το Mesh Collider. Χωρίς αυτό ο δρόμος μας θεωρείται "αόρατος" δηλαδή δεν υπάρχει έλεγχος για συγκρούσεις

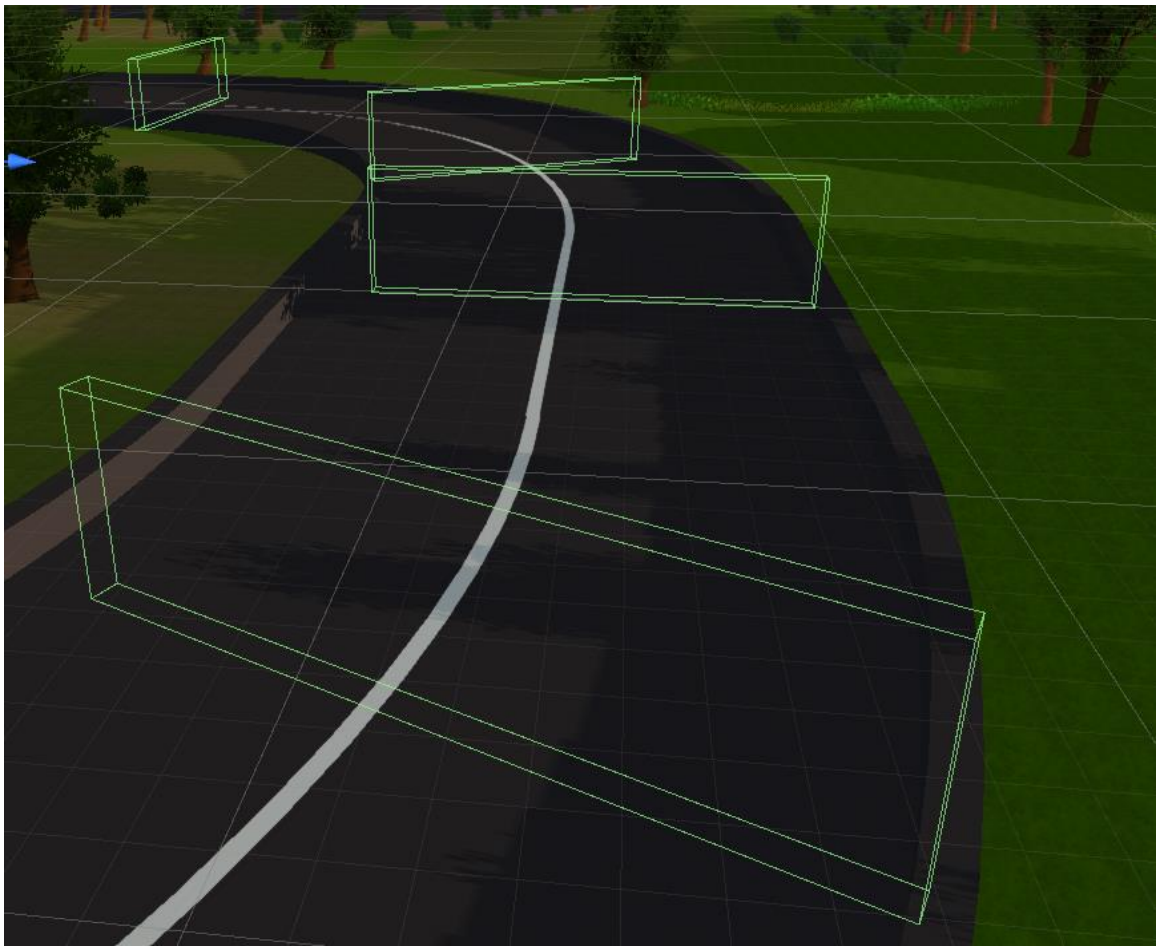


Εικόνα 25 - Track Components

3.2.3 Checkpoints

Για να μπορέσουμε να εκπαιδεύσουμε τον πράκτορα μας ώστε να μάθει να οδηγεί στην πίστα πρέπει να φτιάξουμε κάποια σημεία στα οποία κάθε φορά που θα τα περνάει να επιβραβεύεται. Για την δημιουργία ενός checkpoint έχουμε πάει στο panel της ιεραρχίας και έχουμε δημιουργήσει ένα empty object που το έχουμε ονομάσει Checkpoint. Ύστερα

κάνουμε copy-paste όσα πιστεύουμε ότι είναι αρκετά ώστε να καλύψουν όλο το επίπεδο της πίστας όπως βλέπουμε στην εικόνα 26. Κάθε checkpoint έχει ένα C# script που θα αναλύσουμε στο επόμενο κεφάλαιο και ένα Box Collider με την επιλογή IsTrigger ενεργοποιημένη, με διαστάσεις όσο και το πλάτος του του δρόμου. Δεν χρειαζόμαστε κάποιο Mesh Renderer στην συγκεκριμένη περίπτωση διότι δεν μας ενδιαφέρει τα checkpoints να είναι ορατά σε εμάς παρά μόνο στον πράκτορα μας. Επιπλέον κάθε checkpoint έχει Tag με το όνομα checkpoint, ώστε να μπορέσουμε να το αναφέρουμε στα scripts που θα χρησιμοποιήσουμε.

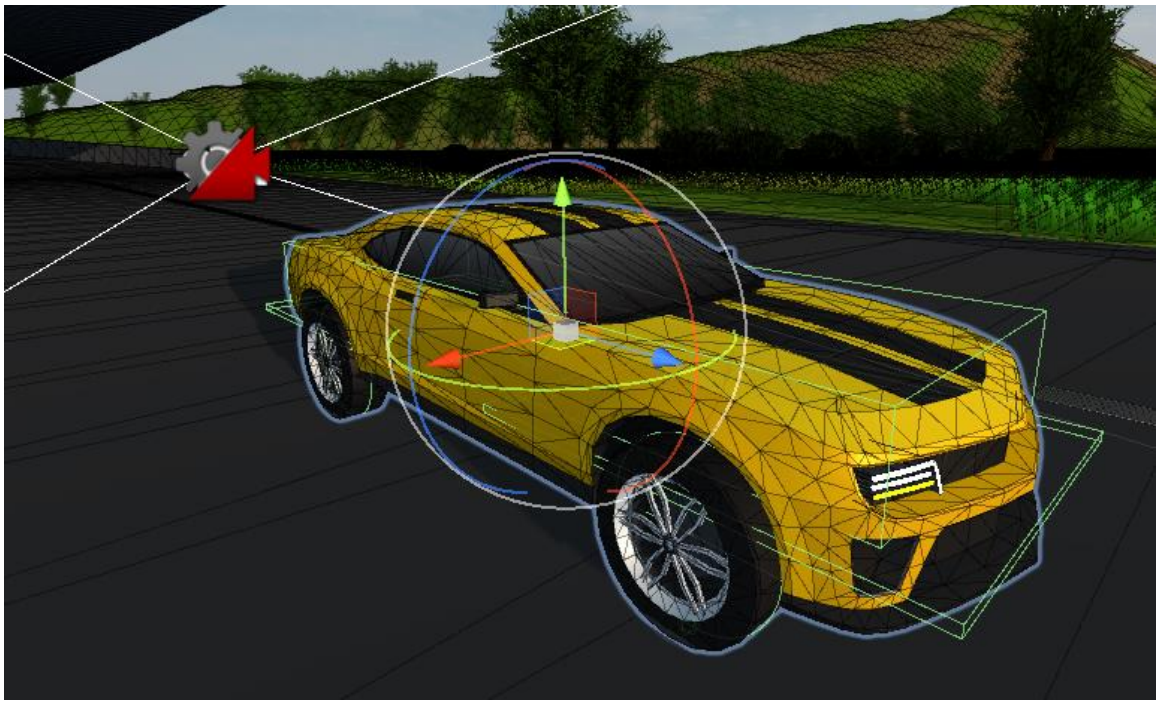


Εικόνα 26 - Checkpoints

Όλα τα Checkpoints που έχουμε δημιουργήσει είναι παιδιά ενός empty object που το έχουμε ονομάσει checkpoints, ώστε να είναι σωστά οργανωμένα και ομαδοποιημένα. Η λειτουργία τους καθορίζεται από δυο Scripts, το Trackcheckpoints που βρίσκεται στο TrainingSet και το Checkpoint Script που όπως αναφέραμε υπάρχει σε κάθε ένα checkpoint.

3.3 AI Agent

Στην εικόνα 27 βλέπουμε το μοντέλο του χαρακτήρα μας. Παρακάτω θα δούμε όλα τα Components που έχουμε χρησιμοποιήσει για την δημιουργία του πράκτορα μας.

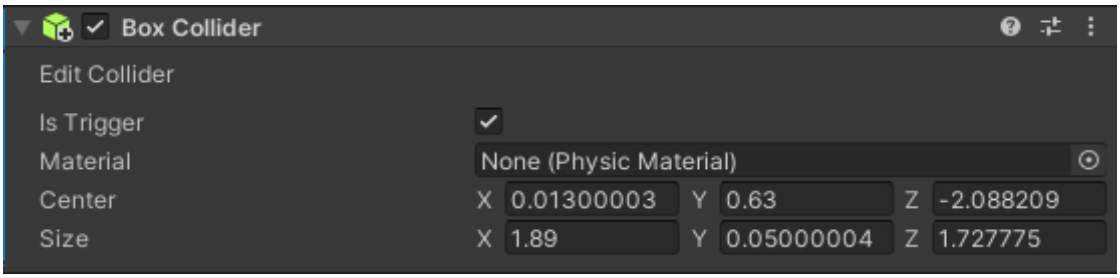


Εικόνα 27 - AI Agent

- Το πράσινο περίγραμμα που βλέπουμε στην παραπάνω εικόνα ονομάζεται **Box Collider**. Με τα Colliders μπορούμε να καθορίσουμε το σχήμα ενός αντικειμένου και είναι αυτοί που επιτρέπουν στην μηχανή φυσικής να χειρίζεται τις συγκρούσεις. Μπορούμε έτσι να προσδιορίσουμε ποια αντικείμενα θα συγκρουστούν μεταξύ τους ή πως αυτά θα συμπεριφερθούν στην περίπτωση σύγκρουσης. Οι απλούστεροι και λιγότερο απαιτητικοί 3D Colliders από άποψη επεξεργαστικής ισχύος είναι οι εξής: Box Collider, Sphere Collider και Capsule Collider. Για 2D παιχνίδια υπάρχουν οι αντίστοιχοι Colliders με την κατάληξη 2D.

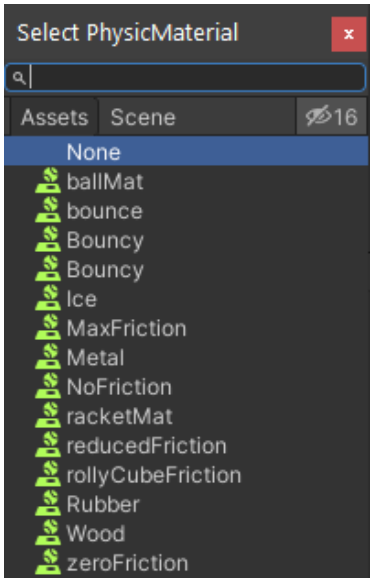
Κάθε Collider έχει κάποιες ειδικές ρυθμίσεις. Οι πιο βασικές είναι οι παρακάτω: **To Edit Collider** που με αυτό δίνουμε τις διαστάσεις που θέλουμε εμείς όπως το μήκος, το πλάτος και το ύψος, το **Is Trigger** και το **Material (Physics Material)**. Ο σκοπός του Is Trigger είναι να προκαλέσει συμβάντα όταν επικαλύπτονται περισσότερα από ένα αντικείμενα. Εάν κάποιο αντικείμενο έχει Collider με την επιλογή Is Trigger, τότε το αντικείμενο αυτό δεν συγκρούεται με κανένα άλλο, αλλά επικαλύπτεται αυτού. Θα αναφέρουμε

λεπτομερειακά αυτή την επιλογή παρακάτω που θα αναφερθούμε στην ανάπτυξη κώδικα.



Εικόνα 28 - Box Collider

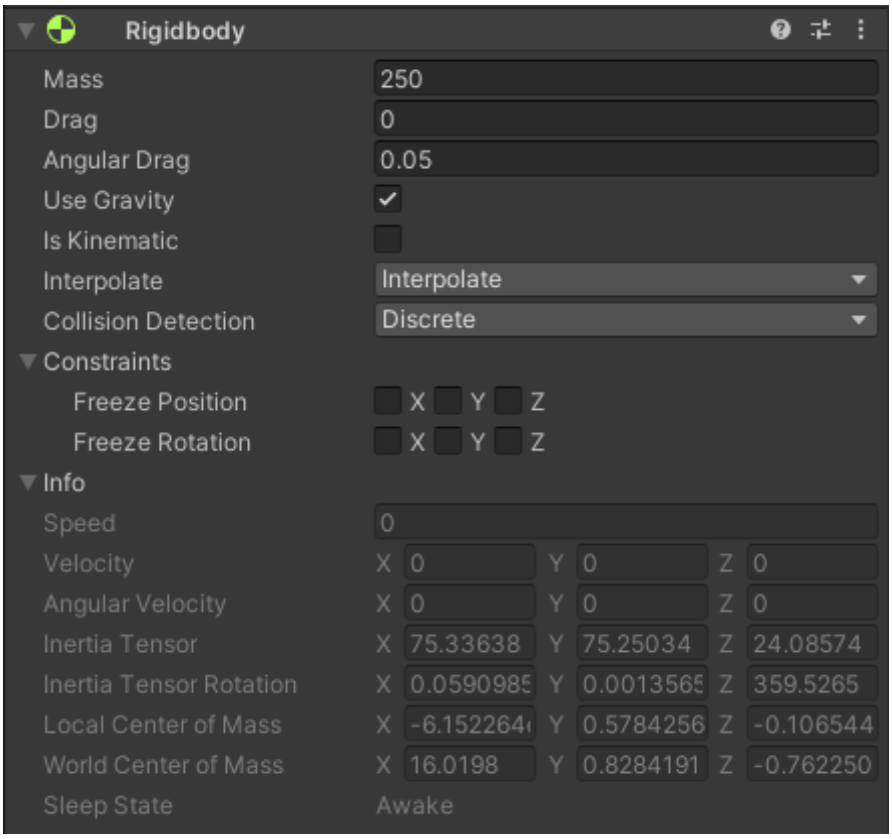
Το Physics Material δίνει την δυνατότητα σε ένα αντικείμενο να συμπεριφερθεί όπως και στην πραγματικότητα. Όταν κάνουμε κλικ στην επιλογή Material εμφανίζονται διάφορες επιλογές όπως η αναπήδηση, η τριβή και πολλά ακόμα. Αυτό είναι πολύ χρήσιμο όταν έχουμε ένα αντίστοιχο περιβάλλον και θέλουμε να κάνουμε αληθοφανές την αλληλεπίδραση του χαρακτήρα αυτό.



Εικόνα 29 - Physics Material

- Επόμενο βασικό Component που έχουμε στον χαρακτήρα μας είναι η **Camera**. Χωρίς αυτήν δεν υπάρχει καμία οπτική έξοδος στην οθόνη. Η κάμερα έχει επιλογές όπως τι θα γίνεται Render σε πραγματικό χρόνο , το FOV (Field Of View), το πόσο μακριά και πόσο κοντά θα βλέπει η κάμερα και άλλες ρυθμίσεις που αφορούν τα γραφικά. Μπορούν να τοποθετηθούν παραπάνω από μία κάμερες στην σκηνή και να γίνετε η εναλλαγή μεταξύ αυτών είτε μέσω κάποιου Trigger που αναφέραμε παραπάνω ή με οποιοδήποτε άλλο έλεγχο μέσω κώδικα που επιθυμεί ο developer.

- Ένα από τα βασικότερα Components για τον χειρισμό του πράκτορά μας καθώς και άλλων αντικειμένων είναι το Rigidbody. Το Rigidbody δίνει την δυνατότητα σε ένα αντικείμενο να ενεργεί υπό τον έλεγχο της φυσικής. Λαμβάνει δυνάμεις και ροπές από το περιβάλλον και αυτό έχει ως αποτέλεσμα τα αντικείμενα αυτά να κινούνται με πιο ρεαλιστικό τρόπο. Επιπρόσθετος ακόμα και χωρίς την χρήση κώδικα, τα αντικείμενα αυτά θα επιδράσουν με την βαρύτητα και θα συγκρουστούν με άλλα αντικείμενα εάν υπάρχει και το σωστό Collider. Στην φωτογραφία που βλέπουμε παρακάτω εμφανίζονται οι βασικές ιδιότητες του.



