ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ
ΠΛΗΡΟΦΟΡΙΚΗΣ

Σχεδιασμός και ανάπτυξη μηχανισμού επιβράβευσης μέσω Blockchain για ενισχυτική μάθηση (Reinforcement Learning)

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Θεόδωρος - Θηρίμαχος Κ. Δαβαράκης

Επιβλέπουσα : Θεοδώρα Βαρβαρίγου
Καθηγήτρια Ε.Μ.Π.

Αθήνα, Ιούλιος 2020

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ
ΠΛΗΡΟΦΟΡΙΚΗΣ

Σχεδιασμός και ανάπτυξη μηχανισμού επιβράβευσης μέσω Blockchain για ενισχυτική μάθηση (Reinforcement Learning)

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Θεόδωρος - Θηρίμαχος Κ. Δαβαράκης

Επιβλέπουσα : Θεοδώρα Βαρβαρίγου
Καθηγήτρια Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 29η Ιουλίου 2020.

.....	.....	.....
Θ.Βαρβαρίγου	Ε. Βαρβαρίγος	Σ. Παπαβασιλείου
Ιδίοτητα Μέλους Δ.Ε.Π	Ιδίοτητα Μέλους Δ.Ε.Π	Ιδίοτητα Μέλους Δ.Ε.Π

Αθήνα, Ιούλιος 2020

.....

Θεόδωρος-Θηρίμαχος Κ. Δαβαράκης

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Copyright © Θεόδωρος-Θηρίμαχος Κ. Δαβαράκης, 2020

Με επιφύλαξη παντός δικαιώματος. All rights reserved.

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα.

Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και δεν πρέπει να ερμηνευθεί ότι αντιπροσωπεύουν τις επίσημες θέσεις του Εθνικού Μετσόβιου Πολυτεχνείου.

ΠΕΡΙΛΗΨΗ

Δύο από τις πιο hot τεχνολογίες των τελευταίων χρόνων είναι ίσως οι τεχνολογίες της μηχανικής μάθησης και του blockchain. Η μηχανική μάθηση που αποτελεί και υποσύνολο της τεχνητής νοημοσύνης χρησιμοποιείται για την εξαγωγή γνώσης από τα δεδομένα. Το blockchain είναι ένα είδος ‘δημόσιου καθολικού΄ στο οποίο καταγράφονται αμετάβλητα και με ασφάλεια δεδομένα (όπως οικονομικές δοσοληψίες ή γενικότερα οτιδήποτε έχει αξία).

Ο συνδυασμός της μηχανικής μάθησης με την τεχνολογία blockchain αποτελεί μία δύσκολη, απαιτητική αλλά και προκλητική διαδικασία. Ο χώρος έρευνας και πρακτικής υλοποίησης που ανοίγεται είναι ευρύτατος και υπό εξερεύνηση. Η κάθε μία τεχνολογία μπορεί να αντιμετωπίσει τα κενά της άλλης. Η μηχανική μάθηση μπορεί να χρησιμοποιήσει δεδομένα που είναι αποθηκευμένα σε Blockchain και σε κατανεμημένη υπολογιστική ισχύ. Με την σειρά του το Blockchain μπορεί να χρησιμοποιήσει την μηχανική μάθηση για να κεφαλαιοποιήσει δεδομένα που ελέγχονται από χρήστες ή/και να δημιουργήσει marketplaces για μοντέλα μηχανικής μάθησης.

Στα πλαίσια της εργασίας προτείνεται ένας συνδυασμός της τεχνολογίας μηχανικής μάθησης και πιο συγκεκριμένα της ενισχυτικής μάθησης (reinforcement learning) και της μάθησης με απομίμηση (imitation learning) με την τεχνολογία blockchain. Η ενισχυτική μάθηση δίνει την δυνατότητα σε έναν software agent να αλληλοεπιδρά με το περιβάλλον του και να μαθαίνει - μέσω της διαδικασίας μαθαίνω από τα λάθη (trial and error) - βασιζόμενος αποκλειστικά στις δικές του ενέργειες, εμπειρίες και παρατηρήσεις. Επί της ουσίας ο software agent μαθαίνει μέσω της επιβράβευσης / αποθάρρυνσης που δέχεται από το περιβάλλον του. Ο σχεδιασμός του μηχανισμού επιβράβευσης είναι σημαντικός και σε πολύπλοκα συστήματα πραγματικά πολύ δύσκολος. Οι σχεδιαστές πρέπει να σχεδιάσουν τεχνικές που θα διασφαλίζουν ότι το περιβάλλον με συνέπεια αναγνωρίζει και επιβραβεύει την επιθυμητή συμπεριφορά του agent και ότι ο μηχανισμός επιβράβευσης δεν καταστρατηγείται ούτε παρακάμπτεται. Μία προσέγγιση στο ζήτημα είναι να συνδυαστεί η ενισχυτική μάθηση με την μάθηση με απομίμηση. Στην μάθηση με απομίμηση εκτός του software agent που πρόκειται να εκπαιδευτεί υπάρχει και ο software agent που θα τον εκπαιδεύσει μέσω της καταγραφής της συμπεριφοράς του σε αρχεία επίδειξης. Ο εκπαιδευόμενος – επί της ουσίας – εκπαιδεύεται μιμούμενος την συμπεριφορά του εκπαιδευτή.

Η πρόταση που γίνεται στα πλαίσια αυτής της εργασίας περιλαμβάνει τις έννοιες του άρτια εκπαιδευμένου software agent (Trainer agent) ο οποίος καταγράφει την συμπεριφορά του σε αρχεία επίδειξης (demo) και τα διαθέτει μέσω του blockchain σε άλλους software agents (Trainee agents) που αλληλοεπιδρούν σε ίδιο ή παρόμοιο περιβάλλον με αυτό του Trainer agent και θέλουν να εκπαιδευτούν. Η εκπαίδευση τους γίνεται με αλγορίθμους ενισχυτικής μάθησης (μέσω επιβράβευσης / αποθάρρυνσης) σε συνδυασμό με αλγορίθμους μάθησης με απομίμηση (μέσω αρχείων επίδειξης). Τα αρχεία επίδειξης ‘αποθηκεύονται΄ σε blockchain έξυπνα συμβόλαια (smart contracts) και στο τέλος, το blockchain επιβραβεύει τον Trainer agent ανάλογα με το πόσο βοήθησε στην βελτίωση των μοντέλων του Trainee agent. Η αμετάβλητη δομή του blockchain και τα έξυπνα συμβόλαια που δεν τροποποιούνται διασφαλίζουν την ποιότητα και το αμετάβλητο των αρχείων επίδειξης καθώς και την εγκυρότητα των συναλλαγών μεταξύ των συμβαλλόμενων μερών. Η αποκεντρωμένη (dApp) εφαρμογή που υλοποιήθηκε **αυτοματοποιεί πλήρως σαν μία ροή** την αγοραπωλησία των αρχείων επίδειξης και την εκπαίδευση του Trainee agent σε έναν κύκλο λειτουργίας.

Λέξεις Κλειδιά: Blockchain, Ethereum, Έξυπνα συμβόλαια, Μηχανική Μάθηση, Ενισχυτική Μάθηση, Μάθηση με απομίμηση, ML-Agents

ABSTRACT

In the last years Machine Learning and Blockchain technologies have been at the spearhead of innovation, both in the research and application fields. Machine Learning, being a sub-domain of Artificial Intelligence, is predominantly used to enable data knowledge extraction. Blockchain on the other hand is respected for providing the opportunity of a 'public ledger' upon which data are securely, consistently and irreversibly recorded.

Combining Machine Learning and Blockchain technologies determines a difficult, very demanding and at the same time challenging process. This process broadens field research extending it to new frontiers, as well as it enables numerous diverse new applications and use cases. It is common knowledge that these 'two leg' technologies may act in complementarity, especially as each technology leg may address weaknesses of the other.

Machine Learning may use data stored on Blockchains and pursue to exploit distributed computing resources. On the other hand, Blockchain may exploit Machine Learning and capitalise user generated (and controlled) data and possibly establish market-places for Machine Learning models.

In this work we propose a combination of Machine Learning technologies; and in particular reinforcement learning and imitation learning; with Blockchain.

Reinforcement learning allows a software agent to interact with its environment and learn – via 'trial and error' techniques – based exclusively on its own activity, experiences and observations. In essence this software agent will learn via an interactions' reward/ penalise set of measures, immediately received from its own environment. Designing an interactions' reward/ penalise mechanism is very important and challenging especially in the case of complex and connected cyber-physical systems. Designers need to draw focused techniques securing that the agents' immediate environment will consistently recognize and reward desirable agent behaviour and that the rewarding mechanism can not be tapped, corrupted or circumvented. In this work this has been approached via a coordinated collaboration of reinforcement and imitation learning.

Using imitation learning apart from the trainee software agent, a very important subject is the trainer software agent. The latter takes on the task of training via recording its own environmental behaviour in a demonstration file. In this respect, the trainee may imitate tis trainers' good practices, and ultimately become trained.

This work proposes the concept of an expert trainer software agent (the Trainer Agent) who records its own behaviour in a demonstration file (demo) and distributes these files via Blockchain to other (receiving) software agents (Trainee agents). The Trainees' training is applied using reinforcement learning techniques (i.e. reward/ penalise) in conjunction with imitation learning (based on demo files). The demo files are 'stored' on smart contract Blockchains, who in the end get to reward the Trainer agent; pro-rated according to the level with which the Trainer has assisted to the improvement of the Trainee agent models. The invariant Blockchain structure with the unmodifiable nature of the smart contracts secure the quality and protection of the demo files as well as the credibility of the interactions among all stakeholders involved. The dApp application produced **fully automates** the workflow of the demonstrations' trading and the Trainee agent training process.

Key words: Blockchain, Ethereum, Smart contracts, Machine Learning, Reinforcement Learning, Imitation Learning, ML-Agents

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

1. ΕΙΣΑΓΩΓΗ

Η προσέγγιση των τεχνολογιών blockchain και μηχανικής μάθησης είναι αναπόφευκτη καθώς και οι δύο τεχνολογίες χρησιμοποιούν και διαχειρίζονται δεδομένα και αξίες. Η τεχνολογία blockchain προσφέρει ασφαλή αποθήκευση και διαμοιρασμό δεδομένων ή αξιών ενώ η τεχνολογία της μηχανικής μάθησης αναλύει δεδομένα και παράγει νέες γνώσεις – αξίες. Ήδη εφαρμοσμένες λύσεις μηχανικής μάθησης τρέχουν μέσω των blockchains αυξάνοντας έτσι τις δυνατότητες εφαρμογής των αλγορίθμων μηχανικής μάθησης ή/και βελτιώνοντας την ποιότητα στις αποκεντρωμένες εφαρμογές blockchain.

Στα πλαίσια της εργασίας αυτής θα μελετηθούν οι περιοχές στις οποίες μπορεί να γίνει συνδυασμός των τεχνολογιών blockchain και μηχανικής μάθησης και θα παρουσιασθεί η υλοποίηση ενός τέτοιου συνδυασμού κατά τον οποίο μοντέλα ενισχυτικής μάθησης (reinforcement learning) εκπαιδεύονται ή / και βελτιώνονται με την χρήση δεδομένων (αρχείων επίδειξης) που διατίθενται μέσω blockchain.

1.1 ΑΝΤΙΚΕΙΜΕΝΟ ΤΗΣ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Η τεχνητή νοημοσύνη (artificial intelligence) και το blockchain αποτελούν ίσως τις πιο συζητημένες τεχνολογίες της τελευταίας διετίας, τουλάχιστον.

Το blockchain είναι ένα είδος ‘δημόσιου καθολικού΄ στο οποίο καταγράφονται όχι μόνο οικονομικές δοσοληψίες αλλά γενικότερα οτιδήποτε έχει αξία. Το βασικό χαρακτηριστικό του blockchain είναι ότι επιτρέπει σε δύο μέρη που δεν γνωρίζονται μεταξύ τους να εκτελούν δοσοληψίες και να ανταλλάσσουν δεδομένα μεταξύ τους μέσω ενός κοινού ‘λογιστικού καθολικού’.

Η τεχνητή νοημοσύνη επιτρέπει στους υπολογιστές να εκτελούν ενέργειες που εξομοιώνουν φυσικές συμπεριφορές ή/και προσομοιάζουν στην ανθρώπινη νοημοσύνη. Τα μοντέλα τεχνητής νοημοσύνης αναλύουν δεδομένα, ταξινομούν και μαθαίνουν γνώσεις και προβλέπουν. Σε αντίθεση με τα παραδοσιακά λογισμικά τα μοντέλα τεχνητής νοημοσύνης βελτιώνονται συνεχώς όσο τροφοδοτούνται/εκπαιδεύονται με νέα δεδομένα.

Η μηχανική μάθηση που αποτελεί υποσύνολο της τεχνητής νοημοσύνης χρησιμοποιείται για την εξαγωγή γνώσης από τα δεδομένα. Όσο περισσότερα δεδομένα είναι διαθέσιμα τόσο καλύτερα γίνονται τα μοντέλα μηχανικής μάθησης. Ταυτόχρονα, η διάθεση ποιοτικών, επίκαιρων και σχετικών δεδομένων οδηγεί σε πιο αποδοτικά μοντέλα.

Η ύπαρξη των δεδομένων είναι καίριας σημασίας για την τεχνητή νοημοσύνη. Από την άλλη μεριά το blockchain επιτρέπει τον προστατευμένο και ασφαλή διαμοιρασμό των δεδομένων. Το blockchain μπορεί να διασφαλίσει τόσο την αξιοπιστία των δεδομένων όσο και την ασφαλή διάθεσή τους στα μοντέλα μηχανικής μάθησης.

Προσφάτως, στον χώρο της τεχνητής νοημοσύνης παρατηρούνται σημαντικές πρόοδοι αλλά η πρόσβαση σε αυτές, ώστε να αντληθούν τα πλεονεκτήματα της μηχανικής μάθησης και να κάνουν αυτές τις προόδους εφαρμόσιμες, μπορεί να αποβεί δύσκολο στοίχημα ειδικά για όσους διαθέτουν περιορισμένους πόρους.

Συνήθως, οι αλγόριθμοι μηχανικής μάθησης τρέχουν σε μεγάλα κεντρικά υπολογιστικά συστήματα στο υπολογιστικό νέφος, ενώ τα δεδομένα εισόδου που απαιτούνται για την εκπαίδευση τους είναι ιδιωτικά και προστατευμένα ως προς τα πνευματικά δικαιώματα.

Ταυτόχρονα η δημιουργία αξιόπιστων και μεγάλου όγκου δεδομένων ή η αγορά τους είναι πολύ ακριβή. Παράλληλα οι δημοσιευμένοι αλγόριθμοι μηχανικής μάθησης πολύ γρήγορα απαξιώνονται αν δεν τροφοδοτούνται συνεχώς με νέα δεδομένα για την συνεχή εκπαίδευσή τους.

Για τους παραπάνω λόγους είναι χρήσιμη η συνεργασία της τεχνολογίας μηχανικής μάθησης με την τεχνολογία blockchain.. Τα μοντέλα μηχανικής μάθησης μπορούν να εκπαιδεύονται ή/και να βελτιώνονται με χρήση δεδομένων που διατίθενται μέσω blockchain. Ενώ παράλληλα υπηρεσίες blockchain μπορούν να ανταμείβουν αυτούς που προσφέρουν δεδομένα για την βελτίωση των μοντέλων μηχανικής μάθησης.

Στα πλαίσια της εργασίας προτείνεται ένας συνδυασμός της τεχνολογίας μηχανικής μάθησης και πιο συγκεκριμένα της ενισχυτικής μάθησης (reinforcement learning) με την τεχνολογία blockchain. Η ενισχυτική μάθηση δίνει την δυνατότητα σε έναν software agent να αλληλοεπιδρά με το περιβάλλον του και να μαθαίνει - μέσω της διαδικασίας μαθαίνω από τα λάθη (trial and error) - βασιζόμενος αποκλειστικά στις δικές του ενέργειες, εμπειρίες και παρατηρήσεις. Πρόκειται για μια τεχνική που προσομοιάζει με την τεχνική που μαθαίνουν οι άνθρωποι και που μπορεί να παράξει γρήγορα το καλύτερο δυνατό μοντέλο για ένα συγκεκριμένο πρόβλημα όπου άλλες τεχνικές μηχανικής μάθησης αποτυγχάνουν. Εφαρμογές όπου η τεχνική της ενισχυτικής μάθησης είναι αποδοτική είναι τα ρομποτικά συστήματα για βιομηχανικούς αυτοματισμούς, συστήματα αυτόματης κίνησης, συστήματα ελέγχου αεροσκαφών, gaming κλπ.

Όμως οι αλγόριθμοι ενισχυτικής μάθησης απαιτούν αλληλοεπίδραση με το περιβάλλον, το οποίο πρέπει να είναι αρκετά ‘έξυπνο’ ώστε να παρέχει στο μοντέλο την απαραίτητη ανάδραση (feedback) σε κάθε ενέργειά του. Έτσι οι σχεδιαστές των μοντέλων ενισχυτικής μάθησης είναι υποχρεωμένοι να κατασκευάζουν συχνά πολύπλοκα ‘virtual’ περιβάλλοντα που εξομοιώνουν τον πραγματικό κόσμο. Επιπρόσθετα, καθώς οι εφαρμογές της ενισχυτικής μάθησης εφαρμόζονται κυρίως σε συστήματα υψηλού ρίσκου (π.χ. συστήματα αυτόματης κίνησης) ή/και σε συστήματα υψηλού κόστους επιδιόρθωσης ή αντικατάστασης (π.χ. ρομποτικά συστήματα) οι σχεδιαστές κατασκευάζουν συχνά ‘virtual’ software agents που εξομοιώνουν το αντικείμενο που πρόκειται να εκπαιδευτεί (π.χ. ρομποτικός βραχίονας). Τέλος, οι αλγόριθμοι ενισχυτικής μάθησης χρειάζονται μεγάλο όγκο δεδομένων (ενέργειες, αναδράσεις, εμπειρίες, παρατηρήσεις) και μεγάλη υπολογιστική ισχύ.

Η πρόταση που γίνεται στα πλαίσια αυτής της εργασίας αφορά την διάθεση του μεγάλου αλλά αναγκαίου όγκου δεδομένων με την μορφή αρχείων επίδειξης μέσω αποκεντρωμένων blockchain εφαρμογών. Η πρόταση περιλαμβάνει τις έννοιες του άρτια εκπαιδευμένου software agent (Trainer agent) ο οποίος καταγράφει την συμπεριφορά του σε αρχεία επίδειξης (demo) και τα διαθέτει μέσω του blockchain σε άλλους software agents (Trainee agents) που αλληλοεπιδρούν σε ίδιο ή παρόμοιο περιβάλλον με αυτό του Trainer agent και θέλουν να εκπαιδευτούν. Η εκπαίδευση τους ακολουθεί την βασική αρχή της ενισχυτικής μάθησης, δηλαδή επιβράβευση / αποθάρρυνση για κάθε ενέργεια έως την επίτευξη του στόχου. Εμπλουτίζεται όμως και με τεχνικές behavioural cloning ή/και imitation learning οι οποίες λαμβάνουν υπόψη τα προσφερόμενα αρχεία επίδειξης. Μέσω αυτών των τεχνικών ο Trainee εκπαιδεύεται στην επίτευξη του στόχου με το να ‘μιμείται’ την συμπεριφορά του Trainer agent που λεπτομερώς έχει καταγραφεί στο αρχείο επίδειξης. Τα αρχεία επίδειξης ‘αποθηκεύονται΄ σε blockchain έξυπνα συμβόλαια (smart contracts) και στο τέλος, το blockchain επιβραβεύει τον Trainer agent ανάλογα με το πόσο βοήθησε στην βελτίωση των μοντέλων του Trainee agent. Η αμετάβλητη δομή του blockchain και τα έξυπνα συμβόλαια που δεν τροποποιούνται και που αξιολογούνται συνεχώς από την πληθώρα μηχανών που συμμετέχουν στην αρχιτεκτονική του blockchain διασφαλίζουν την ποιότητα και το αμετάβλητο των αρχείων επίδειξης.

Για την υλοποίηση αυτής της πρότασης, σχεδιάστηκε μια αποκεντρωμένη (dApp) εφαρμογή, ένα είδος marketplace αρχείων επίδειξης που **αυτοματοποιεί πλήρως σαν μία ροή** την αγοραπωλησία των αρχείων επίδειξης και την εκπαίδευση του agent σε έναν κύκλο

λειτουργίας. Δυστυχώς δεν κατέστη δυνατό να ενσωματωθεί σε αυτή την αυτόματη ροή και η καταγραφή των αρχείων επίδειξης.

1.2 ΟΡΓΑΝΩΣΗ ΤΟΥ ΚΕΙΜΕΝΟΥ

Το κείμενο της διπλωματικής εργασίας οργανώνεται ως εξής:

Το Κεφάλαιο 1 περιλαμβάνει την εισαγωγή στο ζήτημα του συνδυασμού των τεχνολογιών της τεχνητής νοημοσύνης και του blockchain και το αντικείμενο της διπλωματικής εργασίας.

Το Κεφάλαιο 2 θέτει το θεωρητικό υπόβαθρο της εργασίας. Περιγράφονται με συντομία η τεχνολογία του blockchain και η φιλοσοφία και οι αλγόριθμοι που υποστηρίζουν την ενισχυτική μάθηση και την μάθηση με απομίμηση. Επιπλέον επίσης με συντομία περιγράφεται και η τεχνολογία IPFS.

Στο Κεφάλαιο 3 καταγράφονται και επεξηγούνται τα εργαλεία και οι τεχνικές που έχουν χρησιμοποιηθεί κατά την υλοποίηση της εργασίας.

Η ανάλυση των απαιτήσεων της εφαρμογής παρουσιάζεται στο Κεφάλαιο 4. Περιγράφονται οι λειτουργικές και μη απαιτήσεις που πρέπει να ικανοποιούνται από την εφαρμογή, η δομή και τα συστήματα που την απαρτίζουν και τέλος περιγράφεται η μεθοδολογία που ακολουθήθηκε για την ανάπτυξη της εφαρμογής.

Το κεφάλαιο 5 περιγράφει την βασική ροή εργασιών που υλοποιεί η εφαρμογή, τα συστήματα που την απαρτίζουν, το πως επικοινωνούν μεταξύ τους και στην συνέχεια αναλύει την λειτουργικότητα που το κάθε σύστημα υποστηρίζει.

Το κεφάλαιο 6 περιγράφει τον σχεδιασμό και την εκτέλεση των πειραμάτων που πραγματοποιήθηκαν στα πλαίσια της εφαρμογής.

Τέλος, το κεφάλαιο 7 περιλαμβάνει τα συμπεράσματα που προέκυψαν κατά την διάρκεια της εργασίας, το κεφάλαιο 8 καταγράφει την βιβλιογραφία που χρησιμοποιήθηκε ενώ στο κεφάλαιο Παράρτημα επισυνάπτεται ο κώδικας των έξυπνων συμβολαίων και ένα δείγμα αρχείου παραμετροποίησης (configuration file) εκπαίδευσης.

2 ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ

2.1 BLOCKCHAIN – ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ

Η **Τεχνητή Νοημοσύνη (ΤΝ)** και το **Blockchain (BCHN)** είναι οι πλέον πολυσυζητημένες τάσεις καινοτομίας τα δύο τελευταία χρόνια και βέβαια υπάρχει πραγματικός λόγος που έχουν τραβήξει τόσο προσοχή πάνω τους:

- Η ΤΝ υπόσχεται να αυτοματοποιήσει πολλές δράσεις και συχνά αποδεικνύεται καλύτερη στην μοντελοποίηση πολύπλοκων καταστάσεων από ότι ο ανθρώπινος νους.

- Το Blockchain προσφέρει μεγαλύτερη ασφάλεια δεδομένων και προστασία ιδιωτικότητας ενώ παράλληλα μειώνει τα κόστη και την συγκεντρωμένη στους μεγάλους οργανισμούς εξουσία.

Αν και μπορεί να μοιάζει περίεργο, υπάρχουν πραγματικές ανάγκες πού εξηγούν γιατί αυτές οι δύο τεχνολογίες θα μπορούσαν να προσθέσουν αξία όταν συνεργαστούν. Σε γενικές γραμμές η κάθε μία αντιμετωπίζει τα κενά της άλλης αλληλοσυμπληρούμενα. Ήδη επιστημονικά άρθρα αναφέρουν ότι η ΤΝ μπορεί να βασίζεται σε datasets που είναι αποθηκευμένα σε Blockchain και σε κατανεμημένη υπολογιστική ισχύ. Με την σειρά του το Blockchain μπορεί να χρησιμοποιήσει την ΤΝ για να κεφαλαιοποιήσει δεδομένα που ελέγχονται από χρήστες, να δημιουργήσει marketplaces για μοντέλα ΤΝ.

Είναι σημαντικό να σημειώσουμε ότι και τα δύο ΤΝ και Blockchain είναι στα πρωταρχικά στάδια ανάπτυξης. Σίγουρα θα υπάρξουν πολλές και σημαντικές εξελίξεις στην επόμενη δεκαετία που θα αναπτύξουν καινοτόμες μεθόδους.

Ο συνδυασμός των δύο δεν γίνεται χωρίς προβλήματα. Οι αδυναμίες του Blockchain ως μέσω αποθήκευσης και ανάκτησης πληροφοριών το καθιστά δυσκολότερο στην χρήση, τουλάχιστον ως προς τις παραδοσιακές βάσεις δεδομένων για ΤΝ εφαρμογές. Επιπροσθέτως όμως η κωδικοποίηση ενός smart contract είναι πολύ ευκολότερη από την σχεδίαση ενός μοντέλου ΤΝ που θα υλοποιεί ένα έξυπνο συμβόλαιο με ασφάλεια. Εάν όμως φτάσουμε σε αυτό το σημείο οι ευκαιρίες που θα ανοιχτούν είναι πολλές και πολύ ενδιαφέρουσες.

Οι δυσκολίες που παρουσιάζονται κατά την προσπάθεια συνδυασμού των ΤΝ και Blockchain οφείλονται κατά κύριο λόγο στις βασικές και θεμελιώδεις διαφορές τους.

Η πρώτη διαφορά ΤΝ και BCHN αφορά στην θέση, τις διαστάσεις και το μέγεθος της υπολογιστικής ισχύος. Η ΤΝ βασίζεται σε τεράστιο όγκο αποθηκευμένων δεδομένων, με την μεγαλύτερη δυνατή πληρότητα, και έχει ανάγκη μαζικής υπολογιστικής ισχύς για να εκπαιδεύσει τούς αλγορίθμους της. Η ΤΝ είναι μία κεντροποιημένη τεχνολογία.

Σε αντίθεση, το BCHN προβάλλει τον κατανεμημένο έλεγχο δεδομένων και υπολογιστικών πόρων, ενώ συγχρόνως καθιστά την πρόσβαση στα δεδομένα, και τούς πόρους του, δικτυακά διαθέσιμα. Αυτή η κατανομή έρχεται με κόστος την υπολογιστική καθυστέρηση, ενώ εξίσου σημαντική εργασία χρειάζεται να γίνει στην επιτάχυνση των BCHN λογιστικών ελέγχων ώστε να μπορούν να χρησιμοποιηθούν μαζί με την ΤΝ.

Η δεύτερη θεμελιώδης διαφορά ανάμεσα στην ΤΝ και το BCHN είναι το πως αντιμετωπίζουν την ανάγκη για διαφάνεια. Η θεμελιώδης αρχή του BCHN είναι η διαφάνεια αφού το 'δημόσιο καθολικό' των δημόσιων BCHNs είναι ανοικτό σε όλους. Ενώ τα δεδομένα είναι ανώνυμα, είναι πάντα διαθέσιμα και έτσι το BCHN γίνεται διαφανές όσον αφορά το πως οι πράξεις καταγράφονται και προστίθενται. Από την φύση τους τα ανοικτά ομότιμα δίκτυα δημιουργούν εμπιστοσύνη μεταξύ των εταίρων χρησιμοποιώντας δημόσια κρυπτογράφηση.

Σε αντιδιαστολή οι αλγόριθμοι ΤΝ και τα νευρωνικά δίκτυα είναι πολύ δύσκολα στην κατανόηση τους (άρα όχι διάφανα). Για την κατανόηση των στατιστικών αλγορίθμων απαιτείται προχωρημένη γνώση γραμμικής άλγεβρας, ενώ ακόμα και οι ερευνητές που αναπτύσσουν τα deep learning μοντέλα συχνά τα αντιμετωπίζουν σαν μαύρα κουτιά.

Η διαφάνεια στην ΤΝ είναι ένα πρόβλημα που ίσως στο μέλλον καταφέρει να αντιμετωπίσει το BCHN. Η φύση του deep learning και το τρέχων επίπεδο κατανόησης, γύρω από την διαδικασία εκπαίδευσης των αλγορίθμων, το χαρακτηρίζει αδιαφανές.

Αυτό όμως που μπορεί να αντιμετωπίσει/λύσει το BCHN είναι η δημόσια πρόσβαση στα δεδομένα που χρησιμοποιούνται για να εκπαιδευτούν αυτοί οι αλγόριθμοι. Πράγμα που μπορεί να φωτίσει σημαντικές πλευρές των αδυναμιών ή προκαταλήψεων που τα μοντέλα αυτά δημιουργούν. Εδώ πρέπει να αναφερθεί ο 'εμπειρικός' ισχυρισμός ότι ένα μοντέλο ΤΝ είναι τόσο καλό όσο τα δεδομένα στα οποία εκπαιδεύτηκε. Δημόσιες και ανεξάρτητες

εποπτείες (audits) των δεδομένων εκπαίδευσης αποτελούν κλειδί σε αξιόπιστη ΤΝ, και αυτό το BCHN μπορεί να το διευκολύνει.

Είναι σαφές λοιπόν, ότι η ΤΝ και το BCHN βρίσκονται σε διαμετρικά αντίθετες διαδρομές, όσον αφορά την στόχευση επίτευξης γρήγορων νοητικών καταγραφών και αποφάσεων, που θα βασίζονται σε τεράστια μάζα δεδομένων εκπαίδευσης και αφθονία υπολογιστικών πόρων.

Το BCHN μπορεί να παρέχει αυτά τα δεδομένα και τούς πόρους αλλά με πιο αργή, περισσότερο διάφανη και κατανεμημένη διεργασία. Η ταχύτητα είναι σήμερα το εμπόδιο στην συνεργασία. Η ανάπτυξη και επεκτασιμότητα τού BCHN μπορεί να λύσει το θέμα στο μέλλον.

Ο συνδυασμός λοιπόν μπορεί να υλοποιήσει σειρά εφαρμογών που σε γενικές γραμμές στοχεύουν στον εκδημοκρατισμό του οφέλους και της πρόσβασης στην ΤΝ. [1][2][3][4][5][58][59]

Μία από τις κυρίαρχες εφαρμογές που συνδυάζει τις δύο περιοχές χρησιμοποιεί την έννοια των δικτύων εξόρυξης (mining network), σαν ένα μέσο για την ανεύρεση υπολογιστικής ισχύος ώστε να εκπαιδεύονται οι αλγόριθμοι ΤΝ. Η μεγάλη χρονική αναμονή, λόγω της ανάγκης εκπαίδευσης των μοντέλων, είναι ένα από τα βασικά στοιχεία καθυστέρησης στην ανάπτυξη της μηχανικής μάθησης. Η τάση είναι να οργανωθούν αποκεντρωμένα δίκτυα εξόρυξης GPUs, ώστε ο οποιοσδήποτε, οπουδήποτε στον κόσμο, να μπορεί να αποκτά πρόσβαση σε υπερ-υπολογιστικές υπηρεσίες.

Μία δεύτερη κατηγορία εφαρμογών αφορά την διατήρηση της ιδιωτικότητας των δεδομένων. Το BCHN μπορεί να μετατρέψει τα δεδομένα σε ανώνυμα, παρόλο που υπάρχει τρόπος να εξαχθεί η αρχική ονομασία. Αυτή όμως η περιοχή ενδείκνυται για ερευνητικούς σκοπούς, όπου θα καθίστανται διαθέσιμα ανώνυμα datasets, για να γίνονται αναλύσεις, προβλέψεις και εκπαιδεύσεις αλγορίθμων. Χρησιμοποιώντας αποκεντρωμένα και ανώνυμα δεδομένα μπορεί να επιτευχθεί εκπαίδευση μοντέλων λιγότερο biased, ώστε να είναι περισσότερο αξιόπιστα και αντιπροσωπευτικά τα αποτελέσματα τους.

Άλλη μία ευκαιρία συνδυασμού των ΤΝ και blockchain είναι η ανάγκη για αιτιολόγηση από πλευράς της ΤΝ της συμπεριφοράς και των αποτελεσμάτων των μοντέλων της. Καθώς η ΤΝ λειτουργεί σχεδόν σαν ‘μαύρο κουτί’ δεν μπορεί να αιτιολογήσει το σκεπτικό με το οποίο λαμβάνονται οι αποφάσεις των αλγορίθμων της. Η ύπαρξη καθαρού μονοπατιού ιχνηλάτησης αποφάσεων που προσφέρει το Blockchain θα βελτιώσει όχι μόνο την εμπιστοσύνη αλλά και θα παράσχει και τον ξεκάθαρο τρόπο ιχνηλάτησης των μηχανικών αποφάσεων. Επίσης, στην περίπτωση που οι αλγόριθμοι της ΤΝ κωδικοποιηθούν σε ένα αποκεντρωμένο περιβάλλον με την μορφή έξυπνων συμβολαίων τότε θα επιτρέπεται η εκτέλεση μόνον εκείνων των αποφάσεων που θα προσδιορίζονται ρητά από τα έξυπνα συμβόλαια και θα αποτρέπεται η εκτέλεση άλλων που μπορεί να οδηγήσουν σε καταστροφικές συνέπειες.

Μια άλλη κατηγορία συνδυασμού των τεχνολογιών ΤΝ και BCHN θα μπορούσε να είναι η αγοραπωλησία αλγορίθμων. Στην περίπτωση που θα απελευθερωθεί η πρόσβαση στα δεδομένα και την υπολογιστική ισχύ μέσω του BCHN, η ΤΝ θα αποκτήσει μία ευκαιρία ανάπτυξης και διαμοιρασμού. Προς αυτή την κατεύθυνση ήδη υπάρχουν περιπτώσεις ανοικτών πακέτων, open source packages, που ενέχουν κοινά μοντέλα μηχανικής μάθησης όπως τα TensorFlow, openAI, PyTorch, ML-Agents.

Λύσεις BCHN που βασίζονται σε δεδομένα και υπολογιστική ισχύ θα μπορούσαν να επεκτείνουν αυτή την τάση και να δημιουργήσουν μία Αγορά ΤΝ Αλγορίθμων. Έτσι οι επιχειρήσεις μπορεί να πληρώνουν τούς επιστήμονες για την ανάπτυξη του αλγορίθμου ή να αγοράζουν απευθείας την πρόσβαση στον αλγόριθμο για συνεχή χρήση.

Η ιδέα είναι, αντί η ανάπτυξη της ΤΝ να είναι εσωτερική και προστατευμένη, οι αλγόριθμοι ΤΝ να διατίθενται μέσω μιας ανοικτής αγοράς όπου ο οποιοσδήποτε να μπορεί να συμμετέχει και να προσθέτει αξία.

Όπως φάνηκε στο παρελθόν μέσα από την τάση Open Source και την ανάπτυξη των as-a-service λογισμικών, τέτοια είδη ανοικτών αγορών ενθαρρύνουν πολύ περισσότερο την καινοτομία από ότι η αποκλειστικά προστατευμένη ιδιωτική ανάπτυξη.

Τέλος, η αγοραπωλησία δεδομένων αποτελεί μία άλλη ευκαιρία συνδυασμού των ΤΝ και BCHN καθώς ένα χαρακτηριστικό τού BCHN είναι ότι οι χρήστες του αποκτούν μεγαλύτερο έλεγχο στα δεδομένα τους. Σε αντίθεση με την πρακτική που ακολουθούν οι μεγάλοι οργανισμοί (όπως η Google) να εκμεταλλεύονται τα ιδιωτικά δεδομένα με άρρητο τρόπο (χωρίς να το πολύ-καταλαβαίνουν οι χρήστες), μία υποδομή BCHN θα μπορέσει να δώσει στους χρήστες τον έλεγχο των δικών τους δεδομένων.

Αυτό ανοίγει την πόρτα σε όλους να μπορούν να διαπραγματεύονται τα δικά τους δεδομένα. Έτσι θα μπορούσαν να δημιουργηθούν ηλεκτρονικά Εμπορικά Κέντρα που θα διαθέτουν υπάρχοντα δεδομένα προς πώληση από τούς ίδιους του ιδιοκτήτες τους. Εδώ αξίζει να σημειωθεί ότι η Microsoft έχει ήδη ανακοινώσει μία ανάλογη προσπάθεια [1]

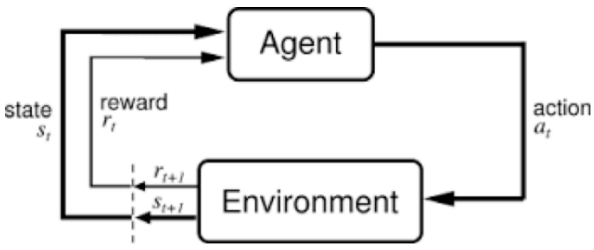
2.2 ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ – REINFORCEMENT LEARNING

2.2.1 Reinforcement Learning

Η ενισχυτική μάθηση (τμήμα της μηχανικής μάθησης) [6][7] δίνει την δυνατότητα σε έναν software agent να αλληλοεπιδρά με το περιβάλλον του και να μαθαίνει - μέσω της διαδικασίας μαθαίνω από τα λάθη (trial and error) βασιζόμενος αποκλειστικά στις δικές του ενέργειες, εμπειρίες και παρατηρήσεις.

Η βασική διαφορά της ενισχυτικής μάθησης από την supervised μάθηση είναι ότι αν και οι δύο τεχνικές επί της ουσίας χρησιμοποιούν δεδομένα για να παράξουν γνώση, στην μεν supervised μάθηση ο software agent χρειάζεται ως είσοδο ένα σύνολο από σωστά δεδομένα / ενέργειες, στην δε ενισχυτική μάθηση ο software agent χρειάζεται την επιβράβευση / αποθάρρυνση ως οδηγό για την θετική ή αρνητική συμπεριφορά του.

Σε σχέση με την unsupervised μάθηση που έχει σαν στόχο την αποκάλυψη των πιθανών μοτίβων που ακολουθούν τα δεδομένα, η ενισχυτική μάθηση έχει σαν στόχο είναι να βρεθεί εκείνο το μοντέλο ενεργειών που θα μεγιστοποιήσει τη συνολική επιβράβευση του agent.



Εικόνα 1 - Τυπικό μοντέλο ενισχυτικής μάθησης (Πηγή : Richard S. Sutton and Andrew G. Barto, 2018, MIT Press, 2nd edition, Reinforcement Learning – An Introduction)

Στην ενισχυτική μάθηση ορίζονται οι εξής έννοιες:

- Agent: ο agent ενεργεί (π.χ. ένας ρομποτικός βραχίονας που συσκευάζει προϊόντα)
- Περιβάλλον (Environment E): ο κόσμος μέσα στον οποίο ο agent ενεργεί, ο κόσμος που ανταποκρίνεται στις ενέργειες του agent. Το περιβάλλον δέχεται σαν είσοδο την τρέχουσα κατάσταση και την ενέργεια του agent και επιστρέφει τόσο την επιβράβευση (θετική / αρνητική) του agent όσο και την νέα του κατάσταση.
- Ενέργεια (Action a_t): η ενέργεια του agent. Ο agent επιλέγει μία ενέργεια από ένα σύνολο πιθανών ενεργειών A.

- Κατάσταση (State s_t): πρόκειται για την θέση στην οποία βρίσκεται ο agent.
- Επιβράβευση (Reward $r_{t)}$: πρόκειται για μια ένδειξη που μετρά πόσο πετυχημένη ή αποτυχημένη είναι μία ενέργεια του agent σε μία δεδομένη κατάσταση. Για κάθε κατάσταση του, ο agent στέλνει στο περιβάλλον μία ενέργεια και το περιβάλλον αναλόγως του επιστρέφει ένα ζεύγος τιμών που αντιστοιχούν στην επιβράβευσή του και στην νέα του κατάστασή (που είναι αποτέλεσμα της ενέργειας του agent με δεδομένη την τρέχουσα κατάστασή του). Μέσω της επιβράβευσης / αποθάρρυνσης ο agent αποφασίζει την επόμενη κίνησή του.
- Πολιτική (Policy π): είναι η στρατηγική με την οποία ο agent αποφασίζει την επόμενη κίνησή του δεδομένης της τρέχουσας κατάστασής του. Η πολιτική συσχετίζει τις καταστάσεις του agent με εκείνες τις ενέργειες που του επιφέρουν την μέγιστη επιβράβευση.
- Value (V): η προσδοκώμενη μακροπρόθεσμη αξία που θα κατακτηθεί σε αντιδιαστολή με την βραχυπρόθεσμη επιβράβευση (reward R)
- Q value: ορίζεται παρόμοια με την Value V με μόνη διαφορά ότι δέχεται μία επιπλέον παράμετρο, την τρέχουσα κατάσταση s.

Επομένως το περιβάλλον είναι μία συνάρτηση που δέχεται σαν είσοδο μία ενέργεια του agent που πραγματοποιήθηκε στα πλαίσια της τρέχουσας κατάστασής του και επιστρέφει την επόμενη κατάσταση του agent και την επιβράβευσή του. Ο agent είναι επίσης μία συνάρτηση που δέχεται σαν είσοδο την νέα του κατάσταση και την επιβράβευση του σε σχέση με την προηγούμενη ενέργεια του και επιστρέφει μία νέα ενέργεια με στόχο την μεγιστοποίηση της συνολικής του επιβράβευσης. Είναι φανερό ότι η συνάρτηση του agent είναι εύκολο να προσδιοριστεί. Η συνάρτηση όμως του περιβάλλοντος είναι σχεδόν στις περισσότερες περιπτώσεις άγνωστη, μπορεί μόνο να προσδιορισθεί μέσω των εισόδων που δέχεται και των εξόδων που επιστρέφει. Η ενισχυτική μάθηση επομένως είναι η προσπάθεια του agent να προσεγγίσει την συνάρτηση του περιβάλλοντος με στόχο την μεγιστοποίηση της επιβράβευσης που δέχεται από αυτό.

Τρεις είναι οι κατηγορίες αλγορίθμων που υλοποιούν ενισχυτική μάθηση:

- Value-Based αλγόριθμοι: προσπαθούν να βελτιστοποιήσουν την συνάρτηση V(s) η οποία υπολογίζει την μακροπρόθεσμη αξία που προσδοκά ο agent να κατακτήσει με δεδομένες τις καταστάσεις S και την πολιτική π.
- Policy-based αλγόριθμοι: προσπαθούν να βρουν την βέλτιστη πολιτική που θα επιτρέψει στον agent να κατακτήσει την μέγιστη επιβράβευση στο μέλλον. Υπάρχουν δύο υποκατηγορίες policy based αλγορίθμων, οι ντετερμινιστικοί αλγόριθμοι όπου για κάθε κατάσταση και για την ίδια πολιτική αποφασίζεται η ίδια ακριβώς ενέργεια και οι στοχαστικοί αλγόριθμοι όπου σε κάθε ενέργεια αποδίδεται μία πιθανότητα
- Model-based αλγόριθμοι: για κάθε περιβάλλον κατασκευάζεται ένα 'virtual' μοντέλο και ο agent μαθαίνει να λειτουργεί στο συγκεκριμένο περιβάλλον.

2.2.2 Το πρόβλημα του σχεδιασμού του μηχανισμού επιβράβευσης

Η ενισχυτική μάθηση προαπαιτεί την αποστολή εκ μέρους του περιβάλλοντος σε τακτά διαστήματα την επιβράβευση / αποθάρρυνση. Ο agent έχει στόχο την μεγιστοποίηση της συνολικής επιβράβευσής του από το περιβάλλον. Συνεπώς, οι σχεδιαστές των μοντέλων ενισχυτικής μάθησης δίνουν έμφαση στον σχεδιασμό των τεχνικών εκείνων που θα επιτρέψουν στον agent να πετύχει τον στόχο του βάζοντας σε δεύτερη προτεραιότητα τον σχεδιασμό του μηχανισμού επιβράβευσης.

Στην υποθετική περίπτωση που ένας χειριστής – άνθρωπος είναι αυτός που ανταποκρίνεται στις ενέργειες ενός agent επιβραβεύοντας ή αποθαρρύνοντάς τον τότε η εκπαίδευση του agent μέσω της ενισχυτικής μάθησης δημιουργεί έναν τέλειο agent. Στην πραγματικότητα όμως μία τέτοια λύση είναι ανέφικτη. Έτσι οι σχεδιαστές αναγκάζονται να σχεδιάζουν το

περιβάλλον με τέτοιο τρόπο ώστε να δημιουργεί αξιόπιστα σχήματα επιβράβευσης και στις περισσότερες περιπτώσεις να αποκλείει την ανάγκη ύπαρξης του χειριστή και αν' αυτού να δημιουργούν αυτόματους μηχανισμούς ή αλλιώς 'virtual' περιβάλλοντα. Με τον τρόπο αυτό ενώ δεν αλλάζει ο στόχος του agent, τροποποιείται ο μηχανισμός παραγωγής των επιβραβεύσεων.

Καθώς όμως τα συστήματα ενισχυτικής μάθησης γίνονται όλο και πιο γενικά και αυτόνομα, ο σχεδιασμός του μηχανισμού επιβράβευσης γίνεται όλο και πιο σημαντικός και δύσκολος. Επομένως οι σχεδιαστές πρέπει να έχουν υπόψη τους ότι δεν αρκεί να σχεδιάσουν ένα σύστημα ενισχυτικής μάθησης που επιλύει ένα πρόβλημα, θα πρέπει επιπλέον να σχεδιάσουν τεχνικές που θα διασφαλίζουν ότι το περιβάλλον με συνέπεια αναγνωρίζει και επιβραβεύει την επιθυμητή συμπεριφορά του agent και ότι ο μηχανισμός επιβράβευσης δεν καταστρατηγείται ούτε παρακάμπτεται.

