ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ – ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Προηγμένα Συστήματα Πληροφορικής-Ανάπτυξη Λογισμικού και Τεχνητής Νοημοσύνης»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**(Εκπαίδευση ευφυών πρακτόρων μέσω τεχνικών ενισχυτικής μάθησης)** **(Intelligent Agent training using reinforcement learning techniques)**
Ονοματεπώνυμο Φοιτητή	**Κωστάλας Στέφανος-Δρόσος**
Πατρώνυμο	**Γεώργιος**
Αριθμός Μητρώου	**ΜΠΣΠ19023**
Επιβλέπων	**Παναγιωτόπουλος Θεμιστοκλής, Καθηγητής**

Ημερομηνία Παράδοσης **Νοέμβριος 2022**

Τριμελής Εξεταστική Επιτροπή

Θεμιστοκλής Παναγιωτόπουλος Καθηγητής	Διονύσιος Σωτηρόπουλος Επίκουρος Καθηγητής	Ιωάννης Τασούλας Επίκουρος Καθηγητής

Αρχικά, θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή μου κ. Παναγιωτόπουλο Θεμιστοκλή για την ανάθεση της παρούσας εργασίας και την εμπιστοσύνη που μου έδειξε καθ' όλη τη διάρκεια συγγραφής της, καθώς και τις πολύτιμες γνώσεις που μου παρείχε κατά τη διάρκεια των σπουδών μου.

Τέλος, θέλω να ευχαριστήσω τους γονείς μου για την συνεχή υποστήριξη και συμπαράσταση σε όλα αυτά τα χρόνια.

Περίληψη

Οι συνεχόμενες εξελίξεις στον τομέα της Τεχνητής Νοημοσύνης επηρεάζουν σχεδόν όλα τα νέα προϊόντα της τεχνολογίας, καθώς και τις προσδοκίες των ανθρώπων που τα χρησιμοποιούν. Η παρούσα εργασία αφορά στη μελέτη και χρήση της ενισχυτικής μάθησης ως τρόπο εκπαίδευσης ευφυών πρακτόρων για περισσότερο ρεαλιστικές συμπεριφορές σε ηλεκτρονικά παιχνίδια, αλλά και προσομοιώσεις. Συγκεκριμένα, γίνεται αξιοποίηση ενός εργαλείου με το οποίο τα παιχνίδια και οι προσομοιώσεις μπορούν να λειτουργήσουν σαν περιβάλλοντα για την εκπαίδευση αυτών των πρακτόρων χρησιμοποιώντας βαθιά ενισχυτική μάθηση. Το συγκεκριμένο εργαλείο λέγεται ML-Agents και η υλοποίηση αυτή θα γίνει σε περιβάλλον της μηχανής παιχνιδιών Unity.

Abstract

Continuous developments in the area of Artificial Intelligence are affecting almost every new technology product, as well as the expectation of the people using those products. This thesis is about the study and use of reinforcement learning as a way of intelligent agent training for more realistic behaviors in computer games and simulations. In particular, a tool is used which can be utilized for making simulations and computer games work as environments in which agents can be trained using deep reinforcement learning. The specific tool used is called ML-Agents and the application development will be done in the Unity game engine.

Περιεχόμενα

Κατάλογος Εικόνων / Σχημάτων

1 Εισαγωγή

Τα ηλεκτρονικά παιχνίδια εδώ και πολύ καιρό είναι ένας δημοφιλής τομέας για έρευνα στον τομέα της Τεχνητής Νοημοσύνης και υπάρχουν πολύ καλοί λόγοι γι' αυτό. Ένας από αυτούς τους λόγους είναι ότι μπορούν να χρησιμοποιηθούν ως πλατφόρμες για την ανάπτυξη καινούριων μεθόδων Τεχνητής Νοημοσύνης και ως τρόπος μέτρησης για το πόσο ακριβείς είναι. Επιπλέον, μπορούν να αναπαράγουν συμπεριφορές οι οποίες χρήζουν σημαντικού επιπέδου νοημοσύνης, χωρίς να υπάρχει κίνδυνος για υλικές ζημιές ή να τεθούν σε ρίσκο ανθρώπινες ζωές.

1.1 Ευφυείς πράκτορες σε παιχνίδια

Μία από τις κυριότερες προκλήσεις του τομέα της Τεχνητής Νοημοσύνης είναι η δημιουργία ευφυών πρακτόρων οι οποίοι μπορούν να εξελίξουν την ευφυΐα τους στην πάροδο του χρόνου καθώς και την ικανότητά τους να προσαρμόζονται σε διαφορετικές καταστάσεις [1]. Αυτές οι ικανότητες είναι απαραίτητες για τα ρομπότ τα οποία χρησιμοποιούνται σε ανθρώπινα περιβάλλοντα, καθώς και για διάφορες περιπτώσεις λογισμικών τα οποία μπορούν να λειτουργούν ως συνεργατικά ή ως τρόποι βοήθειας σε ανθρώπους που τα χρησιμοποιούν, όπως για παράδειγμα ένας ηλεκτρονικός βοηθός σε ηλεκτρονικές συνομιλίες, φαινόμενο πολύ συχνό τη σημερινή εποχή.

Παρόλο που έχουν αναπτυχθεί αρκετοί τρόποι εκπαίδευσης, έρευνες έχουν δείξει ότι υπάρχουν τομείς στους οποίους διαπιστώνονται σοβαρά μειονεκτήματα. Ένας από τους τομείς αυτούς είναι και ο τομέας του ρεαλισμού. Όπως αναφέρει ο Αβραντινής, ένα στοιχείο του ρεαλισμού συνίσταται σε μεγάλο βαθμό στην επίδειξη αυτόνομης συμπεριφοράς, σε συμφωνία με την προσωπικότητα και τα κίνητρα του πράκτορα, καθώς και με τις συνθήκες του περιβάλλοντος [2]. Για παράδειγμα εάν ένας ευφυής πράκτορας έχει ενσωματωθεί σε ένα παιχνίδι και ο ρόλος του είναι να εκτελεί καταστάσεις οι οποίες απαιτούν ευφυΐα, υπάρχει ανάγκη να γίνει με τρόπο με τον οποίο δε θα αλλάξει η γνώμη του χρήστη για το μέγεθος της ευφυΐας του πράκτορα.

Μεγάλες εταιρείες που ανήκουν στον τομέα της ανάπτυξης ηλεκτρονικών παιχνιδιών έχουν διαθέσει στο ευρύ κοινό λογισμικά τα οποία βοηθούν στην περαιτέρω εξέλιξη των ορίων της Τεχνητής Νοημοσύνης. Το 2016 τίθεται σε λειτουργία το OpenAI Gym, ένα λογισμικό το οποίο χρησιμεύει στην έρευνα για την ενισχυτική μάθηση. Περιλαμβάνει μια αυξανόμενη συλλογή από συγκριτικά προβλήματα που αποκαλύπτουν μια κοινή διεπαφή και έναν ιστότοπο όπου οι χρήστες μπορούν να μοιράζονται τα αποτελέσματά τους και να συγκρίνουν την απόδοση των αλγορίθμων τους [3]. Η εταιρεία Facebook AI αποθέτει στο ευρύ κοινό το ELF (An Extensive, Lightweight, and Flexible Platform for Game Research) για περαιτέρω έρευνα στον τομέα της ενισχυτικής μάθησης [4]. Η εταιρεία Microsoft ανακοίνωσε το The Malmo Platform Project το οποίο είναι μία πλατφόρμα πειραματισμού χτισμένη πάνω στο γνωστό παιχνίδι Minecraft και είναι σχεδιασμένη ώστε να υποστηρίζει τη θεμελιώδη έρευνα πάνω στην Τεχνητή Νοημοσύνη [5].

1.2 Σκοπός της διατριβής

Ο σκοπός της παρούσας πτυχιακής είναι η αξιοποίηση ενός εργαλείου μιας μηχανής παιχνιδιών, για τη δημιουργία ενός συστήματος το οποίο είναι ικανό να εκπαιδεύσει ευφυείς πράκτορες οι οποίοι με τη σειρά τους μπορούν να ενσωματωθούν σε ηλεκτρονικά παιχνίδια, προσομοιώσεις και γενικά τρισδιάστατα περιβάλλοντα.

Για να υλοποιηθεί αυτό το σύστημα, έγινε έρευνα απαιτήσεων και τρόπων υλοποίησης εκπαίδευσης έξυπνων πρακτόρων γενικά, αλλά και πιο συγκεκριμένα σε μηχανές παιχνιδιών και αποφασίστηκε να χρησιμοποιηθεί η μηχανή παιχνιδιών Unity και το εργαλείο ML-Agents.

1.3 Περιγραφή κεφαλαίων

Στο κεφάλαιο αυτό γίνεται μία εισαγωγή στον τομέα των ευφυών πρακτόρων, καθώς και αναφορές σε έρευνες που έχουν γίνει γύρω του. Στο δεύτερο κεφάλαιο αναλύονται πρακτικές εκπαίδευσης ευφυών

πρακτόρων και αναφέρεται ο τρόπος που ακολουθείται στην παρούσα διατριβή. Στο κεφάλαιο 3 αναλύεται το περιβάλλον στο οποίο αναπτύχθηκε η εφαρμογή η οποία αποτελεί μέρος της διατριβής και γίνεται περιγραφή των απαιτήσεων του συστήματος. Στο κεφάλαιο 4 αναφέρεται η σχεδίαση του περιβάλλοντος και οι λόγοι για τους οποίους έχει σχεδιαστεί με αυτόν τον τρόπο. Το κεφάλαιο 5 αναλύει την υλοποίηση της εφαρμογής σε επίπεδο κώδικα και το κεφάλαιο 6 αναλύει τα αποτελέσματα που προέκυψαν από την τελική εκπαίδευση του πράκτορα. Τέλος στο κεφάλαιο 7 παρατίθενται τα συμπεράσματα της διατριβής, καθώς και μελλοντικές επεκτάσεις.

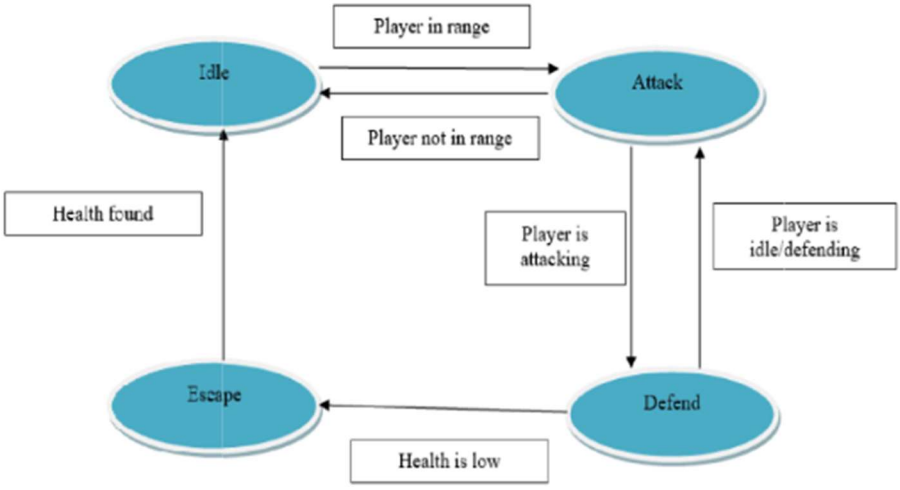
2 Τρόποι εκπαίδευσης

Υπάρχουν αρκετοί τρόποι με τους οποίους μπορεί να προγραμματιστούν οι χαρακτήρες ενός παιχνιδιού, έτσι ώστε να επηρεαστεί η ικανότητα του να παίρνει μόνος του αποφάσεις. Παρ 'όλα αυτά, οι υλοποιήσεις κάποιων από αυτών μπορεί να γίνουν ιδιαίτερα πολύπλοκοι και δύσχρηστοι όσο προστίθεται λειτουργικότητα. Σε αυτό το κεφάλαιο θα γίνει ανάλυση κάποιων από αυτούς.

2.1 Μηχανές πεπερασμένων καταστάσεων (Finite State Machines)

Οι μηχανές πεπερασμένων καταστάσεων είναι ένας από τους πιο κοινούς τρόπους για τον προγραμματισμό NPC (non-player character) στα ηλεκτρονικά παιχνίδια και ίσως και από τους καλύτερους για απλές συμπεριφορές. Μηχανή πεπερασμένων καταστάσεων σχεδιάζεται συνήθως με ένα γράφημα όπου οι κόμβοι αναπαριστούν τις καταστάσεις και οι ακμές τις γραμμές μετάβασης. Στην περίπτωση που αναλύεται, οι κόμβοι μπορούν να αντιστοιχούν σε μία πράξη ή σε μία κατάσταση του ευφυούς πράκτορα. Όλες οι καταστάσεις είναι συνδεδεμένες τουλάχιστον με μία, έτσι ώστε να μπορεί ο πράκτορας να φτάσει σε αυτή. Ο πράκτορας προχωράει από τη μία κατάσταση στην επόμενη μόνο εάν τηρούνται κάποιες συγκεκριμένες συνθήκες, ενημερώνοντας έτσι το ποια συμπεριφορά πρέπει να ακολουθήσει.

Στο παράδειγμα που μας δείχνει ο Jagdale [6, p. 388] και φαίνεται στην Εικόνα 1 Simple FSM παρατηρούμε ότι η μηχανή καταστάσεων αποτελείται από 4 καταστάσεις (κόμβοι) και από 6 μεταβάσεις, οι οποίες ενεργοποιούνται όταν πληρούνται οι συνθήκες που περιγράφονται. Στη συγκεκριμένη περίπτωση, η αρχική κατάσταση είναι η Idle στην οποία ο πράκτορας θα είναι αδρανής, ενώ θα γίνει μετάβαση στην κατάσταση επίθεσης, όταν βρεθεί κάποιος παίκτης στην ακτίνα επίθεσής του. Στη συνέχεια, ο πράκτορας μεταβαίνει στην κατάσταση άμυνας, εάν ο αντίπαλος παίκτης του κάνει επίθεση και μεταβαίνει στην κατάσταση διαφυγής και ψαξίματος ζωής όταν εκείνη πέφτει κάτω από κάποιο επίπεδο. Μόλις βρει κάποια ζωή τότε επιστρέφει στην κατάσταση αδράνειας και είναι σε ετοιμότητα να μεταβεί στην επόμενη κατάσταση.



Εικόνα 1 Simple FSM

Κύρια μειονεκτήματα αυτής της μεθόδου για τον προγραμματισμό της ευφυΐας ενός πράκτορα είναι τα εξής:

- Ο πράκτορας γίνεται αρκετά προβλέψιμος.
- Όσο προστίθενται περισσότερες καταστάσεις η σχεδίαση των μεταβάσεων γίνεται αρκετά πιο πολύπλοκη.
- Δεν είναι ικανή να προγραμματίσει τον πράκτορα για πραγματικά σύνθετες συμπεριφορές.

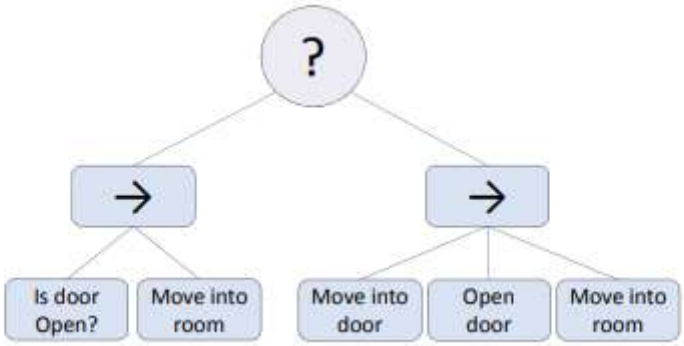
2.2 Δέντρα συμπεριφοράς (Behavior Trees)

Μία ακόμα τεχνική η οποία επίσης χρησιμοποιείται πολύ στον προγραμματισμό NPC και ανήκει στον τομέα της Τεχνητής Νοημοσύνης είναι τα δέντρα συμπεριφοράς. Τα δέντρα συμπεριφοράς επιτρέπουν στους προγραμματιστές να ορίζουν ιεραρχίες αποφάσεων και πράξεων. Βοηθούν επίσης στο να προστίθενται στοιχεία αληθοφάνειας στη συμπεριφορά του πράκτορα, καθώς και περισσότερη διαδραστικότητα.