Εικόνα 30 - Rigidbody

- **Mass:** Η μάζα του αντικειμένου σε κιλά (από προεπιλογή)
- **Drag:** Πόση είναι η αντίσταση του αέρα που επηρεάζει το αντικείμενο όταν αυτό κινείται από δυνάμεις. Παίρνει τιμές από 0 έως άπειρο, όπου 0 σημαίνει ότι δεν υπάρχει αντίσταση στον αέρα και άπειρο ότι το αντικείμενο θα σταματήσει να κινείται αμέσως.
- **Angular Drag:** Πόση είναι η αντίσταση του αέρα που επηρεάζει το αντικείμενο κατά την περιστροφή από ροπή. Η τιμή 0 σημαίνει ότι δεν υπάρχει αντίσταση στον αέρα, ενώ δίνοντας την τιμή άπειρο πρέπει να λάβουμε υπόψη μας ότι το αντικείμενο δεν θα σταματήσει να περιστρέφεται.

- **Use Gravity:** Εάν η επιλογή αυτή είναι ενεργοποιημένη, το αντικείμενο επηρεάζεται από την βαρύτητα.
- **Is Kinematic:** Όταν η επιλογή αυτή είναι ενεργοποιημένη, το αντικείμενο δεν θα επηρεάζεται από την μηχανή φυσικής, αλλά μπορεί να χειριστεί μόνο το Transform του.
- **Interpolate:** Διορθώνει διάφορες ανωμαλίες που μπορεί να υπάρχουν στην κίνηση του αντικειμένου
- **Collision Detection:** Χρησιμοποιείται στο να εμποδίσουμε αντικείμενα τα οποία κινούνται με μεγάλη ταχύτητα από το να περάσουν μέσα από άλλα αντικείμενα, χωρίς έλεγχο σύγκρουσης.
- **Constraints:** Περιέχει επιλογές για περιορισμό στην κίνηση του αντικειμένου.
    - **Freeze Position:** Σταματάει το αντικείμενο να κινείται στον κόσμο στον άξονα X,Y και Z επιλεκτικά.
    - **Freeze Rotation:** Σταματάει το αντικείμενο να περιστρέφεται γύρο από τους τοπικούς άξονες X,Y και Z επιλεκτικά.

3.3.1 WheelCollider

Το wheelCollider είναι ένα ειδικό component στην unity καθαρά υπεύθυνο για τον έλεγχο ενός τροχού. Έχει ενσωματωμένο ένα σύστημα ελέγχου σύγκρουσης το οποίο υπολογίζει την τριβή ανεξάρτητα από την μηχανή φυσικής, χρησιμοποιώντας ένα μοντέλο τριβής με βάση την ολίσθηση. Επειδή έχει δικό του σύστημα ελέγχου σύγκρουσης, αγνοεί οποιοδήποτε Physics Material όπως αναφέραμε προηγούμενος. Η προσομοίωση διαφορετικών τύπου δρόμου γίνεται αλλάζοντας δυο μεταβλητές, το forwardFriction και το sidewaysFriction με βάση το υλικό που χτυπάει ο τροχός. Αυτό μας επιτρέπει να έχουμε μια ποιο ρεαλιστική συμπεριφορά του οχήματος. Επιπλέον προσομοιώνει το σύστημα ανάρτησης των ελατηρίων. Η ανίχνευση σύγκρουσης του τροχού πραγματοποιείται με μία ακτίνα που επεκτείνεται από το κέντρο του τροχού, προς τα κάτω στον τοπικό άξονα y. Ο τροχός ελέγχεται με βάση τις 3 παρακάτω ιδιότητες:

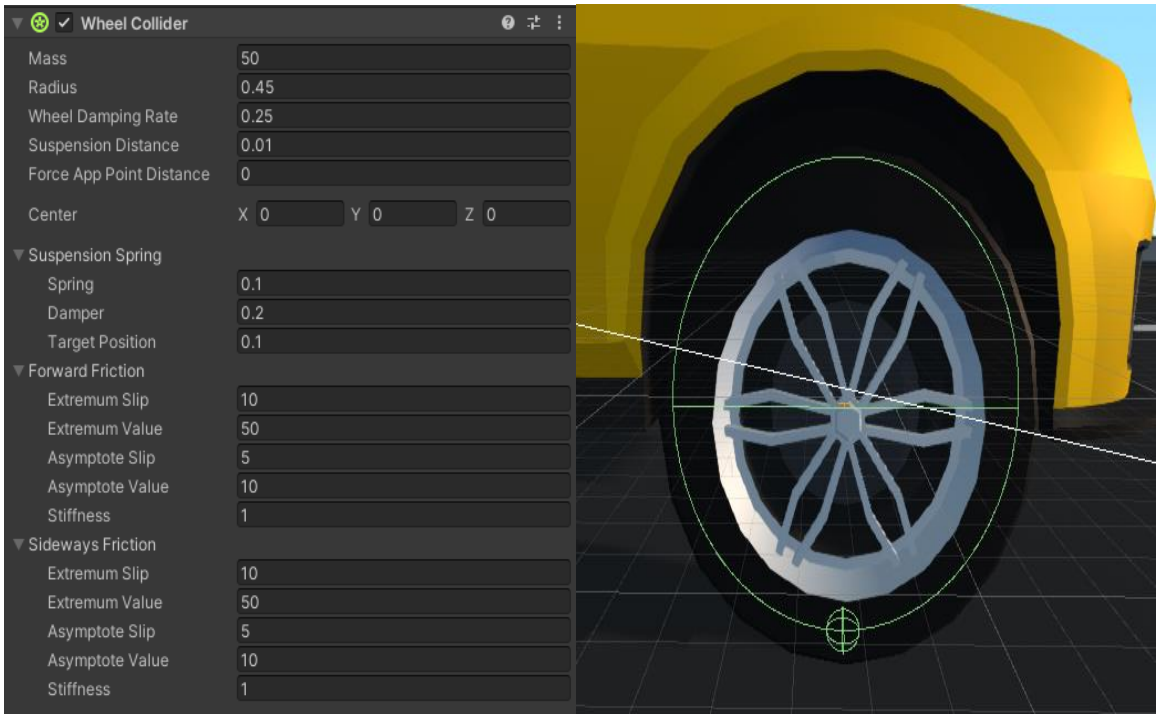
Την ροπή του κινητήρα, την ροπή του φρένου και την γωνία διεύθυνσης. Η unity μας παρέχει πολλές ιδιότητες στο WheelCollider, εμείς θα δούμε και θα αναλύσουμε αυτά που χρησιμοποιήσαμε για το όχημα μας.

Στην εικόνα 31 μπορούμε να δούμε τις ρυθμίσεις για κάθε ένα από τα τέσσερα Wheel Collider που χρησιμοποιούμε.

- Mass: Είναι η μάζα του τροχού σε κιλά. Η τιμή του πρέπει να είναι μεγαλύτερη του μηδενός. Στην δική μας περίπτωση η τιμή είναι 50.
- Radius: Είναι η ακτίνα του τροχού με τιμή 0.45. Η τιμή 0.45 εφάπτεται ακριβώς με το μέγεθος του τροχού μας.
- Wheel Damping Rate: Είναι η ευαισθησία που έχει η ανάρτηση του οχήματός μας.

Αυτή η μεταβλητή είναι ανάλογη με το βάρος που θα δώσουμε στο όχημα.

- Suspension Distance: Είναι η απόσταση επιμήκυνσης της ανάρτησης των τροχών, μετρούμενη από το κέντρο του τροχού και επεκτείνεται προς τα κάτω στον άξονα y.
- Force App Point Distance: Αυτή η παράμετρος καθορίζει το σημείο στο οποίο θα εφαρμοστούν οι δυνάμεις στον τροχό. Στην περίπτωση του οχήματος μας, η τιμή αυτή είναι 0 που δηλώνει ότι οι δυνάμεις θα εφαρμοστούν στην βάση του τροχού.
- Center: Δηλώνει σε τοπικό χώρο την τοποθεσία του Collider.
- Suspension Spring: Δηλώνει την προσπάθεια της ανάρτησης να φτάσει ένα στόχο. Στο μοντέλο μας δεν έχουμε βάλει κάποιο στόχο οπότε αφήνουμε τις μεταβλητές ως έχουν.
- Forward/Sideways Friction: Ιδιότητες για την τριβή του λάστιχου, όταν ο τροχός κινείται είτε μπροστά-πίσω είτε πλάγια.



Εικόνα 31 - Wheel Collider

3.3.2 CarMovement Script

Για τον έλεγχο και την κίνηση του αυτοκινήτου μας έχουμε αναπτύξει ένα C# Script με όνομα κλάσης CarMovement. Οι τρόποι για την κίνηση ενός αυτοκινήτου στην Unity ποικίλουν και η χρήση τους γίνεται ανάλογα με τις ανάγκες του προγραμματιστή. Στη δική μας περίπτωση η κίνηση του αυτοκινήτου γίνεται κυρίως με το σύστημα φυσικής της unity. Ξεκινάμε δηλώνοντας τους τύπους και τις μεταβλητές που θα χρειαστούμε. Δηλώνουμε αρχικά τις μεταβλητές για τους τροχούς, μια μεταβλητή τύπου WheekCollider, που παίρνει τις ιδιότητες που αναφέραμε προηγούμενος καθώς και μια μεταβλητή τύπου transform που επιστρέφει την τοποθεσία του αντικειμένου.

```
public class CarMovement : MonoBehaviour
{
    public WheelCollider WheelFL;
    public WheelCollider WheelFR;
    public WheelCollider WheelRL;
    public WheelCollider WheelRR;
    public Transform WheelFLtrans;
    public Transform WheelFRtrans;
    public Transform WheelRLtrans;
    public Transform WheelRRtrans;
```

Στην συνέχεια δηλώνουμε τις μεταβλητές για την κίνηση του αυτοκινήτου.

```
    private bool braked = false;
    public float maxTorque;
    public float steering;
    private Rigidbody rb;
    public Transform centreofmass;
```

Αφού δηλώσαμε τις μεταβλητές προχωράμε στην αρχικοποίηση κάποιων παραμέτρων στην μέθοδο Start() της unity, η οποία εκτελείται μία φορά κατά την δημιουργία του αντικειμένου.

```
void Start()
    {
        FLWheel = true;
        FRWheel = true;
        rb = GetComponent<Rigidbody>();
        rb.centerOfMass = centreofmass.transform.localPosition;
    }
```

Ο τύπος rigidbody επιστρέψει όλες τις ιδιότητες που αναφέραμε στην αντίστοιχη ενότητα. Το centerofmass είναι μια μέθοδος της unity από το rigidbody που θα χρησιμοποιηθεί παρακάτω.

Για την κίνηση του αυτοκινήτου μας δημιουργούμε την μέθοδο Move. Η μέθοδος αυτή ελέγχει την επιτάχυνση και την κατεύθυνση του οχήματος. Οι παράμετροι acceleration και turning είναι τιμές που λαμβάνει από την κλάση AiAgent.

```
public void Move(float acceleration, float turning)
{
    //speed of car, car will move as we will provide the input to it.
```

```
    WheelFL.motorTorque = maxTorque * acceleration;
    WheelFR.motorTorque = maxTorque * acceleration;
//Here we are changing the steer angle of the front tyres of the car so that we
can change the car direction.
    WheelFL.steerAngle = steering * turning;
    WheelFR.steerAngle = steering * turning;
 }
```

Κεφάλαιο 4: AI ML-Agents

4.1 Εισαγωγή

Η μηχανική μάθηση είναι ένας κλάδος της τεχνητής νοημοσύνης που εστιάζεται στην εκμάθηση μοτίβων από δεδομένα. Υπάρχουν τρείς βασικοί αλγόριθμοι πάνω στους οποίους βασίζεται η μηχανική μάθηση.

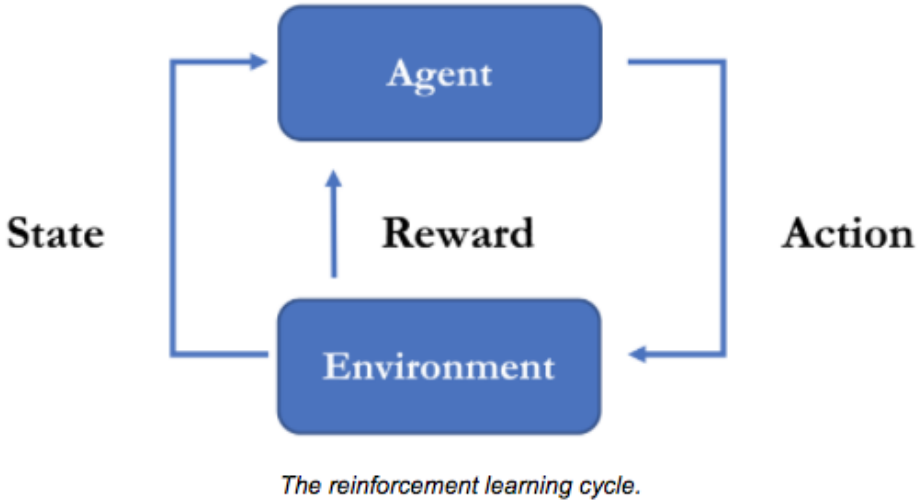
- Unsupervised Learning (Μάθηση χωρίς επίβλεψη)
- Supervised Learning (Μάθηση με επίβλεψη)
- Reinforcement Learning (Ενισχυτική Μάθηση)

Ο στόχος της μάθησης χωρίς επίβλεψη είναι να ομαδοποιήσει παρόμοια στοιχεία σε ένα σύνολο δεδομένων. Όπως αναφέρεται και στο όνομα, είναι μάθηση χωρίς την ανθρώπινη παρέμβαση. Αρχικά παρέχετε στο AI ένα σύνολο δεδομένων που υπάρχει ήδη. Το AI μελετά και αναλύει τα δεδομένα εισόδου, μεγάλο μέρος των οποίων είναι χωρίς ετικέτα, δηλαδή χωρίς περιγραφή και μη δομημένα, και βρίσκει μοτίβα και σχέσεις μεταξύ των δεδομένων αυτών. Η μάθηση χωρίς επίβλεψη βασίζεται στο πως οι άνθρωποι αντιλαμβάνονται και παρατηρούν τον κόσμο. Αυτό γίνεται μέσα από την εμπειρία για να ομαδοποιηθούν τα στοιχεία εισόδου. Η επανάληψη με περισσότερα παραδείγματα δίνει την ικανότητα στο AI να κατηγοριοποιήσει τα δεδομένα με όλο και πιο εύστοχο τρόπο.

Στην μάθηση με επίβλεψη τα δεδομένα εισόδου έχουν μια ετικέτα πράγμα που βοηθάει τον άνθρωπο να γνωρίζει ποια δεδομένα αντιστοιχούν σε ποια έξοδο. Στην μάθηση με επίβλεψη το AI μαθαίνει μέσω παραδειγμάτων. Αποτελείται από ομάδες εισόδου και εξόδου όπου η έξοδος επισημαίνεται από την επιθυμητή τιμή. Τα δεδομένα εκπαίδευσης τροφοδοτούνται στο AI και όσο προχωράει η εκπαίδευση, είναι σε θέση να προσδιορίσει τις ομοιότητες, διαφορές μέχρι να είναι σε θέση να προβλέπει τις σωστές απαντήσεις.

Η ενισχυτική μάθηση χρησιμοποιείται για τη λήψη διαδοχικών και συνεχών αποφάσεων. Εφαρμόζεται κυρίως για τον έλεγχο των ρομπότ. Λαμβάνει συνεχώς αποφάσεις ανάλογα με το περιβάλλον στο οποίο βρίσκεται και εξάγει τα επιθυμητά αποτελέσματα. Στην ενισχυτική μάθηση, δεν περιλαμβάνεται κάποια είσοδος με ετικέτα αλλά εισάγεται ένα σύνολο επιτρεπόμενων ενεργειών και πιθανών τελικών αποφάσεων. Ο τρόπος για να έχουμε το επιθυμητό αποτέλεσμα είναι το AI να μάθε από την εμπειρία και την ανταμοιβή. Η ανταμοιβή είναι μια αριθμητική μεταβλητή και προγραμματίζεται στον κώδικα, πράγμα που κάνει το AI πρόθυμο να συλλέξει.

Σκοπός της ενισχυτικής μάθησης είναι η εκμάθηση μιας συμπεριφοράς η οποία είναι μια χαρτογράφηση από τις παρατηρήσεις στις ενέργειες που παίρνει ο Agent. Μια παρατήρηση είναι οι είσοδοι που παίρνει ο Agent από το περιβάλλον μέσω των αισθητήρων εισόδου και μια ενέργεια είναι η οποιαδήποτε αλλαγή στην συμπεριφορά του. Στόχος του Agent είναι να μεγιστοποιήσει την συνολική ανταμοιβή του. Επαινείται κάθε φορά που πετυχαίνει να κάνει αυτό που έχουμε ορίσει σαν στόχο και τιμωρείται όταν δεν το κάνει. Έτσι μαθαίνει να συμπεριφέρεται σωστά.