Μία προσέγγιση [8][9] στο ζήτημα αυτό είναι η υπόθεση ότι αντί για τον χειριστή – άνθρωπο, ένας άλλος software agent (Η) με ιδιαίτερα χαρακτηριστικά είναι αυτός που αλληλοεπιδρά με τον agent (Α) επιβραβεύοντας ή αποθαρρύνοντάς τον. Τα ιδιαίτερα χαρακτηριστικά του (Η) agent είναι: α) είναι 'πιο' έξυπνος από τον Α, και β) παίρνει 'καλές' αποφάσεις. Έτσι κατά την διάρκεια της εκπαίδευσης ο (Α) εκμαιεύει από τον (Η) τις ΄καλές' αποφάσεις του για μια δεδομένη κατάσταση του (Α). Το ζήτημα είναι αν οι ΄καλές΄ αποφάσεις του (Η) αποτελούν τμήμα της βέλτιστης στρατηγικής για τον (Α). Για να αντιμετωπισθεί το ζήτημα αυτό υπάρχουν τουλάχιστον τρεις προσεγγίσεις:

- **Άμεση επιτήρηση**: Ο (Η) αξιολογεί την συμπεριφορά του (Α) και ο (Α) εκπαιδεύεται έτσι ώστε να μεγιστοποιεί την αξιολόγηση του (Η). Ο (Η) πρέπει να είναι πιο 'έξυπνος΄ από τον (Α) ώστε η αξιολόγησή του να είναι αποδοτική. Ένα πρόβλημα που προκύπτει είναι όταν ο (Α) επιλέγει μια ενέργεια έχοντας σαν βάση ένα γεγονός ενώ ο (Η) δεν έχει πρόσβαση σε αυτό το γεγονός. Τότε ο (Η) θα αξιολογήσει αρνητικά μια 'καλή' ενέργεια του (Α).

- **Μάθηση με απομίμηση (Imitation Learning)**: Ο (Η) καταγράφει την συμπεριφορά του σε αρχεία επίδειξης και ο (Α) εκπαιδεύεται έτσι ώστε να συμπεριφέρεται όπως προδιαγράφεται στα αρχεία επίδειξης. Υπάρχουν πολλές τεχνικές μάθησης με απομίμηση. Μία από αυτές είναι η τεχνική GAIL όπου ένα μοντέλο – διαχωριστής εκπαιδεύεται να ξεχωρίζει την συμπεριφορά του (Α) από την συμπεριφορά του (Η) και ο (Α) επιβραβεύεται όταν ξεγελά το μοντέλο – διαχωριστή. Ένα πρόβλημα που προκύπτει είναι όταν ο (Η) είναι πολύ περισσότερο άξιος από τον (Α) και γενικά ο (Α) αδυνατεί να μιμηθεί την συμπεριφορά του (Η).

- **Inverse reinforcement learning**: Ο (Η) καταγράφει την συμπεριφορά του και από την μελέτη της συμπεριφοράς του εξάγεται η συνάρτηση επιβράβευσης που προσεγγίζει την συμπεριφορά του. Ο (Α) επιβραβεύεται βάση της εξαγόμενης συνάρτησης επιβράβευσης.

2.3 ΜΑΘΗΣΗ ΜΕ ΑΠΟΜΙΜΗΣΗ (IMITATION LEARNING)

Ο στόχος της ενισχυτικής μάθησης είναι ο agent να μάθει την βέλτιστη πολιτική μέσω της οποίας θα μεγιστοποιήσει την συνολική ανταμοιβή του από το περιβάλλον. Στις περισσότερες περιπτώσεις η ενισχυτική μάθηση έχει αποδοτικά αποτελέσματα. Στην περίπτωση όμως που η επιβράβευση από το περιβάλλον επιστρέφει στον agent σε αραιά χρονικά διαστήματα ή ακόμη και σπάνια τότε η τεχνική της ενισχυτικής μάθησης αποτυγχάνει να εκπαιδεύσει καλά μοντέλα. Σε τέτοιες περιπτώσεις το πρόβλημα προσεγγίζεται με τεχνικές μάθησης με απομίμηση.

Στα πλαίσια της μάθησης με απομίμηση αντί ο agent να μαθαίνει μέσω των αραιών επιβραβεύσεων, ένας expert agent (τυπικά ο άνθρωπος αλλά ουσιαστικά ένας software agent) καταγράφει την συμπεριφορά του σε αρχεία επίδειξης (demonstrations). Στην συνέχεια ο agent προσπαθεί να μάθει την βέλτιστη πολιτική με το να μιμείται την συμπεριφορά του expert agent.

Η μάθηση με απομίμηση είναι χρήσιμη στις περιπτώσεις όπου είναι εύκολο για τον expert agent να καταγράψει την συμπεριφορά του σε αρχεία επίδειξης από το να προδιαγραφεί η συνάρτηση επιβράβευσης.

Στα πλαίσια της μάθηση με απομίμηση ορίζεται ένα Markov Decision Process (MDP) που περιλαμβάνει ένα σύνολο καταστάσεων ***S,*** ένα σύνολο ενεργειών **A,** ένα μοντέλο μετάβασης **P(s'|s,a)** που ορίζει την πιθανότητα σε μια κατάσταση **s** η ενέργεια **a** να οδηγήσει στην κατάσταση **s'**) και μια άγνωστη συνάρτηση επιβράβευσης **R(s,a)**. Ο agent εκτελει ενέργειες με βάση την πολιτική **π**. Επιπλέον, ορίζονται και οι καταγραφές επίδειξης του expert **τ = (s0, a0, s1, a1, …) ,** όπου οι ενέργειες βασίζονται στην (βέλτιστη) πολιτική **π*** του expert. Σε μερικές περιπτώσεις υπάρχει η δυνατότητα απευθείας σύνδεσης με τον expert κατά την διάρκεια της εκπαίδευσης που σημαίνει ότι τα μοντέλα μάθησης με απομίμηση μπορούν ανά πάσα στιγμή να απαιτήσουν από τον expert είτε περισσότερες καταγραφές επίδειξης είτε άμεση αξιολόγηση.

2.3.1 Behavioural Cloning

Η πιο απλή μορφή μάθησης με απομίμηση είναι η μέθοδος behavioural cloning. Ο agent δέχεται σαν δεδομένα εκπαίδευσης τις καταστάσεις και τις ενέργειες που έχουν καταγραφεί στο αρχείο επίδειξης και στην συνέχεια χρησιμοποιώντας έναν classifier ή regressor (supervised μάθηση) αναπαράγει την πολιτική του expert agent. Το κύριο πλεονέκτημα αυτής της μεθόδου είναι ότι μπορεί να μιμείται την συμπεριφορά του expert agent χωρίς να χρειάζεται να αλληλοεπιδρά με το περιβάλλον. Υπάρχουν περιπτώσεις στις οποίες η μέθοδος behavioural cloning παράγει αποδοτικούς agents. Υπάρχουν όμως και περιπτώσεις κατά τις οποίες η μέθοδος behavioural cloning μπορεί να οδηγήσει τον agent σε αποτυχημένες επιλογές. Και αυτό γιατί μία λάθος επιλογή της επόμενης κατάστασης μπορεί να οδηγήσει τον agent σε μία κατάσταση που ο expert agent να μην είχε ποτέ οδηγηθεί και επομένως ο agent να μην έχει δεδομένα να εκπαιδευθεί.

2.3.2 Direct Policy Learning (μέσω διαδραστικού expert agent)

Η μέθοδος Direct Policy Learning (DPL) είναι βελτιωμένη έκδοση της μεθόδου behavioural cloning. Προϋποθέτει ότι κατά την διάρκεια της εκπαίδευσης είναι άμεσα διαθέσιμος ο expert agent και μπορεί ο agent να εκμαιεύει αποκρίσεις από αυτόν. Όπως και στο behavioural cloning ο expert agent καταγράφει την συμπεριφορά του σε αρχεία επίδειξης και ο agent εφαρμόζει supervised τεχνικές σε αυτά για να εξάγει την βέλτιστη πολιτική. Στην συνέχεια η πολιτική εφαρμόζεται στο περιβάλλον ενώ ταυτόχρονα ζητείται από τον expert agent να επιβεβαιώσει την εγκυρότητα της πολιτικής. Η απόκριση του expert agent τροφοδοτεί το μοντέλο της supervised μάθησης με νέα δεδομένα εκπαίδευσης. Ο αέναος κύκλος τελειώνει όταν υπάρξει σύγκλιση. Ο αλγόριθμος γίνεται πιο αποδοτικός αν σε κάθε στάδιο της εκπαίδευσης χρησιμοποιούνται όλα τα προηγούμενα δεδομένα εκπαίδευσης με αποτέλεσμα ο agent να 'θυμάται' όλες τις λάθος ενέργειες που είχε κάνει στο παρελθόν,

Η μέθοδος Iterative direct policy learning είναι ιδιαιτέρως αποδοτική μέθοδος μάθησης και δεν υποφέρει από τα προβλήματα του BC. Το βασικό της μειονέκτημα είναι ότι ο expert agent πρέπει να έχει την ικανότητα να αξιολογεί άμεσα τις ενέργειες του agent πράγμα που δεν είναι δυνατό σε πολλές περιπτώσεις.

2.3.3 Inverse Reinforcement Learning

Η μέθοδος Inverse reinforcement learning (IRL) αποτελεί διαφορετική προσέγγιση της μάθησης με απομίμηση. Η βασική ιδέα είναι να εξαχθεί η συνάρτηση επιβράβευσης από τα αρχεία επίδειξης που καταγράφει ο expert agent και στην συνέχεια να βρεθεί η βέλτιστη πολιτική για τον agent με χρήση ενισχυτικής μάθησης. Στο τέλος συγκρίνονται οι πολιτικές του expert agent και του προς εκπαίδευση agent.

Υπάρχουν δύο προσεγγίσεις στην διαδικασία εξαγωγής της συνάρτησης επιβράβευσης. Η πρώτη – model given προσέγγιση – θεωρεί ότι η συνάρτηση επιβράβευσης είναι γραμμική και ότι ο χώρος καταστάσεων είναι μικρός. Η εξαγωγή της συνάρτησης επιβράβευσης είναι ένας κύκλος από προσπάθειες. Σε κάθε προσπάθεια, υπολογίζεται μια προσέγγιση της συνάρτησης επιβράβευσης, εκτελείται μία πλήρης εκπαίδευση με RL και στο τέλος συγκρίνεται η πολιτική του expert agent και με την πολιτική που κατέκτησε ο agent στην διάρκεια της τρέχουσας προσπάθειας.

Η δεύτερη προσέγγιση – model free προσέγγιση – πιο γενική και ως εκ τούτου, θεωρείται ότι η συνάρτηση επιβράβευσης είναι πολύπλοκη και μοντελοποείται με την βοήθεια δικτύου νευρώνων. Επίσης θεωρείται ότι ο χώρος καταστάσεων είναι μεγάλος ή συνεχής. Και πάλι, η εξαγωγή της συνάρτησης επιβράβευσης είναι ένας κύκλος από προσπάθειες. Σε κάθε προσπάθεια, υπολογίζεται μια προσέγγιση της συνάρτησης επιβράβευσης, εκτελείται ένα βήμα της εκπαίδευσης με RL και στο τέλος συγκρίνεται η πολιτική που κατέκτησε ο agent στην διάρκεια της τρέχουσας προσπάθειας με την απόδοση του expert agent.

2.3.4 Generative adversarial imitation learning (GAIL)

Πρόκειται για μια deep neural network αρχιτεκτονική που εφαρμόζεται στη μάθηση με απομίμηση. Ένας agent (ο εκπαιδευόμενος – trainee) μαθαίνει μία δεξιότητα μέσω της παρατήρησης της συμπεριφοράς ενός άλλου agent (ο εκπαιδευτής – trainer).

Η μέθοδος GAIL βασίζεται στην generative adversarial network (GAN) αρχιτεκτονική που περιλαμβάνει δύο δίκτυα των οποίων η συνέργεια είναι καθαρά ανταγωνιστική. Το πρώτο δίκτυο (ο ‘επικριτικός’ - “discriminator” / “critic”) μαθαίνει να ξεχωρίζει τα πραγματικά δεδομένα εκπαίδευσης από τα παραδείγματα ενώ το δεύτερο δίκτυο (ο ‘δημιουργός’ - “generator” / “actor”) μαθαίνει να παράγει πειστικά αλλά πλαστά παραδείγματα με σκοπό να ξεγελάσει τον discriminator. Τα πλαστά παραδείγματα αποτελούν και την χρήσιμη έξοδο ενός GAN.

Όταν ένα GAN χρησιμοποιείται για μάθηση με απομίμηση, η έξοδος του GAN προσδιορίζεται από την έξοδο του discriminator και όχι του generator. Στο GAIL, ο discriminator μαθαίνει να ξεχωρίζει την συμπεριφορά του expert από την πλαστή συμπεριφορά που παράγει ο generator. Ενώ ο generator με την σειρά του προσπαθεί να μιμηθεί τον expert τόσο πιστευτά ώστε να ξεγελάσει τον discriminator και να πιστέψει ότι η συμπεριφορά του generator είναι στην πραγματικότητα συμπεριφορά του expert agent.

Ο discriminator κάθε φορά εξετάζει μία νέα ενέργεια και της αποδίδει μια επιβράβευση ανάλογα με το πόσο ισχυρά πιστεύει ότι αυτή η ενέργεια έχει προέλθει από τον expert agent μέσω του αρχείου επίδειξης.

Σε κάθε στάδιο της εκπαίδευσης, ο trainee agent προσπαθεί να μεγιστοποιήσει αυτή την επιβράβευση. Έτσι σταδιακά ο trainee agent βελτιώνει την ικανότητά του να μιμείται την συμπεριφορά του expert ενώ ο discriminator γίνεται ολοένα και πιο αυστηρός στην κρίση του.

Μέσω της μεθόδου GAIL, ο agent μαθαίνει την βέλτιστη πολιτική η οποία δημιουργεί καταστάσεις και ενέργειες παρόμοιες με αυτές που καταγράφονται στα αρχεία επίδειξης, απαιτώντας πολύ λιγότερα demonstrations από την μέθοδο behavioural cloning.

Επιπρόσθετα, η μέθοδος GAIL μπορεί να συνυπάρξει με μία καθαρά RL μέθοδο βελτιώνοντας την εκπαίδευση τόσο ως προς την απόδοσή της όσο και ως προς την ταχύτητα της.

2.4 BLOCKCHAIN - ETHEREUM

Το blockchain [17] είναι ένα ψηφιακό, κατανεμημένο, δημόσιο καθολικό (ledger), μέσω του οποίου καταγράφονται με ασφάλεια συναλλαγές, συμφωνίες, συμβόλαια και γενικώς οτιδήποτε θεωρείται ότι έχει αξία, χρειάζεται να έχει καταγραφεί και να είναι διαθέσιμο προς επαλήθευση.

Το blockchain λειτουργεί στο διαδίκτυο, πάνω σε ένα δίκτυο ομότιμων (P2P) κόμβων που διατηρούν ο κάθε ένας τους ένα πιστό αντίγραφο του καθολικού των συναλλαγών / δεδομένων, επιτρέποντας την πραγματοποίηση συναλλαγών χωρίς την παρουσία κάποιου έμπιστου ενδιάμεσου, παρά μόνο με την βοήθεια των υπολοίπων κόμβων του δικτύου.

Το blockchain είναι μία ακολουθία από blocks, που καταγράφουν την πλήρη λίστα των συναλλαγών που έχουν πραγματοποιηθεί όπως ακριβώς τα συμβατικά δημόσια καθολικά. Κάθε block δείχνει στο αμέσως προηγούμενο του block ενώ το πρώτο block του blockchain ονομάζεται genesis block. Κάθε block περιλαμβάνει την κεφαλή και το σώμα του. Στην κεφαλή αποθηκεύονται πληροφορίες που αφορούν το τρέχον block ενώ το σώμα του περιλαμβάνει τις συναλλαγές. Μόνο ένα μπλοκ μπορεί να προστεθεί κάθε φορά στο blockchain και κάθε μπλοκ περιέχει την απόδειξη εργασίας (Proof of Work) που προσθέτουν οι κόμβοι miners. Το blockchain χρησιμοποιεί μηχανισμούς κρυπτογράφησης (ψηφιακή υπογραφή) για την επικύρωση της αυθεντικότητας των συναλλαγών.

Κάθε χρήστης του blockchain διατηρεί ένα ζεύγος κλειδιών, το ιδιωτικό και το δημόσιο κλειδί. Το ιδιωτικό κλειδί χρησιμοποιείται για την υπογραφή των συναλλαγών. Οι ψηφιακά υπογεγραμμένες συναλλαγές μεταδίδονται παντού στο δίκτυο και γίνονται αποδεκτές μέσω του δημόσιου κλειδιού που είναι διαθέσιμο σε όλους.

Το blockchain χαρακτηρίζεται από τα εξής:

- Αποκέντρωση (Decentralisation). Στα συμβατικά κατανεμημένα συστήματα, η κάθε συναλλαγή επικυρώνεται από ένα κεντρικό ‘οργανισμό’ που τον εμπιστεύονται όλοι. Αντίθετα στο blockchain μια συναλλαγή μπορεί να γίνει μεταξύ δύο ομότιμων κόμβων χωρίς την ανάγκη της επικύρωσης από έναν κεντρικό ‘οργανισμό’. Με τον τρόπο αυτό το blockchain μειώνει τα κόστη εξυπηρέτησης και αντιμετωπίζει κάθε πιθανή συμφόρηση στον κεντρικό ‘οργανισμό’.
- Διατηρησιμότητα (Persistency). Το γεγονός ότι κάθε συναλλαγή μεταδίδεται σε όλο το δίκτυο, επιβεβαιώνεται και καταγράφεται σε blocks που επίσης είναι κατανεμημένα σε όλο το δίκτυο καθιστά αδύνατη την αλλοίωση / τροποποίηση της συναλλαγής. Επιπλέον, καθώς κάθε block επίσης μεταδίδεται σε όλους τους κόμβους, επικυρώνεται από αυτούς και ελέγχονται οι συναλλαγές που περιέχει καθιστά κάθε προσπάθεια πλαστογράφησης εύκολα ανιχνεύσιμη.
- Ανωνυμία (Anonymity). Κάθε χρήστης ενεργεί μέσα στο blockchain network χρησιμοποιώντας μία ή περισσότερες διευθύνσεις. Πουθενά στο blockchain δεν καταγράφονται πληροφορίες σχετικές με την ιδιωτικότητα του χρήστη
- Ελεγξιμότητα (Auditability). Καθώς κάθε συναλλαγή επικυρώνεται και καταγράφεται στο blockchain με ένα timestamp, οι χρήστες μπορούν εύκολα να επικυρώσουν και να ανιχνεύσουν προηγούμενες συναλλαγές

2.4.1 Ethereum

Το Ethereum είναι μια αποκεντρωμένη, open source πλατφόρμα που επιτρέπει την δημιουργία έξυπνων συμβολαίων και αποκεντρωμένων εφαρμογών (dApps), είναι δηλαδή ένα προγραμματιζόμενο blockchain.

Ένα έξυπνο συμβόλαιο (smart contract) είναι μία συμφωνία μεταξύ δύο συμβαλλομένων που εξασφαλίζει συγκεκριμένους όρους και κανόνες. Είναι κώδικας που εφαρμόζει τους όρους και τους κανόνες, είναι αποθηκευμένος στο Ethereum blockchain, εκτελείται αυτόματα όταν συγκεκριμένες συνθήκες ικανοποιηθούν και η συναλλαγή που υλοποιεί είναι πλήρως ανιχνεύσιμη και μη αναστρέψιμη.

Οι αποκεντρωμένες εφαρμογές δεν αναφέρονται σε ένα κεντρικό σύστημα, αλλά συνεργάζονται απευθείας με το Ethereum blockchain.

Το Ethereum χρησιμοποιεί ένα πρωτόκολλο δικτύου ομότιμων κόμβων (P2P) το οποίο είναι και υπεύθυνο για τον συντονισμό των διασυνδεδεμένων κόμβων, με σκοπό την απρόσκοπτη λειτουργία του δικτύου. Κάθε κόμβος «τρέχει» ένα EVM (Ethereum Virtual Machine) που είναι υπεύθυνο να εκτελεί τον κώδικα των έξυπνων συμβολαίων και όλοι οι κόμβοι εκτελούν τις ίδιες ακριβώς εντολές. Έτσι οι υπολογισμοί είναι πολύ αργοί και ακριβοί αλλά επιτυγχάνεται η πολύτιμη ομοφωνία / συναίνεση (consensus). Αυτό το χαρακτηριστικό δίνει στο Ethereum πολύ μεγάλη αντοχή σε σφάλματα, αδιάλειπτη λειτουργία και εγγυάται ότι τα δεδομένα μέσα στο blockchain θα παραμείνουν αμετάβλητα.

Το Ethereum περιέχει λογαριασμούς (account). Τόσο η κατάσταση κάθε λογαριασμού όσο και οι συναλλαγές μεταξύ των λογαριασμών καταγράφονται στο Ethereum blockchain. Υπάρχουν δύο είδη λογαριασμών: α) οι λογαριασμοί των χρηστών που ελέγχονται από τα ιδιωτικά τους κλειδιά και β) οι λογαριασμοί συμβολαίων, που ελέγχονται από τον κώδικα του συμβολαίου και μπορούν να ενεργοποιηθούν μόνο από έναν λογαριασμό χρήστη.

Όταν εκτελείται μία συναλλαγή προς έναν λογαριασμό συμβολαίου τότε στην πραγματικότητα εκτελείται ο κώδικας εκείνου του συμβολαίου που είναι συνδεδεμένος με τον εν λόγω λογαριασμό. Οι χρήστες που έχουν Ethereum λογαριασμό μπορούν να δημιουργήσουν νέα συμβόλαια, δημοσιεύοντάς τα στο blockchain.

Για κάθε συναλλαγή πληρώνεται ένα μικρό τέλος στο δίκτυο (gas). Αυτό προστατεύει το blockchain από επιπόλαιες συναλλαγές αλλά και από κακόβουλες επιθέσεις. Το gas πληρώνεται στο κρυπτονόμισμα του Ethereum, Ether. Τα τέλη εισπράττονται από τους κόμβους που επικυρώνουν το δίκτυο, τους miners. Οι miners είναι κόμβοι του δικτύου που λαμβάνουν, διαδίδουν, επικυρώνουν και εκτελούν συναλλαγές. Συγκεντρώνουν έναν συγκεκριμένο αριθμό συναλλαγών σε ένα block και στη συνέχεια ανταγωνίζονται μεταξύ τους για το ποιος θα καταφέρει γρηγορότερα να εισάγει το νέο block στο blockchain. Ο γρηγορότερος miner επιβραβεύεται με τα gas που αντιστοιχούν στις συναλλαγές του block.

Για να εισαχθεί ένα μπλοκ στο blockchain θα πρέπει να συνοδεύεται από την απόδειξη εργασίας (nonce) που αποτελεί την λύση ενός δύσκολου μαθηματικού προβλήματος. Στην περίπτωση του Ethereum η επίλυση του nonce έχει μεγάλες απαιτήσεις σε μνήμη και το «mining» είναι μια αργή και κοστοβόρα διαδικασία που οδηγεί όμως σε μεγαλύτερη ασφάλεια.

2.5 IPFS

Η διαδικασία mining στο Ethereum είναι μία διαδικασία αργή, απαιτητική και κοστοβόρα και για αυτό τον λόγο το Ethereum blockchain δεν ενδείκνυται για την αποθήκευση μεγάλου όγκου δεδομένων όπως είναι τα αρχεία επίδειξης της συμπεριφοράς ενός RL agent.

Μία λύση για την επίλυση αυτού του προβλήματος είναι να αποθηκεύονται τα δεδομένα εκτός blockchain όπως για παράδειγμα σε κατανεμημένα συστήματα αρχείων. Τέτοια συστήματα είναι το IPFS [18] και το Swarm [19].

Για τις ανάγκες της εργασίας χρησιμοποιήθηκε το IPFS.

Το IPFS είναι ένα δίκτυο ομότιμων κόμβων peer-to-peer (p2p) για την αποθήκευση δεδομένων. Τα έγγραφα είναι διαθέσιμα μέσω των ομότιμων κόμβων, είναι αποθηκευμένα διάσπαρτα σε όλο τον κόσμο και αναγνωρίζονται με βάση το περιεχόμενό τους.

Κάθε έγγραφο που αποθηκεύεται στο IPFS έχει ένα μοναδικό content identifier (CID) που δεν είναι άλλο από το hash code των περιεχομένων του. Ο μοναδικός αυτός content identifier hash code επιστρέφεται στο ιδιοκτήτη του εγγράφου μετά την επιτυχημένη αποθήκευσή του. Αν ακριβώς το ίδιο έγγραφο προστεθεί στο IPFS ακριβώς το ίδιο hash code θα δημιουργηθεί.

Για την ανάκτηση ενός εγγράφου από το IPFS δίδεται στο IPFS το hashcode που του αντιστοιχεί. Το IPFS εγγυάται ότι θα επιστραφεί ακριβώς το ίδιο έγγραφο το οποίο αντιστοιχεί στο hashcode.

Με άλλα λόγια, το IPFS επιτρέπει την αποθήκευση δεδομένων και την ανάκτησή τους με την εγγύηση ότι τα δεδομένα δεν θα αλλοιωθούν στο μεταξύ.

Το μέγεθος του hashcode που επιστρέφει το IPFS είναι 46 bytes και επομένως είναι πολύ φθηνότερο να αποθηκεύεται αυτό στο Ethereum αντί του ίδιου του αρχείου.

3 ΕΡΓΑΛΕΙΑ ΚΑΙ ΤΕΧΝΟΛΟΓΙΕΣ

3.1 REINFORCEMENT LEARNING

Καθώς τα τελευταία χρόνια παρατηρείται μία ραγδαία ανάπτυξη των αλγορίθμων ενισχυτικής μάθησης, έχουν αναπτυχθεί και μία σειρά από εργαλεία που υποβοηθούν την τάση αυτή. Στα πλαίσια της εργασίας, φάνηκε ότι είναι δύσκολο να επιλέξει κανείς την ‘καλύτερη’ βιβλιοθήκη κυρίως λόγω της έλλειψης συγκριτικών μελετών. Στο άρθρο ‘On choosing a Deep Reinforcement Learning Library’ (T. Simonini, 05.2019) [19] παρουσιάζονται και συγκρίνονται πέντε (5) βιβλιοθήκες που υλοποιούν αλγόριθμους ενισχυτικής μάθησης: TF-Agents [27], OpenAI Baselines [25], Stable Baselines [26], TensorForce [23] και KerasRL [22]. Επιπλέον των (5) αυτών βιβλιοθηκών, στα πλαίσια της εργασίας μελετήθηκε και η βιβλιοθήκη ML-Agents [21]. Αρχικά μελετήθηκε ποιες από τις παραπάνω (6) βιβλιοθήκες υποστηρίζουν εκτός των RL αλγορίθμων και αλγόριθμους μάθησης με απομίμηση (Imitation Learning). Ο πίνακας που ακολουθεί παρουσιάζει τα αποτελέσματα:

Library	GAIL	BC	PPO	Github stars (in K) As of 06.2020	Tensorboard[50]
KerasRL	x	x	x	4.7	x
Tensorforce		x	ϒ	2.7	ϒ
OpenAI Baselines	ϒ	partially	ϒ	10	ϒ
Stable Baselines	ϒ	x	ϒ	2.2	ϒ
TF-Agents	x	x	ϒ	1.4	ϒ
ML-Agents	ϒ	ϒ	ϒ	9	ϒ

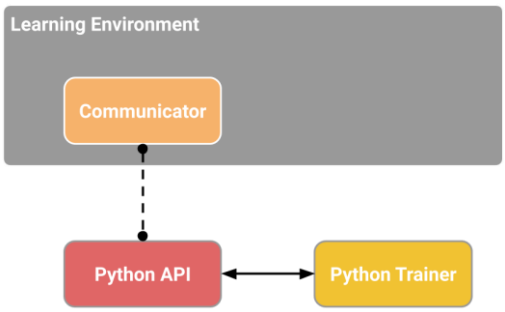
Πίνακας 1 - ML βιβλιοθήκες που υποστηρίζουν Ενισχυτική Μάθηση & Μάθηση με απομίμηση

Από τα αποτελέσματα του πίνακα φαίνεται ότι μόνο οι OpenAI baselines και ML-Agents βιβλιοθήκες υποστηρίζουν και αλγόριθμους μάθησης με απομίμηση. Επιπλέον παρατηρήθηκε ότι η ML-Agents βιβλιοθήκη μπορεί να συνδεθεί με το Unity όπου σχετικά εύκολα μπορεί να σχεδιασθεί και να υλοποιηθεί τόσο ο RL agent όσο και το περιβάλλον με το οποίο αυτός αλληλοεπιδρά. Επιπλέον, η ML-Agents βιβλιοθήκη είναι κατασκευασμένη με την χρήση του TensorFlow [24] της πιο διαδεδομένης ίσως open source πλατφόρμας για μηχανική μάθηση. Έτσι, προτιμήθηκε η χρήση της ML-Agents βιβλιοθήκης.

3.1.1 ML-Agents

Το Unity Machine Learning Agents Toolkit (ML-Agents Toolkit) είναι μία open-source πλατφόρμα εκπαίδευσης ‘έξυπνων’ agents. Οι agents εκπαιδεύονται κάνοντας χρήση τεχνικών μηχανικής μάθησης όπως reinforcement learning, imitation learning, neuroevolution και άλλων, μέσω ενός απλού Python API (που είναι κατασκευασμένο για να τρέχει πάνω σε TensorFlow).

Η πλατφόρμα ML-Agents περιλαμβάνει (5) βασικά τμήματα:



Εικόνα 2 – Αρχιτεκτονική της ML-agents πλατφόρμας (Πηγή:ML-Agents Toolkit overview. 2020. Available Online - https://github.com/Unity-Technologies/ml-agents/blob/release_2_docs/docs/ML-Agents-Overview.md

- Περιβάλλον εκπαίδευσης (Learning Environment) – Το περιβάλλον εκπαίδευσης είναι μέρος του Unity και πιο συγκεκριμένα είναι μια σκηνή Unity που περιλαμβάνει όλα τα απαραίτητα components για την εκπαίδευση του agent. Μέσω του περιβάλλοντος, οι agents παρατηρούν, δρουν και μαθαίνουν. Επιπλέον μέσω της Unity σκηνής δίδεται η δυνατότητα να σχεδιασθεί και να υλοποιηθεί ο ίδιος ο agent αρκεί να ορισθεί τόσο ο ίδιος (Agent) όσο και η συμπεριφορά του (Behavior).
- Python Trainers: περιλαμβάνει τους αλγόριθμους μηχανικής μάθησης και υλοποιεί την εκπαίδευση του agent. Έχει υλοποιηθεί ως ένα Python πακέτο (mlagents) που υποστηρίζεται από ένα command-line utility mlagents-learn που επιτρέπει την διαχείριση της εκπαίδευσης. Το τμήμα Python Trainers επικοινωνεί με το περιβάλλον εκπαίδευσης μόνο μέσω του Python Low-Level API.

- Python Low-Level API **-** είναι ένα low-level Python interface που υλοποιεί την επικοινωνία μεταξύ των Python Trainers και περιβάλλοντος. Δεν είναι μέρος του Unity αλλά επικοινωνεί με αυτό μέσω του External Communicator.
- External Communicator – συνδέει το Learning Environment με το Python Low-Level API. Είναι ενσωματωμένο στο Learning Environment.
- Gym Wrapper (δεν εμφανίζεται στην Εικόνα 2). Πρόκειται για έναν wrapper που επιτρέπει την χρήση από την πλατφόρμα ML-Agents αλγορίθμων μηχανικής μάθησης που έχουν ως περιβάλλον το gym [28] της OpenAI

Κάθε περιβάλλον εκπαίδευσης ορίζει για κάθε χαρακτήρα που υπάρχει στην σκηνή έναν Agent. Κάθε Agent είναι συνδεδεμένος με ένα Behaviour το οποίο μπορεί να θεωρηθεί ως μια συνάρτηση που δέχεται παρατηρήσεις που αφορούν το περιβάλλον (observations) και επιβραβεύσεις και επιστρέφει ενέργειες. Υπάρχουν τρία είδη Behaviour: Learning, Heuristic και Inference. Η Learning Behavior είναι αυτή που πρόκειται να εκπαιδευτεί. Η Heuristic Behavior είναι αυτή που περιγράφεται είτε μέσω κώδικα είτε προσδιορίζεται από τον χρήστη. Η Inference Behavior είναι αυτή που έχει ήδη εκπαιδευτεί και επομένως υπάρχει Neural Network μοντέλο που έχει συσχετισθεί με αυτή.

Είναι φανερό ότι στα πλαίσια ενός περιβάλλοντος μπορεί να υπάρχουν όλοι οι δυνατοί συνδυασμοί. Έτσι μπορεί να συνυπάρχουν πολλοί agents με την ίδια συμπεριφορά, agents που έχουν ήδη εκπαιδευτεί και agents που είναι προς εκπαίδευση (διαφορετικοί μεταξύ τους). Επομένως, διάφορα σενάρια εκπαίδευσης είναι πιθανά, όπως:

- Single-Agent. Πρόκειται για το παραδοσιακό σενάριο εκπαίδευσης όπου εκπαιδεύεται ένας μόνο agent με τις δικές του επιβραβεύσεις και την δική του συμπεριφορά.
- Simultaneous Single-Agent. Πολλοί ανεξάρτητοι μεταξύ τους agents. Ο κάθε ένας με τις δικές του επιβραβεύσεις, όλοι όμως με την ίδια συμπεριφορά. Πρόκειται για παράλληλη εκπαίδευση των agents που επιταχύνει και σταθεροποιεί την εκπαιδευτική διαδικασία.
- Adversarial Self-Play. Δύο agents που αλληλοεπιδρούν και έχουν αντίστροφες επιβραβεύσεις. Σε ένα σενάριο ενός παιχνιδιού με δύο παίκτες η adversarial self-play εκπαίδευση παράγει ιδιαίτερα επιδέξιους agents.
- Cooperative Multi-Agent. Πολλοί agents που αλληλοεπιδρούν μεταξύ τους και διαμοιράζονται τις επιβραβεύσεις με την ίδια ή διαφορετική συμπεριφορά. Σε αυτό το σενάριο, οι agents πρέπει να συνεργαστούν για την επίτευξη ενός κοινού στόχου.
- Competitive Multi-Agent. Πολλοί agents που αλληλοεπιδρούν μεταξύ τους με αντίστροφες επιβραβεύσεις και με την ίδια ή διαφορετική συμπεριφορά. Σε αυτό το σενάριο, οι agents ανταγωνίζονται μεταξύ τους είτε για την νίκη είτε για την κατάκτηση κάποιου πόρου
- Ecosystem. Πολλοί agents που αλληλοεπιδρούν μεταξύ τους με ανεξάρτητες επιβραβεύσεις και με την ίδια ή διαφορετική συμπεριφορά.

Οι αλγόριθμοι μηχανικής μάθησης που υποστηρίζονται είναι:

- Proximal Policy Optimization (PPO) [55]
- Soft Actor-Critic (SAC) [56]
- Curiosity for Sparce-Reward environments [57]
- GAIL [13]
- Behavioural Cloning [11]
- Adversarial Self-Play [57]
- Curriculum Learning [29]

3.1.2 Unity

Το Unity [30] είναι μία cross-platform μηχανή ανάπτυξης παιχνιδιών που δίνει την δυνατότητα ανάπτυξης παιχνιδιών 2D , 3D, εικονικής πραγματικότητας (virtual reality), επαυξημένης πραγματικότητας (augmented reality) και simulations. Η υποστηριζόμενη γλώσσα προγραμματισμού είναι η C#.

3.1.3 TensorBoard

Πρόκειται για το εργαλείο απεικόνισης που προσφέρει η TensorFlow. Το TensorBoard [50] προσφέρει όλα εκείνα τα μέσα απεικόνισης που είναι απαραίτητα κατά την διάρκεια της εκπαίδευσης των μοντέλων μηχανικής μάθησης

3.1.4 Περιβάλλον Anaconda

Το περιβάλλον Anaconda [31] είναι μία δωρεάν open source πλατφόρμα εργαλείων Python για χρήση κυρίως σε εφαρμογές με επιστημονικούς υπολογισμούς (π.χ, data science, machine learning, large scale data processing κλπ).

Για τις ανάγκες της εργασίας χρησιμοποιήθηκε μόνο το εργαλείο Anaconda Prompt

3.2 ETHEREUM-BLOCKCHAIN

3.2.1 Truffle

Το Truffle [37] είναι ένα περιβάλλον ανάπτυξης, ελέγχου και δημοσίευσης έξυπνων συμβολαίων στο Ethereum blockchain μέσω του Ethereum Virtual Machine (EVM). Πιο συγκεκριμένα, το Truffle επιτρέπει:

- Μετάφραση, δημοσίευση και διαχείριση των έξυπνων συμβολαίων
- Αυτόματη διαδικασία ελέγχου των έξυπνων συμβολαίων για γρήγορη και αξιόπιστη δημοσίευση
- Δημιουργία σεναρίων δημοσίευσης έξυπνων συμβολαίων
- Δυνατότητα δημοσίευσης συμβολαίων τόσο σε δημόσια όσο και σε ιδιωτικά blockchains
- Διαδραστική κονσόλα για την άμεση επικοινωνία με τα δημοσιευμένα συμβόλαια

3.2.2 MetaMask

Το Metamask [51] είναι ένα browser plugin που παρέχει στον χρήστη την δυνατότητα να διαχειρίζεται τους Ethereum λογαριασμούς / διευθύνσεις του, έτσι ώστε να αλληλοεπιδρά με dApp εφαρμογές, να στέλνει ή και να υπογράφει συναλλαγές και να υπογράφει δεδομένα.

3.2.3 Solidity

Η Solidity [36] είναι μία αντικειμενοστραφής, υψηλού επιπέδου γλώσσα προγραμματισμού για την υλοποίηση έξυπνων συμβολαίων (smart contracts). Τα έξυπνα συμβόλαια είναι προγράμματα που διαχειρίζονται την συμπεριφορά των λογαριασμών (accounts) μέσα στο Ethereum blockchain.

Πρόκειται για μια γλώσσα προγραμματισμού που είναι επηρεασμένη από τις C++, Python και JavaScript και που έχει σχεδιασθεί ώστε να εκτελείται στο Ethereum Virtual Machine (EVM).

3.2.4 Ganache

Το Ganache [38] είναι ένα ιδιωτικό Ethereum blockchain για την γρήγορη ανάπτυξη και έλεγχο των έξυπνων συμβολαίων. Επιπλέον δίνει την δυνατότητα ανάπτυξης, δημοσίευσης και ελέγχου dApp εφαρμογών σε ένα testing περιβάλλον πριν αυτές δημοσιευθούν σε πραγματικό Ethereum blockchain.

Αποτελεί μέρος της Truffle σουίτας εργαλείων και υποστηρίζεται από το Ganache UI (desktop εφαρμογή) και από το ganache-cli (command line tool)

3.2.5 Βιβλιοθήκη Web3.js

Η web3.js [43] είναι μία JavaScript βιβλιοθήκη που επιτρέπει την αλληλεπίδραση με ένα Ethereum blockchain μέσω RPC calls. Με την web3.js, μία dApp εφαρμογή μπορεί να διαχειρίζεται τα έξυπνα συμβόλαια, να εκτελεί συναλλαγές, να ζητά υπόλοιπα λογαριασμών, κ.α.

3.2.6 Βιβλιοθήκη Web3.py

Η web3.py [42] είναι μία Python βιβλιοθήκη που επιτρέπει την αλληλεπίδραση με ένα Ethereum blockchain μέσω RPC calls κατά αναλογία με την αντίστοιχη web3.js JavaScript βιβλιοθήκη

3.3 INFURA IPFS

Για τις ανάγκες της εργασίας χρησιμοποιήθηκε το Infura IPFS καθώς η Infura προσφέρει δωρεάν έναν IPFS κόμβο μόνο για τις ανάγκες της υλοποίησης και ελέγχου των dApps εφαρμογών. Επιπλέον, προσφέρει και ένα ασφαλες, αξιόπιστο και εύκολο στην χρήση του IPFS API [32] μέσω του οποίου οι dApp εφαρμογές μπορούν και αλληλοεπιδρούν με τον IPFS κόμβο.

3.4 ΑΝΑΠΤΥΞΗ ΕΞΥΠΗΡΕΤΗΤΩΝ

3.4.1 node.js

Το Node.js [54]είναι ένα open-source και cross-platform περιβάλλον εκτέλεσης JavaScript το οποίο εκτελεί κώδικα JavaScript code εκτός του web browser. Το Node.js επιτρέπει την συγγραφή server-side JavaScript προγραμμάτων.

3.4.2 Flask

Το Flask [44] είναι ένα Python web framework, ελαφρύ γιατί δεν προαπαιτεί συγκεκριμένα εργαλεία ή βιβλιοθήκες αλλά παράλληλα εύκολα επεκτάσιμο. Δεν περιλαμβάνει λειτουργικότητα (όπως form validation) που μπορεί εύκολα να ενσωματωθεί με την χρήση άλλων βιβλιοθηκών που εξειδικεύονται σε αυτή.

3.4.3 python

Η python [53] είναι μια γενικού σκοπού, υψηλού επιπέδου, αντικειμενοστραφής γλώσσα προγραμματισμού. Υποστηρίζεται από διερμηνέα (interpreter) και όχι από μεταφραστή (compiler) όπως άλλες επίσης υψηλού επιπέδου γλώσσες προγραμματισμού.

3.5 ΑΝΑΠΤΥΞΗ WEB ΕΦΑΡΜΟΓΩΝ

3.5.1 JavaScript

Η JavaScript ή JS [45] είναι μία γενικού σκοπού, υψηλού επιπέδου, αντικειμενοστραφής γλώσσα προγραμματισμού. Όταν συνδυάζεται με HTML και CSS αποτελεί την βασική τεχνολογία με την οποία αναπτύσσονται οι web εφαρμογές. Η JavaScript επιτρέπει την δημιουργία διαδραστικών ιστοσελίδων. Η πλειονότητα των web εφαρμογών την χρησιμοποιούν τουλάχιστον στο client-side τμήμα τους ενώ η πλειονότητα των web browsers έχουν μια JavaScript μηχανή να τους εξυπηρετεί.

3.5.2 Bootstrap

Το Bootstrap [46] είναι ένα δωρεάν, open source CSS framework. Περιλαμβάνει σχεδιαστικά πρότυπα για διάφορα UI components όπως φόρμες, buttons κ.α. γραμμένα κυρίως σε CSS και δευτερεύοντος σε JavaScript

3.5.3 Lite-server

Ο Lite-server [47] είναι ένας ελαφρύς εξυπηρετητής web εφαρμογών, ο οποίος τις ανοίγει στον browser, τις ανανεώνει όταν υπάρχουν αλλαγές σε HTML ή JavaScript και προσθέτει σε αυτές πιθανές CSS αλλαγές. Ο Lite-server είναι γραμμένος σε node.js και χρησιμοποιείται αποκλειστικά και μόνο κατά την ανάπτυξη των web εφαρμογών.

3.6 INTEGRATED DEVELOPER ENVIRONMENT – VISUAL STUDIO CODE

Το Visual Studio Code [52] είναι ένας δωρεάν editor που έχει κατασκευαστεί από την Microsoft για Windows, Linux και macOS. Προσφέρει υποστήριξη για debugging, υπογράμμιση του κειμένου, έξυπνη συμπλήρωση του κώδικα, αναδόμηση του κώδικα και ενσωμάτωση με Git. Οι χρήστες μπορούν να αλλάξουν την εμφάνιση του editor, να χρησιμοποιήσουν keyboard shortcuts και να εγκαταστήσουν επεκτάσεις και άλλες συνεργαζόμενες εφαρμογές.

Στα πλαίσια της εργασίας στο Visual Studio Code χρησιμοποιήθηκε και η επέκταση για την υποστήριξη της γλώσσας Solidity

4 ΑΝΑΛΥΣΗ ΑΠΑΙΤΗΣΕΩΝ ΣΥΣΤΗΜΑΤΟΣ

Σκοπός του συστήματος

Σκοπός της εργασίας είναι η ανάπτυξη μίας αποκεντρωμένης εφαρμογής (DApp – Decentralized Application) μέσω της οποίας πραγματοποιούνται αγοραπωλησίες αρχείων επίδειξης εκτέλεσης μιας εργασίας (demonstrations) μεταξύ εκπαιδευτών και εκπαιδευόμενων . Ο εκπαιδευτής καταχωρεί τα αρχεία επίδειξης στο Ethereum blockchain ενώ ο εκπαιδευόμενος έχει την δυνατότητα να χρησιμοποιεί τα αρχεία εκπαίδευσης έναντι αμοιβής ώστε να προχωρεί στην εκπαίδευση του.

Χρήστες της εφαρμογής

Εκπαιδευτής (trainer agent)

Πρόκειται για ένα εκπαιδευμένο σε μια εργασία μοντέλο μηχανικής μάθησης (ML Trainer Agent) το οποίο καταγράφει την συμπεριφορά του κατά την διάρκεια εκτέλεσης της εργασίας σε αρχεία επίδειξης. Τα αρχεία αυτά τα καταχωρεί σε ένα έξυπνο συμβόλαιο του Ethereum blockchain με σκοπό να τα κάνει διαθέσιμα έναντι αμοιβής σε μοντέλα μηχανικής μάθησης που θέλουν να εκπαιδευτούν στην συγκεκριμένη εργασία.

Εκπαιδευόμενος (trainee agent)

Πρόκειται για ένα μοντέλο μηχανικής μάθησης που θέλει να εκπαιδευτεί εξ αρχής ή να βελτιώσει την εκπαίδευσή του σε μια συγκεκριμένη εργασία (ML Trainee Agent). Το μοντέλο αυτό έχει την δυνατότητα να ‘αγοράζει’ αρχεία επίδειξης εκτέλεσης της εργασίας μέσω του έξυπνου συμβολαίου του Ethereum blockchain και στην συνέχεια να χρησιμοποιεί τα αρχεία αυτά για την εκπαίδευσή του.