Ένα δέντρο συμπεριφοράς είναι ένα κατευθυνόμενο γράφημα το οποίο περιέχει κόμβους και ακμές. Η ρίζα του είναι ένας κόμβος χωρίς γονείς και οι κόμβοι χωρίς παιδιά είναι τα φύλλα του [7, p. 3]. Σε ένα βασικό δέντρο συμπεριφοράς, ένας κόμβος χωρίς φύλλα μπορεί να είναι ένας κόμβος επιλογής ή ένας κόμβος ακολουθίας. Οι κόμβοι επιλογής χρησιμοποιούνται όταν σκοπός μας είναι να βρούμε και να εκτελέσουμε τον πρώτο κόμβο-παιδί που μπορεί να τρέξει χωρίς να υπάρξει αποτυχία. Ο κόμβος επιλογής επιτυγχάνει μόνο όταν κάποιος κόμβος-παιδί έχει εκτελεστεί με επιτυχία. Σε έναν κόμβο ακολουθίας όλοι οι κόμβοι αξιοποιούνται διαδοχικά και ο κόμβος επιτυγχάνει μόνο εάν όλοι οι κόμβοι-παιδιά έχουν εκτελεστεί επιτυχώς. Στην περίπτωση που αναλύεται, ένας κόμβος-φύλλο μπορεί να είναι μία πράξη ή μία κατάσταση, τα οποία μπορεί να είναι η ενεργοποίηση ενός animation, η αλλαγή της κατάστασης του πράκτορα ή οποιαδήποτε άλλη δραστηριότητα η οποία μπορεί να αλλάξει την κατάσταση του παιχνιδιού.

Η ιδέα είναι ότι οι συμπεριφορές εκτελούνται είτε μία κάθε φορά είτε με τη σειρά. Η απόφαση για το ποια συμπεριφορά θα εκτελεστεί αξιολογείται με συγκεκριμένο ρυθμό. Σε κάθε κύκλο αξιολογείται ολόκληρο το δέντρο και εάν επιλεγεί άλλη συμπεριφορά από την τελευταία που επικαλέστηκε, η κατάσταση εκτέλεσης μπορεί να αλλάξει, διαφορετικά η συμπεριφορά συνεχίζει να εκτελείται. Με αυτόν τον τρόπο, ξεπερνιούνται ορισμένες από τις ελλείψεις των FSM, όπως το να κολλήσουν σε μια κατάσταση και να έχουν απρόσμενες αλλαγές καταστάσεων.

Στην Εικόνα 2Εικόνα 2 Παράδειγμα δέντρου συμπεριφοράς, μπορούμε να δούμε ένα παράδειγμα ενός τέτοιου δέντρου συμπεριφοράς, το οποίο περιέχει έναν κόμβο επιλογής στη ρίζα του δέντρου αναπαριστώμενο από ένα ερωτηματικό (?), δύο κόμβους ακολουθίας στα παιδιά της ρίζας τα οποία αναπαρίστανται από τα βέλη και 4 κόμβους δράσης. Ο πρώτος κόμβος που εκτελείται σε αυτό το δέντρο είναι η ερώτηση εάν είναι η πόρτα ανοιχτή. Εάν είναι ανοιχτή επιστρέφει επιτυχώς και ακολουθεί το επόμενο παιδί που είναι η είσοδος στο δωμάτιο. Εάν και η είσοδος στο δωμάτιο επιστρέψει επιτυχώς, τότε όλο το δέντρο είναι επιτυχές και ολοκληρώνεται έτσι ένας κύκλος εκτέλεσης. Στην περίπτωση που η πόρτα είναι κλειστή ή η είσοδος στο δωμάτιο είναι ανεπιτυχής το δέντρο θα προχωρήσει στον επόμενο κόμβο ακολουθίας και θα εκτελέσει διαδοχικά τις επόμενες πράξεις.



Εικόνα 2 Παράδειγμα δέντρου συμπεριφοράς

Κύρια μειονεκτήματα αυτής της μεθόδου είναι τα εξής:

- Ο σχεδιασμός πολύπλοκων συμπεριφορών μπορεί να είναι αρκετά απαιτητικός και δύσκολος.
- Για πολύ μεγάλα δέντρα συμπεριφοράς, αυξάνεται η πολυπλοκότητα του κώδικα και έτσι αυξάνεται ο χρόνος υπολογισμού της απόφασης.

2.3 Σχεδιασμός δράσης με προσανατολισμό στον στόχο (Goal oriented Action Planning)

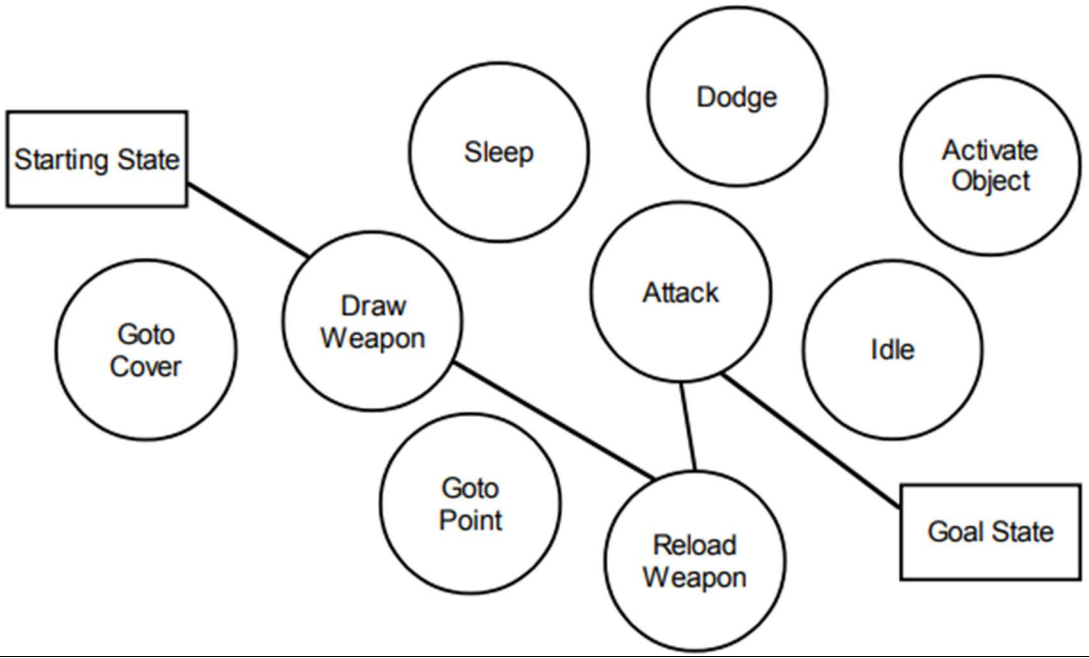
Ο σχεδιασμός δράσης με προσανατολισμό στον στόχο ή αλλιώς GOAP (Goal Oriented Action Planning) είναι μία μέθοδος προγραμματισμού ευφυών πρακτόρων, η οποία τους επιτρέπει να σχεδιάζουν δυναμικά μία ακολουθία πράξεων που θα ικανοποιήσει έναν επιθυμητό στόχο. Η ακολουθία των πράξεων η οποία επιλέγεται από τον πράκτορα εξαρτάται από την τρέχουσα κατάσταση του πράκτορα και την τρέχουσα κατάσταση του περιβάλλοντος στο οποίο βρίσκονται. Από αυτό προκύπτει ότι δύο διαφορετικοί πράκτορες με τον ίδιο στόχο μπορεί να ακολουθήσουν δύο εντελώς διαφορετικές ακολουθίες πράξεων.

Τέσσερις είναι οι βασικές έννοιες του GOAP[8, p. 1]:

- **Στόχος (Goal):** Στόχος είναι κάθε συνθήκη που ο πράκτορας θέλει να επιτύχει. Ένας πράκτορας μπορεί να έχει παραπάνω από έναν στόχους. Κάθε στιγμή μόνο ένας είναι ενεργός, ορίζοντας έτσι τη συμπεριφορά του πράκτορα. Τη στιγμή που ενεργοποιείται ένας στόχος, ο χαρακτήρας κάνει μία προκαθορισμένη ακολουθία βημάτων η οποία ορίζεται από αυτόν.
- **Σχέδιο (Plan):** Το σχέδιο είναι μία ακολουθία πράξεων η οποία θα μεταφέρει τον πράκτορα από την αρχική κατάσταση σε μία κατάσταση στην οποία ικανοποιείται ο στόχος.
- **Πράξη (Action):** Η πράξη είναι ένα απλό βήμα το οποίο είναι μέρος ενός σχεδίου που κάνει τον πράκτορα να κάνει κάτι. Η διάρκειά της μπορεί να είναι είτε μικρή είτε απεριόριστη. Μία πράξη έχει προϋποθέσεις για να εκτελεστεί και προκαλεί επιδράσεις στο περιβάλλον του παιχνιδιού όταν εκτελείται.
- **Οργανωτής (Planner):** Το σχέδιο που ακολουθεί ένας πράκτορας είναι εκείνο που παρέχει έναν στόχο ο οποίος ικανοποιεί έναν οργανωτή. Ο GOAP planner αναλύει τις προϋποθέσεις και τις επιδράσεις των πράξεων, ώστε να αποφασίσει μία σειρά από πράξεις που ικανοποιούν τον στόχο. Ο επιθυμητός στόχος δίνεται στον πράκτορα μαζί με την κατάσταση του περιβάλλοντος και μίας λίστας από πράξεις οι οποίες μπορούν να εκτελεστούν. Εάν ο οργανωτής είναι επιτυχής, επιστρέφει ένα σχέδιο για να ακολουθήσει ο πράκτορας, το οποίο εκτελείται μέχρι να ολοκληρωθεί, να ακυρωθεί, ή όταν κάποιος πιο σχετικός στόχος ανακαλυφθεί στην πορεία. Στην περίπτωση που κάποιος σχετικότερος στόχος ανακαλυφθεί, ο προηγούμενος ακυρώνεται και ο οργανωτής δημιουργεί ένα καινούριο σχέδιο.

Στην Εικόνα 3 περιγράφεται ένα παράδειγμα της διαδικασίας σχηματισμού του σχεδίου. Τα ορθογώνια αναπαριστούν την αρχική κατάσταση και την κατάσταση επίτευξης στόχου και κάθε κύκλος αναπαριστά μία πράξη. Ο στόχος στη συγκεκριμένη εικόνα είναι να εξοντωθεί ο εχθρός, άρα ο Οργανωτής πρέπει να υπολογίσει μία ακολουθία πράξεων με την οποία ο πράκτορας θα μεταβεί από την τρέχουσα κατάστασή του στην κατάσταση στην οποία ο εχθρός θα έχει εξοντωθεί. Σε πολλές περιπτώσεις μπορεί να υπάρχουν παραπάνω από ένα σχέδια, οπότε υπολογίζεται εκείνο του οποίου οι διαδοχικές πράξεις έχουν λιγότερο κόστος.

Κύριο μειονέκτημα του GOAP είναι η απόδοση λόγω της φύσης του, η οποία ορίζει ότι κάθε φορά που πρέπει να παράγει ένα σχέδιο πρέπει να υπολογίζει την κατάσταση του περιβάλλοντος γύρω του, ενώ για παράδειγμα τα δέντρα συμπεριφοράς που προαναφέρθηκαν θα επέτρεπαν να γίνει κάποια πράξη αμέσως.



Εικόνα 3 Διαδικασία σχηματισμού σχεδίου

2.4 Ενισχυτική μάθηση (Reinforcement Learning)

Η ενισχυτική μάθηση είναι μία τεχνική μηχανικής μάθησης η οποία μπορεί να χρησιμοποιηθεί για την εκπαίδευση ευφυών πρακτόρων. Στην τεχνική αυτή ένας πράκτορας εκπαιδεύεται κάνοντας πράξεις οι οποίες αλλάζουν την κατάσταση ενός περιβάλλοντος. Μετά από κάθε πράξη ο πράκτορας μπορεί να λάβει μία "επιβράβευση" από το περιβάλλον, του οποίου η αξία εξαρτάται από το πόσο κοντά είναι ο πράκτορας στο να έχει το επιθυμητό αποτέλεσμα. Ο στόχος του πράκτορα είναι να μεγιστοποιήσει το αθροιστικό αποτέλεσμα που λαμβάνει στην πάροδο του χρόνου. Μετά από μία σειρά πράξεων, ο πράκτορας τελικά φτάνει σε μία κατάσταση επίτευξης στόχου ή σε μία τερματική κατάσταση η οποία υποδηλώνει το τέλος ενός επεισοδίου. Τότε, το περιβάλλον και ο πράκτορας επαναφέρονται στις αρχικές τους καταστάσεις και η διαδικασία επαναλαμβάνεται.

Τυπικά, ένα μοντέλο ενισχυτικής μάθησης είναι ένα σύνολο από καταστάσεις S, ένα σύνολο από πράξεις A, και κανόνες μετάβασης μεταξύ καταστάσεων βάση της δράσης που έχει ακολουθηθεί [9]. Για την κατάσταση s που ανήκει στο σύνολο S στον χρόνο t, ένας πράκτορας εκτελεί μία δράση a που ανήκει στο σύνολο A, μεταβαίνει σε μία καινούρια κατάσταση s' και λαμβάνει μία επιβράβευση μεγέθους r_t. Ο πράκτορας τότε ακολουθεί μία τακτική p για να επιλέξει την πράξη που θα εκτελέσει και π(s, a) είναι η πιθανότητα ενός πράκτορα να επιλέξει την πράξη a όταν βρίσκεται στην κατάσταση s. Ο σκοπός του πράκτορα είναι να μεγιστοποιήσει την επιβράβευση R_t (Εικόνα 4).

$$R_t = \sum_{k=0}^{\infty} \lambda^k r_{t+k}$$

Εικόνα 4 Τύπος υπολογισμού επιβράβευσης

Υπάρχουν δύο τύποι επιβράβευσης:

- **Θετική:** Θετική επιβράβευση ορίζεται ως το γεγονός όταν μια ακολουθία πράξεων έχει φέρει το επιθυμητό αποτέλεσμα και ενισχύει έτσι τη συχνότητα και την ένταση της συγκεκριμένης συμπεριφοράς.

- **Αρνητική:** Αρνητική επιβράβευση ορίζεται ως η ενίσχυση μιας συμπεριφοράς λόγω της αποφυγής μίας συγκεκριμένης συνθήκης η οποία φέρει αρνητικό αποτέλεσμα.

Αποτελεί έναν από τους μεγαλύτερους τομείς έρευνας τα τελευταία χρόνια στον τομέα εκπαίδευσης αυτόνομων ευφυών πρακτόρων, όπως αναφέρθηκε στο κεφάλαιο 1 και έναν από τους περισσότερο αποτελεσματικούς στον τομέα των ηλεκτρονικών παιχνιδιών. Παράδειγμα αποτελεσματικότητας είναι ότι τον Απρίλιο του 2019 το OpenAI Five έγινε το πρώτο σύστημα τεχνητής νοημοσύνης που κατάφερε να κερδίσει τους παγκόσμιους πρωταθλητές στο ηλεκτρονικό παιχνίδι της Dota 2 [10]. Το παιχνίδι της Dota 2 παρουσιάζει ιδιαίτερες προκλήσεις για τα συστήματα Τεχνητής Νοημοσύνης, όπως η παροχή ημιτελούς πληροφορίας, οι μεγάλοι χρονικοί ορίζοντες και οι συνεχείς και πολύπλοκες αλλαγές καταστάσεων. Αναπτύχθηκε ένα κατανεμημένο σύστημα εκπαίδευσης, καθώς και εργαλεία για συνεχή εκπαίδευση τα οποία επέτρεψαν να γίνει εκπαίδευση για 10 μήνες. Η νίκη αυτή επί των παγκόσμιων πρωταθλητών αναδεικνύει ότι η ενισχυτική εκπαίδευση μπορεί να αποκτήσει ιδιαίτερα μεγάλη απόδοση σε ένα δύσκολο καθήκον.

Η παρούσα διατριβή χρησιμοποιεί τη μέθοδο της ενισχυτικής μάθησης χάριν του εργαλείου που χρησιμοποιείται για την εκπαίδευση και χάριν της πολυπλοκότητας της συμπεριφοράς που αναμένεται να ακολουθεί ο πράκτορας.

3 Περιβάλλον ανάπτυξης και απαιτήσεις εφαρμογής

Στο κεφάλαιο αυτό περιγράφεται η πλατφόρμα στην οποία υλοποιήθηκε η εφαρμογή (Unity3D) και το εργαλείο της για εκπαίδευση ευφυών πρακτόρων (ML-Agents), καθώς και τα υπόλοιπα εργαλεία που χρησιμοποιήθηκαν για να ολοκληρωθεί η εκπαίδευση.

3.1 Unity

Η Unity είναι μια πλατφόρμα ανάπτυξης τρισδιάστατου περιεχομένου σε πραγματικό χρόνο, η οποία αποτελείται από μία μηχανή απεικόνισης, μία μηχανή προσομοίωσης φυσικών φαινομένων, καθώς και μία διεπαφή χρήστη για γραφικά γνωστή ως Unity Editor. Η Unity έχει γίνει ευρέως γνωστή στον τομέα των ηλεκτρονικών παιχνιδιών, καθώς και στους τομείς της Αρχιτεκτονικής, της Μηχανικής, των κατασκευών, της αυτοκινητοβιομηχανίας και του κινηματογράφου και χρησιμοποιείται από μία μεγάλη κοινότητα προγραμματιστών, οι οποίοι την αξιοποιούν για να δημιουργήσουν διαδραστικές προσομοιώσεις, από εφαρμογές κινητών και παιχνίδια με βάση τον περιηγητή μέχρι υψηλού επιπέδου παιχνίδια κονσόλας και εμπειρίες εικονικής και επαυξημένης πραγματικότητας [11].