Εικόνα 32 - Κύκλος εκπαίδευσης του Agent

Για να πετύχει ο τελικός στόχος απαιτούνται πολλές δοκιμές και συνεχόμενες ενημερώσεις της συμπεριφοράς. Ο Agent τοποθετείται σε πολλές ίδιες καταστάσεις είτε απλές είτε πιο περίπλοκες και με την πάροδο του χρόνου μαθαίνει μια πιο βέλτιστη συμπεριφορά, πράγμα που τον κάνει πιο αποτελεσματικό. Όπως είναι φανερό, στην παρούσα πτυχιακή εργασία το μοντέλο εκπαίδευσης που έχει χρησιμοποιηθεί για την εκμάθηση του Agent είναι η ενισχυτική μάθηση.

4.2 Εγκατάσταση Python & ML-agents

Για την χρήση των ML-Agents 2.0.1 που θα χρησιμοποιήσουμε στην παρούσα πτυχιακή, απαιτείται η εγκατάσταση της Unity 2019.4 ή μεταγενέστερης, καθώς και μια σειρά πακέτων της Unity και της Python.

Αρχικά θα χρειαστεί να κατεβάσουμε μέσα από το περιβάλλον της Unity το com.unity.ml-agents package. Για να το κάνουμε αυτό θα περιηγηθούμε στο menu της Unity/Window/Package Manager. Εκεί θα αναζητήσουμε το πακέτο ML Agents και στην συνέχεια Install και η Unity θα κάνει τα υπόλοιπα μόνη της.

Στην συνέχεια θα χρειαστεί να εγκαταστήσουμε την python έκδοσης 3.6.1 ή

μεταγενέστερης. Εμείς θα εγκαταστήσουμε από το site της python την έκδοση 3.7.9. Για να ελέγξουμε ότι η python έχει εγκατασταθεί επιτυχώς θα εκτελέσουμε το Powershell ώς διαχειριστές και θα γράψουμε την εντολή python, όπου θα μας εμφανίσει την έκδοση που μόλις εγκαταστήσαμε. Ύστερα εγκαθιστούμε την τελευταία έκδοση του Pip. Το Pip είναι το προεγκατεστημένο και προτεινόμενο τυπικό σύστημα διαχείρισης της python και χρησιμοποιείται για την εγκατάσταση και διαχείριση πακέτων και βιβλιοθηκών λογισμικού. Για την εγκατάσταση του ακολουθούμε την εντολή ***get-pip.py.*** Εάν υπάρχει προεγκατεστημένη έκδοση την python καθώς και του pip, θα πρέπει να εκτελέσουμε την εντολή ***pip install –upgrade pip.*** Για να ελέγξουμε εάν η εγκατάσταση ήταν επιτυχής τρέχουμε την εντολή ***pip –V.***

Για να προχωρήσουμε, θα δημιουργήσουμε ένα εικονικό περιβάλλον της python στον υπολογιστή μας όπου μέσα εκεί θα εγκαταστήσουμε όλες τις βιβλιοθήκες και τα πακέτα που θα χρησιμοποιήσουμε. Ένα εικονικό περιβάλλον είναι ένα αυτοτελές περιβάλλον που περιέχει μια συγκεκριμένη εγκατάσταση της python καθώς και έναν αριθμό πρόσθετων πακέτων. Επομένως κάθε φορά που θα εκπαιδεύουμε το μοντέλο μας, θα ενεργοποιούμε το εικονικό αυτό περιβάλλον και θα τρέχουμε μέσα σε αυτό τις εντολές εκπαίδευσης που θα αναφέρουμε παρακάτω. Για την δημιουργία του εικονικού περιβάλλοντος ακολουθούμε τα εξής βήματα:

- Δημιουργία φακέλου για το εικονικό περιβάλλον: ***md python-envs*** (όπου python-envs αντικαθιστούμε με το επιθυμητό όνομα που θέλουμε για τον φάκελο).
- Για την δημιουργία του περιβάλλοντος εκτελούμε την εντολή: ***python -m venv python-envs\sample-env*** (όπου sample-env αντικαθιστούμε με το επιθυμητό όνομα του περιβάλλοντος.)
- Για την ενεργοποίηση του περιβάλλοντος εκτελούμε την εντολή ***python-envs\sample-env\Scripts\activate***.
- Για την απενεργοποίηση του περιβάλλοντος εκτελούμε την εντολή ***Deactivate.***

Ύστερα θα εγκαταστήσουμε το PyTorch το οποίο είναι μια βιβλιοθήκη machine learning ανοιχτού κώδικα. Χρησιμοποιείται κυρίως για Computer Vision και Natural Language Processing. Για την εγκατάσταση του PyTorch θα χρειαστεί να εκτελέσουμε την παρακάτω εντολή εντός του εικονικού περιβάλλοντος που δημιουργήσαμε. Η εντολή είναι η εξής: ***pip3 install torch~=1.7.1 -f https://download.pytorch.org/whl/torch_stable.html.***

Τέλος εγκαθιστούμε το πακέτο βιβλιοθηκών ML-Agents της python εκτελώντας την παρακάτω εντολή: ***python -m pip install mlagents==0.26.0.*** Για να βεβαιωθούμε ότι η εγκατάσταση ήταν επιτυχής εκτελούμε την εντολή ***mlagents-learn –help.***

4.3 Περιβάλλον Εκπαίδευσης

Μετά την ολοκλήρωση της εγκατάστασης των ML-Agents, θα πρέπει να δημιουργήσουμε το κατάλληλο για μας περιβάλλον εκπαίδευσης. Ένα περιβάλλον περιέχει έναν η περισσότερους Agents καθώς και άλλα αντικείμενα ή οντότητες με τα οποία αλληλοεπιδρά. Το περιβάλλον που θα αναπτύξουμε στο συγκεκριμένο παιχνίδι θα αποτελείται από τρία διαφορετικά Training Set τα οποία υιοθετούν παρόμοια χαρακτηριστικά και είναι αναγκαία για την εξέλιξη της εκμάθησης. Στο κεφάλαιο αυτό θα αναλύσουμε ένα από τα τρία αυτά Training Set καθώς το μόνο που διαφέρει μεταξύ τους είναι ο βαθμός δυσκολίας της διαδρομής. Το Training Set αποτελείται από την πίστα που έχουμε διαμορφώσει κατάλληλα με την τοποθέτηση των απαραίτητων checkpoints, το σημείο εκκίνησης και επαναφοράς των Agents, καθώς και τους ίδιους τους Agents.

Τα checkpoints είναι απαραίτητα για την εκμάθηση της εκάστοτε πίστας. Είναι τοποθετημένα σε σχετικά μικρή απόσταση μεταξύ τους ώστε οι αισθητήρες του Agent να μπορούν να εντοπίζουν το επόμενο σε σειρά checkpoint. Το σημείο εκκίνησης είναι ένα απλό empty object το οποίο το έχουμε τοποθετήσει στο σημείο που θέλουμε να ξεκινάει και να επαναφέρεται ο Agent κάθε φορά που τελειώνει ένα επεισόδιο εκπαίδευσης. Ένα επεισόδιο εκπαίδευσης τελειώνει όταν ο Agent βγει εκτός των ορίων της πίστας, όταν βρεθεί σε θέση μη ικανή να συνεχίσει την πορεία του, καθώς και όταν τελειώσουν τα MaxSteps που του έχουμε ορίσει.

Για τον Agents θα χρησιμοποιήσουμε τέσσερα διαφορετικά Scripts. Αυτά ελέγχουν τις αποφάσεις που θα πάρει ο Agent, πότε θα τις πάρει καθώς και το πώς θα τις πάρει ανάλογα με τις πληροφορίες που παίρνει από το περιβάλλον χρησιμοποιώντας Raycasts.

4.3.1 TrackCheckpoints Script

Για τον έλεγχο και χειρισμό των checkpoints δημιουργήσαμε δυο λίστες. Επίσης δημιουργούμε μία λίστα ακόμα για τον έλεγχο του Agent μας. Οι λίστες είναι οι ακόλουθες:

- Η πρώτη λίστα περιέχει όλα τα checkpoints που έχουμε τοποθετήσει στο Track.
- Η δεύτερη λίστα είναι ένας δείκτης που δείχνει ποιο είναι το επόμενο checkpoint που πρέπει να περάσει ο Agent.
- Η τρίτη λίστα περιέχει όλους τους Agents που θα εκπαιδεύσουμε στο TrainingSet.

Την λίστα AgentList την έχουμε ως [SerializedField] διότι θέλουμε να είναι ορατή στον Editor της Unity, ώστε να δηλώσουμε εκεί τους όλους Agents που θα εκπαιδευτούν.

```
private List<Checkpoint> checkpointList;
private List<int> CheckpointIndexList = new List<int>();
[SerializeField] private List<GameObject> AgentList = new List<GameObject>();
```

Ύστερα καλούμε την εσωτερική μέθοδο της Unity Awake(). Στην Awake() αναζητούμε όλα τα checkpoints που βρίσκονται στο συγκεκριμένο TrainingSet και τα προσθέτουμε στην λίστα checkpointList. Επίσης δημιουργούμε έναν δείκτη για κάθε Agent του TrainingSet, τον προσθέτουμε στην λίστα CheckpointIndexList και τον αρχικοποιούμε να δείχνει στο πρώτο checkpoint της λίστας.

```
void Awake()
    {
        checkpointList = new List<Checkpoint>();
        Transform checkpointsTransform = transform.Find("checkpoints");
        foreach (Transform Checkpoints in checkpointsTransform)
        {
            Checkpoint checkpoint = Checkpoints.GetComponent<Checkpoint>();
            checkpoint.SetTrackCheckpoints(this);
            checkpointList.Add(checkpoint);
        }
        for (int i = 0; i < AgentList.Count; i++)
        {
            CheckpointIndexList.Add(0);
        }
    }
```

Στη συνέχεια δημιουργούμε μία συνάρτηση Set, η οποία δέχεται από την συνάρτηση reset της κλάσης AiAgent, τον συγκεκριμένο Agent για τον οποίο και επαναφέρει τον δείκτη των checkpoint στην αρχική τιμή.

```
public void SetIndex(GameObject index)
 {
     if (CheckpointIndexList.Count != AgentList.Count)
     {
         foreach (var item in AgentList)
         {
             CheckpointIndexList.Add(0);
         }
     }
     CheckpointIndexList[AgentList.IndexOf(index)] = 0;
 }
```

Τέλος δημιουργούμε την μέθοδο VehiclePassCheckpoint με δυο παραμέτρους, το checkpoint που πέρασε ο Agent και ποιος Agent ήταν. Η μέθοδος αυτή παίρνει από τη λίστα checkpointIndexList το σωστό checkpoint που πρέπει να περάσει ο Agent, ελέγχει εάν αυτό είναι το ίδιο με το checkpoint που προσπεράστηκε, μεταθέτει το σωστό checkpoint και δίνει μια θετική επιβράβευση. Αλλιώς δίνει στον Agent αρνητική επιβράβευση.

```
public void VehiclePassCheckpoint(Checkpoint checkpoint, GameObject vehicle)
    {
        AiAgent agent = AgentList[AgentList.IndexOf(vehicle)].GetComponent<AiAgent>();
        if (checkpointList.IndexOf(checkpoint) == CheckpointIndexList[AgentList.IndexOf(vehicle)])
        {
            CheckpointIndexList[AgentList.IndexOf(vehicle)] = (CheckpointIndexList[AgentList.IndexOf(vehicle)] + 1) % checkpointList.Count;
            agent.Reward(1f);
        }
        else
        {
            agent.Reward(-0.05f);
        }
    }
```

4.3.2 Checkpoints Script

Η κλάση αυτή έχει δυο λειτουργείες που συνδέονται άμεσα με την κλάση TrackCheckpoints. Η πρώτη μέθοδος δέχεται τα checkpoints που βρέθηκαν στο track, ενώ η δεύτερη στέλνει τα στοιχεία του checkpoint και του agent. Σκοπός της είναι να δείξει στην μέθοδο VehiclePassCheckpoint ποιο checkpoint περάστηκε από ποιον Agent.

```
private TrackCheckpoints trackCheckpoints;
    public void SetTrackCheckpoints(TrackCheckpoints trackCheckpoints)
    {
        this.trackCheckpoints = trackCheckpoints;
    }
    private void OnTriggerEnter(Collider other)
    {
        if (other.TryGetComponent<AiAgent>(out AiAgent aiagent))
        {
            trackCheckpoints.VehiclePassCheckpoint(this,other.gameObject);
        }
    }
```

Σε κάθε Agent έχει οριστεί ένα tag με όνομα AIcarTrigger.

4.3.3 AI Agent Script

Το AiAgent Script είναι το κυριότερο script για την εκπαίδευση. Περιέχει όλες τις απαραίτητες συναρτήσεις και μεθόδους που καθορίζουν τις αποφάσεις που μπορεί να πάρει, με ποια αντικείμενα στο περιβάλλον θα αλληλοεπιδρά καθώς και άλλες παραμέτρους που έχουμε ορίσει εμείς με βάση το πως ο Agent θα εκπαιδευτεί. Επίσης έχουμε δημιουργήσει ένα σύστημα που μετράει το χρόνο του κάθε Agent από την στιγμή που θα ξεκινήσει μέχρι να ολοκληρώσει επιτυχώς ένα γύρο της πίστας και αποθηκεύει τον καλύτερο χρόνο.

Δημιουργούμε μια κλάση AiAgent η οποία κληρονομεί από την κλάση Agent, η οποία είναι εσωτερική κλάση που κατεβάσαμε με το πακέτο ML-Agents η οποία επικοινωνεί με το pyTorch της python, και περιέχει όλες τις βασικές συναρτήσεις και μεθόδους για την εκπαίδευση. Οι συναρτήσεις αυτές περιέχουν τις λειτουργίες απαραίτητες για την δημιουργία του νευρωνικού δικτύου, και εμείς με βάση το τι θέλουμε να κάνει ο Agent, προγραμματίζουμε τις κατάλληλες παραμέτρους και μεταβλητές.

Με την δημιουργία της κλάσης AiAgent, στο παράθυρο του Inspector εμφανίζεται η επιλογή MaxSteps. Το MaxSteps καθορίζει πόσα βήματα μπορούν να συμβούν στην εκπαίδευση πριν το τέλος του επεισοδίου. Στην δική μας περίπτωση ο Agent που έχουμε δημιουργήσει, επανεκκινείται μετά από 5000 βήματα.

Ξεκινάμε δηλώνοντας τις μεταβλητές που θα χρησιμοποιήσουμε.

```
1  [SerializeField] private CarMovement agent;
2      [SerializeField] private Transform player;
3      [SerializeField] private Transform resPoint;
4      [SerializeField] private GameObject Track;
5      private Rigidbody rb;
```

Η πρώτη συνάρτηση είναι η OnEpisodeBegin(), η οποία καλείται στην αρχή κάθε επεισοδίου εκπαίδευσης. Ένα επεισόδιο εκπαίδευσης ορίζεται από τα max steps που έχουμε ορίσει σαν παράμετρο και τις συνθήκες που θα αναλύσουμε στην πορεία. Για τον δικό μας Agent, στην OnEpisodeBegin() καλούμε την μέθοδο reset(). Η reset επαναφέρει τον Agent στην θέση εκκίνηση που έχουμε προκαθορίσει για κάθε ένα, δίνοντας του μηδενική ταχύτητα για το ξεκίνημα του νέου επεισοδίου. Επίσης στέλνει στην κλάση TrackCheckpoints το gameObject του Agent, για την επαναφορά του δείκτη των Checkpoints.

```
1  public override void OnEpisodeBegin()
2      {
3          reset();
4      }
5      public void reset()
6      {
7          resetCurrentTime = true;
8          startTimerBool = false;
9          TrackCheckpoints nextIndex = Track.GetComponent<TrackCheckpoints>();
10         nextIndex.SetIndex(this.gameObject);
11         player.transform.position = resPoint.transform.position;
12         player.transform.rotation = resPoint.transform.rotation;
13         rb.velocity = new Vector3(0, 0, 0);
14     }
```

Επόμενη σημαντική και απαραίτητη μέθοδος είναι η OnActionReceived(ActionBuffers actions). Η μέθοδος αυτή καλείται κάθε φορά που ο Agent αποφασίζει να πάρει μια απόφαση. Το δέντρο ενεργειών που του έχουμε ορίσει ότι μπορεί να κάνει αποτελείται από

μία ενέργεια κίνησης, δηλαδή να διαλέξει μεταξύ της επιτάχυνσης, την επιβράδυνσης και της ακινησίας και από μία ενέργεια χειρισμού κατεύθυνσης δηλαδή την αλλαγή πορείας