Παραδοχές - Απλοποιήσεις

Οι παραδοχές που έχουν γίνει για την μείωση της πολυπλοκότητας του συστήματος είναι:

- Η καταγραφή της συμπεριφοράς του trainer agent κατά την διάρκεια εκτέλεσης μιας εργασίας σε αρχεία επίδειξης μέσω της πλατφόρμας Unity δεν κατέστη τεχνικά να ενσωματωθεί *αυτόματα* στην εφαρμογή. Σας αποτέλεσμα, θεωρείται ότι ο trainer agent έχει εκ των προτέρων καταγράψει τα demos προς διάθεση και τα έχει αποθηκευμένα προς άμεση χρήση
- Το Ethereum blockchain προσφέρει ασφάλεια στις συναλλαγές καθώς οι αλγόριθμοι κρυπτογράφησης που χρησιμοποιούνται είναι ασφαλείς. Δεν απαιτείται επομένως επιπρόσθετη μέριμνα για την ασφάλεια των συναλλαγών εκ μέρους της εφαρμογής

4.1 Λειτουργικες Απαιτησεις

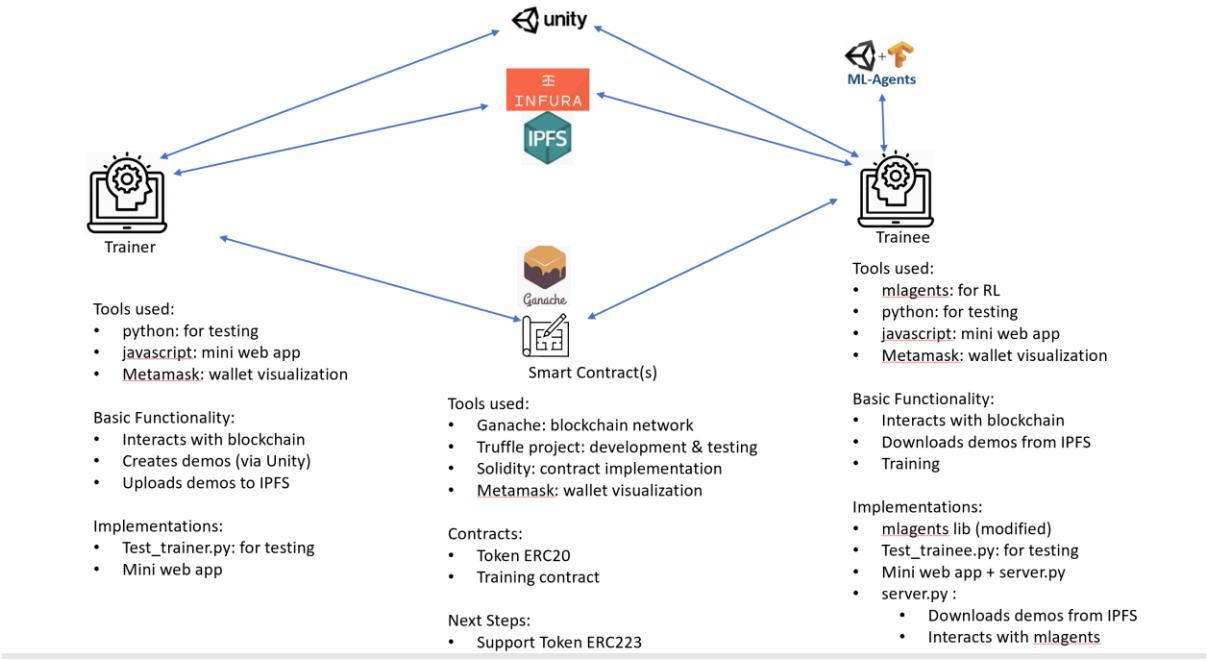
- Η εφαρμογή να επικοινωνεί με το Ethereum blockchain
- Δημιουργία του Ntua Token που θα είναι πλήρως συμβατό με το ERC20 token standard
- Τα μοντέλα μηχανικής μάθησης του εκπαιδευτή και του εκπαιδευόμενου να είναι συμβατά με ποικίλες τεχνικές μάθησης μέσω ενθάρρυνσης (reinforcement learning)
- Η καταγραφή της συμπεριφοράς του trainer agent κατά την διάρκεια εκτέλεσης μιας εργασίας σε αρχεία επίδειξης (demo) μέσω της πλατφόρμας Unity να ενεργοποιείται *αυτόματα* μέσω της εφαρμογής *[Η απαίτηση δεν ικανοποιήθηκε]*
- Η έναρξη της εκπαίδευσης του εκπαιδευόμενου μοντέλου (trainee agent) να πραγματοποιείται αυτόματα και αμέσως μετά την λήψη του αρχείου επίδειξης (demo)
- Τα αρχεία επίδειξης να μην αποθηκεύονται στο Ethereum blockchain αλλά σε ένα κατανεμημένο σύστημα αρχείων. Στο Ethereum blockchain να αποθηκεύεται μόνο το ίχνος (hashcode) του αρχείου στο κατανεμημένο σύστημα αρχείων
- Η επικοινωνία του εκπαιδευόμενου μοντέλου με την πλατφόρμα ML-Agents να είναι αυτόματη με την μικρότερη δυνατή παρέμβαση του χρήστη του εκπαιδευόμενου μοντέλου.
- Η διαπίστευση των χρηστών της εφαρμογής να γίνεται μέσω των εργαλείων που προσφέρονται από το Ethereum blockchain
- Κάθε χρήστης της εφαρμογής να διαθέτει μία διεύθυνση στο Ethereum blockchain της οποίας το ιδιωτικό κλειδί να μην ζητείται σε καμία λειτουργία της εφαρμογής
- Να είναι διαθέσιμο σε Ntua tokens το υπόλοιπο της Ethereum blockchain διεύθυνσης κάθε χρήστη της εφαρμογής
- Δημιουργία ενός νέου έξυπνου συμβολαίου (Training contract) που θα διαχειρίζεται την αγοραπωλησία των αρχείων επίδειξης

4.2 Μη Λειτουργικες απαιτησεις

- Μέριμνα για υψηλό επίπεδο ασφάλειας
- Διασφάλιση της ακεραιότητας των συναλλαγών
- Διασφάλιση της ακεραιότητας και της άμεσης διαθεσιμότητας των αρχείων επίδειξης
- Εύκολη επεκτασιμότητα της εφαρμογής
- Γενικευμένη χρήση της έννοιας «μοντέλο μηχανικής μάθησης». Η εφαρμογή δεν εξειδικεύεται σε έναν τύπο μοντέλου μηχανικής μάθησης
- Όσο το δυνατόν βέλτιστος κώδικας του Training έξυπνου συμβολαίου

4.3 Συστηματα που Απαρτιζουν την Εφαρμογη

Το σχήμα που ακολουθεί τα συστήματα που απαρτίζουν την εφαρμογή και πως αλληλοεπιδρούν μεταξύ τους.



Εικόνα 3 – Συστήματα της εφαρμογής

Η εφαρμογή αποτελείται από τα εξής μέρη:

Εκπαιδευτής (Trainer agent)

Οι βασικές του λειτουργίες είναι:

- Καταγράφει την συμπεριφορά του κατά την διάρκεια εκτέλεσης μιας εργασίας σε αρχεία επίδειξης μέσω της πλατφόρμας Unity και τα διαθέτει μέσω του Ethereum blockchain
- Αλληλοεπιδρά με το Ethereum blockchain μέσω των έξυπνων συμβολαίων Ntua token και Training.
- Αποθηκεύει τα αρχεία επίδειξης στο κατανεμημένο σύστημα αρχείων IPFS

Εκπαιδευόμενος (Trainee agent)

Οι βασικές του λειτουργίες είναι:

- Αλληλοεπιδρά με το Ethereum blockchain μέσω των έξυπνων συμβολαίων Ntua token και Training.
- Ανακτά τα αρχεία επίδειξης από το κατανεμημένο σύστημα αρχείων IPFS
- Εκτελεί την εκπαίδευση του μέσω της επέκτασης της πλατφόρμας ML-agents.

Έξυπνα συμβόλαια υλοποιημένα στο Ethereum blockchain

Δύο είναι τα έξυπνα συμβόλαια που χρησιμοποιούνται από την εφαρμογή:

- Ntua Token (ERC20 Token standard) το οποίο υλοποιεί με ασφάλεια της συναλλαγές
- Training συμβόλαιο το οποίο διαχειρίζεται την αγοραπωλησία των αρχείων επίδειξης

Επέκταση των ML-Agents

Για τις ανάγκες της εφαρμογής, η πλατφόρμα παροχής μοντέλων μάθησης μέσω εκβάθυνσης, ML-Agents, επεκτάθηκε ώστε να παρέχει επιπλέον τις εξής λειτουργίες:

- Άμεση έναρξη της εκπαίδευσης του μοντέλου του εκπαιδευόμενου
- Στο τέλος κάθε κύκλου εκπαίδευσης ενημέρωση του εκπαιδευόμενου για την μέση επιβράβευση που κατέκτησε
- Ενημέρωση του εκπαιδευόμενου για το τέλος της εκπαίδευσης

Unity

Η Πλατφόρμα Unity δημιουργίας περιβάλλοντος που αλληλοεπιδρά με τα μοντέλα μηχανικής μάθησης χρησιμοποιείται κυρίως για:

- Ανάπτυξη τόσο του hummingbird agent (που αναπτύχθηκε αποκλειστικά για την υλοποίηση των πειραμάτων) όσο και του περιβάλλοντος με το οποίο αλληλοεπιδρά
- Επίδειξη την συμπεριφοράς των ήδη εκπαιδευόμενων μοντέλων και την καταγραφή αυτής σε αρχεία επίδειξης
- Απεικόνιση σε πραγματικό χρόνο της εξέλιξης της εκπαίδευσης

IPFS

- Αποθήκευση των αρχείων επίδειξης

4.4 ΜΕΘΟΔΟΛΟΓΙΑ ΑΝΑΠΤΥΞΗΣ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

Η εφαρμογή αναπτύχθηκε μέσω συνεχών επαναλήψεων, όπου σε κάθε επανάληψη γινόταν ο σχεδιασμός και η υλοποίηση λίγων νέων λειτουργιών. Κάθε επανάληψη τελείωνε με πλήρη έλεγχο της εφαρμογής.

Πιο συγκεκριμένα: αρχικά σχεδιάστηκε ένας βασικός σκελετός του Training contract και ενεργοποιήθηκε στο Ganache blockchain. Παράλληλα, αναπτύχθηκαν δύο python scripts που λειτουργούσαν ως τον εκπαιδευτή (test_trainer.py) και τον εκπαιδευόμενο (test_trainee.py). Σκοπός ήταν η σταδιακή βελτίωση και έλεγχος του Training contract.

Στην συνέχεια προστέθηκε το Ntua Token contract και έτσι δημιουργήθηκε (και ελέγχθηκε) ο σκελετός μίας mini κατανεμημένης εφαρμογής αγοραπωλησίας αρχείων επίδειξης.

Στο αμέσως επόμενο βήμα, έγινε η επέκταση της πλατφόρμας των ML-Agents και πλέον η mini κατανεμημένη εφαρμογή μπορούσε αυτόματα να ενεργοποιεί την εκπαίδευση του trainee agent με βάση ένα αρχείο εκπαίδευσης και να ενημερώνεται για την εξέλιξη και το τέλος αυτής.

Σε επόμενο βήμα ενσωματώθηκε η χρήση του IPFS. Στο σημείο αυτό ήταν έτοιμο το 1ο πρωτότυπο της εφαρμογής (Testing Prototype).

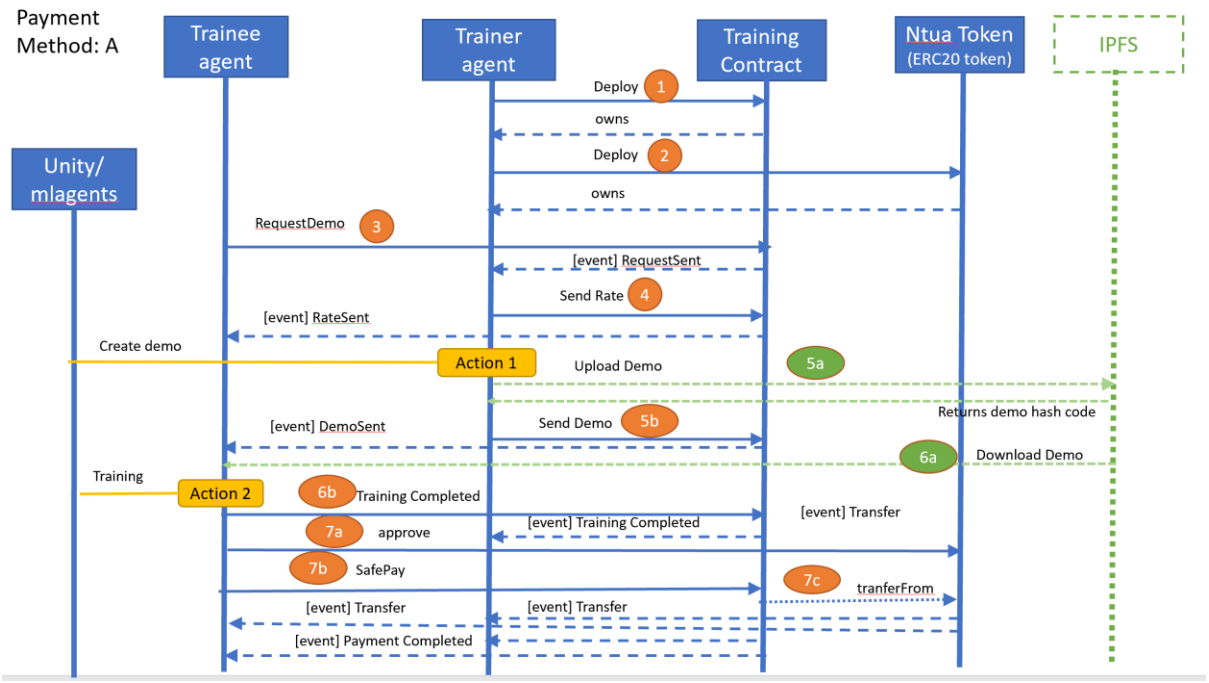
Για την ανάπτυξη του τελικού πρωτοτύπου της εφαρμογής τα δύο αρχικά python scripts (test_trainer.py και test_trainee.py) αντικαταστάθηκαν από αντίστοιχα mini web apps (Final Prototype)

Στην φάση της διεξαγωγής των πειραμάτων κατασκευάστηκε ο hummingbird agent, ένα πουλί που ζεί σε ένα νησί και τρέφεται με νέκταρ λουλουδιών. Τόσο ο νέος agent όσο και το περιβάλλον με το οποίο αλληλοεπιδρά ο agent κατασκευάστηκαν στο Unity περιβάλλον (Prototype for Experiments)

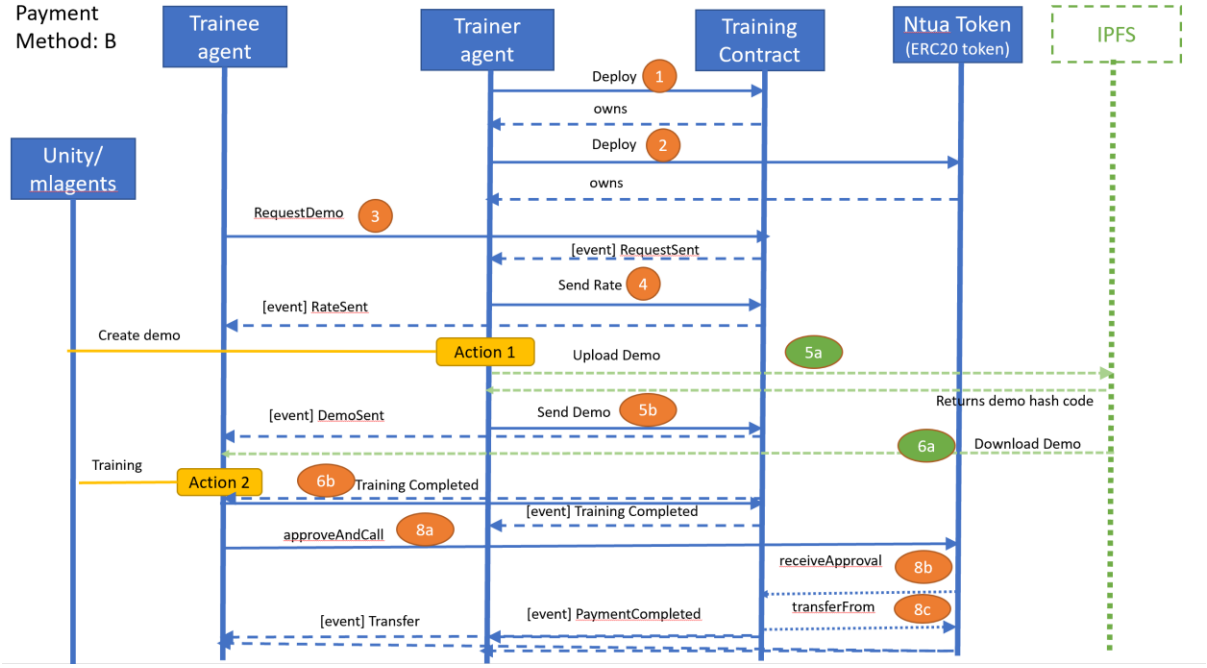
5 ΣΧΕΔΙΑΣΜΟΣ ΚΑΙ ΥΛΟΠΟΙΗΣΗ ΕΦΑΡΜΟΓΗΣ

5.1 ΒΑΣΙΚΗ ΡΟΗ ΕΡΓΑΣΙΩΝ (BASIC WORKFLOW)

Τα διαγράμματα που ακολουθούν παρουσιάζουν την βασική ροή εργασιών που υποστηρίζονται από την εφαρμογή. Τα βήματα 1-6 αποτελούν μέρος της βασικής ροής εργασιών η οποία ανάλογα με την μέθοδο αποπληρωμής ακολουθείται είτε από το βήμα 7 (Payment Method A) είτε από το βήμα 8 (Payment Method B)



Εικόνα 4 - Ροή εργασιών με μέθοδο αποπληρωμής A



Εικόνα 5 - Ροή εργασιών με μέθοδο αποπληρωμής B

1. Ο Trainer agent εγκαθιστά το Training smart contract στο Ethereum blockchain και θεωρείται πλέον ο κάτοχος του έξυπνου συμβολαίου
2. Ο Trainer agent εγκαθιστά το Ntua token (ERC20 Token) smart contract στο Ethereum blockchain και θεωρείται πλέον ο κάτοχος του έξυπνου συμβολαίου
3. Ο Trainee agent ζητά ένα αρχείο επίδειξης από το Training contract με την κλήση της 'RequestDemo' συνάρτησής του. Μέσω του 'RequestSent' γεγονότος ο Trainer agent δέχεται την αίτηση του Trainee.
4. Ο Trainer ενημερώνει το Training contract μέσω της κλήσης της 'SendRate' συνάρτησης του για το κόστος ανά μονάδα μέσης ανταμοιβής (mean reward) που θα επιτύχει ο Trainee κατά την διάρκεια της εκπαίδευσης.
5. Ο Trainer παράγει το αρχείο επίδειξης με σκοπό να το αποστείλει στον Trainee. Επειδή τα αρχεία επίδειξης είναι μεγάλα σε μέγεθος και το κόστος αποθήκευσής τους στο Ethereum blockchain είναι πολύ μεγαλύτερο, επιλέχθηκε η χρήση του IPFS. Έτσι, ο Trainer ανεβάζει το αρχείο επίδειξης στο IPFS βήμα 5α και αυτό με την σειρά του επιστρέφει ένα hashcode μοναδικό για το αρχείο. Στην συνέχεια ο Trainee αποστέλλει το hashcode του αρχείου επίδειξης στο Training contract μέσω της κλήσης της 'SendDemo' συνάρτησής του.
6. Μέσω του 'Demo Sent' γεγονότος ο Trainee αναγνωρίζει ότι το demo που είχε αιτηθεί είναι διαθέσιμο και μέσω του βήματος 6α το λαμβάνει και αυτόματα ξεκινά η εκπαίδευσή του με την χρήση της επέκτασης των πλατφόρμας ML-Agents . Κατά την διάρκεια της εκπαίδευσης και στο τέλος κάθε κύκλου αυτής η επέκταση των ML-Agents ενημερώνει τον Trainee για την μέση ανταμοιβή (mean reward) που έχει ήδη κατακτηθεί από τον Trainee. Όταν τελειώσει εκπαίδευση ο Trainee ενημερώνει για το τελική μέση ανταμοιβή που κατακτήθηκε το Training contract με την κλήση της 'Training Completed' συνάρτησης.

Δύο εναλλακτικές υλοποιήσεις αποπληρωμής του Trainer έχουν υλοποιηθεί και ελεγχθεί (Βήματα 7 και 8).

7. <u>(Α) μέθοδος αποπληρωμής</u>

A. Ο Trainee agent ενημερώνει το Ntua token contract ότι εγκρίνει την πληρωμή ενός ποσού προς το Training contract (εκτελώντας την ‘approve‘ συνάρτηση του Ntua token contract).
B. Ο Trainee agent μέσω της κλήσης ‘SafePay’ συνάρτησης δίνει εντολή στο ‘Training’ contract να αποπληρώσει τον Trainer
C. Το ‘Training’ contract, καλώντας την ‘transferFrom’ συνάρτηση του Ntua token contract, αποπληρώνει τον Tainer agent.

8. (Β) μέθοδος αποπληρωμής
   A. Ο Trainee ενημερώνει το Ntua token contract ότι εγκρίνει την πληρωμή ενός ποσού προς το Training contract (εκτελώντας την συνάρτηση του ‘approveAndCall’ Ntua token contract).

   B. Το Ntua token contract ενημερώνει το ‘Training’ contract ότι έχει εγκριθεί πληρωμή ενός ποσού για τον Trainer (εκτελώντας την receiveApproval() συνάρτηση του ‘Training’ contract)

   C. Το ‘Training’ contract ζητά από το Ntua token contract να μεταφέρει το εγκεκριμένο ποσό από τον λογαριασμό του Trainee προς τον λογαριασμό του Trainer (μέσω της κλήσης της transferFrom() συνάρτησης του Ntua token contract)

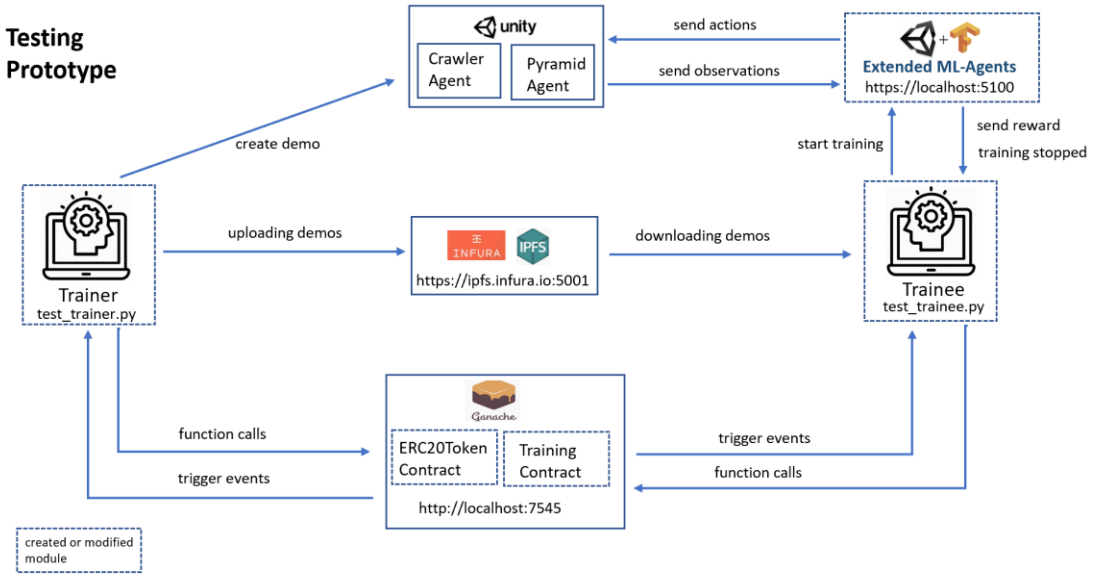
Σύγκριση των δύο διαφορετικών μεθόδων αποπληρωμής

- Η (Α) μέθοδος αποπληρωμής απαιτεί την εκτέλεση δύο δοσοληψιών συμβολαίων του Ethereum blockchain, μία είναι η ‘approve’ δοσοληψία και η άλλη είναι η ‘SafePay’. Η (Β) μέθοδος αποπληρωμής επιτρέπει στον Trainee να εκτελέσει την ίδια ακριβώς λειτουργία με μία μόνο δοσοληψία προς συμβόλαια του Ethereum blockchain, την ‘ApproveAndCall’.
- Η (Β) μέθοδος αποπληρωμής απαιτεί υλοποίηση πολύπλοκου κώδικα στα έξυπνα συμβόλαια καθώς τα δεδομένα αποθηκεύονται και μεταφέρονται σαν bytes στην approveAndCall() συνάρτηση.
- Η approveAndCall() συνάρτηση δεν αποτελεί μέρος του ERC20 Token standard αν και υπάρχει σχεδόν σε όλες τις διαθέσιμες standard υλοποιήσεις του.

5.2 ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

Τα διαγράμματα που ακολουθούν περιγράφουν την αρχιτεκτονική της εφαρμογής κατά τα διάφορα στάδια της ανάπτυξης της.

Testing Prototype



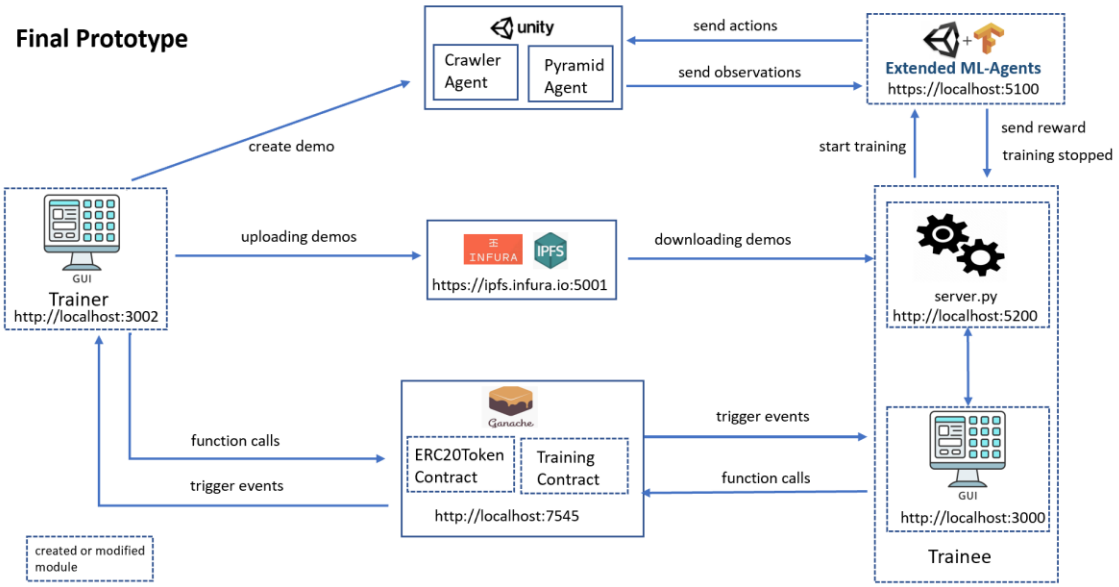
Εικόνα 6 - Αρχιτεκτονική της εφαρμογής - Testing prototype

Η απόφαση για δημιουργία Testing πρωτότυπου πάρθηκε ώστε α) να υπάρχει γρήγορα μία lite εφαρμογή προς επίδειξη και β) έγκαιρα να αναγνωρισθούν και να επιλυθούν τα προβλήματα της ολοκλήρωσης των διαφορετικών τεχνολογιών που χρησιμοποιεί η εφαρμογή.

Τα τμήματα που αναπτύχθηκαν εξ’ αρχής ή τροποποιήθηκαν στην φάση Testing Prototype είναι:

- ERC20Token contract: πρόκειται για ERC20 token standard που τροποποιήθηκε για την υλοποίηση του NTUA Token
- Training contract: αναπτύχθηκε για να περιγράψει την συμφωνία (agreement) μεταξύ του εκπαιδευόμενου (trainee) agent και του εκπαιδευτή (trainer) agent
- test_trainer.py: αναπτύχθηκε για να εξομοιώσει τις λειτουργίες του εκπαιδευτή (trainer) agent
- test_trainee.py: αναπτύχθηκε για να εξομοιώσει τις λειτουργίες του εκπαιδευόμενου (trainee) agent
- extended ML-agents: πρόκειται για τροποποιημένη έκδοση της βιβλιοθήκης ML-Agents ώστε να αυτοματοποιήσει την έναρξης της εκπαίδευσης του εκπαιδευόμενου (trainee) agent και να υποστηρίξει την (έμμεση) ενημέρωση του Training contract για την εξέλιξη της εκπαίδευσης

Final Prototype

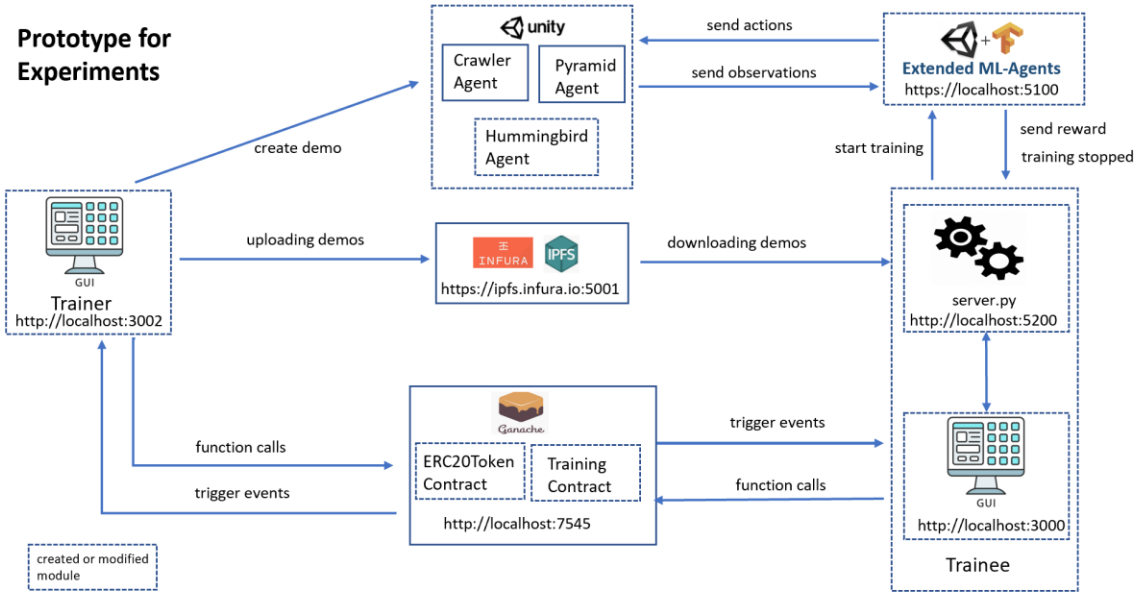


Εικόνα 7 - Αρχιτεκτονική της εφαρμογής - Final prototype

Τα τμήματα που αναπτύχθηκαν εξ' αρχής στην φάση Final Prototype είναι:

- Trainer web app: αναπτύχθηκε για να εξομοιώσει τις λειτουργίες του εκπαιδευτή (trainer) agent
- Trainee web app: αναπτύχθηκε για να εξομοιώσει τις λειτουργίες του εκπαιδευόμενου (trainee) agent
- server.py: αναπτύχθηκε για την υποστήριξη του Trainee web app.

Prototype for Experiments

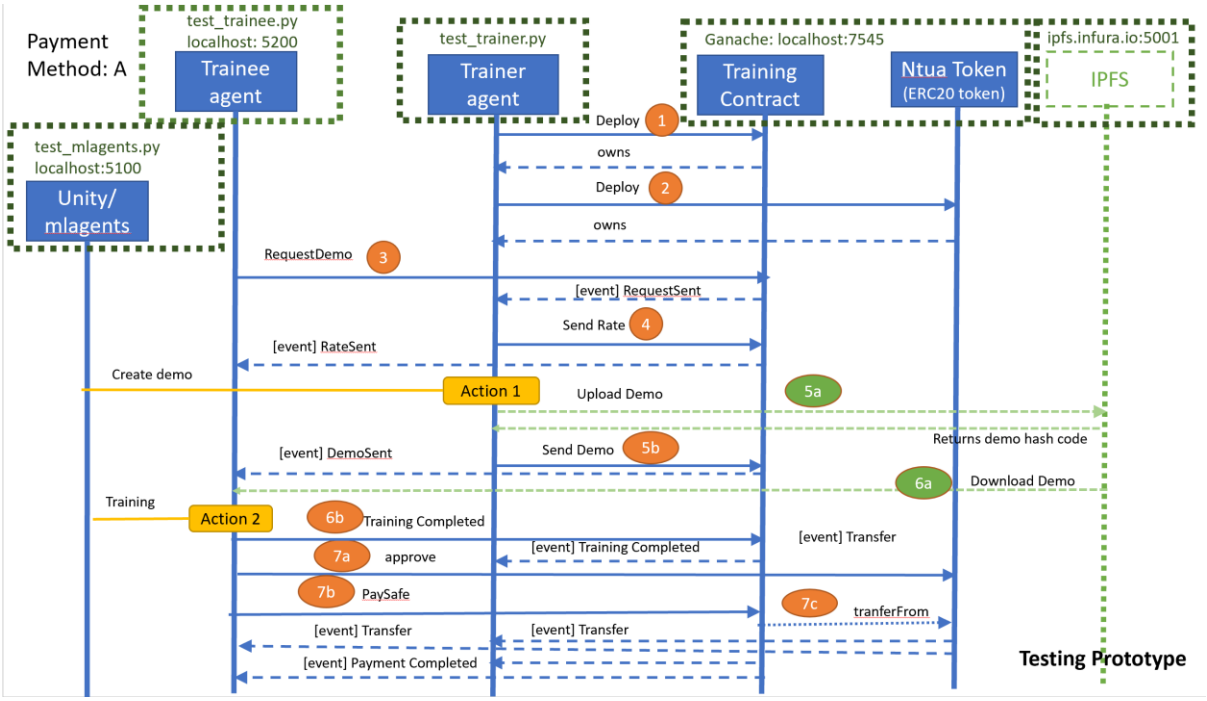


Εικόνα 8 - Αρχιτεκτονική της εφαρμογής – Prototype for experiments

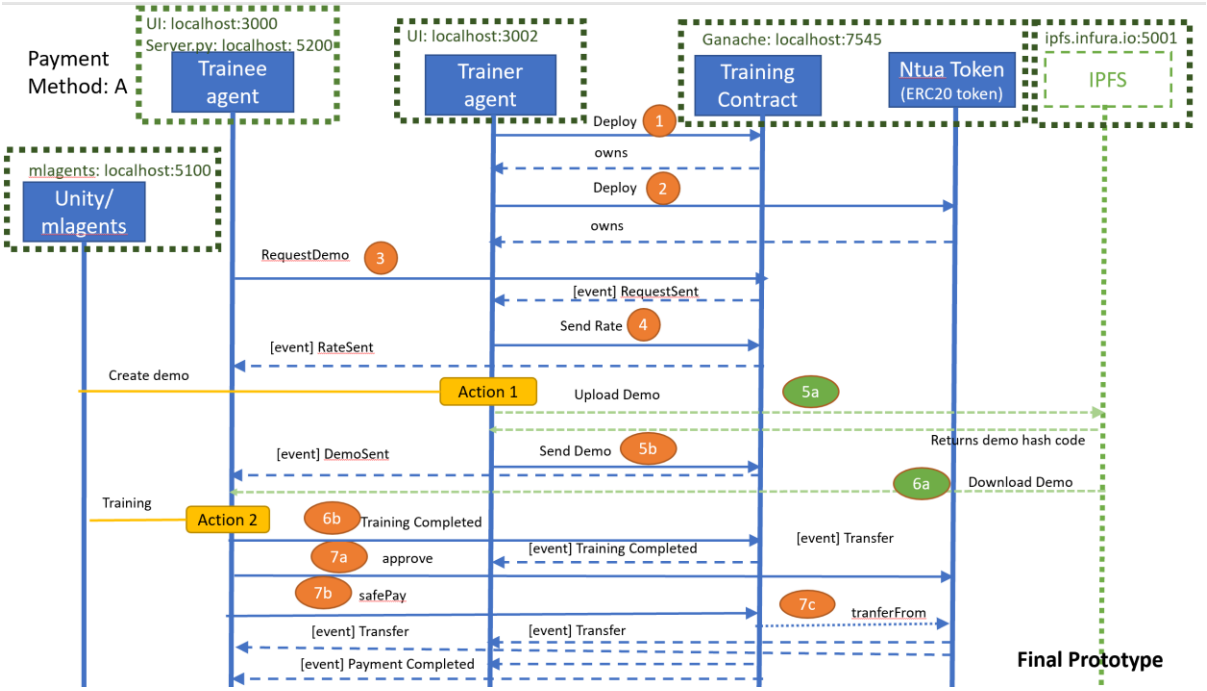
Στην φάση Prototype for Experiments αναπτύχθηκε εξ' αρχής:

- Hummingbird agent: πρόκειται για ένα νέο agent που αλληλοεπιδρά με ένα νέο περιβάλλον

Τα διαγράμματα που ακολουθούν παρουσιάζουν πως οι ροές εργασιών κατανέμονται και υλοποιούνται από τα συστήματα της εφαρμογής σε κάθε φάση ανάπτυξης αυτής.



Εικόνα 9 - Κατανομή της ροής εργασιών κατά τη μέθοδο αποπληρωμής Α ανά σύστημα της εφαρμογής κατά την φάση ανάπτυξης του Testing Prototype



Εικόνα 10 - Κατανομή της ροής εργασιών κατά τη μέθοδο αποπληρωμής Α ανά σύστημα της εφαρμογής κατά την φάση ανάπτυξης του τελικού πρωτότυπου (Final Prototype)

5.3 Έξυπνα Συμβολαια

Τα έξυπνα συμβόλαια (Smart contract) περιέχουν τον κώδικα που εκτελείται στο Ethereum VM και αποτελούν τα κεντρικά σημεία με τα οποία αλληλοεπιδρούν όλα τα επιμέρους συστήματα μίας αποκεντρωμένης εφαρμογής.

Η κύρια ιδιαιτερότητά τους, που αποτελεί και μέρος της ασφάλειας που προσφέρει το Ethereum blockchain, είναι ότι ο κώδικάς τους είναι οριστικός και δεν μπορεί να αλλάξει. Μετά την ενεργοποίηση ενός συμβολαίου στο blockchain, είναι αδύνατο να αλλάξει η συμπεριφορά του, είναι αδύνατη η αλλαγή του κώδικά του για την διόρθωση κάποιου σφάλματος στην λειτουργία του και με αυτό τον τρόπο τα περιεχόμενα του blockchain είναι αδύνατο να μεταβληθούν.

Για τις ανάγκες τις εφαρμογής δημιουργήθηκαν δύο έξυπνα συμβόλαια: το TokenERC20 Ntua token και το Training με την χρήση της γλώσσας Solidity [36]

5.3.1 Έξυπνο συμβόλαιο - TokenERC20

Το TokenERC20 Ntua συμβόλαιο ακολουθεί το ERC20 Token standard [33] και ανήκει στην κατηγορία των token έξυπνων συμβολαίων. Πρόκειται στην πραγματικότητα για ένα έξυπνο συμβόλαιο το οποίο συντηρεί ένα κατάλογο με διευθύνσεις Ethereum λογαριασμών (account address) και τα υπόλοιπά (balance) τους. Η έννοια του υπολοίπου προσδιορίζεται κάθε φορά από τον δημιουργό του token contract και ποικίλει ανά εφαρμογή. Μπορεί να

αναφέρεται σε φυσικά αντικείμενα, σε κάποιο νόμισμα, κλπ. Η μονάδα μέτρησης του υπολοίπου ονομάζεται token. Το token συμβόλαιο αρχικοποιείται με ένα σύνολο tokens (totalSupply). Όταν tokens μεταφέρονται από έναν λογαριασμό σε έναν άλλο το token contract ενημερώνει το υπόλοιπο των δύο λογαριασμών. Το αρχικό ποσό σε tokens του συμβολαίου μπορεί να αλλάξει με δύο τρόπους μόνο αν επιτρέπεται από το ίδιο το συμβόλαιο. Ο ένας τρόπος είναι καταστρέφοντας (burn) υπάρχοντα tokens και ο δεύτερος αποστέλλοντας tokens σε μια διεύθυνση που το ιδιωτικό της κλειδί δεν είναι γνωστό (συνήθως στην 0 address). Ο τρόπος αυτός αν και έχει το ίδιο αποτέλεσμα με τον πρώτο τρόπο καθώς το ποσό που μεταφέρεται παύει να είναι διαθέσιμο παρόλα αυτά δεν μειώνεται ο συνολικός αριθμός των tokens του συμβολαίου.

Ένα ERC20 token συμβόλαιο ορίζεται κυρίως από την διεύθυνση του στο Ethereum blockchain και το αρχικό ποσό των tokens (totalSupply) που διαπραγματεύεται. Επίσης προσδιορίζεται από ένα όνομα (name), από ένα (symbol) και από τον αριθμό των δεκαδικών (decimals) του token.

Στα πλαίσια της εφαρμογής το ERC20 token που ορίστηκε έχει όνομα NtuaToken, σύμβολο NtuaTok και συνολικό διαθέσιμο ποσό από NtuaTokens 10000.

Συναρτήσεις

Το ERC20 token συμβόλαιο περιλαμβάνει συναρτήσεις που επιτρέπουν στους χρήστες τους να γνωρίζουν το υπόλοιπο των λογαριασμών τους και να μεταφέρουν tokens μεταξύ λογαριασμών.

Η balanceOf() συνάρτηση επιστρέφει το υπόλοιπο ενός λογαριασμού. Να σημειωθεί ότι οποιοσδήποτε μπορεί να ρωτήσει για το υπόλοιπο οποιουδήποτε λογαριασμού καθώς όλα τα δεδομένα που αποθηκεύονται στο blockchain είναι δημόσια και διαθέσιμα σε όλους.

Η tranfer() συνάρτηση μεταφέρει tokens από την διεύθυνση του αποστολέα προς μία άλλη διεύθυνση. Δεν υπάρχει έλεγχος ως προς την διεύθυνση του παραλήπτη και επομένως είναι στην αρμοδιότητα του αποστολέα να διασφαλίσει την ορθότητα της διεύθυνσης του παραλήπτη. Η μέθοδος αυτή αποπληρωμής δεν είναι ενδεδειγμένη στην περίπτωση που ο αποστολέας είναι η διεύθυνση ενός συμβολαίου που εκτελεί χρέη μεσάζοντα μεταξύ του πραγματικού αποστολέα και του παραλήπτη. Και αυτό γιατί το συμβόλαιο δεν γνωρίζει λεπτομέρειες για την διεύθυνση του αποστολέα και επομένως δεν μπορεί να διασφαλίσει ότι ο χρήστης που καλεί το συμβόλαιο έχει πληρώσει το απαιτούμενο ποσό για να λειτουργεί το συμβόλαιο.

Έτσι επιλέγεται η σε δύο βήματα αποπληρωμή. Στο πρώτο βήμα ο αποστολέας καλεί την συνάρτηση approve() του ERC20 Token συμβολαίου εξουσιοδοτώντας (allowance), μία άλλη διεύθυνση, αυτή του συμβολαίου-μεσάζοντα να πραγματοποιήσει την μεταφορά των tokens. Στο επόμενο βήμα, το συμβόλαιο-μεσάζοντας μεταφέρει το ποσό στον παραλήπτη καλώντας την συνάρτηση tranferFrom().

Οι συναρτήσεις burn() και burnForm() χρησιμοποιούνται για την μείωση του αρχικού διαθέσιμου ποσού των tokens.

Γεγονότα

Κάθε φορά που ένα συμβόλαιο εκτελεί μία συγκεκριμένη ενέργεια ενεργοποιούνται γεγονότα (events) τα οποία και καταγράφονται στο Ethereum blockchain. Στο ERC20Token δύο γεγονότα έχουν δηλωθεί: Το Tranfer() γεγονός που εμπεριέχει πληροφορίες για την συναλλαγή που πραγματοποιήθηκε και το Burn() γεγονός που εμπεριέχει πληροφορίες για την μείωση του αρχικού διαθέσιμου ποσού των tokens που διατάχθηκε.

5.3.2 Έξυπνο συμβόλαιο – Training

Το Training συμβόλαιο διαχειρίζεται την αγοραπωλησία των αρχείων επίδειξης. Ορίζεται κυρίως από την διεύθυνση του στο Ethereum blockchain, τις διευθύνσεις των Trainer και Trainee agent αλλά και την διεύθυνση του Ntua Token. Περιλαμβάνει μία εσωτερική δομή (requests) μέσω της οποίας καταγράφει όλες τις λεπτομέρειες μιας αίτησης από έναν Trainee για αρχείο επίδειξης δηλαδή το είδος αρχείου που απαιτήθηκε, την διάρκεια του, το κόστος μονάδας, το hashcode του αρχείου επίδειξης και τέλος την μέση τιμή επιβράβευσης που κατέκτησε τελικά ο Trainee και έτσι έμμεσα τον αριθμό των Ntua tokens που μεταφέρθηκαν από τον Trainee στον Trainer.

Το Training συμβόλαιο περιλαμβάνει συναρτήσεις που επιτρέπουν στους Trainee και Trainer agents να επιδρούν μέσω του Training συμβολαίου με σκοπό την επίτευξη μιας αγοραπωλησίας αρχείων επίδειξης.

Με την κλήση της requestDemo() συνάρτησης ο Trainee απαιτεί ένα συγκεκριμένο αρχείο επίδειξης. Το Training contract καταγράφει την αίτηση και πυροδοτεί το RequestSent γεγονός.

Ο Trainer μέσω της κλήσης sendRate() συνάρτησης προσδιορίζει το κόστος της μονάδας χρήσης του αρχείου επίδειξης, το Training contract ενημερώνει την αίτηση αντίστοιχα και πυροδοτεί το RateSent γεγονός.