Η συγκεκριμένη μηχανή γραφικών από την αρχή της ιστορίας της είχε ως κύριο σκοπό την ανάπτυξη μιας μηχανής γραφικών, η οποία υποστηρίζει διάφορες πλατφόρμες, επίπεδα εμπειρίας προγραμματιστών και τύπους παιχνιδιών, κάτι το οποίο την καθιστά ιδανική επιλογή για έρευνα στην Τεχνητή Νοημοσύνη. Η ευελιξία της μηχανής αυτής επιτρέπει την δημιουργία προϊόντων, από απλά δισδιάστατα μέχρι πολύπλοκα τρισδιάστατα παιχνίδια στρατηγικής, παιχνίδια γρίφων με βάση τη φυσική και ανταγωνιστικά παιχνίδια με πολλαπλούς πράκτορες.

Περιέχει επίσης ενσωματωμένη αποθήκη με έτοιμα assets τα οποία είτε προέρχονται είτε από άλλες εταιρίες είτε από άλλους προγραμματιστές, είτε είναι επίσημα assets της Unity. Κάποια από αυτά είναι δωρεάν, ενώ κάποια επί πληρωμή. Η εφαρμογή που αναλύεται σε αυτή τη διατριβή χρησιμοποιεί την 2021.3.6f1 εκδοχή της Unity.

3.2 Unity ML-Agents

Η εργαλειοθήκη ML-Agents(Machine Learning Agents) είναι ένα λογισμικό ανοιχτού πηγαίου κώδικα, το οποίο επιτρέπει σε ερευνητές και προγραμματιστές να δημιουργήσουν περιβάλλοντα προσομοίωσης χρησιμοποιώντας τον Unity Editor και να αλληλοεπιδράσουν μαζί τους μέσω ενός Python API. Παρέχει το ML-Agents SDK (κιτ ανάπτυξης λογισμικού) το οποίο περιέχει όλες τις απαραίτητες λειτουργίες για την ανάπτυξη περιβαλλόντων μέσω του Unity Editor και βασικά σενάρια C# για την εκπαίδευση ευφυών πρακτόρων.

Οι τρεις βασικές οντότητες στο SDK είναι οι Αισθητήρες (Sensors), οι πράκτορες (Agents) και η Ακαδημία (Academy). Το στοιχείο του Agent χρησιμοποιείται για να δείξει ότι ένα αντικείμενο μέσα στο περιβάλλον είναι ένας πράκτορας, οπότε μπορεί να συλλέξει παρατηρήσεις, να κάνει πράξεις και να δεχτεί επιβραβεύσεις. Ο κάθε Agent μπορεί να συλλέξει παρατηρήσεις χρησιμοποιώντας με διαφορετικούς τρόπους συλλογής πληροφορίας, όπως για παράδειγμα αποτελέσματα από ray-casts. Κάθε πράκτορας περιέχει έναν κανόνα με τίτλο behavior name (όνομα συμπεριφοράς).

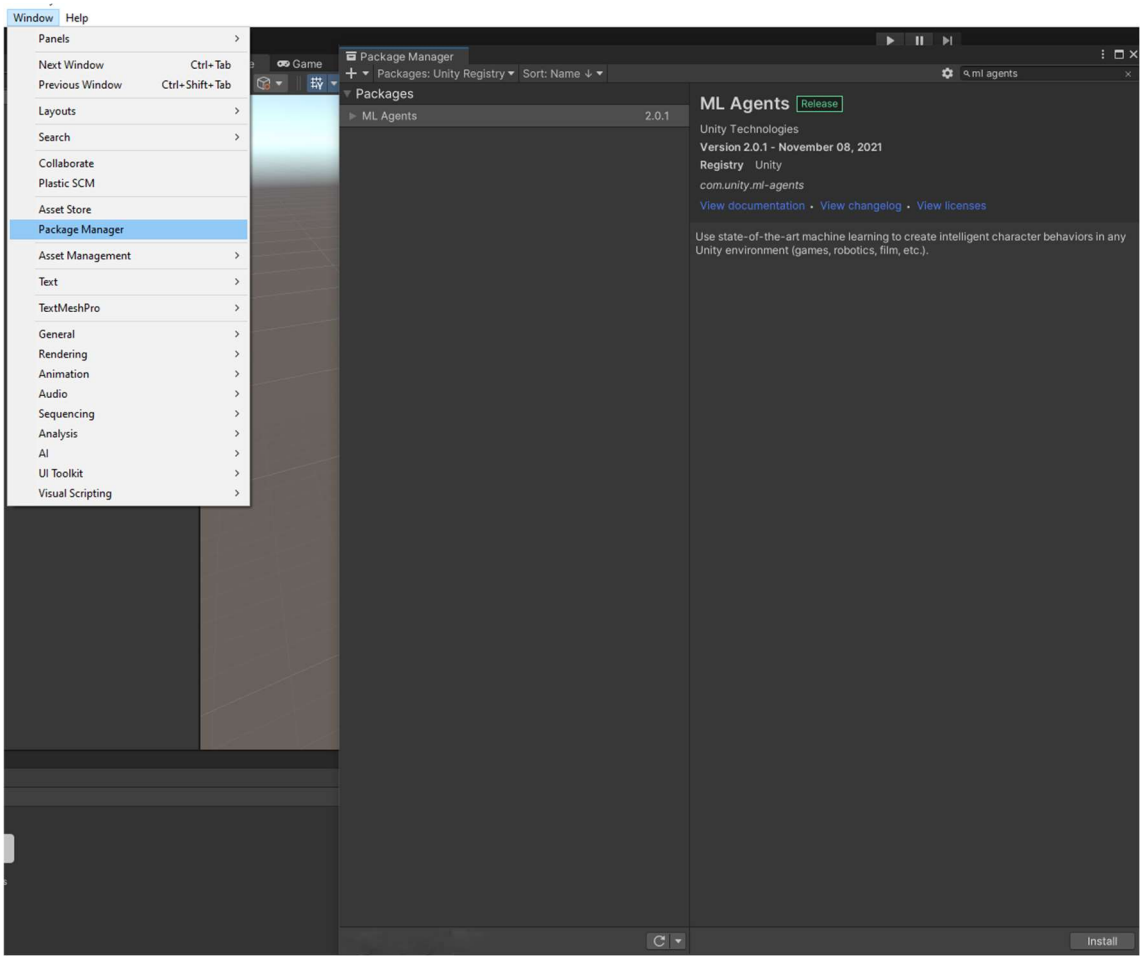
Πολλοί agents παράλληλα μπορούν να περιέχουν κανόνα με τη συγκεκριμένη συμπεριφορά,

εκτελώντας έτσι παράλληλα την ίδια συμπεριφορά και μοιράζοντας έτσι τα δεδομένα εκπαίδευσης. Με αυτόν τον τρόπο μπορεί να πολλαπλασιαστεί ο χρόνος εκπαίδευσης του πράκτορα και έτσι να παραχθεί πιο γρήγορα το νευρωνικό δίκτυο που αποτελεί τον εγκέφαλό του.

Η συνάρτηση επιβράβευσης μπορεί οποιαδήποτε στιγμή κατά τη διάρκεια της προσομοίωσης να οριστεί ή να προσαρμοστεί χρησιμοποιώντας το σύστημα scripting της Unity. Αντίστοιχα, η κατάσταση του πράκτορα μπορεί να οριστεί ως τελική χρησιμοποιώντας κάποιο script ή φθάνοντας σε ένα προκαθορισμένο μέγιστο πλήθος βημάτων.

Η Academy (Ακαδημία) χρησιμοποιείται για να ελέγχει τα βήματα της προσομοίωσης και τους πράκτορες. Επίσης έχει τη δυνατότητα να ορίζει τις παραμέτρους του περιβάλλοντος, οι οποίες μπορούν να χρησιμοποιηθούν για να αλλάξουν τις ρυθμίσεις του σε πραγματικό χρόνο.

Η εγκατάσταση του πακέτου γίνεται πηγαίνοντας στον package manager και επιλέγοντας packages: Unity Registry, όπως φαίνεται στην Εικόνα 5. Έπειτα, πληκτρολογώντας το όνομα του πακέτου στο επάνω δεξιά μέρος, όπου υπάρχει ένα πεδίο για αναζήτηση πακέτων, εμφανίζεται κατευθείαν, ενώ στο κάτω δεξιά μέρος υπάρχει ένα κουμπί, το οποίο όταν πατηθεί, γίνεται εγκατάστασή του. Για τη συγκεκριμένη εφαρμογή έγινε χρήση της εκδοχής 2.0.1 των ML Agents.



Εικόνα 5 Εγκατάσταση ML Agents

3.3 Πακέτο Python

Το παρεχόμενο πακέτο της γλώσσας Python [12] συμπεριλαμβάνει μία κλάση που λέγεται UnityEnvironment, η οποία μπορεί να χρησιμοποιηθεί για την εκκίνηση και τη διασύνδεση με εκτελέσιμα αρχεία της Unity συμπεριλαμβανόμενου του Unity editor και περιέχει τα απαιτούμενα στοιχεία που περιγράφηκαν στο προηγούμενο κεφάλαιο. Η επικοινωνία μεταξύ Python και Unity πραγματοποιείται μέσω ενός πρωτοκόλλου επικοινωνίας gRPC [13] και χρησιμοποιεί μηνύματα protobuff [14].

Στη συγκεκριμένη υλοποίηση χρησιμοποιήθηκε η έκδοση της python 3.7.6 και ο ενσωματωμένος διαχειριστής πακέτων pip, ο οποίος έρχεται, δεν χρειάζεται ξεχωριστή εγκατάσταση στις εκδόσεις της python οι οποίες είναι μεγαλύτερες από την 3.4.0. Επιπλέον, χρησιμοποιήθηκε για την καλύτερη διαχείριση των πακέτων το pipenv [15], ένα εργαλείο υποβοήθησης εγκατάστασης και απεγκατάστασης πακέτων της Python.

Το βασικό πακέτο της Python που χρειάζεται για τη χρήση της εφαρμογής που υλοποιείται σε αυτή την διατριβή είναι το mlagents και η εκδοχή του είναι η 0.28.0. Η εντολή εγκατάστασής του είναι: pipenv install mlagents==0.28.0. Μαζί με αυτό εγκαθίστανται και τα προαπαιτούμενα πακέτα για να τρέχει σωστά η εφαρμογή κάποια από τα οποία αξίζουν να παρατεθούν, καθώς είτε έχουν ήδη αναφερθεί είτε θα αναφερθούν σε επόμενα κεφάλαια.

- mlagents (0.28.0)
- pip (22.2.2)
- protobuf (3.19.6)
- PyYAML (6.0)
- tensorboard (2.10.1)
- torch (1.12.1)

4 Σχεδίαση εφαρμογής

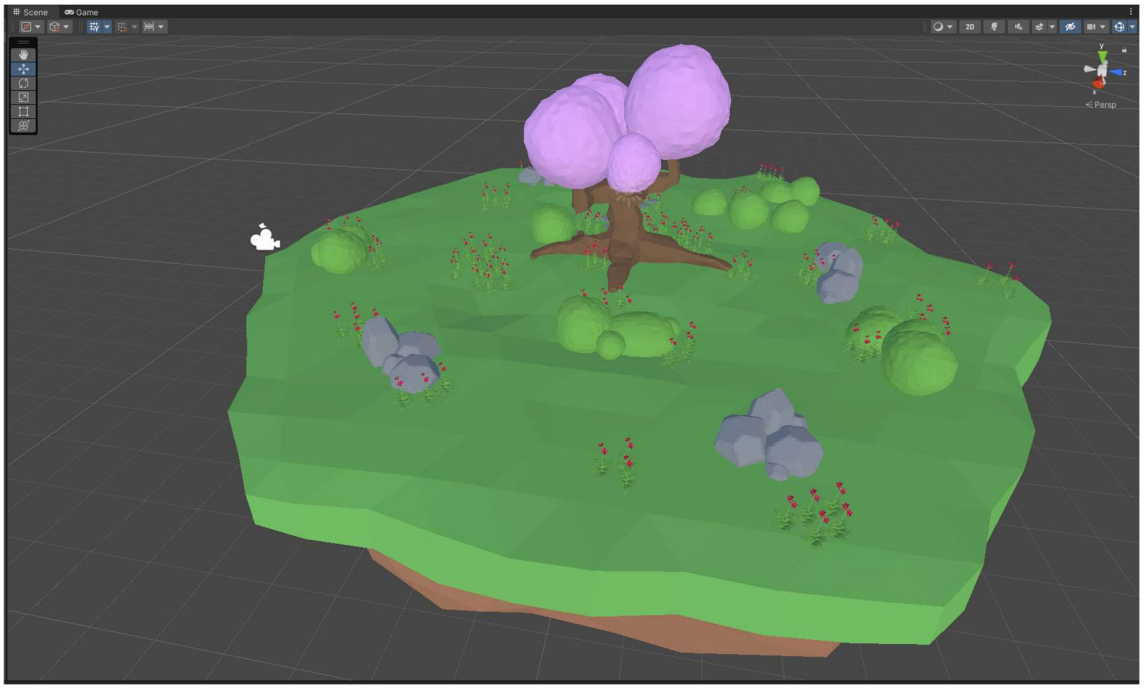
Μεγάλη σημασία για να επιτευχθεί ένα επιθυμητό αποτέλεσμα είναι ο σχεδιασμός του κατάλληλου περιβάλλοντος στο οποίο θα γίνει η εκπαίδευση του πράκτορα, καθώς σε αυτό βρίσκονται τα κύρια ερεθίσματα τα οποία θα προκαλέσουν τις αντίστοιχες συμπεριφορές που έχουν τεθεί ως στόχοι.

Ο στόχος της εκπαίδευσης για την οποία γίνεται η περιγραφή σε αυτή την διατριβή, είναι ο πράκτορας ο οποίος είναι ένα κολίμπρι, να μπορεί να μετακινηθεί μόνος του στο τρισδιάστατο περιβάλλον στο οποίο βρίσκεται, να παρατηρεί που βρίσκονται τα λουλούδια στο περιβάλλον αυτό και να ψάχνει σε αυτά ζωύφια για τροφή.

Στην συνέχεια θα γίνει περιγραφή του περιβάλλοντος που χρησιμοποιήθηκε για την εκπαίδευση αυτή, καθώς και των λόγων για τους οποίους έχει γίνει η συγκεκριμένη επιλογή.

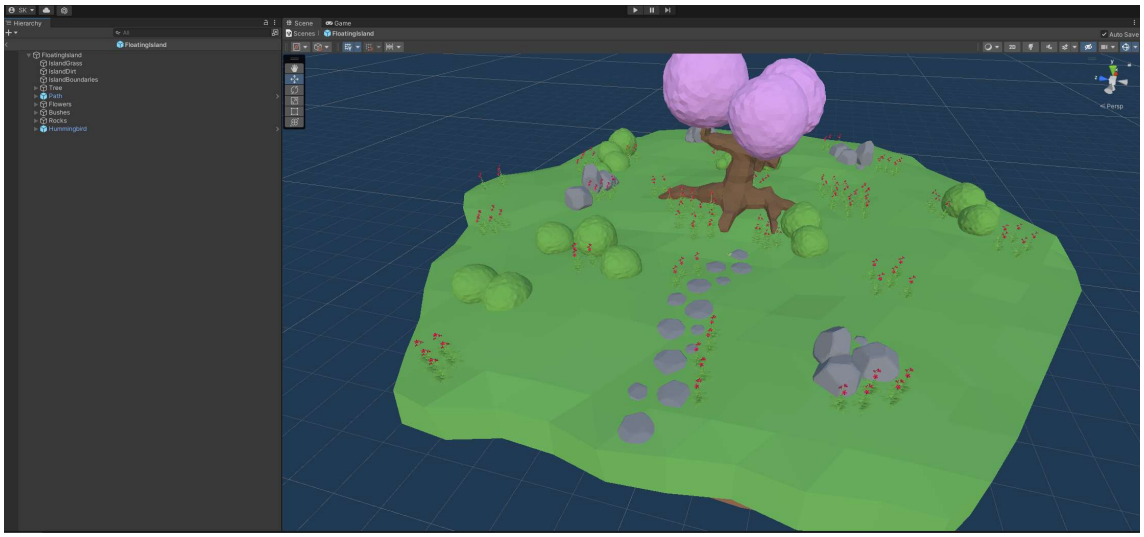
4.1 Περιβάλλον

Το περιβάλλον στο οποίο βρίσκεται πάνω ο πράκτορας είναι ένα ιπτάμενο νησίΕικόνα 6 Ιπτάμενο νησί το οποίο φαίνεται στην Εικόνα 6, πάνω στο οποίο βρίσκονται ο πράκτορας, κάποιες συστάδες από λουλούδια πάνω στα οποία βρίσκονται τα ζωύφια τα οποία είναι και ο τελικός στόχος, καθώς και κάποια αντικείμενα που εμποδίζουν τον πράκτορα από το να πετάξει πιο εύκολα προς αυτόν.