προς τα αριστερά, δεξιά ή την συνέχιση της πορεία του ευθεία. Τέλος, στέλνουμε την ενέργεια που αποφάσισε να κάνει ο Agent στην συνάρτηση move() της κλάσης CarMovement, η οποία είναι υπεύθυνη για τον τρόπο κίνησης του Agent.

```
public override void OnActionReceived(ActionBuffers actions)
    {
        float speed = 0f;
        float steering = 0f;
        switch (actions.DiscreteActions[0])
        {
            case 0:
                speed = 0f;
                break;
            case 1:
                speed = +1f;
                break;
            case 2:
                speed = -1f;
                break;
        }
        switch (actions.DiscreteActions[1])
        {
            case 0:
                steering = 0f;
                break;
            case 1:
                steering = +1f;
                break;
            case 2:
                steering = -1f;
                break;
        }
        agent.Move(speed, steering);
    }
```

Η συνάρτηση που μας επιτρέπει να χειριστούμε εμείς τις ενέργειες του Agent είναι η Heuristic(in ActionBuffers actionsOut). Η μέθοδος αυτή παίρνει είσοδο από το πληκτρολόγιο μας, στην συγκεκριμένη περίπτωση τα κλασσικά πλήκτρα WSDA, τα στέλνει στην μέθοδο OnActionReceived() και κατά συνέπεια χειριζόμαστε εμείς τον Agent.

```
public override void Heuristic(in ActionBuffers actionsOut)
    {
        int forward = 0;
        if (Input.GetKey(KeyCode.W))
        {
            forward = 1;
        }
        if (Input.GetKey(KeyCode.S))
        {
            forward = 2;
        }
        int turn = 0;
        if (Input.GetKey(KeyCode.D))
        {
            turn = 1;
        }
        if (Input.GetKey(KeyCode.A))
        {
            turn = 2;
        }
        ActionSegment<int> discreteActions = actionsOut.DiscreteActions;
        discreteActions[0] = forward;
        discreteActions[1] = turn;
    }
```

Οι επόμενες τρείς μέθοδοι είναι υπεύθυνες στο να δώσουν στον Agent μια επιβράβευση, είτε αυτή είναι θετική είτε αρνητική, ανάλογα με το πως ο Agent αλληλοεπιδρά με το περιβάλλον, πάντα βέβαια με την δικής μας αρέσκεια.

```
private void OnCollisionEnter(Collision collision)
    {
        if (collision.gameObject.tag == "AIcarTrigger")
        {
            Reward(-0.6f);
        }
    }
    private void OnTriggerEnter(Collider other)
    {
        if (other.gameObject.tag == "reset")
        {
           SetReward(-1);
           EndEpisode();
        }
    }
    private void OnTriggerStay(Collider other)
    {
        if (other.gameObject.tag == "wall")
        {
            Reward(-0.05f);
        }
        if (other.gameObject.tag == "AIcarTrigger")
        {
            Reward(-0.05f);
        }
    }
```

Η πρώτη μέθοδος ελέγχει εάν υπάρξει σύγκρουση του Agent με κάποιο άλλο αντικείμενο με tag AIcarTrigger, και εφόσον ανιχνευθεί τότε δίνει στον Agent αρνητική επιβράβευση τιμής -1. Η δεύτερη μέθοδος ελέγχει εάν ο Agent βγει εκτός των ορίων της πίστας και του δίνει αρνητική επιβράβευση τιμής -1. Επίσης καλείται η μέθοδος EndEpisode() η οποία τερματίζει το επεισόδιο εκπαίδευσης. Αυτό καθορίζει την έναρξη νέου επεισοδίου. Η Τρίτη μέθοδος δίνει στον Agent αρνητική επιβράβευση με τιμή -0.05 εάν ακουμπήσει και για όσο παραμένει σε επαφή με τον τοίχο ή με κάποιο άλλο όχημα.

```
public void Reward(float reward)
    {
        AddReward(reward);
    }
```

Για τις επιβραβεύσεις που αναφέραμε δημιουργούμε την μέθοδο Reward που καλεί την συνάρτηση AddReward() την οποία την κληρονομούμε από την κλάση Agent. Στην AddReward δίνουμε την τιμή reward ως παράμετρο.

Τέλος, έχουμε το σύστημα χρονομέτρησης για το οποίο έχουμε δηλώσει τις εξής μεταβητές:

```
1  public bool startTimerBool;
2     public bool bestTimeBool;
3     public bool resetCurrentTime;
4     public float bestLap;
5     public float timer;
6     public float currentBestTime;
7     public Text timerText;
8     public Text bestTime;
```

Στην συνάρτηση Awake() αρχικοποιούμε όσες μεταβλητές είναι απαραίτητο.

```
1  private void Awake()
2     {
3         bestLap = 999f;
4         timer = 0;
5         startTimerBool = false;
6         bestTimeBool = false;
7         resetCurrentTime = false;
8         rb = GetComponent<Rigidbody>();
9     }
```

Στη συνέχεια καλούμε την μέθοδο FixedUpdate() η οποία είναι συνάρτηση της Unity και καλείται κάθε καρέ.

```
void FixedUpdate()
    {
        if (startTimerBool == true)
        {
            timer = timer + Time.deltaTime;
            timerText.text = this.gameObject.name + ": " + timer.ToString("f2");
        }
        if (bestTimeBool == true)
        {
            bestTime.text = this.gameObject.name + " Best Time :  " + bestLap.ToString("f2");
        }
    }
```

Οι παραπάνω μεταβλητές παίρνουν τις τιμές τους μετά από έλεγχο που έχει γίνει στο Script TrackCheckpoints. Η timer μετράει σε πραγματικό χρόνο το χρόνο που κάνει ο Agent από την στιγμή που θα περάσει την εκκίνηση δηλαδή το πρώτο Checkpoint και το εμφανίζει στην οθόνη μέσω ενός text component. Η μεταβλητή bestLap είναι ο καλύτερος χρόνος που έχει κάνει ο Agent σε ένα ολοκληρωμένο γύρο και το εμφανίζει επίσης στην οθόνη μετά την πρώτη επιτυχημένη ολοκλήρωση ενός γύρου. Το κομμάτι κώδικα που ακολουθεί αποτελεί μέρος της κλάσης TrackCheckpoints μέσα στην μέθοδο VehiclePassCheckpoint.

```
if (checkpointList.IndexOf(checkpoint) == 0)
        {
            agent.currentBestTime = agent.timer;
            agent.timer = 0;
            agent.startTimerBool = true;
            if (agent.resetCurrentTime == false)
            {
                if ((agent.currentBestTime <= agent.bestLap) )
                {
                    agent.bestTimeBool = true;
                    agent.bestLap = agent.currentBestTime;
                }
            }
            else
            {
                agent.resetCurrentTime = false;
            }
        }
```

Αξίζει να αναφέρουμε και την μέθοδο CollectObservations(VectorSensor sensor) του πακέτου ML-Agents όπου μπορούμε να δημιουργήσουμε και να χρησιμοποιήσουμε δικούς μας αισθητήρες για να παρατηρεί το περιβάλλον. Εμείς δεν θα κάνουμε χρήση αυτής της μεθόδου διότι θα χρησιμοποιήσουμε το Ray Perception Sensor 3D.

4.3.4 Behavior Parameters And Decision Requester

Στην συνέχεια έχουμε τα Scripts των ML-Agents, Behavior Parameters και Decision Requester. Το Behavior parameters δημιουργείται αυτόματα όταν ορίσουμε ένα prefab ως Agent βάζοντας του Agent script. Ελέγχει τις παραμέτρους που θα λάβει ο Agent. Συγκεκριμένα μπορούμε να παραμετροποιήσουμε και να ορίσουμε τις ακόλουθες ιδιότητες.

- Behavior Name: Με το behavior name μπορούμε να ορίσουμε το όνομα που θα λάβει το μοντέλο συμπεριφοράς. Οι Agents που έχουν το ίδιο όνομα, θα έχουν και την ίδια συμπεριφορά. Το όνομα στο δικό μας μοντέλο είναι το cartAI.
- Vector Observation: Χωρίζεται σε δυο κατηγορίες. Το Space size και το Stacked Vectors. Με το Space size ορίζουμε το πλήθος των διανυσματικών παρατηρήσεων του Agent. Το Stacked Vectors είναι ο αριθμός των διανυσματικών παρατηρήσεων του Agent που μπορεί να στοιβαχθεί για την χρησιμοποίησή του στην λήψη αποφάσεων. Με αποτέλεσμα το πραγματικό μέγεθος παρατήρησης του διανύσματος να είναι το μέγεθος των διανυσματικών παρατηρήσεων επί των στοιβαγμένων διανυσμάτων. Στο δικό μας μοντέλο έχουμε αφήσει το μέγεθος του Space size μηδέν διότι κάνουμε χρήση άλλης τεχνικής λήψης παρατηρήσεων.

- Actions: Επίσης χωρίζεται σε δυο κατηγορίες. Το Continuous Actions και Discrete Branches. Το Continuous Actions είναι ο αριθμός των ταυτόχρονων συνεχώς ενεργειών που μπορεί να κάνει ο Agent. Είναι ένας πίνακας πραγματικών αριθμών και μπορεί να πάρει τιμές από -1 έως 1. Το Discrete Branches, αυτό δηλαδή που θα χρησιμοποιήσουμε στην εκπαίδευση του μοντέλου μας, είναι ένας πίνακας ακεραίων που ορίζει ταυτόχρονες πολλαπλές διακριτές ενέργειες. Όπως είδαμε και στο Script AiAgent, έχουμε δυο κλάδους ενεργειών. Ένα για την κίνηση του αυτοκινήτου και ένα για το στρίψιμο. Κάθε κλάδος έχει τρεις πιθανές τιμές που μπορεί να πάρει, -1,0,+1. Αυτές οι τιμές στον πρώτο κλάδο αντιστοιχούν στην ταχύτητα του αυτοκινήτου, η οποία μπορεί να είναι θετική, αρνητική ή μηδενική. Στον δεύτερο κλάδο ορίζουν την κατεύθυνση του αυτοκινήτου αριστερά, δεξιά η ευθεία.

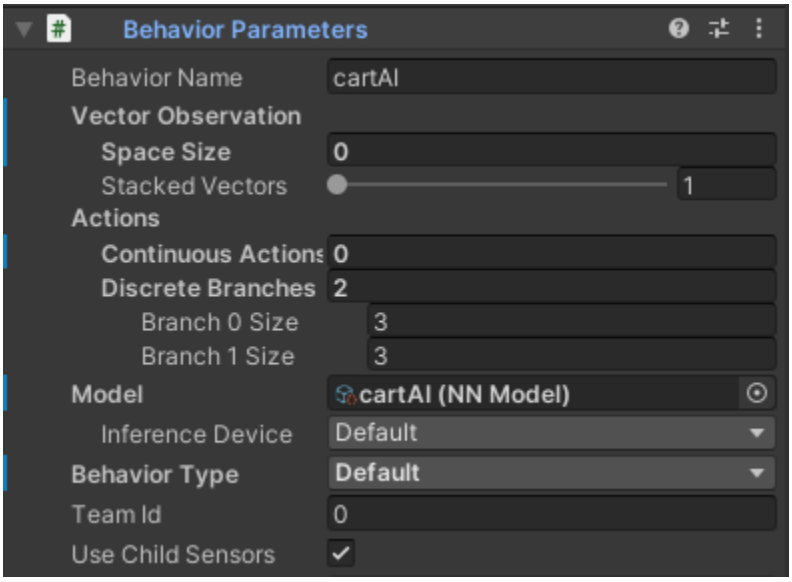
- Model: Εδώ τοποθετούμε το μοντέλου το νευρονικού δικτύου που δημιουργείται μετά την ολοκλήρωση της εκπαίδευσης.

- Inference Device: Ορίζει εάν θα εκτελέσει το μοντέλο της εκπαίδευσης ο επεξεργαστής ή η κάρτα γραφικών. Υπάρχουν τέσσερις επιλογές, cpu,gpu,default και burst. Εμείς θα χρησιμοποιήσουμε την επιλογή default ώστε να κάνει την επιλογή η Unity.

- Behavior Type: Καθορίζει τον χειρισμό του Agent. Είτε θα εκτελέσει το μοντέλο που του έχει δοθεί είτε ο χειρισμός θα γίνει από εμάς. Οι επιλογές είναι Default, στην οποία το μοντέλο θα εκπαιδευτεί εάν συνδεθεί με την Python ή θα χειριστεί το μοντέλο που του έχουμε δώσει, Heuristic που δηλώνει ότι θα χειριζόμαστε εμείς τον Agent και Inference που δηλώνει ότι ο Agent θα χειρίζεται με βάση το νευρονικό δίκτυο που του έχουμε δώσε στο model.

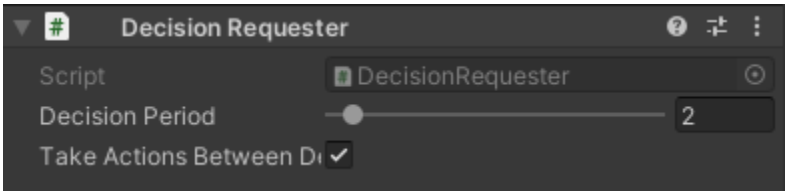
- Team Id: Χρησιμοποιείται για την ομαδοποίηση των Agent

- Child Sensor: Ορίζει εάν θα χρησιμοποιηθούν αισθητήρες που είναι συνδεδεμένοι σε gameObject, σε παιδία του Agent.



Εικόνα 33 - Behavior Parameters

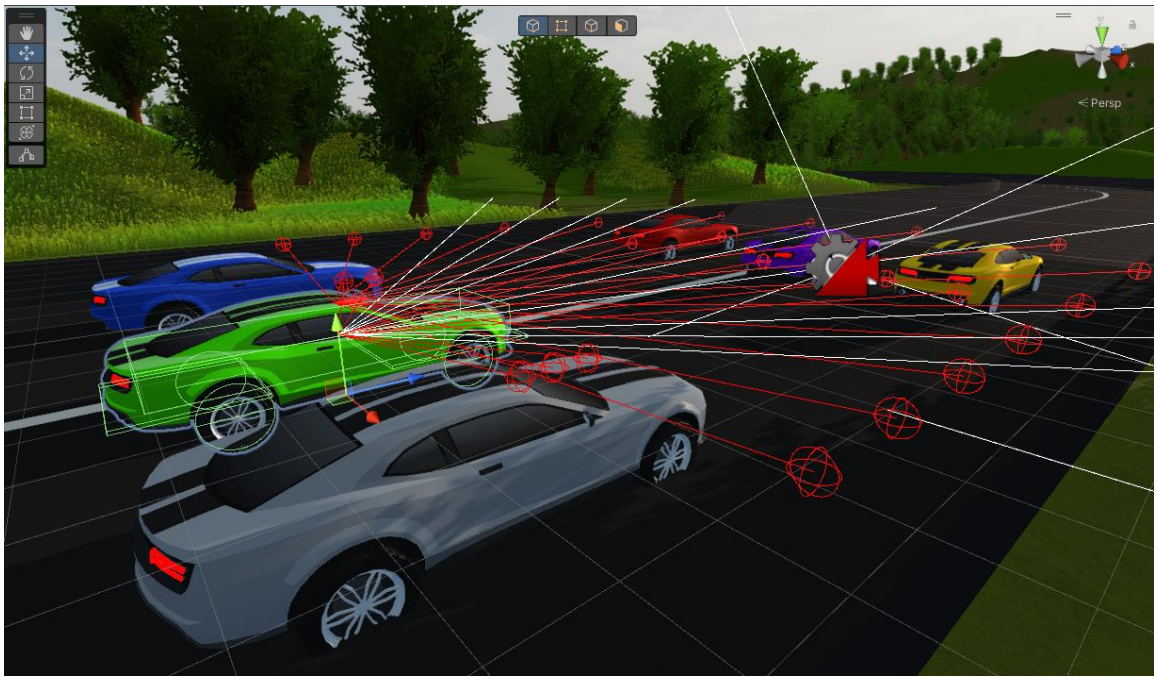
Το δεύτερο component που χρειάζεται να συμπεριλάβουμε στον Agent είναι το Decision Requester. Αυτό αποτελείται από το Decision period που καθορίζει κάθε πότε ο Agent θα ζητήσει να πάρει μια απόφαση και από το Take Actions Between Decisions που ορίζει αν ο Agent θα κάνει ενέργειες μεταξύ των αποφάσεων, όταν δεν ζητάει να πάρει μια απόφαση.



Εικόνα 34 - Decision Requester

4.3.5 Ray Perception Sensor 3D

Τα Raycasts είναι ένας εναλλακτικός τρόπος παροχής παρατηρήσεων του περιβάλλοντος στον Agent. Αυτό γίνεται με το να προσθέσουμε το Script Ray Perception Sensor 3D στο gameObject του Agent. Ο τρόπος λειτουργίας του είναι η δημιουργία ακτινών που ξεκινάνε από το κέντρο του Agent και κατευθύνονται σε διάφορες κατευθύνσεις μέχρι να χτυπήσουν κάποιο αντικείμενο.