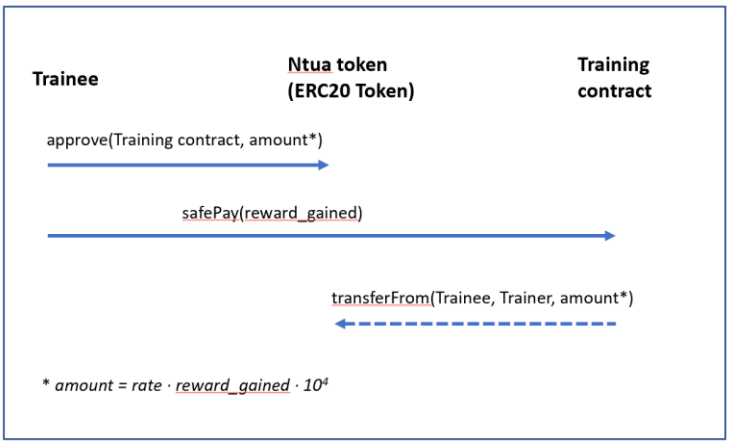
Η sendDemo() συνάρτηση καλείται από τον Trainer, ενημερώνει την αντίστοιχη αίτηση με το IPFS hashcode του αρχείου επίδειξης και πυροδοτεί το DemoSent γεγονός.

Η trainingCompleted() συνάρτηση καλείται από τον Trainee, ενημερώνει την αντίστοιχη αίτηση με τη μέση τιμή επιβράβευσης που κατέκτησε ο Trainee κατά την διάρκεια της εκπαίδευσης και πυροδοτεί το trainingCompleted γεγονός.

Για την αποπληρωμή του Trainer έχουν αναπτυχθεί δύο διαφορετικές συναρτήσεις ανάλογα με το τρόπο αποπληρωμής (Εικόνα 4 - Ροή εργασιών με μέθοδο αποπληρωμής Α και Εικόνα 5 - Ροή εργασιών με μέθοδο αποπληρωμής Β) που επιλέγεται κάθε φορά.

Μέθοδος αποπληρωμής Α

Η διαδικασία που ακολουθείται είναι η εξής:



Εικόνα 11 - Μέθοδος αποπληρωμής Α

- Ο Trainee ενημερώνει το Ntua Token ότι εξουσιοδοτεί την πληρωμή του Training contract με ποσό ίσο με το amount καλώντας την κλήση της approve() συνάρτησης του Ntua token συμβολαίου
- Ο Trainee ζητά από το Training contract να εκτελέσει την υπηρεσία αποπληρωμής καλώντας την safePay() συνάρτηση του Training contract
- Το Training contract καθοδηγεί το Ntua Token contract να εκτελέσει την μεταφορά ενός ποσού ίσο με το amount από τον λογαριασμό του Trainee προς τον λογαριασμό του Trainer (καλώντας την tranferFrom() συνάρτηση του NtuaToken contract). Η transferFrom() συνάρτηση καλείται άμεσα από την συνάρτηση safePay() και επομένως δεν εμφανίζεται σαν ανεξάρτητη δοσοληψία στο blockchain.

Η συνάρτηση approve() είναι :

```
function approve(address _spender, uint256 _value) public
    returns (bool success) {
    allowance[msg.sender][_spender] = _value;
    return true;
}
```

ενώ η συνάρτηση safePay() είναι:

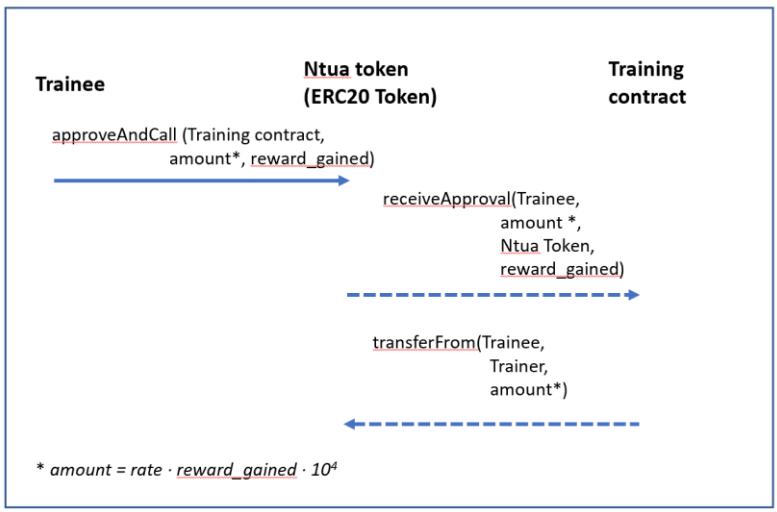
```
function safePay(uint requestno, uint reward_gained) public payable {

    require(msg.sender == traineeAddr);

    require(requests[requestno].init);

    uint amount = reward_gained * requests[requestno].rate * 10 ** 4;

    require(Ntua.transferFrom(msg.sender, trainer, amount));

    emit PaymentCompleted(traineeAddr, trainer, requestseq, amount);
}
```

όπου στην πρώτη γραμμή επιβεβαιώνεται ότι ο Trainee είναι αυτός που κάλεσε την συνάρτηση και στην συνέχεια, αφού υπολογισθεί το ποσό προς πληρωμή, καλείται η συνάρτηση transferFrom() του NtuaToken contract και πυροδοτείται το PaymentCompleted γεγονός.

Μέθοδος αποπληρωμής Β

Η προηγούμενη μέθοδος αποπληρωμής, μέθοδος Α, απαιτεί 2 δοσοληψίες από την πλευρά του Trainee (approve() και safePay()). Αυτό δεν είναι πάντα επιθυνητό καθώς προσθέτει φόρτο εργασίας στον Trainee. Η χρήση της approveAndCall() συνάρτησης του ERC20Token επιτρέπει στον Trainee να εκτελέσει την αποπληρωμή με μία μόνο δοσοληψία.

Η διαδικασία που ακολουθείται είναι η εξής:



Εικόνα 12 - Μέθοδος αποπληρωμής Β

- Ο Trainee ενημερώνει το Ntua Token ότι εξουσιοδοτεί την πληρωμή του Training contract με ποσό ίσο με το amount καλώντας την κλήση της approveAndCall() συνάρτησης του Ntua token συμβολαίου
- Το Ntua Token συμβόλαιο ενημερώνει το Training contract ότι μία πληρωμή ποσού ίσου με το amount εχει εξουσιοδοτηθεί (καλώντας της συνάρτηση receiveApproval() του Training contract)

- Το Training contract καθοδηγεί το Ntua Token contract να εκτελέσει την μεταφορά ενός ποσού ίσο με το amount από τον λογαριασμό του Trainee προς τον λογαριασμό του Trainer (καλώντας την tranferFrom() συνάρτηση του NtuaToken contract).

Η μέθοδος αυτή απαιτεί από το NtuaToken contract να έχει υλοποιήσει την συνάρτηση approveAndCall() :

```
function approveAndCall(address _spender, uint256 _value, bytes _extraData)
    public
    returns (bool success) {
    tokenRecipient spender = tokenRecipient(_spender);
    if (approve(_spender, _value)) {
        spender.receiveApproval(msg.sender, _value, this, _extraData);
        return true;
    }
}
```

μέσω της οποίας καλείται η συνάρτηση approve() και στην συνέχεια καλείται η συνάρτηση receiveApproval() του Training contract.

Η συνάρτηση receiveApproval() του Training contract είναι:

```
function receiveApproval(address _sender,
                         uint256 _value,
                         TokenERC20 _tokenContract,
                         bytes _extraData) public {
        require(Ntua == _tokenContract);
        // the _extraData is passed as bytes decode the value according
        uint256 payloadSize;
        uint256 reward_gained;
        uint256 amount;
        assembly {
            payloadSize := mload(_extraData)
            reward_gained := mload(add(_extraData, 0x20))
        }
        reward_gained = reward_gained >> 8*(32 - payloadSize);

        require(requests[requestseq].reward_gained == reward_gained);

        amount = requests[requestseq].rate * reward_gained * 10 ** 4;
        require(_value == amount);

        require(_tokenContract.transferFrom(_sender, address(this), _value));

        emit PaymentCompleted(traineeAddr, trainer, requestseq, _value);

    }
```

όπου στην πρώτη γραμμή επιβεβαιώνεται ότι το Ntua Token έχει καλέσει αυτή την συνάρτηση, στην συνέχεια υπολογίζονται το απαραίτητο ποσό προς πληρωμή, καλείται η συνάρτηση transferFrom() του NtuaToken contract και πυροδοτείται το PaymentCompleted γεγονός.

5.3.3 Επέκταση συμβολαίων

Αν και το ERC30Token αποτελεί μία καλή βάση για τον ορισμό token συμβολαίων έχει ένα βασικό πρόβλημα. Σύμφωνα με το [34] η χρήση του ERC20 Token standard μπορεί να επιφέρει για τους χρήστες του απώλεια χρημάτων.

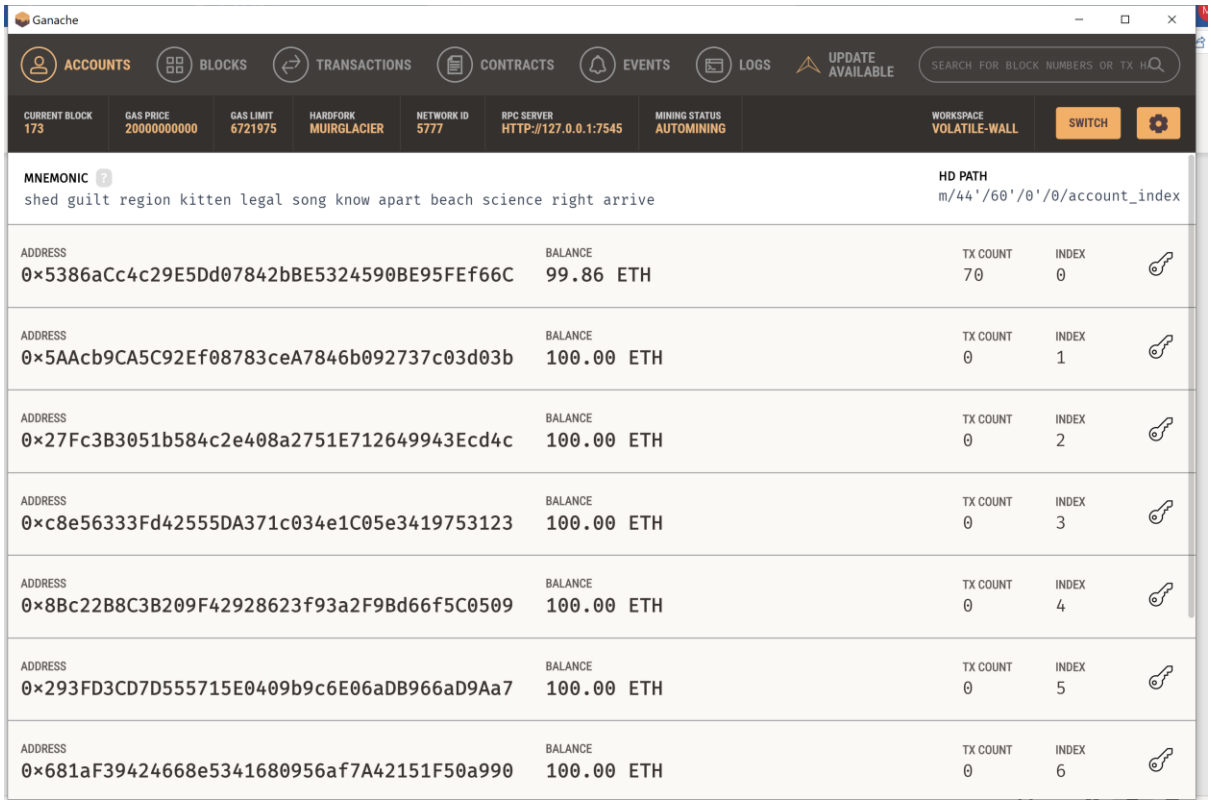
Το κύριο πρόβλημα είναι ότι αν κάποιος - κατά λάθος - στείλει ένα ποσό από ERC20 tokens σε ένα συμβόλαιο που δεν έχει σχεδιαστεί να διαχειρίζεται ERC20 tokens τότε το συμβόλαιο αυτό δεν θα απορρίψει τα tokens γιατί απλά δεν αναγνωρίζει την εισερχόμενη δοσοληψία.

Αντίθετα, αν κάποιος στείλει – κατά λάθος – ένα ποσό από ETH σε ένα συμβόλαιο που δεν διαχειρίζεται Ether τότε το συμβόλαιο αυτό θα απορρίψει αυτόματα την συναλλαγή.

Προς επίλυση αυτού του προβλήματος προτάθηκε η χρήση του ERC223Token standard [35].

5.4 ΔΗΜΙΟΥΡΓΙΑ (DEPLOYMENT) ΣΥΜΒΟΛΑΙΩΝ ΣΤΟ GANACHE

Για την υλοποίηση και τον έλεγχο της εφαρμογής κατασκευάστηκε ένα ιδιωτικό Ethereum blockchain μέσω του εργαλείου Truffle-Ganache [38]. Όταν το Ganache αρχικοποιείται δημιουργεί αυτόματα 10 Ethereum διευθύνσεις – πορτοφόλια (accounts). Για λόγους απλότητας και εύκολης μνημόνευσης ορίζεται ότι ο πρώτος από τους δέκα λογαριασμούς (account[0]) ανήκει στον Trainer agent ενώ ο τελευταίος λογαριασμός (account[9]) ανήκει στον Trainee agent.



Εικόνα 13 - Ganache Blockchain

Στην συνέχεια για την υλοποίηση, τον έλεγχο και την δημιουργία στο Ganache των έξυπνων συμβολαίων χρησιμοποιήθηκε το Truffle framework [37]. Αρχικά δημιουργήθηκε ένα truffle project με την εξής δομή:

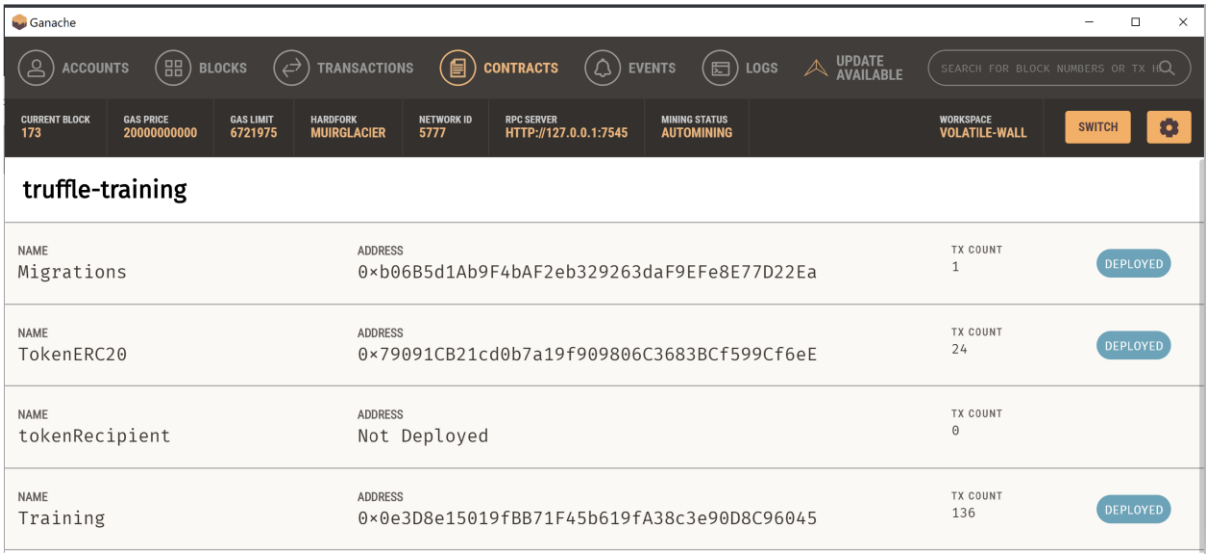
contracts/ : περιλαμβάνει τους κώδικες σε Solidity των έξυπνων συμβολαίων

migrations/ : καθώς το Truffle χρησιμοποιεί ένα αυτόματο σύστημα για την δημιουργία (deployment) στο Ganache των έξυπνων συμβολαίων, το directory αυτό περιλαμβάνει ένα ειδικού σκοπού script που αυτόματα δημιουργεί τα έξυπνα συμβόλαια στο Ganache blockchain. [39]

Για την δημιουργία των έξυπνων συμβολαίων στο Ganache, κατασκευάστηκε κάτω από το migrations/ directory το 2_deploy_training.js αρχείο:

```
var TokenERC20 = artifacts.require("TokenERC20");
var Training = artifacts.require("./Training.sol");

module.exports = function(deployer, network, accounts) {

    // Deploy the TokenERC20 contract
    deployer.deploy(TokenERC20, "10000", "NtuaToken", "NtuaTok")
        // Wait until the TokenERC20 contract is deployed
        .then(() => TokenERC20.deployed())
        // Deploy the Training contract, while passing the address of the
        // TokenERC20 contract
        .then(() => deployer.deploy(Training, TokenERC20.address, accounts[9]));
}
```

Όταν το αυτό script τρέξει, μέσω του truffle console, αρχικά θα επιβεβαιώσει την ύπαρξη των κωδίκων των δύο συμβολαίων στο /contracts directory. Στην συνέχεια θα προχωρήσει στην δημιουργία του ERC20Token στο Ganache blockchain. Καλεί τον constructor αυτού και δηλώνει ότι το αρχικό διαθέσιμο ποσό σε tokens θα είναι 10000, ότι το νέο token θα ονομάζεται NtuaToken και θα έχει σύμβολο το NtuaTok. Στο τέλος και αφού δημιουργηθεί το ERC20Token, το script θα προχωρήσει στην δημιουργία του Training συμβολαίου. Καλεί στην πραγματικότητα τον constructor αυτού και δηλώνει την διεύθυνση του ERC20Token συμβολαίου που λίγο πριν δημιουργήθηκε και την διεύθυνση του Trainee agent (account[9]).



Εικόνα 14 - Deployed contracts on Ganache

5.4.1 Έλεγχος συμβολαίων

Μετά την δημιουργία των συμβολαίων στο Ganache blockchain ακολουθεί ο βασικός έλεγχος τους. Για τον σκοπό γίνεται χρήση του εργαλείου truffle console [40].

Τα βήματα ελέγχου που ακολουθούν ελέγχου μία πλήρη διαδικασία αγοραπωλησίας ενός αρχείου επίδειξης του Hummingbird μοντέλου ακολουθώντας την μέθοδο αποπληρωμής Α.

```
// Initialization
var dm; Training.at("0x0e3D8e15019fBB71F45b619fA38c3e90D8C96045").then( function(x) { dm = x });     // 1
var tk; TokenERC20.at("0x79091CB21cd0b7a19f909806C3683BCf599Cf6eE").then( function(x) { tk = x });  // 2
let accounts = await web3.eth.getAccounts()                                                         // 3

// check default values of Training contract
dm.trainer().then( function(x) { return x.toString(); });                                           // 4
dm.traineeAddr().then( function(x) { return x.toString(); });                                       // 5

// check  default values of ERC20 Token contract
tk.totalSupply().then( function(x) { return x.toString(); });                                       // 6
tk.balanceOf(accounts[0]).then( function(x) { return x.toString(); });                              // 7

// Basic process
dm.requestDemo("Hummingbird", 10, {from: accounts[9]})                                              // 8
dm.sendRate(5, 4, {'from': accounts[0]})                                                            // 9
dm.sendDemo(5, 'QmQJFMHq6FopbfWJVkAppXYxm3jJYGH2QQmDuCJTZDBN49', {'from': accounts[0]})              // 10
dm.trainingCompleted(5, 133, {from: accounts[9]})                                                   // 11

// Payment method A
tk.approve("0x0e3D8e15019fBB71F45b619fA38c3e90D8C96045", 5320000, {from: accounts[9]})              // 12
dm.safePay(5, 133, {'from': accounts[9]})                                                           // 13
```

Η εντολή (1) αρχικοποιεί την μεταβλητή dm με ένα αντικείμενο τύπου Training contract. Η `0x0e3D8e15019fBB71F45b619fA38c3e90D8C96045` είναι η διεύθυνση του Training contract.

Η εντολή (2 αρχικοποιεί την μεταβλητή tk με ένα αντικείμενο τύπου ERC20Token contract (όπου η `0x79091CB21cd0b7a19f909806C3683BCf599Cf6eE` είναι η διεύθυνση του ERC20 Token contract.

Η εντολή (3) αρχικοποιεί την μεταβλητή accounts έτσι ώστε να περιλαμβάνει την λίστα των accounts που είναι διαθέσιμα στο Ganache blockchain.

Η εντολή (4) επιστρέφει την διεύθυνση του Trainer agent, και κατά αντιστοιχία η εντολή (5) επιστρέφει την διεύθυνση του Trainee agent.

Η εντολή (6) επιστρέφει το αρχικό συνολικό διαθέσιμο σε Ntua tokens του ERC20 Token ενώ η εντολή (7) επιστρέφει το υπόλοιπο του πρώτου λογαριασμού του Ganache blockchain και έχει ορισθεί να είναι ο λογαριασμός του Trainer agent.

Με την εντολή (8) το Training contract να καταγράφει στο blockchain μία αίτηση του Trainee (account[9]) για ένα demo του μοντέλου Hummingbird διάρκειας 10. Η εντολή (8) μεταξύ άλλων επιστρέφει και τον αύξοντα αριθμό της αίτησης (εδώ #5).

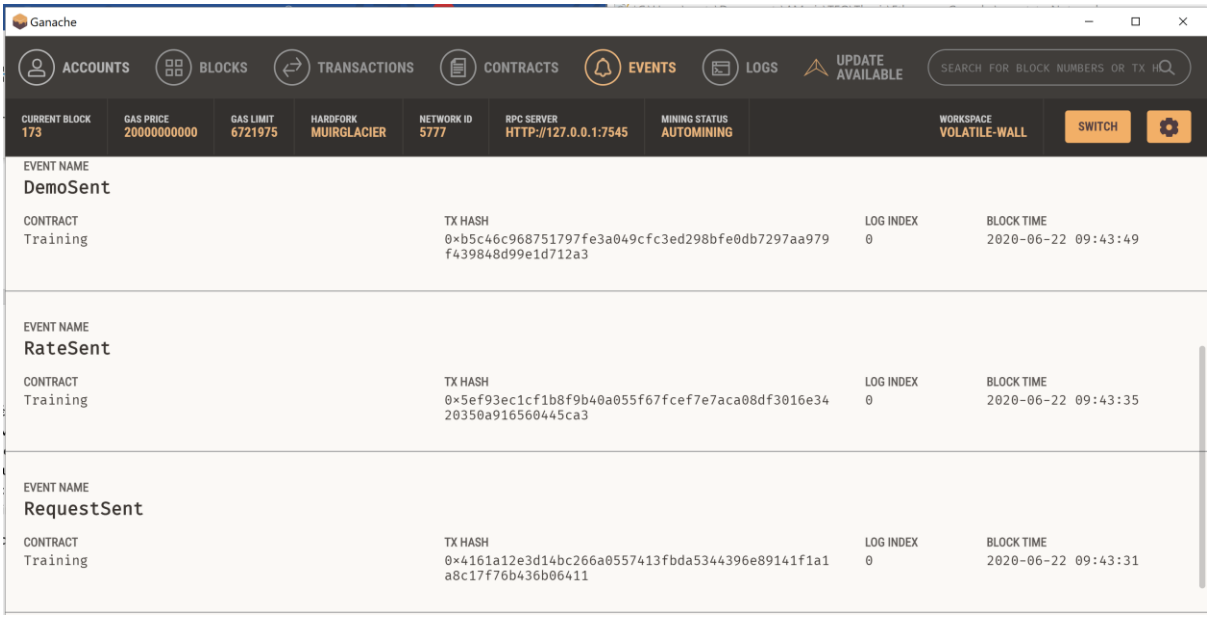
Με την εντολή (9) που αποστέλλεται από τον Trainer (account[0]), το Training contract ενημερώνει την αίτηση (#5) με το rate (εδώ 4)

Με την εντολή (10) που αποστέλλεται και πάλι από τον Trainer (account[0]), το Training contract ενημερώνει την αίτηση (#5) με το IPFS hashcode (εδώ `QmQJFMHq6FopbfWJVkAppXYxm3jJYGH2QQmDuCJTZDBN49`) του αρχείου επίδειξης.

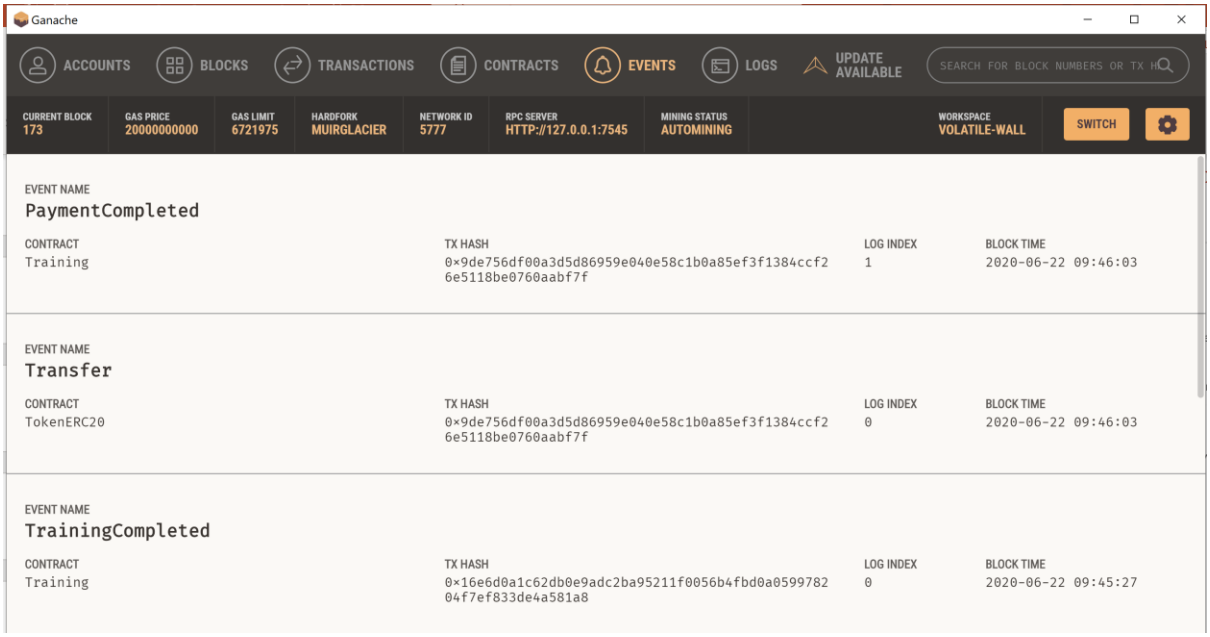
Με την εντολή (11) το Training contract ενημερώνει την αίτηση (#5) με το reward (εδώ 133) που κατακτήθηκε από τον Trainee (account[9]) μετά την ολοκλήρωση της εκπαίδευσης.

Οι εντολές (12) και (13) ολοκληρώνουν την αγοραπωλησία και αναφέρονται στην αποπληρωμή του Trainer. Με την εντολή (12) ο Trainee (account[9]) ενημερώνει το ERC20 Token contract ότι εξουσιοδοτεί το Training contract (εδώ `0x0e3D8e15019fBB71F45b619fA38c3e90D8C96045`) να εκτελέσει πληρωμή (εδώ ποσού ίσου με 5320000). Τέλος, με την εντολή (13), το Training contract διατάσσει έμμεσα το ERC20Token contract να αποπληρώσει τον Trainer και ενημερώνει το blockchain ότι η αίτηση (#5) έχει αποπληρωθεί.

Η εκτέλεση των εντολών (8) – (13) τροποποιεί την κατάσταση του Ethereum blockchain ως εξής:



Εικόνα 15 - Κατάσταση του Ethereum Blockchain μετά την εκτέλεση των συναρτήσεων: SendRequest, SendRate και SendDemo



Εικόνα 16 - Κατάσταση του Ethereum Blockchain μετά την εκτέλεση των συναρτήσεων: TrainingCompleted, Tranfer και PaymentCompleted

5.5 TRAINER AGENT

Ο Trainer Agent αναπτύχθηκε και ελέγχθηκε μέσα από δύο διαφορετικές υλοποιήσεις :

- Trainer agent για το Testing Prototype
- Trainer agent για το Final Prototype

5.5.1 Trainer agent για το Testing Prototype

Πρόκειται για το test_trainer.py python script το οποίο αναπτύχθηκε στα πλαίσια της δημιουργίας του Testing πρωτότυπου έτσι ώστε έγκαιρα να αναγνωρισθούν και να επιλυθούν τα προβλήματα της ολοκλήρωσης των διαφορετικών τεχνολογιών που χρησιμοποιεί η εφαρμογή.

Οι αρχικοποιήσεις που υλοποιούνται από το test_trainer.py script είναι:

- Ενσωμάτωση της web3.py βιβλιοθήκης και αρχικοποίηση ενός web3 httpProvider ο οποίος συνδέεται με το Ganache blockchain (http://localhost:7545)
- Ενσωμάτωση της ipfshttpclient βιβλιοθήκης και δημιουργία ενός http client οποίος συνδέεται με το IPFS του infura.io (/dns/ipfs.infura.io/tcp/5001/https)
- Αρχικοποίηση των λογαριασμών στο Ganache blockchain του Training contract, του ERC20Token contract και του Trainee
- Δημιουργία instances των Training contract και ERC20token contract με την χρήση των αντίστοιχων abi artifacts που δημιουργούνται αυτόματα κατά την διάρκεια του deployment των contracts στο Ganache blockchain

Η βασική λειτουργία του test_trainer.py script είναι η ασύγχρονη αναμονή γεγονότων που πυροδοτούν τα δύο συμβόλαια Training και ERC20Token. Πρόκειται για ένα Asynchronous Filter Polling [42] μηχανισμό ο οποίος κάνοντας χρήση των δυνατοτήτων της asyncio βιβλιοθήκης ενεργοποιεί ένα ασύγχρονο διαχειριστή blockchain γεγονότων χωρίς την ανάγκη δημιουργίας νέου thread.

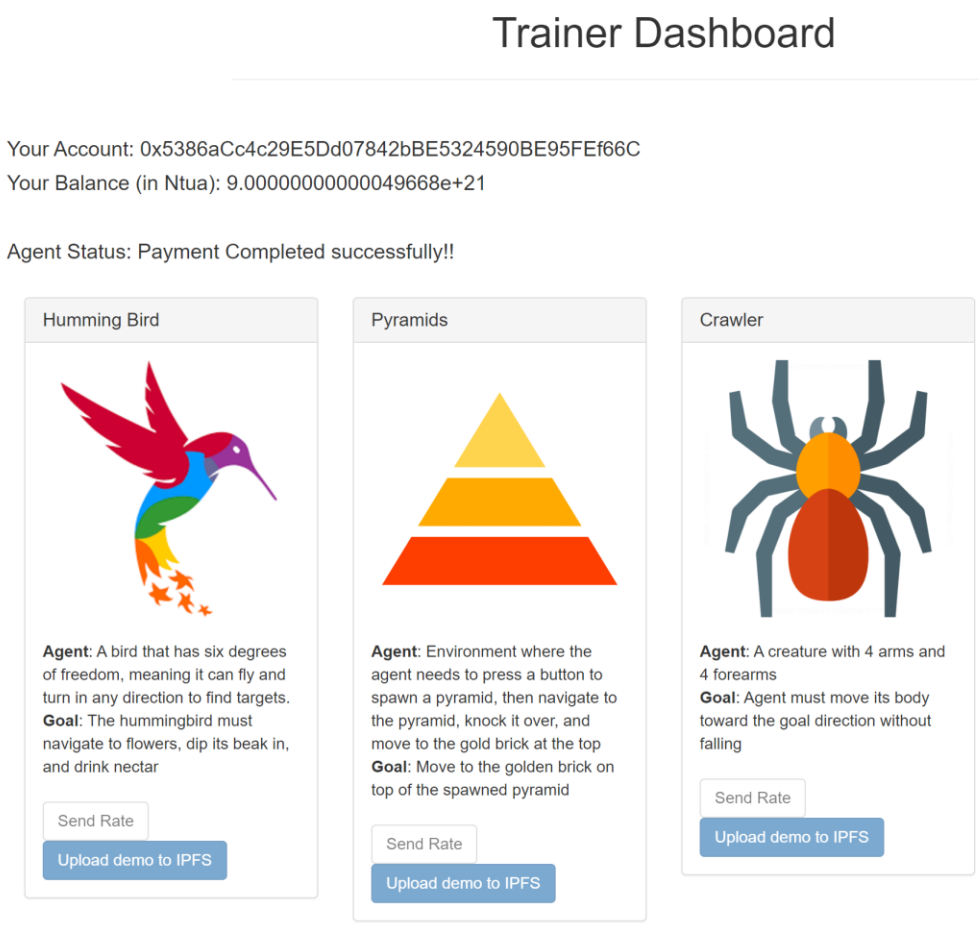
Η διαχείριση του κάθε blockchain γεγονότος που λαμβάνεται είναι διαφορετική και υλοποιείται μέσω διαφορετικής python function:

- RequestSent event: Καλείται η send_rate() function, η οποία αρχικά καλεί την SendRate() function του Training contract. Στην συνέχεια, με την λήψη του RateSent event ανεβάζει στο infura ipfs το αρχείο επίδειξης. Το hashcode του αρχείου που επιστρέφεται από το IPFS αποστέλλεται στο Training contract μέσω της κλήσης της SendDemo() function.
- TrainingCompleted event: υπολογίζει και εκτυπώνει το amount που πρόκειται να πιστωθεί στον Trainee.
- Transfer event και PaymentCompleted events: απλά εκτυπώνονται τα events.

5.5.2 Trainer agent για το Final Prototype

Πρόκειται για ένα mini web app μέσω του οποίου ο κάτοχος των ήδη εκπαιδευμένων μοντέλων παρουσιάζει τους τύπους των διαθέσιμων μοντέλων και προχωρεί στην πώληση αρχείων επίδειξης.

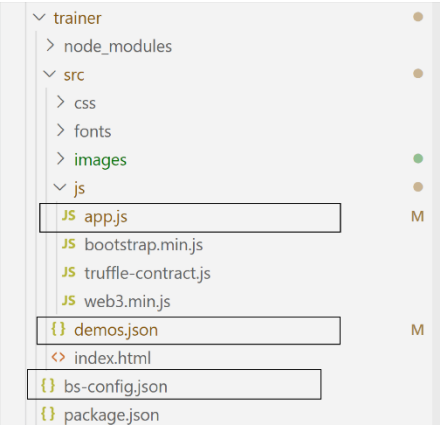
***Παρατήρηση**: Βασική προϋπόθεση και περιορισμός του συστήματος είναι τα αρχεία επίδειξης να έχουν προετοιμασθεί εκ των προτέρων και να υπάρχουν διαθέσιμα κατά την διάρκεια μιας αγοραπωλησίας.*



Εικόνα 17 - Trainer web app (Final Prototype)

Έχει υλοποιηθεί ένα λιτό web app που περιλαμβάνει μόνο τις άκρως απαραίτητες λειτουργίες που απαιτούνται για την επίδειξη της εφαρμογής. Εμφανίζει την διεύθυνση του κατόχου των αρχείων επίδειξης στο Ganache blockchain και το τρέχον υπόλοιπό του σε Ntua Token. Επιπλέον δίνεται η δυνατότητα στον τελικό χρήστη να εκτελεί τις απαραίτητες ενέργειες που απαιτούνται για την αγοραπωλησία αρχείων εκπαίδευσης ξεχωριστά για κάθε διαθέσιμο εκπαιδευμένο μοντέλο. Οι διαθέσιμες ενέργειες ενεργοποιούνται ανάλογα με την εξέλιξη της διαδικασίας μίας αγοραπωλησίας καταγράφοντας και εμφανίζοντας κάθε φορά την τρέχουσα κατάσταση του Trainer agent.

Το Trainer mini web app είναι υλοποιημένο σε JavaScript με χρήση του Bootstrap framework [46] για την μορφοποίηση των UI components και με εξυπηρετητή lite-server web server [47]. Είναι ένα npm project με δομή:



όπου:

το *app.js* περιλαμβάνει την λογική του Trainer web app,

το *demos.json* περιλαμβάνει τα χαρακτηριστικά των αρχείων επίδειξης που είναι άμεσα διαθέσιμα,

και το *bs-config.js* καθορίζει την σχετική θέση των contracts στην δομή του project.

Αρχικά, στο app.js ορίζεται ένα γενικό App object για να διαχειρίζεται εξ' ολοκλήρου την εφαρμογή και καλείται η init() function για να φορτώνει τα δεδομένα των αρχείων επίδειξης. Στην συνέχεια καλείται η initWeb3() function για την διασύνδεση της εφαρμογής με το Ethereum blockchain με την χρήση της web3.js βιβλιοθήκης [43] μέσω της οποίας μπορεί να εμφανίσει Ethereum λογαριασμούς, να εκτελέσει δοσοληψίες και γενικά να αλληλοεπιδράσει με τα έξυπνα συμβόλαια.

Για την έναρξη της αλληλοεπίδρασης με τα έξυπνα συμβόλαια καλείται η initContracts() function όπου δημιουργούνται instances αυτών και ενεργοποιούνται οι κατάλληλοι listeners των γεγονότων που πυροδοτούνται από τα έξυπνα συμβόλαια (listenForTrainingEvents() και listenForTokenEvents()). Στο τέλος, καλείται η showTrainerData() function για να αναζητήσει και να εμφανίσει τόσο την διεύθυνση του Trainer στο Ganache blockchain όσο και το τρέχον υπόλοιπό του σε Ntua Token.

Πιο συγκεκριμένα,

- init(): εμφανίζει τα δεδομένα / χαρακτηριστικά των διαθέσιμων αρχείων επίβλεψης
- initWeb3(): διασυνδέει το web app με το Ganache blockchain.
- initContracts(): δημιουργεί instances των συμβολαίων Training και Ntua Token
- listenForTrainingEvents():ενεργοποιεί event listeners για τα events του Training contract και εμφανίζει επεξεργασμένα τα δεδομένα που αυτά περιέχουν
- listenForTokenEvents(): ενεργοποιεί έναν event listener για το Transfer event του Ntua Token contract και εμφανίζει επεξεργασμένα τα δεδομένα που αυτά περιέχουν
- showTrainerData(): καλεί την balanceOf() function του NtuaToken contract
- sendRate(): καλεί την sendRate() function του Training contract
- sendDemo(): καλεί την sendDemo() function του Training contract
- uploadDemoToIPFS(): ανεβάζει το αρχείο επίδειξης στο infura IPFS

5.6 Trainee Agent

Ο Trainee Agent αναπτύχθηκε και ελέγχθηκε μέσα από δύο διαφορετικές υλοποιήσεις :

- Trainee agent για το Testing Prototype
- Trainee agent για το Final Prototype

5.6.1 Trainee agent για το Testing Prototype

Πρόκειται για έναν web app server, ο οποίος αναπτύχθηκε στα πλαίσια της δημιουργίας του Testing πρωτότυπου έτσι ώστε έγκαιρα να αναγνωρισθούν και να επιλυθούν τα προβλήματα της ολοκλήρωσης των διαφορετικών τεχνολογιών που χρησιμοποιεί η εφαρμογή.

Η υλοποίηση του Trainee agent αποτελείται από δύο python scripts:

test_trainee.py

Ένας web app server (http://localhost:5200) που αναπτύχθηκε με την χρήση του Flask web application framework με κύριο σκοπό να συνεργάζεται με την επέκταση της ML-Agents βιβλιοθήκης. Δηλώνει δύο rest api endpoints:

/completion: ενεργοποιείται όταν σταματά η εκπαίδευση του Trainee που λαμβάνει χώρα μέσω της επέκτασης της ML-Agents βιβλιοθήκης και καλεί την μέθοδο training_completed() της κλάσης Training

/reward: ενεργοποιείται όταν τελειώνει ένας κύκλος της εκπαίδευσης του Trainee και ενημερώνει το reward_gained χαρακτηριστικό της κλάσης Trainee

Παράλληλα, το test_tainee.py δημιουργεί ένα αντικείμενο της κλάσης Trainee και καλεί την μέθοδο start_transaction() της κλάσης Trainee.

trainee.py

Εκτελεί τις απαραίτητες αρχικοποιήσεις και ορίζει την κλάση Trainee. Οι αρχικοποιήσεις περιλαμβάνουν:

- Ενσωμάτωση της web3.py βιβλιοθήκης και αρχικοποίηση ενός web3 httpProvider ο οποίος συνδέεται με το Ganache blockchain (http://localhost:7545)
- Ενσωμάτωση της ipfshttpclient βιβλιοθήκης και δημιουργία ενός http client οποίος συνδέεται με το IPFS του infura.io (/dns/ipfs.infura.io/tcp/5001/https)
- Αρχικοποίηση των λογαριασμών στο Ganache blockchain του Training contract, του ERC20Token contract και του Trainee
- Δημιουργία instances των Training contract και ERC20token contract με την χρήση των αντίστοιχων abi artifacts που δημιουργούνται αυτόματα κατά την διάρκεια του deployment των contracts στο Ganache blockchain

Η κλάση Trainee περιλαμβάνει τα χαρακτηριστικά:

- requestno: αύξων αριθμός της αίτησης του Trainee για ένα αρχείο επίδειξης
- demotype: ο τύπος του αρχείου επίδειξης
- demohash: το IPFS hashcode του αρχείου επίδειξης
- duration: το μήκος του αρχείου επίδειξης
- rate: η τιμή μονάδας του το mean reward που κατακτήθηκε κατά την εκπαίδευση
- reward_gained: το mean reward που κατακτήθηκε κατά την εκπαίδευση

Οι μέθοδοι της κλάσης Trainee είναι:

Μέθοδος start_transaction(): καλεί την request_demo() μέθοδο και στην συνέχεια ενεργοποιεί μία ασύγχρονη αναμονή γεγονότων που πυροδοτούν τα δύο συμβόλαια Training και ERC20Token. Πρόκειται για ένα Asynchronous Filter Polling [42] μηχανισμό ο οποίος κάνοντας χρήση των δυνατοτήτων της asyncio βιβλιοθήκης ενεργοποιεί έναν ασύγχρονο διαχειριστή για κάθε blockchain γεγονός.

Η διαχείριση του κάθε blockchain γεγονότος είναι διαφορετική, υλοποιείται μέσω διαφορετικής python function και εκτελείται σε διαφορετικό thread:

- RateSent event: ενημερώνει το rate χαρακτηριστικό του αντικειμένου Trainee

- DemoSent event: με βάση το IPFS hashcode που περιλαμβάνεται σαν πληροφορία στο event, κατεβάζει από το infura ipfs το αρχείο επίδειξης. Στην συνέχεια, μέσω της requests python βιβλιοθήκης, στέλνει ένα /training post rest api request στην επέκταση της ML-Agents βιβλιοθήκης (http://localhost:5100) ζητώντας να ξεκινήσει άμεσα η εκπαίδευση του Trainee με χρήση του αρχείου επίδειξης.
- Transfer event και PaymentCompleted event: απλά εκτυπώνει τα events.

Μέθοδος request_demo(): καλείται η requestDemo() function του Training contract

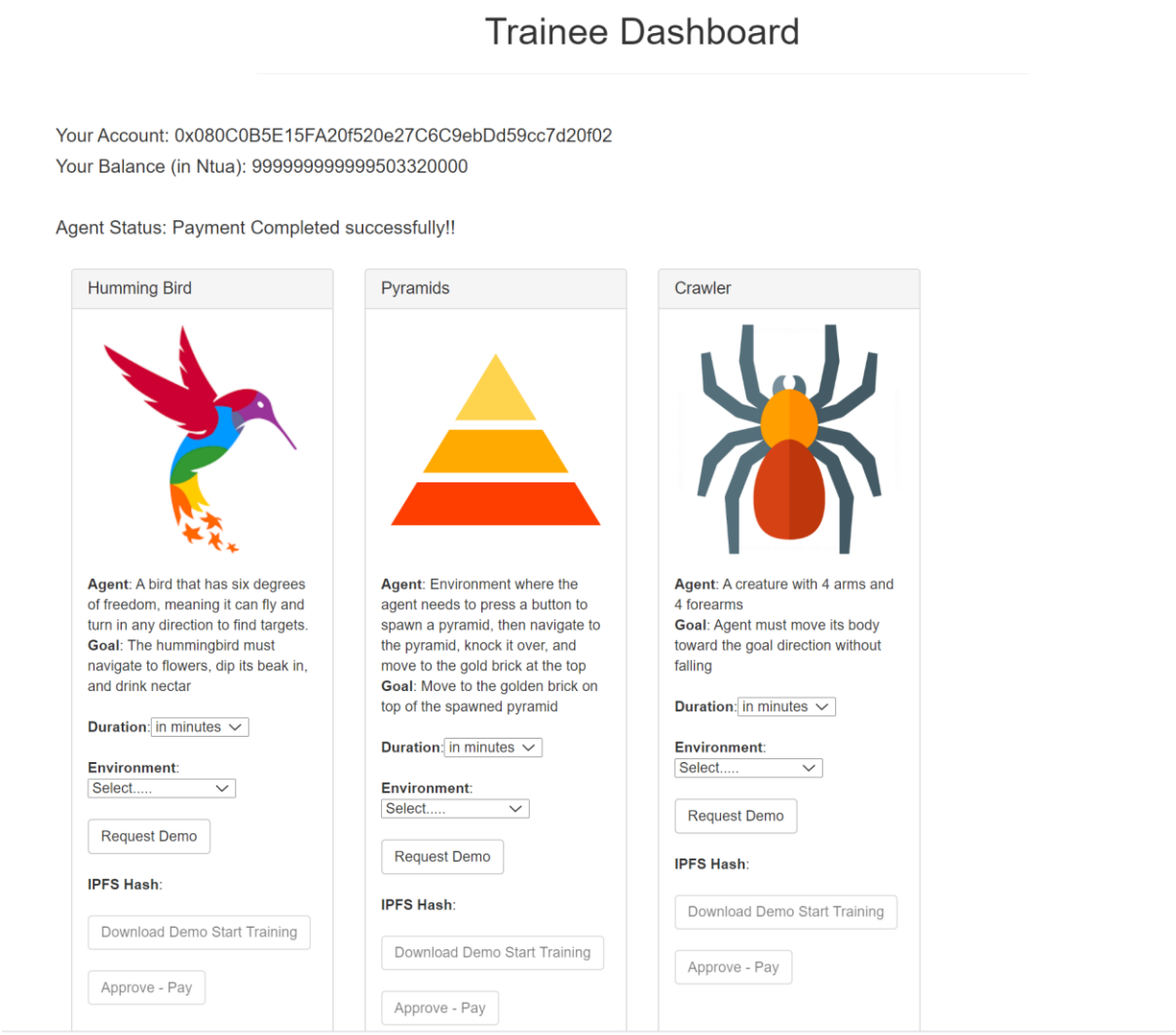
Μέθοδος training_completed(): ενεργοποιείται με την λήψη του /training_completed rest api request και καλείται η trainingCompleted() function του Training contract. Έτσι ενημερώνεται το Training contract για το τελικό mean reward που κατακτήθηκε κατά την διάρκεια της εκπαίδευσης. Με την λήψη του TrainingCompleted event, ξεκινά η διαδικασία αποπληρωμής του Trainer. Έχουν υλοποιηθεί και ελεγχθεί και οι δύο μέθοδοι αποπληρωμής.