Εικόνα 6 Ιπτάμενο νησί

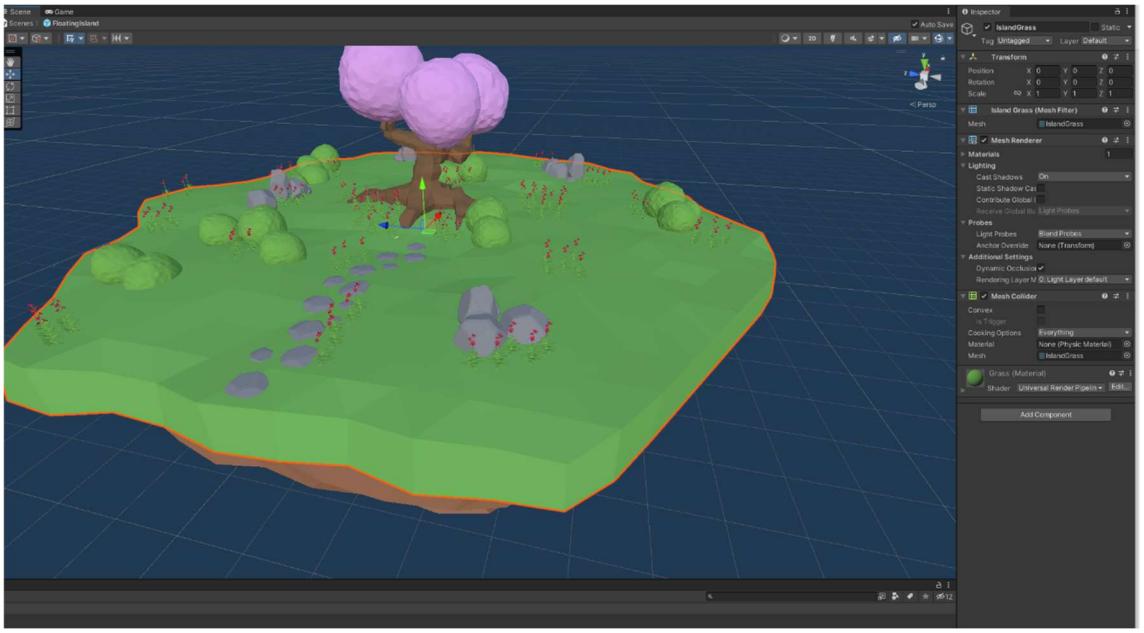
Στην Εικόνα 7 παρατηρούμε ότι αποτελείται από 9 αντικείμενα και είναι ορισμένο ως prefab. Σύμφωνα με το επίσημο εγχειρίδιο που βρίσκεται στη σελίδα της unity, prefab είναι μία λειτουργία αντικειμένου της unity η οποία επιτρέπει να διαμορφώνεται και να αποθηκεύεται ένα αντικείμενο με όλα τα στοιχεία, τις ιδιότητες και τα αντικείμενα παιδιά που έχει ως επαναχρησιμοποιήσιμο [16]. Η συγκεκριμένη λειτουργία είναι σημαντική να επισημανθεί στο σημείο αυτό γιατί θα χρησιμοποιηθεί όπως έχει αναφερθεί σε προηγούμενο για την παράλληλη εκτέλεση συμπεριφορών και μοιράζοντας έτσι τα δεδομένα μεταξύ των πρακτόρων.



Εικόνα 7 Prefab ιπτάμενου νησιού

4.1.1 Γρασίδι

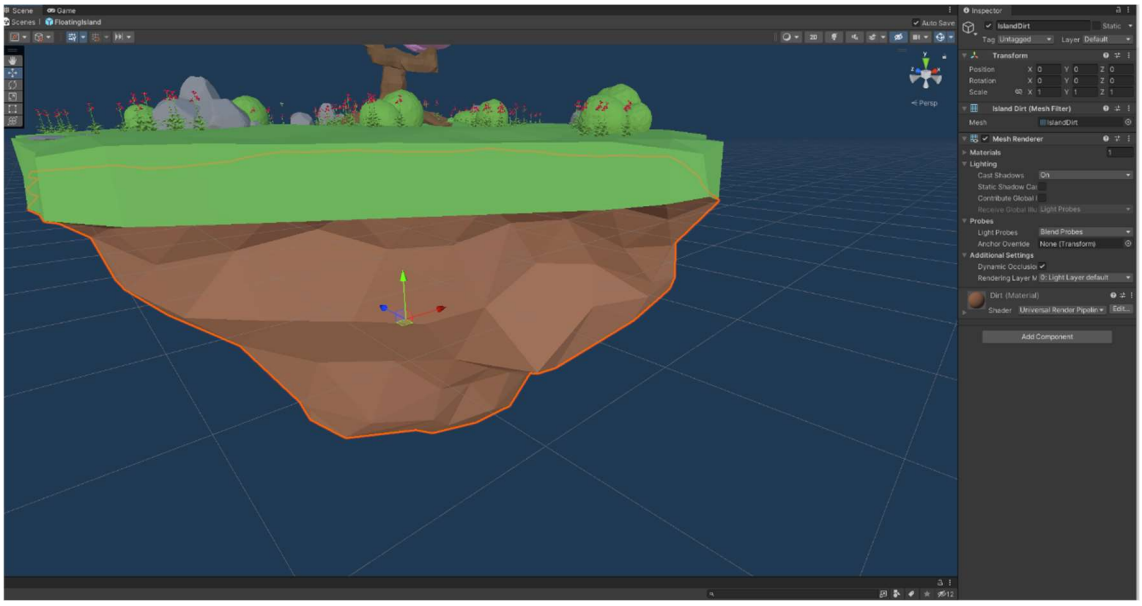
Για το αντικείμενο που περιέχει το γρασίδι του νησιού, όπως φαίνεται στην Εικόνα 8, αξίζει να αναφερθεί ότι έχει ένα Mesh Collider ο οποίος αποτρέπει τον πράκτορα από το μετακινηθεί από κάτω του.



Εικόνα 8 Γρασίδι

4.1.2 Χώμα

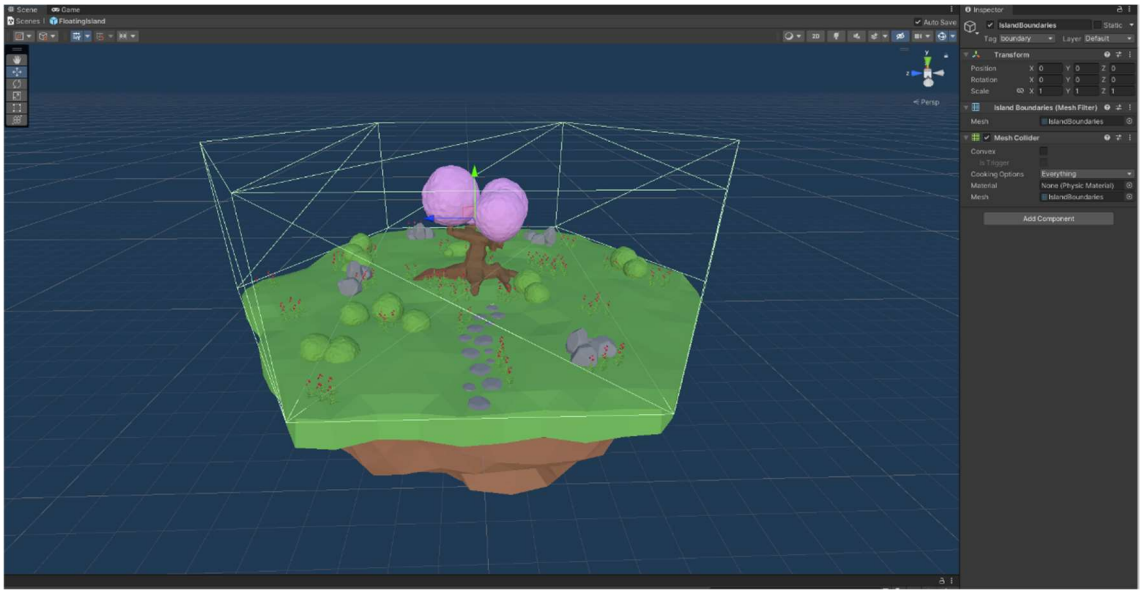
Το αντικείμενο που περιέχει το χώμα κάτω από το γρασίδι και φαίνεται στην Εικόνα 9 βρίσκεται στη σκηνή καθαρά για λόγους εμφάνισης και δεν επηρεάζει την εκπαίδευση με οποιοδήποτε τρόπο.



Εικόνα 9 Χώμα

4.1.3 Όρια νησιού

Για την οριοθέτηση του περιβάλλοντος εκπαίδευσης, όπως φαίνεται στην Εικόνα 10, σημαντικό είναι να δημιουργηθεί ένα collider το οποίο αποτρέπει τον πράκτορα από το να βγει εκτός των ορίων του ιπτάμενου νησιού. Στη συγκεκριμένη υλοποίηση χρειάζεται να προσθέσουμε και ένα tag στο αντικείμενο που περιέχει το collider αυτό, για να είναι εύκολη η αναγνώρισή του μέσα στον κώδικα των scripts που θα αναπτυχθούν. Έτσι στην περίπτωση που ο πράκτορας θα πέφτει πάνω σε αυτό το collider θα λαμβάνει αρνητική επιβράβευση και με αυτό τον τρόπο θα ενισχυθεί η εκπαίδευση.

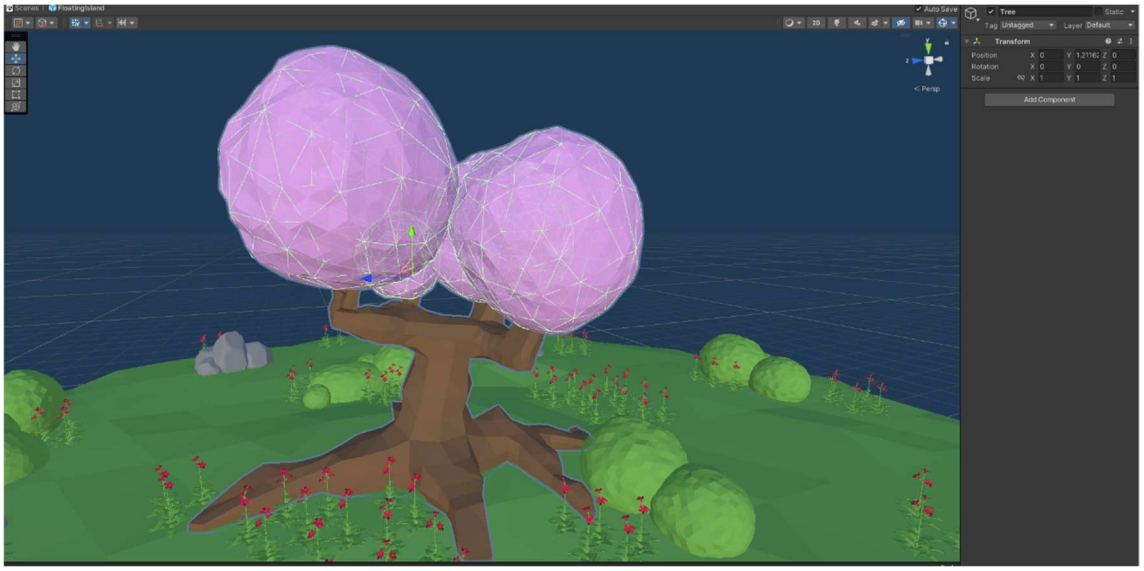


Εικόνα 10 Όρια νησιού

4.1.4 Δέντρο

Το δέντρο το οποίο βρίσκεται στη μέση του νησιού, όπως φαίνεται στην Εικόνα 11, είναι απλά ένα εμπόδιο για την επίτευξη του στόχου του πράκτορα και η τοποθέτησή του έχει σημασία γιατί αυξάνει τη

δυσκολία πλοήγησης του πράκτορα στο τρισδιάστατο περιβάλλον. Σημαντικό είναι να αναφερθεί το ότι για να θεωρείται εμπόδιο για τον πράκτορα το αντικείμενο αυτό και τα παιδιά του, θα πρέπει να έχουν και εκείνα colliders για να γίνεται η αναγνώριση της σύγκρουσης του πράκτορα από τον κώδικα.



Εικόνα 11 Δέντρο

4.1.5 Μονοπάτι

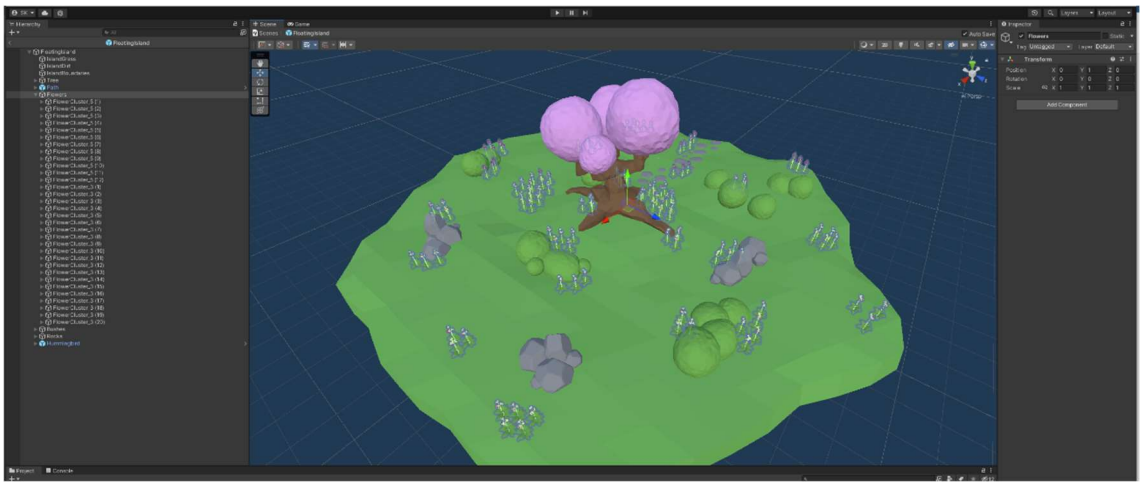
Το αντικείμενο που περιέχει το μονοπάτι και φαίνεται στην Εικόνα 12, βρίσκεται στη σκηνή καθαρά για λόγους εμφάνισης και δεν επηρεάζει την εκπαίδευση με οποιοδήποτε τρόπο.

Εικόνα 12 Μονοπάτι

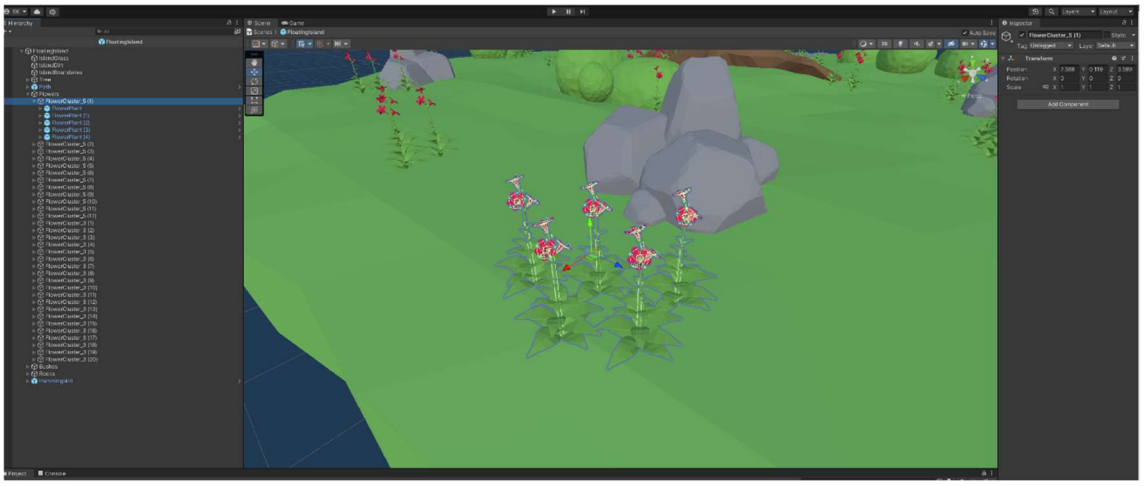
4.1.6 Λουλούδια

Ιδιαίτερη προσοχή χρειάζεται να δοθεί στην ανάλυση αυτού του αντικειμένου, καθώς είναι ένα από τα κύρια συστατικά για να γίνει σωστή υλοποίηση της εκπαίδευσης που έχει τεθεί σαν στόχος. Όπως φαίνεται στην Εικόνα 13, το κύριο αντικείμενο αποτελείται από συστάδες λουλουδιών (Εικόνα 14) τα οποία με τη σειρά τους αποτελούνται από τα αντίστοιχα φυτά. Σε αυτό το σημείο σημαντική λεπτομέρεια

είναι ότι στο κάθε φυτό έχει προστεθεί ένα tag με το όνομα flower-plant για την ευκολότερη ανίχνευση του συγκεκριμένου αντικειμένου μέσω κώδικα.