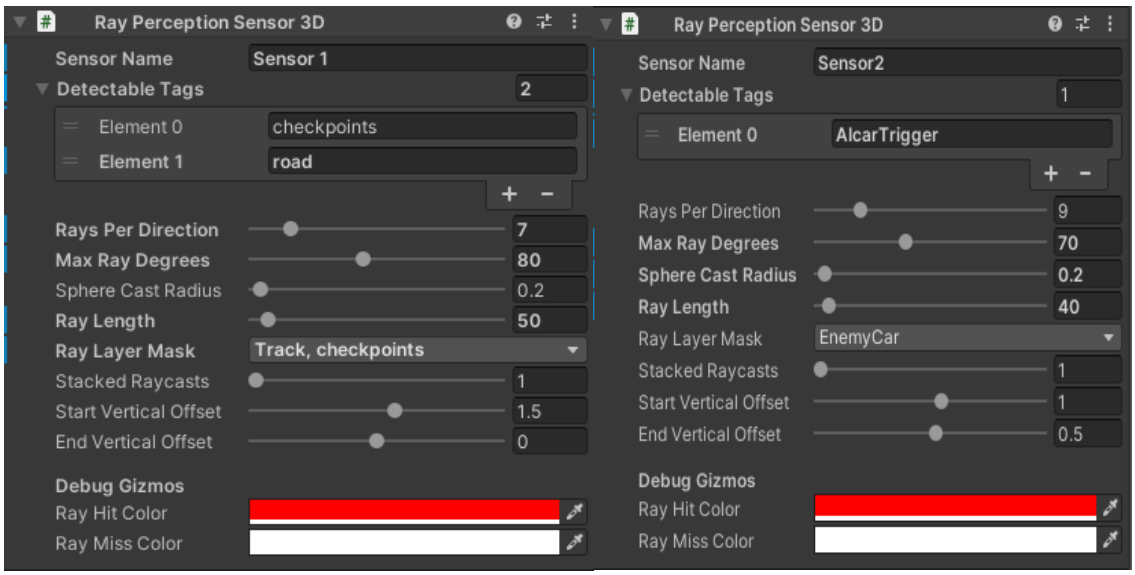
Εικόνα 35 - Ray Perception Sensor On The Agents

Όπως και τα παραπάνω, το Ray Perception Sensor 3D έχει τις δικιές του ρυθμίσεις. Στον Agent έχουμε δημιουργήσει δυο αισθητήρες που αποσκοπούν σε δυο διαφορετικές συνθήκες

- Sensor Name: Δηλώνει το όνομα του αισθητήρα. Τα ονόματα που έχουμε δώσει εμείς είναι Sensor 1 και Sensor 2
- Detectable Tags: Όπως είδαμε, όλα τα game objects που έχουμε δημιουργήσει στο περιβάλλον έχουν ένα μοναδικό tag. Με αυτά μπορούμε να ορίσουμε στον αισθητήρα από ποια αντικείμενα θα πάρει πληροφορίες. Για το sensor 1 έχουμε δηλώσει ως detectable tags τα checkpoints και το road. Για το sensor 2 έχουμε ένα detectable tag, το AIcarTrigger.
- Rays Per Direction: Καθορίζει τον αριθμό των ακτινών που εκπέμπονται. Πάντα υπάρχει μια ακτίνα που εκπέμπεται προς τα εμπρός και ανάλογα με τον αριθμό που δίνουμε δημιουργούνται ακτίνες δεξιά και αριστερά. Στο Sensor 1 έχουμε βάλει 7 άρα δημιουργούνται 15 ακτίνες και στο sensor 2 έχουμε βάλει 9 άρα 19 ακτίνες σύνολο.
- Max Ray Degrees: Είναι ο αριθμός σε μοίρες που θα επεκταθούν οι ακτίνες που έχουμε δημιουργήσει. Στο sensor 1 τις έχουμε επεκτείνει στις 80 μοίρες και στο sensor 2 στις 70 μοίρες. Ο λόγος για τις παραπάνω τιμές είναι διότι ο Agent θέλουμε να βλέπει περισσότερο μπροστά και στο πλάι και καθόλου πίσω.
- Spherer Cast Radius: Μία ακτίνα όταν επεκτείνεται, στο άκρο της υπάρχει μία σφαίρα. Αυτή ανάλογα με το μέγεθος που θα της δώσουμε στη σφαίρα, μπορεί να

εντοπίσει ένα αντικείμενο με μεγαλύτερη ακρίβεια. Στις σφαίρες και των δυο sensor έχουμε δώσει την τιμή 0.2

- Ray Length: Είναι το μήκος των ακτινών. Στο sensor 1 έχουμε δώσει ως μήκος την τιμή 50 ενώ στο sensor 2 την τιμή 50
- Ray Layer Mask: Δηλώνουμε τα Layers που θέλουμε να μπορούν να παρατηρηθούν από τους αισθητήρες. Ο sensor 1 θέλουμε να παρατηρεί το track και τα checkpoints και ο sensor 2 τα άλλα αυτοκίνητα που υπάρχουν στο track.
- Stacked Raycast: Δηλώνει το τον αριθμό των προηγούμενων παρατηρήσεων που μπορούν να χρησιμοποιηθούν για την λήψη μιας απόφασης. Ο αριθμός που έχουμε δηλώσει και στους δυο sensors είναι 5.
- Start Vertical Offset: Είναι η κατακόρυφη μετατόπιση των ακτινών από το σημείο έναρξης.
- End Vertical Offset: Είναι η κατακόρυφη μετατόπιση των ακτινών στο τελικό τους σημείο.
- Ray Hit Color: Δηλώνει το χρώμα την ακτίνας εάν αυτή χτυπήσει κάποιο detectable αντικείμενο.
- Ray Miss Color: Δηλώνει το χρώμα της ακτίνας όταν δεν εντοπίσει κάποιο αντικείμενο στην πορεία της.



Εικόνα 36 - Ray Perception Sensors

4.4 Εκπαίδευση ML-Agent

Εφόσον έχουμε ολοκληρώσει το περιβάλλον εκπαίδευσης καθώς και τα απαραίτητα scripts, μπορούμε να ξεκινήσουμε την διαδικασία της εκπαίδευσης. Για να ξεκινήσει η εκπαίδευση πρέπει πρώτα να ανοίξουμε το PowerShell και να ενεργοποιήσουμε το εικονικό περιβάλλον που έχουμε δημιουργήσει. Το μόνο που αρκεί για να ξεκινήσει η εκπαί-

δευση είναι να γράψουμε την εντολή mlagents-learn. Με αυτόν τον τρόπο ο Agent θα εκπαιδευτεί με τις προεπιλεγμένες παραμέτρους που ενσωματώνει το πακέτο των ML-Agents. Αυτές τις παραμέτρους μπορούμε βέβαια να τις αλλάξουμε και να τις παραμετροποιήσουμε με βάση τις δικές μας ανάγκες, είναι ένα training configuration file το οποίο διαβάζει και χρησιμοποιεί αυτές τις τιμές και έχει κατάληξη .yaml. Για να δημιουργήσουμε αυτό το αρχείο αρκεί να ανοίξουμε έναν οποιοδήποτε text editor και να το αποθηκεύσουμε με την κατάληξη .yaml. Το αρχείο που φτιάξαμε εμείς για την εκπαίδευση το ονομάσαμε training.yaml. Οι παράμετροι που μας δίνει την δυνατότητα να παραμετροποιήσουμε ποικίλουν ανάλογα με το μοντέλο που θέλουμε να εκπαιδεύσουμε. Για την δική μας εκπαίδευση θα χρησιμοποιήσουμε τις παραμέτρους που αναγράφονται στην παρακάτω εικόνα και θα τις δούμε αναλυτικά.

```
default_settings: null
behaviors:
  cartAI:
    trainer_type: ppo
    hyperparameters:
      batch_size: 256
      buffer_size: 10240
      learning_rate: 0.0003
      beta: 0.0005
      epsilon: 0.2
      lambd: 0.95
      num_epoch: 3
      learning_rate_schedule: linear
    network_settings:
      normalize: false
      hidden_units: 128
      num_layers: 2
      vis_encode_type: simple
      goal_conditioning_type: none
    reward_signals:
      extrinsic:
        gamma: 0.99
        strength: 1.0
        network_settings:
          normalize: false
          hidden_units: 128
          num_layers: 2
          vis_encode_type: simple
          goal_conditioning_type: none
    init_path: null
    keep_checkpoints: 5
    checkpoint_interval: 500000
    max_steps: 10000000
    time_horizon: 128
    summary_freq: 100000
    threaded: false
```

Εικόνα 37 - Configuration File

4.4.1 Παράμετροι Εκπαίδευσης

Behaviors:

- Trainer_type: Είναι ο τύπος της εκπαίδευσης που χρησιμοποιείται. Υπάρχουν τρείς τύποι εκπαίδευσης. Το PPO, SAC και POCA. Εμείς χρησιμοποιήσαμε το ppo διότι έχει αποδειχθεί πιο σταθερό στην εκπαίδευση και είναι γενικού σκοπού από τα υπόλοιπα, τα οποία σε πολύ ειδικές περιπτώσεις είναι πιο αποτελεσματικά. Και οι τρείς τύποι εκπαίδευσης έχουν κάποιες κοινές παραμέτρους αλλά ανάλογα με τον τύπο εκπαίδευσης, υπάρχουν και κάποιες οι οποίες διαφέρουν. Αυτό θα το δούμε στην κατηγορία Hyperparameters.
- Batch_size: Είναι ο αριθμός των εμπειριών που συλλέγει ο Agent σε κάθε συγκεκριμένο αριθμό ενεργειών. Ο αριθμός αυτός πρέπει πάντα να διαιρείται ακριβώς με το buffer_size. Εάν χρησιμοποιήσουμε Continuous Actions το εύρος τιμών που μπορεί να πάρει αυτή η παράμετρος είναι μεταξύ του 512-5120. Αντίθετα, όπως και στην δική μας περίπτωση που χρησιμοποιούμε Discrete Actions, το εύρος τιμών που μπορεί να πάρει είναι μεταξύ του 32-512.
- Buffer_size: Αντιστοιχεί στο πλήθος των εμπειριών που πρέπει να συλλεχθούν πριν την οποιαδήποτε εκμάθηση ή ενημέρωση του μοντέλου εκπαίδευσης του Agent. Θα πρέπει να είναι πολλαπλά μεγαλύτερο από το batch_size. Όσο μεγαλύτερο είναι το buffer_size, τόσο ποιο σταθερές είναι συνήθως οι ενημερώσεις που λαμβάνει το μοντέλο εκπαίδευσης. Οι σύνηθες εύρος τιμών που μπορεί να πάρει είναι μεταξύ του 2048-409600. Ο λόγος που εμείς επιλέξαμε την τιμή 10240 είναι διότι επιθυμούμε ο Agent λόγο τον γρήγορων και αρκετών αποφάσεων που πρέπει να παίρνει σε πολύ σύντομο χρονικό διάστημα να λαμβάνει γρήγορες ενημερώσεις στην πορεία της εκπαίδευσης του.
- Learning_rate: Είναι ο ρυθμός μάθησης του μοντέλου εκπαίδευσης. Η τιμή που δίνουμε στον ρυθμό μάθησης καθορίζει το πόσο έντονα θα μεταβάλλονται οι τιμές των βαρών του νευρωνικού δικτύου. Η τιμή αυτή θα πρέπει να αρκετά μεγάλη ώστε να επιτυγχάνεται γρήγορα η επιθυμητή εκπαίδευση αλλά όχι τόσο που να δημιουργεί μια ασταθές ενημέρωση στο μοντέλο εκπαίδευσης. Το εύρος τιμών που μπορεί να πάρει είναι από $1*10^{-5}$ - $1*10^{-3}$.
- Beta: Ρυθμίζει το πόσο τυχαίες θα είναι οι ενέργειες που κάνει ο Agent. Δίνοντας μεγάλη τιμή στο beta, διασφαλίζουμε ότι ο Agent θα πάρει όλες τις τυχαίες αποφάσεις εξερευνώντας σωστά όλο το χώρο δράσης κατά την διάρκεια της εκπαίδευσης. Πρέπει να σημειωθεί ότι η τιμή αυτή πρέπει να ρυθμιστεί με τέτοιο τρόπο ώστε να μειώνεται ή να αυξάνεται ανάλογα με την μεταβολή της συνολικής ανταμοιβής που λαμβάνει ο Agent στο τέλος κάθε επεισοδίου εκπαίδευσης.
- Epsilon: Επηρεάζει πόσο γρήγορα μπορεί να εξελιχθεί οι ενημερώσεις κατά την διάρκεια της εκπαίδευσης. Ρυθμίζοντας μικρή τιμή χαμηλά, θα έχουμε ως αποτέλεσμα πιο σταθερές ενημερώσεις αλλά ποιο αργή την διαδικασία εκπαίδευσης.
- Lambd: Χρησιμοποιείται κατά τον υπολογισμό εκτίμησης γενικευμένου πλεονε-

κτήματος (GAE). Το GAE δείχνει το πόσο ο Agent μπορεί να βασίζεται στην τρέχουσα αξία του κατά τον υπολογισμό μιας ενημέρωσης του μοντέλου εκπαίδευσης. Χαμηλότερη τιμή αντιστοιχεί στο να βασίζεται περισσότερο στην τρέχουσα αξία του ενώ υψηλότερη τιμή αντιστοιχεί στο να βασίζεται περισσότερο στα reward που δέχεται από το περιβάλλον. Το εύρος τιμών που μπορεί να πάρει είναι μεταξύ του 0.9- 0.95.

- Num_epoch: Ο αριθμός αυτός δηλώνει πόσες φορές θα τρέξει ο αλγόριθμος διόρθωσης λάθους όταν γεμίσει το buffer_size. Στην συγκεκριμένη περίπτωση ο αλγόριθμος θα περάσει τρείς φορές από το buffer_size δημιουργώντας νέα βάρη, καλύτερα σε κάθε πέρασμα. Αφότου τελειώσει, το buffer_size ξαναγεμίζει και η διαδικασία επαναλαμβάνεται. Επίσης ανάλογα με το batch_size, μπορεί να αυξηθεί και το num_epoch. Το κόστος μικρότερης τιμής στο num_epoch, εξασφαλίζει πιο σταθερή εκμάθηση αλλά πιο αργή.
- Learning_rate_schedule: Καθορίζει την αλλαγή του ρυθμού μάθησης κατά την διάρκεια την εκπαίδευσης. Εμείς έχουμε δηλώσει linear που προτείνεται από το μοντέλο εκπαίδευσης ppo, ώστε η μάθηση να συγκλίνει πιο σταθερά. Το linear μειώνει γραμμικά τον ρυθμό εκμάθησης, φτάνοντας το μηδέν όταν επιτυγχάνονται τα συνολικά Max_steps.

Network_setting:

- Normalize: Εφαρμόζεται στις εισόδους των διανυσματικών παρατηρήσεων. Βασίζεται στον μέσο όρο και κανονικοποιεί την διακύμανση της παρατήρησης του διανύσματος. Είναι χρήσιμη σε περιπτώσεις με πολύπλοκα συστήματα συνεχούς ελέγχου αλλά σε περιπτώσεις απλών προβλημάτων όπως η δική μας περίπτωση, μπορεί να είναι επιβλαβής για το αποτέλεσμα της εκπαίδευσης. Οι τιμές που μπορεί να πάρει είναι true ή false.
- Hidden_units. Υποδηλώνει τον αριθμό των κρυφών νευρώνων στο νευρωνικό δίκτυο. Ο αριθμός των νευρώνων στο κρυφό επίπεδο θα πρέπει να είναι ανάλογος με την πολυπλοκότητα του προβλήματος. Για απλά προβλήματα όπου οι είσοδοι είναι ένας απλός συνδυασμός παρατηρήσεων, το νούμερο των κρυφών νευρώνων θα πρέπει να είναι μικρό. Για προβλήματα όπου οι παρατηρήσεις είναι περίπλοκες τότε ο αριθμός πρέπει να είναι μεγαλύτερος. Στην δική μας περίπτωση όπου έχουμε ένα σχετικά μεγάλο αριθμό παρατηρήσεων, ο αριθμός που επιλέξαμε για τους κρυφούς νευρώνες είναι 128. Το τυπικό εύρος τιμών που μπορεί να δοθεί είναι μεταξύ του 32-512.
- Num_layers: Δηλώνει τον αριθμό των κρυφών επιπέδων στο νευρωνικό δίκτυο. Αντιστοιχεί στο πόσα κρυφά επίπεδα υπάρχουν μετά την είσοδο παρατήρησης. Για απλά προβλήματα, το να υπάρχουν λιγότερα επίπεδα είναι πιθανό η εκπαίδευση να είναι ταχύτερη και αποτελεσματικότερη. Αντίθετα σε προβλήματα πιο περίπλοκα μπορεί να απαιτούνται περισσότερα επίπεδα. Το τυπικό εύρος τιμών είναι μεταξύ το 1-3.
- Vis_encode_type: Ο τύπος κωδικοποιητή που χρησιμοποιείται για την κωδικοποίηση των παρατηρήσεων. Υπάρχουν τέσσερις επιλογές. Η simple, που είναι μια απλή κωδικοποίηση αποτελείται από δυο συνελικτικά επίπεδα. Το Nature_cnn που χρησιμοποιεί την υλοποίηση cnn και αποτελείται από τρία συνελικτικά επίπεδα. Το

resnet που χρησιμοποιεί το impala resnet και αποτελείται από τρία στοιβαγμένα επίπεδα καθώς και το Match3 που είναι βελτιστοποιημένο κυρίως για επιτραπέζια παιχνίδια.