5.6.2 Trainee agent για το Final Prototype

Απαρτίζεται από δύο modules:

5.6.2.1 Trainee mini web app

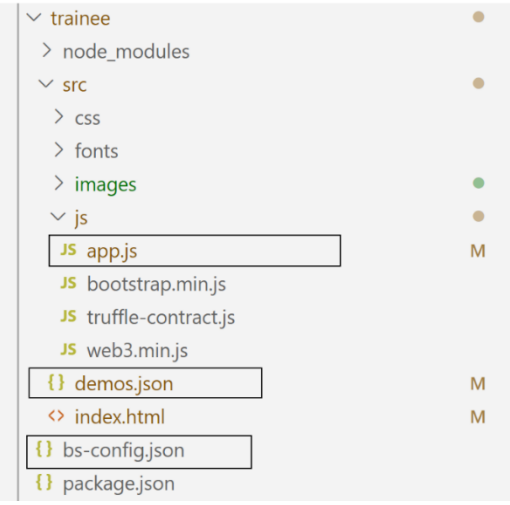
Πρόκειται για ένα mini web app μέσω του οποίου ο ενδιαφερόμενος για ένα διαθέσιμο αρχείο επίδειξης το επιλέγει και προχωρεί στην αγορά του.



Εικόνα 18 - Trainee web app (Final Prototype)

Όμοια με το Trainer we app, έχει υλοποιηθεί ένα λιτό web app που περιλαμβάνει μόνο τις άκρως απαραίτητες λειτουργίες που απαιτούνται για την επίδειξη της εφαρμογής. Εμφανίζει την Ganache blockchain διεύθυνση του ενδιαφερόμενου να αγοράσει αρχεία επίδειξης και το τρέχον υπόλοιπό του σε Ntua Token. Επιπλέον δίνεται η δυνατότητα στον τελικό χρήστη να εκτελεί τις απαραίτητες ενέργειες που απαιτούνται για την αγοραπωλησία αρχείων εκπαίδευσης ξεχωριστά για κάθε διαθέσιμο εκπαιδευμένο μοντέλο. Οι διαθέσιμες ενέργειες ενεργοποιούνται ανάλογα με την εξέλιξη της διαδικασίας μίας αγοραπωλησίας καταγράφοντας και εμφανίζοντας κάθε φορά την τρέχουσα κατάσταση του Trainee agent.

Το Trainee mini web app είναι υλοποιημένο σε JavaScript με χρήση του Bootstrap framework [46] για την μορφοποίηση των UI components και με εξυπηρετητή lite-server web server [47]. Είναι ένα npm project με δομή:



όπου:

το *app.js* περιλαμβάνει την λογική του Trainee web app,

το *demos.json* περιλαμβάνει τα χαρακτηριστικά των αρχείων επίδειξης που είναι άμεσα διαθέσιμα,

και το *bs-config.js* καθορίζει την σχετική θέση των contracts στην δομή του project.

Παρόμοια με το Trainee web app, αρχικά, στο app.js ορίζεται ένα γενικό App object για να διαχειρίζεται εξ' ολοκλήρου την εφαρμογή και καλείται η init() function για να φορτώνει τα δεδομένα των αρχείων επίδειξης. Στην συνέχεια καλείται η initWeb3() function για την διασύνδεση της εφαρμογής με το Ethereum blockchain με την χρήση της web3.js βιβλιοθήκης [43] μέσω της οποίας μπορεί να εμφανίσει Ethereum λογαριασμούς, να εκτελέσει δοσοληψίες και γενικά να αλληλοεπιδράσει με τα έξυπνα συμβόλαια.

Για την έναρξη της αλληλοεπίδρασης με τα έξυπνα συμβόλαια καλείται η initContracts() function όπου δημιουργούνται instances αυτών και ενεργοποιούνται οι κατάλληλοι listeners των γεγονότων που πυροδοτούνται από τα έξυπνα συμβόλαια (listenForTrainingEvents() και listenForTokenEvents()). Στο τέλος, καλείται η showTrainerData() function για να αναζητήσει και να εμφανίσει τόσο την διεύθυνση του Trainee στο Ganache blockchain όσο και το τρέχον υπόλοιπό του σε Ntua Token.

Πιο συγκεκριμένα,

- init(): εμφανίζει τα δεδομένα / χαρακτηριστικά των διαθέσιμων αρχείων επίβλεψης
- initWeb3(): διασυνδέει το web app με το Ganache blockchain.
- initContracts(): δημιουργεί instances των συμβολαίων Training και Ntua Token
- listenForTrainingEvents():ενεργοποιεί event listeners για τα events του Training contract και εμφανίζει επεξεργασμένα τα δεδομένα που αυτά περιέχουν
- listenForTokenEvents(): ενεργοποιεί έναν event listener για το Transfer event του Ntua Token contract και εμφανίζει επεξεργασμένα τα δεδομένα που αυτά περιέχουν
- showTraineeData(): καλεί την balanceOf() function του NtuaToken contract
- requestDemo(): καλεί την requestDemo() function του Training contract
- downloadDemo(): ζητά από τον Trainee server (http://localhost://5200) (server.py) να προχωρήσει στο κατέβασμα του αρχείου επίδειξης από το infura IPFS και στην συνέχεια να ξεκινήσει την εκπαίδευση.
- approveAndPay(): καλεί την approveAndPay() function του NtuaToken contract.

5.6.2.2 server.py - web app server

Είναι ο Trainee web app server (http://localhost:5200) που αναπτύχθηκε με την χρήση του Flask web application framework με κύριο σκοπό α) να εξυπηρετεί το Trainee mini web app, β) να αποτελεί τον μεσάζοντα μεταξύ του Trainee mini web app και με της επέκτασης της ML-Agents βιβλιοθήκης και γ) να κατεβάζει τα απαιτούμενα αρχεία επίδειξης από το infura IPFS.

***Παρατήρηση**: Η χρήση ενός web app server ήταν επιβεβλημένη καθώς η επικοινωνία μεταξύ του Trainee mini web app και της επέκτασης της ML-Agents είναι ασύγχρονη και κυρίως αμφίδρομη. Η υλοποίηση μιας τέτοιας επικοινωνίας θα ήταν τεχνικά εφικτή αν σχεδιαζόταν και κατασκευαζόταν από το μέρος του Trainee web app ένας polling μηχανισμός. Καθώς όμως η τεχνική αυτή λύση θεωρήθηκε ότι ήταν εκτός εμβέλειας της συγκεκριμένης εργασίας επιλέχθηκε η λύση της δημιουργίας του ενδιάμεσου web app server.*

Ο Trainee web app server περιλαμβάνει μια ενσωματωμένη κλάση Trainee και δηλώνει τρία rest api endpoints:

/completion: ενεργοποιείται όταν σταματά η εκπαίδευση του Trainee που λαμβάνει χώρα μέσω της επέκτασης της ML-Agents βιβλιοθήκης και καλεί την μέθοδο training_completed() της κλάσης Trainee

/reward: ενεργοποιείται όταν τελειώνει ένας κύκλος της εκπαίδευσης του Trainee και ενημερώνει το reward_gained χαρακτηριστικό της κλάσης Trainee

/start_training: ενεργοποιείται όταν ο χρήστης μέσω του Trainee mini web app απαιτήσει ξεκινήσει η εκπαίδευση και καλεί την μέθοδο download_demo() της κλάσης Trainee

Η κλάση Trainee περιλαμβάνει τα χαρακτηριστικά:

- requestno: αύξων αριθμός της αίτησης του Trainee για ένα αρχείο επίδειξης
- demohash: το IPFS hashcode του αρχείου επίδειξης
- reward_gained: το mean reward που κατακτήθηκε κατά την εκπαίδευση

και οι μέθοδοι της είναι:

Μέθοδος downloadDemo(): με βάση το IPFS hashcode που περιλαμβάνεται σαν πληροφορία στο request, κατεβάζει από το infura ipfs το αρχείο επίδειξης και το αποθηκεύει στο folder Demos/. Στην συνέχεια, μέσω της requests python βιβλιοθήκης, στέλνει ένα /training post rest api request στην επέκταση της ML-Agents βιβλιοθήκης (http://localhost:5100) ζητώντας να ξεκινήσει άμεσα η εκπαίδευση του Trainee με χρήση του αρχείου επίδειξης.

Μέθοδος trainingCompleted(): ενεργοποιείται με την λήψη του /training_completed rest api request και καλείται η trainingCompleted() function του Training contract. Έτσι ενημερώνεται το Training contract για το τελικό mean reward που κατακτήθηκε κατά την διάρκεια της εκπαίδευσης.

5.7 Επεκταση της βιβλιοθηκης ML-Agents

Στόχος της επέκτασης της βιβλιοθήκης ML-Agents ήταν να ενεργοποιείται αυτόματα η εκπαίδευση του Trainee agent μετά από απαίτηση του τελικού χρήστη και να ενημερώνεται αυτόματα ο Trainee agent τόσο για την μέση επιβράβευση (mean reward) που κερδίζεται κατά την διάρκεια της εκπαίδευσης όσο και για την εξέλιξη αυτής. Για την επίτευξη αυτού του στόχου ακολουθήθηκαν μια σειρά από βήματα. Αρχικά, μελετήθηκε με προσοχή η δομή της βιβλιοθήκης με σκοπό να εντοπιστούν τα modules που θα έπρεπε να τροποποιηθούν. Μετά τον εντοπισμό των απαραίτητων modules και την επισκόπηση του κώδικά τους ξεκίνησε η τροποποίηση της βιβλιοθήκης. Οι τροποποιήσεις που έγιναν είναι:

- προστέθηκε rest api υποδομή έτσι ώστε να μπορεί η βιβλιοθήκη να δέχεται http post rest api requests με την χρήση του Flask web application framework. Δηλώθηκε το **/training** rest api endpoint μέσω του οποίου η βιβλιοθήκη λαμβάνει εντολή να ξεκινήσει αυτόματα την εκπαίδευση με χρήση του αρχείου επίδειξης που εμπεριέχεται στο /training post request.

[αρχείο που τροποποιήθηκε:
- *ml-agents\mlagents\training\learn.py]*

- από την κατασκευή της η βιβλιοθήκη ML-Agents έχει σχεδιασθεί να διαβάζει τις παραμέτρους της εκπαίδευσης από ένα configuration αρχείο. Καθώς μία από τις παραμέτρους της εκπαίδευσης είναι το αρχείο επίδειξης, έγιναν οι απαραίτητες τροποποιήσεις ώστε η επέκταση της βιβλιοθήκης να χρησιμοποιεί τα αρχεία επίδειξης που έρχονται μέσω του /training post request αντί αυτών που καταγράφονται στο configuration file.

  [αρχεία που τροποποιήθηκαν:
  - *ml-agents\mlagents\training\learn.py*
  - *ml-agents\mlagents\training\training_controller.py*
  - *ml-agents\mlagents\training\training_util.py*
  - *ml-agents\mlagents\training\rl_trainer.py]*

- στο τέλος κάθε κύκλου εκπαίδευσης η επέκταση της βιβλιοθήκης αποστέλλει ένα **/reward** post request στον web app server που χτίζει ο server.py (http://localhost:5200) ενημερώνοντας τον για το mean reward που έχει κατακτηθεί κατά την διάρκεια του τρέχοντος κύκλου εκπαίδευσης

  [αρχείο που τροποποιήθηκε:
  - *ml-agents\mlagents\training\stats.py]*

- όταν η εκπαίδευση τελειώσει ή όταν σταματήσει μετά από επέμβαση του τελικού χρήστη, η επέκταση της βιβλιοθήκης αποστέλλει ένα **/completion** post request στον web app server που, χτίζει ο server.py (http://localhost:5200) ενημερώνοντας τον για το τέλος της εκπαίδευσης

  [αρχείο που τροποποιήθηκε:
  - *ml-agents\mlagents\training\training_controller.py]*

Μετά το τέλος κάθε εκπαίδευσης, η επέκταση της ML-Agents βιβλιοθήκης μπαίνει σε κατάσταση αναμονής εντολής έναρξης εκπαίδευσης.

5.8 ΔΙΑΣΥΝΔΕΣΗ ΜΕ ΤΟ INFURA - IPFS ΣΥΣΤΗΜΑ

Δύο είναι οι διαφορετικές μορφές διασύνδεσης με το Infura – IPFS σύστημα:

Α. Αποθήκευση αρχείου ή ‘ανέβασμα’ αρχείου στο Infura – IPFS (file upload)

Β. Ανάκτηση αποθηκευμένου αρχείου ή κατέβασμα αρχείου από το Infura – IPFS (file download)

Η διαδικασία ανεβάσματος του αρχείου επίδειξης υλοποιείται από τα εξής modules:

- Στο Testing prototype το test_trainer.py module
- Στο Final prototype και στο Trainer web app το app.js module

Και στα δύο modules αρχικοποιείται ένας ipfs-client ως εξής:

test_trainer.py : `ipfs_client = ipfshttpclient.connect('/dns/ipfs.infura.io/tcp/5001/https')`

app.js : `ipfs : window.IpfsHttpClient('ipfs.infura.io', '5001', { protocol: 'https' }),`

και στην συνέχεια καλείται η μέθοδος ipfs.add() (η αντίστοιχη σε κάθε ipfs client) η οποία αναλαμβάνει να ανεβάσει το αρχείο επίδειξης στο infura ipfs. Η μέθοδος ipfs.add() επιστρέφει το hashcode που μοναδιαία αντιστοιχίζεται από το IPFS στο αρχείο επίδειξης. Αυτό το hashcode αποθηκεύεται στην συνέχεια στο Training contract αντί του ίδιου του αρχείου επίδειξης.

Η διαδικασία κατεβάσματος του αρχείου επίδειξης υλοποιείται από τα εξής modules:

- Στο Testing prototype το trainer.py module
- Στο Final prototype το server.py module του Trainee web app server

Και στα δύο modules αρχικοποιείται ένας ipfs-client (όμοιος με παραπάνω) και καλείται η μέθοδος ipfs.get() η οποία έχοντας σαν παράμετρο το hashcode του αρχείου επίδειξης αναλαμβάνει να το κατεβάσει από το infura ipfs.

5.9 ML-AGENTS ΜΟΝΤΕΛΑ

Για τα Tesing και Final prototypes, χρησιμοποιήθηκαν 2 υπάρχοντα ήδη εκπαιδευμένα ML-Agents μοντέλα [48]:

- Dynamic Crawler agent
- Pyramids agent

Για τις ανάγκες του 3ου πρωτότυπου (Prototype for experiments) κατασκευάστηκε ένας νέος agent, ο hummingbird agent.

5.9.1 Dynamic Crawler agent

Πρόκειται για ένα αρθρόποδο (αράχνη) με 4 πόδια που μετακινείται προς έναν στόχο του οποίου η θέση καθορίζεται δυναμικά και δεν είναι κατ' ανάγκη πάντα μπροστά από την αράχνη. Η επιβράβευση (reward) του crawler agent έχει προκαθορισθεί ως :

- + 0.03 x την ταχύτητα της κίνησης της αράχνης προς τον στόχο
- + 0.01 κατά την προσέγγιση της αράχνης στον στόχο

Ο crawler agent προσφέρεται ήδη εκπαιδευμένος με benchmark mean reward : 400

5.9.2 Pyramids agent

Πρόκειται για έναν agent που έχει στόχο να πατήσει ένα κουμπί για να εμφανίσει μία πυραμίδα από πέτρες κάπου τυχαία στον χώρο και στην συνέχεια αφού βρει την πυραμίδα και την διαλύσει να μετακινηθεί προς την κίτρινη πέτρα που είναι τοποθετημένη στην κορυφή της. Η επιβράβευση (reward) του agent έχει προκαθορισθεί ως :

- + 2 για την μετακίνηση προς το κίτρινο

Ο pyramids agent προσφέρεται ήδη εκπαιδευμένος με benchmark mean reward : 1.75

5.9.3 Hummingbird agent

Καθώς τα υπάρχοντα μοντέλα της βιβλιοθήκης ML-Agents είναι πλήρως εκπαιδευμένα, δεν μπορούσαν να χρησιμοποιηθούν για τις ανάγκες των πειραμάτων. Έτσι δημιουργήθηκε ένας νέος agent, ο hummingbird agent ακολουθώντας το 'ML-Agents – Hummingbird' course [49] που προσφέρει η Unity με σκοπό την εμβάθυνση στην υλοποίηση περιβαλλόντων και agents που εκπαιδεύονται μέσω της ML-Agents πλατφόρμα.

Το περιβάλλον που κατασκευάστηκε αποτελείται από ένα νησί με τον ορίζοντα του όπου πάνω στο νησί υπάρχουν βράχια, δέντρα, θάμνοι και λουλούδια. Το hummingbird ζει σε αυτό το νησί και τρέφεται με το νέκταρ των λουλουδιών. Έχει την δυνατότητα να βρίσκει τα λουλούδια που υπάρχουν στο νησί, να τοποθετεί το ράμφος του στο μίσχο τους και να πίνει το νέκταρ τους. Το hummingbird μπορεί να κινεί το σώμα του προς τα μπρος, προς τα πάνω και προς τα κάτω (δηλαδή να πετά στον ορίζοντα του νησιού) και παράλληλα να μετακινεί το ράμφος του κατ' ύψος πάνω / κάτω (pitch) και να το στρέφει αριστερά / δεξιά (yaw). Σκοπός του hummingbird είναι να μαζέψει όση περισσότερη τροφή μπορεί. Το hummingbird επιβραβεύεται κατά +0.01 για κάθε στιγμή που τρώει νέκταρ και αποθαρύνεται με -0.5 όταν βρεθεί στα όρια του νησιού.

Μετά την κατασκευή του περιβάλλοντος του νησιού και του humming bird agent ακολούθησε αρχικά η βασική εκπαίδευση του μοντέλου με RL τεχνικές και στην συνέχεια η περαιτέρω εκπαίδευσή του με στόχο την εύρεση εκείνων των παραμέτρων εκπαίδευσης που δίδουν το καλύτερο mean reward στον bumming bird (benchmarking) [Πειράματα]

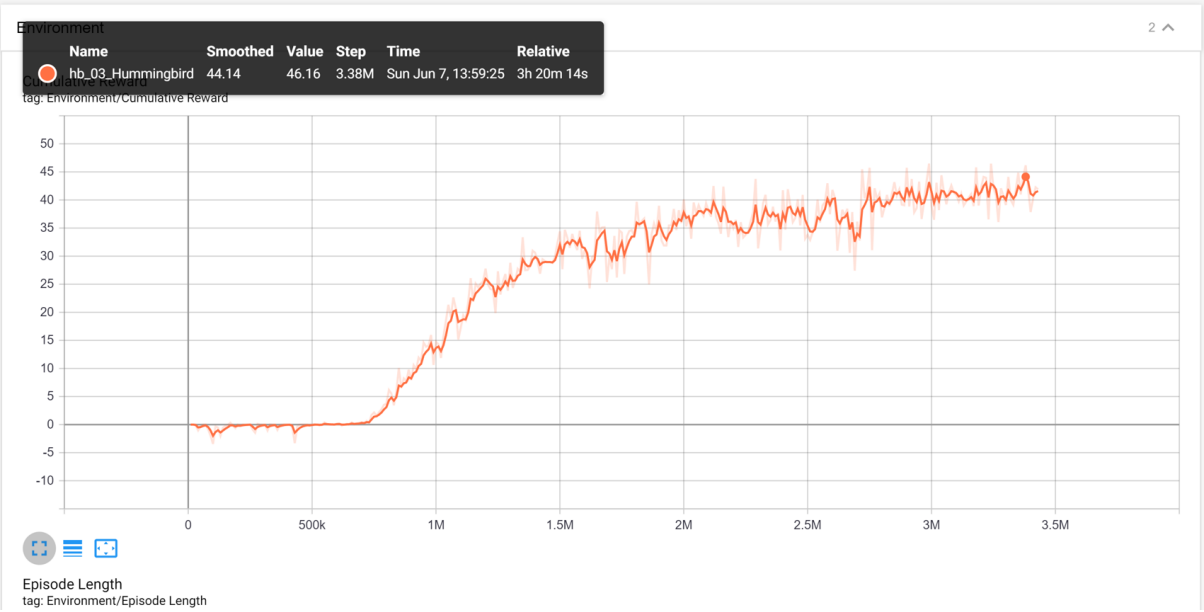
6 ΠΕΙΡΑΜΑΤΑ

6.1 ΒΑΣΙΚΗ ΕΚΠΑΙΔΕΥΣΗ

Για την υλοποίηση των πειραμάτων, ο Hummingbird agent αρχικά εκπαιδεύτηκε (βασική εκπαίδευση: hb_03[1]) με Reinforcement Learning λαμβάνοντας υπόψη μόνο την επιβράβευση (extrinsic reward) που δέχεται ο agent από το περιβάλλον του. *[Υπενθύμιση: Το hummingbird επιβραβεύεται κατά +0.01 για κάθε στιγμή που τρώει νέκταρ και αποθαρύνεται με -0.5 όταν βρεθεί στα όρια του νησιού]*.

Κατά την διάρκεια μίας εκπαίδευσης η πλατφόρμα ML-Agents αποθηκεύει στατιστικά που αφορούν την εκπαίδευση τα οποία μπορούν να αναπαρασταθούν μέσω του TensorBoard [50]. Έτσι, το tensorboard διάγραμμα που ακολουθεί παρουσιάζει την εξέλιξη του mean reward που κατακτήθηκε σε κάθε επεισόδιο της βασικής εκπαίδευσης (hb_03)

[1] Σημείωση: Στο τέλος κάθε εκπαίδευσης, ένα μοντέλο παράγεται και αποθηκεύεται για μελλοντική χρήση ή αναφορά ή επίδειξη της εξέλιξης της εκπαίδευσης μέσω του tensorboard. Το μοντέλο ονοματίζεται ως hb_XX, όπου XX μία ένδειξη που χαρακτηρίζει μοναδιαία το πείραμα



Διάγραμμα 1 - Humming bird - Βασική εκπαίδευση

Παρατηρώντας το Διάγραμμα 1, φαίνεται ότι με το μοντέλο της βασικής εκπαίδευσης το hummingbird έχει εκπαιδευτεί να επισκέπτεται κατά μέσο όρο ~46 λουλούδια ανά επεισόδιο διάρκειας περίπου 1 λεπτού.

Έχοντας το πρώτο εκπαιδευμένο μοντέλο hummingbird (hb_03) καταγράφηκε ένα αρχείο επίδειξης (βασικό demo - humming03.demo) που θα χρησιμοποιηθεί για την υλοποίηση των πειραμάτων που ακολουθούν.

Σχεδιάστηκαν και πραγματοποιήθηκαν 5 τύποι πειραμάτων:

6.2 Πειραματα Α – Ευρεση αποδοτικοτερου μοντελου (Benchmarking)

Σκοπός του πειράματος ήταν η βελτιστοποίηση του βασικού μοντέλου έτσι ώστε να βρεθεί εκείνο το μοντέλο που παρουσιάζει την καλύτερη απόδοση. Κατά την διάρκεια του πειράματος εκτός της χρήσης Reinforcement Learning (extrinsic reward) χρησιμοποιήθηκαν και οι τεχνικές εκπαίδευσης GAIL, BC και Curiosity σε διάφορους συνδυασμούς όπως φαίνεται στον πίνακα που ακολουθεί:

Πίνακας 2 - Πειράματα για την εύρεση αποδοτικότερου μοντέλου:

Run No	BC[1]	Gail[2]	Curiosity[3]	RL[4]	Μοντέλο που παρήχθηκε
1-3				x	hb-03
4	x	x	x	x	hb-04
5-6	x	x			hb-05, hb-06
7	x	x	x		hb-07
8		x		x	hb-08
9		x	x	x	hb-09
10	x			x	hb-10
11	x		x	x	hb-11
12	x	x		x	hb-12
13[5]	x	x	x	x	hb-13
14[6]	x	x	x	x	hb-14

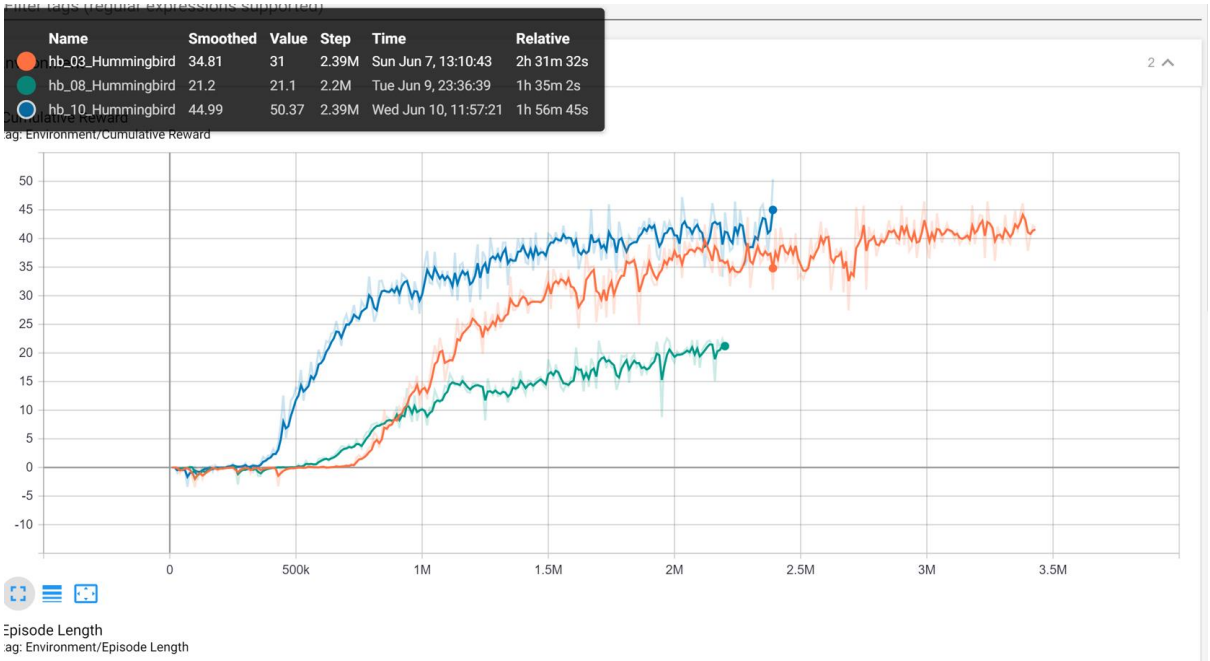
[1] Parameters used: strength: 0.5, steps: 150000

(2) Parameters used: strength: 0.01, gamma: 0.99, encoding_size: 128
(3) Parameters used: strength: 0.02, gamma: 0.99, encoding_size: 256
(4) Parameters used: strength: 1.0, gamma: 0.99
(5) Parameters used: strength: 0.01, gamma: 0.99, encoding_size: 128, use_actions=true
(6) Parameters used: strength: 0.01, gamma: 0.99, encoding_size: 128, user_actions=true, use_vail=true

Τα tensorboard διαγράμματα που ακολουθούν παρουσιάζουν τα αποτελέσματα των πειραμάτων. Πιο συγκεκριμένα, το Διάγραμμα 2 παρουσιάζει συγκριτικά τις αποδόσεις 3 μοντέλων:

●	hb_03	Μόνο RL (extrinsic reward)
●	hb_08	RL + GAIL
●	hb_10	RL + BC

Με το μοντέλο hb_10 (RL + BC) το hummingbird έχει εκπαιδευτεί να επισκέπτεται κατά μέσο όρο ~50 λουλούδια ανά επεισόδιο διάρκειας περίπου 1 λεπτού.

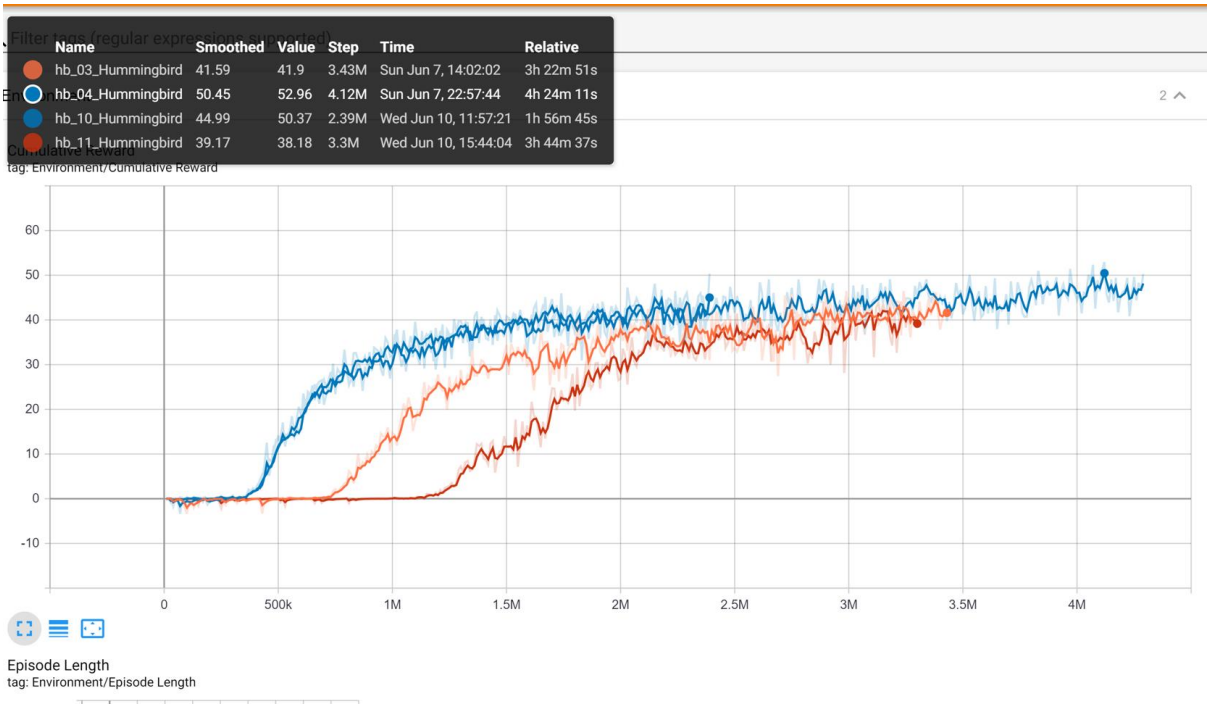


Διάγραμμα 2 – Hummingbird trained models, where hb_03 is based only on RL (extrinsic reward), hb_08 on RL + GAIL and hb_10 on RL + BC

Το Διάγραμμα 3 παρουσιάζει συγκριτικά τις αποδόσεις 4 μοντέλων:

●	hb_03	Μόνο RL (extrinsic reward)
●	hb_10	RL + BC
●	hb_11	RL + BC + Curiosity
●	hb_04	RL + BC + GAIL + Curiosity

Με το μοντέλο hb_04 (RL + BC + Gail + Curiosity) το hummingbird έχει εκπαιδευτεί να επισκέπτεται κατά μέσο όρο ~53 λουλούδια ανά επεισόδιο διάρκειας περίπου 1 λεπτού. Επίσης τα μοντέλα hb_04 και hb_10 παρουσιάζουν παρόμοια συμπεριφορά και απόδοση.

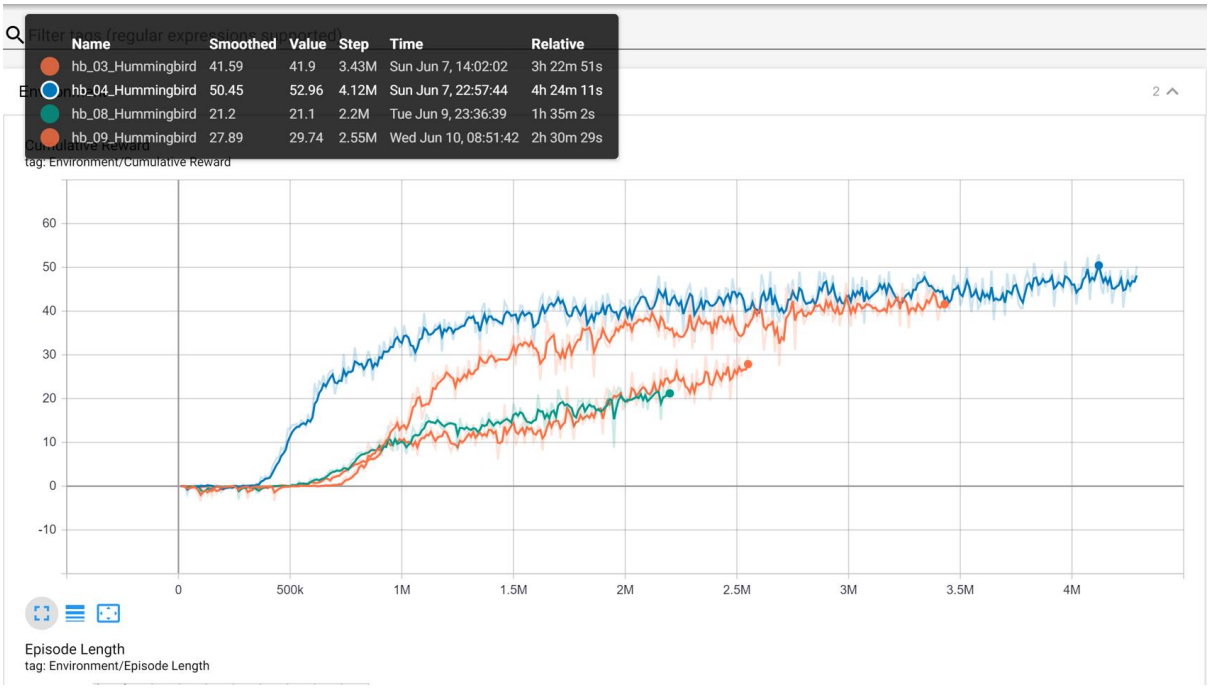


Διάγραμμα 3 - Hummingbird trained models, where hb_03 is based only on RL (extrinsic reward), hb_10 on RL + BC, hb_11 on RL + BC + Curiosity and hb_04 on RL + BC + GAIL + Curiosity

Το Διάγραμμα 4 παρουσιάζει συγκριτικά τις αποδόσεις 4 μοντέλων:

●	hb_03	Μόνο RL (extrinsic reward)
●	hb_08	RL + GAIL
●	hb_09	RL + GAIL + Curiosity
●	hb_04	RL + BC + GAIL + Curiosity

Το μοντέλο hb_04 (RL + BC + Gail + Curiosity) με μέσο όρο ~53 λουλούδια ανά επεισόδιο διάρκειας περίπου 1 λεπτού έχει και πάλι την καλύτερη επίδοση.

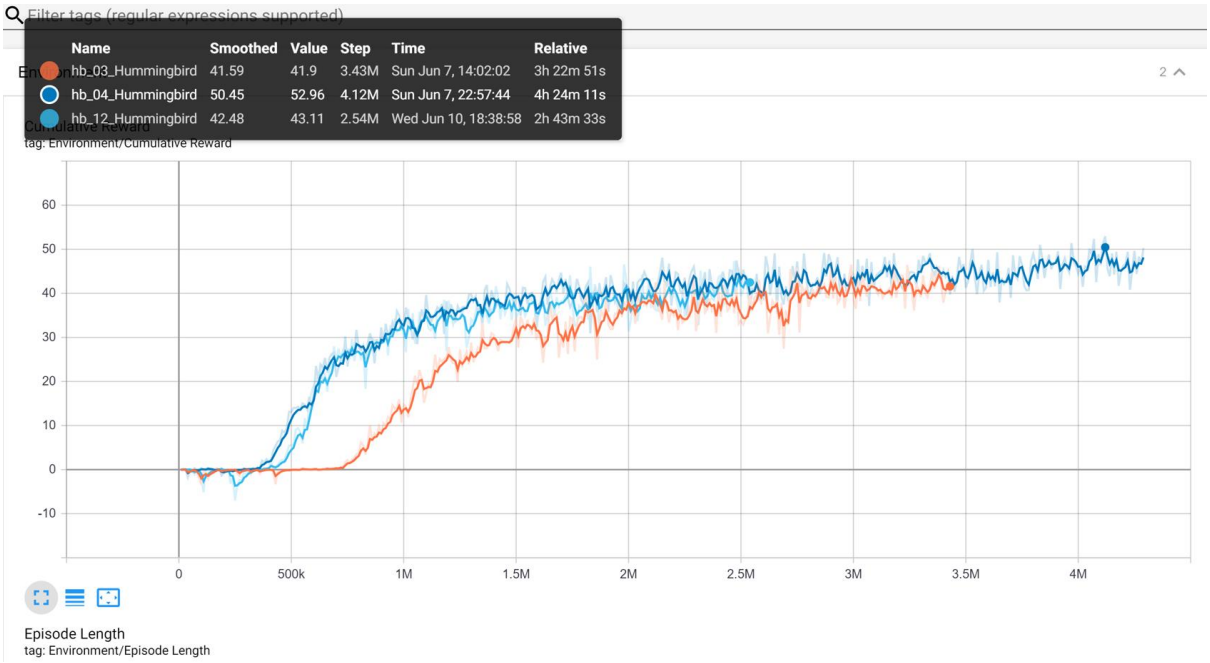


Διάγραμμα 4 - Hummingbird trained models, where hb_03 is based only on RL (extrinsic reward), hb_08 on RL + GAIL, hb_11 on RL + GAIL + Curiosity and hb_04 on RL + BC + GAIL + Curiosity

Το Διάγραμμα 5 παρουσιάζει συγκριτικά τις αποδόσεις 3 μοντέλων:

●	hb_03	Μόνο RL (extrinsic reward)
●	hb_12	RL + BC + GAIL
●	hb_04	RL + BC + GAIL + Curiosity

Το μοντέλο hb_04 (RL + BC + Gail + Curiosity) με μέσο όρο ~53 λουλούδια ανά επεισόδιο διάρκειας περίπου 1 λεπτού έχει και πάλι την καλύτερη επίδοση. Επίσης τα μοντέλα hb_04 και hb_12 παρουσιάζουν παρόμοια συμπεριφορά και απόδοση.

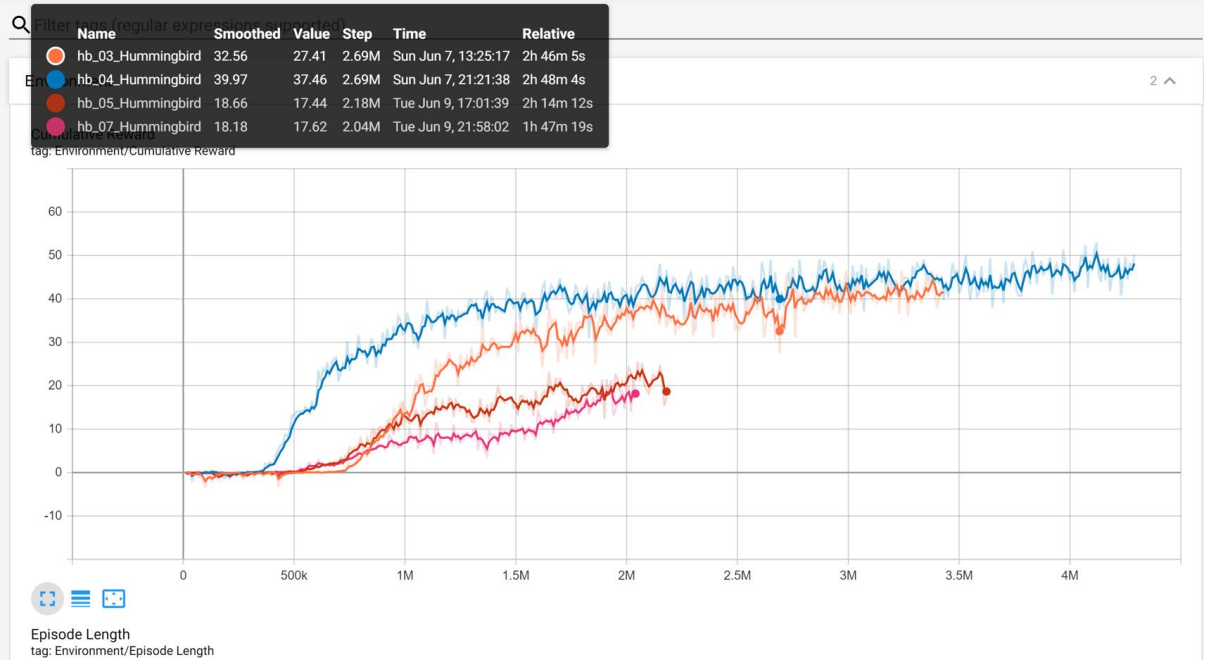


Διάγραμμα 5 - Hummingbird trained models, where hb_03 is based only on RL (extrinsic reward), hb_12 on RL + BC + GAIL, and hb_04 on RL + BC + GAIL + Curiosity

Το Διάγραμμα 6 παρουσιάζει συγκριτικά τις αποδόσεις 4 μοντέλων:

●	hb_03	Μόνο RL (extrinsic reward)
●	hb_05	BC + GAIL
●	hb_07	BC + GAIL + Curiosity
●	hb_04	RL + BC + GAIL + Curiosity

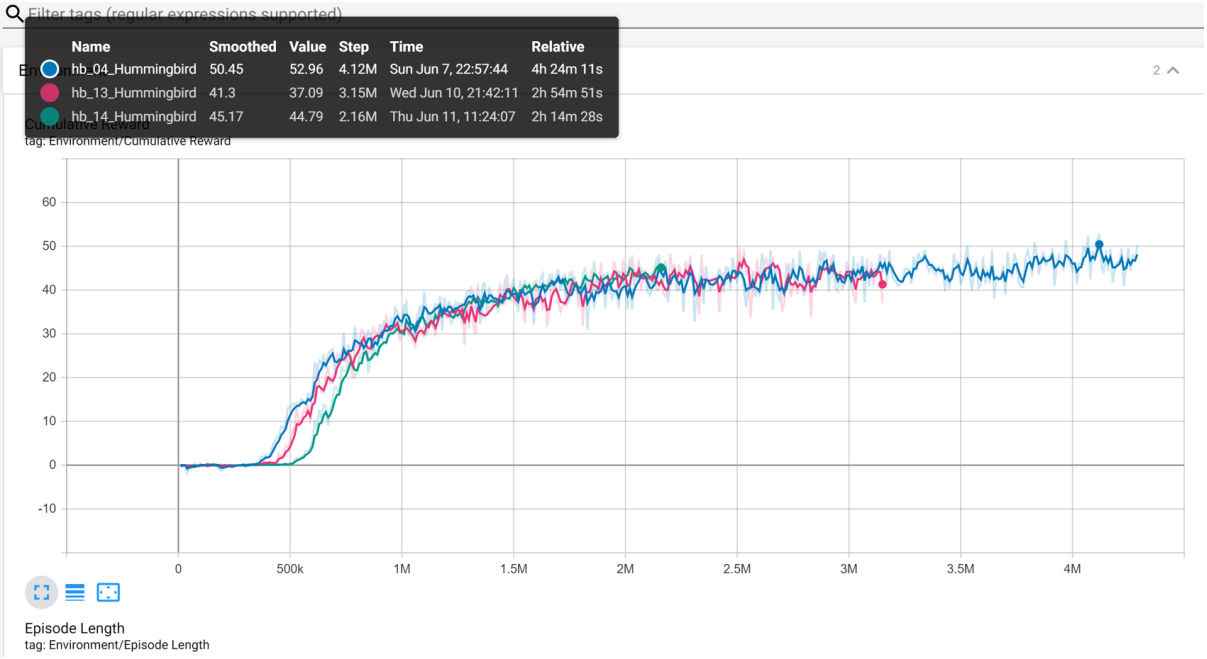
Το μοντέλο hb_04 (RL + BC + Gail + Curiosity) με μέσο όρο ~53 λουλούδια ανά επεισόδιο διάρκειας περίπου 1 λεπτού έχει και πάλι την καλύτερη επίδοση. Επίσης τα μοντέλα hb_04 και hb_12 παρουσιάζουν παρόμοια συμπεριφορά και απόδοση.



Διάγραμμα 6 - Hummingbird trained models, where hb_03 is based only on RL (extrinsic reward), hb_05 on BC + GAIL, hb_07 on BC + GAIL + Curiosity, and hb_04 on RL + BC + GAIL + Curiosity

Προκύπτει λοιπόν ότι το μοντέλο hb_04 που εμπεριέχει και τις 4 τεχνικές εκπαίδευσης (RL + BC + GAIL + Curiosity) έχει την καλύτερη απόδοση. Το Διάγραμμα 7 που ακολουθεί παρουσιάζει τις αποδόσεις 2 επιπλέον μοντέλων, των hb_13 και hb_14 παρόμοιων με το μοντέλο hb_04 αλλά με διαφορετικές παραμέτρους εκπαίδευσης. Οι αποδόσεις είναι παραπλήσιες.