Εικόνα 13 Λουλούδια



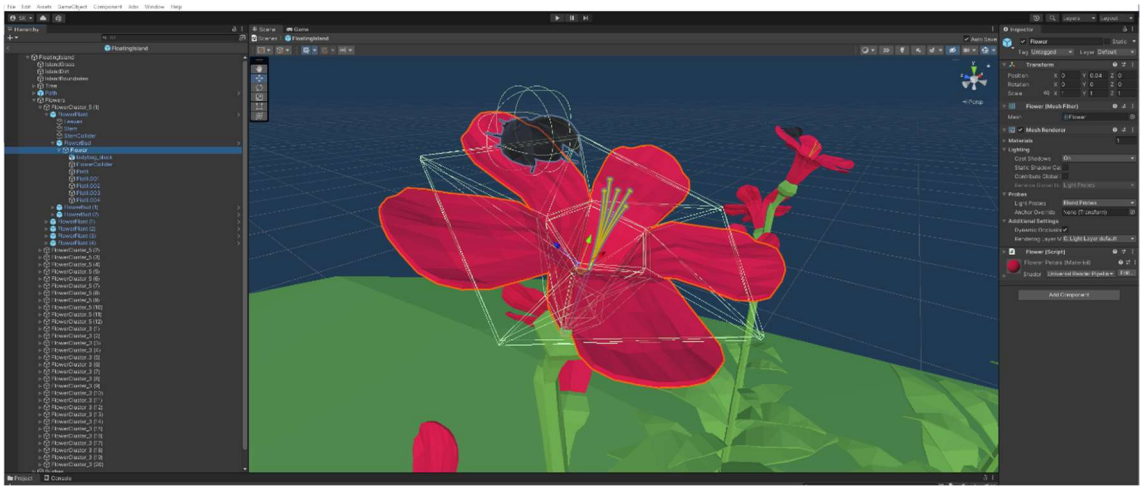
Εικόνα 14 Συστάδα λουλουδιών

Εικόνα 15 Φυτό

Το κάθε φυτό είναι ένα prefab το οποίο αποτελείται από μικρότερα αντικείμενα. Ένα από αυτά που χρήζουν αναφοράς είναι το StemCollider, το οποίο είναι ένα capsule collider που βοηθάει στη συμπεριφορά που θα αποκτήσει ο πράκτορας μετά την εκπαίδευσή του, ώστε να είναι περισσότερο ρεαλιστική, καθώς θα μάθει να περιφέρεται γύρω από το φυτό και όχι να περνάει από μέσα του.

Το σημαντικότερο αντικείμενο όμως που πρέπει να αναφερθεί και να αναλυθεί είναι το λουλούδι (Εικόνα 16) πάνω στο οποίο υπάρχει ένα collider το οποίο θα αναφερθεί στο επόμενο κεφάλαιο, καθώς και το ζωύφιο (ladybug_black) το οποίο είναι και ο τελικός στόχος του πράκτορα.

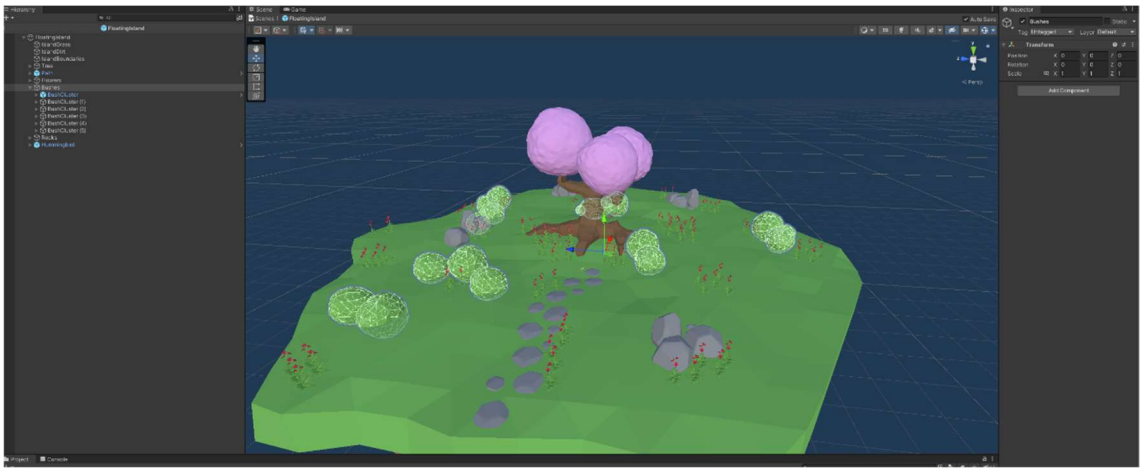
Το ζωύφιο περιέχει ένα capsule collider το οποίο βοηθάει στην αναγνώριση της επίτευξης του στόχου κατά τη διάρκεια της εκπαίδευσης, καθώς και ένα tag για την αναγνώρισή του μέσα στα scripts.



Εικόνα 16 Λουλούδι

4.1.7 Θάμνοι

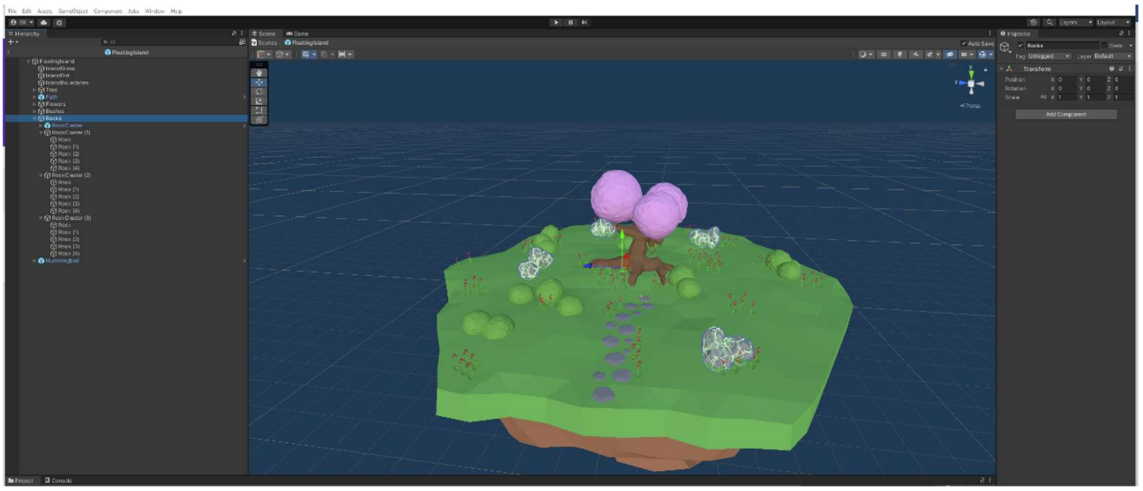
Οι θάμνοι που απεικονίζονται στην Εικόνα 17 είναι αντικείμενα τα οποία αποτελούν εμπόδια για την πλοήγηση του πράκτορα μέσα στο περιβάλλον. Για τον λόγο αυτό, αντίστοιχα και με τα προηγούμενα που έχουν αναφερθεί που είναι τύπου εμπόδια, έχει προστεθεί collider έτσι ώστε να αναγνωρίζεται η σύγκρουση του πράκτορα μέσω του κώδικα κατά τη διάρκεια της εκπαίδευσης.



Εικόνα 17 Θάμνοι

4.1.8 Πέτρες

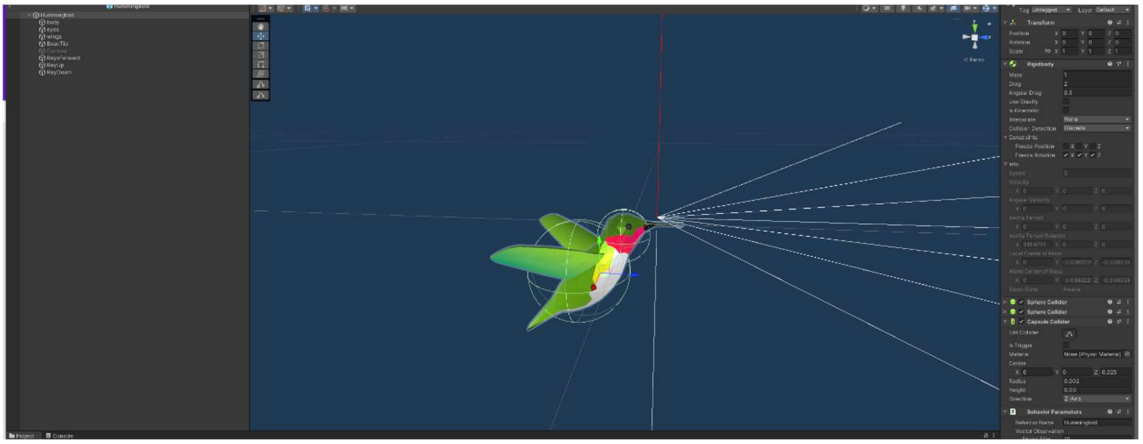
Αντίστοιχα με τα προηγούμενα αντικείμενα και οι πέτρες που φαίνονται στην εικόνα είναι απλά εμπόδια και έχουν colliders για την αναγνώρισή τους.



Εικόνα 18 Πέτρες

4.2 Πράκτορας

Ο πράκτορας, όπως προαναφέρθηκε στην αρχή του κεφαλαίου, είναι ένα κολίμπρι του οποίου ο στόχος είναι να μετακινείται προς τα λουλούδια και να τρέφεται με τα ζωύφια. Όπως φαίνεται στην εικόνα Εικόνα 19, αποτελείται από 8 αντικείμενα - παιδιά. Στο σημείο αυτό θα πρέπει να γίνει ανάλυση για τα σημαντικότερα.



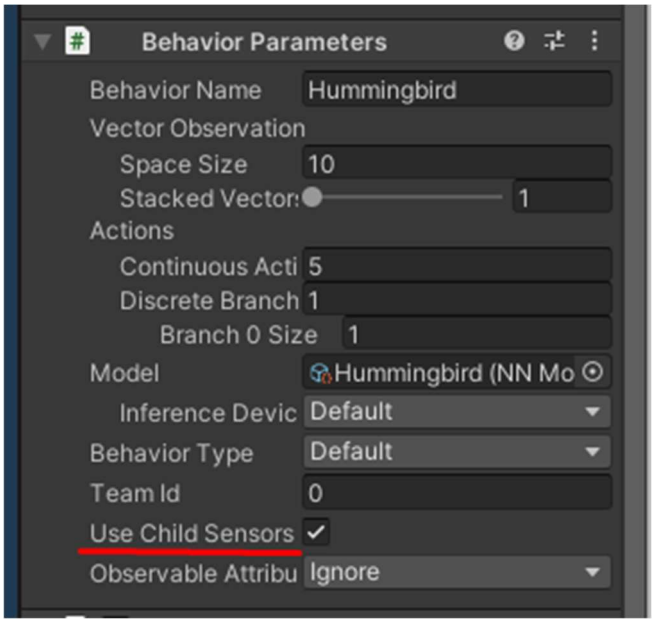
Εικόνα 19 Πράκτορας

4.2.1 Colliders

Ο πράκτορας έχει τρία colliders τα οποία φαίνονται στην Εικόνα 19. Τα δύο πρώτα βρίσκονται στο σώμα και στο κεφάλι, έτσι ώστε να γίνεται η αναγνώριση συγκρούσεων με το περιβάλλον και το τρίτο είναι το collider που βρίσκεται στο ράμφος, το οποίο συμβάλει στην θετική επιβράβευση του πράκτορα, όταν εκείνο συγκρουστεί με το collider που υπάρχει στο ζωύφιο.

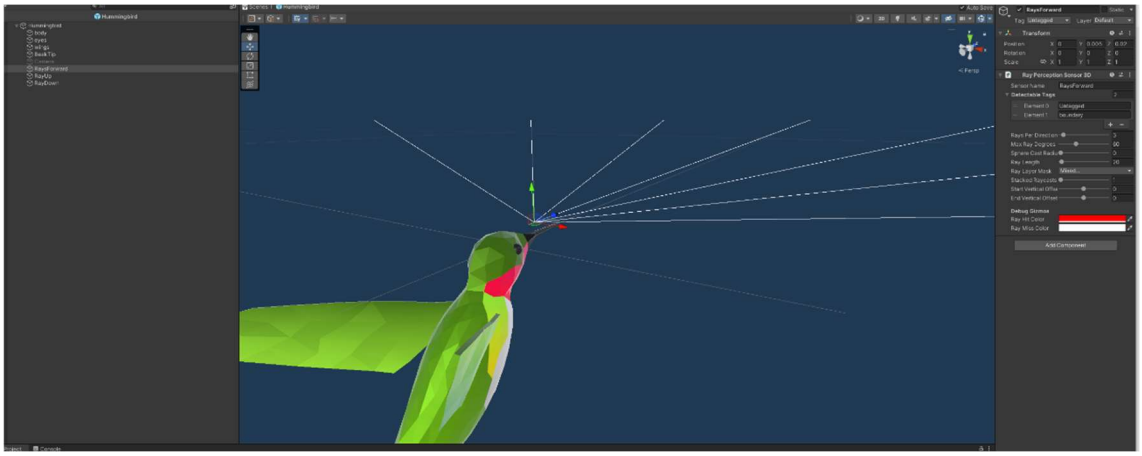
4.2.2 Αισθητήρες

Τρεις είναι οι αισθητήρες οι οποίοι συμβάλλουν στην συλλογή παρατηρήσεων του πράκτορα όσον αφορά το περιβάλλον γύρω του. Ο σκοπός των αισθητήρων αυτών είναι ουσιαστικά να ρίχνουν ακτίνες προς κάποια κατεύθυνση και να αναφέρουν πίσω στον πράκτορα ότι αυτές οι ακτίνες έχουν πετύχει κάτι το οποίο είναι στέρεο. Για να λαμβάνονται υπόψιν οι παρατηρήσεις από αυτούς τους αισθητήρες, πρέπει να έχει επιλεχθεί στο script με το όνομα Behavior Parameters, το οποίο προκύπτει από την κλάση του Agent, η επιλογή use child sensors (Εικόνα 20).



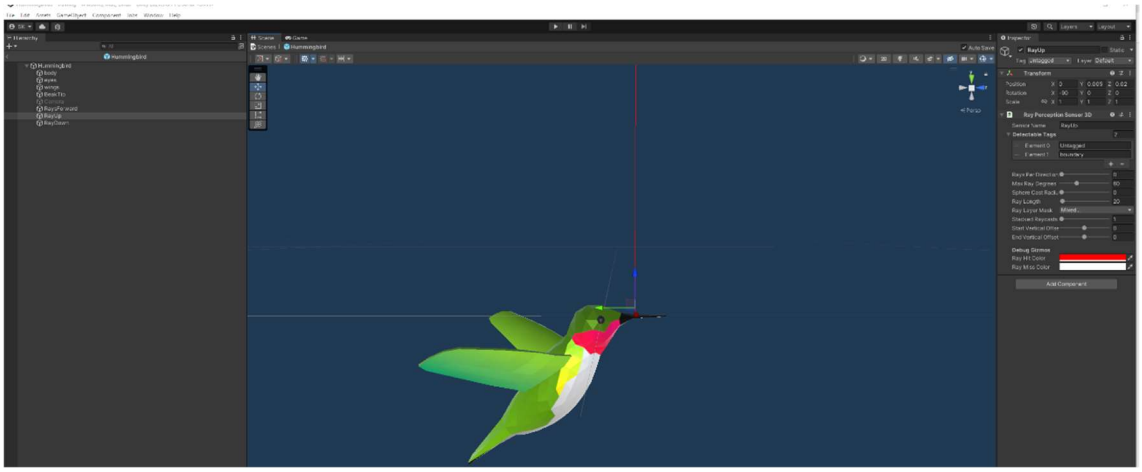
Εικόνα 20 Behavior Parameters

Ο ένας βρίσκεται στο πάνω μέρος του ράμφους και κοιτάζει ευθεία, όπως φαίνεται στην εικόνα Εικόνα 21, του οποίου σκοπός είναι να βλέπει πράγματα τα οποία δεν έχουν κάποιο tag, όπως τα εμπόδια που προαναφέρθηκαν ή εάν χτυπάει στο collider που έχει εκχωρηθεί το tag boundary, δηλαδή τα όρια του περιβάλλοντος.