- Goal_conditioning_type: Είναι η πολιτική που χρησιμοποιείται για τις παρατηρήσεις στόχου. None, σημαίνει ότι αντιμετωπίζει τις παρατηρήσεις στόχου σαν κανονικές παρατηρήσεις. Hyper, αντιμετωπίζει τις παρατηρήσεις στόχου διαφορετικά από τις κανονικές, δημιουργώντας ένα πιο περίπλοκο δίκτυο για αυτό και προτείνεται να ελαττώσουμε τον αριθμό των κρυφών νευρώνων.

Reward_signals: Η ενότητα αυτή καθορίζει τις ρυθμίσεις των ανταμοιβών. Υπάρχουν δυο μέθοδοι καθορισμού των ανταμοιβών ανάλογα εάν ελέγχουν τα εξωγενή η εγγενή σήματα ανταμοιβής, αυτές είναι οι extrinsic και intrinsic. Και οι δυο μέθοδοι θα πρέπει να ορίζουν τουλάχιστον δυο παραμέτρους, την ισχύ (strength) και (gamma). Εμείς θα χρησιμοποιήσουμε το εξωγενές σήμα ανταμοιβής διότι παίρνει τις ανταμοιβές με βάση το περιβάλλον.

- Gamma: Είναι μια παράμετρος που καθορίζει για το πόσο μελλοντικά ο Agent θα πρέπει να ενδιαφερθεί για πιθανές ανταμοιβές. Σε περίπτωση που ο Agent θα πρέπει να κάνει ενέργειες αρκετά νωρίτερα ώστε να προετοιμαστεί για τυχόν μελλοντική ανταμοιβή, τότε η τιμή πρέπει να είναι υψηλή. Αντίθετα εάν οι ανταμοιβές θα είναι άμεσες, τότε η τιμή μπορεί να είναι μικρότερη. Δεν μπορεί να πάρει τιμή ίση η μεγαλύτερη του ενός.
- Strength: Είναι μια μεταβλητή με την οποία πολλαπλασιάζονται οι ανταμοιβές που λαμβάνει ο Agent από το περιβάλλον.

Ύστερα υπάρχουν και άλλες παράμετροι που αποσκοπούν σε διάφορες λειτουργίες.

- Init_path: Η χρησιμότητα του είναι στην περίπτωση που έχουμε ένα ήδη υπάρχον εκπαιδευμένο μοντέλο, να το χρησιμοποιήσουμε για την εκπαίδευση ενός νέου με τις ίδιες παραμέτρους.
- Keep_checkpoints: Δηλώνει τον μέγιστο αριθμό των μοντέλων εκπαίδευσης που θα αποθηκευτούν μέχρι να ολοκληρωθούν τα max_steps που έχουμε δηλώσει ή μέχρι να διακόψουμε χειροκίνητα την εκπαίδευση. Όταν φτάσει στον μέγιστο αριθμό μοντέλων που έχουμε δηλώσει, τότε το νέο μοντέλο αποθηκεύεται στην θέση του παλαιότερου. Στη δική μας περίπτωση έχουμε ορίσει να αποθηκεύονται πέντε μοντέλα κάθε φορά.
- Checkpoint_interval: Δηλώνει κάθε πόσα βήματα θα αποθηκεύεται ένα μοντέλο εκπαίδευσης. Κάθε μοντέλο που αποθηκεύεται έχει την κατάληξη .onnx.
- Max_steps: Είναι ο συνολικός αριθμός των βημάτων δηλαδή οι παρατηρήσεις και οι ενέργειες που έχουν ληφθεί από ένα η πολλαπλά περιβάλλοντα, πριν τον τερματισμό της εκπαίδευσης.
- Time_horizon: Πόσα steps πρέπει να συλλεχθούν ανά Agent πριν προστεθούν στο buffer_size. Όταν επιτευχθεί το όριο που έχει δηλωθεί πριν το τέλος του επεισοδίου, γίνεται μια εκτίμηση για το ποια θα είναι η συνολική ανταμοιβή με βάση την τρέχουσα κατάσταση του Agent. Στην περίπτωση όπου υπάρχουν συχνές ανταμοιβές σε ένα επεισόδιο ή τα επεισόδια είναι πολύ μεγάλης διάρκειας, τότε είναι προ-

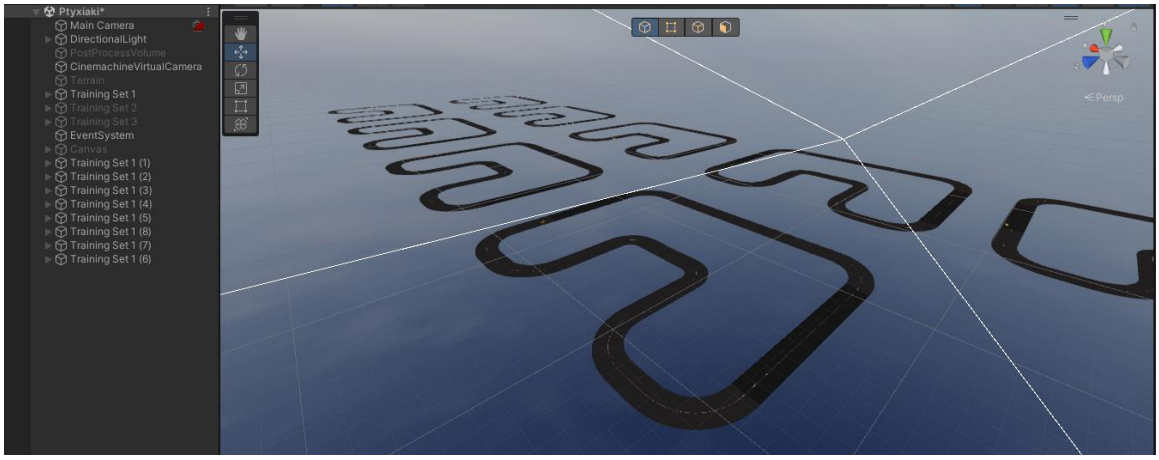
τιμότερο να χρησιμοποιήσουμε ένα μικρότερο αριθμό. Παρόλα αυτά ο αριθμός αυτός θα πρέπει να είναι αρκετά μεγάλος ώστε να καταγράφει όλη την σημαντική συμπεριφορά του Agent καθ΄ όλη την διάρκεια της εκπαίδευσης.

- Summary_freq: Ο αριθμός των βημάτων που πρέπει να ολοκληρωθούν ώστε να δημιουργήσει και να εμφανίσει τα στατιστικά της εκπαίδευσης στην οθόνη.
- Threaded: Επιτρέπει να γίνονται βήματα κατά την διάρκεια ενημέρωσης του μοντέλου εκπαίδευσης. Αυτό οδηγεί σε ταχύτερη εκπαίδευση σε ορισμένες περιπτώσεις.

4.4.2 Εκπαίδευση

Την εκπαίδευση του Agent την έχουμε χωρίσει σε πέντε φάσεις. Για μπορέσουμε να επιτύχουμε το επιθυμητό αποτέλεσμα η εκπαίδευση του Agent θα πρέπει να γίνει σταδιακά. Αρχικά η εκπαίδευση θα γίνει σε μια απλή σχετικά πίστα χωρίς εμπόδια η άλλα αυτοκίνητα ώστε να μάθει τον χειρισμό του αυτοκινήτου. Ύστερα η εκπαίδευση θα συνεχίσει σε μια πιο περίπλοκη πίστα και εφόσον την ολοκληρώσει επιτυχώς τότε θα προστεθούν και τα εμπόδια. Επειδή θα έχουμε πολλαπλά training set, τα εμπόδια στο δρόμο θα βρίσκονται σε διαφορετικές θέσεις ώστε να μάθει να τα αποφεύγει. Όταν ολοκληρωθεί αυτή η φάση, τότε θα μπουν και άλλοι Agent στο ίδιο training set ώστε το τελικό επιθυμητό αποτέλεσμα να είναι ο Agent να μάθει να ολοκληρώνει την πίστα, αποφεύγοντας τα εμπόδια στον δρόμο του και παράλληλα όσο το δυνατόν να αποφεύγει την σύγκρουση με τους άλλους Agent. Η εκπαίδευση χωρίζεται στις πέντε διαφορετικές φάσεις που θα δούμε στην συνέχεια.

Ξεκινάμε την εκπαίδευση στην πρώτη πίστα δημιουργώντας κλώνους των training set ώστε η εκπαίδευση να γίνει ποιο γρήγορα. Όπως βλέπουμε και στην παρακάτω εικόνα, έχουμε δημιουργήσει εννέα training set, όπου στο κάθε ένα υπάρχει μόνο ένας Agent. Επίσης έχουμε αφαιρέσει το terrain για να μειώσουμε τους πόρους που χρειάζονται για την εκπαίδευση.



Εικόνα 38 - Phase One

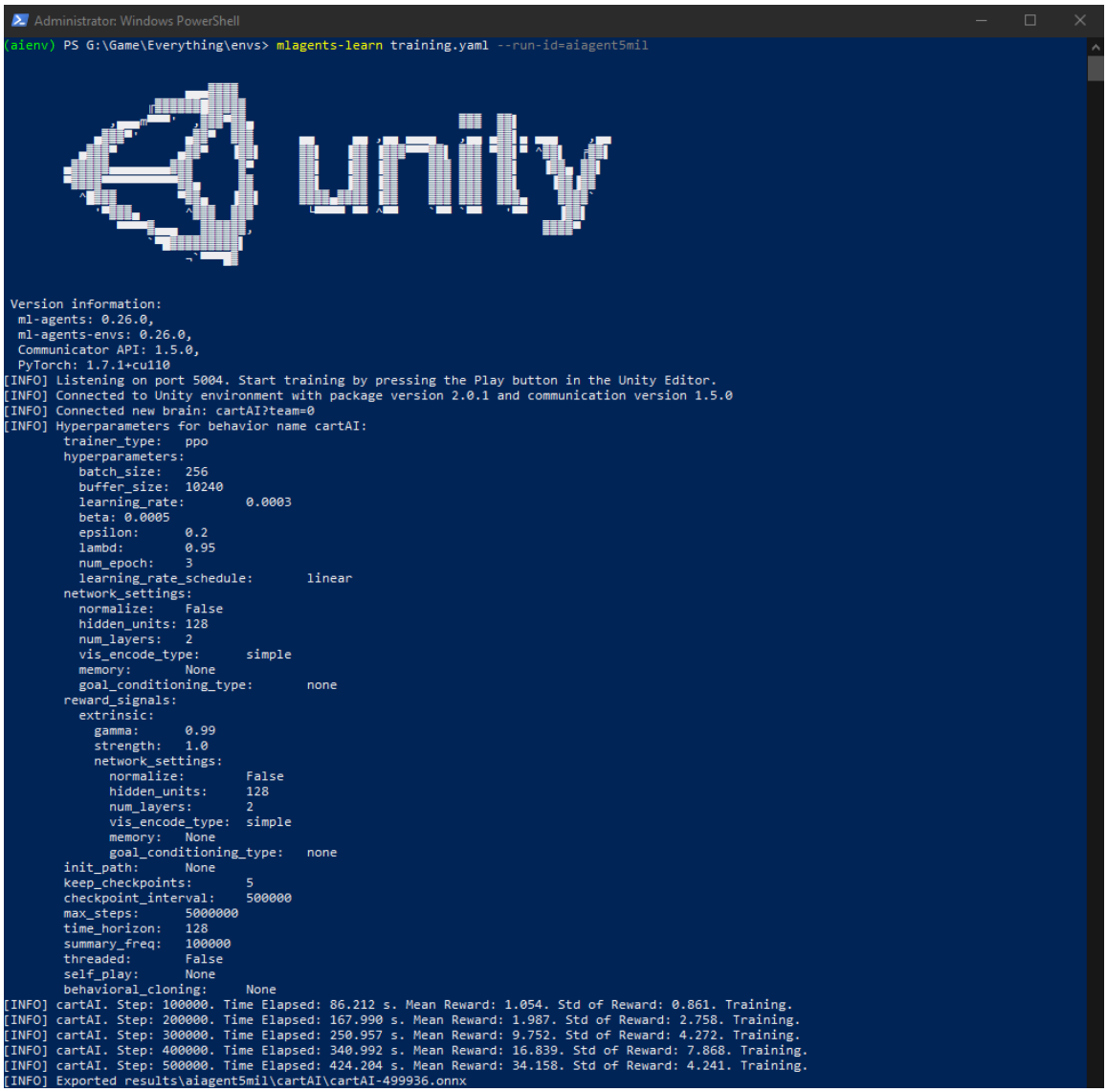
1. Πρώτη Εκπαίδευση

Στόχος της πρώτης φάσης εκπαίδευσης είναι ο Agent να μάθει να προχωράει εμπρός καθώς και να στρίβει το αυτοκίνητο αριστερά και δεξιά. Επίσης πρέπει να μάθει να μένει εντός των ορίων του δρόμου όσο το δυνατόν περισσότερο γίνεται.

Αφού έχουμε ετοιμάσει το περιβάλλον τότε εκτελούμε το powershell και ενεργοποιούμε το εικονικό περιβάλλον που έχουμε δημιουργήσει μέσα στο οποίο και έχουμε το training configuration file training.yaml. Για να ξεκινήσει η εκπαίδευση τρέχουμε την εντολή:

mlagents-learn training.yaml --run-id=aiagent5mil.

Στην εντολή αυτή τρέχουμε την βασική εντολή εκπαίδευσης mlagents-learn, δηλώνουμε το configuration file που έχουμε δημιουργήσει και την εντολή –run-id=aiagent5mil που δηλώνει το όνομα του φακέλου στον οποίο θα αποθηκευτεί το τελικό μοντέλο. Θα ζητηθεί να πατήσουμε το Play button από την Unity για να ξεκινήσει η εκπαίδευση. Όταν ξεκινήσει η εκπαίδευση, στο powershell εμφανίζονται οι παράμετροι που θέσαμε το training.yaml και ανάλογα τα step που έχουμε δηλώσει, μας δείχνει τα rewards που έχει συλλέξει ο Agent από το περιβάλλον.



Εικόνα 39 - Πρώτη Φάση Εκπαίδευσης

Στα πρώτα στάδια της εκπαίδευσης του Agent, παρατηρούμε ότι δεν έχει καμία γνώση για το τί πρέπει να κάνει οπότε πειραματίζεται με βάση τις παραμέτρους κίνησης που του έχουμε δώσει. Παρατηρείται ότι στην αρχή μαθαίνει να κινείται προς τα εμπρός, αριστερά και δεξιά. Ύστερα πειραματίζεται με τον χώρο. Υπήρχαν πολλές φορές που έβγαινε εκτός πίστας όπου και δεχόταν αρνητικό reward. Με αυτόν τον τρόπο κατάλαβε ότι πρέπει να κινείται εντός των ορίων της πίστας. Εφόσον έφτασε το πρώτο checkpoint και πήρε το πρώτο θετικό reward, ξεκίνησε να αναζητά για τα υπόλοιπα. Μετά από μερικά steps παρατηρούμε ότι έχει καταλάβει το πρώτος μέρος της εκπαίδευσης και στόχος του είναι να παίρνει όσο πιο πολλά θετικά reward γίνεται κάνοντας καλύτερο χρόνο.

Παρατηρούμε από το powershell την πορεία της εκπαίδευσης με βάση το στοιχείο mean

rewards όπου είναι η συνολική ανταμοιβή που έχει λάβει κατά την διάρκεια συγκεκριμένου αριθμού steps, τον αριθμών των οποίων έχουμε ορίσει στο configuration file. Βλέπουμε ότι στα πρώτα steps έχει μάθει ικανοποιητικά τον χειρισμό του αυτοκινήτου καθώς και να ακολουθεί τα checkpoints ώστε να ολοκληρώσει την πίστα.

Σε αυτό το σημείο διακόπτουμε την εκπαίδευση πατώντας το Play button από την unity και ξεκινάμε το δεύτερο μέρος της εκπαίδευσης.