●	hb_04	RL + BC + GAIL + Curiosity
●	hb_13	RL + BC + GAIL + Curiosity Parameter(s) set: use_actions=true
●	hb_14	RL + BC + GAIL + Curiosity Parameter(s) set: use_actions=true use_vail=true



Διάγραμμα 7 - Hummingbird trained models, where hb_13 and hb_14 are similar to hb_04 (based on RL + BC + GAIL + Curiosity) with slightly different training parameters

Συγκεντρωτικά οι αποδόσεις των μοντέλων που εκπαιδεύτηκαν στα πλαίσια του τρέχοντος πειράματος είναι:

Πίνακας 3 - Αποδόσεις μοντέλων πειραμάτων κατηγορίας Α

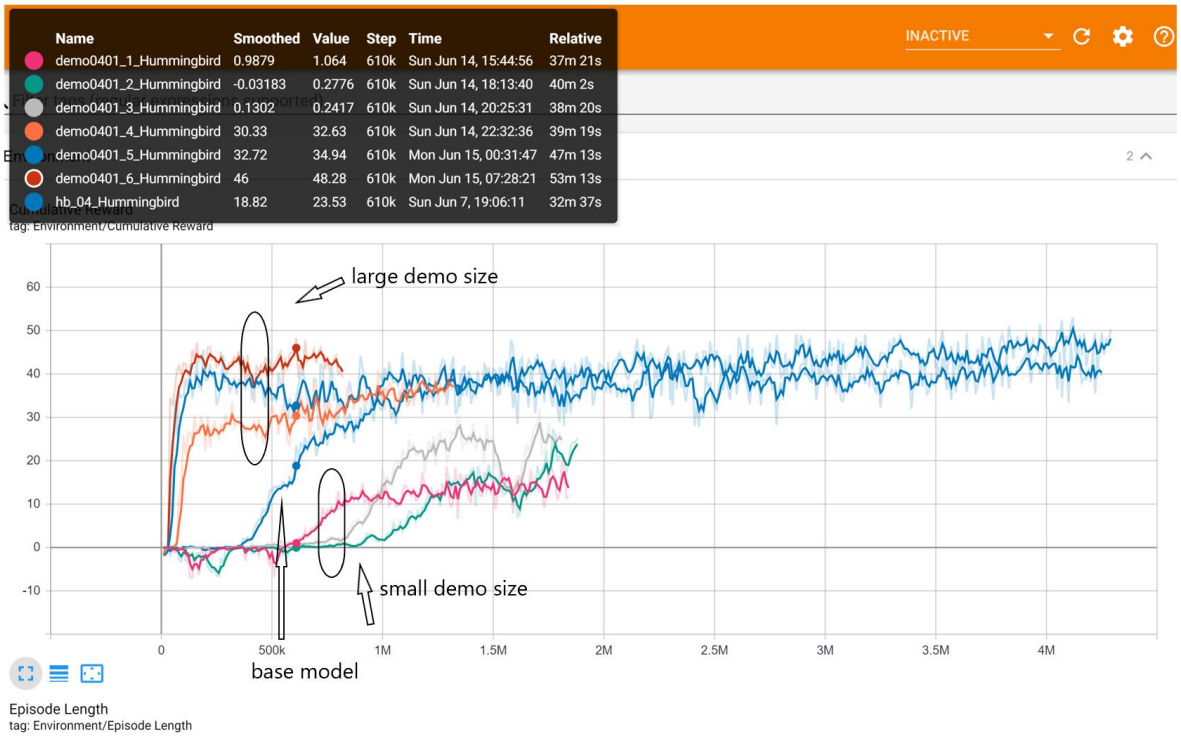
Μοντέλο που παρήχθηκε	**BC**	**Gail**	**Curiosity**	**RL**	**Mean Reward**
hb-06	x	x			16
hb-07	x	x	x		18
hb-08		x		x	22
hb-05	x	x			25
hb-09		x	x	x	30
hb-11	x		x	x	42
hb-14	x	x	x	x	45
hb-03				x	46
hb-12	x	x		x	46
hb-13	x	x	x	x	49
hb-10	x			x	50
hb-04	x	x	x	x	53

6.3 ΠΕΙΡΑΜΑΤΑ Β – ΕΠΙΔΟΣΗ ΕΚΠΑΙΔΕΥΣΗΣ ΜΕ ΒΑΣΗ ΜΕΓΕΘΟΣ ΑΡΧΕΙΟΥ ΕΠΙΔΕΙΞΗΣ

Σκοπός του πειράματος Β ήταν να παρατηρηθεί αν επηρεάζεται η επίδοση της εκπαίδευσης από το μέγεθος του αρχείου επίδειξης. Η διαδικασία που ακολουθήθηκε είναι η εξής: Επιλέχθηκε το μοντέλο hb_04 που παρουσίασε την καλύτερη επίδοση (expert trainer) κατά την διάρκεια του Πειράματος Α ως **μοντέλο βάσης** και παρήχθησαν έξι διαφορετικά αρχεία επίδειξης της συμπεριφοράς του expert trainer διαφορετικής διάρκειας. Στην συνέχεια πραγματοποιήθηκαν έξι νέες εκπαιδεύσεις (trainee training) μία για κάθε διαφορετικό αρχείο επίδειξης με τον ίδιο όμως συνδυασμό μεθόδων εκπαίδευσης: BC και GAIL.

Run No	BC	Gail	demo	Size (KB)	model
1 ●	x	x	demo0401_1	323	demo0401_1
2 ●	x	x	demo0401_2	1245	demo0401_2
3 ●	x	x	demo0401_3	4646	demo0401_3
4 ●	x	x	demo0401_4	11786	demo0401_4
5 ●	x	x	demo0401_5	23422	demo0401_5
6 ●	x	x	demo0401_6	50600	demo0401_6

Το Διάγραμμα 8 που ακολουθεί παρουσιάζει τα αποτελέσματα των πειραμάτων. Προκύπτει ότι τα μοντέλα που εκπαιδεύτηκαν με μικρού μεγέθους αρχεία επίδειξης είχαν χειρότερη συμπεριφορά και επίδοση από ότι το μοντέλο βάσης που παρήγαγε τα αρχεία. Σε αντιδιαστολή, τα μοντέλα που εκπαιδεύτηκαν με μεγάλου μεγέθους αρχεία είχαν καλύτερη επίδοση και εκπαιδεύτηκαν πιο γρήγορα από το μοντέλο βάσης.



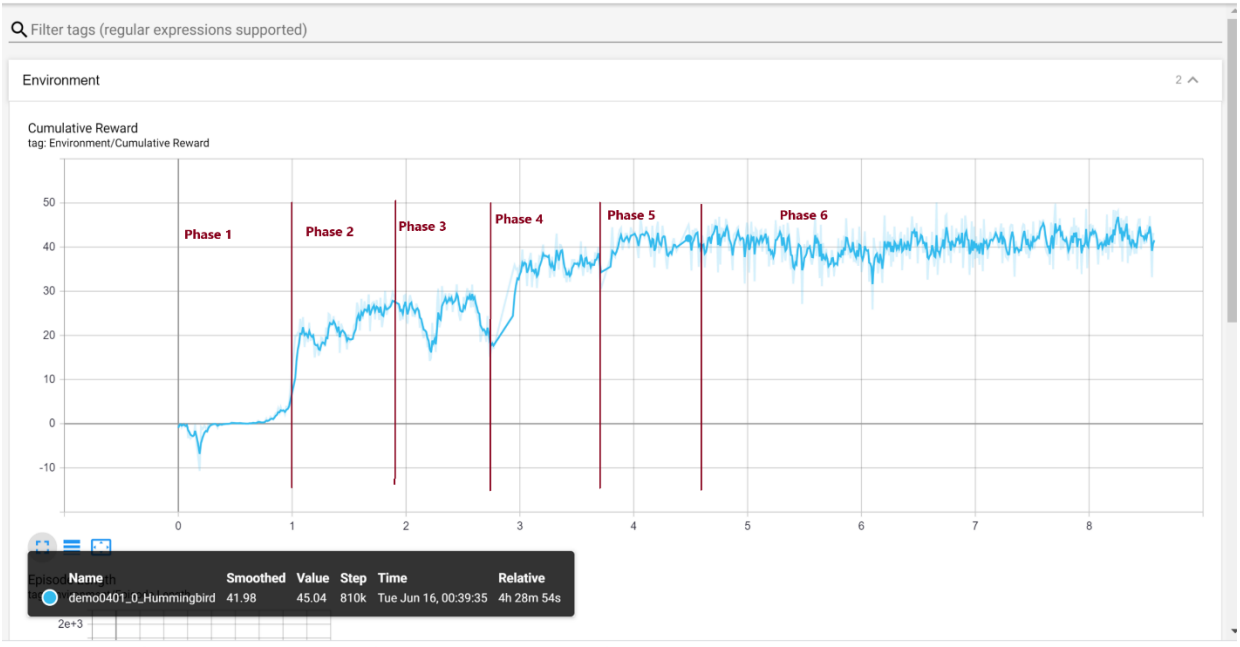
Διάγραμμα 8 – Αποτελέσματα Πειράματος Β

6.4 Πειραματα Γ – Εξελιξη εκπαιδευσης με μεταβλητο μεγεθος αρχειου επιδειξης

Σκοπός του πειράματος Γ ήταν να παρατηρηθεί αν κατά την διάρκεια της εκπαίδευσης ενός μοντέλου η πρόσβαση σε μεγαλύτερου μεγέθους αρχείων επίδειξης βελτιώνει την επίδοση της εκπαίδευσης. Η διαδικασία που ακολουθήθηκε είναι η εξής: Πραγματοποιήθηκε μία νέα εκπαίδευση με συνδυασμό μεθόδων BC και GAIL (όπως και στα πειράματα Β) σε έξι διαδοχικές φάσεις διάρκειας περίπου 1 ώρας όπου σε κάθε φάση οι μέθοδοι BC και GAIL είχαν πρόσβαση σε αρχεία επίδειξης διαφορετικού μεγέθους κάθε φορά (Σημείωση: χρησιμοποιήθηκαν τα αρχεία επίδειξης που παρήχθησαν στα πλαίσια του πειράματος Β)

Phase	BC	Gail	demo	Model (used)	Start	End	Model produced
1	x	x	demo0401_1	demo0401_1	20:10	21:09	demo0401_0
2	x	x	demo0401_2	demo0401_2	21:12	22:04	demo0401_0
3	x	x	demo0401_3	demo0401_3	22:05	22:56	demo0401_0
4	x	x	demo0401_4	demo0401_4	23:05	23:53	demo0401_0
5	x	x	demo0401_5	demo0401_5	23:54	00:33	demo0401_0
6	x	x	demo0401_6	demo0401_6	00:38	04:44	demo0401_0

Το Διάγραμμα 9 που ακολουθεί παρουσιάζει τα αποτέλεσμα του πειράματος.



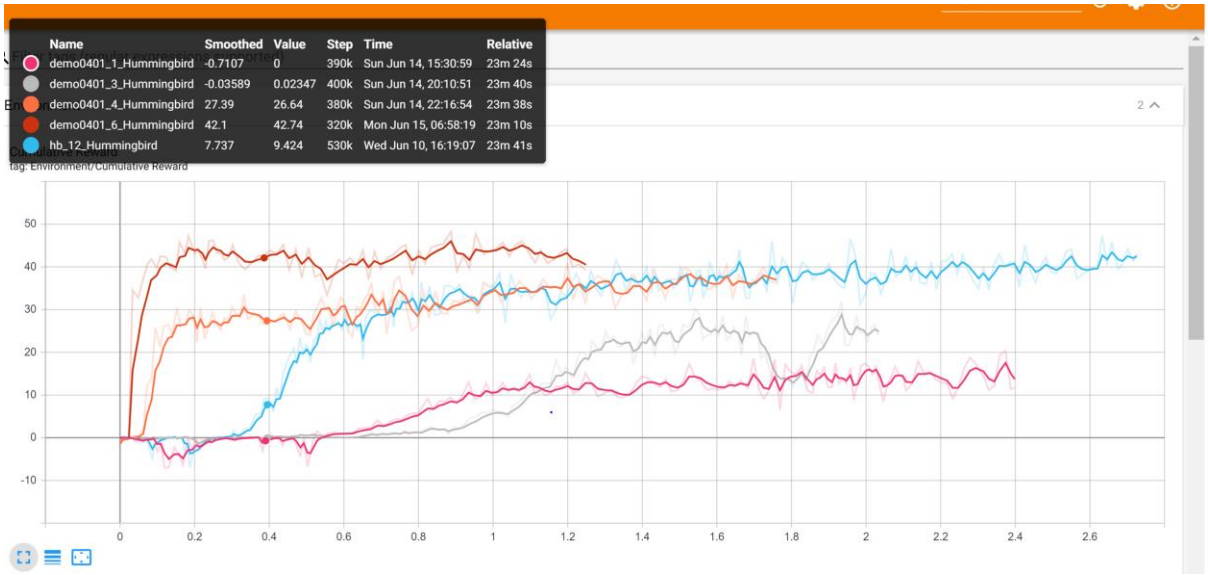
Διάγραμμα 9 - Αποτελέσματα Πειράματος Γ

6.5 Πειραματα Δ – Επιδοση εκπαιδευσης με βαση το Επιπεδο δεξιοτητας του Εκπαιδευτη

Σκοπός του πειράματος Δ ήταν να παρατηρηθεί αν κατά την διάρκεια της εκπαίδευσης η πρόσβαση του εκπαιδευόμενου σε αρχεία επίδειξης μεγαλύτερης δεξιότητας βελτιώνει την επίδοση της εκπαίδευσης. Η διαδικασία που ακολουθήθηκε είναι η εξής: Από τα μοντέλα των πειραμάτων Α και Β επιλέχθηκαν πέντε μοντέλα που αντιστοιχούν σε πέντε διακριτά επίπεδα δεξιότητας (naive, low, middle, good, expert) του εκπαιδευτή – μοντέλου. Κάθε διακριτό μοντέλο παρήγαγε ένα αρχείο επίδειξης ίσης (περίπου) διάρκειας (~ 5000KB).

Skill	**demo**	**Model (used)**	**Start**	**End**	**Model produced**
Naïve	demo04011	demo04_01	19:49	20:47	hb_D
Low	demo04013	demo04_03	20:49	21:46	hb_D
Middle	demo04014	demo04_04	22:08	23:03	hb_D
Good	hb12	hb_12	23:08	00:03	hb_D
Expert	demo04016	demo04_06	24:05	05:05	hb_D

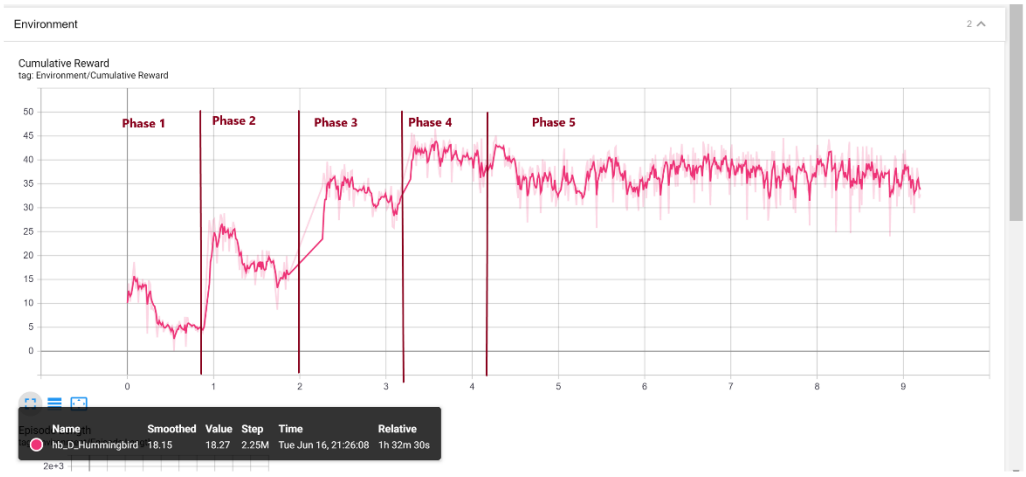
Οι επιδόσεις των μοντέλων διαφορετικής δεξιότητας που επιλέχθηκαν παρουσιάζονται στο Διάγραμμα 10.



Διάγραμμα 10 - Πείραμα Δ – Μοντέλα διαφορετικής δεξιότητας

Στην συνέχεια πραγματοποιήθηκε εκπαίδευση με συνδυασμό μεθόδων BC και GAIL ενός νέου μοντέλου σε πέντε διαδοχικές φάσεις διάρκειας περίπου 1 ώρας.

Το Διάγραμμα 11 που ακολουθεί παρουσιάζει τα αποτέλεσμα του πειράματος όπου παρατηρείται ότι η δεξιότητα του μοντέλου που παράγει το αρχείο επίδειξης επηρεάζει άμεσα και ανάλογα την επίδοση της εκπαίδευσης.



Διάγραμμα 11 - Αποτελέσματα Πειράματος Δ

6.6 ΠΕΙΡΑΜΑΤΑ Ε – ΕΚΠΑΙΔΕΥΣΗ ΣΕ ΔΙΑΦΟΡΕΤΙΚΑ ΠΕΡΙΒΑΛΛΟΝΤΑ

Όλα τα προηγούμενα πειράματα είχαν μια σταθερή παράμετρο, το περιβάλλον στο οποίο λειτουργούσε, εκπαιδευόταν και δεχόταν ερεθίσματα ο hummingbird agent ήταν σταθερό και αμετάβλητο.

Σκοπός του πειράματος Ε ήταν να παρατηρηθεί αν η επίδοση της εκπαίδευσης του Hummingbird agent και γενικότερα ενός οποιουδήποτε μοντέλου επηρεάζεται από το περιβάλλον με το οποίο αλληλοεπιδρά το μοντέλο.

Για τον σκοπό αυτό, δημιουργήθηκαν (4) τέσσερα περιβάλλοντα, διαφορετικών επιπέδων δυσκολίας όπου εκπαιδεύτηκε ο hummingbird agent. Το 1ο περιβάλλον αποτελείται από ένα νησί με ένα μόνο φυτό (oneflower), το 2ο περιβάλλον από ένα νησί με λίγα φυτά (clusterflowers), το 3ο περιβάλλον από ένα νησί με φυτά κρυμμένα πίσω από θάμνους-βράχους (hiddenflowers) και τέλος το 4ο περιβάλλον περιλαμβάνει ένα νησί με βράχους, θάμνους, δέντρα και φυτά διάσπαρτα ανάμεσά τους (floatingisland).

Το υψηλής δεξιότητας μοντέλο hb_04 (μοντέλο – βάση που επιλέχθηκε για το πείραμα Β) παρήγαγε (4) τέσσερα αρχεία επίδειξης ένα για κάθε περιβάλλον – επίπεδο δυσκολίας. Αυτά τα αρχεία επίδειξης αποτελούν την βάση των εκπαιδεύσεων που πραγματοποιήθηκαν στα πλαίσια του τρέχοντος πειράματος και είναι περίπου ίδιου μεγέθους (~5000KB).

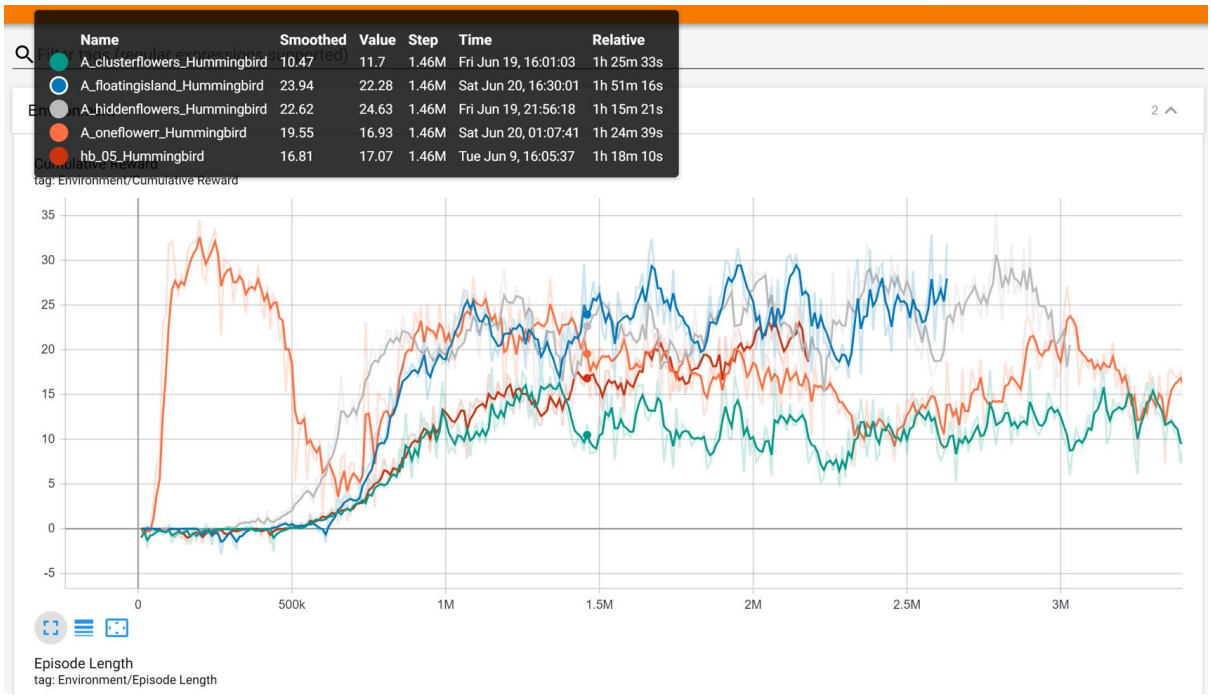
Σχεδιάστηκαν και υλοποιήθηκαν 3 διαφορετικού είδους πειράματα.

<u>Πειράματα Ε.1</u>

Στο ίδιο περιβάλλον – το πιο περίπλοκο περιβάλλον (floatingisland) πραγματοποιήθηκαν 4 διαφορετικές εκπαιδεύσεις, κάθε μία με συνδυασμό πάντα μεθόδων BC και GAIL αλλά με αρχεία επίδειξης του μοντέλου βάσης σε διαφορετικά κάθε φορά επίπεδα δυσκολίας. Για παράδειγμα, στην 1η εκπαίδευση, ο hummingbird agent αλληλοεπιδρά με το πιο περίπλοκο περιβάλλον (υψηλής δυσκολίας) αλλά χρησιμοποιείται ως αρχείο επίδειξης αυτό του πιο εύκολου περιβάλλοντος (χαμηλής δυσκολίας).

Run	Environment	demo	BC	Gail	Model produced
1	floatingisland	oneflower	x	x	A_oneflowerr
2	floatingisland	clusterflowers	x	x	A_clusterflowers
3	floatingisland	hiddenflowers	x	x	A_hiddenflowers
4	floatingisland	floatingisland	x	x	A_floatingisland
Reference model from A experiments					
-	floatingisland		x	x	hb_05

Το Διάγραμμα 12 που ακολουθεί παρουσιάζει τα αποτέλεσμα των πειραμάτων. Στο διάγραμμα προστέθηκε μοντέλο αναφοράς και η εκπαίδευση του μοντέλου hb_05 που πραγματοποιήθηκε στα πλαίσια του πειράματος Α και έκανε χρήση των BC και GAIL τεχνικών.



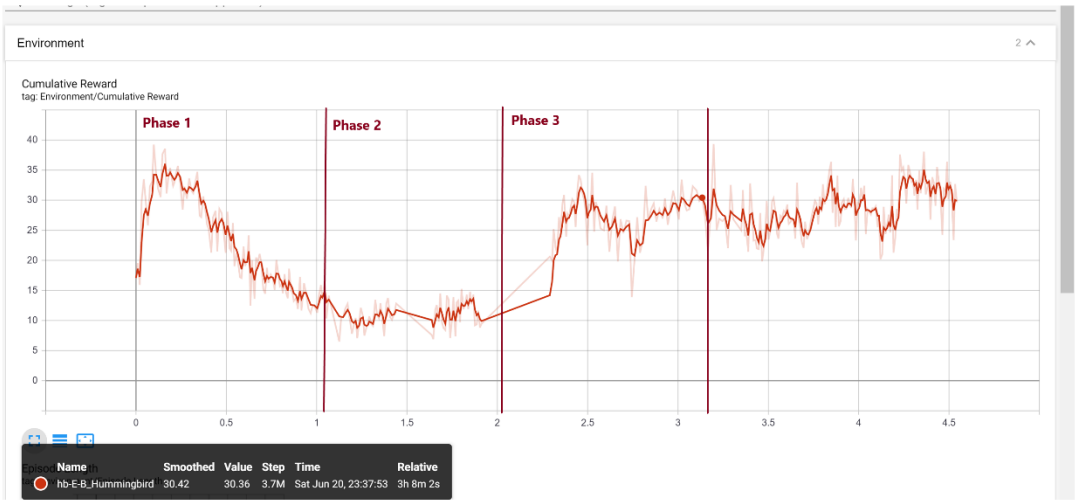
Διάγραμμα 12 - Αποτελέσματα πειράματος Ε.1

Πείραμα Ε.2

Σκοπός του πειράματος ήταν να παρατηρηθεί αν επηρεάζεται η εξέλιξη της εκπαίδευσης ενός μοντέλου από την χρήση διαφορετικού επιπέδου δυσκολίας και περιπλοκότητας αρχείων επίδειξης που προέρχονται όμως από ίδιας δεξιότητας εκπαιδευμένου μοντέλου. Για τον σκοπό αυτό πραγματοποιήθηκε εκπαίδευση ενός μοντέλου που αλληλοεπιδρά με το πιο περίπλοκο περιβάλλον σε (4) τέσσερις διαδοχικές φάσεις διάρκειας περίπου 1 ώρας. Σε κάθε φάση η εκπαίδευση έκανε χρήση διαφορετικού επιπέδου δυσκολίας αρχείου επίδειξης με συνδυασμό μεθόδων BC και GAIL.

Running environment	demo	Model (used)	Start	End	Model produced
floatingisland	oneflower	A_oneflowerr	20:28	2:34	hb_E_B
floatingisland	clusterflowers	A_clusterflowers	21:36	22:24	hb_E_B
floatingisland	hiddenflowers	A_hiddenflowers	22:45	23:49	hb_E_B
floatingisland	floatingisland	A_floatingisland	23:50	01:02	hb_E_B

Το Διάγραμμα 13 που ακολουθεί παρουσιάζει τα αποτέλεσμα του πειράματος.



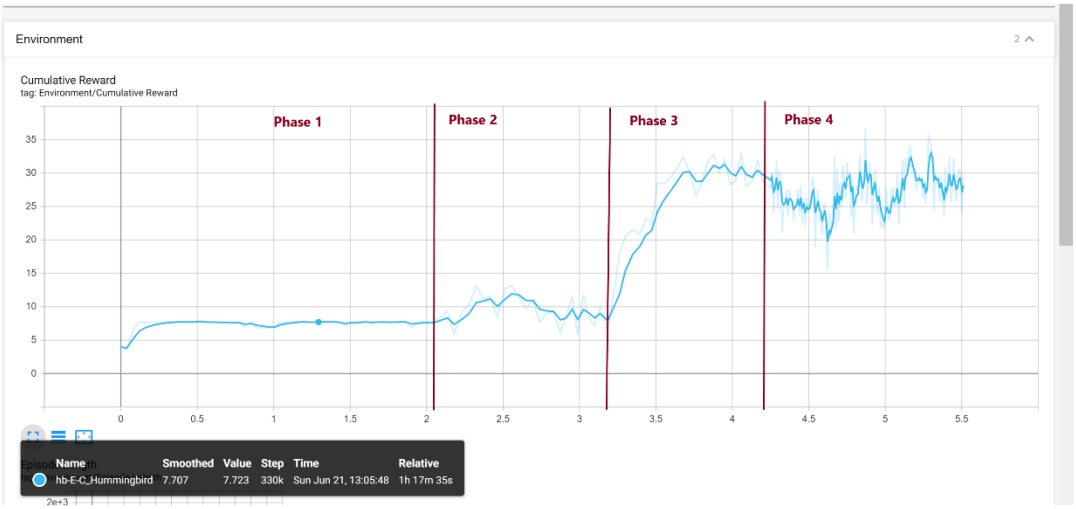
Διάγραμμα 13 - Αποτέλεσμα πειράματος E.2

Πείραμα E.3

Στα πλαίσια του πειράματος αυτού πραγματοποιήθηκε εκπαίδευση ενός μοντέλου σε (4) τέσσερις διαδοχικέςφάσεις διάρκειας περίπου 1 ώρας. Σε κάθε φάση το μοντέλο εκπαιδεύεται σε διαφορετικής δυσκολίας και πολυπλοκότητας επίπεδο (με αύξουσα πολυπλοκότητα) και με αρχείο επίδειξης αντίστοιχο του τρέχοντος περιβάλλοντος.

Running environment	demo	Model (used)	Start	End	Model produced
oneflower	oneflower	A_oneflowerr	11:44	13:53	hb_E_C
clusterflowers	clusterflowers	A_clusterflowers	13:54	14:59	hb_E_C
hiddenflowers	hiddenflowers	A_hiddenflowers	15:01	16:01	hb_E_C
floatingisland	floatingisland	A_floatingisland	16:02	17:19	hb_E_C

Το Διάγραμμα 14 που ακολουθεί παρουσιάζει τα αποτελέσματα του πειράματος.



Διάγραμμα 14 - Αποτέλεσμα πειράματος E.3

7 ΕΠΙΔΕΙΞΗ ΛΕΙΤΟΥΡΓΙΚΟΤΗΤΑΣ ΕΦΑΡΜΟΓΗΣ

7.1 ΕΓΚΑΤΑΣΤΑΣΗ ΕΦΑΡΜΟΓΗΣ

Unity ML-Agents

- Εγκατάσταση του Unity Hub 2.3.1 και στην συνέχεια το Unity 2019.3.15f1 ακολουθώντας τις οδηγίες που περιγράφονται στο https://docs.unity3d.com/Manual/GettingStartedInstallingHub.html
- Εγκατάσταση του Anaconda prompt όπως περιγράφεται στο https://docs.anaconda.com/anaconda/install/windows/
- Κατέβασμα από το https://github.com/Unity-Technologies/ml-agents του ml-agents release 2 zip αρχείου και unzip σε ένα folder με όνομα ml-agents-release-2.
- Άνοιγμα του Unity Hub, προσθήκη νέου project (Add), από το file dialog που εμφανίζεται επιλογή του Project folder κάτω από το ml-agent-release-2 folder και επιλογή Open.
- Αφού ανοίξει το Unity project, εγκατάσταση την com.unity.ml-agents βιβλιοθήκης όπως περιγράφεται στο https://github.com/Unity-Technologies/ml-agents/blob/release_2_docs/docs/Installation.md
- Στο Project windows, άνοιγμα της pyramid's scene στο Pyramids/Scenes folder
- Αναλόγως προσθήκη νέου Unity project του Hummingbird project
- Σε ένα Anaconda prompt
    - conda create -n mlagents-2 python=3.7 *[create conda virtual env]*
    - conda activate mlagents-2 *[activate conda vitual env]*
- Από ένα Windows Command prompt,
    - cd ml-agents-release-2
    - pip install -e ./ml-agents-envs
    - pip install -e ./ml-agents
    - pip install web3, flask, flask-cors

Εγκατάσταση της dApp εφαρμογής

- Κατέβασμα της dApp εφαρμογής από το https://github.com/teodav/truffle-training και unzip αυτής σε ένα folder
- Από ένα Windows Command prompt:
  - cd client\trainee
  - npm install
  - cd ..\trainer
  - npm install

Truffle – Ganache

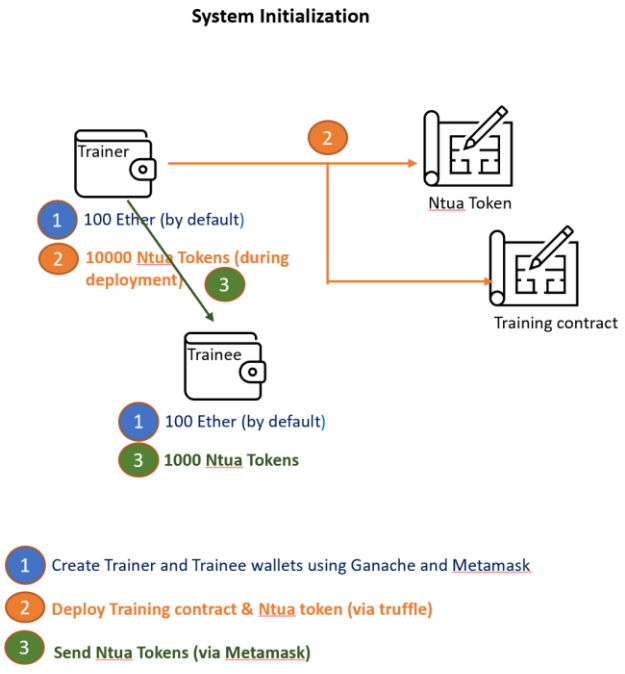
- Εγκατάσταση του Ganache και δημιουργία ενός workspace ακολουθώντας τις οδηγίες https://www.trufflesuite.com/docs/ganache/quickstart
- Στο Windows Command prompt:
  - cd ..\..\.. *[navigate to the root of the project]*
  - npm install -g truffle *[εγκατάσταση του truffle]*
  - truffle console *[άνοιγμα του truffle console]*
  - migrate *[δημοσίευση των έξυπνων συμβολαίων στο Ganache]*

MetaMask

- Εγκατάσταση του Metamask Chrome extension όπως περιγράφεται στο https://metamask.io/

7.2 Αρχικοποιηση Ethereum Λογαριασμων

Το σχήμα που ακολουθεί παρουσιάζει τα βήματα που πρέπει να ακολουθηθούν ώστε να γίνουν οι απαραίτητες αρχικοποιήσεις πριν την πρώτη χρήση της εφαρμογής



Εικόνα 19 - Αρχικοποίηση εφαρμογής

Βήμα 1 – Δημιουργία των blockchain λογαριασμών του trainer agent και trainee agent

- Αφού εγκατασταθεί το Ganache αρχικοποιεί μία λίστα άμεσα διαθέσιμων blockchain λογαριασμών με αρχικό υπόλοιπο 100 Ether ο κάθε ένας. Δύο από αυτούς τους λογαριασμούς ανατίθενται προς χρήση στους trainer agent και trainee agent. Για λόγους γρήγορης μνημόνευσης, έχει θεωρηθεί ότι ο πρώτος στην λίστα λογαριασμός ανατίθεται στον trainer agent ενώ ο τελευταίος ανατίθεται στον trainee agent.
- Χρήση του Metamask [51] ώστε να δημιουργηθεί το blockchain πορτοφόλι των λογαριασμών των trainer και trainee agents αντίστοιχα

Βήμα 2 – Εγκατάσταση των Training και Ntua Token συμβολαίων (Deploy) στο Ganache blockchain

- Χρήση της πλατφόρμας truffle (και πιο συγκεκριμένα της truffle console) για την εγκατάσταση των συμβολαίων στο Ganache blockchain.
- Τροποποίηση των Metamask πορτοφολιών των trainee και trainer ώστε εκτός του Ether να υποστηρίζουν και το Ntua Token δείχνοντας το υπόλοιπο του λογαριασμού και στα δύο tokens.

Βήμα 3 – Αποστολή στον πορτοφόλι του trainee ενός ποσού από Ntua Tokens

- Μέσω του Metamask αποστέλλονται 1000 Ntua Tokens από το πορτοφόλι του Trainer προς το πορτοφόλι του Trainee.

7.3 Εκτελεση Εφαρμογης

Unity

- Μέσω του Unity Hub, άνοιγμα του project ‘Hummingbird’
- Αφού ανοίξει το Unity project, στο Project windows, άνοιγμα της Training scene στο Assets/Hummingbird/Scenes folder

Unity ML-Agents

Σε ένα Anaconda prompt

- conda activate mlagents-2
- cd ~ cd ml-agents-release-2
- mlagents-learn ./config/gail_config.yaml --run-id=hb_04 --resume

Εκτέλεση Ganache και επιλογή του workspace

Εκτέλεση της dApp

Σε νέο windows command prompt

- cd ~ truffle_taining\server
- python server.py *[runs trainee server]*

Σε νέο windows command prompt

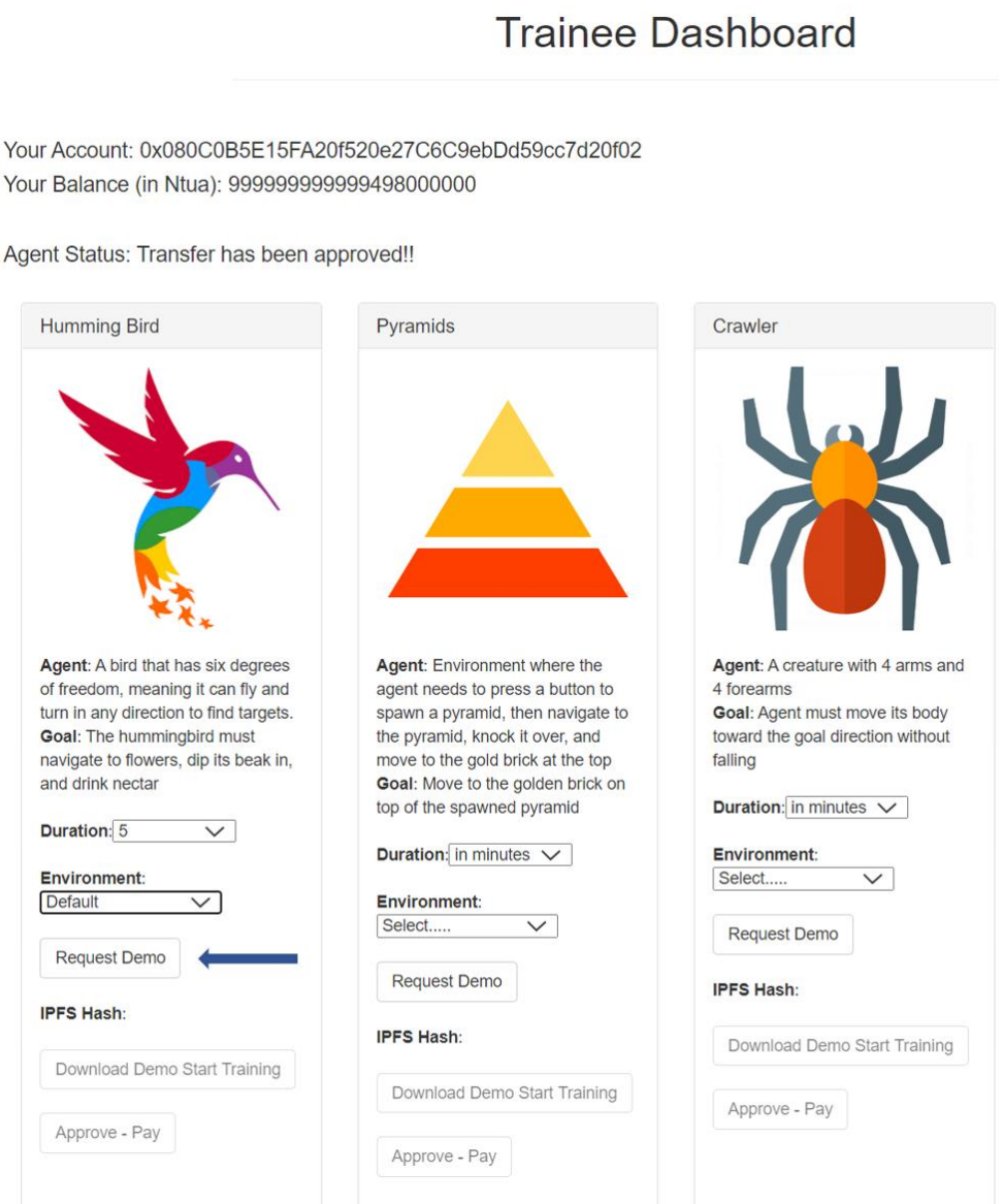
- cd ~ truffle_taining\client\trainer
- npm run dev *[trainer web app: http://localhost:3000/]*

Σε νέο windows command prompt

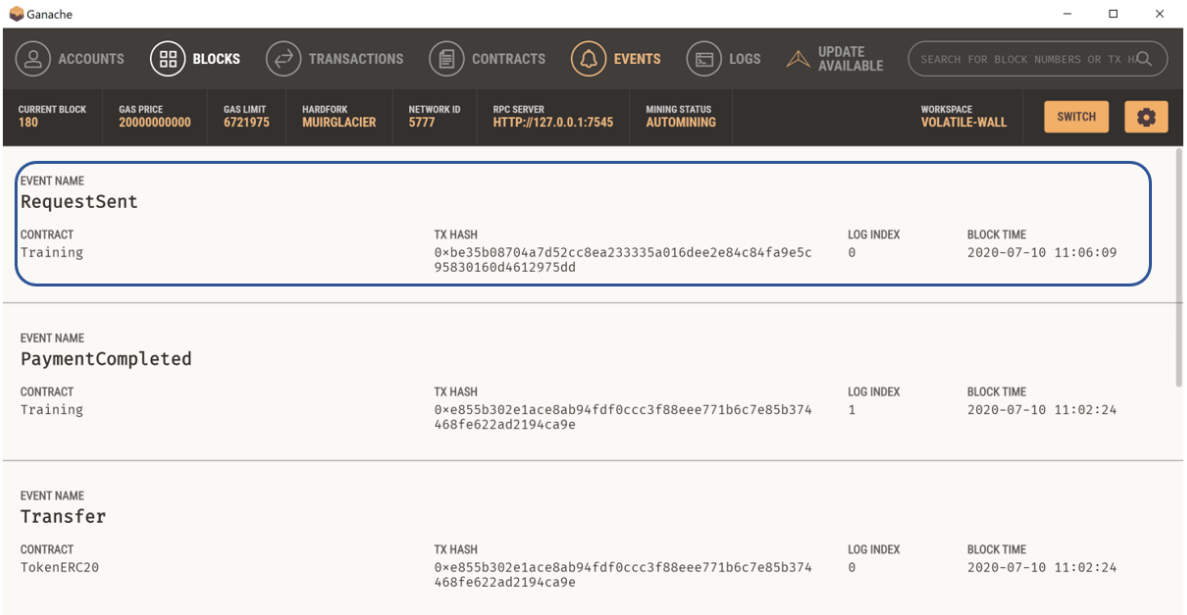
- cd ~ truffle_taining\client\trainee
- npm run dev *[trainer web app: http://localhost:3002/]*

7.4 ΛΕΙΤΟΥΡΓΙΚΟΤΗΤΑ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

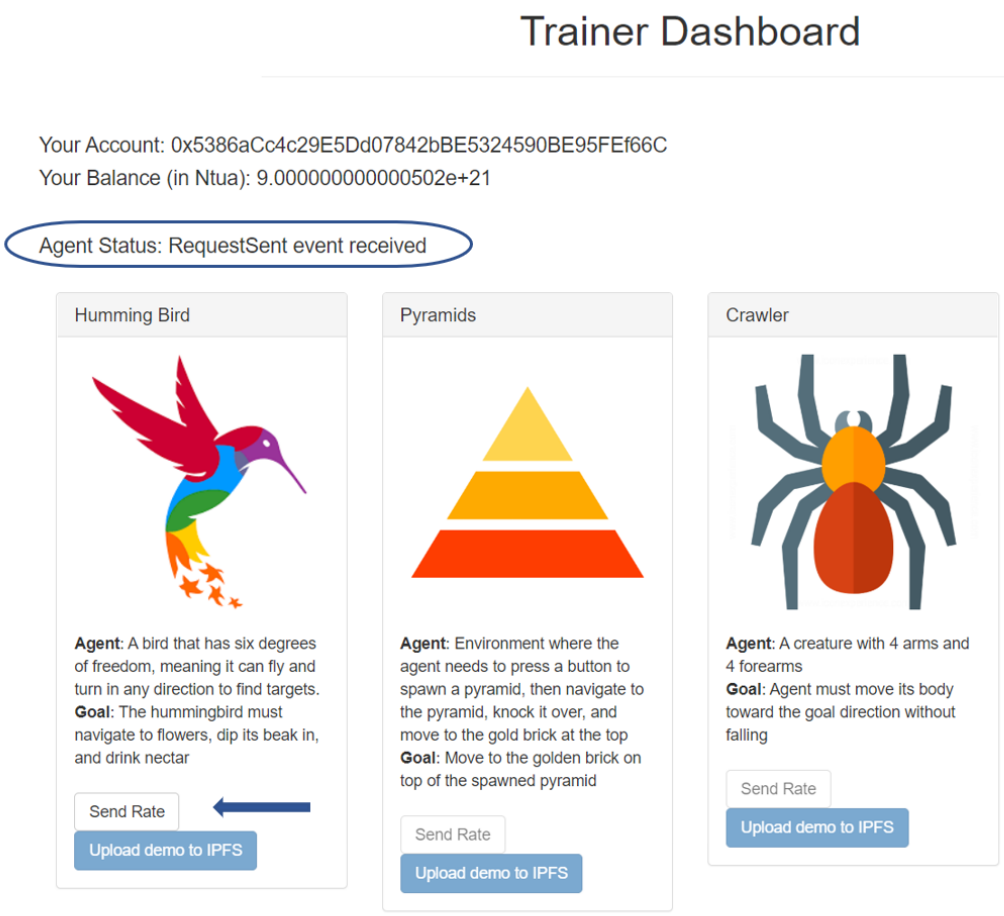
Ο trainee ζητά από το trainer μέσω του Training smart contract ένα αρχείο επίδειξης για τον Hummingbird agent.