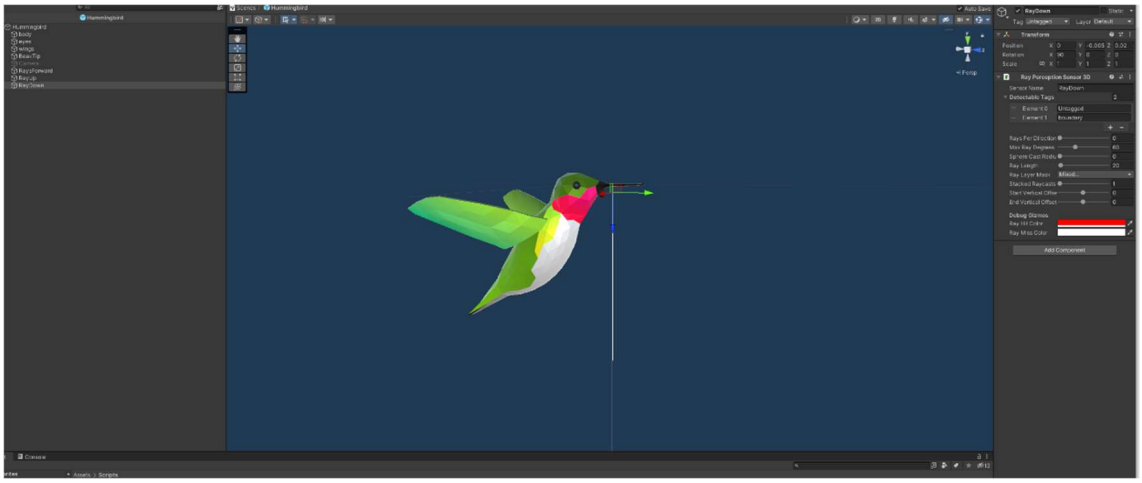
Εικόνα 21 Ακτίνες μπροστά

Ακόμα ένας βρίσκεται πάλι στο πάνω μέρος του ράμφους και κοιτάει προς τα πάνω, όπως φαίνεται στην Εικόνα 22, με αντίστοιχη λειτουργικότητα με την προηγούμενη.Εικόνα 22



Εικόνα 22 Ακτίνα πάνω

Ο τελευταίος βρίσκεται στο κάτω μέρος του ράμφους έτσι ώστε να μην συμπίπτει με τον ίδιο τον πράκτορα και κοιτάζει προς τα κάτω, όπως φαίνεται στην Εικόνα 23, ενώ η λειτουργικότητα είναι αντίστοιχη με τις προηγούμενες.



Εικόνα 23 Ακτίνα κάτω

5 Υλοποίηση

Σε αυτό το κεφάλαιο θα αναλυθεί ο κώδικας ο οποίος χρειάζεται για να τρέξει η εκπαίδευση, καθώς και η μεθοδολογία που χρησιμοποιήθηκε.

Το σενάριο εκπαίδευσης αποτελείται από την ακόλουθη διαδικασία. Ο πράκτορας εμφανίζεται σε τυχαίο σημείο εντός των ορίων του περιβάλλοντος εκπαίδευσης και κάνει προσπάθεια για να μετακινηθεί και να φτάσει στα λουλούδια για να βρει ζωύφια για να τραφεί. Σε περίπτωση που ο πράκτορας συγκρουστεί με τα όρια του περιβάλλοντος δέχεται αρνητική επιβράβευση, ενώ αντίστοιχα όταν το ράμφος του συγκρουστεί με το ζωύφιο, τότε ξεκινάει να τρέφεται και μόλις το καταναλώσει, το αντίστοιχο λουλούδι αλλάζει χρώμα. Στο τέλος του κάθε επεισοδίου το περιβάλλον έρχεται στην αρχική του κατάσταση και ο πράκτορας εμφανίζεται σε διαφορετικό σημείο. Στο τέλος όλων των επεισοδίων παράγεται το νευρωνικό δίκτυο το οποίο μπορούμε να χρησιμοποιήσουμε ως εγκέφαλο του πράκτορα και να παρατηρήσουμε τη συμπεριφορά για την οποία έχει εκπαιδευτεί.

5.1 Flower

Το script αυτό συμπεριλαμβάνεται στο αντικείμενο που αντικατοπτρίζει το λουλούδι και περιέχει δύο βασικές συναρτήσεις οι οποίες αξίζει να αναφερθούν και οι οποίες αποτελούν μέρος της εκπαίδευσης του νευρωνικού δικτύου.

Η μία από αυτές είναι η συνάρτηση Feed η οποία φαίνεται στην Εικόνα 24 και καλείται όταν το ράμφος του πράκτορα συγκρούεται με το ζωύφιο και επιστρέφει την ποσότητα του ζωυφίου που έχει καταναλωθεί. Σε περίπτωση που καταναλωθεί ολόκληρο, τότε το λουλούδι αλλάζει χρώμα και το ζωύφιο εξαφανίζεται.

```
1 reference
public float Feed(float amount)
{
    // Track how much bug was successfully taken (cannot take more than is available)
    float bugTaken = Mathf.Clamp(amount, 0f, BugAmount);

    // Subtract the bug
    BugAmount -= amount;

    if(BugAmount <= 0)
    {
        // No bug remaining
        BugAmount = 0;

        //Disable the flower and bug colliders

        flowerCollider.gameObject.SetActive(false);
        ladyBugCollider.gameObject.SetActive(false);

        // Change the flower color to indicate that it is empty
        flowerMaterial.SetColor("_BaseColor", emptyFlowerColor);
    }

    //Return the amount of bug that was taken
    return bugTaken;
}
/// <summary>
```

Εικόνα 24 Συνάρτηση Feed

Η επόμενη είναι η συνάρτηση ResetFlower η οποία φαίνεται στην Εικόνα 25 και καλείται στην αρχή κάθε επεισοδίου. Ουσιαστικά, επιστρέφει πίσω το ζωύφιο σε περίπτωση που έχει καταναλωθεί και γυρίζει το χρώμα του λουλουδιού πίσω στο αρχικό.

```
/// <summary>
/// Resets the flower
/// </summary>
1 reference
public void ResetFlower()
{
    //Refill the bug
    BugAmount = 1f;

    // Enable the flower and bug colliders
    flowerCollider.gameObject.SetActive(true);
    ladyBugCollider.gameObject.SetActive(true);

    //Change the flower color to indicate that it is full
    flowerMaterial.SetColor("_BaseColor", fullFlowerColor);
}
```

Εικόνα 25 Συνάρτηση ResetFlower

5.2 FlowerArea

Το συγκεκριμένο script χρησιμεύει για τη διαχείριση μιας συστάδας λουλουδιών. Στο συγκεκριμένο σημείο δύο είναι οι πιο σημαντικές συναρτήσεις οι οποίες αξίζουν να αναφερθούν.

Η πρώτη είναι η ResetFlowers η οποία φαίνεται στην Εικόνα 26 και ουσιαστικά καλείται στην αρχή του κάθε επεισοδίου, η οποία επιστρέφει όλα τα λουλούδια και τα ζωύφια στην αρχική τους κατάσταση.

```
public void ResetFlowers()
{
    // Rotate each flower plant around the Y axis and subtly around X and Z
    foreach (GameObject flowerPlant in flowerPlants)
    {
        float xRotation = UnityEngine.Random.Range(-5f, 5f);
        float yRotation = UnityEngine.Random.Range(-180f, 180f);
        float zRotation = UnityEngine.Random.Range(-5f, 5f);
        flowerPlant.transform.localRotation = Quaternion.Euler(xRotation, yRotation, zRotation);
    }

    //Reset each flower
    foreach (Flower flower in Flowers)
    {
        flower.ResetFlower();
    }
}
```

Εικόνα 26 Συνάρτηση ResetFlowers

Η δεύτερη είναι η FindChildFlowers, η οποία φαίνεται στην Εικόνα 27.Ουσιαστικά, η λειτουργία της είναι να ψάχνει και να βρίσκει αναδρομικά όλα τα λουλούδια και τα φυτά που είναι παιδιά ενός αντικείμενου γονέα.

```
private void FindChildFlowers(Transform parent)
{
    for (int i = 0; i < parent.childCount; i++)
    {
        Transform child = parent.GetChild(i);

        if (child.CompareTag("flower_plant"))
        {
            // Found a flower plant, add it to the flowerPlants list
            flowerPlants.Add(child.gameObject);

            // Look for flowers within the flower plant
            FindChildFlowers(child);
        }
        else
        {
            // Not a flower plant, look for a Flower component
            Flower flower = child.GetComponent<Flower>();
            if (flower != null)
            {
                //Found a flower, add it to the Flowers list
                Flowers.Add(flower);

                // Add the bug collider to the lookup dictionary
                bugFlowerDictionary.Add(flower.ladyBugCollider, flower);

                // Note: there are no flowers that are children of other flowers
            }
            else
            {
                // Flower component not found, so check children
                FindChildFlowers(child);
            }
        }
    }
}
```

Εικόνα 27 Συνάρτηση FindChildFlowers

5.3 HummingBirdAgent

Σε αυτό το script βρίσκεται η κύρια λειτουργεία του πράκτορα. Κληρονομεί τα χαρακτηριστικά της κύριας κλάσης Agent του πακέτου ML-Agents της Unity, η οποία ουσιαστικά είναι μία κλάση η οποία παίρνει αποφάσεις χρησιμοποιώντας νευρωνικά δίκτυα. Στην αρχή του κάθε επεισοδίου γίνεται η αρχικοποίηση του πράκτορα και καλείται η OnEpisodeBegin (Εικόνα 28) στην οποία γίνεται αρχικοποίηση των λουλουδιών στο αρχικό τους στάδιο, όπως προαναφέρθηκε. Επίσης ορίζει ότι 50% των φορών που ξεκινάει ένα καινούριο επεισόδιο ο πράκτορας θα εμφανίζεται στην αρχή μπροστά από ένα λουλούδι. Στη συνέχεια θα προσπαθεί να μεταφερθεί στο κοντινότερο ασφαλές σημείο, καλώντας τη συνάρτηση MoveToSafeRandomPosition (Εικόνα 29). Στη συνάρτηση αυτή και στην περίπτωση που ο πράκτορας έχει εμφανιστεί στην αρχή του επεισοδίου μπροστά από ένα λουλούδι, τότε διαλέγει ένα τυχαίο λουλούδι και στρέφει το ράμφος του προς το λουλούδι διαφορετικά, διαλέγει μία τελείως τυχαία μετακίνηση στον τρισδιάστατο άξονα. Ύστερα καλείται η UpdateNearestFlower η οποία επιστρέφει το κοντινότερο ενεργό λουλούδι στον πράκτορα, το οποίο έχει ζωύφια για να τραφεί.

```
public override void OnEpisodeBegin()
{
    if (trainingMode)
    {
        // Only reset flowers in training when there is one agent per area
        flowerArea.ResetFlowers();
    }

    // Reset bug obtained
    BugObtained = 0f;

    // Zero out velocities so that movement stops before a new episode begins
    rigidbody.velocity = Vector3.zero;
    rigidbody.angularVelocity = Vector3.zero;

    // Default to spawning in front of a flower
    bool inFrontOfFlower = true;
    if (trainingMode)
    {
        // Spawn in front of flower 50% of the time during training
        inFrontOfFlower = UnityEngine.Random.value > .5f;
    }

    // Move the agent to a new random position
    MoveToSafeRandomPosition(inFrontOfFlower);

    // Recalculate the nearest flower now that the agent has moved
    UpdateNearestFlower();
}
```

Εικόνα 28 Συνάρτηση OnEpisodeBegin

```
private void MoveToSafeRandomPosition(bool inFrontOfFlower)
{
    bool safePositionFound = false;
    int attemptsRemaining = 100; // Prevent an infinite loop
    Vector3 potentialPosition = Vector3.zero;
    Quaternion potentialRotation = new Quaternion();

    // Loop until a safe position is found or we run out of attempts
    while(!safePositionFound && attemptsRemaining > 0)
    {
        attemptsRemaining--;
        if (inFrontOfFlower)
        {
            // Pick a random flower
            Flower randomFlower = flowerArea.Flowers[UnityEngine.Random.Range(0, flowerArea.Flowers.Count)];

            // Position 10 to 20 cm in front of the flower
            float distanceFromFlower = UnityEngine.Random.Range(.1f, .2f);
            potentialPosition = randomFlower.transform.position + randomFlower.FlowerUpVector * distanceFromFlower;

            // Point beak at flower (bird's head is center of transform)
            Vector3 toFlower = randomFlower.FlowerCenterPosition - potentialPosition;
            potentialRotation = Quaternion.LookRotation(toFlower, Vector3.up);
        }
        else
        {
            //Pick a random height from the ground
            float height = UnityEngine.Random.Range(1.2f, 2.5f);

            // Pick a random radius from the center of the area
            float radius = UnityEngine.Random.Range(2f, 7f);

            //Pick a random direction rotated around the y axis
            Quaternion direction = Quaternion.Euler(0f, UnityEngine.Random.Range(-180f, 180f), 0f);

            // Combine height, radius, and direction to pick a potential position
            potentialPosition = flowerArea.transform.position + Vector3.up * height + direction * Vector3.forward * radius;

            // Choose and set random starting pitch and yaw
            float pitch = UnityEngine.Random.Range(-60f, 60f);
            float yaw = UnityEngine.Random.Range(-180f, 180f);
            potentialRotation = Quaternion.Euler(pitch, yaw, 0f);
        }

        // Check to see if the agent will collide with anything
        Collider[] colliders = Physics.OverlapSphere(potentialPosition, 0.05f);

        // Safe position has been found if no colliders are overlapped
        safePositionFound = colliders.Length == 0;
    }

    Debug.Assert(safePositionFound, "Could not find a safe position to spawn");

    // Set the position and rotation
    transform.position = potentialPosition;
    transform.rotation = potentialRotation;
}
```

Εικόνα 29 Συνάρτηση MoveToSafeRandomPosition

```
private void UpdateNearestFlower()
{
    foreach (Flower flower in flowerArea.Flowers)
    {
        if (nearestFlower == null && flower.HasBug)
        {
            // No current nearest flower and this flower has bug, so set to this flower
            nearestFlower = flower;
        }
        else if (flower.HasBug)
        {
            // Calculate distance to this flower and distance to the current nearest flower
            float distanceToFlower = Vector3.Distance(flower.transform.position, beakTip.position);
            float distanceToCurrentNearestFlower = Vector3.Distance(nearestFlower.transform.position, beakTip.position);

            // If current nearest flower is empty OR this flower is closer, update the nearest flower
            if (!nearestFlower.HasBug || distanceToFlower < distanceToCurrentNearestFlower)
            {
                nearestFlower = flower;
            }
        }
    }
}
```

Εικόνα 30 Συνάρτηση UpdateNearestFlower

Η συνάρτηση OnActionReceived (Εικόνα 31) καλείται όταν το νευρωνικό δίκτυο πάρει οποιαδήποτε απόφαση.

```
public override void OnActionReceived(ActionBuffers actions)
{
    // Don't take actions if frozen
    if (frozen) return;

    // Calculate movement vector
    Vector3 move = new Vector3(actions.ContinuousActions[0], actions.ContinuousActions[1], actions.ContinuousActions[2]);

    // Add force in the direction of the move vector
    rigidbody.AddForce(move * moveForce);

    // Get the current rotation
    Vector3 rotationVector = transform.rotation.eulerAngles;

    // Calculate pitch and yaw rotation
    float pitchChange = actions.ContinuousActions[3];
    float yawChange = actions.ContinuousActions[4];

    // Calculate smooth rotation changes
    smoothPitchChange = Mathf.MoveTowards(smoothPitchChange, pitchChange, 2f * Time.fixedDeltaTime);
    smoothYawChange = Mathf.MoveTowards(smoothYawChange, yawChange, 2f * Time.fixedDeltaTime);

    // Calculate new pitch and yaw based on smoothed values
    // Clamp pitch to avoid flipping upside down
    float pitch = rotationVector.x * smoothPitchChange * Time.fixedDeltaTime * pitchSpeed;
    if (pitch > 100f) pitch -= 360f;
    pitch = Mathf.Clamp(pitch, -MaxPitchAngle, MaxPitchAngle);

    float yaw = rotationVector.y + smoothPitchChange * Time.fixedDeltaTime * yawSpeed;

    // Apply the new rotation
    transform.rotation = Quaternion.Euler(pitch, yaw, 0f);
}
```