2. Δεύτερη Εκπαίδευση

Το δεύτερο μέρος αποτελείται από την ίδια πίστα αλλά αυτή τη φορά με εμπόδια σε διάφορα σημεία του χώρου. Σκοπός της δεύτερης φάσης είναι να μάθει να μπορεί να αποφεύγει τυχών εμπόδια που θα βρίσκονται στο δρόμο του και να βελτιστοποιήσει την κίνηση του στο δρόμο. Αυτό θα το επιτύχουμε με την τοποθέτηση εμποδίων σε διάφορα σημεία του δρόμου, τόσο σε σημεία εύκολα να αποφευχθούν όπως είναι οι ευθείες όσο και σε δύσκολες στροφές για να μπορέσει να εκπαιδευτεί σε όλες τις συνθήκες. Για να συνεχίσουμε την εκπαίδευση αρκεί να τρέξουμε στο Powershell την εντολή:

mlagents-learn training.yaml --run-id=aiagent5mil –resume

Με την εντολή ***–resume*** το μοντέλο θα συνεχίσει την εκπαίδευση από το σημείο που είχε σταματήσει η προηγούμενη εκπαίδευση, αυτό βοηθάει να μάθει κάτι νέο αφού έχει ήδη μια γνώση από πριν.

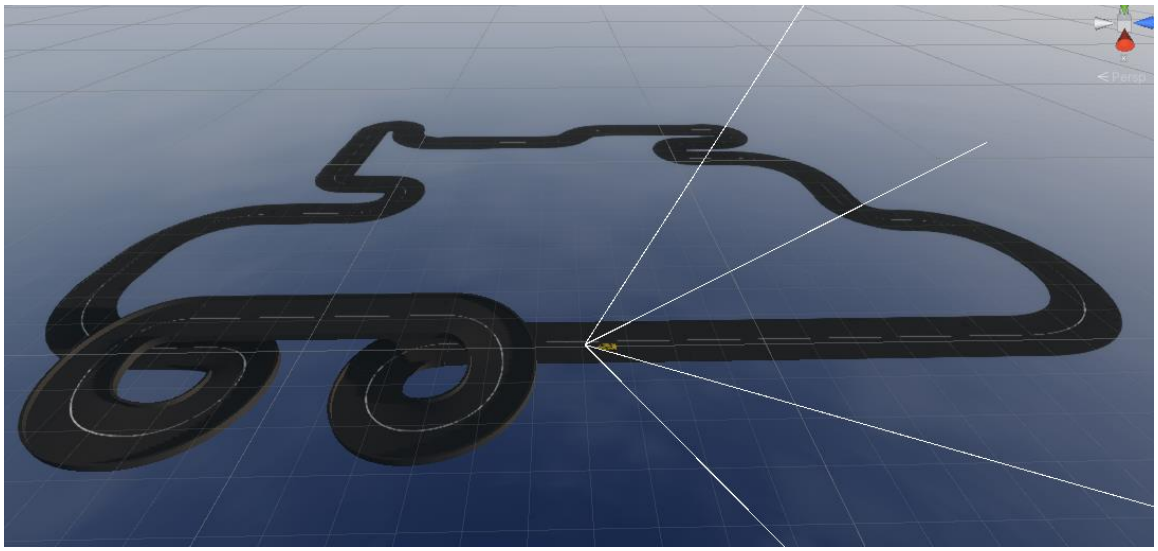
```
[INFO] Resuming from results\aiagent5mil\cartAI.
[INFO] Resuming training from step 504944.
[INFO] cartAI. Step: 600000. Time Elapsed: 76.031 s. Mean Reward: -149.039. Std of Reward: 47.053. Training.
[INFO] cartAI. Step: 700000. Time Elapsed: 153.051 s. Mean Reward: -152.871. Std of Reward: 72.766. Training.
[INFO] cartAI. Step: 800000. Time Elapsed: 230.391 s. Mean Reward: -28.576. Std of Reward: 92.509. Training.
[INFO] cartAI. Step: 900000. Time Elapsed: 306.194 s. Mean Reward: -41.015. Std of Reward: 88.230. Training.
[INFO] cartAI. Step: 1000000. Time Elapsed: 380.333 s. Mean Reward: -0.072. Std of Reward: 102.770. Training.
[INFO] Exported results\aiagent5mil\cartAI\cartAI-999913.onnx
[INFO] cartAI. Step: 1100000. Time Elapsed: 455.495 s. Mean Reward: 20.221. Std of Reward: 92.030. Training.
[INFO] cartAI. Step: 1200000. Time Elapsed: 531.135 s. Mean Reward: 54.975. Std of Reward: 65.177. Training.
[INFO] cartAI. Step: 1300000. Time Elapsed: 608.820 s. Mean Reward: 46.207. Std of Reward: 72.979. Training.
[INFO] cartAI. Step: 1400000. Time Elapsed: 684.205 s. Mean Reward: 64.497. Std of Reward: 38.913. Training.
[INFO] cartAI. Step: 1500000. Time Elapsed: 759.573 s. Mean Reward: 43.518. Std of Reward: 80.461. Training.
[INFO] Exported results\aiagent5mil\cartAI\cartAI-1499938.onnx
```

Εικόνα 40 - Δεύτερη Φάση Εκπαίδευσης

Στη φάση αυτή παρατηρούμε ότι στην αρχή ο Agent δεν γνωρίζει πως πρέπει να ενεργήσει όταν βρεθεί στην πορεία του ένα εμπόδιο, πάρα μόνο να ακολουθεί τον δρόμο ώστε να ολοκληρώσει την πίστα. Στην περίπτωση σύγκρουσης με κάποιο εμπόδιο που βρεθεί στην πορεία του, προσπαθεί να προχωρήσει σαν να μην υπήρχε. Κάθε φορά που χτυπάει και παραμένει στο εμπόδιο δέχεται αρνητικό reward. Αυτό μπορούμε να το δούμε και στην εικόνα 40 όπου από τα πρώτα κιόλας steps έχει αρνητική τιμή το mean reward για αρκετό χρονικό διάστημα. Με το πέρας μερικών step, παρατηρούμε ότι το reward αυξάνεται με μεγάλο ρυθμό. Βλέπουμε όμως ότι στη τελευταία ανασκόπηση της εκπαίδευσης έχουμε μια πτώση του reward, κάτι που οφείλεται στο γεγονός ότι επειδή έχει μάθει αρκετά καλά να αποφεύγει τα εμπόδια, προσπαθεί να αυξήσει τα θετικά reward που παίρνει ψάχνοντας τρόπους να περνάει τα checkpoints γρηγορότερα. Αυτό οδηγεί στο να κάνει περισσότερα

λάθη, να συγκρούεται στα εμπόδια ή να βγαίνει εκτός πίστας. Αυτό έχει ως αποτέλεσμα να μάθει να αποφεύγει τα εμπόδια και στην περίπτωση που συγκρουστεί με κάποιο να κάνει όπισθεν και να συνεχίζει την πορεία του.

Στις επόμενες φάσεις ο Agent θα εκπαιδευτεί στην πιο σύνθετη πίστα, όπου έχει να αντιμετωπίσει περισσότερες και πιο απότομες στροφές καθώς ανηφόρες και κατηφόρες. Εδώ θα γίνει η τελική εκπαίδευση του Agent και θα μάθει να αντιμετωπίζει οποιοδήποτε εμπόδιο συναντήσει στο δρόμο του. Στις προηγούμενες δυο φάσεις εκπαίδευσης, ο Agent έμαθε τις βασικές λειτουργίες κίνησης και απέκτησε την ικανότητα να στρίβει σε απλές στροφές και να αποφεύγει εμπόδια.



Εικόνα 41 - Δεύτερη Πίστα

3. Τρίτη Εκπαίδευση

Στην τρίτη φάση της εκπαίδευσης ο Agent θα εκπαιδευτεί χωρίς τα εμπόδια σε μια πιο περίπλοκη πίστα.

Ξεκινάμε την εκπαίδευση και αμέσως βλέπουμε ότι ο Agent έχει μια γνώση από την προηγούμενη πίστα, κυρίως στις στροφές και στα σημεία όπου ο δρόμος είναι παρόμοιος. Τα σημεία που παρατηρούμε ότι δυσκολεύεται είναι τα σημεία που δεν έχει αντιμετωπίσει ξανά στο παρελθόν. Μετά από μερικά steps, βλέπουμε ότι τα rewards έχουν αυξηθεί και παρατηρώντας τον Agent κατά την διάρκεια της εκπαίδευσης του βλέπουμε ότι το ποσοστό των λαθών για την φάση αυτή έχει μειωθεί αρκετά. Παρόμοια με την πρώτη πίστα, ο Agent αφού έχει μάθει αρκετά καλά την πίστα, προσπαθεί να συλλέξει όσο το δυνατόν περισσότερα rewards. Παρατηρήθηκε λοιπόν ότι σε μερικές απότομες στροφές λόγο ταχύτητας έβγαινε εκτός των ορίων με αποτέλεσμα την πτώση του mean reward περίπου στα μέσα την εκπαίδευσης. Για να διορθώσουμε αυτό το πρόβλημα τοποθετήσαμε reset points στα σημεία τα οποία έβγαινε εκτός της πίστας. Αυτό το κάναμε ώστε η εκπαίδευση να πραγματοποιηθεί ταχύτερα. Επιπλέον παρατείναμε την εκπαίδευση για μερικά ακόμα steps έως ότου το mean reward να είναι όσο το δυνατόν υψηλότερο. Παρόλα αυτά και χωρίς αυτές

τις αλλαγές η εκπαίδευση θα ολοκληρωνόταν με επιτυχία αλλά με πιο αργό ρυθμό. Σκοπός είναι να βοηθήσουμε τον Agent να εκπαιδευτεί καλύτερα και ταχύτερα.

```
[INFO] Resuming from results\aiagent5mil\cartAI.
[INFO] Resuming training from step 1514274.
[INFO] cartAI. Step: 1600000. Time Elapsed: 78.022 s. Mean Reward: 12.459. Std of Reward: 5.420. Training.
[INFO] cartAI. Step: 1700000. Time Elapsed: 159.298 s. Mean Reward: 11.589. Std of Reward: 24.988. Training.
[INFO] cartAI. Step: 1800000. Time Elapsed: 242.331 s. Mean Reward: -5.443. Std of Reward: 44.200. Training.
[INFO] cartAI. Step: 1900000. Time Elapsed: 323.900 s. Mean Reward: 48.925. Std of Reward: 57.971. Training.
[INFO] cartAI. Step: 2000000. Time Elapsed: 406.051 s. Mean Reward: 70.302. Std of Reward: 39.497. Training.
[INFO] Exported results\aiagent5mil\cartAI\cartAI-1999881.onnx
[INFO] cartAI. Step: 2100000. Time Elapsed: 486.275 s. Mean Reward: 89.606. Std of Reward: 28.574. Training.
[INFO] cartAI. Step: 2200000. Time Elapsed: 567.383 s. Mean Reward: 68.246. Std of Reward: 43.168. Training.
[INFO] cartAI. Step: 2300000. Time Elapsed: 648.374 s. Mean Reward: 85.450. Std of Reward: 34.876. Training.
[INFO] cartAI. Step: 2400000. Time Elapsed: 726.390 s. Mean Reward: 38.119. Std of Reward: 38.265. Training.
[INFO] cartAI. Step: 2500000. Time Elapsed: 807.263 s. Mean Reward: 32.193. Std of Reward: 31.517. Training.
[INFO] Exported results\aiagent5mil\cartAI\cartAI-2499911.onnx
[INFO] cartAI. Step: 2600000. Time Elapsed: 887.350 s. Mean Reward: 43.850. Std of Reward: 41.159. Training.
[INFO] cartAI. Step: 2700000. Time Elapsed: 968.244 s. Mean Reward: 63.744. Std of Reward: 47.356. Training.
[INFO] cartAI. Step: 2800000. Time Elapsed: 1047.764 s. Mean Reward: 99.775. Std of Reward: 17.665. Training.
[INFO] cartAI. Step: 2900000. Time Elapsed: 1128.844 s. Mean Reward: 61.736. Std of Reward: 42.911. Training.
[INFO] cartAI. Step: 3000000. Time Elapsed: 1210.702 s. Mean Reward: 63.064. Std of Reward: 42.348. Training.
[INFO] Exported results\aiagent5mil\cartAI\cartAI-2999879.onnx
```

Εικόνα 42 - Τρίτη Φάση Εκπαίδευσης

4. Τέταρτη Εκπαίδευση

Στη τέταρτη φάση της εκπαίδευσης ο Agent έχει μάθει όλες σχεδόν τις λειτουργίες που είχαμε θέσει ως στόχο. Σκοπός της φάσης αυτής είναι να τελειοποιήσει την οδηγική του συμπεριφορά ανεξαρτήτως των εμποδίων που μπορεί να συναντήσει. Ο συνδυασμός δύσκολων στροφών και εμποδίων μας δίνει την δυνατότητα να πετύχουμε το σκοπό αυτό και να βελτιστοποιήσουμε τις λειτουργίες του Agent στο μέγιστο.

Συνεχίζουμε την εκπαίδευση του Agent και παρατηρούμε ότι η πορεία του αλλά και η ικανότητα του να αποφεύγει τα εμπόδια είναι ήδη σε πολύ καλό επίπεδο. Σε αυτό το σημείο βλέπουμε ότι κάνει λάθη όπως και στην Τρίτη φάση όταν τα εμπόδια βρίσκονται σε δύσκολα σημεία. Τα σημεία αυτά είναι κυρίως στροφές όπου δεν έχει προλάβει να δει τα εμπόδια από πριν. Μετά το πέρας μερικών steps και αρκετών αποτυχιών, παρατηρούμε ότι σε οποιοδήποτε σημείο και αν προσθέσουμε κάποιο εμπόδιο καταφέρνει να το αποφύγει. Επίσης όταν βρεθεί σε σημείο να συγκρουστεί με κάποιο εμπόδιο, εντυπωσιακά έμαθε ότι πρέπει να πάει λίγο προς τα πίσω και ύστερα να το προσπεράσει. Θα του πάρει λίγο χρόνο μέχρι να καταλάβει ακριβώς πως πρέπει να το κάνει αλλά στο μεγαλύτερο ποσοστό των περιπτώσεων, θα το προσπεράσει.

```
[INFO] Resuming training from step 3003007.
[INFO] cartAI. Step: 3100000. Time Elapsed: 89.096 s. Mean Reward: 79.525. Std of Reward: 29.550. Training.
[INFO] cartAI. Step: 3200000. Time Elapsed: 173.344 s. Mean Reward: 85.487. Std of Reward: 30.152. Training.
[INFO] cartAI. Step: 3300000. Time Elapsed: 256.426 s. Mean Reward: 86.597. Std of Reward: 26.564. Training.
[INFO] cartAI. Step: 3400000. Time Elapsed: 337.987 s. Mean Reward: 75.743. Std of Reward: 41.525. Training.
[INFO] cartAI. Step: 3500000. Time Elapsed: 421.365 s. Mean Reward: 66.554. Std of Reward: 60.323. Training.
[INFO] Exported results\aiagent5mil\cartAI\cartAI-3499977.onnx
```

Εικόνα 43 - Τέταρτη Φάση Εκαπίδευσης

Η τέταρτη φάση αποτελείται μόνο από 500000 steps καθώς οι απαιτήσεις που είχαμε για την φάση αυτή είχαν ήδη ολοκληρωθεί από τις προηγούμενες. Όπως παρατηρούμε και από την εικόνα 43, η τιμή του mean reward είναι σχετικά σταθερή, πράγμα που προσδιορίζει ότι ο συνδυασμός των ενεργειών του Agent σε αυτή τη φάση έχει επιτευχθεί ικανοποιητικά.

5. Πέμπτη Εκπαίδευση

Στην τελευταία φάση της εκπαίδευσης, τοποθετούμε πέντε ακόμα Agents στην ίδια πίστα. Έτσι ο Agent έρχεται αντιμέτωπος και με άλλα αυτοκίνητα που κινούνται ελεύθερα στην πίστα. Σε κάθε Agent έχουμε ορίσει διαφορετική ταχύτητα ώστε να υπάρχει ανταγωνισμός μεταξύ τους και να εκπαιδευτούν και σε διαφορετικές ταχύτητες.