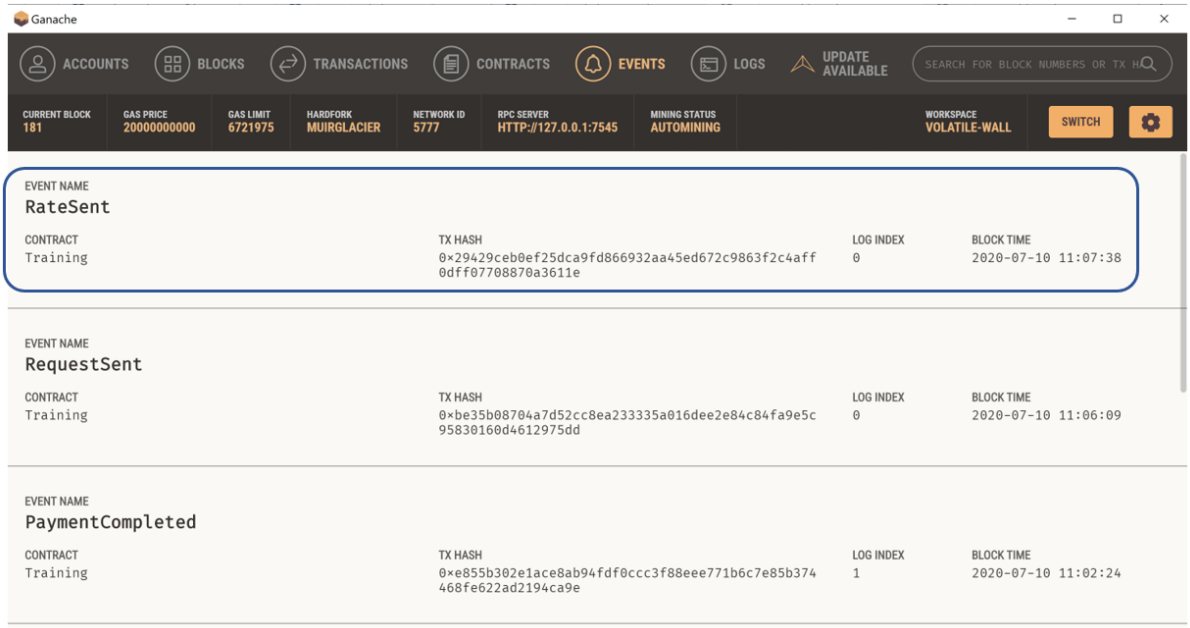
Το Ganache blockchain έχει ενημερωθεί ανάλογα:



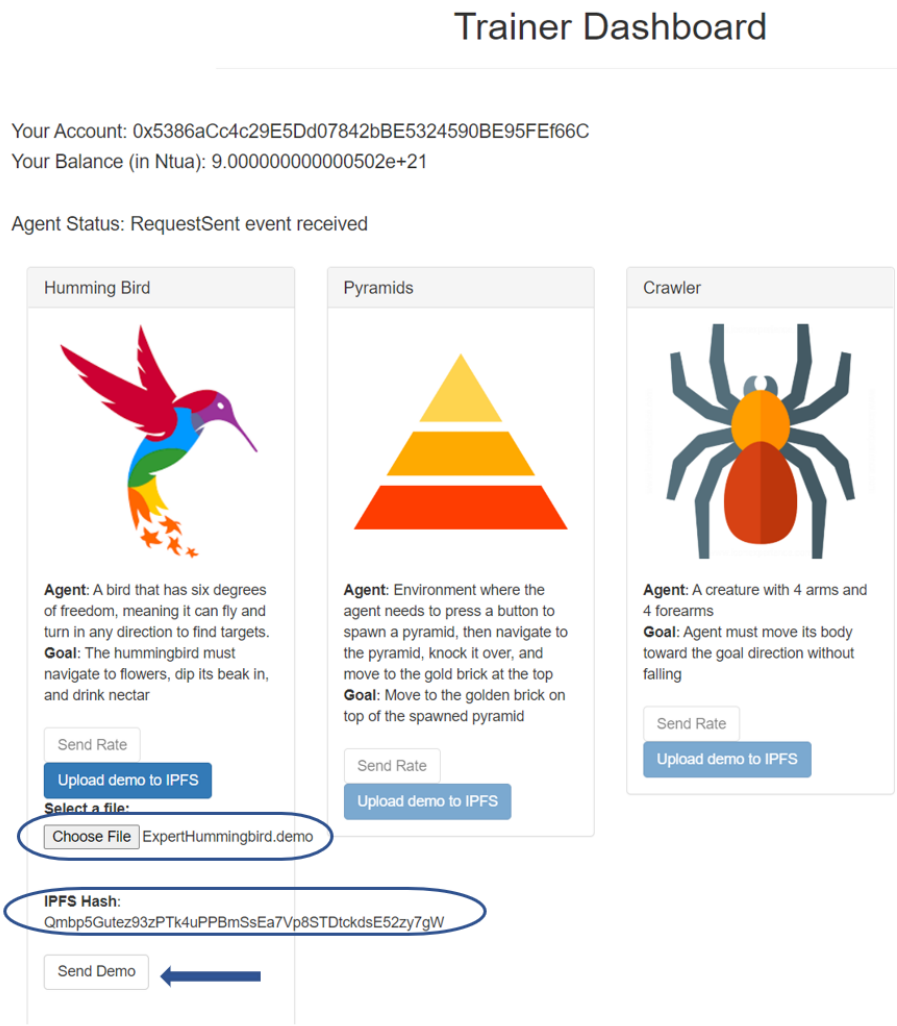
Ο Trainer δέχεται το αίτημα και αποστέλλει στον Trainee μέσω του Training contract το rate της συναλλαγής:



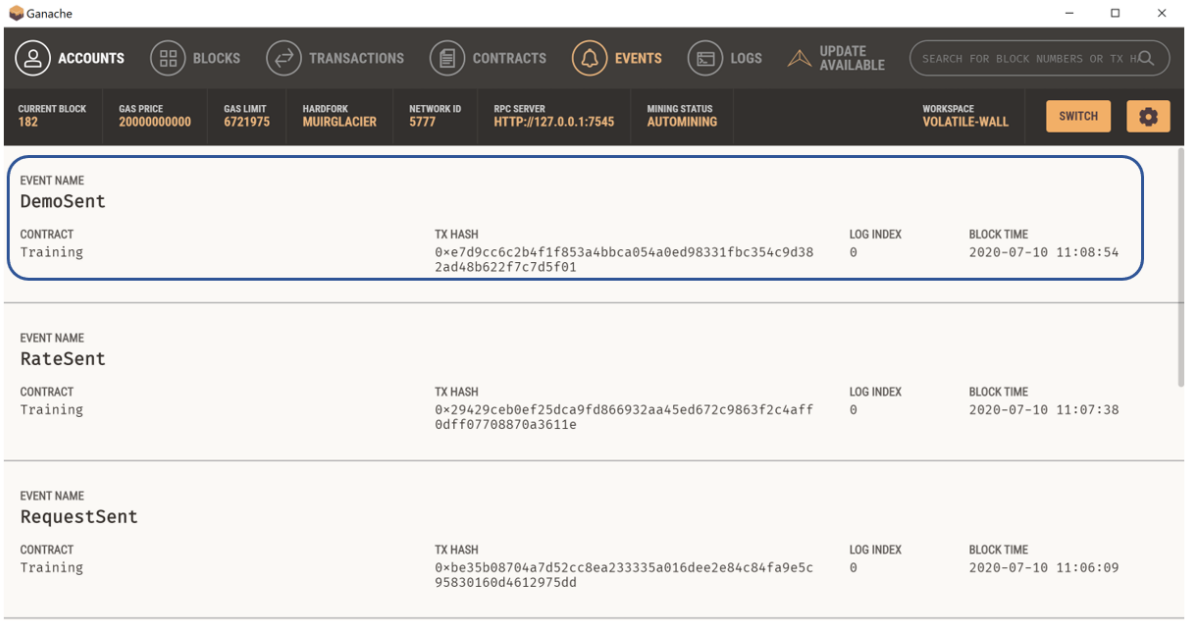
Το Ganache blockchain έχει ενημερωθεί ανάλογα:



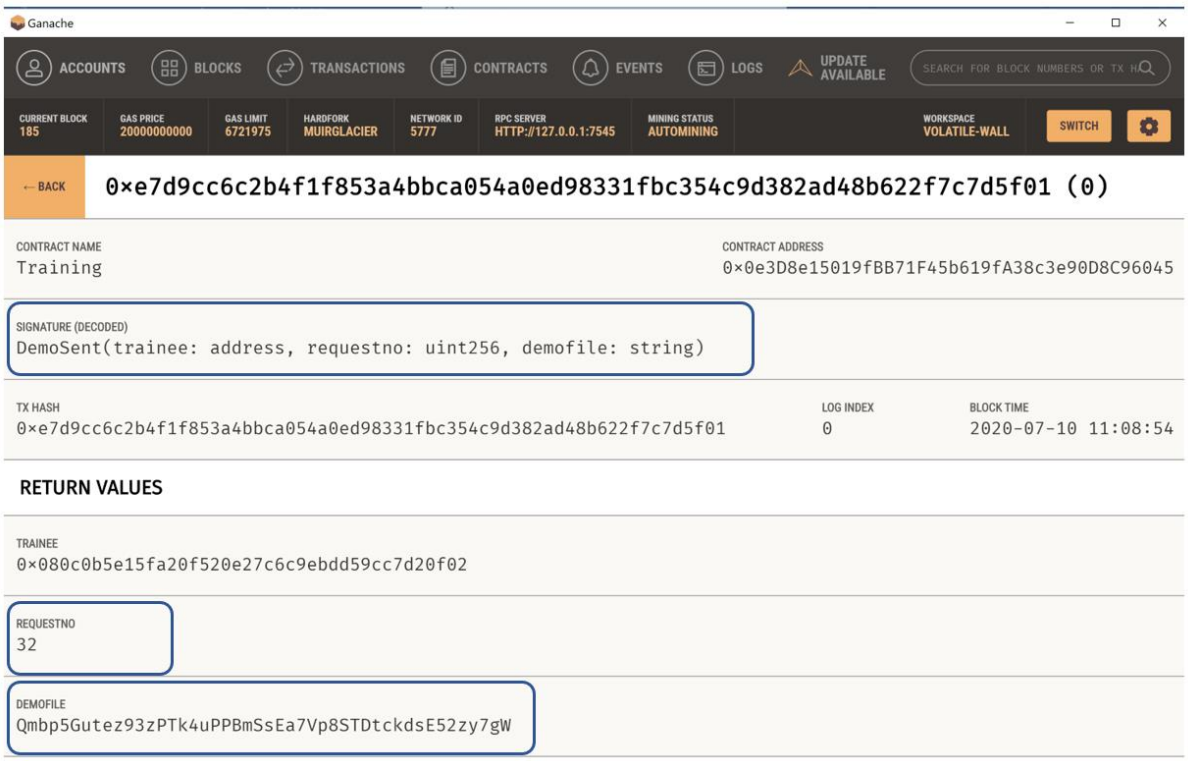
Ο Trainer αποστέλλει το κατάλληλο αρχείο επίδειξης στον Trainee μέσω του Training contract :



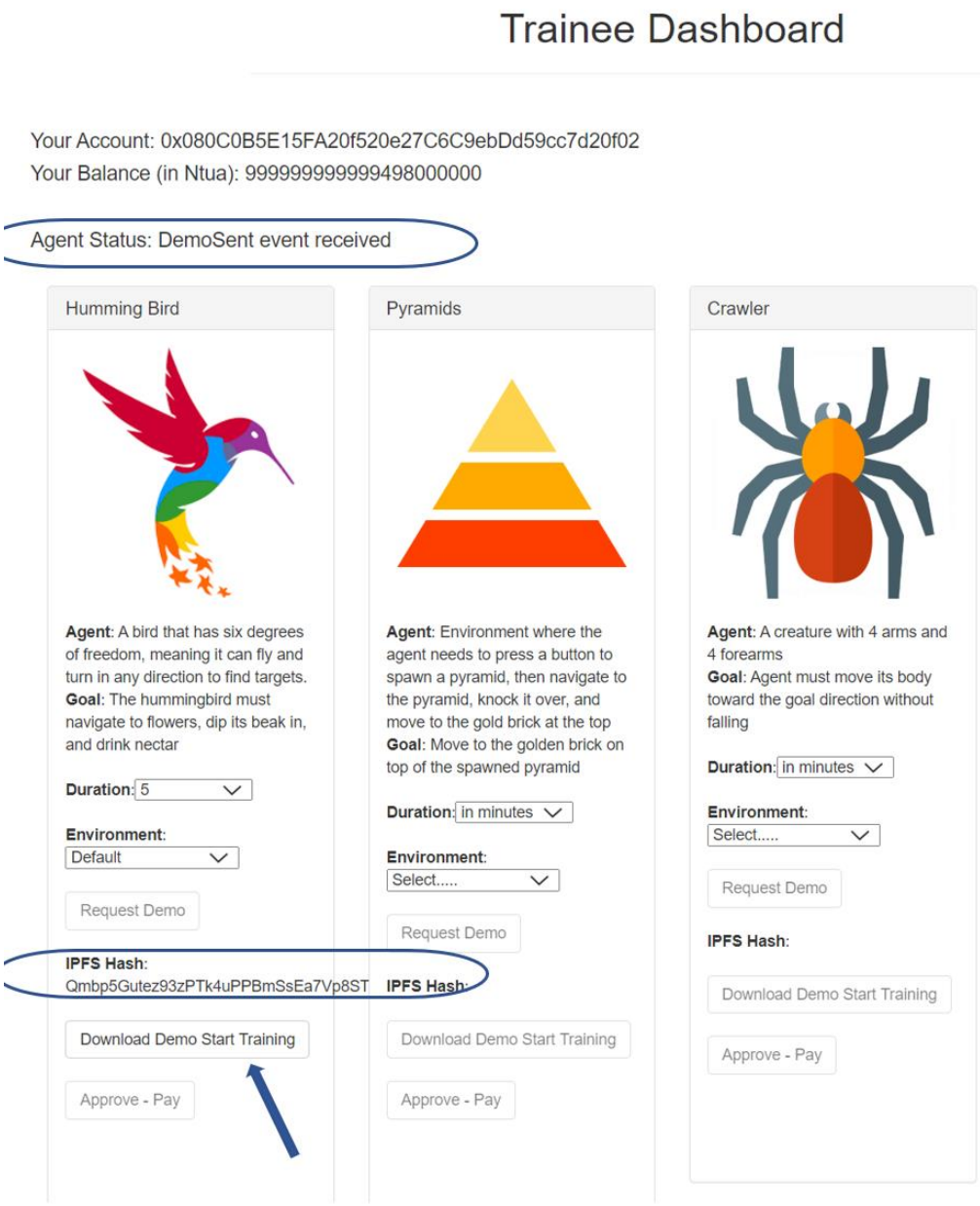
Το Ganache blockchain έχει ενημερωθεί ανάλογα:



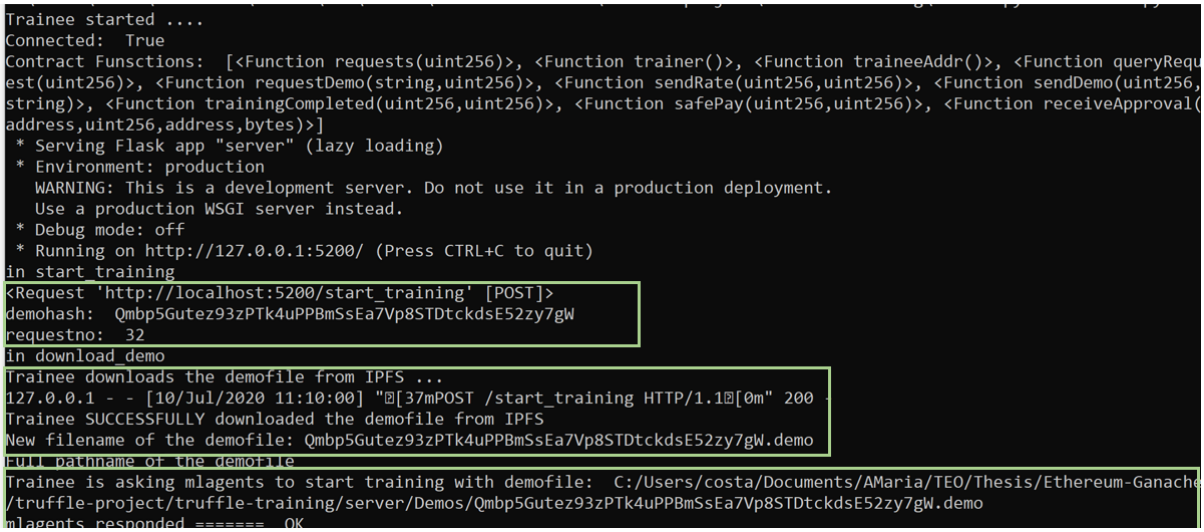
Το blockchain γεγονός DemoSent περιλαμβάνει το IPFS hashcode του αρχείου επίδειξης:



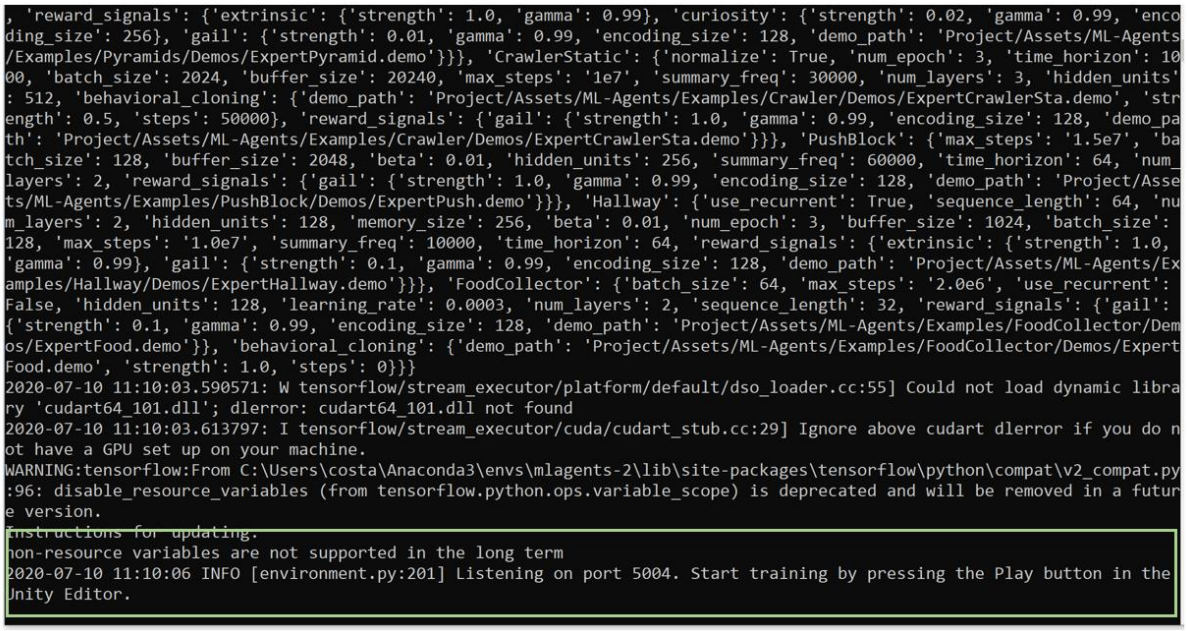
Ο Trainee ενημερώνεται ότι το αρχείο επίδειξης είναι διαθέσιμο στο IPFS και αφού το κατεβάσει στέλνει αίτημα στην εκτεταμένη έκδοση της βιβλιοθήκης ML-Agents να ξεκινήσει η εκπαίδευση:



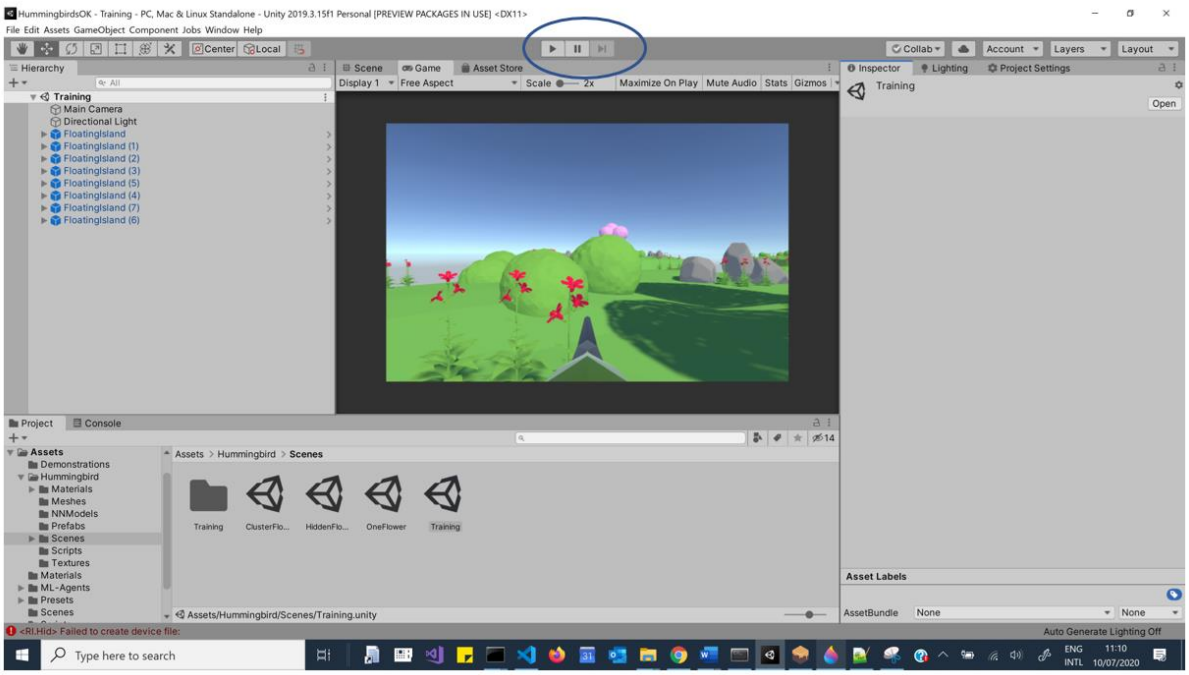
Ο trainee server δέχεται την εντολή /start_training κατεβάζει από το IPFS το αρχείο επίδειξης και ζητά από την ML-Agents βιβλιοθήκη να ξεκινήσει η εκπαίδευση:



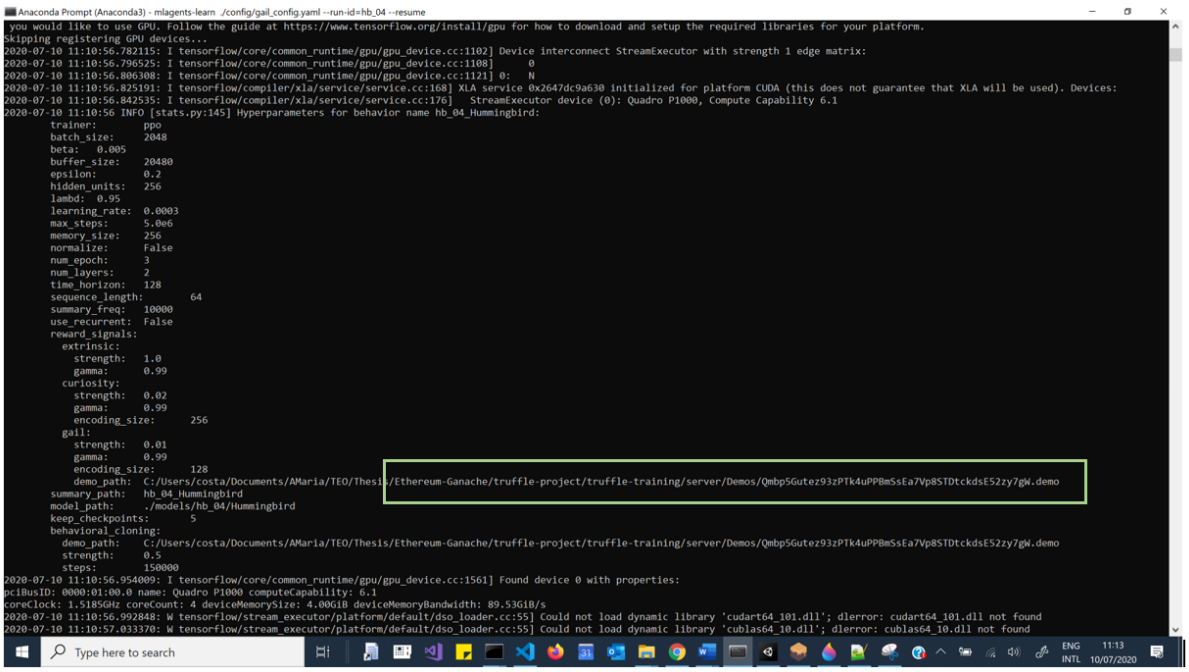
Η ML-Agents βιβλιοθήκη είναι έτοιμη να ξεκινήσει η εκπαίδευση:



Ο χρήστης επιβεβαιώνει ότι το Hummingbird περιβάλλον είναι επίσης έτοιμο για την έναρξη της εκπαίδευσης:



Η εκπαίδευση ξεκινά χρησιμοποιώντας το αρχείο επίδειξης που κατέβηκε από το IPFS:

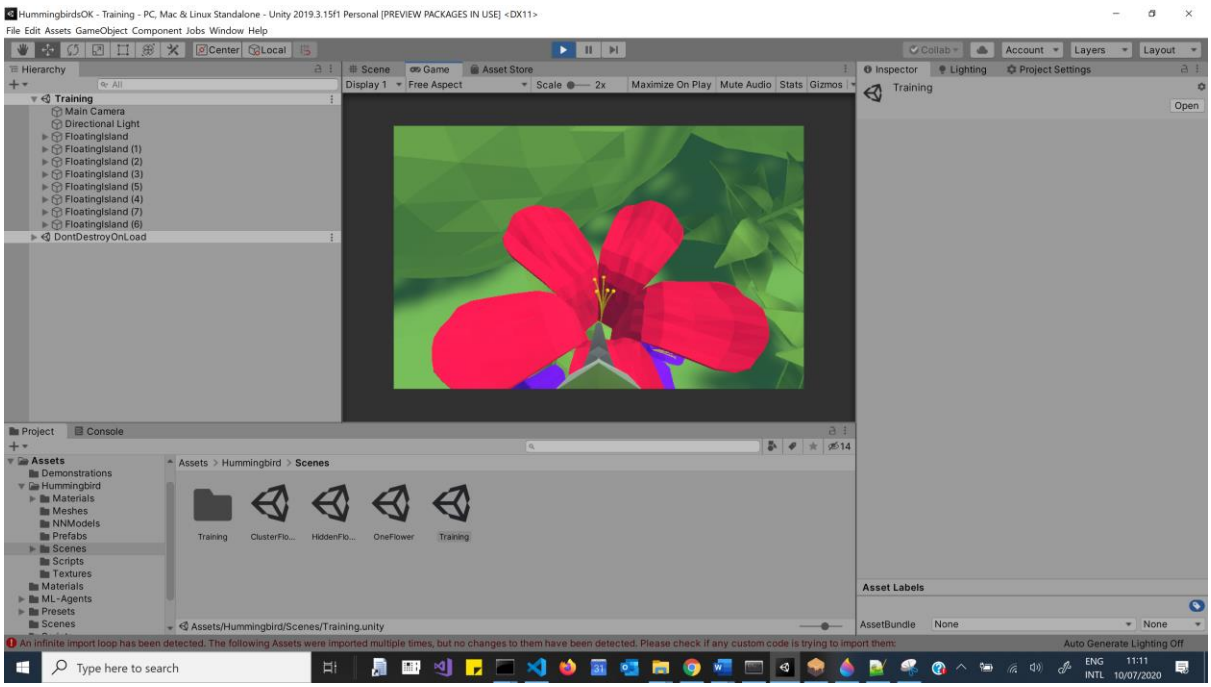


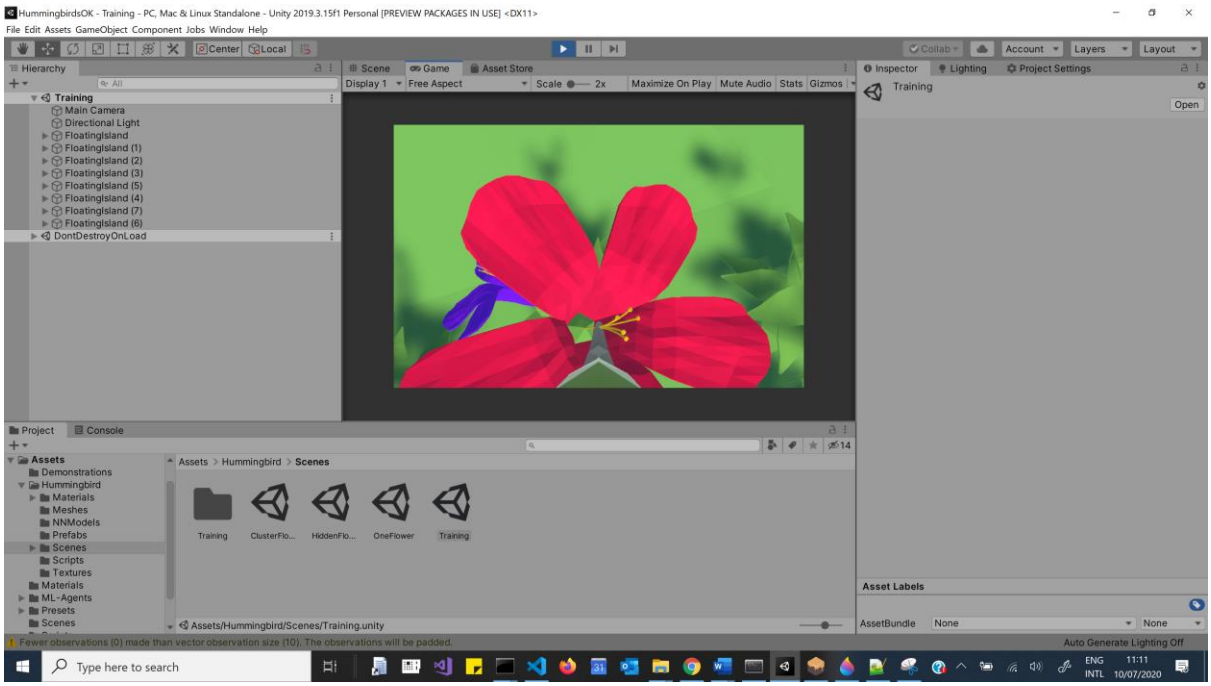
(ακολουθούν στιγμιότυπά κατά την διάρκεια της εκπαίδευσης)

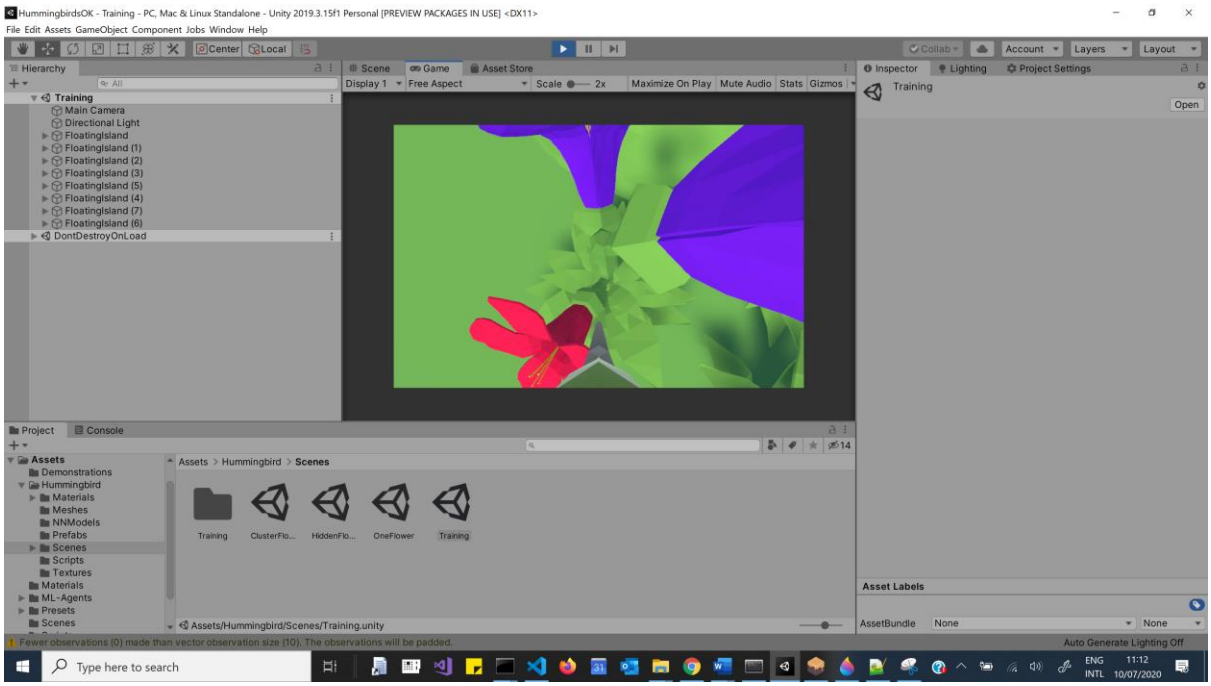
Η εκπαίδευση είναι σε εξέλιξη – Unity environment:

Σημείωση 1: Σκοπός του Hummingbird agent είναι να βρει τα λουλούδια που έχουν νέκταρ (κόκκινα λουλούδια) και να πιεί το νέκταρ. (τα λουλούδια που έχει ήδη επιστεφθεί ο hummingbird agent είναι τα μωβ)

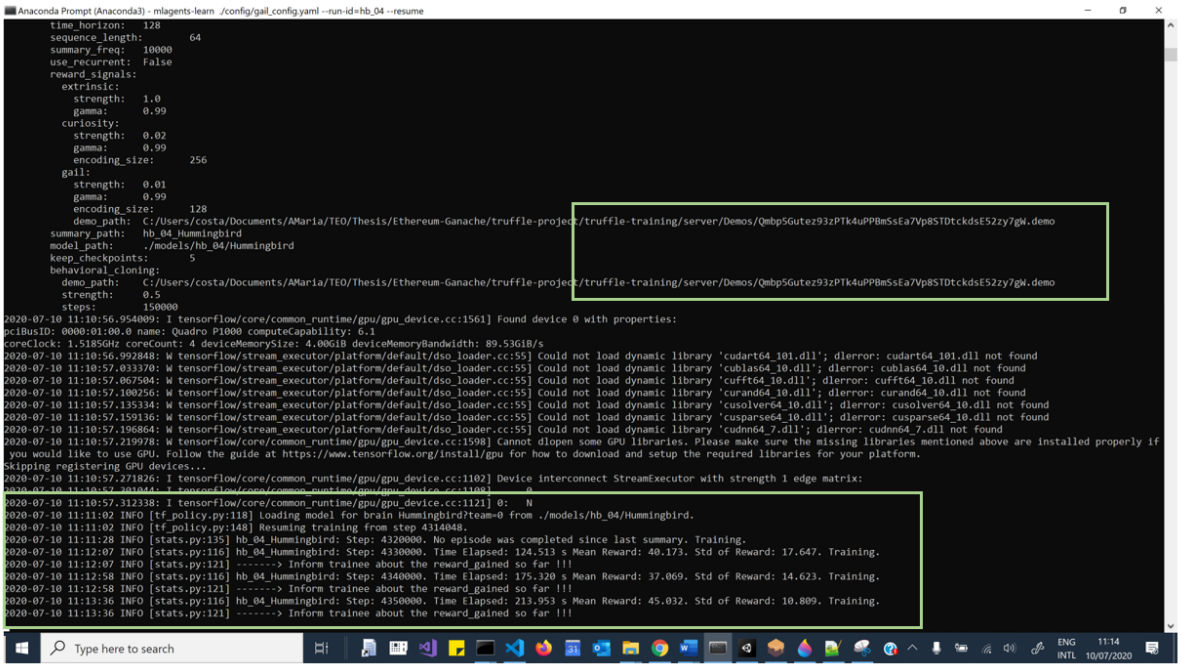
Σημείωση 2: Η κάμερα δείχνει την οπτική γωνία του Hummingbird agent



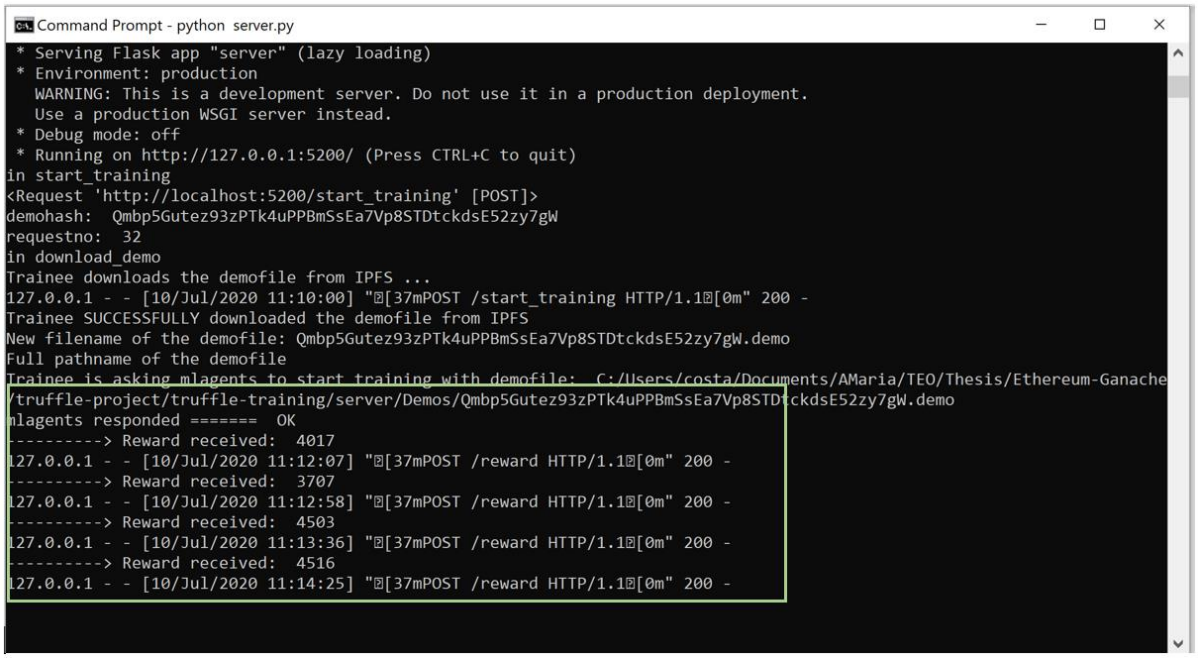




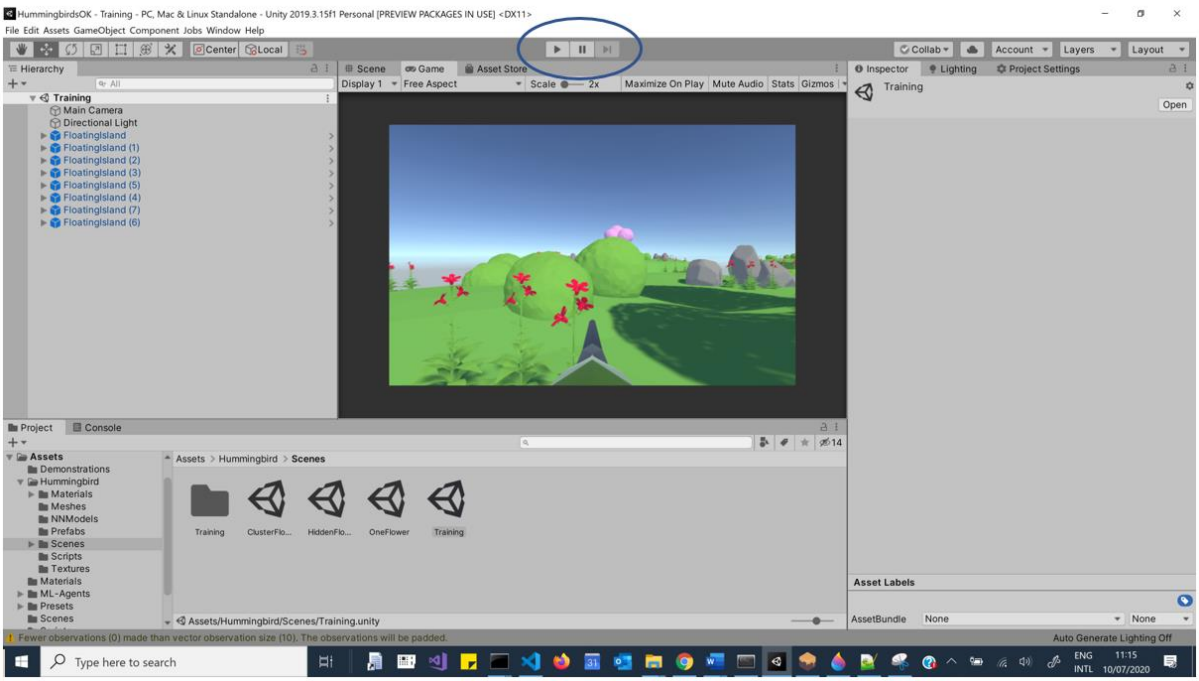
Η εκπαίδευση είναι σε εξέλιξη – ML-Agents βιβλιοθήκη



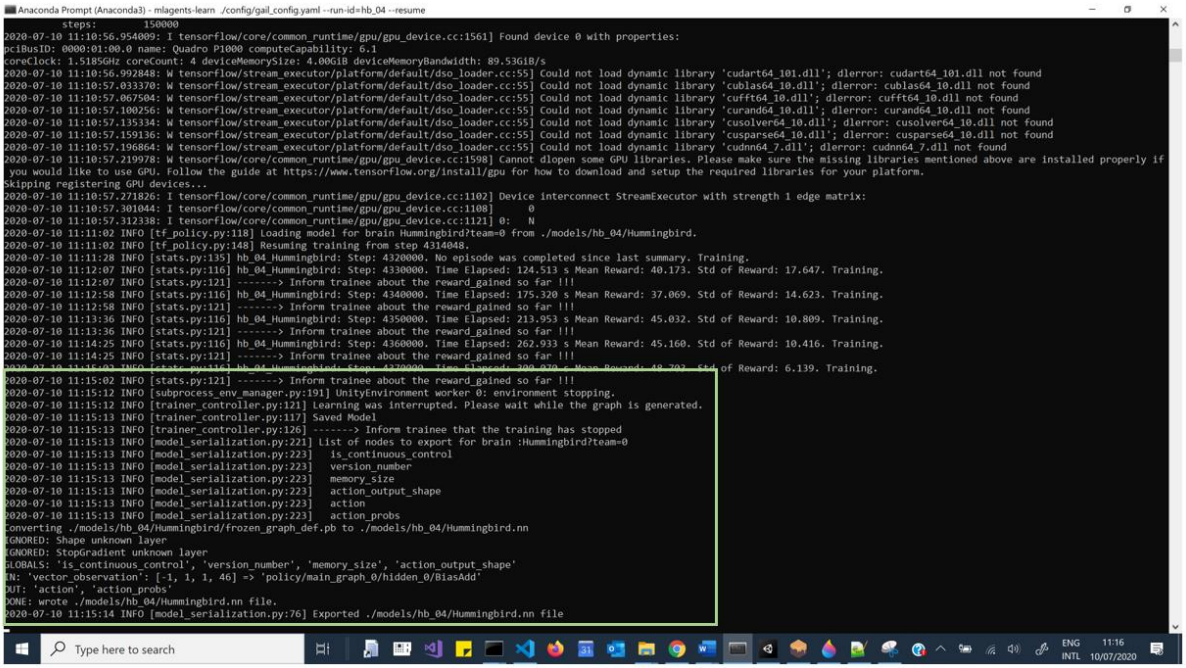
Η εκπαίδευση είναι σε εξέλιξη – Trainee server



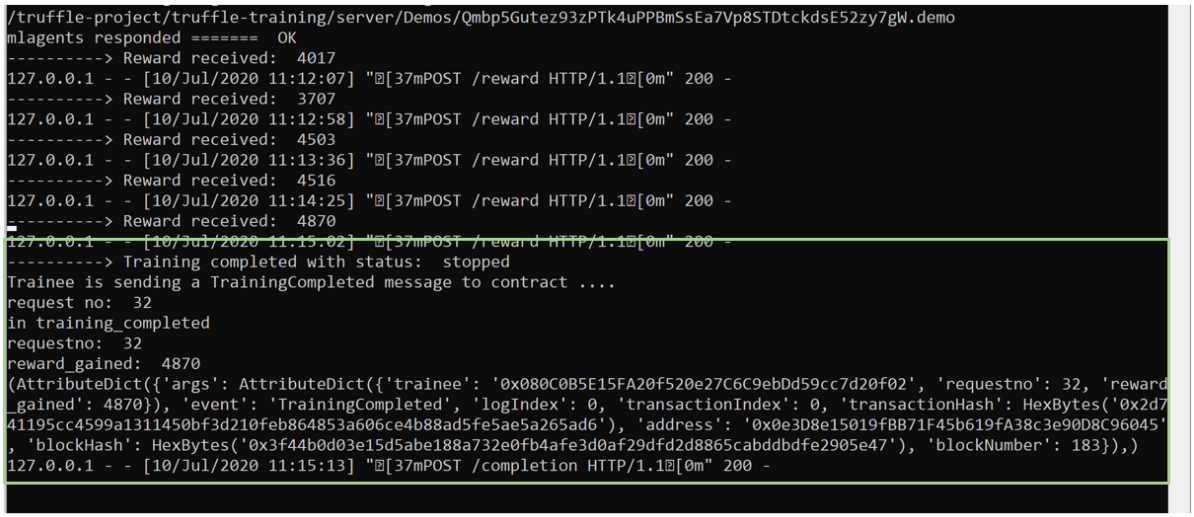
Η εκπαίδευση τελειώνει ή ο χρήστης αποφασίζει να σταματήσει την εκπαίδευση:



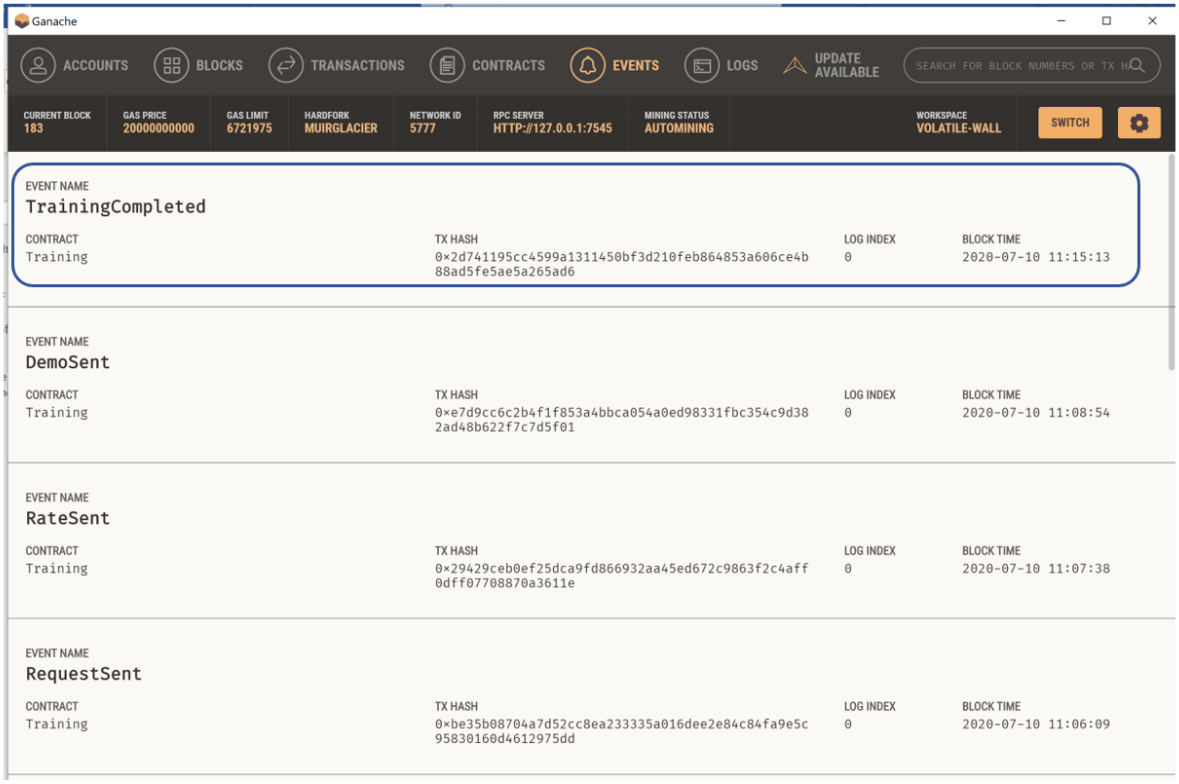
Η ML-agents βιβλιοθήκη σταματά την εκπαίδευση και αποστέλλει ανάλογο μήνυμα στον trainee server.