Εικόνα 31 Συνάρτηση OnActionReceived

Η συνάρτηση CollectObservations (Εικόνα 32) είναι υπεύθυνη για τη συλλογή διανυσματικών παρατηρήσεων, οι οποίες περιλαμβάνουν τις μεταβλητές που απαιτούνται για να λάβει ο πράκτορας την καλύτερη απόφαση.

```
public override void CollectObservations(VectorSensor sensor)
{
    // If nearestFlower is null, observe an empty array and return early
    if (nearestFlower == null)
    {
        sensor.AddObservation(new float[10]);
        return;
    }

    // Observe the agent's local rotation (4 observations)
    sensor.AddObservation(transform.localRotation.normalized);

    // Get a vector from the beak tip to the nearest flower
    Vector3 toFlower = nearestFlower.FlowerCenterPosition - beakTip.position;

    // Observe a normalized vector pointing to the nearest flower (3 observations)
    sensor.AddObservation(toFlower.normalized);

    // Observe a dot product that indicates where the beak tip is in front of the flower (1 observation)
    // (+1 means that the beak tip is directly in front of the flower, -1 means directly behind)
    sensor.AddObservation(Vector3.Dot(toFlower.normalized, -nearestFlower.FlowerUpVector.normalized));

    // Observe a dot product that indicates whether the beak is pointing toward the flower (1 observation)
    // (+1 means that the beak is pointing directly at the flower , -1 means directly away
    sensor.AddObservation(Vector3.Dot(beakTip.forward.normalized, -nearestFlower.FlowerUpVector.normalized));

    // Observe the relative distance from the beak tip to the flower (1 observation)
    sensor.AddObservation(toFlower.magnitude / FlowerArea.AreaDiameter);

    // 10 total observations
}
```

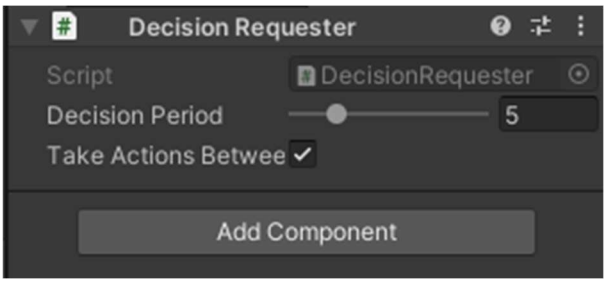
Εικόνα 32 Συνάρτηση CollectObservations

Τελευταία συνάρτηση που πρέπει να αναφερθεί είναι η TriggerEnterOrStay (Εικόνα 33), η οποία καλείται όταν το collider του πράκτορα συγκρουστεί με κάποιο άλλο collider. Εκεί γίνεται έλεγχος εάν η σύγκρουση έχει γίνει με το ζωύφιο και δίνει αντίστοιχα θετική επιβράβευση.

```
private void TriggerEnterOrStay(Collider collider)
{
    // Check if agent is colliding with bug
    if (collider.CompareTag("lady_bug"))
    {
        Vector3 closestPointToBeakTip = collider.ClosestPoint(beakTip.position);

        // Check if the closest collision point is close to the beak tip
        // Note: a collision with anything but the beak tip should not count
        if (Vector3.Distance(beakTip.position, closestPointToBeakTip) < BeakTipRadius)
        {
            // Look up the flower for this bug collider
            Flower flower = flowerArea.GetFlowerFromBug(collider);

            // Attempt to take .01 bug
            // Note: this is per fixed timestep, meaning it happens every .02 seconds, or 50 times per second
            float bugReceived = flower.Feed(.01f);

            // Keep track of bug obtained
            BugObtained += bugReceived;

            if (trainingMode)
            {
                // Calculate reward for getting bug
                float bonus = 0.2f * Mathf.Clamp01(Vector3.Dot(transform.forward.normalized, -nearestFlower.FlowerUpVector.normalized));
                AddReward(.01f + bonus);
            }

            // If flower is empty, update the nearest flower
            if (!flower.HasBug)
            {
                UpdateNearestFlower();
            }
        }
    }
}
```

Εικόνα 33 Συνάρτηση TriggerEnterOrStay

Για την εκπαίδευση χρειάζεται επιπλέον να είναι προσκολλημένα στον πράκτορα και 2 ακόμα scripts, το Behavior Parameters το οποίο έχει προαναφερθεί (Εικόνα 20) και το Decision Requester τα οποία προστίθενται αυτόματα, όταν το HummingBirdAgent script τοποθετείται στον πράκτορα, καθώς είναι προαπαιτούμενα της κλάσης Agent από την οποία κληρονομεί η HummingBirdAgent.



Στο Behavior Parameters ορίζονται οι διανυσματικές παρατηρήσεις στο αντίστοιχο σημείο κάτω από το Vector observations space size (Εικόνα 20), ο αριθμός των οποίων φαίνεται στη συνάρτηση CollectObservations.

Στο σημείο Vector actions, χρησιμοποιείται η επιλογή continues, το οποίο σημαίνει ότι το νευρωνικό δίκτυο παίρνει αποφάσεις, με βάση το εάν ο πράκτορας επιλέγει έναν αριθμό μεταξύ του μείον ένα και του ένα. Ο αριθμός του space size είναι 5, γιατί πέντε είναι οι διαφορετικές αποφάσεις που παίρνει ο πράκτορας. Οι αποφάσεις αυτές μεταφράζονται σε διαφορετικές κινήσεις μέσα στο περιβάλλον. Η περίπτωση αυτή αναλύεται ως εξής: 3 είναι για την κίνησή του στους 3 άξονες και 2 για την περιστροφή του. Τέλος, στο σημείο model τοποθετείται το εκπαιδευμένο νευρωνικό δίκτυο.

6 Εκτέλεση και αποτελέσματα

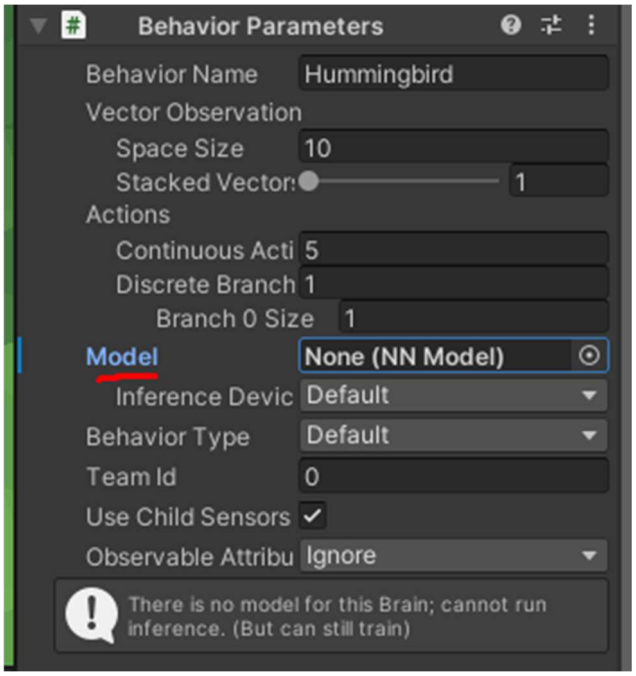
Στο κεφάλαιο αυτό γίνεται αναφορά των βημάτων που χρειάζονται για να γίνει η εκπαίδευση του πράκτορα στο περιβάλλον της Unity, συμπεριλαμβανομένων και των ρυθμίσεων και των βημάτων που πρέπει να εκτελεστούν σε περιβάλλον Python.

6.1 Εκτέλεση

Όπως προαναφέρθηκε στο κεφάλαιο 3, για την εκπαίδευση του πράκτορα χρειάζεται να γίνει διασύνδεση του Unity Editor με μία εφαρμογή σε Python η οποία χρησιμοποιεί τη βιβλιοθήκη torch για την εκπαίδευση του νευρωνικού δικτύου. Στη συγκεκριμένη ανάλυση, για το περιβάλλον της python χρησιμοποιήθηκε το εργαλείο Pycharm της εταιρείας Jetbrains.

Βήματα εκτέλεσης:

- Εισαγωγή βιβλιοθηκών και πακέτου ml-agents, όπως έχει αναφερθεί στο κεφάλαιο 3.
- Εισαγωγή στο περιβάλλον της python του αρχείου trainer_config.yaml το οποίο περιέχει όλες τις ρυθμίσεις εκπαίδευσης.
- Προετοιμασία της σκηνής στον Unity Editor. Η θέση του μοντέλου εκπαίδευσης του πράκτορα θα πρέπει να είναι κενή για να γίνει μία εκπαίδευση από την αρχή (Εικόνα 34).
- Εκτέλεση της εντολής mlagents-learn στη γραμμή εντολών της python, προσθέτοντας το μονοπάτι που βρίσκεται το Yaml αρχείο, ακολουθούμενο από την εντολή –run-id και το όνομα του φακέλου που είναι επιθυμητό να αποθηκευτούν τα αποτελέσματα εκπαίδευσης. Στο συγκεκριμένο παράδειγμα η εντολή εκτέλεσης είναι mlagents-learn .\trainer_config.yaml --run-id hb_03. Εφόσον όλα τα βήματα έχουν εκτελεστεί επιτυχώς, η εικόνα που αντικρίζεται στη γραμμή εντολών της python θα πρέπει να είναι όπως φαίνεται στην Εικόνα 35.
- Αμέσως μετά θα πρέπει να πατηθεί το πλήκτρο Play στον Unity Editor για να ξεκινήσει η εκπαίδευση, οπότε στη γραμμή εντολών της python θα αρχίσουν να φαίνονται πληροφορίες για την εκπαίδευση, όπως φαίνονται στην Εικόνα 36. Κατά τη διάρκεια της εκπαίδευσης φαίνονται ακόμα πληροφορίες για κάθε επεισόδιο εκπαίδευσης, όπως η επιβράβευση και η τυπική απόκλιση, τα βήματα και ο χρόνος που έχουν παρέλθει από την αρχή της εκπαίδευσης, όπως φαίνεται στην Εικόνα 37. Στον editor της Unity κατά τη διάρκεια της εκπαίδευσης επίσης μπορεί να παρατηρηθούν οι κινήσεις του πράκτορα σε κάθε επεισόδιο, καθώς και η συλλογή παρατηρήσεων όταν είναι ενεργά τα gizmos στον Editor (Εικόνα 38).
- Στο τέλος της εκπαίδευσης ο Unity Editor βγαίνει από το play mode και το περιβάλλον της python παράγει το νευρωνικό δίκτυο με όνομα HummingBird.onnx και διάφορες άλλες πληροφορίες για την εκπαίδευση (Εικόνα 39).
- Τέλος, για να χρησιμοποιήσουμε το νευρωνικό δίκτυο και να παρατηρηθεί στον Unity Editor η συμπεριφορά του πράκτορα μετά την εκπαίδευση, χρειάζεται να εισαχθεί στο πεδίο Model το νευρωνικό δίκτυο που παράχθηκε και να πατηθεί το κουμπί Play.



Εικόνα 34 Κενό μοντέλο

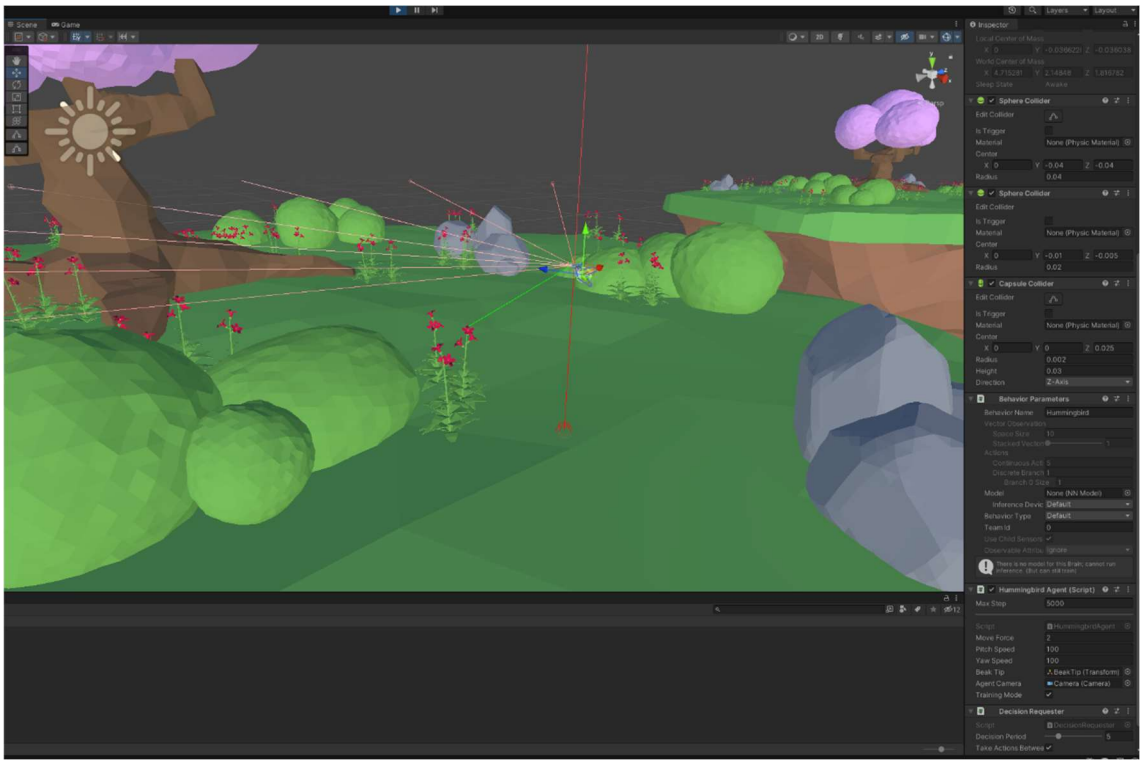
Εικόνα 35 Περιβάλλον Python 1

```
[INFO] Connected to Unity environment with package version 2.0.1 and communication version 1.5.0
[INFO] Connected new brain: Hummingbird?team=0
[INFO] Hyperparameters for behavior name Hummingbird:
        trainer_type:   ppo
        hyperparameters:
          batch_size:   2048
          buffer_size:  20480
          learning_rate:        0.0003
          beta: 0.005
          epsilon:      0.2
          lambd:        0.95
          num_epoch:    3
          learning_rate_schedule:       linear
          beta_schedule:        linear
          epsilon_schedule:     linear
        network_settings:
          normalize:    False
          hidden_units: 256
          num_layers:   2
          vis_encode_type:      simple
          memory:       None
```

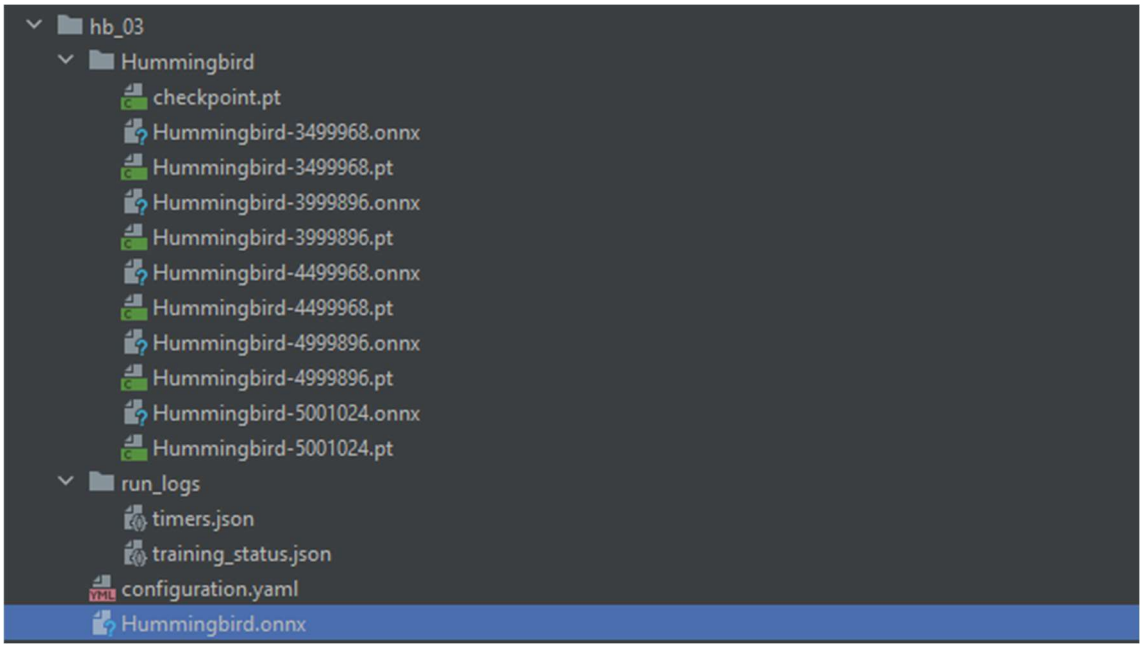
Εικόνα 36 Περιβάλλον Python 2

```
[INFO] Hummingbird. Step: 30000. Time Elapsed: 82.930 s. Mean Reward: -0.312. Std of Reward: 0.827. Training.
[INFO] Hummingbird. Step: 40000. Time Elapsed: 104.569 s. Mean Reward: -0.133. Std of Reward: 0.386. Training.
[INFO] Hummingbird. Step: 50000. Time Elapsed: 134.813 s. Mean Reward: -0.389. Std of Reward: 1.100. Training.
[INFO] Hummingbird. Step: 60000. Time Elapsed: 157.327 s. Mean Reward: -1.312. Std of Reward: 3.473. Training.
[INFO] Hummingbird. Step: 70000. Time Elapsed: 187.928 s. Mean Reward: 0.000. Std of Reward: 0.000. Training.
[INFO] Hummingbird. Step: 80000. Time Elapsed: 209.398 s. Mean Reward: -1.667. Std of Reward: 2.669. Training.
[INFO] Hummingbird. Step: 90000. Time Elapsed: 239.752 s. Mean Reward: -0.222. Std of Reward: 0.629. Training.
[INFO] Hummingbird. Step: 100000. Time Elapsed: 262.033 s. Mean Reward: 0.031. Std of Reward: 1.562. Training.
[INFO] Hummingbird. Step: 110000. Time Elapsed: 292.148 s. Mean Reward: 0.000. Std of Reward: 0.000. Training.
```

Εικόνα 37 Περιβάλλον Python 3



Εικόνα 38 Φάση εκπαίδευσης



Εικόνα 39 Παραγωγή Νευρωνικού Δικτύου

6.2 Αποτελέσματα

Για την ανάλυση των αποτελεσμάτων χρησιμοποιήθηκε η βιβλιοθήκη tensorboard της python, η οποία βοηθάει στην απεικόνιση στο τέλος, αλλά και κατά τη διάρκεια φάσης εκπαίδευσης [17]. Είναι ένα ανοιχτού τύπου λογισμικό το οποίο χρησιμοποιείται για την ανάλυση μετρήσεων πειραμάτων

μηχανικής μάθησης και για την απεικόνισή τους σε γραφήματα, όπως ιστογράμματα βαρών, ανάλυση επιβραβεύσεων και λαθών και άλλα πολλά.