Μετά το τέλος της τέταρτης φάσης περιμένουμε ο Agent να έχει μάθει πολύ καλά την πίστα και την αποφυγή των εμποδίων και σε αυτή τη φάση θα ολοκληρωθεί η εκπαίδευση του μαθαίνοντας την σωστή αλλά και ανταγωνιστική ταυτόχρονα συμπεριφορά απέναντι στα υπόλοιπα αυτοκίνητα. Να σημειωθεί ότι όλα τα αυτοκίνητα έχουν Tag AicarTrigger. Ο λόγος είναι ότι θέλουμε να αντιλαμβάνεται τα υπόλοιπα αυτοκίνητα σαν κινούμενα εμπόδια.

```
[INFO] Resuming from results\aiagent5mil\cartAI.
[INFO] Resuming training from step 3503057.
[INFO] cartAI. Step: 3600000. Time Elapsed: 52.305 s. Mean Reward: -3.604. Std of Reward: 17.020. Training.
[INFO] cartAI. Step: 3700000. Time Elapsed: 101.377 s. Mean Reward: 25.043. Std of Reward: 35.030. Training.
[INFO] cartAI. Step: 3800000. Time Elapsed: 155.352 s. Mean Reward: 3.814. Std of Reward: 91.440. Training.
[INFO] cartAI. Step: 3900000. Time Elapsed: 212.045 s. Mean Reward: 100.627. Std of Reward: 17.894. Training.
[INFO] cartAI. Step: 4000000. Time Elapsed: 264.018 s. Mean Reward: 104.638. Std of Reward: 23.943. Training.
[INFO] Exported results\aiagent5mil\cartAI\cartAI-3999971.onnx
[INFO] cartAI. Step: 4100000. Time Elapsed: 313.911 s. Mean Reward: 44.272. Std of Reward: 180.864. Training.
[INFO] cartAI. Step: 4200000. Time Elapsed: 365.993 s. Mean Reward: 141.058. Std of Reward: 67.672. Training.
[INFO] cartAI. Step: 4300000. Time Elapsed: 419.251 s. Mean Reward: 70.543. Std of Reward: 101.963. Training.
[INFO] cartAI. Step: 4400000. Time Elapsed: 468.513 s. Mean Reward: 73.755. Std of Reward: 89.497. Training.
[INFO] cartAI. Step: 4500000. Time Elapsed: 521.757 s. Mean Reward: 28.425. Std of Reward: 154.605. Training.
[INFO] Exported results\aiagent5mil\cartAI\cartAI-4499975.onnx
[INFO] cartAI. Step: 4600000. Time Elapsed: 572.256 s. Mean Reward: 16.915. Std of Reward: 211.629. Training.
[INFO] cartAI. Step: 4700000. Time Elapsed: 624.259 s. Mean Reward: 157.614. Std of Reward: 49.516. Training.
[INFO] cartAI. Step: 4800000. Time Elapsed: 676.135 s. Mean Reward: 98.625. Std of Reward: 82.778. Training.
[INFO] cartAI. Step: 4900000. Time Elapsed: 727.035 s. Mean Reward: 126.406. Std of Reward: 51.854. Training.
[INFO] cartAI. Step: 5000000. Time Elapsed: 779.189 s. Mean Reward: 107.629. Std of Reward: 84.215. Training.
[INFO] Exported results\aiagent5mil\cartAI\cartAI-4999876.onnx
[INFO] Exported results\aiagent5mil\cartAI\cartAI-5000004.onnx
[INFO] Copied results\aiagent5mil\cartAI\cartAI-5000004.onnx to results\aiagent5mil\cartAI.onnx.
```

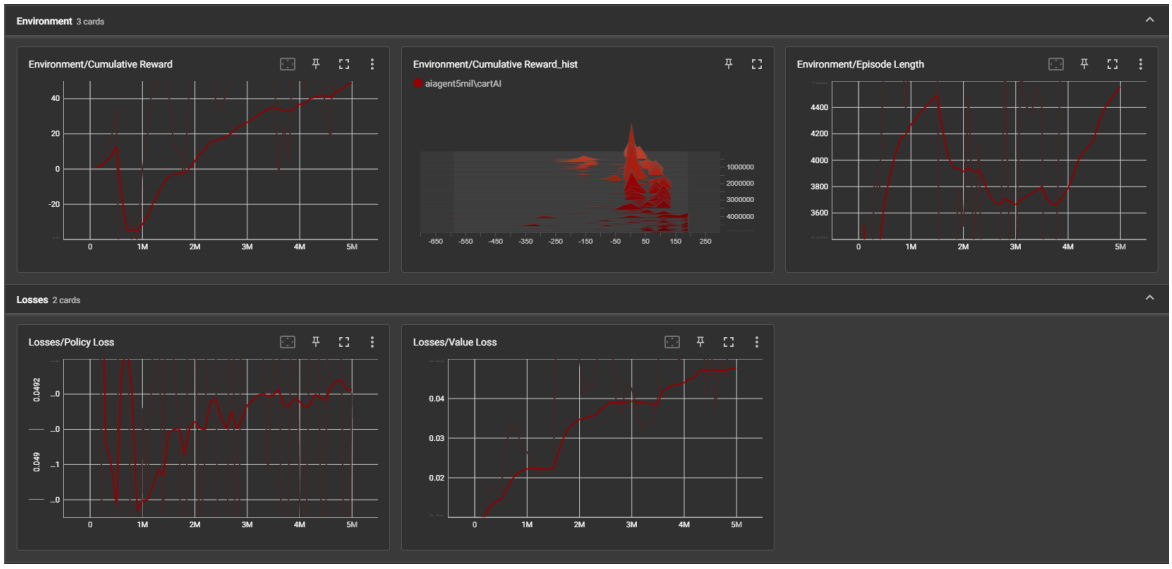
Εικόνα 44 - Πέμπτη Φάση Εκπαίδευσης

Ξεκινάει η τελευταία φάση της εκπαίδευσης και αμέσως βλέπουμε ότι ο Agent κάνει ήδη προσπάθειες να μην συγκρουστεί με τα υπόλοιπα αυτοκίνητα αλλά δεν το καταφέρνει διότι δεν έχει μάθει ακόμα σε κινούμενα εμπόδια. Αυτό οδηγεί σε μερικές συγκρούσεις τις οποίες όμως γρήγορα μαθαίνει να τις αποφεύγει. Παρατηρούμε από πολύ νωρίς στη φάση αυτή ότι τα rewards αυξάνονται σε κάθε step πράγμα που σημαίνει ότι έχει μειώσει τα λάθη και επικεντρώνεται μόνο στην ανταμοιβή. Όπως αναφέραμε, οι Agent έχουν διαφορετική ταχύτητα μεταξύ τους, οπότε υπάρχουν στιγμές όπου πάει να γίνει μια προσπέραση πάνω σε μια στροφή. Σε αυτή την φάση ενδέχεται να χτυπήσουν λόγο του ότι δεν προλαβαίνουν να σταματήσουν και αυτό οδηγεί στην επαναφορά κάποιου Agent.

Αφού ολοκληρωθούν όλες οι φάσεις της εκπαίδευσης, παρατηρούμε ότι στο μεγαλύτερο

ποσοστό των περιπτώσεων ο Agent πέτυχε τον στόχο που είχαμε θέσει από την αρχή με το μεγαλύτερο πιθανό reward. Τα λάθη εννοείται θα υπάρχουν καθώς κανένα AI δεν είναι και ούτε θα γίνει ποτέ τέλειο.

Η Unity μας παρέχει ένα μέσο προβολής των στατιστικών της εκπαίδευσης, το Tensor-Board. Για να δούμε αυτά τα στατιστικά πρέπει να εκτελέσουμε την εντολή *tensorboard --logdir results --port 6006.* Ύστερα ανοίγουμε τον browser μας και γράφουμε localhost:6006. Το αποτέλεσμα της αναζήτησης τα βλέπουμε στην εικόνα 45.



Εικόνα 45 - Στατιστικά Εκπαίδευσης (Enviroment, Losses)

Στο tensor board μπορούμε να δούμε αναλυτικά την πορεία της εκπαίδευσης από το πρώτο μέχρι το τελευταίο step. Στην εικόνα 45 βλέπουμε την ανάλυση των ανταμοιβών καθ΄ όλη την διάρκεια της εκπαίδευσης. Στο πλαίσιο Cumulative Reward και Cumulative Reward_hist, παρατηρούμε ότι στην αρχή είχε μεγάλη τιμή το reward αλλά σύντομα μειώνεται λόγο την περιπλοκότητας της φάσης εκπαίδευσης. Ύστερα παρατηρούμε ότι υπάρχει συνεχόμενη άνοδος του reward μέχρι και την τελευταία φάση της εκπαίδευσης, το οποίο μας υποδηλώνει την επιτυχία της εκπαίδευσης. Στο πλαίσιο Episode Length μπορούμε να δούμε την μέση διάρκεια των επεισοδίων στην πορεία της εκπαίδευσης. Το πλαίσιο αυτό μας δείχνει την διάρκεια που βρισκόταν ο Agent εντός της πίστας. Παρατηρούμε ότι στις πρώτες φάσεις ο Agent έμενε για αρκετή ώρα εντός των ορίων της πίστας. Αργότερα όμως με την εισαγωγή της πιο σύνθετης πίστας, παρατηρούμε ότι η διάρκεια των επεισοδίων μειώνεται διότι ο Agent έβγαινε εκτός των ορίων του δρόμου. Με την πάροδο του χρόνο, βλέπουμε ότι η διάρκεια κατά την οποία παρέμενε εντός της πίστας έχει αυξηθεί και αυξάνεται όλο και περισσότερο όσο προχωράει η εκπαίδευση. Αυτό δηλώνει ότι όσο περισσότερο εκπαιδεύεται ο Agent, τόσο λιγότερα λάθη θα κάνει.

Κεφάλαιο 5: Συμπεράσματα

Η μεγάλη τεχνολογική εξέλιξη των τελευταίων χρόνων έχει ωφελήσει σε πολύ μεγάλο βαθμό την ανάπτυξη των βιντεοπαιχνιδιών και ιδιαίτερα στο κομμάτι της Τεχνητής Νοημοσύνης. Πολλά παιχνίδια ενσωμάτωσαν ευφυείς πράκτορες για να προσδώσουν ρεαλισμό και αληθοφάνεια στην εμπειρία που προσφέρουν. Η χρήση τους στα βιντεοπαιχνίδια ποικίλλει ανάλογα με το είδος του παιχνιδιού. Από εκπαιδευτικά παιχνίδια γνώσεων όπου τα λεγόμενα Bots, ανάλογα με το επίπεδο και τις γνώσεις του χρήστη, προσαρμόζουν τη δυσκολία του παιχνιδιού στο κατάλληλο επίπεδο, έως παιχνίδια πολέμου, όπου τα Bots μπορούν να λειτουργήσουν ως σύμμαχοι ή αντίπαλοι και να προσφέρουν μια πραγματική εμπειρία για τον παίκτη.

Μετά την υλοποίηση του παιχνιδιού, στα πλαίσια της πτυχιακή εργασίας, εξήχθησαν αρκετά συμπεράσματα και διάφορες παρατηρήσεις, όσον αφορά την εφαρμογή και υλοποίηση της τεχνητής νοημοσύνης στα βιντεοπαιχνίδια. Είδαμε ότι δεν είναι απλή διαδικασία να εκπαιδευτεί σωστά ένας Agent και χρειάζεται υπομονή, αρκετό χρόνο και αρκετές προσπάθειες μέχρι να πετύχουμε τον επιθυμητό στόχο. Οι δυσκολίες που αντιμετωπίσαμε αφορούν τόσο το πρακτικό όσο και το θεωρητικό κομμάτι. Υπήρχαν φορές που έπρεπε να διακόψουμε την εκπαίδευση, να προγραμματίσουμε ξανά τις παραμέτρους του Agent και να διορθώσουμε το περιβάλλον εκπαίδευσης καθώς παρατηρήθηκαν δυσκολίες στην εκμάθησή του. Αρκετές ήταν οι στιγμές όπου για αρκετά steps παρέμενε ακίνητο ή κολλημένο σε κάποιο εμπόδιο ή θα έβγαινε επανειλημμένα εκτός των ορίων της πίστας. Κάτι που αξίζει να σημειωθεί είναι ότι στις πρώτες προσπάθειες εκμάθησης, παρατηρήθηκε ότι ο Agent περνούσε επανειλημμένα το ίδιο checkpoint πηγαίνοντας συνεχώς μπροστά- πίσω. Σε αυτό το σημείο καταλάβαμε ότι σκεφτόταν έξυπνα και κάθε φορά που έκανε αυτή την κίνηση έπαιρνε πάντα θετική ανταμοιβή. Αυτό γινόταν διότι δεν είχαμε φτιάξει σωστά το σύστημα για τα checkpoints, το οποίο και διορθώθηκε με την τροποποίηση του κώδικα ώστε ο Agent να παίρνει την ανταμοιβή από το επόμενο σε σειρά σωστό checkpoint. Οι τεχνικές που χρησιμοποιήθηκαν για την αποφυγή των περισσότερων λαθών ήταν κυρίως αλλαγές στο σύστημα ανταμοιβής του Agent καθώς και παραμετροποιήσεις στο περιβάλλον. Για τις ανταμοιβές χρειάστηκε αρκετές φορές να αναπροσαρμόσουμε τις τιμές για κάθε αντικείμενο με το οποίο αλληλοεπιδρά στο περιβάλλον, καθώς παρατηρούσαμε ότι μικρές αλλαγές στην τιμή αυτών, έφερνε μεγάλες αλλαγές στην εκμάθηση του Agent. Αρκετά επίσης ήταν τα εμπόδια κοντά σε απότομες στροφές, όπου ο Agent είχε αποφασίσει ότι καλύτερη τεχνική ήταν να προσπαθεί να κατευθυνθεί από το εξωτερικό μέρος της στροφής, πράγμα που ήταν λάθος στην εκπαίδευση του εγκεφάλου διότι έφευγε συνεχώς εκτός της πίστας.

Παρά τις δυσκολίες που αντιμετωπίσαμε, στο τέλος καταφέραμε να δημιουργήσουμε ένα AI το οποίο εκτελεί ικανοποιητικά όλους τους στόχους που είχαμε θέσει από την αρχή. Σε πολλά σημεία κατάφερε να ξεπεράσει τις προσδοκίες μας, πραγματοποιώντας ενέργειες που δεν είχαμε προβλέψει κατά την διάρκεια της εκπαίδευσης. Βέβαια, η εκπαίδευση θα μπορούσε να βελτιωθεί ακόμα περισσότερο έχοντας τους κατάλληλους πόρους και χρόνο.

Θα μπορούσαν να δημιουργηθούν περισσότερες πίστες εκπαίδευσης όπου θα μάθαινε και σε άλλες συνθήκες δρόμου και ένα πιο ισχυρό υπολογιστή ο οποίος θα ελάττωνε τον χρόνο εκπαίδευσης. Η βελτίωση αυτή θα μας έφερνε καλύτερα αποτελέσματα στην συμπεριφορά του Agent, μειώνοντας το ποσοστό λαθών και εκπληρώνοντας τον στόχο σε όσο το δυνατόν λιγότερο χρόνο γίνεται.

Μπορούμε να πούμε με βεβαιότητα ότι η τεχνητή νοημοσύνη έχει ενσωματωθεί ήδη σε πολύ μεγάλο βαθμό στα βιντεοπαιχνίδια και αποτελεί μέρος της καθημερινότητας για χιλιάδες κόσμο που βιώνει αυτή την εμπειρία μέσα από αυτά. Η τεχνητή νοημοσύνη παρά τις τεχνικές δυσκολίες που φέρνει, είναι ικανή να χαρίσει μια διαδραστική και αληθοφανή εμπειρία ανεξαρτήτως το είδος του παιχνιδιού. Υπάρχουν περιθώρια βελτίωσης της τεχνητής νοημοσύνης από την μεριά της βιομηχανίας των παιχνιδιών, αλλά έχουν καταφέρει να ανταγωνιστούν και να δημιουργήσουν συναισθήματα στους χρήστες πράγμα που αποτελεί μοναδικό γεγονός σε ένα παιχνίδι.

Βιβλιογραφία

1. Russel,S. και Norvig,P. 2004. *Τεχνητή Νοημοσύνη, Μια Σύγχρονη Προσέγγιση*. Κλειδάριθμος
2. Βλαχάβας,Ι. και Βασιλειάδης,Ν. και Κόκκορας,φ. και Σακελλαρίου,Η. και Κεφαλάς,Π. 2011. *Τεχνητή Νοημοσύνη*. Εκδόσεις Πανεπιστημίου Μακεδονίας
3. Nilsson Nils J. 2009. *The Quest for Artificial Intelligence*. Cambridge University Press
4. Alpaydin,E. 2016. *Machine Learning, The New AI*. The MIT Press
5. Kent,Steven,L. 2001. *The Ultimate History of Video Games*. Prima Publishing
6. Lukosek,G.2016. *Learning C# by Developing Games with Unity 5.x*. Packt Publishing
7. IBM Cloud Education. 2020. *Machine Learning*. https://www.ibm.com/clouc/learn/machine-learning
8. Unity- Technologies. 2022. *Ml-agents. https://github.com/Unity-Technologies/ml-agents*
9. Wikipedia. 2022. *Artificial Intelligence*. https://en.wikipedia.org/wiki/Artificial_intelligence
10. Unity. 2021. *Unity Manual. https://docs.unity3d.com/Manual/index.html*
11. Bonzon,T. 2020. *An Introduction to Unity's ML-Agents*. https://gamedevacademy.org/unity-ml-agents-tutorial/
12. Smith,H. 2014. *The History of Game Engines*. https://prezi.com/w56f8xawwcyg/the-history-of-game-engines/
13. Tyler,D. 2022. *How to Choose the Best Video Game Engine*. https://www.gamedesigning.org/career/video-game-engines/