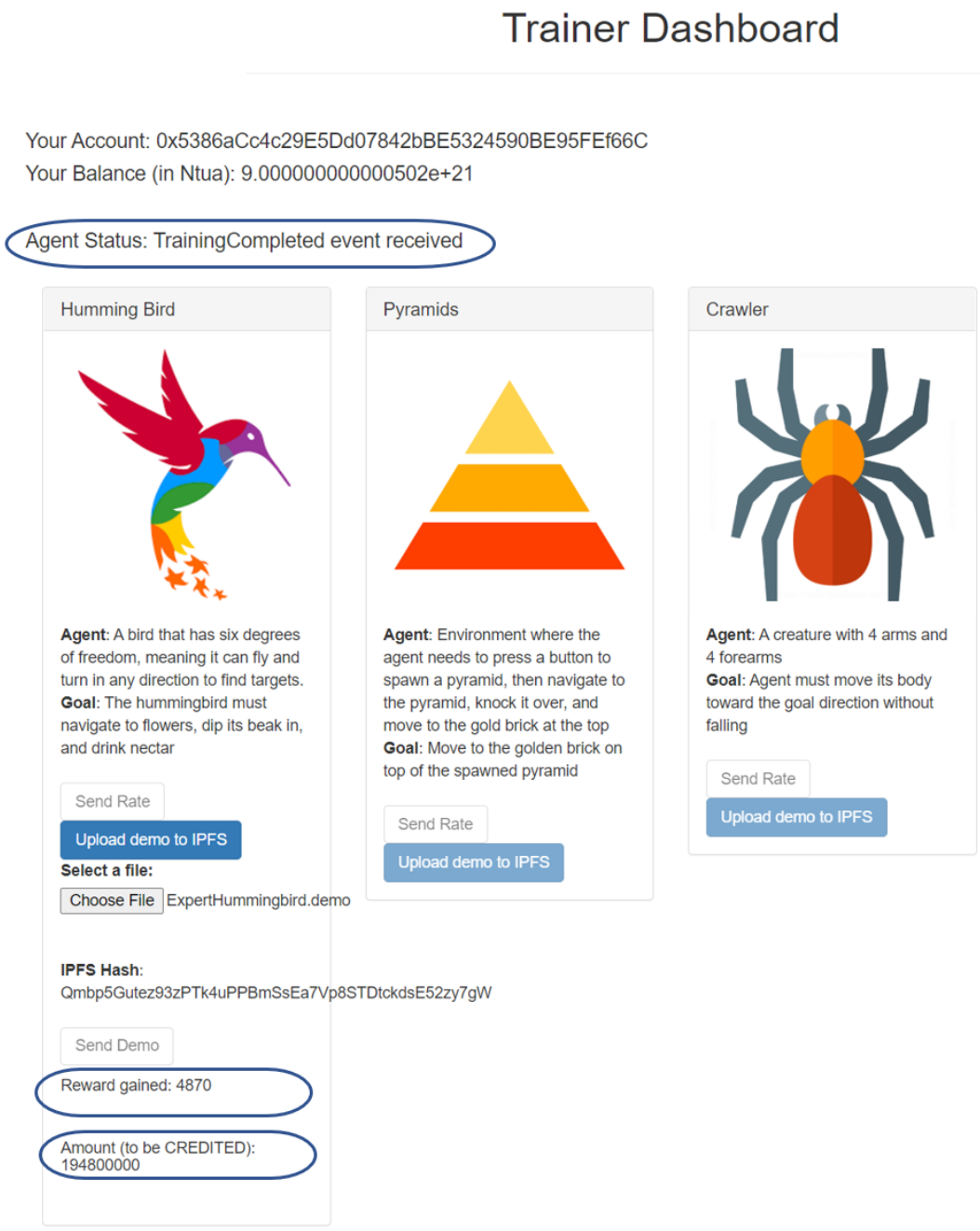
Ο trainee server ενημερώνεται ότι η εκπαίδευση έχει τελειώσει και ενημερώνει με την σειρά του το Training contract.



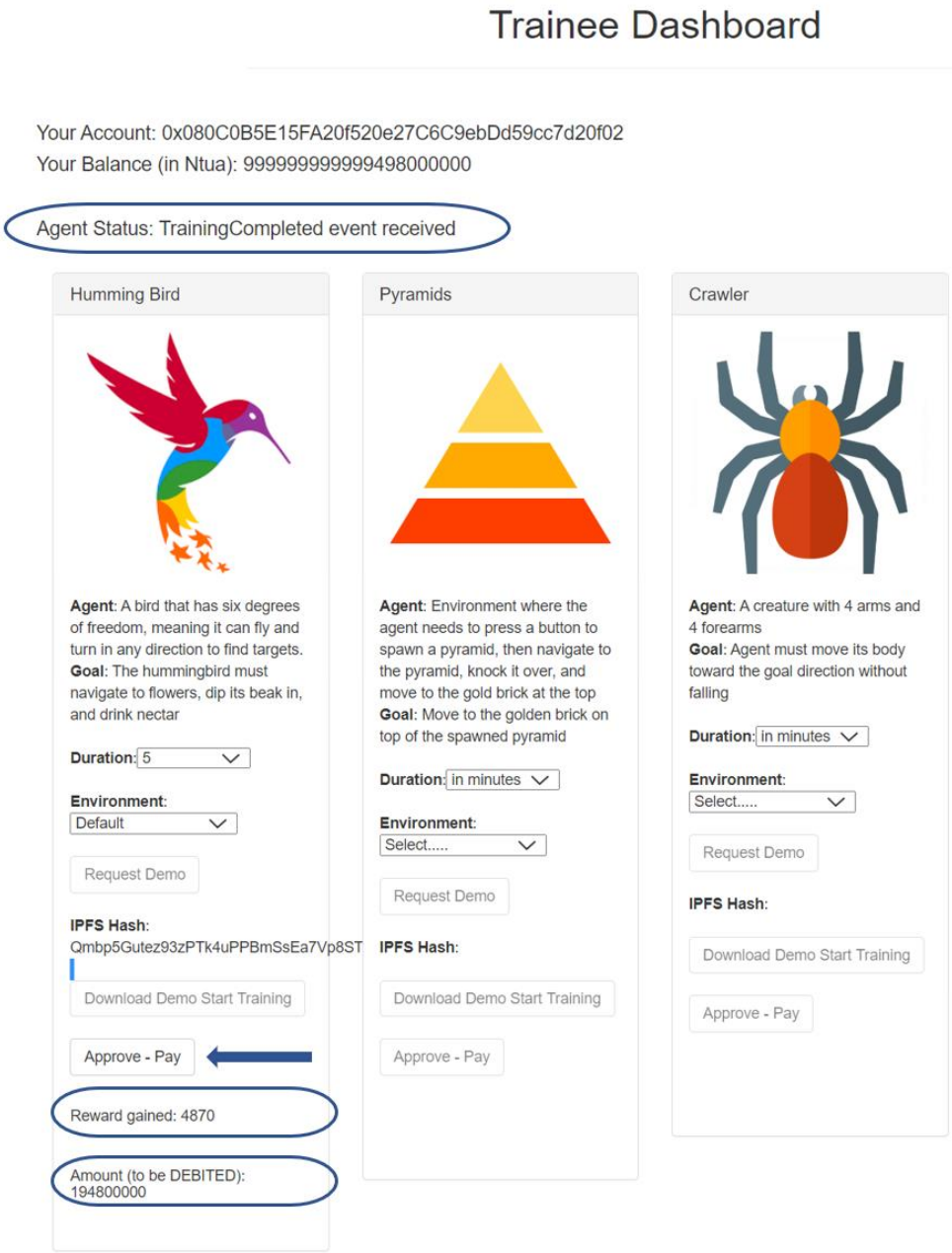
Το Ganache blockchain ενημερώνεται αναλόγως:



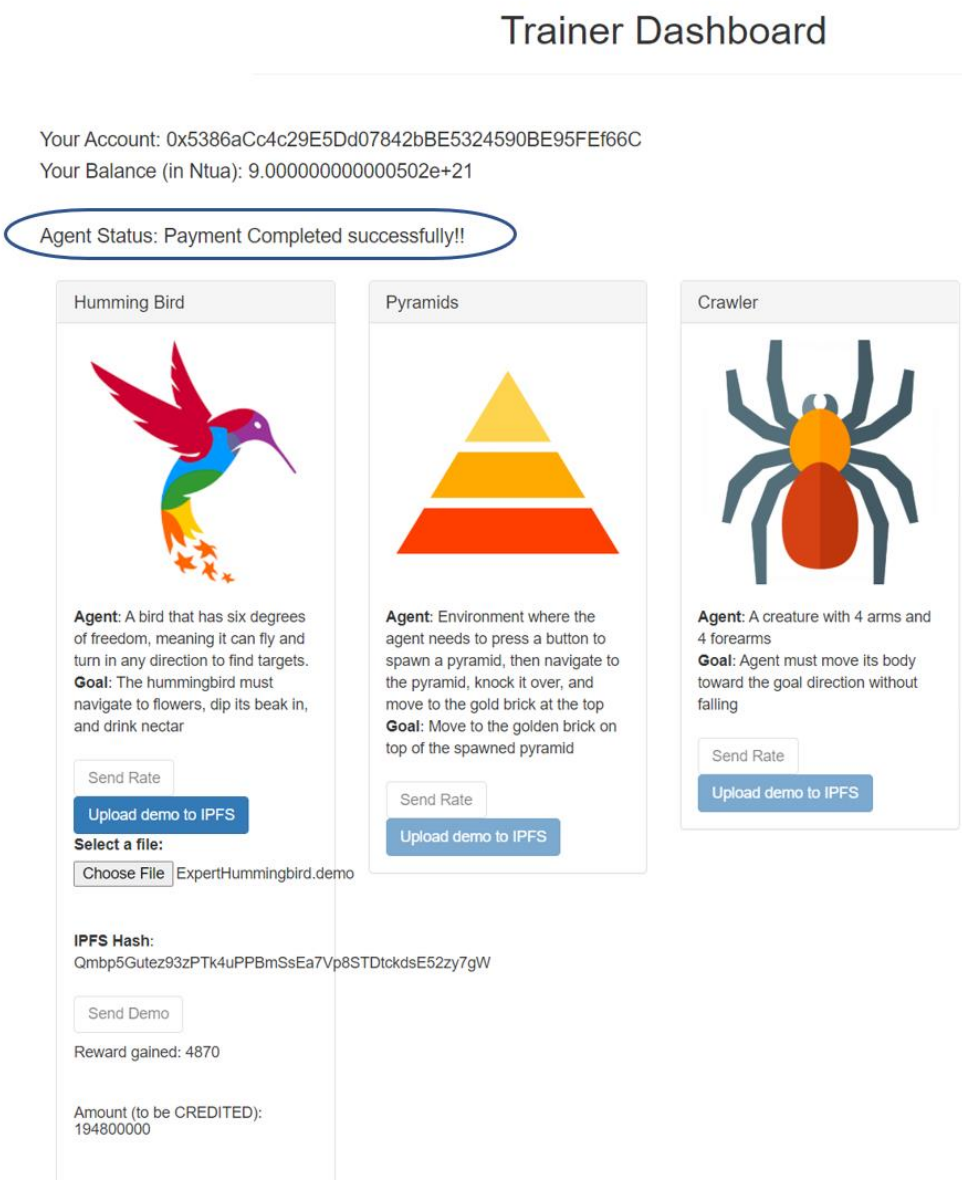
Ο Trainer ενημερώνεται ανάλογα:



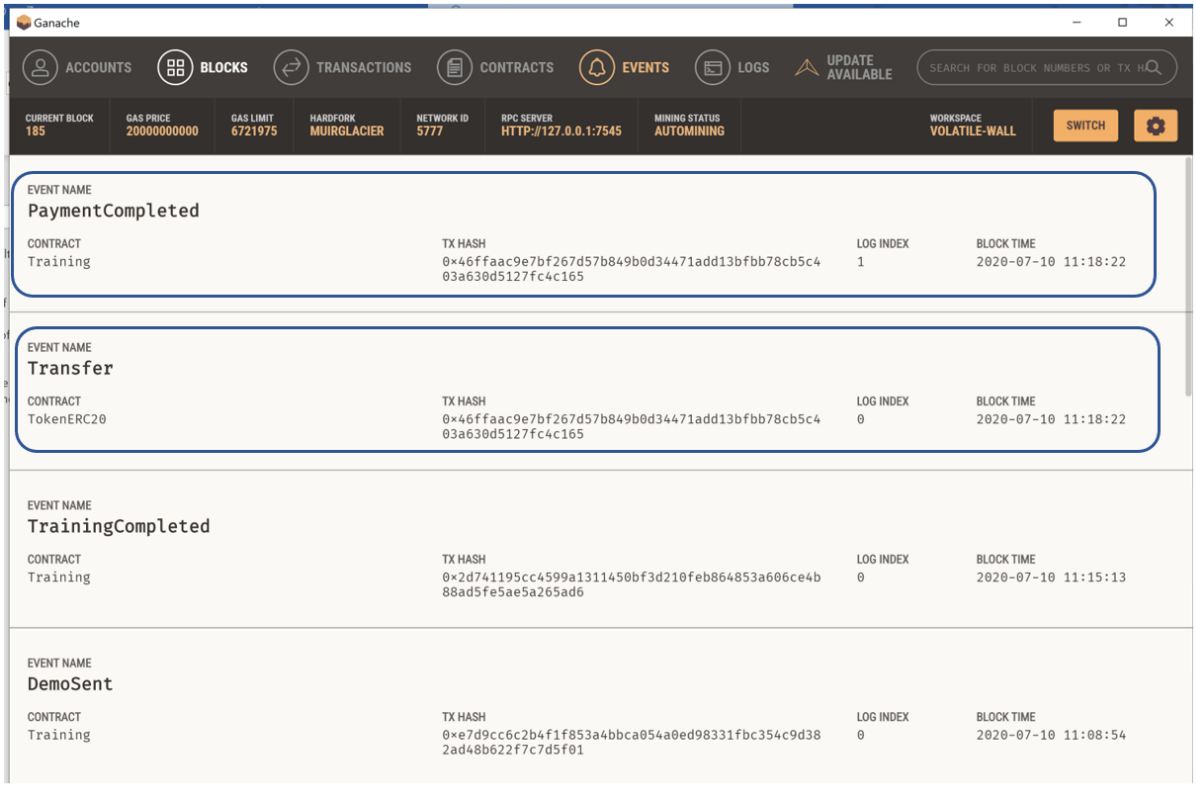
Ο Trainee ενημερώνεται ανάλογα και προχωρά στην αποπληρωμή του Trainer:



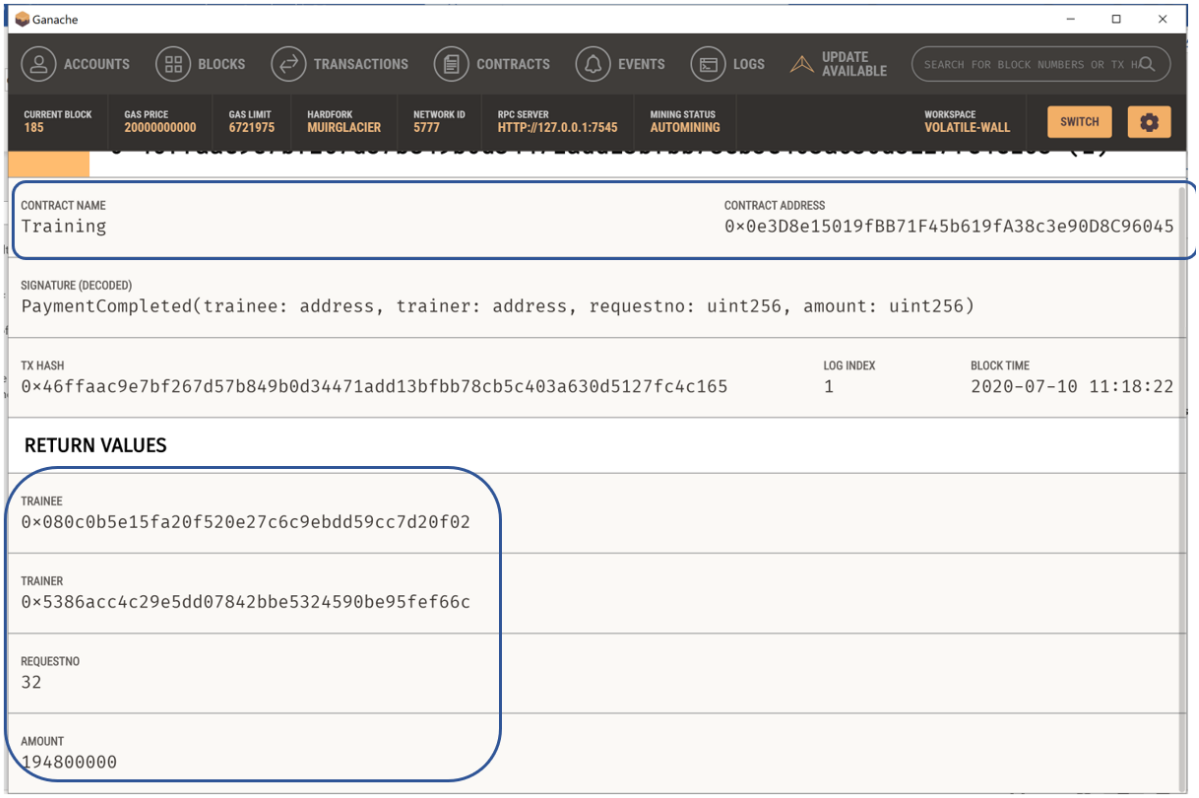
Ο Trainer ενημερώνεται ότι έχει γίνει η αποπληρωμή του:



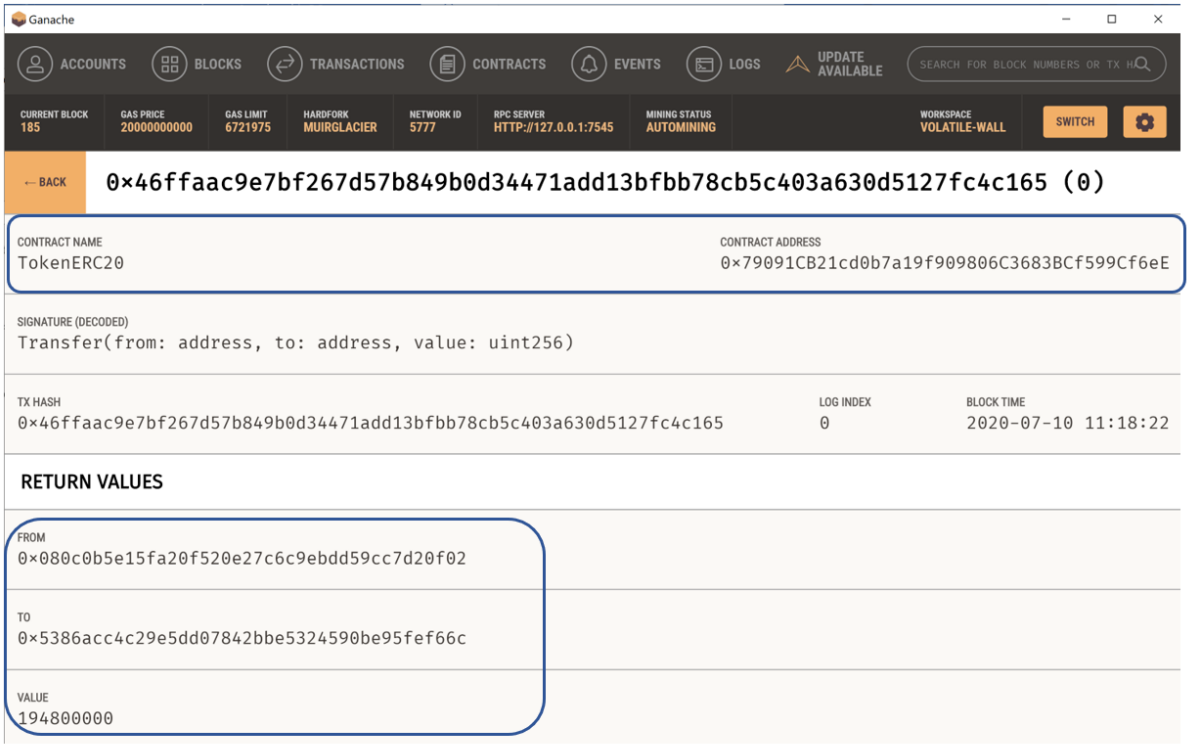
Το Ganache blockchain ενημερώνεται ανάλογα:



Το γεγονός PaymentCompleted επιβεβαιώνει ότι το Training contract εκτέλεση το αίτημα για αποπληρωμή.



Τέλος, το γεγονός Transfer επιβεβαιώνει ότι TokenERC20 Ntua token εκτέλεσε την αποπληρωμή.



8 Επιλογος

Σκοπός της διπλωματικής εργασίας ήταν η εύρεση ενός συνδυασμού δύο τεχνολογιών, της μηχανικής μάθησης (και πιο συγκεκριμένα της ενισχυτικής μάθησης) και του blockchain, που σε πολλά ζητήματα δρουν ανταγωνιστικά. Η μηχανική μάθηση απαιτεί κατά την εκπαίδευση μεγάλο όγκο δεδομένων και ισχυρή υπολογιστική ισχύ για να κατορθώσει να εξάγει γρήγορα συμπεράσματα. Από την άλλη μεριά η τεχνολογία blockchain έχει την ικανότητα να αποθηκεύει και να διαχειρίζεται μεγάλους όγκους δεδομένων αλλά απαιτεί και αυτή ισχυρή υπολογιστική ισχύ κυρίως κατά την mining διαδικασία. Κατά την έναρξη της εργασίας, η επίτευξη αυτού του σκοπού έμοιαζε τουλάχιστον δύσκολη σίγουρα όμως ήταν μια πρόκληση.

Αρχικά, μελετήθηκε η τεχνολογία blockchain και πιο συγκεκριμένα το Ethereum blockchain, η φιλοσοφία των έξυπνων συμβολαίων, τα διαθέσιμα εργαλεία και τεχνικές.

Στην συνέχεια, μελετήθηκε η περιοχή της ενισχυτικής μάθησης, η φιλοσοφία της και οι αλγόριθμοί της. Ιδιαίτερη βαρύτητα δόθηκε στη μελέτη του προβλήματος εύρεσης του

κατάλληλου μηχανισμού επιβράβευσης ώστε το περιβάλλον που αλληλοεπιδρά με τον εκπαιδευόμενο agent με **συνέπεια** να επιβραβεύει την επιθυμητή συμπεριφορά του. Αυτό οδήγησε στην μελέτη των αλγορίθμων μάθησης με απομίμηση και του τρόπου που αυτοί συνδυάζονται με την ενισχυτική μάθηση. Ακολούθησε η προσπάθεια εύρεσης κατάλληλων εργαλείων και τεχνικών για την υλοποίηση αλγορίθμων ενισχυτικής μάθησης και επιλέχθηκε η ML-Agents βιβλιοθήκη.

Τέλος, σχεδιάστηκε και υλοποιήθηκε μια αποκεντρωμένη (dApp) εφαρμογή, ένα είδος marketplace αρχείων επίδειξης που **αυτοματοποιεί πλήρως σαν μία ροή** την αγοραπωλησία των αρχείων επίδειξης και την εκπαίδευση του agent σε έναν κύκλο λειτουργίας. Δυστυχώς δεν κατέστη δυνατό να ενσωματωθεί σε αυτή την αυτόματη ροή και η καταγραφή των αρχείων επίδειξης. Η πλήρης αυτοματοποίηση της ροής διεργασιών αποτελεί μια μελλοντική επέκταση της εφαρμογής.

9 ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Justin D. Harris, Bo Waggoner (2019). Decentralized & Collaborative AI on Blockchain. arXiv print arxiv:1907.07247.pdf. https://www.microsoft.com/en-us/research/blog/leveraging-blockchain-to-make-machine-learning-models-more-accessible/

[2] Thang N. Dinh, My T. Thai (2018). AI and Blockchain: A Disruptive Integration. Thang N Dinh and My T Thai. “AI and Blockchain: A Disruptive Integration”. eng. In: Computer 51.9 (2018), pp. 48–53. issn: 0018-9162 https://www.researchgate.net/publication/328085550_AI_and_Blockchain_A_Disruptive_Integration

[3] S. Vyas, M. Gupta, and R. Yadav. “Converging Blockchain and Machine Learning for Healthcare”. In: 2019 Amity International Conference on Artificial Intelligence (AICAI). Feb. 2019, pp. 709–711. doi: 10.1109/AICAI.2019.8701230. https://www.researchgate.net/publication/332761355_Converging_Blockchain_and_Machine_Learning_for_Healthcare

[4] SUDEEP TANWAR, QASIM BHATIA, PRUTHVI PATEL, APARNA KUMARI, PRADEEP KUMAR SINGH, AND WEI-CHIANG HONG (2019) Machine Learning Adoption in Blockchain-Based Smart Applications: The Challenges, and a Way Forward https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8938741

[5] Ta Duong, Ketan Kumar Todi, Umang Chaudhary, Hong-Linh Truong (2019). Decentralizing Air Traffic Flow Management with Blockchain-based Reinforcement Learning, https://ieeexplore.ieee.org/abstract/document/8972225

[6] Mance E. Harmon, Stephanie S. Harmon. Reinforcement Learning: A Tutorial. http://www.cs.toronto.edu/~zemel/documents/411/rltutorial.pdf

[7] Richard S. Sutton and Andrew G. Barto, 2018, MIT Press, 2nd edition, Reinforcement Learning – An Introduction

[8] Daniel Dewey (2014) Reinforcement Learning and the Reward Engineering Principle. https://www.semanticscholar.org/paper/Reinforcement-Learning-and-the-Reward-Engineering-Dewey/fddab6f5edb57d3cbb7583c564b19586352705be

[9] Aleksandra Faust, Anthony Francis, Dar Mehta (2019) Evolving Rewards to Automate Reinforcement Learning https://storage.googleapis.com/pub-tools-public-publication-data/pdf/8af48862c365c72034597d68c8edf0c977dcd070.pdf

[10] Michael Bain and Claude Sommut. A framework for behavioural cloning. Machine Intelligence 15, 15:103, 1999. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.88.8215&rep=rep1&type=pdf

[11] Faraz Torabi , Garrett Warnell , Peter Stone (2018) Behavioural Cloning from Observation https://arxiv.org/pdf/1805.01954.pdf

[12] Eric Zhan, Stephan Zheng, Yisong Yue, Patrick Lucey. Generative Multi-Agent Behavioral Cloning. http://www.stephanzheng.com/pdf/Zhan_Zheng_Lucey_Yue_Generative_Multi_Agent_Behavioral_Cloning.pdf
[13] Jonathan Ho, Stefano Ermon Generative Adversarial Imitation Learning. https://arxiv.org/pdf/1606.03476.pdf
[14] ML-Agents : Imitation Learning https://github.com/Unity-Technologies/ml-agents/blob/release_2_docs/docs/ML-Agents-Overview.md#imitation-learning
[15] Γ. Παπαδόδημας, Διπλωματική Εργασία (2018). Ανάπτυξη Έξυπνων Συμβολαίων στο Blockchain και εφαρμογή στο IoT
[16] G. Papadodimas, G. Palaiokrasa, A. Litke, T. Varvarigou. (2018). Implemntation of smart contracts for blockchain based IoT applications
[17] Zibin Zheng and Shaoan Xie (2018). Blockchain challenges and opportunities: a survey. https://www.researchgate.net/publication/328338366_Blockchain_challenges_and_opportunities_A_survey
[18] InterPlanetary File System – IPFS : https://docs.ipfs.io/concepts/how-ipfs-works/
[19] Swarm - https://swarm-guide.readthedocs.io/en/latest/introduction.html
[20] Thomas Simonini. On Choosing a Deep Reinforcement Learning Library (2019). https://medium.com/data-from-the-trenches/choosing-a-deep-reinforcement-learning-library-890fb0307092
[21]Juliani, A., Berges, V., Teng, E., Cohen, A., Harper, J., Elion, C., Goy, C., Gao, Y., Henry, H., Mattar, M., Lange, D. (2020). Unity: A General Platform for Intelligent Agents. *arXiv preprint arXiv:1809.02627*. https://github.com/Unity-Technologies/ml-agents.
[22] Keras-RL Deep Reinforcement Learning for Keras : https://github.com/keras-rl/keras-rl
[23] TensorForce : a TensorFlow library for applied reinforcement learning - https://github.com/tensorforce/tensorforce
[24] TensorFlow – An end-to-end open source machine learning platform: https://www.tensorflow.org/
[25] OpenAI Baselines - A set of high quality implementations of reinforcement learing algorithms : https://github.com/openai/baselines
[26] Stable Baselines : A set of improved implementations of reinforcement learing algorithms based on OpenAI Baselines - https://github.com/hill-a/stable-baselines
[27]TF-Agents: A reliable, scalable and easy to use Reinforcement Learning library for TensorFlow - https://github.com/tensorflow/agents
[28] Open AI – Gym : A collection of test environments for reinforcement learning algorithms - https://gym.openai.com/
[29]Yoshua Bengio, Jermoe Louradour, Ronan Collobert, Jason Weston. Curriculum Learning. https://ronan.collobert.com/pub/matos/2009_curriculum_icml.pdf
[30] Unity - https://unity.com/
[31] Anaconda : Data science technology for groundbreaking research.a competitive edge.a better world.human sensemaking - https://www.anaconda.com/
[32] Infura IPFS API https://infura.io/docs/ipfs/get/block-get
[33]Ethereum ERC20 Token Standard https://eips.ethereum.org/EIPS/eip-20
[34]Ethereum ERC20 Token Standard Issue resolved by Ethereum ERC223 Token Standard https://github.com/ethereum/eips/issues/223
[35]Ethereum ERC223 Token Standard https://github.com/Dexaran/ERC223-token-standard
[36]Solidity - https://solidity.readthedocs.io/en/develop/index.html
[37]Truffle Framework - https://www.trufflesuite.com/docs/truffle/overview
[38]Truffle Ganache - https://www.trufflesuite.com/docs/ganache/overview
[39]Truffle – Running Migrations - https://www.trufflesuite.com/docs/truffle/getting-started/running-migrations

[40] Truffle Console - https://www.trufflesuite.com/docs/truffle/getting-started/using-truffle-develop-and-the-console
[41] web3.py - A Python implementation of web3.js : https://web3py.readthedocs.io/en/stable/quickstart.html
[42] web3.py - Asynchronous Filter polling https://web3py.readthedocs.io/en/stable/filters.html
[43] web3.js – Ethereum Javascript API : https://web3js.readthedocs.io/en/v1.2.9/
[44] Flask : a python web application framework - https://flask.palletsprojects.com/en/1.1.x/
[45] JavaScript Programming Language - https://developer.mozilla.org/en-US/docs/Web/JavaScript
[46] Bootstrap CSS framework- https://getbootstrap.com/docs/4.5/getting-started/introduction/
[47] Lite-server: a lightweight development only web app server : https://github.com/johnpapa/lite-server
[48] ML-Agents – Example Learning environment : https://github.com/Unity-Technologies/ml-agents/blob/release_2_docs/docs/Learning-Environment-Examples.md
[49] Unity Learn: ML-Agents – Hummingbird course: https://learn.unity.com/project/course-overview?uv=2019.3&courseId=5e470160edbc2a15578b13d7
[50] TensorBoard: TensorFlow's Visualization Toolkit : https://www.tensorflow.org/tensorboard/
[51] MetaMask- A crypto wallet and a gateway to blockchain apps : https://metamask.io/
[52] Visual Studio Code : A source code editor - https://code.visualstudio.com/
[53]Python programming language - https://www.python.org/
[54]Node.js – a JavaScript runtime environment https://nodejs.org/en/
[55]Proximal Policy Optimization (PPO) - https://openai.com/blog/openai-baselines-ppo/
[56]Software Actor Critic (SAC) - https://bair.berkeley.edu/blog/2018/12/14/sac/
[57] Curiosity driven exploration by Self-supervised prediction: https://pathak22.github.io/noreward-rl/
[58]Fang Chen, Hong Wan, Hua Cai, Guang Cheng (2020). Machine Learning in/for Blockchain: Future and Challenges. https://arxiv.org/pdf/1909.06189.pdf
[59] K. Salah, M. H. U. Rehman, N. Nizamuddin and A. Al-Fuqaha, "Blockchain for AI: Review and Open Research Challenges," in *IEEE Access*, vol. 7, pp. 10127-10149, 2019, doi: 10.1109/ACCESS.2018.2890507. https://www.researchgate.net/publication/330009592_Blockchain_for_AI_Review_and_Open_Research_Challenges

10 ΠΑΡΑΡΤΗΜΑ : ΚΩΔΙΚΕΣ

10.1 ΄ΕΞΥΠΝΑ ΣΥΜΒΟΛΑΙΑ

10.1.1 ERC20 Token

```
pragma solidity ^0.4.24;

interface tokenRecipient {function receiveApproval(address _from, uint256 _value,
                                                    address _token, bytes _extraData) external; }

contract TokenERC20 {
   // Public variables of the token
   string public name;
   string public symbol;
```

```
uint8 public decimals = 18;                // 18 decimals is the strongly suggested default, avoid changing it
uint256 public totalSupply;

mapping (address => uint256) public balanceOf;              // This creates an array with all balances
mapping (address => mapping (address => uint256)) public allowance;

// This generates a public event on the blockchain that will notify clients
event Transfer(address indexed from, address indexed to, uint256 value);

// This notifies clients about the amount burnt
event Burn(address indexed from, uint256 value);

/**
 * Constrctor function : Initializes contract with initial supply tokens to the creator of the contract
 */
constructor(
    uint256 initialSupply,
    string tokenName,
    string tokenSymbol
) public {
    totalSupply = initialSupply * 10 ** uint256(decimals);      // Update total supply with the decimal amount
    balanceOf[msg.sender] = totalSupply;                        // Give the creator all initial tokens
    name = tokenName;                                           // Set the name for display purposes
    symbol = tokenSymbol;                                       // Set the symbol for display purposes
}

/**
 * Internal transfer, only can be called by this contract
 */
function _transfer(address _from, address _to, uint _value) internal {
    require(_to != 0x0);                                        // Prevent transfer to 0x0 address. Use burn() instead
    require(balanceOf[_from] >= _value);                        // Check if the sender has enough
    require(balanceOf[_to] + _value > balanceOf[_to]);          // Check for overflows
    uint previousBalances = balanceOf[_from] + balanceOf[_to]; // Save this for an assertion in the future
    balanceOf[_from] -= _value;                                 // Subtract from the sender
    balanceOf[_to] += _value;                                   // Add the same to the recipient
    emit Transfer(_from, _to, _value);
    // Asserts are used to use static analysis to find bugs in your code. They should never fail
    assert(balanceOf[_from] + balanceOf[_to] == previousBalances);
}

/**
 * Transfer tokens : Send `_value` tokens to `_to` from your account
 * @param _to The address of the recipient
 * @param _value the amount to send
 */
function transfer(address _to, uint256 _value) public {
    _transfer(msg.sender, _to, _value);
}

/**
 * Transfer tokens from other address : Send `_value` tokens to `_to` in behalf of `_from`
 * @param _from The address of the sender
 * @param _to The address of the recipient
 * @param _value the amount to send
 */
function transferFrom(address _from, address _to, uint256 _value) public returns (bool success) {
    require(_value <= allowance[_from][msg.sender]);        // Check allowance
    allowance[_from][msg.sender] -= _value;
    _transfer(_from, _to, _value);
    return true;
}

/**
 * Set allowance for other address : Allows `_spender` to spend no more than `_value` tokens in your behalf
 * @param _spender The address authorized to spend
 * @param _value the max amount they can spend
 */
function approve(address _spender, uint256 _value) public
    returns (bool success) {
    allowance[msg.sender][_spender] = _value;
    return true;
}

/**
```

```
    * Set allowance for other address and notify :
    * Allows `_spender` to spend no more than `_value` tokens in your behalf, and then ping the contract about it
    * @param _spender The address authorized to spend
    * @param _value the max amount they can spend
    * @param _extraData some extra information to send to the approved contract
    */
    function approveAndCall(address _spender, uint256 _value, bytes _extraData)
        public
        returns (bool success) {
        tokenRecipient spender = tokenRecipient(_spender);
        if (approve(_spender, _value)) {
            spender.receiveApproval(msg.sender, _value, this, _extraData);
            return true;
        }
    }

    /**
    * Destroy tokens : Remove `_value` tokens from the system irreversibly
    * @param _value the amount of money to burn
    */
    function burn(uint256 _value) public returns (bool success) {
        require(balanceOf[msg.sender] >= _value);               // Check if the sender has enough
        balanceOf[msg.sender] -= _value;                        // Subtract from the sender
        totalSupply -= _value;                                  // Updates totalSupply
        emit Burn(msg.sender, _value);
        return true;
    }

    /**
    * Destroy tokens from other account : Remove `_value` tokens from the system irreversibly on behalf of `_from`.
    * @param _from the address of the sender
    * @param _value the amount of money to burn
    */
    function burnFrom(address _from, uint256 _value) public returns (bool success) {
        require(balanceOf[_from] >= _value);                    // Check if the targeted balance is enough
        require(_value <= allowance[_from][msg.sender]);        // Check allowance
        balanceOf[_from] -= _value;                             // Subtract from the targeted balance
        allowance[_from][msg.sender] -= _value;                 // Subtract from the sender's allowance
        totalSupply -= _value;                                  // Update totalSupply
        emit Burn(_from, _value);
        return true;
    }
}
```

10.1.2 Training Contract

```
pragma solidity ^0.4.24;

import "./TokenERC20.sol";

contract Training {

    TokenERC20 private Ntua;                    // include TokenERC20 as Ntua token

    address public trainer;                     // The trainer's address
    address public traineeAddr;                 // The trainee's address

    struct Request {                            // The Request struct
        string demotype;
        uint duration;
        uint number;
        uint rate;
        bool init;
        string demofile;
        uint reward_gained;
    }

    mapping (uint => Request) public requests;  // The mapping to store the requests for demo
    uint requestseq;                            // The sequence number of the requests for demo
```

```
// Event triggered when a new request for demo received
event RequestSent(address trainee, string demotype, uint duration, uint requestno);

// Event triggered when the trainer has evaluate the rate of the reward for the requested demo
event RateSent(address trainee, uint requestno, uint rate);

// Event triggered when the trainer has prepared requested demo
// demofile: contains the hash code provided by IPFS
event DemoSent(address trainee, uint requestno, string demofile);

// Event triggered when the trainee has completed the training
event TrainingCompleted(address trainee, uint requestno, uint reward_gained);

// Event triggered when the payment took place
event PaymentCompleted(address trainee, address trainer, uint requestno, uint amount);

// Constructor
constructor(TokenERC20 _Ntua, address _traineeAddr) public payable {
    trainer = msg.sender;                           // Trainer is the contract owner
    traineeAddr = _traineeAddr;
    Ntua = _Ntua;
}

// The function to query requests by its sequence number
function queryRequest(uint number) constant public returns (uint duration, uint rate, string demofile, uint reward_gained ) {
    require(requests[number].init);                 // Validate the request number
    // Return the order data
    return(requests[number].duration,requests[number].rate, requests[number].demofile, requests[number].reward_gained);
}

// The function to request for Demo
function requestDemo(string demotype, uint duration) public payable {
    // Accept requests only from trainee
    require(msg.sender == traineeAddr);
    // Increment the sequence number of requests
    requestseq++;
    // Update the requests list
    requests[requestseq] = Request(demotype, duration, requestseq, 0, true, "", 0);
    /// Trigger the event
    emit RequestSent(msg.sender, demotype, duration, requestseq);

}

// The function to send the rate to pay for the demo
function sendRate(uint requestno, uint rate) public payable {
    // Only the trainer can use this function
    require(msg.sender == trainer);
    /// Validate the request number
    require(requests[requestno].init);
    requests[requestno].rate = rate;
    /// Trigger the event
    emit RateSent(traineeAddr, requestno, rate);
}

// The function to send the rate to pay for the demo
function sendDemo(uint requestno, string demofile) public payable {
    // Only the trainer can use this function
    require(msg.sender == trainer);
    /// Validate the request number
    require(requests[requestno].init);
    requests[requestno].demofile = demofile;
    /// Trigger the event
    emit DemoSent(traineeAddr, requestno, demofile);
}

// The function to inform that the training has completed
function trainingCompleted(uint requestno, uint reward_gained) public payable {
    // Accept requests only from trainee
    require(msg.sender == traineeAddr);
        /// Validate the request number
    require(requests[requestno].init);
    // Update the requests list
    requests[requestseq].reward_gained = reward_gained;

    // Trigger the event
```

```
        emit TrainingCompleted(traineeAddr, requestseq, reward_gained);
    }

    // The function to inform that the training has completed
    function safePay(uint requestno, uint reward_gained) public payable {
        // Accept requests only from trainee
        require(msg.sender == traineeAddr);
              /// Validate the request number
        require(requests[requestno].init);
        // Update the requests list

        uint amount = reward_gained * requests[requestno].rate * 10 ** 4;

        // Training contract should ask from ERC20 Token to perform the transaction
        //Ntua.transferFrom(msg.sender, trainer, amount);
        //require(Ntua.transferFrom(msg.sender, address(this), amount));
        require(Ntua.transferFrom(msg.sender, trainer, amount));

        // Trigger the event
        emit PaymentCompleted(traineeAddr, trainer, requestseq, amount);
    }

    // _value: holds the amount approved by trainee
    // _extraData: holds the reward_gained by trainee
    function receiveApproval(address _sender,
                    uint256 _value,
                    TokenERC20 _tokenContract,
                    bytes _extraData) public {
        require(Ntua == _tokenContract);
        // the _extraData is passed as bytes
        // decode the value according
        uint256 payloadSize;
        uint256 reward_gained;
        uint256 amount;
        assembly {
            payloadSize := mload(_extraData)
            reward_gained := mload(add(_extraData, 0x20))
        }
        reward_gained = reward_gained >> 8*(32 - payloadSize);
        //requests[requestseq].reward_gained = reward_gained;
        require(requests[requestseq].reward_gained == reward_gained);

        amount = requests[requestseq].rate * reward_gained * 10 ** 4;
        require(_value == amount);

        require(_tokenContract.transferFrom(_sender, address(this), _value));
        // Trigger the event
        emit PaymentCompleted(traineeAddr, trainer, requestseq, _value);

    }
}
```

10.2 ΑΡΧΕΙΟ ΠΑΡΑΜΕΤΡΟΠΟΙΗΣΗΣ ΤΗΣ ΕΚΠΑΙΔΕΥΣΗΣ ΤΩΝ ΜΟΝΤΕΛΩΝ

Ενδεικτικά παρατίθεται ένα αρχείο παραμετροποίησης της εκπαίδευσης των μοντέλων

```
default:
    trainer: ppo
    batch_size: 1024
    beta: 5.0e-3
    buffer_size: 10240
    epsilon: 0.2
    hidden_units: 128
    lambd: 0.95
    learning_rate: 3.0e-4
    max_steps: 5.0e5
    memory_size: 256
    normalize: false
    num_epoch: 3
    num_layers: 2
```

```
    time_horizon: 64
    sequence_length: 64
    summary_freq: 10000
    use_recurrent: false
    reward_signals:
      extrinsic:
        strength: 1.0
        gamma: 0.99

Pyramids:
    summary_freq: 30000
    time_horizon: 128
    batch_size: 128
    buffer_size: 2048
    hidden_units: 512
    num_layers: 2
    beta: 1.0e-2
    max_steps: 1.0e7
    num_epoch: 3
    behavioral_cloning:
      demo_path: c:/users/teo/thesis/Pyramids/Demos/ExpertPyramid.demo
      strength: 0.5
      steps: 150000
    reward_signals:
      extrinsic:
        strength: 1.0
        gamma: 0.99
      curiosity:
        strength: 0.02
        gamma: 0.99
        encoding_size: 256
      gail:
        strength: 0.01
        gamma: 0.99
        encoding_size: 128
        demo_path: c:/users/teo/thesis/Pyramids/Demos/ExpertPyramid.demo

Hummingbird:
    time_horizon: 128
    batch_size: 2048
    buffer_size: 20480
    hidden_units: 256
    max_steps: 5.0e6
    behavioral_cloning:
      demo_path: c:/users/teo/thesis/Humming/Demos/ExpertHumming.demo
      strength: 0.5
      steps: 150000
    reward_signals:
      extrinsic:
        strength: 1.0
        gamma: 0.99
      curiosity:
        strength: 0.02
        gamma: 0.99
        encoding_size: 256
      gail:
        strength: 0.01
        gamma: 0.99
        encoding_size: 128
        demo_path: c:/users/teo/thesis/Humming/Demos/ExpertHumming.demo

CrawlerStatic:
    normalize: true
    num_epoch: 3
    time_horizon: 1000
    batch_size: 2024
    buffer_size: 20240
    max_steps: 1e7
    summary_freq: 30000
    num_layers: 3
    hidden_units: 512
    behavioral_cloning:
      demo_path: c:/users/teo/thesis/Crawler/Demos/ExpertCrawler.demo
      strength: 0.5
      steps: 50000
```

```
reward_signals:
    gail:
        strength: 1.0
        gamma: 0.99
        encoding_size: 128
        demo_path: c:/users/teo/thesis/Crawler/Demos/ExpertCrawler.demo
```