Για την ανάλυση αυτών των αποτελεσμάτων, χρειάζεται να γίνει μετάβαση στο περιβάλλον ανάπτυξης της python και να εισαχθεί στη γραμμή εντολών η εντολή tensorboard --logdir results/hb_03/HummingBird. Αμέσως μετά εμφανίζεται στη γραμμή εντολών ένας σύνδεσμος ο οποίος ανακατευθύνει τον χρήστη να μεταβεί τοπικά στο περιβάλλον που τρέχει το tensorboard σε τοπική πόρτα, όπως φαίνεται στην Εικόνα 40.

```
PS C:\Users\skwst\PycharmProjects\MlAgents>  tensorboard --logdir results/hb_02/HummingBird
TensorFlow installation not found - running with reduced feature set.
Serving TensorBoard on localhost; to expose to the network, use a proxy or pass --bind_all
TensorBoard 2.10.1 at http://localhost:6006/ (Press CTRL+C to quit)
```

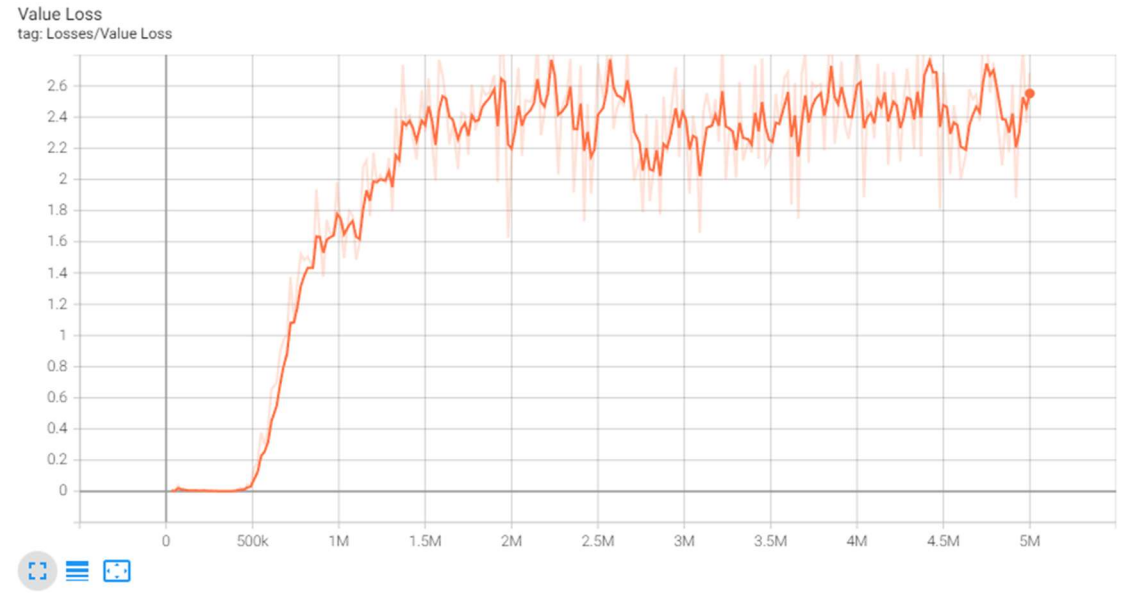
Εικόνα 40 Tensorboard Γραμμή εντολών

Η διάρκεια της εκπαίδευσης κατέληξε στις 3 ώρες και 50 λεπτά και στην Εικόνα 41 μπορεί κανείς να παρατηρήσει την επιβράβευση ανά βήματα εκπαίδευσης του πράκτορα. Αυτό σημαίνει ότι στα 5 εκατομμύρια βήματα τα οποία ακολούθησε ο αλγόριθμος εκπαίδευσης η επιβράβευση είναι περίπου 50, το οποίο για την εφαρμογή που αναλύθηκε σημαίνει ότι σε κάθε επεισόδιο ο πράκτορας συλλέγει περίπου 50 ζωύφια από τα λουλούδια. Παρά το ότι τα αποτελέσματα είναι ικανοποιητικά και το νευρωνικό δίκτυο αποδίδει μια ρεαλιστική συμπεριφορά στον εκπαιδευμένο πράκτορα, αυτή η εκπαίδευση θα μπορούσε να είχε σταματήσει περίπου και στα 2,5 εκατομμύρια βήματα όπου θα είχε και καλύτερο αποτέλεσμα, δηλαδή περίπου στα 60, όπως φαίνεται στην εικόνα. Για να γίνει αυτό πρέπει να παρατηρείται το γράφημα κατά τη διάρκεια της εκπαίδευσης και όταν φτάσει σε ένα ικανοποιητικό αποτέλεσμα να σταματήσει είτε από τον Unity Editor πατώντας το κουμπί έναρξης είτε σταματώντας τη διαδικασία που τρέχει στη γραμμή εντολών της Python.

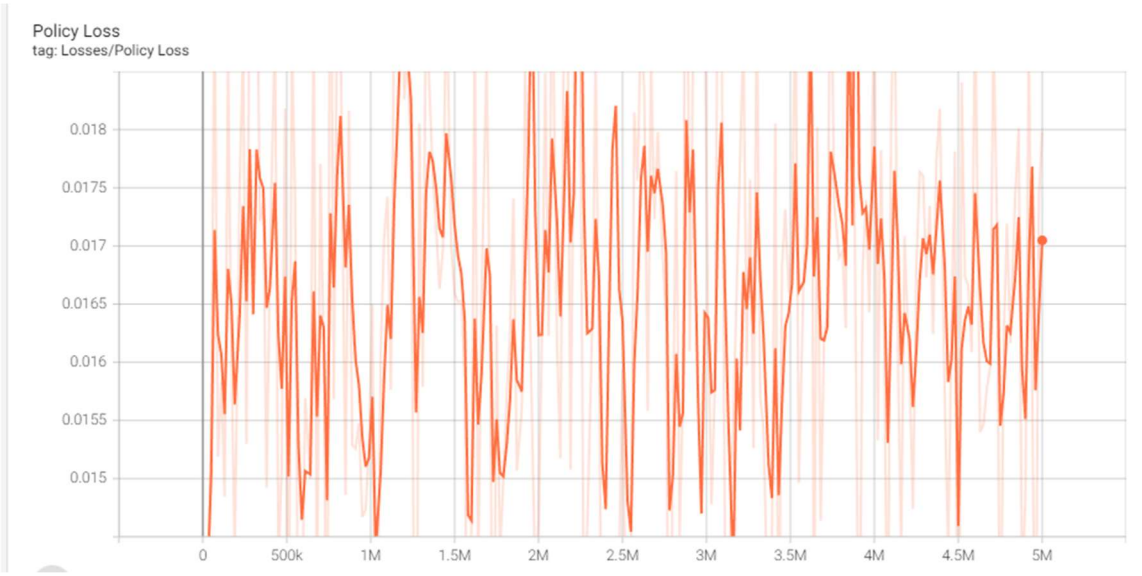
Στην Εικόνα 42 παρατίθεται το γράφημα του value loss ανά βήμα εκπαίδευσης, το οποίο αντικατοπτρίζει την ικανότητα του μοντέλου να προβλέψει την τιμή κάθε κατάστασης. Το αποτέλεσμα είναι ικανοποιητικό, καθώς η μέτρηση του value loss κατά τη διάρκεια της εκπαίδευσης θα πρέπει να μεγαλώνει μέχρι κάποιο σημείο και μετά να μικραίνει έως ότου να σταθεροποιηθεί, κάτι το οποίο έχει συμβεί σε αυτή την εκπαίδευση. Αντίστοιχα στο γράφημα φαίνεται ότι ικανοποιητικό θα ήταν και στα 2,5 εκατομμύρια βήματα, καθώς η διαφορά είναι ελάχιστη σε σχέση με τα 5 εκατομμύρια.

Στην Εικόνα 43 παρατηρείται και το γράφημα του policy loss ανά βήμα κατά τη διάρκεια εκπαίδευσης.

Εικόνα 41 Cumulative Reward



Εικόνα 42 Value loss



Εικόνα 43 Policy Loss

7 Συμπεράσματα

Στο κεφάλαιο αυτό γίνεται μία σύνοψη της έρευνας που ακολουθήθηκε σε αυτή τη διατριβή, καθώς και μελλοντικές βελτιώσεις που μπορούν να προστεθούν.

7.1 Σύνοψη

Ο στόχος της διατριβής αυτής ήταν η παρουσίαση του εργαλείου ML-Agents της Unity, ως τρόπος εκπαίδευσης ευφυούς πράκτορα χρησιμοποιώντας τη μέθοδο της ενισχυτικής μάθησης, καθώς και η χρήση του για την εκπαίδευση ενός πράκτορα. Ακόμα έγινε έρευνα και παρουσίαση διαφορετικών μεθόδων για την εκπαίδευση ευφυών πρακτόρων και δημιουργία τρισδιάστατου περιβάλλοντος στο οποίο μπορεί να εκπαιδευτεί καθώς και παρουσίαση των αποτελεσμάτων εκπαίδευσης. Ο στόχος επετεύχθη καθώς τα αποτελέσματα ήταν ικανοποιητικά και αντικατοπτρίζουν μία αρκετά ρεαλιστική συμπεριφορά του πράκτορα με πολύπλοκο στόχο σε ένα τρισδιάστατο περιβάλλον.

Κατά τη διάρκεια της διατριβής έγιναν αρκετές δοκιμές στις παραμέτρους εκπαίδευσης, καθώς και αλλαγές στο περιβάλλον έτσι ώστε να παραχθεί το επιθυμητό τελικό αποτέλεσμα. Επιπρόσθετα, αποκτήθηκε εμπειρία πάνω στη μηχανή παιχνιδιών Unity, καθώς και στη χρήση της εργαλειοθήκης ML-Agentς.

Για την κατανόηση της εργαλειοθήκης ML-Agents και την υλοποίηση αυτής της διατριβής, λήφθηκε υπόψιν διδακτικό υλικό το οποίο υπάρχει στην επίσημη σελίδα της.

7.2 Μελλοντικές επεκτάσεις

Σε αυτό το σημείο αναφέρονται κάποιες προτάσεις για μελλοντική έρευνα και επέκταση της εφαρμογής η οποία αναπτύχθηκε.

Μία από αυτές τις προτάσεις είναι ο συνδυασμός εκπαιδευμένων μοντέλων για πιο σύνθετη συμπεριφορά του πράκτορα. Θα μπορούσαν να εκπαιδευτούν περισσότερα από ένα νευρωνικά δίκτυα και να τεθούν ανάγκες στον πράκτορα έτσι ώστε ανάλογα με την ανάγκη που έχει τη συγκεκριμένη στιγμή να ενεργοποιείται το αντίστοιχο νευρωνικό δίκτυο. Για παράδειγμα, όταν υπάρχει ανάγκη για τροφή να ενεργοποιείται το νευρωνικό δίκτυο που αναπτύχθηκε, όταν υπάρχει ανάγκη για νερό να χρησιμοποιείται ένα αντίστοιχο το οποίο να του θέτει ως στόχο να μετακινείται προς κάποια πηγή νερού ή όταν υπάρχει ανάγκη για ξεκούραση να ενεργοποιείται κάποιο άλλο που να τον μετακινεί από το σημείο που βρίσκεται προς κάποιο σημείο ξεκούρασης.

Ακόμα μία πρόταση μπορεί να είναι η δοκιμή του νευρωνικού δικτύου σε μεγαλύτερα περιβάλλοντα και η παρατήρηση της συμπεριφοράς του. Στην τρέχουσα υλοποίηση το περιβάλλον είναι χτισμένο για να επιτευχθεί το επιθυμητό αποτέλεσμα, αλλά το νευρωνικό δίκτυο μπορεί να επαναχρησιμοποιηθεί.

Επιπλέον μπορεί να γίνουν αλλαγές στις παραμέτρους εκπαίδευσης και στο τρισδιάστατο περιβάλλον ώστε να παραχθεί ένα περισσότερο ικανοποιητικό αποτέλεσμα σε σχέση με αυτό της υλοποίησης.

Τέλος θα μπορούσε να γίνει εκπαίδευση και σύγκριση με τις μεθόδους που έχουν αναλυθεί στο κεφάλαιο 2 και να παρατηρηθούν, καθώς και να αναλυθούν οι αντίστοιχες συμπεριφορές.

8 Βιβλιογραφία

[1] R. Miikkulainen, “Creating Intelligent Agents in Games,” p. 9, 2006.

[2] N. Avradinis, T. Panayiotopoulos, and G. Anastassakis, “Behavior believability in virtual worlds: agents acting when they need to,” *SpringerPlus*, vol. 2, no. 1, p. 246, Dec. 2013, doi: 10.1186/2193-1801-2-246.

[3] G. Brockman *et al.*, “OpenAI Gym.” arXiv, Jun. 05, 2016. Accessed: Nov. 30, 2022. [Online]. Available: http://arxiv.org/abs/1606.01540

[4] Y. Tian, Q. Gong, W. Shang, Y. Wu, and C. L. Zitnick, “ELF: An Extensive, Lightweight and Flexible Research Platform for Real-time Strategy Games.” arXiv, Nov. 10, 2017. Accessed: Nov. 30, 2022. [Online]. Available: http://arxiv.org/abs/1707.01067

[5] M. Johnson, K. Hofmann, T. Hutton, and D. Bignell, “The Malmo Platform for Artificial Intelligence Experimentation,” p. 2.

[6] D. Jagdale, “Finite State Machine in Game Development,” *Int. J. Adv. Res. Sci. Commun. Technol.*, pp. 384–390, Oct. 2021, doi: 10.48175/IJARSCT-2062.

[7] Y. A. Sekhavat, “Behavior Trees for Computer Games,” *Int. J. Artif. Intell. Tools*, vol. 26, no. 02, p. 1730001, Apr. 2017, doi: 10.1142/S0218213017300010.

[8] J. Orkin, “Applying Goal-Oriented Action Planning to Games,” p. 11.

[9] J. C. G. Noel, “Reinforcement Learning Agents in Colonel Blotto,” arXiv, arXiv:2204.02785, Apr. 2022. Accessed: Nov. 27, 2022. [Online]. Available: http://arxiv.org/abs/2204.02785

[10] OpenAI *et al.*, “Dota 2 with Large Scale Deep Reinforcement Learning,” arXiv, arXiv:1912.06680, Dec. 2019. Accessed: Nov. 27, 2022. [Online]. Available: http://arxiv.org/abs/1912.06680

[11] A. Juliani *et al.*, “Unity: A General Platform for Intelligent Agents,” arXiv, arXiv:1809.02627, May 2020. Accessed: Nov. 27, 2022. [Online]. Available: http://arxiv.org/abs/1809.02627

[12] “mlagents · PyPI.” https://pypi.org/project/mlagents/ (accessed Nov. 28, 2022).

[13] R. Biswas and X. Lu, “Designing a Micro-Benchmark Suite to Evaluate gRPC for TensorFlow: Early Experiences,” p. 9.

[14] D. Blyth, J. Alcaraz, S. Binet, and S. V. Chekanov, “ProIO: An Event-Based I/O Stream Format for Protobuf Messages,” *Comput. Phys. Commun.*, vol. 241, pp. 98–112, Aug. 2019, doi: 10.1016/j.cpc.2019.03.018.

[15] “pipenv · PyPI.” https://pypi.org/project/pipenv/ (accessed Nov. 28, 2022).

[16] “Unity - Manual: Prefabs.” https://docs.unity3d.com/Manual/Prefabs.html (accessed Nov. 29, 2022).

[17] “TensorBoard | TensorFlow.” https://www.tensorflow.org/tensorboard (accessed Nov. 30, 2022).